ABSTRACT

Title of dissertation: Hardware-Based Authentication for the Internet of Things

Md Tanvir Arafin
Doctor of Philosophy, 2018

Dissertation directed by: Prof. Gang Qu
Electrical and Computer Engineering Department

Entity authentication is one of the most fundamental problems in computer security. Implementation of any authentication protocol requires the solution of several sub-problems, such as the problems regarding secret sharing, key generation, key storage and key verification. With the advent of the Internet-of-Things(IoT), authentication becomes a pivotal concern in the security of IoT systems. Interconnected components of IoT devices normally contains sensors, actuators, relays, and processing and control equipment that are designed with the limited budget on power, cost and area. As a result, incorporating security protocols in such resource constrained IoT components can be rather challenging. To address this issue, in this dissertation, we design and develop hardware oriented lightweight protocols for the authentication of users, devices and data. These protocols utilize physical properties of memory components, computing units, and hardware clocks on the IoT device.

Recent works on device authentication using physically uncloneable functions can render the problem of entity authentication and verification based on the hard-

ware properties tractable. Our studies reveal that non-linear characteristics of resistive memories can be useful in solving several problems regarding authentication. Therefore, in this dissertation, first we explore the ideas of secret sharing using threshold circuits and non-volatile memory components. Inspired by the concepts of visual cryptography, we identify the promises of resistive memory based circuits in lightweight secret sharing and multi-user authentication. Furthermore, the additive and monotonic properties of non-volatile memory components can be useful in addressing the challenges of key storage. Overall, in the first part of this dissertation, we present our research on design of low-cost, non-crypto based user authentication schemes using physical properties of a resistive memory based system.

In the second part of the dissertation, we demonstrate that in computational units, the emerging voltage over-scaling (VOS)-based computing leaves a process variation dependent error signature in the approximate results. Current research works in VOS focus on reducing these errors to provide acceptable results from the computation point of view. Interestingly, with extreme VOS, these errors can also reveal significant information about the underlying physical system and random variations therein. As a result, these errors can be methodically profiled to extract information about the process variation in a computational unit. Therefore, in this dissertation, we also employ error profiling techniques along with the basic key-based authentication schemes to create lightweight device authentication protocols.

Finally, intrinsic properties of hardware clocks can provide novel ways of device fingerprinting and authentication. The clock signatures can be used for real-time authentication of electromagnetic signals where some temporal properties of the

signal are known. In the last part of this dissertation, we elaborate our studies on data authentication using hardware clocks. As an example, we propose a GPS signature authentication and spoofing detection technique using physical properties such as the frequency skew and drift of hardware clocks in GPS receivers.

Hardware-Based Authentication for the Internet of Things

by

Md Tanvir Arafin

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Advisory Committee:
Professor Gang Qu, Chair/Advisor
Dr. Dhananjay Anand
Professor Robert Newcomb
Professor Charalampos Papamanthou
Professor Yang Tao, Dean's Representative

Dedication

To my mother

Acknowledgments

I would like to express my profound gratitude to Professor Gang Qu for his support and guidance throughout my journey at the University of Mayland (UMD). I am sincerely thankful for his kindness and patience with me throughout my graduate studies. Over the last five years, he has kindly advised me in our research on hardware-based user authentication using common systems components. This work would not have materialized without his constant encouragement, motivation, and thoughtful guidance.

A significant portion of hardware-based GPS anti-spoofing work is performed with the collaboration of National Institute of Standards and Technology (NIST). I sincerely thank Dr. Dhananjay Anand of NIST for educating me with GNSS systems and setting up our experiments at the NIST facility. He was also very kind in agreeing to serve as a committee member for this dissertation.

I am grateful to Professor Robert Newcomb, Professor Charalampos Papamanthou and Professor Yang Tao for serving on my dissertation committee. I was fortunate to receive their constructive feedback and support in preparation and completion of this work.

My internships were an invaluable experience for me. I thank Dr. Shalabh Jain and Dr. Jorge Guajardo for their kind advice and supervision at Bosch. I learned to appreciate the beauty of simple yet effective lightweight cryptography through this internship. I would also like to acknowledge Philips Internship Program and Ms. Praveen Sharma for her kind support during the internship. The work at Philips

gave me an invaluable lesson on how physical devices and hardware are used for authentication in an industrial setting. I sincerely acknowledge Dr. Haoting Shen (UFL) for fabricating the memristors for our work and Professor Mark Tehranipoor (UFL) for his support of the experiments regarding memristive hardware.

I was privileged to spend the last six years with some very talented colleagues especially Mingze Gao, Qian Wang, Khai Lai, Xi Chen and Omid Aramoon. I also want to thank Ms. Melanie Prange, Ms. Vivian Lu and Ms. Maria Hoo for their help and support regarding administrative processes of the ECE department.

I will be grateful to Prof. Saiful Islam of BUET for his encouragement in my pursuit of graduate studies. Thanks to my friends especially Sayan Bhai, Esha Apu, Saad, Auyon Bhai, Amit Bhai, Tasbir Apu, Sara Apu, Mehbub, and Himu for helping me survive in the US.

Finally, I want to thank my family for their unconditional love and support.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CAN	Controller area Network
CLA	Carry Look-ahead adders
CMOS	Complementary metaloxidesemiconductor
DoS	Denial of service
DRAM	Dynamic random-access memory
GPS	Global positioning system
HB	Hopper and Blum authentication
HCA	Han-Carlson adder
IoT	Internet-of-things
LPN	Learning parity in the presence of noise
MIM	Man-in-the-middle
PUF	Physically uncloneable function
RAM	Random-access memory
RCA	Ripple Carry adders
RNG	Random number generator
RRAM	Resistive random access memory
PVT	Process, supply voltage and temperature
VC	Visual Cryptography
VLSI	Very large scale integration
VOS	Voltage over-scaling

Chapter 1: Introduction

Hardware security is an emerging field that studies the application of security primitives in hardware and their vulnerabilities. As new technologies emerge, the nature of security primitives and vulnerabilities change, hence, studying security features of such technologies have its unique benefits. Moreover, new technologies can provide exclusive hardware features useful for security purpose. Instead of making hardware the weakest link in security (for example, side channel attacks), one can employ hardware's intrinsic properties to enhance security. Therefore, in this work, we have proposed several lightweight hardware intrinsic security solutions which try to solve problems with authentication of entities.

Ubiquitous deployment of sensors, actuators, data acquisition systems, processing modules, cloud servers, and electronic components interconnected by wired and wireless communication technology is leading to the era of the Internet of Things(IoT). Implementation of Internet-of-Things is critically dependent on a well-designed interconnected network between the Things. As the number of Things grows, constraints on the critical factors such as cost, area, and power, *etc.* become tighter, and in most cases forces the designer to either sacrifice some design aspects or innovate new design to meet all the requirements. Unfortunately, one of the stan-

dard sacrifices is made on the security of the low power components in the network such as the sensors, data acquisition units or on-field data processing units[8].

Compromising regarding security for low power modules can pose severe threats to the complete system and its outputs. Therefore, the design should opt for Internet-of-Trusted-Things, where the system components are trusted and provides accurate data to the system. Authentication creates the first layer of trust between two entities-the provers and the verifier. Well-designed secure authentication protocols require sufficient processing powers and hardware available to both prover and the verifier for calculating cryptographic primitives needed for authentication. This classic strategy can become unfeasible for low power modules where the claimants can constitute low-power sensor nodes that do not have enough processing capability to realize essential cryptographic functions.

To address these issues we have worked on hardware-dependent low power lightweight authentication techniques. We are focusing on several hardware primitives: resistive memory based circuits and systems, voltage over-scaled systems and hardware clocks.

In recent years, metal-oxide-based resistive switching memories (RRAMs or memristors) have drawn significant research attention due to their potential application in next-generation low-power nanoelectronic non-volatile memory systems [9]. Simple device geometry, high density, and built-in non-volatile features have made memristors appealing as a replacement for flash memory and DRAM. Moreover, studies on on-chip CMOS compatibility of memristors have shown promising results indicating the capability of replacing on-chip SRAMs [10]. Logic processing and

possible neuromorphic computation in functional resistive switching memories may one day replace CMOS with memristors [11]. With promises regarding the energy efficiency of memristors, it is a question of when not whether memristors will be used routinely in IoT system design. Hardware security research works are mostly built on the available traditional circuit building blocks and components, and therefore, the addition of memristor-based circuits will enrich the toolset for designing hardware security primitives. Hence, we explore exciting properties of memristor-based circuits and systems for developing lightweight authentication protocols.

The second hardware design primitive we are interested is hardware designs based on voltage over-scaling. The design of digital circuits and systems are focused on the deterministic results, *i.e.,* for identical inputs, application of any given design will yield identical output. As we are pushing the boundaries of power efficiency, we are introducing the effects of analog nature of the circuit components involved in the computation. One of the standard power reduction techniques is voltage over-scaling (VOS). In VOS, the digital circuit used for calculation is operated under the nominal voltage, which guarantees correct output for all input condition under the operating environment. Since the dynamic power consumed in a VLSI chip is squarely, and static power is proportional to the operating voltage, reducing the operating voltage under the prescribed margin can result in considerable power savings. However, the effect of this voltage over-scaling will be translated into errors generated during a computation. These errors are a function of supply voltage and physical and environmental variations. Research works in VOS problems focus on reducing these errors to provide acceptable results from a computation. Inter-

estingly, with extreme VOS, this error can reveal significant information about the underlying physical system. Therefore, VOS not only have great promise in low power computing but also can be used for realizing physically unclonable functions without adding hardware overhead. This will, in turn, enable lightweight hardware security primitives in low power devices.

Finally, we revisit the applicability of hardware clocks for authentication and fingerprinting. We find that time-varying intrinsic properties of free-running crystal oscillators can be useful in designing signal authentication and spoofing detection mechanisms. We study the employability of hardware oscillators for authenticating commodity signals used in embedded systems. One such example can be Global positioning system (GPS) signals. GPS has become an essential tool for precise time measurement, positioning, navigation and synchronization over distributed systems, and is an integrated part of IoT architecture. Civilian GPS signals are not encrypted and can easily be replicated to launch spoofing attacks. Demonstration of such GPS spoofing attacks is available in current literature. Although there exist different methods for signal level and data-level spoofing detection for GPS signals, each has their limitations, can be spoofed by a determined adversary and in most case, their implementation is prohibited by the cost of additional circuits and radio-frequency components. To address this problem, we develop a simple, fast, and tamper-proof hardware dependent signal authentication and spoofing detection mechanism for civilian navigation signals.

The following chapters describe our works on designing hardware oriented lightweight authentication protocols for users, devices, and signals using the physi-

cal properties of memory components, computing units, and hardware clocks. We start with a review of memristors used in authentication in chapter 2 and present our single and multi-user authentication protocols in chapter 3. Chapter 4 introduces the concepts of voltage overscaling and discusses how voltage overscaling can be used in hardware-based security in IoT applications. In chapter 5, we demonstrate the promises of hardware clocks in authenticating GPS signals. Finally, we conclude in chapter 6, with future work direction in improving memristor-based security protocols, modeling for VOS systems and designing physical layer key distribution using visual cryptography. In sum, the contribution of this work are the following:

- First, we have presented a comprehensive study of existing memristor based hardware authentication techniques and their weaknesses for extending the state-of-the-art for memory component dependent authentication techniques.

- Then, we have developed two single user authentication technique and one multi-user authentication technique using memristive hardware component. We have discussed the potential attacks on these protocols and provided corresponding countermeasures for understanding the merits of the proposed schemes regarding security. We also performed HSPICE simulation using PTM's 65nm MOSFET model and Stanford memristor model to validate the reliability of these authentication protocols.

- Next, we have demonstrated that voltage over-scaling based computing leaves a process variation dependent device signature in the approximate results. Binding this process variation dependent device signature with basic key based

authentication schemes; we have propose a simple authentication mechanism using voltage overscaling and analyzed the reliability of this scheme using FreePDK's 45nm MOSFET model.

- Finally, we have presented a data-level authentication mechanism for GNSS signals that rely on intrinsic hardware properties of a free-running crystal oscillator. We have developed a Kalman -filter based signal authentiaction and spoofing detection unit to authenticate GPS signals in real time. We have reported the applicability and reliability of this proposed spoofing detection technique for applications targeting the Internet of Things.

Chapter 2: Memristors and Authentication: A Review

2.1 Introduction

High-density memristive crossbar arrays are promising candidates for next-generation memory solutions with in-memory computing capabilities[9]. Simple device geometry, high density, and built-in non-volatile features have made memristors a replacement for flash memory and DRAM. However, commercial products based on memristors are still quite elusive. Therefore, for developing competitive products further research on the fabrication process, yield, novel computer architecture, and circuit design are required. Furthermore, applications outside memory operation such as security and cryptographic applications need to be explored to harness the unique characteristics of memristive devices. For example, time-dependent switching properties of metal-oxide-based resistive memories have demonstrated applications in Physically Unclonable Function (PUF) designs[12],[13]. These works have attracted new ideas and memristor-based research in the area of hardware-dependent security and trust. In the advent of memristors for memory design and in-memory computation, works exploring memristor-based security primitives such as physically unclonable functions[14],[12], random number generators[15] and chaotic circuits for secure communication [16] *etc.* are also gaining research attention. This chapter

chronicles these efforts.

Conventional authentication methods using low-entropy user-passwords are weak and severely vulnerable to dictionary attacks. Although password-authenticated key exchange protocols can render secure authentication against active and passive attackers, they are susceptible to dictionary attacks at the authentication server. Secret sharing schemes can be useful in creating and distribution authentication secrets over multiple servers for resisting dictionary attacks at the server-side. However, general secret sharing mechanisms such as the Shamir's secret sharing algorithm and its derivatives are hardware independent and require significant power and computation capabilities. Therefore, with the advent of distributed Internet-of-Things (IoT) systems, authentication would become one of the standard security challenges for low-power IoT nodes. In chapter 2 and 3, we have explored how the intrinsic physical properties of memristors can be useful in tackling this problem.

2.2 Fundamentals on Authentication

Identity and entity authentication depends on the sharing a secret between two parties- the verifier and the prover. In this section, we introduce common secret sharing and user-authentication mechanisms.

2.2.1 Single User Authentication

For authenticating a single user, the verifier authenticates the singular prover based on a secret that can be derived from (1) something that is known by both parties

(such as passwords), or (2) something possessed by the prover (such as hardware keys), or (3) something inherent (such as signatures, biometric signals etc. of the prover) [1]. According to Menezes *et al.* common properties of authentication protocols includes "*reciprocity of identification, computational efficiency, communication efficiency, nature of security guarantee and the nature of security storage*" [1].

Passwords are the most commonly used user authentication mechanism where the authenticator stores the (username, password (or its hashed value)) pair for different prover and use this pair to identify a prover. To strengthen this protocol several steps can be taken such as (i) different password rules can be introduced, (ii) password salting can be performed, or (iii) password mapping can be slowed down which will make it difficult for an attacker to test a large number of trial passes [1]. Frequent attacks on password schemes are replay attack and exhaustive and dictionary password search. Furthermore, leaking of an authenticator's database containing (username, password) can cause significant threat[17].

For all of these weaknesses of password schemes, challenge-response identification becomes a step toward strong authentication where the authentic parties share a sequence of secret one time passwords or challenge response pairs which are usually derived from some one-way functions or a challenge-response table. Furthermore, a zero-knowledge protocol that verifies an entity through its possession of the knowledge of the secret instead the exact secret is also a strong authentication scheme [1].

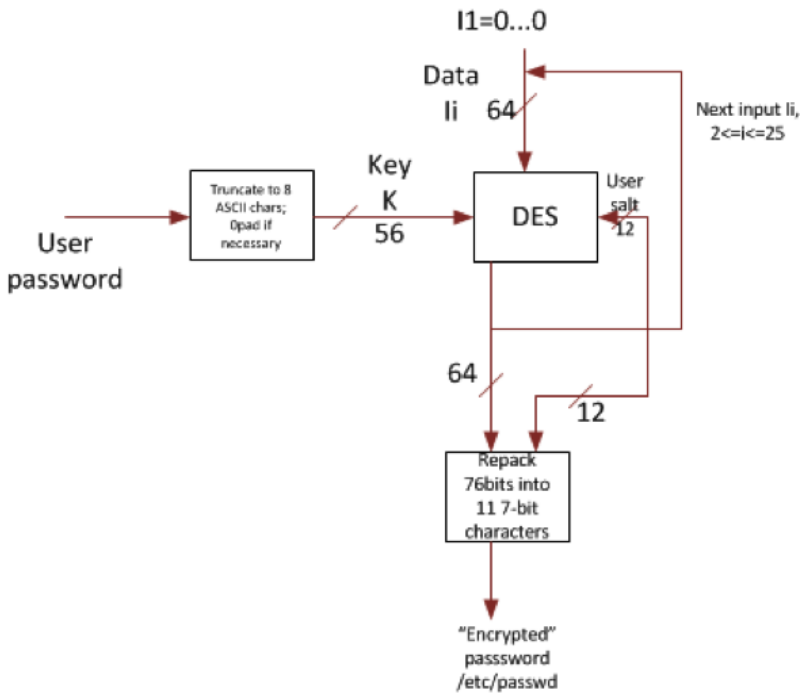

Figure 2.1: An instance of Unix password mapping in practice [1]. User password is truncated to 8 ASCII characters and used as a 56-bit key (k) to encrypt a a known plaintext (64-zero bit seed I_0) using Data Encryption Standard (DES) algorithm. The output of this encryption is fed back into the algorithm again as (I_1) and the process continues for 25 rounds. The final output (*i.e.*, I_{25}) is repacked with a user given 12 bit-salt and stored as an encrypted password at */etc/passwd*.

2.2.2 Secret Sharing and Multiple User Authentication

Secret sharing is a well-studied problem which seeks to distribute pieces of information (or called the secret) to multiple parties in a way such that the information (the secret) can be revealed when all or a sufficiently large subset of the individuals contribute their shares. Shamir's secret sharing algorithm [18] defines the concept of threshold scheme for secret sharing. This (k, n) threshold scheme (where $k \leq n$) can be described as follows:

Definition 2.1 *: Assume that a secret S memristor to be shared by n parties. The secret is divided into n pieces such that, the knowledge of k or more pieces would be sufficient to reconstruct S. However, if the knowledge of any $k-1$ pieces or less is available, it would be impossible to restore S.*

A direct application of this problem is the multi-user authentication problem, where at least k authentic users must present their shares to gain access to a system. One example of this is the two-person rule originally designed to prevent the accidental or malicious launch of nuclear weapons by a single individual. One may argue that authenticating each user and counting the authenticated users can trivially solve the multi-user authentication problem. This approach does address the issue but has two significant drawbacks in scalability and user privacy. First, the expensive authentication protocol has to be applied at least k times, and some mechanism to check duplicate users must be implemented. Second, unlike the traditional secret share dependent approaches [18, 19, 20, 21], this method will reveal

the identity of each authenticated user, creating user privacy concerns.

Shamir [18] and Blakley [19] independently proposed solutions for the aforementioned problem of (k, n)-threshold scheme. More specifically, Blakley used techniques from finite geometry to provide a solution for safeguarding and sharing cryptographic keys. Shamir's solution is designed on the polynomial interpolation over a finite field. It has since become a widely acceptable solution for secure secret sharing and distribution of cryptographic keys. A detailed explanation of the scheme and the impact of subsequent works can be found in [20]. One of the drawbacks of these early solutions is their high computational cost. For example, Shamir's secret sharing algorithm requires calculations over the finite field during both secret share generation phase and secret reconstruction phase. This, in turn, requires complex digital circuitry for hardware implementation, which is known to be more efficient than the software implementation.

A simple hardware-oriented secret sharing model was first proposed by Naor and Shamir [21] as the Visual Cryptography(VC) scheme. It is an alternative lightweight system for secret sharing to Shamir's original scheme. In visual cryptography, the secret shared among multiple parties is merely an image. The secret image is broken into n pieces, and each piece is printed on a transparency. When k or more pieces of these transparencies are placed on a stack, the secret image is revealed and can be comprehended by human eyes. In this scheme, the secret share generation memristor only straightforward calculations, and the revelation of the secret image does not involve any mathematical computation. In essence, this is an example of how inherent physical properties of hardware can be used in design-

ing lightweight security primitives to avoid complex mathematical computation and formal cryptography protocols. Naor and Pinkas first discussed the application of visual cryptography for authentication and identification [22].

Secret sharing has also seen its application in securely sharing and destroying encrypted data. For example, Vanish computer project at the University of Washington uses a random key to encrypt data and then distributes them across several nodes in a P2P network as a shared secret [23]. To decrypt the message, at least k-out-of-n nodes on the system must be accessible. The fundamental idea of data destruction over time was that *"the number of secret-sharing nodes on the network decreases over time and thus once the total number of interested parties becomes less than a given threshold (i.e., k) the secret key eventually becomes non-reconstructible, and thus the encrypted data vanishes"* [23].

2.2.3 Hardware-based Authentication

The first problem for authentication is key-generation. Assume a verifier Alice tries to authenticate/identify a prover Bob. For solving this authentication problem, a signature of Bob is required to be stored by the Alice to verify Bob. This procedure poses several security challenges. First, the human-generated keys such as passwords are rarely random and are vulnerable to dictionary attacks. One can use machine-generated private keys. However, machine-generated keys are difficult to remember, and storing such keys in non-volatile memory can easily leak this secret key to an adversary.

Physically Unclonable Functions [24, 25, 26] can provide some solutions to key generation and storage problem. Silicon PUFs are on-chip circuitry that can extract fabrication variations to generate chip-dependent PUF data that can be used as secret keys or as seeds of random number generators. For using PUFs in device authentication, one can hash a password with the device signature to increase the entropy of the private keys and generate a unique device dependent key. Moreover, PUFs does not store the key; rather the key exists when the device is powered on, and therefore, to extract(or leak) a key an adversary memristor to attack the system while it is up and running, which is significantly harder[27]. Therefore, new design technologies such as resistive memory-based random number generators and PUFs can be useful in solving primary key generation and storage challenges.

In the following sections, we will present how challenge-response pairs(CRP) can be generated from hardware, especially memristive hardware. There are several ways these CRPs are used for device authentication purposes. Physical randomness in these PUF designs are considered inherently independent and identically distributed. Therefore, learning attacks that focus on modeling the internal operation of the PUF fails. The key generation and authentication model for general PUFs are given below.

Algorithm 1 PUF based Key Generation

1: **procedure** $(\mathbf{C}, \mathbf{R}) \leftarrow KeyGen(1^{\lambda})$

2: Assume a PUF can provide ℓ-pairs of challenge-response.

3: Read the PUF and record n-bit responses $\mathbf{R}^{\ell \times \mathbf{n}}$ for the p-bit challenges $\mathbf{C}^{\ell \times \mathbf{p}}$ to the PUF.

4: **return** $(\mathbf{C}, \mathbf{R})$

Enrollment: The verifier uses the key generation algorithm to generate the secret $(\mathbf{C}, \mathbf{R})$ from a PUF P. The verifier keeps $(\mathbf{C}, \mathbf{R})$ and the prover owns the PUF.

Table 2.1: Single round interactive authentication for protocol using a PUF with high entropy content

Prover(P)		Verifier($\mathbf{C}, \mathbf{R}$)
		Select a random challenge $\mathbf{c} \xleftarrow{\$} \mathbf{C}$
	$\xleftarrow{\mathbf{c}}$	
Apply $\mathbf{c}$ to the PUF P and read the response $\mathbf{r}' = PUF(\mathbf{c})$		
	$\xrightarrow{\mathbf{r}'}$	
		If $\mathbf{r}' = \mathbf{r}$ (where $\mathbf{r}$ is the recorded response for $\mathbf{c}$), accept, else reject.

One of the key weakness of such protocol is that the changes in operating conditions and environmental factor can change the physical responses of a PUF system. Therefore, such authentication has high completeness for the systems that have stable PUF bits over all the operating conditions. For example, SHIC PUFs keeps the PUF bits as a non-volatile memory content that has an excellent memory retention capability. However, these PUFs are vulnerable to read-out attack and therefore, slowing down of reading access is required for a security guarantee.

Moreover, common PUF designs require extensive hardware and data process-

ing peripherals to harness the entropy from the physical variation. For example, Figure 2.2 shows a Vivado block diagram of a complete PUF implementation in Xilinx FPGA.

In the first part of this dissertation, we will discuss the use of memristive hardware in design PUFs and other security primitives for authentication. Hence, in the next section, we introduce the physics behind the modeling of memristors for simulation and experimentation.

2.3 Basics of Memristors

According to the classical definition by Leon Chua, instantaneous resistance of a charge-controlled memristor can be written as [28]:

$$M(q) = \frac{d\phi(q)}{dq} \equiv \frac{v(t)}{i(t)} \tag{2.1}$$

where, $q = \int_{-\infty}^{t} i(\tau)d\tau$ is the electric charge, $\phi(q)$ is the magnetic flux, $v(t)$ is the voltage across the device, and $i(t)$ is the electric current flowing in the device at time t [28]. Therefore, from equation 2.1, we can see that the resistance (also known as memristance $M(q)$) of an ideal memristor is dependent on the current that has previously passed through the device.

Chua and Kang generalized the concept of memristors and memristive system in [29]. It was theoretically argued that the dynamic properties of a current controlled memristive system can be expressed with the help of an internal state variable w as:

$$\frac{dw}{dt} = f(w, i) \tag{2.2}$$

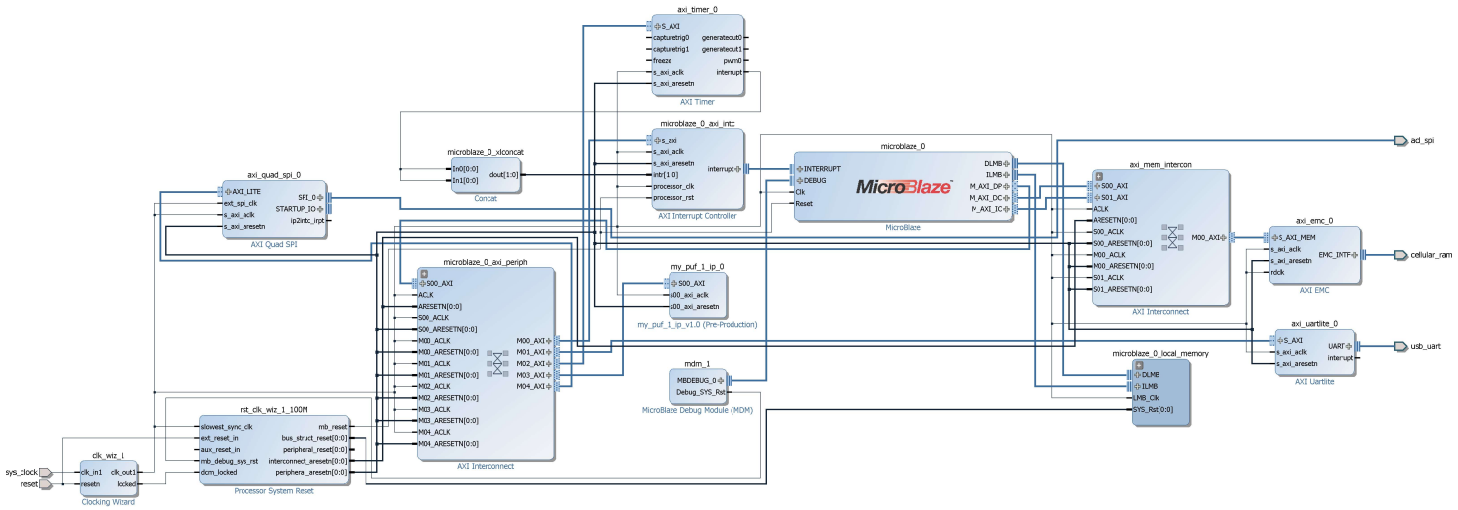

Figure 2.2: A complete PUF controller block diagram in Vivado. A Microblaze microprocessor along with AXI peripherals and memory controller is used for extracting entropy from a PUF IP noted by my_puf_1_ip_v1.0

$$v(t) = M(w, i) \times i(t) \tag{2.3}$$

where, $M(w, i)$ is the generalized resistance of the memristive system and $f(w, i)$ is a function that captures the boundary behavior and various nonlinear dynamical effects. From these equations, it can be seen that for zero input current there will be a zero output voltage, irrespective of the state variable w. Hence, this dynamic system has a zero-crossing Lissajous figure-like input-output relationship [29]. This input-output characteristic in the $v - i$ plane is also known as pinched hysteresis loop of memristors, and it is considered to be a signature property of this circuit element [30, 31]. As the frequency of the signal along the memristor increases, this zero-crossing pinched hysteresis loop shrinks in size, and it becomes a straight line when the frequency approaches infinity [31].

This fundamental circuit theoretic model was first realized in a MIM-based device structure at HP Labs [2]. The memristor discovered by HP Labs consists of a thin film (5 nm) with of insulating TiO_2 sandwiched between platinum contacts in the simple metal-insulator-metal (MIM) structure [2]. Several transport models have been proposed to explain the electronic properties of this memristor and devices of similar construction. These models primarily attempt to relate the carrier transport mechanism in the thin films with the system of equations 2.1-2.3. Each model defines the generalized resistance/memristance $M(w, i)$ and the function $f(w, i)$ which encapsulates the memristive nature of these devices. We discuss some of these memristor models below. Before going into details, we introduce common circuit symbols used for memristors in Figure 2.3.

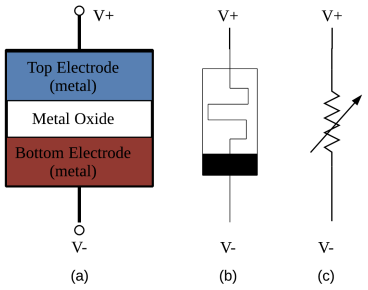

Figure 2.3: (a) Simplified view of MIM structure for memristors; 2.3(b) and 2.3(c) shows the common circuit symbols used for memristors in circuit design.

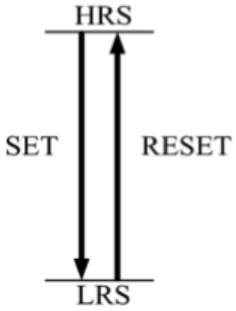

Figure 2.4: SET-RESET operation in a memristor.

2.3.1 Linear and Non-Linear Ion-Drift Models

The simplest model that relates the resistivity of the HP-devices with memristors is the linear ion-drift model proposed by Strukov *et al* [2]. In this model, the governing carrier transport in the thin film of the memristor is described by linear drift of oxygen vacancies. Assume $w(t)$ as the thickness of the doped region in the thin-film which is created by the linear drift of charged oxygen vacancies (dopants) at a given applied bias. Then, considering a linear ion-drift model, one can write

the state equation for the state variable $w(t)$ as:

$$\frac{dw}{dt} = f(w, i) = \frac{\mu_v R_{on} i(t)}{D} \tag{2.4}$$

where D is the film thickness of the memristor, μ_v is the average ion mobility of oxygen vacancies in TiO_2, $(D - w)$ is the size of the undoped region. R_{on} is the resistance of the memristor when it is completely doped (*i.e.*, $w = D$), and R_{off} is the resistance when it is completely undoped (*i.e.*, $w = 0$) as shown in Figure 2.5. The current-voltage relationship for a memristor in this model is defined as [2]:

$$v(t) = \left(\frac{w(t)}{D} R_{on} + (1 - \frac{w(t)}{D}) R_{off}\right) i(t) \tag{2.5}$$

Overall, the effective memristance of this structure can be expressed as:

$$M(w) = \frac{w(t)}{D} R_{on} + (1 - \frac{w(t)}{D}) R_{off} \tag{2.6}$$

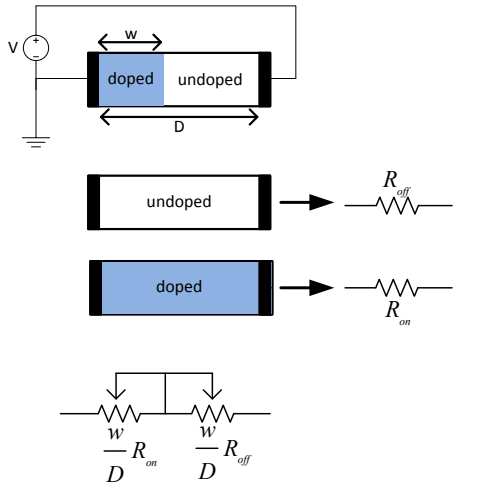

Figure 2.5: Strukov model of a $Pt - TiO_{2-x} - Pt$ device [2].

The linear ion-drift model provides a simple explanation of the transport mechanism in a memristor. It does not consider boundary effects and non-linear dopant

kinetics. To implement boundary behaviors, equation 2.4 can be modified as:

$$\frac{dw}{dt} = \frac{\mu_v R_{on} i(t)}{D} g(w, i) \tag{2.7}$$

where $g(w, i)$ is a window function that captures the physics near the device boundary. Several approximations for the window function can be found in the literature [32, 3, 33, 34]. These window functions are listed in Table 2.2.

Table 2.2: Common window functions to capture physics near device boundary in memristors

Author	Window Function
Joglekar *et al.* [32]	$g(w,i) = 1 - \left(\frac{2w}{D} - 1\right)^{2p}$
Biolek *et al.* [3]	$g(w,i) = 1 - \left(\frac{w}{D} - u(-i)\right)^{2p}$
Prodromakis *et al.* [33]	$g(w,i) = j\left(1 - \left[\left(\frac{w}{D} - 0.5\right)^2 + 0.75\right]^p\right)$

For these window functions, p is a positive integer that controls the rate of change of w near the device boundary, $u(i)$ is the step function, and j controls the maximum value of $g(w)$. Although the models described above capture fundamental device properties, they are insufficient to explain the underlying higher order non-linearity of actual devices.

To simulate these models for VLSI designs, simple SPICE representations are presented in [3, 34, 35] *etc.* The SPICE model proposed by Biolek *et al.* in [3] and its derivatives are widely used in the literature for simulating HP memristors. In this circuit, the window function is incorporated using user-defined function $f()$ and the

memory effects are incorporated using a feedback controlled integrator. The circuit diagram for the model is given in Figure 2.6.

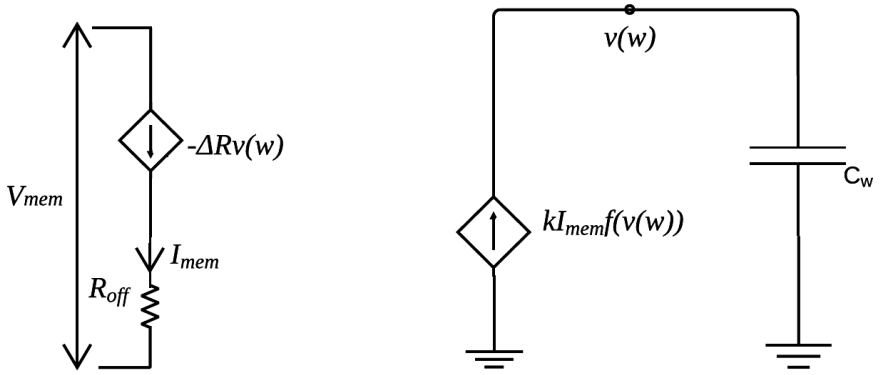

Figure 2.6: SPICE model of a memristor proposed in [3]. Here V_{mem} and I_{mem} are the voltage and current across the memristor, $k = \frac{\mu_v R_{on}}{D^2}$, $\Delta R = R_{on} - R_{off}$, C_w represents the doped layer of width w and $v(w)$ is the voltage across the layer.

All of the discussed linear and non-linear drift models assume that electron transport in memristors is due to the drift of carriers under electric field in the doped and undoped regions. However, quantum mechanical effects need to be considered for accurately describing carrier transport in these nano-devices. Pickett *et al.* first incorporated the quantum mechanical effects in the underlying memristor model to give a detailed description of the complex carrier dynamics in memristors [36] as discussed next.

2.3.2 Quantum Mechanical Models

Pickett *et al.* explain the observed complex carrier dynamics in TiO_2 based memristors using Simmons tunneling theory [37] and drift mechanisms of the carriers. A tunnel barrier is considered between the conducting channel and the platinum electrode of the device. An ohmic resistor in series with this tunnel barrier is used to explain the device characteristics. The state variable in this model is represented by the width of the tunnel barrier. This model is called the Pickett's model or Simmons tunnel barrier model. The state equation for Pickett's model is written as [36]:

$$\frac{dw}{dt} = \begin{cases} f_{off} sinh\left(\frac{|i|}{i_{off}}\right) exp\left[-exp\left(\frac{w-a_{off}}{w_c} - \frac{|i|}{b}\right) - \frac{w}{w_c}\right], & i>0 \\ -f_{on} sinh\left(\frac{|i|}{i_{on}}\right) exp\left[-exp\left(-\frac{w-a_{on}}{w_c} - \frac{|i|}{b}\right) - \frac{w}{w_c}\right], & i<0 \end{cases} \tag{2.8}$$

where, f_{off}, f_{on}, i_{off}, i_{on}, a_{off}, a_{on}, w_c, and b are fitting parameters. The current i through the device is given by [4]:

$$i = \frac{qA}{2\pi h(\Delta w)^2}\left\{\phi_I e^{-B\sqrt{\phi_I}} - (\phi_I + q|v_g|)e^{-B\sqrt{\phi_I + q|v_g|}}\right\} \tag{2.9}$$

Here, q is the elementary electronic charge, A is the average channel area, ϕ_I is the modified barrier height, h is Planck's constant, m is the average effective mass of the carrier and

$$B = 4\pi\Delta w\frac{\sqrt{2m}}{h} \tag{2.10}$$

If we consider R_s as the series resistance of the channel and v_g is the voltage across the tunnel barrier, then the voltage across the device v can be written as:

$$v = v_g + v_R = v_g + iR_s \tag{2.11}$$

To incorporate this model in circuit simulation, Abdalla *et al.* provided a SPICE model based on equation 2.8-2.11 in [4] as given in Figure 2.7. This circuit is derived from equations 2.8-2.11 and uses experimental values to define its parameters. Although this model tries to accurately represent the device physics, simulating this model overestimates current by around 20% and causes the simulated memristor to switch faster than the experimental memristor [4].

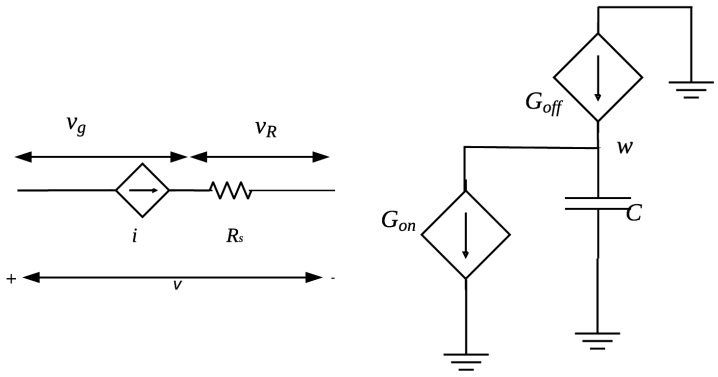

Figure 2.7: (a) SPICE model of a memristor proposed by Abdalla *et al* [4]. Here, the current and voltages are given by equations 2.9-2.11. Figure 2.7(b) represents the state-space model of the device represented by Figure 2.7(a) where C is the width of the tunnel barrier, w is the voltage across the barrier, and $\frac{dw}{dt} = \frac{1}{C}(G_{off} - G_{on})$ where G_{off} and G_{on} are the right hand side of equation 2.8 for $i > 0$ and $i < 0$ respectively.

Pickett's model provides a good insight of the carrier dynamics and considers a near accurate physical model. However, this model is computationally expen-

sive. For device simulation, Kvatinsky *et al.* proposed a simplified version known as the *threshold adaptive memristor (TEAM) model* to reduce the computational complexity of Pickett's model [38]. There exist similar models such as the Boundary Condition memristor (BCM) model presented in [39] which are based on the simplified versions of equations 2.8-2.11. Overall, Pickett's model and the subsequently simplified models of memristors provides a good starting point for understanding fundamental carrier dynamics in a physical memristor.

The equations 2.4-2.11 are specifically derived for the TiO_2-based memristive devices. For other metal-oxide thin film based memristors, the characteristic equation of the device follows a similar pattern.

In this dissertation, we have demonstrated our experiments on the HfO_x based memristors. These devices have an individual design and marked differences from the TiO_2 based memristors developed by HP Labs. HfO_x based memristive devices have large ON/OFF ratio, multi-bit storage capacity and excellent switching endurance ($\geq 10^6$ cycles) [9], and therefore, they are promising candidates for memristive designs. Although our experiments are devised and performed on this device platform, our theoretical constructs for secret-sharing and authentication is by no means restricted only to HfO_x based memristors.

The conductive filament formation in the HfO_x thin-film essentially defines the internal state variable g for a HfO_x based memristor. The current in the memristor is also determined by the electron tunneling in metal-insulator boundary. Thus, the analytical model of the thin-film evolution in a HfO_x-based memristor

can be described using the following equation as [40]:

$$\frac{dg}{dt} = \nu_0 e^{-Ea,m/kT} sinh(\frac{qa\gamma V}{LkT}) \tag{2.12}$$

where g is the state variable for the device which represents the spatial distance between the conductive filament in the oxide and the metal boundary, q is the electron charge, L is the device filament thickness, V is the applied voltage, T is the device temperature, Ea, m, γ, ν_0 are device dependent physical parameters. The current-voltage relationship in the device is given by the following equation:

$$I(g, V) = I_0 e^{-g/g_0} sinh(\frac{V}{V_0}) \tag{2.13}$$

where I_0, V_0, g_0 are device dependent physical parameters.

Memristors are usually fabricated in a crossbar fashion. In a crossbar memory array, a memristor unit can contain a single resistor (1M configuration) or a transistor and a memristor (1T1M configuration). In this work, we consider transistor controlled operations because such designs can provide better control, have immunity to sneak path currents, and can deliver reliable performance [9]. Figure 2.8 shows basic 1T1M configuration.

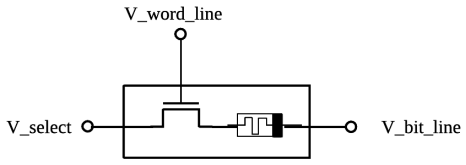

Figure 2.8: One-transistor-one-memristor (1T1M) configuration of an memristor cell. A MOSFET is used for controlling the current flowing through the memristor.

2.4 Memristors in Authentication

Memristors have some unique properties useful for security applications such as non-volatility, bias-dependent write-time, fabrication variations in a filament, non-linearity in the current-voltage relationship [41].

2.4.1 Standard Features of Memristors

We list intrinsic hardware properties of memristors that are used to design the proposed secret sharing based user authentication protocols.

F1 **Non-volatility**: Memristors are nonvolatile in nature. For binary operations, two resistance values are associated with each memristor: resistance due to lowest resistive state (LRS) or resistance due to highest resistive state (HRS). However, in analog operation, one finds that there is a continuous spectrum of resistance in between and beyond LRS and HRS that can be achieved. However, for reliable operation, the LRS and HRS of a given device are usually kept near constant and current limiting approach is used for safeguarding against resistance runaway. LRS and HRS are usually different by several orders of magnitude regarding their resistances.

Moreover, the initial states (*i.e.,* the resistive state of an unutilized device) of a given distribution of memristor population can be random. This random distribution along with the non-volatile nature of the memory can be used as a seed for generating secure random keys.

F2 **Bias dependent write-time**: Write-time of a memristive device can be defined as the time required for switching a device from one resistive state to another (*i.e.,* from LRS to HRS or from HRS to LRS). For example, switching time of the SET/RESET operation of HfO_x devices have been found to be exponentially dependent on the amplitude of the RESET voltage [40] as shown in the Fig 2.9. Thus, by adjusting the bias voltage, one can manipulate the write-time required for a given state transition. The memristor mentioned above based PUF approaches have utilized this feature[12].

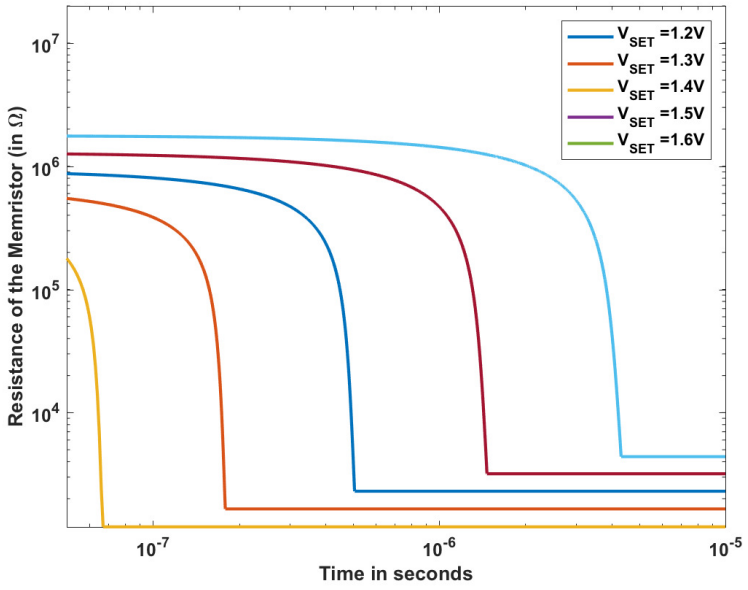

Figure 2.9: Variation of memristor SET-time as a function of bias voltage. We have used 1R configuration with $I_0 = 10\mu A$, $L = 12nm$ and the parameters listed at Table 3.4. Note that, with a linear increase in the SET voltage, there is an exponential decrease in the total time required for the state-transition.

F3 **Pulse-Controlled State Transition**: Further control and reliability on the

filament growth can be achieved by employing pulsed bias voltages, where the duration of the pulse (t_p) is a fraction of the complete write-time (t_{wr}), *i.e.*,

$$t_{wr} = nt_p \tag{2.14}$$

F4 **Fabrication variation in filament thickness**: Physical properties of a memristor vary significantly with the manufacturing variation in the filament thickness. For example, equation 2.4 shows that the change in resistive state is proportional to the $sinh()$ function of the inverse of the film thickness L. Therefore, a small change in film thickness can lead to a significant measurable change in the resistance of the device. Such random variability resulting from fabrication variations can be exploited to design security features such as device authentication. As shown in Figure 2.10, the write-time of HRS to LRS transition is longer for memristor with large filament thickness.

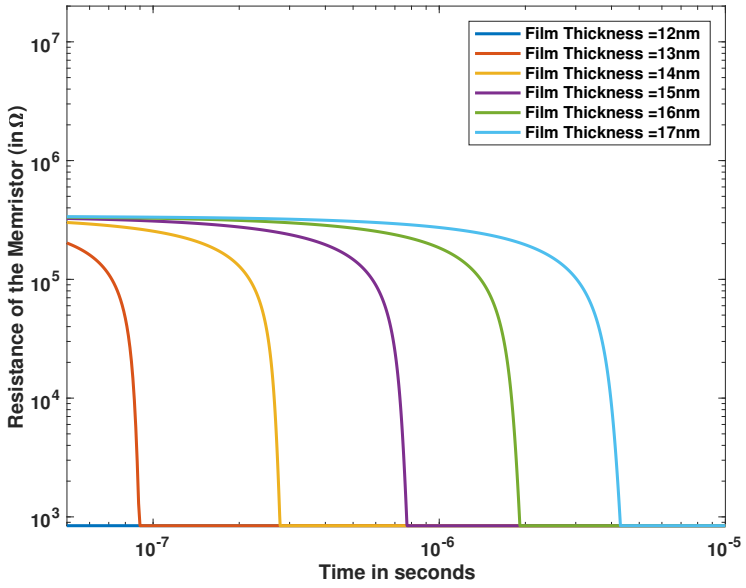

Figure 2.10: Variation of memristor SET-time as a function of filament thickness. We have used 1R configuration with $I_0 = 10\mu A$, $V_{SET} = 1.7V$ and the parameters listed at Table 3.4. Note that, with a linear increase in the filament thickness, there is an exponential decrease in the total time required for the state-transition.

F5 **Non-linearity**: From the basic equations presented in the previous sections, it can be seen that the resistive-switching devices are highly nonlinear. Device nonlinearity is a highly sought after property for secure hardware design, and therefore, harnessing this property will provide a novel implementation of standard hardware security protocols such as PUFs. This feature is confirmed in our experiments and can be seen from the figures presented in this section.

2.4.2 Existing Designs for Memristor-based Physically Uncloneable Functions

Physically Unclonable Functions are hardware-dependent security primitives that harness physical variation of a silicon chip to generate unique chip-dependent challenge-response pairs (CRPs). PUF CRPs can be used for secret key generation, seeding of random number generators, and in CRP-based authentication and attestation. Physical dependence of PUF primitives has opened up new avenues for implementing hardware intrinsic security and trust.

Physical variation in the thin film of memristors has a pronounced effect on their device characteristics such as the random variation in write-time for different devices, random distribution of measured resistance after the device forming step during fabrication, and a random resistive path between higher and lower resistive states. These fundamental properties provide an ideal situation to build memristor-based PUFs.

There have been several potential memristor-based PUF-designs in current literature with different use case and attack models. Since PUFs are inherently related to hardware-based authentication, we will assume a simple example scenario where an entity Alice wants to authenticate another entity, Bob. Malice plays the role of an attacker who subverts the authentication mechanism. Below we have summarized the basics of operation of these PUFs.

2.4.2.1 Nano PPUF

Memristor-based crossbars are used for designing one of the first memristor-based PUFs called nano-PPUF [42]. A simulation model of the physical design of a public PUF is publicly available; however, simulation complexity can create a time-bound authentication protocol. The attack model assumes a computationally bounded adversary unable to simulate the exact output for a given PUF design. The non-linear equations governing the current-voltage relationship of memristors and the viability of fabricating large memristive crossbars provide the simulation complexity required by this PUF model.

In this PUF design, a public registry contains the simulation model for a given user's (Bob's) memristive PUF. When Alice wants to authenticate Bob, she first sends a random challenge vector $\mathbf{V_C} = \{v_1, v_2, \ldots v_n\}$, where, v_i represents a physical input. For the given PPUF at [42], $\mathbf{V_C}$ is the voltage applied to an $n \times n$ memristor-crossbar. Since Bob has the physical memristor, he can correctly respond to Alice's challenge. He sends the correct response vector $\mathbf{V_R}$. For a computationally-bounded attacker Malice, completing this step would require simulating the complete crossbar, which would be computationally prohibitive. For completing the authentication, Alice then picks a subsection of Bob's crossbar (a polyomino) and requests the voltages at the boundaries of this polyomino. Bob sends the measurement and simulation results. Alice can accurately simulate the smaller polyomino using $\mathbf{V_C}$ and $\mathbf{V_R}$, and verify Bob's results. Thus, Alice can authenticate Bob.

This initial PUF design suffers from several crucial drawbacks. The crossbar simulation and the results from the physical crossbar will only match if the physical conditions that affect the current in a memristor (such as temperature, history of current flow, aging) remain the same. This is a difficult condition to fulfill for such design. Moreover, an attacker can try machine learning and model building attacks on passively obtained challenge responses to breaking the authentication scheme. Additional improvements considering these physical effects on this PUF design are discussed in [43] and [44].

2.4.2.2 CMOS-Memristive PUF

Nano-PPUF uses memristive-crossbars to generate unique challenge-response pairs. Unique device properties of single memristor cells are also used for designing other PUFs and authentication system discussed in [5, 45, 46, 47, 41]. Most of these works depend on the bias-dependent write-time and fabrication variations of memristors. For example, memory-based PUF cells proposed in [5] uses the fabrication variation dependent write-time differences as an entropy source. Circuit design for this PUF is shown in Figure 2.11.

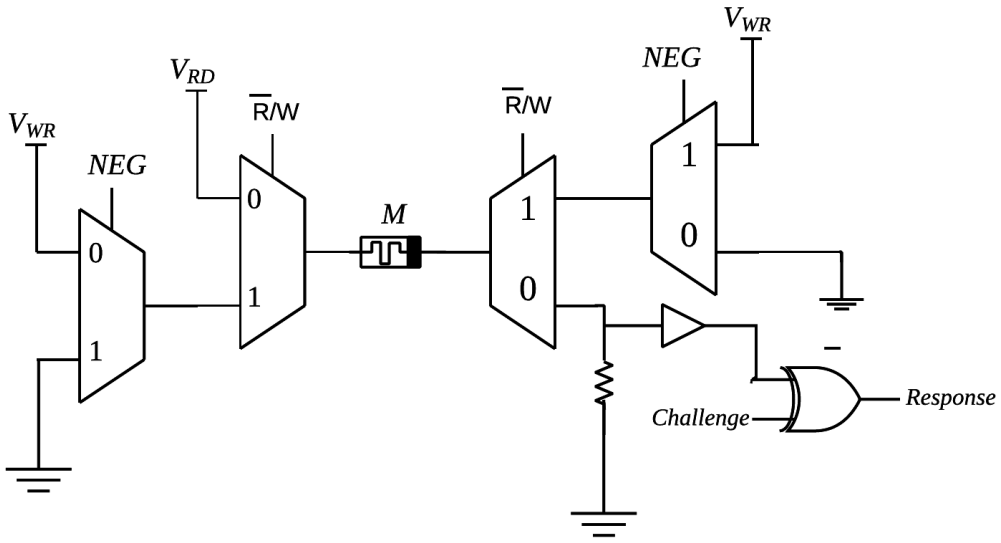

Figure 2.11: 1-bit memristive memory based PUF cell proposed in [5].

The principle of operation for this PUF is simple. First a RESET signal is applied using $NEG = 1$ and $\bar{R}/W = 1$. This puts the memristor M in a high resistive state (HRS). Then a SET operation is performed with a write pulse $t_{wr,min}$ applied at V_{WR} and $NEG = 0$. The write pulse $t_{wr,min}$ is selected in a way that the likelihood of state transition is 50%. Since the write-time of memristors are random for a given voltage due to the physical randomness in their construction, the circuit (Figure 2.11) can harness 1-bit of entropy from the memristor. With a read operation $\bar{R}/W = 0$ and a challenge bit, one can receive a 1-bit response from this PUF circuit.

Similar weaknesses such as the effect of aging, noise at the supply voltage as discussed for the Nano-PPUF exist for this design also. An improvement of this design can be made by dividing the $t_{wr,min}$ into smaller pulses and use the number

of pulses as a challenge vector as described in [41]. Improved PUF design with proper SET time determination for the cell shown in Figure 2.11 is also reported by Rose *et al.* in [45]. Mazady *et. al.* [48] have experimentally verified the proposed design.

The second PUF design proposed in [5] is dependent on the stochastic nature of filament formation of memristors during fabrication. Two memristors connected laterally (*i.e.,* they share the same bottom electrode) is used for a PUF cell in this design. The top electrodes of these two memristors are connected to V_{DD} and GND respectively as shown in Figure 2.12. The operation for bit-generation is simple. Experimental results have suggested that a lateral SET operation can SET both of the devices; however, a lateral RESET puts one of the devices to HRS, and the other remains in LRS. This switching is dependent on the difference in the stochastic variation of these two laterally connected device. By comparing which one of the device has state transition, a single bit of entropy can be generated. The circuit schematics for the bit extraction is given in detail at [5].

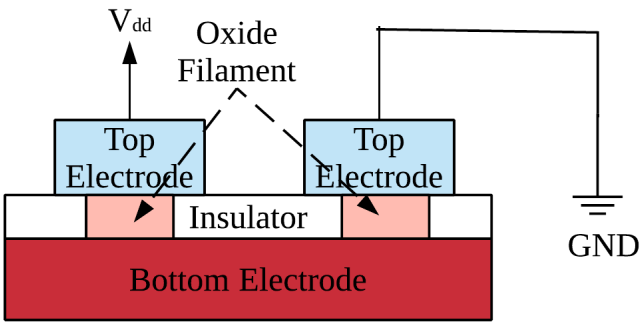

Figure 2.12: Laterally connected memristors for designing PUF cell [5]

2.4.2.3 Memristive Super-High Information Content (SHIC) PUF

The high packing density of memristive crossbar is useful for creating large memories with small area footprints. Passive crossbar arrays for memristors can hold a large amount of information content which can be used for designing SHIC-PUFs discussed in [49]. SHIC-PUFs have two fundamental component: a high-density memory with random information content and a slow readout technique for this memory. It should be noted that the conventional memory designs are focused on fast readout circuitry for memristors due to the requirement of faster data processing and storage. However, this leads to rapid leaking the random information stored in a SHIC-PUF. Therefore, this PUF requires particular readout mechanism to reduce the information leakage. These are strong PUFs with linear CRPs.

The high density of the memristor crossbars makes memristors an ideal candidate for a SHIC-PUF. Some memristor designs require a initial programming step (also known as the *forming* step) after fabrication, where the device is *formed* (*i.e.,* becomes a memristive device instead of a common resistor). Memory contents in a memristive memory are found to be random after a forming step. This random startup resistive states can be the entropy source for such PUFs. Furthermore, probing attack on compact memristive crossbar is difficult, which makes such design secure against probing attacks.

2.5 Conclusions

With the recent advances in memristors as a potential building block for future hardware, it becomes an important and timely topic to study the role that memristors may play in hardware security. To address this issue, this chapter presents a survey on research activities on memristor modeling and potential application of memristors in hardware-based authentication. First, we give an overview of the current literature on memristor experimentation, characterization, and modeling which includes Chua's original theoretical prediction model, more detailed models based on recent memristor implementations, and the SPICE simulation, models. Then, we report the current research efforts on memristor-based physical unclonable function designs. We observe that most of these works have limited scope and are based on simplified memristor models which diminish their practical value in security applications. However, these approaches provide a glimpse of current state-of-the-art for memristor-based CRP generation and authentication techniques.

Chapter 3: User Authentication Protocol Design using Memristive Hardware

3.1 Introduction

In this chapter, we address the problem of secure and lightweight authentication of single and multiple entities in a computationally constrained system. We assume an entity (say Alice the verifier) tries to simultaneously authenticate k-out-of-n-other entities ($B_1, \ldots B_k$). This problem can be simplified as simultaneous verification of k-out-of-n cryptographic keys. If these keys are generated from a secret that Alice knows or possesses, then, instead of checking n-keys separately, Alice can use the keys provided by $B_1, \ldots B_k$ to regenerate her secret and authenticate all entities at once. Note that, as Alice uses a k-out-of-n secret sharing scheme, Alice can reconstruct the secret from any of the k-users shares. The problem of authenticating a single user (say Bob) then reduces to a 1-out-of-1 authentication problem. For avoiding collusion among the users, this procedure requires Alice to have some interference in key/password generation process during the registration of the users.

Since our work focuses on a computationally constrained IoT system, we consider using emerging memristor devices for secret sharing based user authentication.

The key contributions of this chapter can be recognized as follows:

1. We design two single user authentication protocols and one multiple user authentication protocols using memristive hardware.

2. We discuss the potential attacks on these protocols and provide corresponding countermeasures for understanding the merits of the proposed schemes regarding security.

3. Finally, we perform HSPICE simulation using PTM's 65nm MOSFET model and Stanford memristor model to validate the reliability of these authentication protocols.

The main advantage of our approach is that it does not need the computational expensive cryptographic techniques. In that sense, our approach can be considered as a lightweight cryptographic method for secret sharing based authentication. Intrinsic energy efficiency of memristive systems makes this even more attractive. In section 3.2, we present the common requirements required from resistive memory based devices for solving lightweight authentication problems, then section 3.3 presents the common notation used in this chapter and section 3.4 describes the common utility functions applicable for memristive operation. At section 3.5-3.7, we propose single and multiple user authentication protocols using memristive hardware. Finally, section 3.8 and 3.9 illustrates simulation of the authentication protocols in the proposed hardware and reports the reliability issues and their countermeasures.

3.2 Requirements for Memristor-Based Authentication

From the device features described in section 2.4.1, below we specify our assumptions on the expected device behavior that are needed for the design of the secret sharing based user authentication protocols, and also give the justification of these requirements and assumptions. For our discussion, let us assume that the SET operation is achieved by a singular or multiple constant duration voltage-pulses which are denoted by ON-pulses.

R1 Changes due to consecutive ON pulses are *additive* in nature.

When given sufficiently high frequency and proper amplitude, short duration ON pulses can cause memristor resistance state transitions just like a longer single voltage pulse. For example, if the device is not already in LRS, two consecutive ON pulses would put the device in a lower resistive state than a single ON pulse. As we can see from equation 2.4, with V=0, the rate of change of the state is zero, so we would expect a device to retain its resistive state with zero input voltage. Therefore, the device should need the same amount of time with the bias voltage on when it is written using multiple short pulses or a single long pulse. For example, if a device requires a 1.5V-100ns pulse to go from HRS to LRS, it should show similar transition when ten (or a similar number of that order) of 1.5V-10ns pulses are applied. This is validated in our simulations as well.

R2 The memory elements are *monotonic*.

If the device is not in an LRS, applying an ON pulse will always decrease the resistance, and its effect cannot be undone without resetting the device.

R3 Intermediate resistive state transitions are random; however, the number of ON pulses remains nearly the same for different programming cycles.

As we have seen in memristor's nonlinearity feature F5, when the memristor device transitions from one resistive state to another, its intermediate resistance does not change linearly with time. Instead, the change is exponential and has random variations due to local Joule heating and stochastic nature of ionic transportation through the thin-film. The can be verified by the I-V relationship of memristor [9].

R4 The number of short duration pulses during the state transition cannot be predetermined without information about the programming conditions.

This is the discrete version of requirements R1 under the assumption of R3. In another word, one can not imply how many short duration pulses have been applied by observing the intermediate resistance value during the state transition. Furthermore, to learn the exact/an approximate number of pulses required for a given state transition at a given bias, one memristor to observe a complete programming cycle. At a given bias, it would be impossible to extrapolate the number of pulses required by measuring the resistance change of the device for an only small period during a transition.

R5 There exists a higher current level for which the device can be RESET to a

non-recoverable high resistive state (NRHRS), i.e., the device cannot be SET again after it reaches the NRHRS.

R6 Memristive devices stay at random resistive states after fabrication. A forming bias voltage V_{form} is required to make the devices programmable.

3.3 Notations

In this chapter, the set of integers modulo an integer $q \geq 1$ is denoted by $\mathbb{Z}_q$. Matrices, vectors single elements over $\mathbb{Z}_q$ are represented by consecutively upper case bold letter, lower case bold letters and lower case letters such as $\mathbf{X}$, $\mathbf{x}$ and x. For a vector $\mathbf{x}$, the length of the vector is denoted by $|\mathbf{x}|$, i^{th} element is represented by $\mathbf{x}[i]$ and $wt(\mathbf{x})$ denotes the Hamming weight (i.e., the number of indices for which $\mathbf{x}[i] \neq 0$) of the vector $\mathbf{x}$. The Hamming distance between two binary matrices are denoted by $hd(\mathbf{A}, \mathbf{B})$ (i.e., the number of indices for which $\mathbf{A}[i][j] \neq \mathbf{B}[i][j]$). $hdp(\mathbf{A}, \mathbf{B})$ represents the parity of $hd(\mathbf{A}, \mathbf{B})$ (i.e, $hdp(\mathbf{A}, \mathbf{B}) = 0$ for even parity of $hd(\mathbf{A}, \mathbf{B})$ and $hdp(\mathbf{A}, \mathbf{B}) = 1$ for odd parity of $hd(\mathbf{A}, \mathbf{B})$). $\mathbf{0}^{\ell \times n}$ represents a $\ell \times n$ null matrix. The Hadamard product of two matrices $\mathbf{A}, \mathbf{B}$ is given by $\langle A \circ B \rangle$. Given two vectors $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{z} = \mathbf{x} \oplus \mathbf{y}$ represents bitwise XOR operation of $\mathbf{x}$ and $\mathbf{y}$. Concatenation of two vectors are represented with $||$ symbol. $c \xleftarrow{\$} \{\mathbf{x} \in \mathbb{Z}\}$ represents a random sampling of $\mathbf{x}$. We denote *probabilistic polynomial time* (PPT) algorithms with upper case calligraphic alphabets such as $\mathcal{A}$. Therefore, if $\mathcal{A}$ is probabilistic, then for any input $\mathbf{x} \in \{0,1\}^*$ there exists a polynomial $p(.)$ such that the computation of a terminates in at most $p(|\mathbf{x}|)$ steps. All the physical quantities

such as voltage, current and device names are denoted with upper case letters.

3.4 Utility Functions

Before proposing memristor based user authentication protocols, we first describe the basic memristor utility functions that are critical for these protocols.

RESET: A RESET operation uses a fast negative biased pulse between the top and the bottom electrode to put a memristor to the high resistive state (HRS).

SET: A SET operation puts a memristor to the low resistive state (LRS). For authentication purpose, we will use the bias dependent write time variability during set (transition from HRS to LRS) operation. We will use multiple short duration ON pulses for setting the device instead of a longer ON pulse.

SET Pulse Count (SPC): The number of ON pulses required to transit a memristor from HRS to LRS.

Precondition: A k-bit precondition operation applies k consecutive ON pulses to the memristor device. If the device reaches the LRS after k_n pulse where $(k_n < k)$, the device is RESET and the remaining $(i.e., (k - k_n))$ pulses are applied.

Read State: No pulse for one cycle. Then detect whether the memristor device is at HRS or LRS.

3.5 Single User Authentication: Protocol I

For single user authentication, we will assume an interactive protocol between a single prover $\mathcal{P}$ and a verifier $\mathcal{V}$. Both the prover and the verifier has some knowledge about a shared secret $\mathbf{x}$. The secret is generate through a key-generation procedure $KeyGen(1^{\lambda})$,where λ is a security parameter. The authentication protocol responds with the outputs accept or reject after a successful run of the protocol. First, we present a simple key generation procedure using the intermediate state transition of a memristor.

3.5.1 Description of Protocol I

In this protocol, the prover and the verifier participates on one time enrollment phase that generates the secret using the key-generation algorithm $KeyGen01$ and distribute to the parties. For authentication, multiple round of interactive authentication (as presented in table 3.1) is performed.

Assumptions: Along with the requirements R1-R6, assume that there exists high variability in the fabrication of the devices and the state transition of a memristor with small voltage pulses are unique for each device and highly repeatable.

Algorithm 2 Key Generation Using Resistive States

1: **procedure** $\mathbf{x} \leftarrow KeyGen01(1^\lambda, \alpha)$ ▷ $\alpha \in \mathbb{N}$ is the experiment repetition value

2: For a given memristor R, select a set of $\{V_{select}, V_{BL}, V_{WL}\}$ that demonstrates a complete programming cycle with an average write time $t_{wr,avg}$.

3: Assume the key length $\ell = nb$, where $n, b \in \mathbb{N}$. ℓ, $t_{wr,avg}$, $V_{select}, V_{BL}, V_{WL}$ values depend on the security parameter k.

4: **for** j=1 . . . α **do**

5: **RESET** R.

6: Apply n-consecutive pulses (where $n = \ell/b$, and b is the number of bits used for representing a floating point number in the system) at the word_line with pulse width $t_p = t_{wr,avg}/n$ and measure the corresponding current I_i through R after i^{th} pulse, where $i \in \{1, \ldots, n\}$.

7: $\mathbf{X}[j, :] \leftarrow I_1 || I_2 \ldots, || I_n$.

8: $\mathbf{x_a}[i] \leftarrow$ majority values of the i^{th} column of $\mathbf{X}$

9: Use a b-bit ADC to convert $\mathbf{x_a}$ to a binary string $\mathbf{x}$.

10: **return x**

Enrollment: The prover and the verifier use a random memristor M and generates the ℓ-bit secret $\mathbf{x} \leftarrow KeyGen01(1^\lambda)$. They also share a b-bit precondition value s. The authentication protocol assumes that the verifier has the memristor and the prover knows the secret state transitions of the memristor given by $\mathbf{x}$. ϵ is the predetermined error threshold for the authentication.

Table 3.1: Single round interactive authentication for Protocol I

Prover($\mathbf{x}, s$)		Verifier(M, s, ϵ)
		— **RESET** M
— Find the current I_i for the precondition value s using $\mathbf{x}$		— **Precondition**(M,s)
		— $c \xleftarrow{\$} \{i \in \{1, 2, \ldots n-s\} : n = \ell/b\}$
		— **Precondition**(M,c)
		— Measure the current I_{rand} with the parameters $\{V_{select}, V_{BL}, V_{WL}\}$ specified at the key generation step
	$\xleftarrow{I_{rand}}$	
— Find the distance (r) between the current values I_i and I'_{rand} at $\mathbf{x}$, where I'_{rand} is the near-most value of I_{rand}. r is approximately the number of ON pulses required to be applied to memristor R at the s-state to produce I_{rand}		
	$\xrightarrow{r}$	
		— If $abs(r - c) < \epsilon$ accept, otherwise reject.

Update: After one successful round of authentication the prover and the verifier both updates $s \leftarrow s + r$.

3.5.2 Hardware Design

Protocol I requires 1T1M memristive cell (as shown in Figure 2.8) with current measuring capabilities. This protocol is designed based on the assumption that reducing the bias voltage of the memristors increases the total write time and enables multi-level operation of the device that is less susceptible to the effect of environmental variation and Joule heating. Although elongating the write-time of a memristor memory cell is less favorable for memory operation, it is a welcoming feature in security, since it increases the read-out time for the entire contents of the memory in several orders of magnitude. Recognizing the usefulness of this feature in the security context is a key contribution of this work.

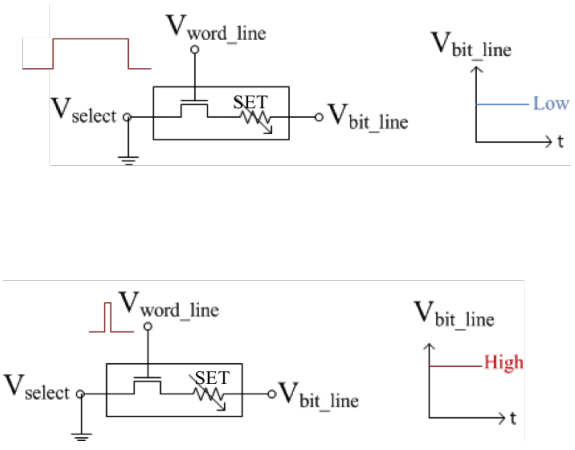

Figure 3.1: Bias dependent write time of a memristor.

For the realization of this protocol, we assume that the I-V curve during state transitions are repeatable for a memristor for a given bias and environmental con-

dition. From equation 2.1 it is evident that the non-linearity between the applied voltage and the corresponding resistance can create the state transitions for a given memristor unique. When the resistive characteristics are different for two different memristors, different amount of voltage is required for obtaining the same level of resistance [6]. Quantifying the application of bias voltage into voltage pulses, one can generate secrets based on the number of pulses needed for a given state transition.

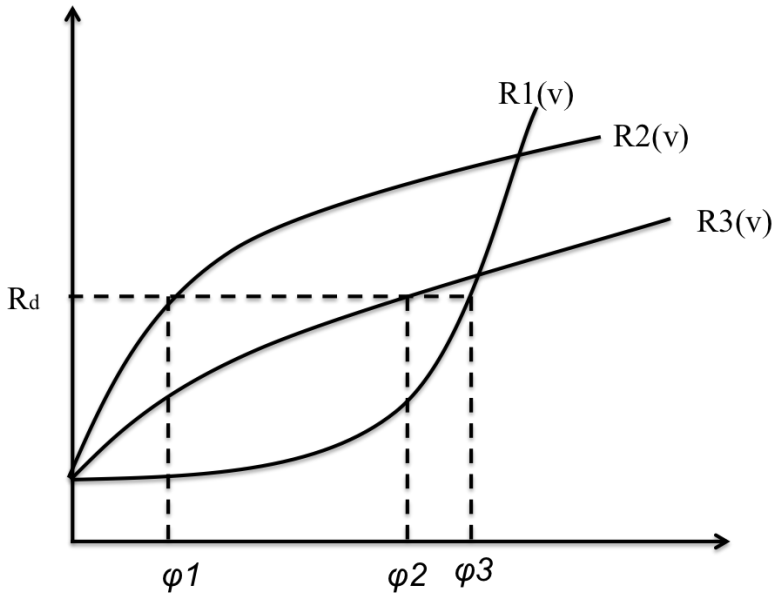

Figure 3.2: For memristive device with different programming cycle different voltages are required to reach the same resistive level [6].

It should be noted that noise in the bias-source or read-out can cause small changes in the resistive states that accumulate with time. From 2.1, we can see how that the flux can accumulate over time with small perturbations as,

$$\phi(t) = \phi_0 + \int_{t_0}^{t} v(\tau) d\tau \tag{3.1}$$

where ϕ_0 is the initial flux. To avert such problem, Leon Chua proposed the application of a voltage doublet with equal positive and negative read-out pulses[6].

3.5.3 Security Analysis

We assume an attacker $\mathcal{A}$ who wants to generate an accept from the verifier $\mathcal{V}$.

Theorem 3.1: If $\mathcal{A}$ guesses at random, she has $(1/n)$ chance of success, where $n = f(\lambda) = \ell/b$ is the number of voltage pulses required for a complete SET-RESET cycle.

Proof: Since, the memristor requires n pulses for a complete SET-RESET cycle, the response r is in $\mathbf{r} = \{0, \ldots n\}$. Therefore, the chance of success for random guessing is $1/|R| = 1/n$

The security can be improved by introducing multiple rounds and for β rounds the chance of success for an attacker reduces to $(1/n^{\beta})$.

Theorem 3.2: Assume $\mathcal{A}$ can eavesdrop on the challenge response pair (I_{rand}, r) for the authentication. Then she can learn the complete challenge-response by eavesdropping on an average of $O(n^2)$ consecutive communications.

Proof: If n-pulses are used for complete programming of the device, then, we can assume that there are n distinct states in the device. Transitions between the states can be represented by a fully connected graph of n vertices with $n(n-1)/2$ edges. Therefore, to completely learn the state transition $\mathcal{A}$ needs to solve an average of $O(n(n-1)/2) = O(n^2)$ equations describing consecutive runs for the authentication protocol.

Therefore, we can see that the protocol is weak against eavesdropping attack. Lowering the bias voltage of the memristor can exponentially increase the number of pulses (n) and thus increase the security against eavesdropping and random guessing attack, however, it would create increasing storage burden of $O(n)$ to the prover. Furthermore, this protocol requires the extensive capability of current measurement. This problem can be addressed by using differential measurement techniques where the resistance of the memristor is compared with other fixed resistors with different resistances.

Observation 3.3: [Secure storage] Access to the memristors without the knowledge of biasing condition will reveal μ-knowledge to an attacker, where μ depends on the number of parameters required to correctly model operating cycles of a given memristor.

In this protocol, we keep the memristors response to the input pulse (the state of the memristor p_i) as the secret shared between the prover and the verifier. Based on device requirement R1 and R4, the attacker with single access may be able to obtain whether a memristor is at HRS or LRS, but he won't be able to find the cycle-accurate state information of the memristor. He will need to access the memristor an average of $O(\mu)$ times (assuming a linear model) to create the programming model of the device.

An essential drawback of this protocol is that it requires the device to have same programming cycles, *i.e.,* the device must maintain the similar programming behavior over different cycles. For a reliable application, we need to either reduce such restrictions or provide proper error-correction mechanisms such as majority

voting and fuzzy extractor algorithms[50] for a secure application. Helper data in error correction can leak information, and therefore, such construction must be carefully designed.

3.6 Single User Authentication Protocol II

Next, we present a protocol using the basic schemes of memristor-based secret sharing for single-user authentication. We use a special access structure that computes the XOR of two different physical properties, namely the input voltage and the resistive state of a given memristor. The circuit is shown in Fig 3.3. was developed by Shin *et. al.* [51] as a resistive computing primitive. The key operating principle of this structure is simple. The resistive state of the output resistance R_P is the XOR of the applied voltage V_X and the input resistance R_Y. Details of the operation of this circuit are discussed in the next section.

3.6.1 Description of Protocol II

In this protocol, the prover and the verifier participate in one-time enrollment phase that generates and share the secret using the key-generation algorithm $KeyGen02$. For authentication, multiple rounds of interactive authentication (as presented in table 3.2) is performed.

Assumptions: We assume that the verifier has an access structure M containing ℓ-three-transistor-two-memristor blocks of Figure 3.3 (where ℓ is the length

of the provers key $\mathbf{x}$. ℓ depends on the security parameter λ. During authentication, the prover has access to the V_x terminal of M during the authentication phase.

Algorithm 3 RNG based Key Generation and Key Storage in Memristor

procedure $\mathbf{x} \leftarrow KeyGen02(1^\lambda)$

    Sample $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_2^\ell$

    return $\mathbf{x}$

procedure $\mathbf{h} \leftarrow KeySto1(\mathbf{x})$

    Sample $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_2^\ell$

    for all $i \in \{1, \ldots, n\}$ **do**,

        Configure the input resistance R_Y of $M[i]^{th}$ block using the following rule

        if $\mathbf{y}[i] = 0$ **then**, **RESET** R_Y.

        else SET R_Y.

    for all $i \in \{1, \ldots, n\}$ **do**,

        Apply the control voltage V_x of $M[i]^{th}$ block using the following rule:

        if $\mathbf{x}[i] = 0$ **then**, apply $V_x = 0$.

        else $V_x = V_{BL}$

    Generate $\mathbf{h}$, by reading the resistive state of control resistance R_p of the $M[i]^{th}$ block using the following rule:

    if $R_p = LRS$ **then**, $\mathbf{h}[i] = 1$

    else $\mathbf{h}[i] = 0$

    return $\mathbf{h}$

Enrollment: The protocol uses $KeyGen02(1^\lambda)$ to generate an ℓ-bit secret $\mathbf{x}$ and uses $KeySto1(\mathbf{x})$ to generate ℓ-bit $\mathbf{h}$. The prover keeps $\mathbf{x}$ and the verifier keeps $\mathbf{h}$.

The verifier also keeps M. Finally, after sharing $\mathbf{h}$ and $\mathbf{x}$, the access structure is locked, *i.e.,* the read/write mechanism for all R_Y in M is disabled.

Table 3.2: Single round interactive authentication for protocol II

Prover($\mathbf{x}$)	Verifier($M, \mathbf{h}$)
	— **RESET** all control resistors (R_p) of M
— The control voltage V_x of $M[i]^{th}$ block is applied using the following rule: a. If $\mathbf{x}[i] = 0$, apply $V_x = 0$. b. else $V_x = V_{BL}$	
	— Generate $\mathbf{h}'$ by reading the resistive state of control resistance R_p of the $M[i]^{th}$ block using the following rule: a. If $R_p = LRS$, $\mathbf{h}'[i] = 1$. b. else $\mathbf{h}'[i] = 0$. — If $\mathbf{h} = \mathbf{h}'$ accept, otherwise reject.

This scenario is one of the most straightforward applications of Ito, Saito, and

Nishizeki's constructions of secret sharing between two parties[52]. For each bit of provers' private key, the verifier stores only the XORed value of this bit with a random resistance. Therefore, the verifiers' authentication server is resistant to off-line dictionary attack. To get the provers private key, an attacker requires not only the content of the authentication server but also a knowledge of the access structure M used to generate $\mathbf{h}$. To create a hardware dependent access structure, the key storage procedure of protocol II can be updated as follows:

Assumption: Assume that there exists high variability in the fabrication of the devices and no forming voltage is applied on the memristors in the access structure.

Algorithm 4 Key Storage in Memristive Storage using random resistive states

1: **procedure** $\mathbf{h} \leftarrow KeySto2(\mathbf{x})$

2: The forming voltage V_{form} is applied to the control memristor R_ps.

3: **RESET** all R_p

4: **for all** $i \in \{1, \ldots, n\}$ **do**,

5: Apply the control voltage V_x of $M[i]^{th}$ block using the following rule:

6: **if** $\mathbf{x}[i] = 0$ **then**, apply $V_x = 0$.

7: **else** $V_x = V_{BL}$

8: Generate $\mathbf{h}$, by reading the resistive state of control resistance R_p of the $M[i]^{th}$ block using the following rule:

a. If $R_p = LRS$, $\mathbf{h}[i] = 1$.

b. else $\mathbf{h}[i] = 0$.

9: **return h**

3.6.2 Hardware Design

The core component of this protocol is the three-transistor-two-memristor configuration for computing XOR value of a logical voltage value and a logical resistive value. The circuit is given in Figure 3.3. For authentication with ℓ-bit secret, ℓ parallel blocks of 3T-2M units are required.

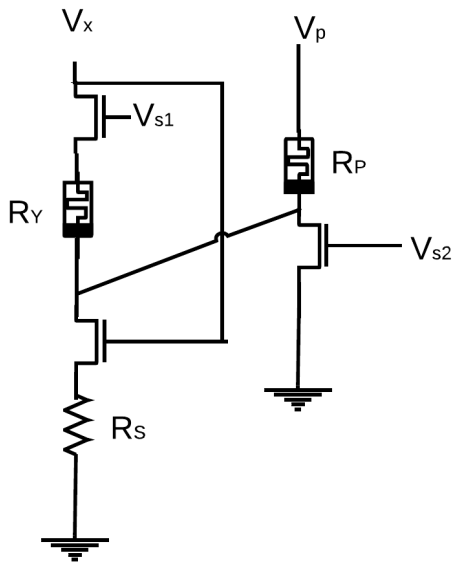

Figure 3.3: 3T-2M circuit configuration for single user authentication Protocol II

During the key storage/enrollment phase of Protocol II, the access structure for a 1-bit private key is created by turning on $s1$ and turning off $s2$ and choosing proper V_x, V_p and R_p. During authentication phase, first resetting of R_p is performed by keeping $s1$ off and $s2$ on and $V_p = V_{RESET}$. Then the key evaluation is done by turning on $s1$ and turning off $s2$. An example simulation of authentication using this hardware is given at 3.8.

3.6.3 Security Analysis

Protocol II defines a simple XOR-based scheme for key storage and authentication. The weakness of this protocol is: it requires that the prover has some form of physical access to the memristive structure M.

Theorem 3.4: The chance of randomly guessing the secret is $1/2^{\ell}$.

Proof: The Verifier ($\mathcal{V}$) compares $h' = h$, the attacker needs to guess the voltages for the bit-stream $\mathbf{y} = \mathbf{h} \oplus \mathbf{x}'$, where, $x'[i]$ denotes the resistive state for R_p of the i^{th} block. Since, this protocol only recognized only two resistive state (i.e., LRS and HRS), the success of random guessing is $1/2^{|\mathbf{h}|} = 1/2^{\ell}$. Similarly, an average of $O(2^{\ell})$ attempts are required to extract the secret $\mathbf{x}$.

Algorithm 3 assumes a random distribution of the resistances before the forming process and uses this randomness to create a one-time pad for generating $\mathbf{h}$. This can lead to the secure erasure of the secret $\mathbf{x}$ by erasing $\mathbf{h}$.

3.7 Multiple User Authentication

To address the problems regarding multi-user authentication, we propose to design a Visual Cryptography inspired memristor based multi-user authentication scheme. A small motivational example for 2-out-of-3-user authentication using memristor based threshold detector circuit is given below to illustrate the key concepts.

Assume that an entity Alice wants to simultaneously authenticate three users B1, B2, and B3. Alice chose a random key K, and for each bit of the key, for each user, Alice randomly choose a row from the following matrix C_0 and C_1 using the

given rules and gives the corresponding 3-bits to the users. The matrices are given as [21]:

$$C_0 = \{ \text{ all the matrices obtained by permuting the columns of } \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} \}$$

$$C_1 = \{ \text{all the matrices obtained by permuting the columns of } \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \}$$

and the distribution rules are:

i If the i^{th}-bit of the key is 0, distribute the rows of C_0 to B1, B2, and B3.

ii If the i^{th}-bit of the key is 1 distribute the rows of C_1 to B1, B2, and B3.

Note that for a single user, it would be impossible to guess the key bit. For authentication, Alice let any two or more users access the memristor for the duration of three pulses. The users apply their keys to each pulse. The keys are taken through an OR gate as shown in Fig 3.4. An ON pulse is used at a cycle if the result of the OR gate is one. Otherwise, no pulse is applied. If the i^{th}-bit of the key is zero, then for only two of the cycles the users will give an ON pulse, but there will be no ON pulse for one cycle. However, if the i^{th}-bit is 1, there will be three ON pulses. As a result, after the duration of three pulses, the device's resistance should be lower if the key was 1 and higher if the key was 0.

A threshold detector can easily detect this and reconstruct the key. This scheme can easily be modified to an n-out-n or a k-out-of-n authentication system.

One weakness of the above example is that to reconstruct a bit of the key; the users do not necessarily need Alice. If any two of the valid users come together, they can restore the bits. If we want a hardware dependent authentication, where the participating device should also be a factor during authentication, this might cause a collusion problem. However, this issue does not exist if Alice has some interference in key generation.

3.7.1 Description of the Protocol

For multiple user authentications, we will assume an interactive protocol between a multiple provers $\mathcal{B}$ and a verifier $\mathcal{V}$. All the provers and the verifier has some knowledge about a shared secret $\mathbf{x}$. The secret is generated through a key-generation procedure $KeyGenM(1^{\lambda})$, where λ is a security parameter. The authentication protocol responds with the outputs accept or reject after a successful run of the protocol. Here we formally describe the simple key generation and distribution procedure which is similar to the one proposed by Naor *et al* [21].

Assumptions: For the authentication scheme described below, we assume that R1-R6 described at 3.2 holds. We also expect that the verifier owns the authentication hardware and controls the bit-line voltage (V_{BL}). The verifier also acts as the dealer $\mathcal{D}$ that knows $\mathbf{p}$ (where $(p[i])$ is the number of ON pulses required for pushing the resistance of the memristor $M[i]$ from a fixed HRS state to the reference resistance (R_{in}).

Algorithm 5 Key Generation and Distribution for Multiple User Authentication

procedure $\mathbf{x} \leftarrow KeyGenM(1^{\lambda})$

  Sample $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_2^{\ell}$

  return x

procedure $(\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_k}) \leftarrow KeyDistM(\mathbf{x})$

  Consider a ground set G consisting of k elements $g_1, g_2, ...g_k$. The subsets of G with even cardinality are $e_1, e_2, .., e_{2^{k-1}}$ and odd cardinality are $q_1, q_2, , q_{2^{k-1}}$

  Construct two Boolean matrices $\mathbf{B_0}$ and $\mathbf{B_1}$ of size $k \times 2^{k-1}$ such that, $\mathbf{B_0}[\mathbf{i}, \mathbf{j}] = 1$ iff $g_i \in e_j$ and $B_1[\mathbf{i}, \mathbf{j}] = 1$ iff $g_i \in q_j$ for $i \in \{1, \ldots k\}$ and $j \in \{1 \ldots 2^{k-1}\}$

  Construct collections ξ_0 and ξ_1 by permuting all the column of $\mathbf{B_0}$ and $\mathbf{B_1}$

  for all $i \in \{1, \ldots, \ell\}$ **do**

    $\mathbf{C_0} \xleftarrow{\$} \xi_0$, $\mathbf{C_1} \xleftarrow{\$} \xi_1$

    for all $j \in \{1, \ldots, k\}$ **do**

      if $\mathbf{x}[i] = 0$ **then** $\mathbf{S_j}[i, :] = \mathbf{C_0}[j, :]$ ▷ distribute the rows of C_0.

      else $\mathbf{S_j}[i, :] = \mathbf{C_1}[j, :]$ ▷ distribute the rows of C_1.

Table 3.3: Single round interactive authentication for multiple prover, single verifier

Provers($\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_k}$)	Verifier($M, \mathbf{x}$)
	—**RESET** all memristors of M
	— Set the bias point such that all memristor requires p-ON pulses to move to the LRS, where p is the number of columns of $\mathbf{B_0}, \mathbf{B_1}$.
for all $i \in \{1, 2, \ldots \ell\}$	
for all $j \in \{1, 2, \ldots p\}$	
Simultaneously apply all $S_*[i, j]$ on the U_* gates at Figure 3.4	
	— If $\mathbf{V_{out}} = \mathbf{x}$ then accept, else reject.

Due to the fabrication variations, each memristor would have different write-time, which will lead to different (but of the same order) values of p_i for various devices at a given V_{BL}. Since the dealer/verifier owns the equipment, she knows the value of p_i for any given Ri. As different memristor requires different numbers of ON pulses, additional 1s need to be padded in each row of C_0 and C_1. A more straightforward solution to this problem is to precondition each memristor with this

additional 1s. Before each round, the verifier can precondition each memristor with precondition value **y** so that each requires the same number of extra ON pulses to reach LRS as discussed in procedure ErrCorr. Another way is to use the block-length defined constructs. Block length is the number of ones resulted by OR-ing all the columns of C_1. Therefore, the dealer can design block length adjusted matrices C_0' and C_1' depending on the number of ON pulses required for a state transition and follow $KeyDistM$ to share a bit ($\mathbf{x}_i$) of the key **x**.

Algorithm 6 Extraction of Correct Precondition Values

1: **procedure y** $\leftarrow ErrCorr(k, M)$ ▷ M is a memristive array of length ℓ
2: **for all** $i \in \{1, \ldots, \ell\}$ **do**
3: **RESET** $M[i]$
4: Apply k ON pulses on M[i]
5: **READ** M[i].
6: **if** $M[i] = HRS$ **then**
7: Apply t-pulses to put $M[i]$ in LRS
8: $y[i] = t$
9: **else** $y[i] = 0$
10: **return y**

To make a hardware dependent authentication scheme, the dealer can choose smaller block length or pad less number of 1s at the end of C_1' for some of the random bits of the key X. Since the provers do not have access to the hardware, they do not know the exact value of p_i's for a given $M[i]$. Therefore, the dealer can share a 0 by sharing contents of C_1' for cases where C_1' is not correctly generated from C_1.

Thus, it would be impossible for k-users (or even all the users) to collude and guess the secret key.

3.7.2 Hardware Design

For developing a memristor-based secret reconstruction and authentication circuit, we need to extract the current state of a given device. It should be noted that improper reading of a memristor cell with high bias voltage across the cell can change its states, and therefore, the read operation should be done carefully. To accomplish this, we have used a simple CMOS compatible differential sense amplifier. The resistance of the device R_x is compared with an on-chip resistance R_{in} that has a threshold resistance value. If $R_x < R_{in}$ the circuit outputs 1, else it outputs a zero. The basic circuit is shown in Figure 3.4 which can be used for reconstructing the shared secret.

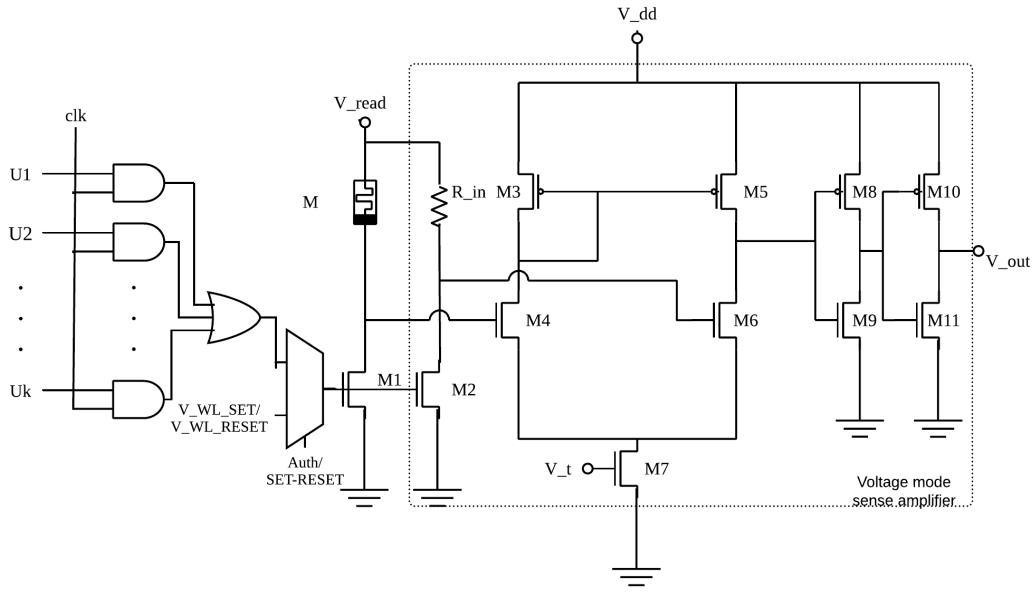

Figure 3.4: Circuit for applying the pulsed input from k-users and the assisting voltage-mode sense amplifier-buffer circuit for reconstructing the secret.

3.7.3 Security Analysis

The construction used for secret sharing in table 3.3 is a k-out-of-k scheme with the security parameters m, r, α with $m = 2^{k-1}, \alpha = 1/2^{k-1}$, and $r = 2^{k-1}!$, where m is the number of pixel in the a share for a single bit, α is the relative difference between the weights in the representation of a '0' and a '1', and r is the size of collections ξ_0, ξ_1 [21].

Key Collision and Contrast Note that, for the given scheme in procedure KeyDistM($\mathbf{x}$), the number of columns of any matrices of the collection ξ_0 that generates an output '1' when all the elements of the columns are "or"-ed is $2^{k-1} - 1$, while the number of columns of any matrices of the collection ξ_1 that generates an

output '1' when all the elements of the columns are "or"-ed is 2^{k-1} [21]. Therefore, there exists a contrast difference in ξ_0 and ξ_1. This is because there is always a column in $\mathbf{C_0}$ that is indexed by the empty set [21].

However, the number of '0' or '1' remains the same for any rows. Thus, the keys will be indistinguishable for each share [21]. Any permutation of the columns will return the same distribution because if we consider the rows as the subset of the ground set of size 2^{k-1}, every intersection of $k-1$ rows and its complement will be of size two [21].

Cheating and Secret Reconstruction by User Collusion A weak adversarial model can be assumed from a group of dishonest participants in a k-out-of-n secret sharing model. Such attack is defined as cheating where a group of unscrupulous users uses fake shares to alter the secret that was distributed at the initial phase. This is a typical attack scenario in visual cryptography-based secret sharing, and there are several practical countermeasures proposed by Horng et al. [53]. Since we have used visual cryptography for authentication purposes with the secret shared at the initialization phase, cooperation between multiple dishonest users would create a denial-of-service attack on the authentication system. If the dealer uses hardware dependent scheme, fraudulent users cannot create a cheating attack or reconstruct the secret by themselves since the dealer/verifier owns the hardware. Additionally, for detecting such attacks, the dealer can put redundant information in each share that constructs a 2-out-of-2 secret sharing with dealer/verifier and the prover.

3.8 Simulations of the Proposed Designs

We have conducted experiments on the performance of the proposed circuit and protocols and explored their reliability over physical and environmental variation. For these experiments, we have used the Verilog-A model of memristors proposed by Guan et al. [40]. This is a variability-aware memristor model that takes account of the critical impact of temperature change and temporal variations [40]. We have used PTM's 65nm MOSFET models for designing the authentication unit. The simulations are performed in HSPICE platform. For calculations presented in this section, we have used the parameters listed in Table 3.4 and 3.6 and 3.7.

Table 3.4: Memristor model parameters used for the experiments presented in this work.

Parameter Name	Value
g_0	$0.25 \times 10^{-9}m$
V_0	0.25V
ν_0	10
I_0	$1000 \times 10^{-6}A$
β	0.8
γ_0	16
T_{crit}	450K
δg_0	0.02
T_{smth}	500K
Ea	0.6
a	$0.25 \times 10^{-9}m$
T_{ini}	298K
F_{min}	$1.4 \times 10^{9}V/m$
g_{ini}	$17 \times 10^{-10}m$
g_{min}	$2 \times 10^{-10}m$
g_{max}	$17 \times 10^{-10}m$
R_{th}	$2.1 \times 10^{3}\Omega$
L	$12 \times 10^{-9}m$

Table 3.5: NMOS and PMOS parameters used for the simulations for the circuit given in Figure 3.4.

Parameter Names	Value
L_{M1}, L_{M2}	65nm
W_{M1}, W_{M2}	130nm
L_{M3}, L_{M5}	65nm
W_{M3}, W_{M5}	130nm
L_{M4}, L_{M6}	65nm
W_{M4}, W_{M4}	260nm
L_{M8}, L_{M10}	65nm
W_{M8}, W_{M10}	260nm
L_{M9}, L_{M11}	65nm
W_{M9}, W_{M11}	260nm
V_{th0} (for all the NMOS)	0.423V
V_{th0} (for all the PMOS)	-0.365V

For our first experiment, we provide a simple demonstration of single user authentication using Protocol II and the access structure shown in Figure 3.3. Here, we assume that the access structure can contain a random resistive state as determined in the registration phase. For this experiment, we consider Bob the prover has a binary string as a private key. We have demonstrated several consecutive authentication attempts in Figure 3.5. Note that if Bob's correct key is '0' (or '1'),

Alice (the verifier) will see an HRS or LRS in R_P based on the random resistive state of R_Y set at the enrollment phase. Therefore, it would be impossible to reconstruct Bob's actual key by Alice without tampering the access structure.

Table 3.6: Operating conditions used for the single-user authentication experiments presented in Protocol II.

Parameter Name	Value
Clock frequency (f_{clk})	20MHz
$V_{p_{RESET}}$	-1.8V
R_S	10kΩ
V_x	0-2V

Table 3.7: Operating conditions used for the multi-user authentication experiments presented in this work.

Parameter Name	Value
Clock frequency (f_{clk})	50MHz
V_BL_SET	1.8V
V_WL_SET	1.0V
V_WL_RESET	2.7V
V_BL_RESET	1.9V
R_in	250$k\Omega$

Next, we provide a simple 3-out-of-3 secret sharing based user authentication

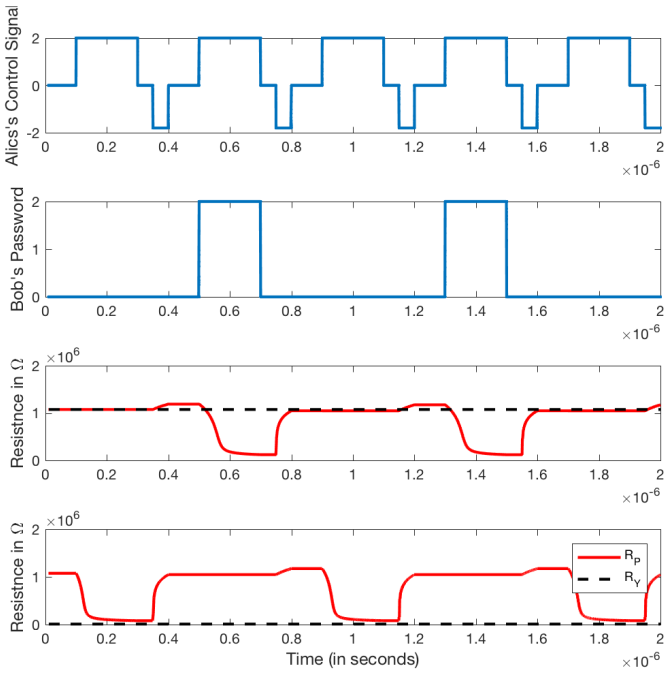

Figure 3.5: Simple demonstration of single user authentication. Here Bob's private key string is "1010" which he applies beginning at 500ns. Before the application of each bit, Alice RESETs the corresponding output memristor R_P. The bottom two figures represent the changes in the resistive state of the output resistance in two different cases. If the configuration resistance in the access structure (i.e., R_Y) is locked at HRS in the enrollment phase (as shown in the third figure from the top) then for Alice would observe a "LRS-HRS-LRS-HRS" pattern in the output resistance R_P, whereas, for R_Y locked in LRS, Alice would see "HRS-LRS-HRS-LRS" pattern. Note that, using one access structure for multiple bits of secret share can reveal information about the secret; therefore, each structure should be used for one-bit of Bob's key.

using Protocol 3. For the given parameters, we find that it requires eight ON pulses (i.e., SET operation) to move the device in question from a high resistive state to the low resistive state. Therefore, after eight consecutive ON pulse to R_x, one would observe logic 1 at V_{out}. Since we want Alice to reveal the secret by applying the final ON pulse, a system with block-length of 7 would be required for doing multi-user authentication. If we consider a 3-out-of-3 user authentication system, then using the constructions provided in [22], we have C'_0 and C'_1 respectively as:

$$C_0 = \{\text{all the matrices obtained by permuting the columns of } \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}\}$$

$$C_1 = \{\text{all the matrices obtained by permuting the columns of } \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}\}$$

In Figure 3.6, we have shown how the verifier (Alice) verifies a 1 or 0 shared to the participating entities.

3.9 Discussions

Apart from computationally-bound eavesdropper and random guessing attackers, there are some other attacker models for the protocols proposed in this chapter, such as attackers creating denial-of-service and side channel attacks.

Denial of service In this attack, an attacker (Malice) attempts to alter the provers'

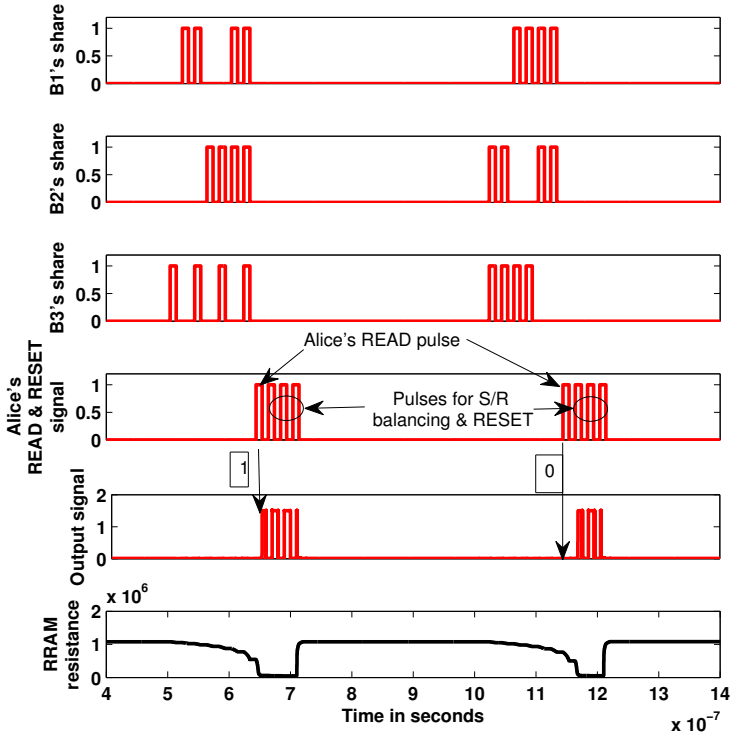

Figure 3.6: Simultaneous multi-user authentication using memristor based hardware. Alice verifies a 1 and a 0 shared to the participating entities. The first pulse from Alice is the READ pulse and the three subsequent pulses are for ensuring balanced SET/RESET operation.

state of the memristor devices by applying more currents to the memristors that store the private key-related information so that when the legitimate user enters the authentic key, the system will not be able to verify that.

Due to a lot of memristor devices in the authentication architecture, in single-user authentication it may not be easy for Malice to launch this attack to a particular victim unless she correctly identifies which set of memristor devices correspond to the victim. However, Malice may succeed to attack random victims. To countermeasure such attack, we can restrict user's access to the memristor devices. Unfortunately, this would lead to a denial of service attack in the authentication verifier side, and like other authentication disabling mechanisms, such preventive measure eventually would require intervention from the verifier. Therefore, one of the weaknesses of these protocols comes from DoS attacks.

However, for multiple user authentications providing a train of 1's at user input will create a practical denial of service attack for all k users. A simple finite state machine consisting one D-flip flop and an AND gate can be placed at each user terminal to detect such series of 1s. for k-users the flip-flops can be reset after each cycle along with the memristors.

Side channel attacks This is a group of influential attacks where Malice target the vulnerabilities in the hardware implementation of the security primitives and protocols. By measuring the side channel information (such as power, timing, and EM emission) during system's execution, secret information of the system can be revealed.

We are not aware of any side channel attacks to memristor based systems. But

should such attacks emerge, most of the existing countermeasures (such as careful engineering, using redundancy, design obfuscation, and so on) should be applicable.

Finally, the effectiveness of the protocols will also depend on private-key collision (false negative) and false positive alarms. False negative refers to the case when different passwords are deemed to be authentic for one user. This is a problem of password/secret selection. One standard solution is to add salt as the UNIX password management system does. When users choose their passwords, the verifier system (Alice) can append/combine certain unique data to the user selected pulse train. This will ensure that uniqueness of each user's password. However, the system can use memristor to store the salt data associated with each user. It should be noted that one can try to replicate single-user authentication using PRNGs and known seeds. However, memristors are giving a unique advantage of hardware-dependent hashing for key storage in both cases.

False positive alarms happen when an authentic private key is declined. It is more of reliability of the proposed protocols than security breaches. For the protocols discussed in this chapter, any of the generated response of the memristive authenticator is entirely dependent on the choice of bias-voltage and the hardware used. Therefore, the reliability of the protocols will be dependent on the cycle accurate performance of the authentication unit over time. Here, we have analyzed the common causes for which an authentication unit will fail to produce the correct response. It should be noted that these weaknesses are mostly due to the imperfect operating conditions of memristor and by addressing these issues, one can develop reliable authentication mechanisms for resistive memory platforms. Primary con-

cerns for the reliability of the proposed protocols are listed below:

1. Unbalanced Set-Reset: The resistive levels of the memristor devices can change due to unbalanced set-reset pulses over a long period of programming cycles, and hence, one of the major sources of error in the authentication protocols can be an unbalanced set-reset cycle. If the RESET pulse pulls up the resistance more than the previous SET pulses, then the overall resistance of the HRS will drift away with time. An example of this effect is illustrated in Fig.3.7 where we show that due to unbalanced reset, the resistance of the memristor drifts to HRS over multiple cycles. For proper operation of the authentication mechanism shown in Fig 6.after authentication Alice still memristor to apply two consecutive SET pulses for overcoming SET-RESET unbalance problem.

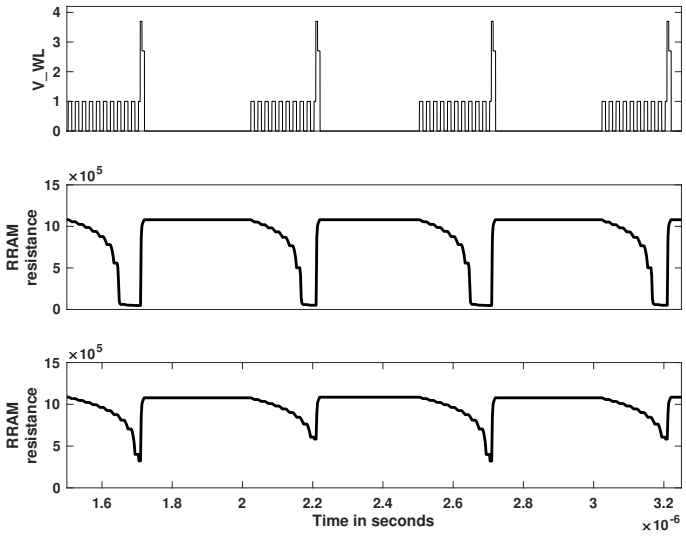

Figure 3.7: Example of SET/RESET unbalance for the circuit given in Fig. 3.4. The top plot shows the typical pulsed programming for a memristive device. The second figure (middle one) lists the changes in a memristor resistance for a stable SET/RESET condition. The plot at the bottom shows the effect of unbalanced biasing where the SET voltage is lowered by 12mV than the previous balanced condition and the RESET voltage is kept the same as before.

To examine the reliability of the circuit, we define a parameter called SET/RESET mismatch as the following: Let us assume that for a given pure LRS it takes a ν_0 volts pulse with τns duration to RESET it to a pure/defined HRS. We define τ as the ideal reset time. Now, a mismatch can be quantified as the difference between the ideal reset time (τ) and the actual reset time τ' while the bias ν_0 remains constant. We can find out the reliability of a circuit (in our case the authenticator unit) by measuring the number of reliable operations it performs before erroneous

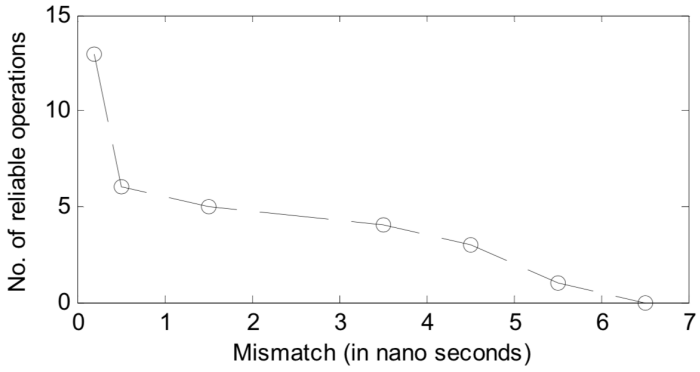

Figure 3.8: SET/RESET mismatch (in ns) vs number of reliable cycles for fast(15 ns) RESET pulse

bit flip is seen at the output (V_{out}) due to SET/RESET mismatch. In Figure 3.8, we have shown the number of reliable operation that can be performed with the authenticator unit as the SET/RESET mismatch increases.

To fight error from unbalanced SET/RESET, the authenticator can perform a full SET/RESET operation after every authentication and RESET operation and verify whether the total number of pulses for a SET remains constant. Another error correction technique is, if the authenticator sees the loss/gain of 1 pulse for SET operation after n' operation, then a correction factor 1 can be subtracted/added to the preconditioning or the hashed value. However, after a bit flip is observed, the best solution would be to recalibrate the device using a reference resistance. It can be seen that with fast reset the number of reliable operations before reset correction is small. Therefore, slower reset with longer pulses and a lower bias voltage is preferable to the protocols described in this work.

2. Temperature dependence of filament formation: Experimental evi-

dence report that the conductive filament formation in a given device is dependent on the device temperature[40]. This can also be seen from equations 2.4 and 2.5. A device that goes through balanced SET/RESET is more reliable with the variance in temperature than the one having an unbalanced operation. Therefore maintaining proper operating temperature for the circuit with balanced SET/RESET will be very critical for the reliable operation of the proposed protocols. Changes in the operating temperature will create unreliable operation similar to unbalanced SET-RESET. Therefore, techniques such as a full SET/RESET operation after every authentication, RESET and verify before authentication, and simple error correction as discussed at the previous paragraph *etc.* will be helpful in addressing issues arising from temperature variation.

3. Spatial variations in filament formation and random filament formation: During SET/RESET process lattice temperature of the filament increases abruptly which gives rise to random non-uniform filament generation. This non-uniformness of filament generation will introduce random fluctuations resistive values during the intermediate states. Overall this would result in cycle-to-cycle write time variability in the device. To reduce this cycle-to-cycle write-time variability, one can use the same techniques discussed for unbiased SET/ RESET case.

4. Noise: The primary feature of our proposed designs are to use the bias dependent write-time variability. However, noise in the writing signal can cause cycle-to-cycle variations in the number of pulses required for performing a given write. Noise on the SET or RESET line can profoundly impact the reliability of the circuit, therefore; care should be taken to remove the bit-line noise.

5. Aging: A proper aging model of memristors is still elusive. However, with aging, the response of the circuit would certainly change, and correct aging model and error correction mechanism are required to combat this phenomenon.

Here we point out that environment variations such as temperature, noise, and device aging may all create false positive alarms. We do not expect this to be a serious problem because we do authentication at the cycle-accurate level, not do the matching in the analog domain. Furthermore, like most of the biometric authentication schemes (such as fingerprint or face recognition), it is possible to authenticate the key with a non-perfect match. That is, we do not need to match every pulse in the pulse train to authentic the key.

We have also fabricated the memristors for experimenting with the ideas presented in this chapter. The details of the fabricated devices are given in chapter 6. Figure 6.2 provides the physical view along with the dimensions of the memristors and Figure 6.3 provides the I-V characteristics of the fabricated memristor.

In summary, when we assume that an outside attacker is only allowed to provide pulse train as the key for authentication and can measure the state changes at a fixed time during the cycles, the attacker can gain very little information, and our user authentication protocols will be stable and secure. However, if the attacker gains physical access to the memristors that we use to store user's private key-related information, there will be a good chance for malicious attacks such as denial of service.

3.10 Conclusions

Non-volatile memory based devices and circuits are monotonic. Exploiting this monotonicity can be useful in designing secure circuits and security protocols. In this work, we have connected the additive and monotonic nature of memristor devices with secret sharing based user authentication ideas. We have reported the designed protocol and the necessary circuits required for single and multiple user authentications using memristor based hardware. These robust, hardware dependent user authentication schemes can be useful in developing security primitives in the extremely resource-constrained IoT applications.

Chapter 4: Authentication Using Voltage Over-scaling

4.1 Introduction

The design of digital circuits and systems are focused on the deterministic results, i.e., for the same inputs, any design will yield the same output. As we push the boundaries of power efficiency, we are introducing the effects of analog nature of the circuit components involved in the computation. In the field of approximate computing, one of the common power reduction techniques is voltage over-scaling (VOS). In VOS, the digital circuit used for computation is operated under the nominal voltage, which guarantees correct output for all input conditions under any given operating environment. Since the dynamic power consumed in a VLSI chip is squarely proportional, and static power is proportional the operating voltage, reducing the operating voltage under the prescribed margin can result in considerable power savings. However, the effect of this voltage over-scaling will be translated into errors generated during a computation. In this chapter, we examine how to use voltage over-scaling to exacerbate the effects of process variation and extract information regarding this variation that can be used for security purposes.

The physical variations have already been widely utilized for security applications in the form of physically unclonable function (PUF). However, PUF is costly

regarding hardware and power consumption, bringing an obstacle for its widespread usage in IoT. By contrast, our proposed lightweight authentication protocol does not require any additional hardware in the low power Things. In this chapter, our main contributions are as follows:

1. We demonstrate that voltage over-scaling based computing leaves a process variation dependent device signature in the approximate results.

2. Binding this process variation dependent device signature with basic key based authentication schemes; we propose a simple authentication mechanism VOLtA: Voltage Over-scaling Based Lightweight Authentication, and perform a security analysis of is protocols.

3. Our work shows that approximate computing can impact security, especially identification and de-anonymization of users and devices.

4.2 Voltage over-scaling: Backgrounds

Variations in the manufacturing process, supply voltage and temperature (PVT) have a major impact on the performance and reliability of a computation performed by an integrated circuit. Fluctuations in PVT affects the time required for a gate to switch to the correct state and thus creates timing errors in the output result. The general trend of digital design is to consider all the corner cases and optimize the design in a way so that fluctuations in PVT have no (or minimal) effect on the output under normal operating conditions.

As the size of the transistor reduces, the effect of process variation becomes a critical issue in digital design. These variations come from the collective factors such as imperfection of the manufacturing process, random dopant fluctuation, and variation in the gate oxide thickness, etc. As the transistor size shrinks, the standard deviation of threshold voltage variation increases, since it is proportional to the square root of the device area as given by [54]

$$\sigma_{\Delta V_t} = A_{\Delta V_t}/\sqrt{WL} \tag{4.1}$$

where $A_{\Delta V_t}$ is characterizing matching parameter for any given process. This variation in V_t will have a direct consequence in the delay of a CMOS gate which can be approximated using the following equation [54]

$$d_{gate} \propto \frac{V_{DD}}{\beta(V_{DD} - V_t)^{\alpha}} \tag{4.2}$$

where α and β are fitting parameters for a gate and the given process. Therefore, to maintain the timing correctness, static timing analysis of a given design is performed on the process corners. The timing analysis ensures that for a given operating condition, all paths meet the timing requirements to produce correct results irrespective of the input vectors. Scaling V_{DD} from the predefined operating voltage creates large timing errors and degrades the output quality. However, V_{DD} scaling can offer great saving is energy budget, and therefore, voltage over-scaling-based approximate computation received significant research attention in the last decade[55, 56, 57].

From equation 4.2 it is evident that with voltage over-scaling and transistor

shrinking, underlying device signature due to process variation manifests itself more prominently in the delay output. If proper correction mechanism is not applied, this variation will cause errors in the output. If V_{DD} and other operating conditions remain fixed, the error generated by a computational unit due to VOS will retain information about the process variation. Since process variation is a random process, by profiling this error, one can distinguish the computational unit and generate a unique device signature for the circuit.

4.3 Errors in an Approximated Circuit

To understand the effect of process variation in voltage over-scaling, we have studied the error profiles generated by adders. Venkatesan et al. have provided process variation independent error profiles for Ripple Carry Adders (RCA), Carry Look-ahead Adders (CLA) and Han-Carlson Adder (HCA)[57]. It was found that the error probability increases as the number of critical paths that fail to meet the timing constraint increase[57]. Therefore, with the presence of randomness in the manufacturing process, the variations in the transistors in the critical paths will have a significant contribution to the errors produced by the adders.

Among the adders, it was found that Han-Carlson adder fares the poorest regarding producing correct result with voltage over-scaling. RCA performs better than HCA and the probability of error increases slowly with the length of the adder [57]. CLA performs best among these three adders. If one wants to extract fabrication variation related information, she needs to be careful on the choice of circuits.

For example, HCA can suppress the variation dependent errors with the errors due to scaling effects, and CLA can repress impact of process variation due to its timing forgiving nature. Therefore, for our discussions, we have used RCA that has the potential to preserve process variation related artifacts.

4.3.1 Error Modeling

If a given circuit is operated with a clock period that is less than the maximum delay produced by the circuit, then the circuit output becomes a function of current and previous input values [57]. Therefore, the output data in a circuit under VOS not only is a function of process variations but also a function of the input values applied to the circuit. Hence, for a combinational adder with two operands $\mathbf{x}$ and $\mathbf{y}$, we can write the current output $\mathbf{z_i}$ of a voltage over-scaled adder as a function of current inputs $\mathbf{x_i}$, $\mathbf{y_i}$, and previous inputs $\mathbf{x_{i-1}}, \mathbf{y_{i-1}}$. Therefore,

$$\mathbf{z_i} = f(\mathbf{x_i}, \mathbf{y_i}, \mathbf{x_{i-1}}, \mathbf{y_{i-1}}) \tag{4.3}$$

where $f()$ defines a process variation dependent addition. This dependence on previous inputs can cause cascading errors in the output. Therefore, to correctly predict the output of a voltage over-scaled adder, one (say Alice) can take the following measures.

1. Save the output data for the set of input patterns that will be used on that circuit. For example, if an only a set S_I containing n-input pattern will be used for processing in a given adder, then Alice needs to save outputs for all

the combination of the pattern resulting from S_I. This would reveal the partial behavior of the circuit for a subset of input data.

2. Profile the adder for all possible input patterns. For profiling the adder, one not only needs to consider all possible current input values but also requires previous input values. Therefore, a correct profile of an n-bit voltage over-scaled adder would consist a table of entries comprised all possible current input value times all possible input values in the previous step. This would amount to $2^{2n} \times 2^{2n} = 2^{4n}$ entries of the input values and the corresponding output values. Since all the additions are not incorrect and dependent on the previous values, one can significantly reduce the size of the profile by strong only the cases where the adders provide inaccurate results.

3. Use a delay-based graphical model to learn the properties of the adder. Since Alice has the adder, she can profile the device, and use this profile to create a conditional probability table for a Bayesian network.

   Our construct of using process dependent delay information for generating responses to random challenges requires the verifier have access to the exact model of the device. Given the access to a large number of input-output of the adder, a verifier can build a model of the adder using Probably Approximately Correct (PAC) learning model in polynomial time. We can prove that such constructs are properly learn-able using the proof presented by Ganji et al. [58].

4.3.2 Assumptions

From the discussions above, we are stating the following requirements to design authentication protocols using a voltage over-scaling based VLSI system. The computing unit (i.e., the adder) must fulfill theses requirements for ensuring a secure authentication.

R1 Voltage over-scaling can produce process variation dependent errors in the computing unit.

R2 Errors produced due to voltage over-scaling are not random noise but reproducible information since they merely reveal the output of the circuit at a lower operating voltage.

R3 If one has access to the input and output ports of this circuit, he can adequately model the behavior the computation performed by this unit.

R4 Such model discussed in R3 would be unique for each computational unit since the manufacturing-dependent process variations are random in nature

.

4.4 Experimental Demonstration

To understand the effects of voltage-over-scaling on processed data, we have simulated simple image processing example using RCA to analyze that effect of process variations, voltage variations, and temperature. We have performed our simulations

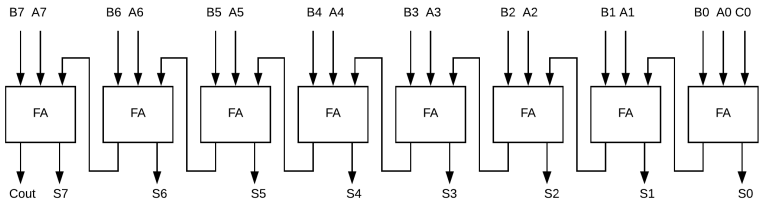

Figure 4.1: 8-bit-ripple carry adder.

in HSpice platform using the FreePDK 45nm libraries[59]. To introduce process variation in our design, we have used 200 modified NMOS and PMOS models with variable threshold voltages. Gaussian distribution with a $\pm 7.5\%$ standard variation is assumed for the variation of the threshold voltages. This modified NMOS and PMOS transistor models are randomly chosen to build 100 different versions of each standard cell in the FreePDK 45nm library. We have designed our digital circuits in Verilog and synthesize them using the Cadence Virtuoso RT compiler. The synthesized design is then converted into an HSpice netlist with standard cells randomly chosen from our modified library.

We present a simple image processing application based on the superimposition of two images under general operating conditions and compare their results. The image processing application superimposition reads the 8-bit values stored at every pixel location for any two given image and add the values. The processing is first done on an accurate adder and then on two voltage over-scaled ripple-carry adders (as shown in 4.1) with process variations.

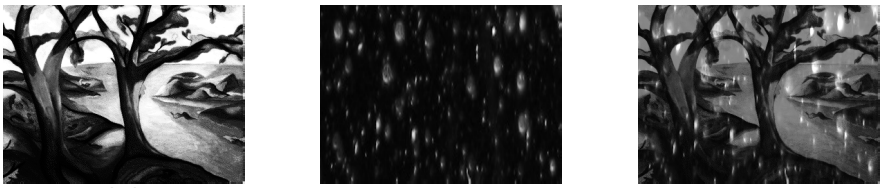

Figure 4.2: An example of superimposing two images. We have used two grayscale images (a) trees and (b) snowflakes from MATLAB library to generate the superimposed image (c) Snowfall.

From visual observation of Figure 4.3(a), 4.3(b) and 4.3(c), one can clearly notice the effect of voltage over-scaling in this simple image processing application. Figure 4.3(b) and 4.3(c) are somewhat distinguishable if an observer pays close attention. We generate the error patterns by calculating the difference between each pixel value in the approximated result and the correct result. Although it is difficult to distinguish the difference between the error patterns shown in Figure 4.3(d) and 4.3(e), if we plot their differences as done in Figure 4.3(f), one can notice the effect of process variation in the approximately computed result.

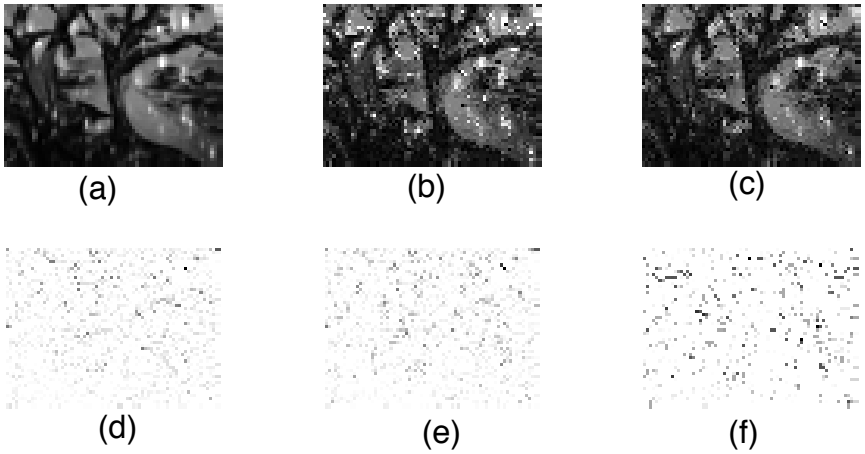

Figure 4.3: An example of the effect of process variations in voltage over-scaling based computation. In (a) the grayscale image Snowfall is computed using trees and snowflakes without voltage over-scaling using $v_{dd} = 1V$; in (b) and (c) the image is computed under voltage over-scaling using $v_{dd} = 0.4V$ with two adders which are identical in every aspect, except the process variation of the transistors; (d) and (e) shows the error pattern found in the figure (b) and (c). This error pattern shows the deviations for each adder from the correct image. Subfigure (f) shows the difference between the two error pattern (d) and (e). The source images were downsized to 52×40 pixels for reducing computation time.

If we plot the histogram of the Euclidean distances between the Figures 4.3(a), 4.3(b) and 4.3(a),4.3(c) (as done in Figure 4.4) then we can see some interesting results on the error pattern. It can be noted that the peaks of this histograms mainly appears at 2,4,8,16,32,64, which suggests that most of these errors in the approximated results come from a single bit-errors. Furthermore, the peaks at 1,2,4,8,16 are higher in most cases, revealing the fact that for these two adders most

of the errors are in the LSBs.

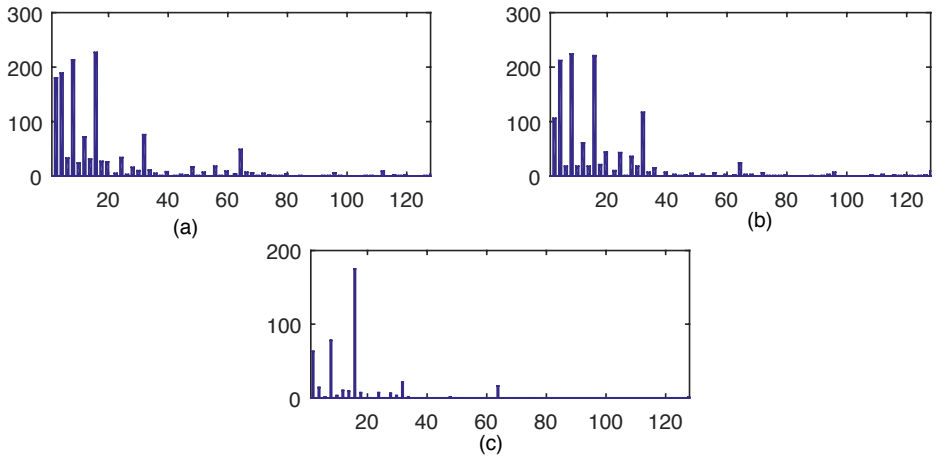

Figure 4.4: A histogram of the Euclidian distances (a) between the figures 4.3(a) and 4.3(b); (b) between the figures 4.3(a) and 4.3(c); between the figures 4.3(b) and 4.3(c). The x-axis in the sub-figures represents bins of all the possible distance values from 1-128, and the y-axis in the sub-figures represents the number of pixels (in the two corresponding computation of Snowfall images) that share the same distance values.

4.5 Simple Authentication Protocol Using Voltage Overscaling

4.5.1 Notations

In this chapter, the set of integers modulo an integer $q \geq 1$ is denoted by $\mathbb{Z}_q$. Matrices, vectors single elements over $\mathbb{Z}_q$ are represented by consecutively upper case bold letter, lower case bold letters and lower case letters such as $\mathbf{X}$, $\mathbf{x}$ and x. For a vector $\mathbf{x}$, the length of the vector is denoted by $|\mathbf{x}|$, i^{th} element is represented by $\mathbf{x}[i]$ and $wt(\mathbf{x})$ denotes the Hamming weight (i.e., the number of indices for which $\mathbf{x}[i] \neq 0$

) of the vector $\mathbf{x}$. The Hamming distance between two binary matrices are denoted by $hd(\mathbf{A}, \mathbf{B})$ (i.e., the number of indices for which $\mathbf{A}[i][j] \neq \mathbf{B}[i][j]$). Concatenation of two vectors are represented with || symbol. $c \xleftarrow{\$} \{\mathbf{x} \in \mathbb{Z}\}$ represents a random sampling of $\mathbf{x}$. We denote *probabilistic polynomial time* (PPT) algorithms with upper case calligraphic alphabets such as $\mathcal{A}$. Therefore, if $\mathcal{A}$ is probabilistic, then for any input $\mathbf{x} \in \{0,1\}^*$ there exists a polynomial $p(.)$ such that the computation of a terminates in at most $p(|\mathbf{x}|)$ steps.

4.5.2 Description of the Protocol

For single user authentication, we will assume an interactive protocol between a single prover $\mathcal{P}$ and a verifier $\mathcal{V}$. Both the prover and the verifier has some knowledge about a shared secret $\mathbf{x}$. The secret is generate through a key-generation procedure $KeyGenVOLtA(1^\lambda)$,where λ is a security parameter. The authentication protocol responds with the outputs accept or reject after a successful run of the protocol.

Assumptions: Assuming that one has access to hardware that satisfies the requirements R1-R4 presented in section 4.3.2. This authentication protocol assumes that the prover has a voltage over-scaled computation unit (H) that generates process dependent errors. The verifier either knows the correct model or profile (M) to simulate the computation unit.

Algorithm 7 RNG based Key Generation for Voltage Overscaling based authentication

1: **procedure** $(\mathbf{x_1}, \mathbf{x_2}) \leftarrow KeyGenVOLtA(1^\lambda)$

2: Sample $\mathbf{x_1} \xleftarrow{\$} \mathbb{Z}_p^\ell$

3: Sample $\mathbf{x_2} \xleftarrow{\$} \mathbb{Z}_p^{\ell}$

4: **return** $\mathbf{x_1}, \mathbf{x_2}$

Enrollment: The prover and the verifier uses the key generation procedure $KeyGenVOLtA$ to generate secrets $\mathbf{x_1}, \mathbf{x_2}$. ϵ is the predetermined error threshold for the authentication.

Table 4.1: Single round interactive authentication VOLtA

Prover$(\mathbf{x_1}, \mathbf{x_2}, H)$		Verifier$(M, \mathbf{x_1}, \mathbf{x_2}, \epsilon)$
		$\mathbf{R} \xleftarrow{\$} \mathbb{Z}_p^{\ell \times n}$
	$\xleftarrow{\mathbf{R}}$	
Calculate $\mathbf{L} = \mathbf{H}(\mathbf{R}, \mathbf{x_1}) = \mathbf{R} + \mathbf{x_1}$ using the adder and then calculate $\mathbf{z} = \mathbf{L} \oplus \mathbf{x_2} = (\mathbf{R} + \mathbf{x_1}) \oplus \mathbf{x_2}$		
	$\xrightarrow{\mathbf{z}}$	
		Calculate $\mathbf{z}' = M(\mathbf{R}, \mathbf{x_1}) \oplus x_2$. If distance $(\mathbf{z}', \mathbf{z}) \leq \epsilon$ accept.

Note that, the distance in the protocol can be measured by standard distance measurement functions such as Hamming distance or Euclidean distance. Also, with multiple keys, the verifier can authenticate numerous users using the same device. Moreover, the verifier can verify the prover over different devices are given that verifier knows the correct model of those devices.

4.6 Evaluation of the Protocol

For our discussions on threat models and attacks, we assume that Alice tries to authenticate Bob over an untrusted channel where Malice performs the following attacks to obtain the security keys or being erroneously recognized as Bob. Below we discuss the potential attacks on VOLtA.

This is a simple two-factor authentication scheme, which requires knowledge of both the secret was known (i.e., key $\mathbf{x_1}, \mathbf{x_2}$) and the secret possessed (i.e., properties of the voltage over-scaled adder). To prove the effectiveness of our proposed VOLtA, we start from analyzing the potential weakness, for the case when we have a perfect adder. If the adder is perfect when calculating $\mathbf{L}$, this protocol is not secure. Assume the following scenario: the malicious attacker Malice is pretending to be Alice, and she wants to resolve Bobs key $\mathbf{x_1}, \mathbf{x_2}$ by sending some messages $\mathbf{R}$ and receiving the corresponding $\mathbf{z}$ from Bob. Then she will apply eavesdropping and bit manipulation techniques to recover the key.

However, when applying the voltage over-scaling approach, the addition will become non-deterministic because the physical variations will affect the arithmetic result as discussed in section 4.3. Therefore, the result of $\mathbf{M}(\mathbf{R}, \mathbf{x_1})$ cannot be accurately predicted. As a result, the bitwise attacking scenario fails. Overall, for the security of this protocol, the uncertainty of the calculation in the arithmetic function needs to be guaranteed.

4.6.1 Random Guessing Attack

The most straightforward attack that an adversary can perform is on the authentication protocol is to try and randomly guess the authentication keys. To accomplish this attack, Malice tries to imitate Bob and responds to Alices query with a random guess $\mathbf{z}'$.

The security of VOLtA is tied not only to the key $\mathbf{x_1}, \mathbf{x_2}$ but also the property of the approximate adder. Since Malice neither has information about the key nor about the hardware properties, the success rate of such attacks exponentially decreases with the increase in the number of bits in the security keys and with the increase in the uncertainty of the results produced by the adder.

4.6.2 Eavesdropping Attack

For eavesdropping attack, Malice eavesdrop on some communications between Alice and Bob and records Bobs response to each challenge from Alice. Later, for a known query of Alice, Malice can answer correctly using her records.

Alice sending random string $\mathbf{R}$ each time can easily thwart such attacks because in that scenario response calculated by Bob will be different for different cases. Since Alice knows the correct model of Bobs circuit, she can send random string every time for authentication. Therefore, VOLtA would be useful in counteracting such attacks.

4.6.3 Man-in-the-middle attack

Man-in-the-middle attack constitutes the case where Malice pretends as Alice and communicates with Bob. Malice sends random authentication strings to Bob and collects his response. This attack would be difficult to perform if Bob has some knowledge about the input sent by Alice. But this would violate the requirement of randomized string to prevent other attacks. However, our authentication mechanism will succumb to a MITM attack that queries Bob with a correct challenge, learn it and give Alice the response.

4.6.4 Compromised key

One of the most active attacks on these protocols is the situation where the key $\mathbf{x_1}, \mathbf{x_2}$ and the model M is leaked to an attacker. Since this case breaks the fundamental Kerckhoffs's principle, both the protocols will fail in the face of such attacks.

Therefore, encryption techniques need to be applied to protect Alices database to ensure the security of the keys and models. It should be noted that this would provide security in the case when the key K is compromised because it is a two-factor authentication protocol where the property of the adder is unknown to the attacker. Therefore, without having the device of the model of the device, the attacker would still have to resort to random guessing or other attacks to resolve a correct response.

4.6.5 Learning-based attack

This attack is a combination of eavesdropping attack and learning attacks. Malice eavesdrop on the communication during authentication and with the challenge and response records, Malice models the voltage over-scaled approximate adder using a learning model. Thus, Malice can create a delay based graphical model for the adder with partial observables. Malice can estimate a conditional probability table, and the chance of success in getting the model for the adder will increase the number of trials that Malice can perform. However, the output of the adder is XORed with $\mathbf{x_2}$, and therefore such attacks will be difficult to perform without the knowledge about $\mathbf{x_2}$.

4.6.6 Side-channel attack

One of the most common side channel attacking techniques for encryption is static or differential power analysis. Researchers have shown that the power analysis can severely reduce the security of Ring Oscillator PUF, which is a well-known secure primitive for key storage and authentication. However, our proposed voltage over-scaling is more resistant to side channel attack because the arithmetic units are working under much lower voltage. As we mentioned before, the dynamic power consumed in a VLSI chip is squarely proportional to the supply voltage, the power consumption during authentication process will be very low, making it difficult to capture accurate power consumption. Since the adder does not generate correct result, even though the attacker can measure the exact power consumption, he

cannot apply the model of an accurate adder for regression. The real model M is hidden in the process variations.

4.7 Experiments and Discussions

We have simulated the basic building blocks of the authentication protocol to analyze that effect of process variations, voltage variations, and temperature, and evaluate the performances of the protocols under general operating conditions. We have performed our simulations in HSpice platform using the FreePDK 45nm libraries. To introduce process variation in our design, we have used 200 modified NMOS and PMOS models with variable threshold voltages. Gaussian distribution with a $\pm 7.5\%$ standard variation is assumed for the variation of the threshold voltages. This modified NMOS and PMOS transistor models are randomly chosen to build 100 different versions of each standard cell in the FreePDK 45nm library. We have designed our digital circuits in Verilog and synthesize them using the Cadence Virtuoso RT compiler. The synthesized design is then converted into an HSpice netlist with standard cells randomly chosen from our modified library.

As our authentication protocol is based on addition performed in a voltage over-scaled system, we have simulated simple 8-bit ripple carry adder for our analysis presented in this work. This choice is justified in section 4.3.2. All the output bits of the adder are fed into edge-triggered D-flip-flop to extract correct output bit values in each clock cycles.

Table 4.2: Parameters used for simulations in this chapter

Parameter Name	Value(s)
Supply voltage (VDD)	0.4V/0.45V/1V
NMOS threshold voltage (Vtn)	$0.322 \pm 0.02415V$
PMOS threshold voltage (Vtp)	$-0.302 \pm 0.02265V$
Operating temperature (T)	25°C
Clock Period (T_{clk})	1ns

4.7.1 Uniqueness & the Effect of Process Variation

In this section, first, we will validate the requirements presented in section 4.3.2. For our experiments, we have used eight adders generated from a process variation-aware 45nm process. Our experiment at VDD=0.4V shows that the results generated from voltage over-scaled adders contain errors. To understand the uniqueness of each approximate adder regarding these errors, we evaluate the variations using the following metrics:

I The pairwise Hamming distance of results between adder i and adder j. We used a 3670-byte random input sequence related to the image processing application discussed later. For each adder, we collect the results on each clock cycle and concatenate all the results to create the complete output bit-stream generated by the adder. Then, we calculate the pairwise Hamming distance in of these output bit-streams. We divide the Hamming distance with the length

of the bit-stream and report the result in percent in Table 4.3.

II The pairwise average numerical difference of adder i and adder j is given by:

$$avg_d = \sum_{l=1}^{N} \frac{|result(i,l) - result(j,l)|}{N} \tag{4.4}$$

where N is the size of the output in bytes. The result is shown in Table 4.4.

Metric I, the Hamming Distance, is widely used to measure the difference between two binary bit-streams. However, it omits the bit orders. For example, the Hamming distance between the pair (00001, 00011) and a pair (00001,10001) are all 1. Therefore, we introduce the second metric to have a measure of the average numerical difference between the values represented by 8-bit outputs of every two adders. Both Table 4.3 and 4.4 are symmetric since for our measures distance(i,j)= distance(j,i).

In both Table 4.3 and 4.4, A1 represents the results from an exact adder. From the first rows of Table 4.3, we can notice that there are about 20% bit flips in the outputs of the voltage over-scaled adders. This provides experimental justification of requirement R1 (discussed in section 4.3.2). From the rows and columns of Table 4.3, it is evident that there is a significant difference in the output of two different voltage over-scaled adders. This justifies R4. Note that R2 is also justifiable since we are not introducing any noise or fault in the device, but we are merely resorting to the analog nature of the device to get information about process variation. Moreover, from table 4.4, it is evident that there is a significant average difference between the numerical results produced from each adder.

Table 4.3: Pairwise hamming distance (in percent)

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	18.82	18.24	18.0	19.4	18.4	18.3	17.52
A2	18.82	0	5.36	5.21	5.67	5.65	3.89	5.39
A3	18.24	5.36	0	4.62	5.98	5.11	5	6.79
A4	18.0	5.21	4.62	0	5.73	3.53	4.13	6.44
A5	19.4	5.67	5.98	5.73	0	6.04	5.59	6.28
A6	18.4	5.65	5.11	3.53	6.04	0	4.96	6.64
A7	18.3	3.89	5	4.13	5.59	4.96	0	5.41
A8	17.52	5.39	6.79	6.44	6.28	6.64	5.41	0

4.7.2 Effect of Variations in Supply Voltage

The proposed authentication protocol is based on the key concept of voltage over-scaling and the errors it produces. Therefore, variations in supply voltage will cause reliability issues for the proposed design. Therefore, care must be taken to ensure voltage supply with a minimal amount of noise for proper implementation of the protocol. In Figure 4.5, we have plotted the response of voltage over-scaled adders at 0.45V. This is a bit higher voltage than the one used for the results reported in Table 4.3. It can be noted that as the voltage increases the overall Hamming distance decrease, which represents the eventual convergence of all the adders to the correct output at sufficiently higher voltage.

Table 4.4: Pairwise average numerical difference between the output from the devices at 0.4V

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	12.01	14.64	16.11	12.30	14.13	12.13	13.25
A2	12.01	0	9.04	12.30	8.03	9.19	8.08	8.97
A3	14.64	9.04	0	8.43	9.03	7.24	9.07	11.13
A4	16.11	12.30	8.43	0	10.68	8.58	8.88	11.13
A5	12.30	8.03	9.03	10.68	0	6.01	6.51	9.45
A6	14.13	9.19	7.24	8.58	6.01	0	8.11	8.89
A7	12.13	8.08	9.07	8.88	6.51	8.11	0	8.47
A8	13.25	8.97	11.13	11.13	9.45	8.89	8.47	0

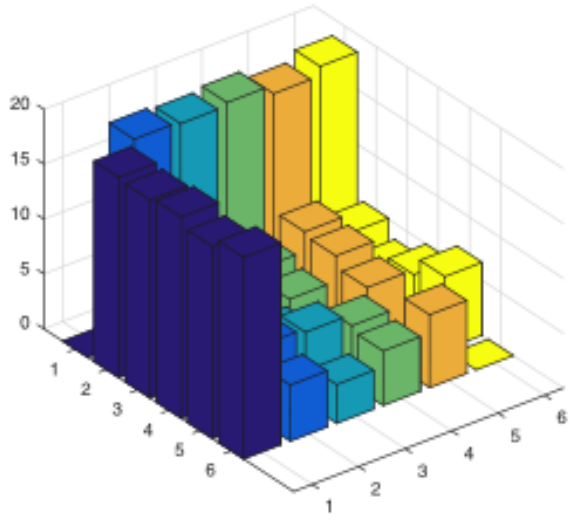

Figure 4.5: Hamming distance (in percent) between devices at 0.45V . Here 1 represents a correct adder (A1) and 2-6 represents voltage over-scaled approximate adders (A2-A6).

4.7.3 Effect of Variations in Temperature

Variations in operating temperature can also affect the protocols. To understand the effect of temperature, we have calculated the percentage Hamming distance between the results of the same adder at different temperatures as shown in Figure 4.6.

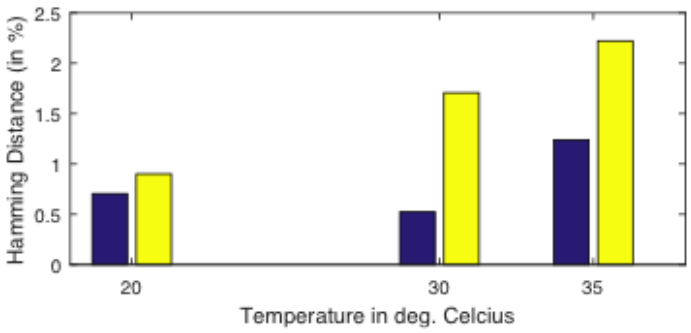

Figure 4.6: Temperature dependent bit flips for two different adders. The distance is calculated from the results produced at T=25°C. The blue (dark) line represents the temperature dependent bit-flip for adder A2 and the yellow (light) line represents the adder A3.

We find that ±5 degree variations result in less than 1% bit-flips. Therefore, by carefully calibrating the threshold of error tolerance ϵ of the protocols, one can negate the effects of minor temperature variations.

4.7.4 Mitigating Reliability Issues

Noise due to environmental variations can cause deviation from the expected result in a voltage over a scaled circuit. However, taking majority votes in multiple reading with the same challenge can reduce error due to environmental variations. Hence in the small error case, suppose that the voltage overscaled device $\mathcal{A}$ produces the expected response x with probability 0.9 over the choice of a challenge c. Then we can amplify this probability to $1/(10n)$ by making $10logn$ repetitions and take the majority vote. In this way, we can correctly recover all of the bits of x with high probability.

Another way of mitigating reliability issues due to environmental variation is to authenticate using multiple challenge-response pairs. Assume that there is an interchip variation τ exists between two given adder. τ is defined to be the probability that for a given random challenge $\mathbf{c}$, the response of the two adders at VOS-limit would be different. Furthermore, let us quantify the noise as μ, where μ represents the probability that for a given challenge the response would be different for the δ change in environmental condition. Then, the probability that at least $2t+1$ out of k reference responses differ between two chips is given by [60]

$$a = 1 - \sum_{i=0}^{2t} \binom{k}{i} \tau^i (1-\tau)^{k-i} \tag{4.5}$$

For a single adder, the probability that at most t out of k responses differ is given by [60] :

$$b = 1 - \sum_{j=t+1}^{k} \binom{k}{j} \mu^j (1-\mu)^{k-j} \tag{4.6}$$

Therefore, the probability of identification of N chips using a set of k challenges is at least [60]

$$p = a^{\binom{N}{2}} b^N \approx (1 - \binom{N}{2}(1-a))(1 - N(1-b)) \tag{4.7}$$

Therefore, we can see that using multiple sets of CRPs for authentication; we can guarantee successful authentication over environmental variations.

4.7.5 Deanonymization Issues

It should be noted that by profiling the input patterns of a voltage over-scaled circuit, one could easily deanonymize the approximate circuit later based on the physical

variations. This exposes a critical flaw in voltage over-scaling-based approximate computing. Therefore, the anonymity of the approximate devices should also be thoroughly studied. This work can be considered a step towards such analysis.

4.8 Conclusions

In this chapter, we introduce VOLtA, a voltage over-scaling based user authentication scheme that uses the random physical variation of a VLSI system as a key for authentication of IoT nodes. VOLtA profiles the hardware used for computation in a reduced voltage operation and uses the underlying hardware fingerprint for authentication purposes. This authentication protocol requires no additional hardware on the claimant side to implement. This lightweight protocol can be useful in IoT applications where the interconnected Things face extreme power, cost and area constraints.

Chapter 5: Signal Authentication and Spoofing Detection Using Hardware Oscillators

5.1 Introduction

The progress towards the Internet of Things (IoT) is highly dependent on the secure and successful integration of a trusted and robust geospatial localization and clock synchronization mechanism for *Things* across a large distributed network. Currently, both these functions are predominantly provided by the Global Positioning System (GPS). As a result, in today's world, GPS receivers have become a ubiquitous component in embedded systems: from the smart-phones and smart devices to automated vehicles and Phasor Measurement Units (PMUs) in the electric grid. However, this pervasive nature of GPS receivers in current and next-generation smart *Things* necessitates a thorough study of their security vulnerabilities and corresponding countermeasures.

Recently, there have been several demonstrations of weaknesses and vulnerabilities of GPS signals and GPS receivers [61, 62, 63, 64, 65]. It has been argued that the future of navigation is crucially dependent on defending spoofing attacks on global navigation satellite system (GNSS) signals such as GPS. Moreover, the

prevailing trend in IoT application designs is to use the civilian GPS signals, these signals are broadcast without encryption, and several practical demonstrations of spoofing mechanisms have been documented in the literature [61, 62, 63, 65]. Most of the analysis on GPS spoofing is directed towards the spoofing of position data. However, GPS system is also used for large area clock synchronization, and therefore, attacks on GPS signals can impact networked infrastructure where accurate timekeeping is essential. For example, Phasor Measurement Units (PMUs) used in the electrical power system synchronize themselves using GPS signals, and an attack on this synchronization can induce failures in the power system across a wide area [66].

Along with research on potential attacks, several authentication and anti-spoofing techniques for GNSS signals have been developed in recent years. These techniques can be broadly categorized as signal and data-level authentication. For signal authentication, received signal characteristics of the civilian signal may be verified against encrypted GPS transmissions. Additionally, the plane-of-polarization and angle-of-arrival can be measured to validate the signal as well. Data-level techniques are based on authentication of the received data with reference to a-priori knowledge of position or authentication using corroborative localization and timing sources. These methods have been shown to be effective; however, they are not usually used in the commercial GPS receivers due to implementation cost[64]. Moreover, the accuracy, computational and power efficiency, security against sophisticated attacks and real-time performance of these approaches are still areas of active research.

In this chapter, we present a data-level authentication mechanism for GNSS signals that rely on intrinsic hardware properties of a free-running crystal oscillator. Since the free-running oscillator is located on the device and not externally synchronized, it presents a minimal attack surface while exhibiting a strong correlation with authentic GPS signals. We propose that '*anomalies*' in the correlation index can authenticate received GPS data. Our approach is simple, fast and can perform in near real-time. Additionally, the design is low cost and can act as an add-on to virtually any GPS receiver.

5.2 Preliminaries

5.2.1 The GPS System

The GPS system consists of satellite transmitters and (usually) terrestrial receivers. Each transmitter satellite broadcasts at two frequencies: 1575.42 MHz (L1) and 1227.6 MHz (L2). The L1 carrier messages are available for civilian purposes. These messages are not encrypted but modulated with pseudo-random noise (PRN) codes to distinguish each satellite. The L2 carrier is modulated by encrypted codes and reserved for military purposes. Message from each GPS satellite contains information about the position of the satellite and the time of the onboard atomic clock[67].

To calculate true receiver-to-satellite distance, the receiver requires the range (r_{true}) of a satellite at a given time. This can be calculated by the multiplying the signal propagation time (from the transmitter to the receiver) with the speed of light (c). Then, for a receiver located in (x_r, y_r, z_r) and a transmitter at (x_t, y_t, z_t)

position, the range is given as:

$$r_{true} = c\ t_{propagation} = \sqrt{(x_t - x_r)^2 + (y_t - y_r)^2 + (z_t - z_r)^2} \tag{5.1}$$

To solve equation 5.1 for (x_r, y_r, z_r), one requires ephemerides for three satellites. However, since the clock on the GPS transmitter (t_{GPS}) and the clock on the receiver (t_{local}) are not perfectly synchronized, there exists an offset t_r between these two time-scales. Therefore, the satellite-to-receiver distance that a receiver perceives is a pseudo-range (r_{pseudo}), where:

$$r_{pseudo} = r_{true} - ct_r \tag{5.2}$$

Therefore, a GPS receiver needs to solve for four unknowns (x_r, y_r, z_r, t_r) for precise location and perfect synchronization. At least four satellite data is required for solving this system of equations. Using this solution, a GPS receiver updates its position, and synchronizes its local clock frequently to t_{sync} for keeping perfect synchronization with the universal coordinated time (UTC) where:

$$t_{sync} = t_{local} + t_r \tag{5.3}$$

In practice, this method provides accuracy in the order of 10 meters in position and nearly $0.1\mu s$ in time [68]. As a result, GPS signals can be used not only for positioning of the receivers but also for precise clock synchronization of receivers across the globe.

5.2.2 GPS Spoofing

Attacks on the GPS signal are usually performed by an adversary by either jamming or spoofing one or both radio channels. For the jamming attack, an adversary transmits overwhelming radio interference over the L1 and/or L2 band. For spoofing attacks, the adversary mimics real GNSS transmissions to intentionally alter data received by a victim receiver. Since civilian GPS signals are not encrypted and the structure of the signal is well known, spoofing attacks are relatively straightforward to execute using a commercial signal generator and RF transmitter.

5.2.3 Existing Spoofing Detection Techniques

As discussed in Section 5.1, signal-level detection methods can take on several forms including (1) check/validate the received RF signal strength against a level threshold, (2) compare L1 and L2 carrier signals, (3) justify the directional characteristics and polarization of the received signal [64]. The strength of a received GPS signal is typically less than -150dBW and presence of substantially stronger signals for a single satellite or over the entire frequency band can be a sign of attack in progress. However, it is possible for a spoofing attacker to adjust power levels to evade detection limiting the usefulness of the threshold detection mechanism in practice[64]. Furthermore, most consumer grade receivers only support a single RF band, and therefore, comparing L1, and L2 carriers would require complete replacement of the internal RF-circuitry. Also, in a case of a replay attack, an attacker can delay both L1 and L2 signals to avoid detection. Finally, using directional characteristics of

the receiver antenna to cross-validate received signals from each satellite requires specialized phase tracking hardware to detect directional variations. Similarly, signal polarization characteristics have been shown to be an effective authentication aid [62]. However, dedicated receiver front ends and signal processing is required to implement the approach efficiently.

Data-level spoofing detection uses demodulated GPS data to detect spoofing. GPS data can be validated with known position data or otherwise obtained time to detect attacks[69]. For example, a stationary receiver can check its known position with the position-solution of the receiver. Since trilateration position error can be ± 10 meters, the position solution is a weak measure of credibility. It has been recently demonstrated by Jiang *et al.* [66] that attackers can use this uncertainty to induce a phase angle error of 52 degrees in a PMU receiver by using simple optimization based evasion algorithms. Monitoring jumps in the time reported by the GPS signals is another possible GPS spoofing countermeasure. One can deploy accurate clocks to measure time deviation between the reported GPS clock and the onboard clock. However, precise clocks (such as atomic clocks *etc.*) are expensive and not used in practice for commercial purposes. For IoT components, one can also compare the GPS time with networked time protocols such as NTP. However, the approach can suffer when the network is down. Moreover, the resolution of attack detection is limited by the accuracy of the networked clocks.

While it is likely that signal threshold based authentication will be integrated into commercial GPS receivers used for critical applications, the solutions discussed in this section are currently cost prohibitive for consumer grade GPS receivers in

the nascent IoT infrastructure. Hence, we propose a low-cost, computation and power-budget friendly data-level GPS spoofing detection mechanism in Section 5.3.

5.3 Approach

The key idea of this work is to authenticate GPS time signals using intrinsic properties of a hardware oscillator. This would essentially construct a spoofing detector for the signal. This spoofing detector will calculate frequency states of this hardware clock using the received GPS signal as a reference. Any attacks on the received GPS data will create anomalies in the internal frequency states of this clock. Once an attack is detected, the design will attempt to generate an approximated version of the correct GPS time t_{GPS} to holdover the timing system during an attack. This design would require two additional resources in addition to the GPS receiver:

1. a single (or multiple) free running oscillator(s),

2. additional data processing capabilities.

The architecture of the authentication system is shown in Figure 5.1.

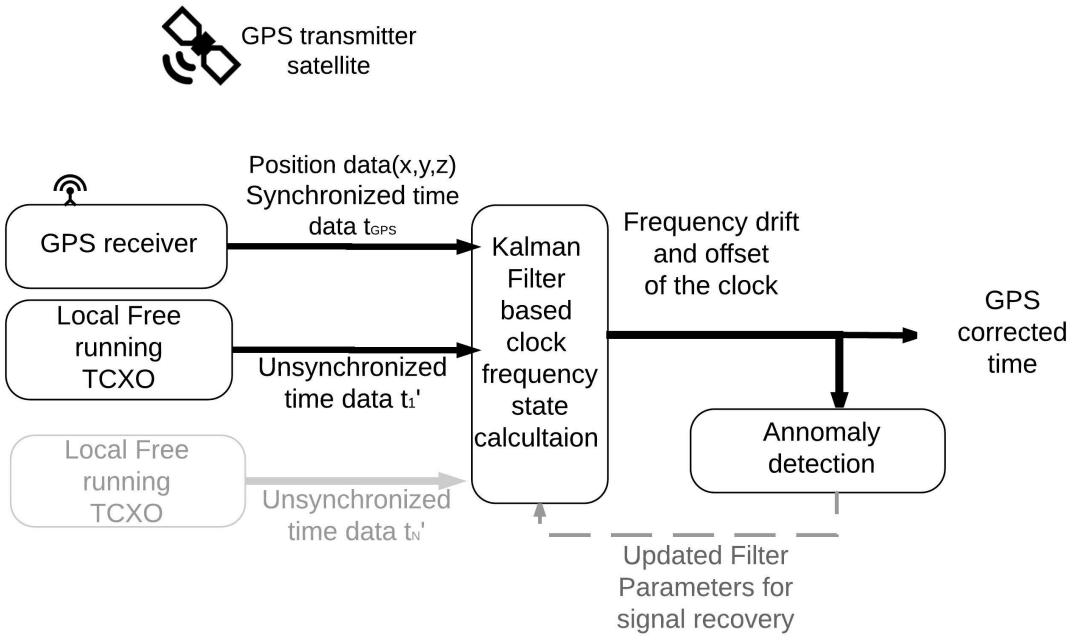

Figure 5.1: Design of the secure GPS receiver with onboard authentication and spoofing detection mechanism. The receiver is equipped with a single (or multiple) free running temperature controlled oscillator(s). Kalman filter-based state estimation is used for calculating frequency states of this clock. In the case of multiple clocks, time from each free-running oscillator can be used for generating a low noise stable virtual clock. Anomalies in the frequency drift and offset of this single clock (or the low noise virtual clock) calculated with respect to the GPS signal can reveal spoofing attacks on GPS signal. Updated filter parameters can help to reconstruct the approximated *true* time-offset during a spoofing attack.

Our approach depends on the internal frequency states (*i.e.,* frequency drifts and skew) of a hardware oscillator for spoofing detection. Hence, we provide details on hardware clock models in the next section to elucidate this approach.

5.3.1 Hardware Clocks

Clocks and oscillators in embedded systems are primarily used for time-keeping and synchronization purposes. In the majority of the embedded systems, crystal-based real-time clocks (RTCs) are used for precise time-keeping. These RTCs are far from perfect, and they deviate from ideal time due to both systematic and random variations. These systematic variations arise from the imperfections in the physical realization of the clock, and they are observed as time and frequency offsets and frequency drift.

For avoiding ambiguity in the rest of the chapter, we reiterate the clock related definitions from Moon *et al.* [70].

Clock: *A clock is a piecewise continuous function that is twice differentiable. If* $x(t)$ *is the time reported by a clock at time* t*, there exists* $x'(t) = dx(t)/dt$ *and* $x''(t) = d^2x(t)/dt^2$ *for* $t \geq 0$*.*

Offset: *Offset is the difference between the time reported by two clocks. If time at two clocks* x_1 *and* x_2 *are* $x_1(t)$ *and* $x_2(t)$ *respectively, then the offset of the clock* x_1 *relative to clock* x_2 *at time* $t \geq 0$ *is* $x_1(t) - x_2(t)$*.*

Frequency: *The rate at which the clock progresses. The frequency of* x_1 *at time* t *is* $x_1'(t)$*.*

Skew*: Clock skew is the difference between the clock frequencies of two clocks. The skew of* x_1 *relative to* x_2 *at time* t *is* $y = x_1'(t) - x_2'(t)$*.*

Drift: *The drift of clock* x_1 *relative to the clock* x_2 *at time* t *is* $x_1''(t) - x_2''(t)$*.*

At a given time t, the deviation of a clock from ideal time can be expressed as [71]:

$$x(t) = x_0 + y_0 t + \frac{1}{2} D t^2 + \epsilon(t) \tag{5.4}$$

where x_0, y_0 and D represents the time offset, frequency offset (also known as skew) and frequency drift. $\epsilon(t)$ represents non-deterministic random deviations. The frequency offsets and drifts of an RTC arise from the microscopic variations in the crystal used in these oscillators. The frequency offsets and drifts also vary with the dissimilarity in design, power supply, and environmental factors. These properties have been found to be different for similar oscillators working in a same operating condition. Therefore, we have the following assumptions regarding the frequency states of hardware clocks:

A1. Frequency drifts and skew of a clock with respect to a more precise reference are nearly constant and unique for a clock reference pair for a given duration.

A2. The states of a known free-running local oscillator, are predictable for a given reference, and one can detect unusual activity in the reference by looking at the properties of the local oscillator.

A3. These properties vary uniquely for different clock pairs due to the random variations in their fabrication and are impossible to recreate without tampering the hardware of both the clocks.

Based on our assumption, the GPS induced internal states of a given free running oscillator are relatively constant, and therefore, can be used to detect spoofing

attacks. This is the key concept for our approach. It should be noted that Khono *et al.* [72] first proposed the idea of remote device fingerprinting using the uniqueness of frequency offset of hardware clocks. Since the publication of Khono's work [72], there has been a significant development in this field of remote device fingerprinting using hardware oscillators. Our assumptions can be validated by Khono's work, the subsequent works in the literature and our experimental results and analysis presented in this work.

5.3.2 State-Space Model of Hardware Clocks

For precisely calculating hardware clock states, we use a stochastic model of the clocks where a clock-state is characterized by a column vector $x(t) = [x_1(t)\ y_1(t)\ D_1(t)]^T$. Here, $x_1(t), y_1(t)$ and $D_1(t)$ represents the time offset state, frequency offset state and frequency drift state respectively. The clock state follows the stochastic difference equations as given in [73]:

$$\frac{dx_1}{dt} = y_1 + w_1; \frac{dy_1}{dt} = D_1 + w_2; \frac{dD_1}{dt} = w_3; \tag{5.5}$$

where, $w_i(t)$ represents the associated zero mean white noise with spectral densities q_i. For an ensemble of q clocks the state vector can be written as $[x_1, y_1, D_1 \ldots x_q, y_q, D_q]^T$. Discrete-time equations for a system described by 5.5 can be written as [73]:

$$\mathbf{X_n} = \mathbf{F_n X_{n-1}} + \mathbf{W_n} \tag{5.6}$$

$$\mathbf{\Xi_n} = \mathbf{H_n X_n} + \mathbf{V_n} \tag{5.7}$$

where, $n = 0, 1, 2, \ldots$ corresponds to discrete time t_n and measuring time interval $\Delta = t_n - t_{n-1}$.

For a single clock measurement, the $\mathbf{X_n} = [x_1, y_1, D_1]^T$ represents the state vector, and $\mathbf{\Xi_n}$ denotes the observation vector. $\mathbf{F_n}$ is the state transition matrix which is calculated as [73]:

$$\mathbf{F_n} = \begin{bmatrix} 1 & \Delta & \Delta^2/2 \\ 0 & 1 & \Delta \\ 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

The process noise $\mathbf{W_n}$ is considered to be zero mean additive white noise with covariance matrix $\mathbf{Q}$, where

$$\mathbf{Q} = \begin{bmatrix} q_1\Delta + q_2\frac{\Delta^3}{3} + q_3\frac{\Delta^5}{20} & q_2\frac{\Delta^2}{2} + q_3\frac{\Delta^4}{8} & q_3\frac{\Delta^3}{6} \\ q_2\frac{\Delta^2}{2} + q_3\frac{\Delta^4}{8} & q_2\Delta + q_3\frac{\Delta^3}{3} & q_3\frac{\Delta^2}{2} \\ q_3\frac{\Delta^3}{6} & q_3\frac{\Delta^2}{2} & q_3\Delta \end{bmatrix} \tag{5.9}$$

This state model for a clock ensemble is amenable to the design of optimal stochastic filters, which are broadly used for minimizing variance within a clock ensemble. In this work, we use this state space model for hardware oscillators and use an optimal filter (Kalman Formulation) to estimate these states for a single oscillator[73]. It should be noted that if there are more than one on-board free running oscillators available, one could create a virtual time reference using an ensemble of clocks offering improved detection thresholds for spoofing attacks.

5.3.3 Kalman Filter Design for Authentication and Spoofing Detection

We use discrete time state-model for developing a Kalman filter based spoofing detector. For our single clock experiment, we have the measurement matrix $\mathbf{H_n} =$

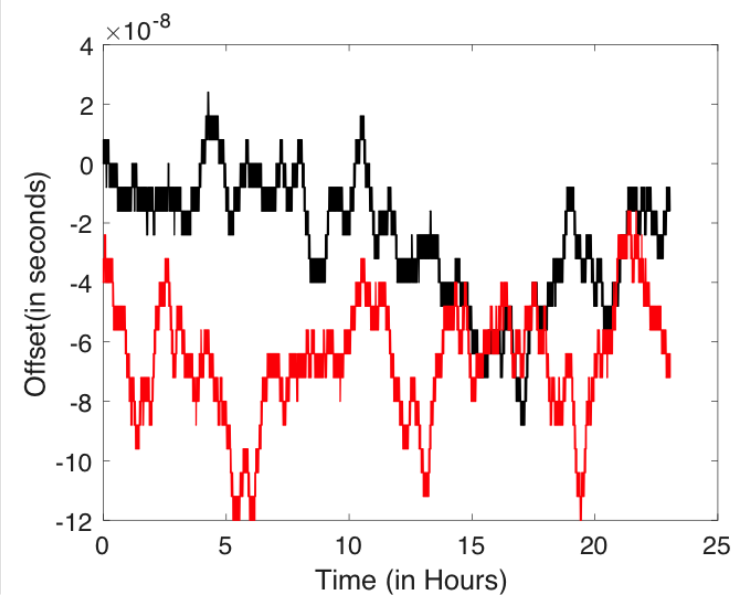

Figure 5.2: Clock offset between two GPS clock measured with respect to a precise rubidium clock.

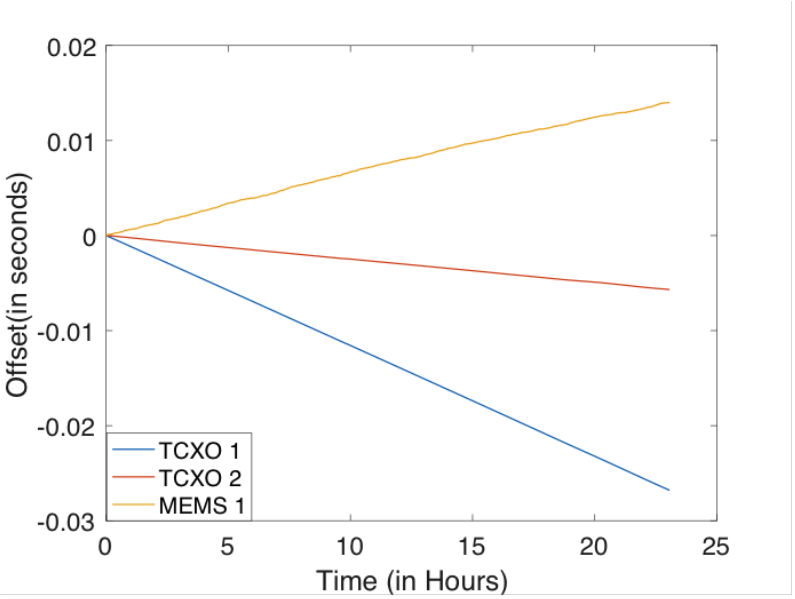

Figure 5.3: Clock offset between two TCXO and a MEMS clock measured with respect to a precise rubidium clock.

$[1, 0, 0]$. $\mathbf{V_n}$ represents the zero mean measurement noise with covariance $\mathbf{R}$. For local measurements, we set $\mathbf{R} = \mathbf{0}$ to neglect the noise term. The algorithm for this linear Kalman filter [74] is given by the following equations:

Prediction Step:

$$\mathbf{m_{n|n-1}} = \mathbf{F_n m_{n-1|n-1}} \tag{5.10}$$

$$\mathbf{P_{n|n-1}} = \mathbf{F_n P_{n-1|n-1} F_n^T} + \mathbf{Q} \tag{5.11}$$

Update Step:

$$\mathbf{K_n} = \mathbf{P_{n|n-1} H_n^T} (\mathbf{H_n P_{n|n-1} H_n^T} + \mathbf{R})^{-1} \tag{5.12}$$

$$\mathbf{m_{n|n}} = \mathbf{m_{n|n-1}} + \mathbf{K_n}(\mathbf{\Xi_n} - \mathbf{H_n m_{n|n-1}}) \tag{5.13}$$

$$\mathbf{P_{n|n}} = \mathbf{P_{n|n-1}} - \mathbf{K_n H_n P_{n|n-1}} \tag{5.14}$$

Here $\mathbf{m_{n|n}}$, $\mathbf{P_{n|n}}$ are the Gaussian posterior mean and covariance at n^{th} time-step, and $\mathbf{K_n}$ is the Kalman gain at that step. The clock states at n^{th} time-step is given by the components of $\mathbf{m_{n|n}}$, since $\mathbf{m_{n|n}}$ is the learned estimate of time offset, frequency offset and drift at that step. For our simulations, we assume the Gaussian posterior mean at the beginning is zero and the initial posterior covariance is $\mathbb{I}^{3\times3}$.

5.3.4 Signal Authentication and Anomaly Detection

Spoofing attacks induce variations in the GPS reference signal ranging from discrete step changes to slowly evolving changes in the demodulated GPS data. Our anomaly detection strategy classifies changes in the time offset, frequency drift, and frequency offset measurements in the received GPS data in relation to the hardware oscillator

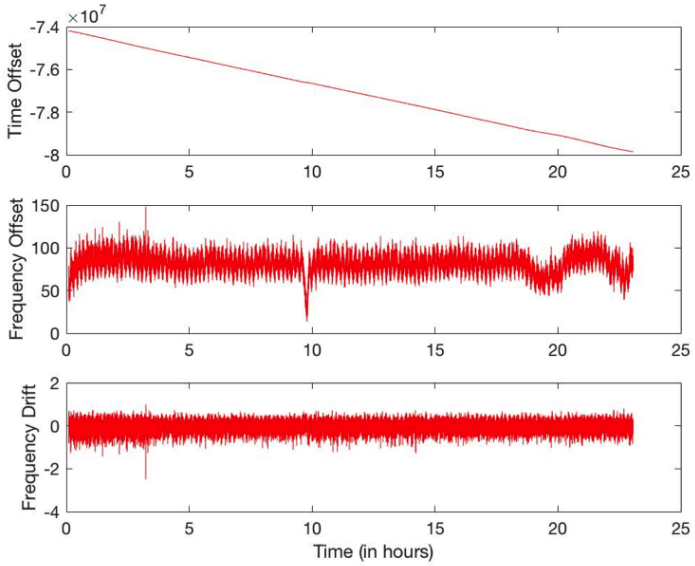

Figure 5.4: Example time offset, frequency offset and frequency drift for a crystal oscillator with respect to a stable GPS clock.

as anomalous when $\mathbf{X_n}$ lies outside the confidence interval of its predicted value $\mathbf{m_{n|n-1}} \pm \mathbf{S_{n-1}}$, where,

$$\mathbf{S_{n-1}} = \mathbf{H_n P_{n|n-1} H_n^T} + \mathbf{R} \tag{5.15}$$

is the predicted variance of the offset. This approach can be used for detecting simpler attacks inducing a step change in the time offset. This technique depends only on a single data point and an estimate, and therefore, leads to a large number of false positives. Moreover, an *advanced* attacker will self-consistently change the offset to avoid such detection.

A better approach is to use a windowed strategy that takes account of a number of recent measurements and find out the likelihood of a new measurement

and estimate. For this approach, we calculate

$$p(m_{i,n}|m_{i,n-1},\dots m_{i,n-k}) = \frac{1}{\sqrt{2\pi\sigma^2_{i,n-1}}} e^{(-\frac{(m_{i,n}-\bar{m}_{i,n-1})^2}{\sigma^2_{i,n-1}})} \tag{5.16}$$

where, $i \in \mathbb{Z}^+$ for a 3×1 Gaussian posterior mean, k is the window size, $\sigma^2_{i,n-1}$ is the variance and $\bar{m}_{i,n-1}$ is the mean of the predicted values inside the window $(n-1)$ to $(n-k)$. By calculating a moving average of the log-likelihood z_n, we can detect an anomalous event when z_n crosses a predefined threshold. Here,

$$z_{i,n} = \alpha z_{i,n-1} + (1-\alpha)\ln(p(m_{i,n})) \tag{5.17}$$

with α as the smoothing factor.

5.4 Experimental Setup and Results

For our experiments, we use a commercial GPS receiver and temperature-compensated crystal oscillators (TCXOs) to build an ensemble of free-running hardware oscillators. We use this design to validate the assumptions and formulation presented in Section 5.3. The measurement infrastructure is comprised of an ARM microprocessor-based data acquisition system with on-board data storage. Since this is a data-level spoofing detection technique, we inject attacks by altering the GPS time-offsets of the received GPS data and detect the attacks by using the free-running TCXO clock offsets measured with respect to this spoofed GPS data.

Figure 5.5: Data collection and instrumentation setup for the experiments presented in this work. The instruments are (from top to bottom): a PTP transparent clock, two GPS receivers, and a Rubidium clock.

5.4.1 Adversary Model

The major goal of the adversary is to produce erroneous time or position measurements in the GPS receiver. We assume that the adversary has complete access to the RF channel during the attack, *i.e.,* he can replay, alter and/or replicate the RF carrier, spreading code and data bits of any or all of the visible satellites. We can divide the attacks in two categories:

1. Temporal-shift injection

2. Meaconing and replay attack

A temporal-shift injection attack changes the time and/or the position bits in the GPS signal, which is reflected a sudden jump in the perceived time/location of the victim. A meaconing and replay attack first induces the receiver to lock onto its spoofed signal by transmitting code, phase, and Doppler-matched signals with

gradually increasing power, and then drags off the code and phase carrier in such a way that he avoids a discontinuous step change in time or the location estimates of the victim receiver. We assume that the adversary is time-bound, *i.e.,* he has a limited time to spoof the receiver.

5.4.2 Attack Example and Spoofing Detection

To demonstrate the proposed spoofing detection approach, first we consider an attacker performs a temporal shift injection attack on a GPS system. A temporal shift injection is initiated at 5000 seconds represented by a sudden jump in the offset as shown in Figure 5.6. During the attack, the attacker keeps maintains the temporal shift. When tha attack ends, there is a sudden jump in the time offset which is represents the recovery of the authentic signal by the receiver.

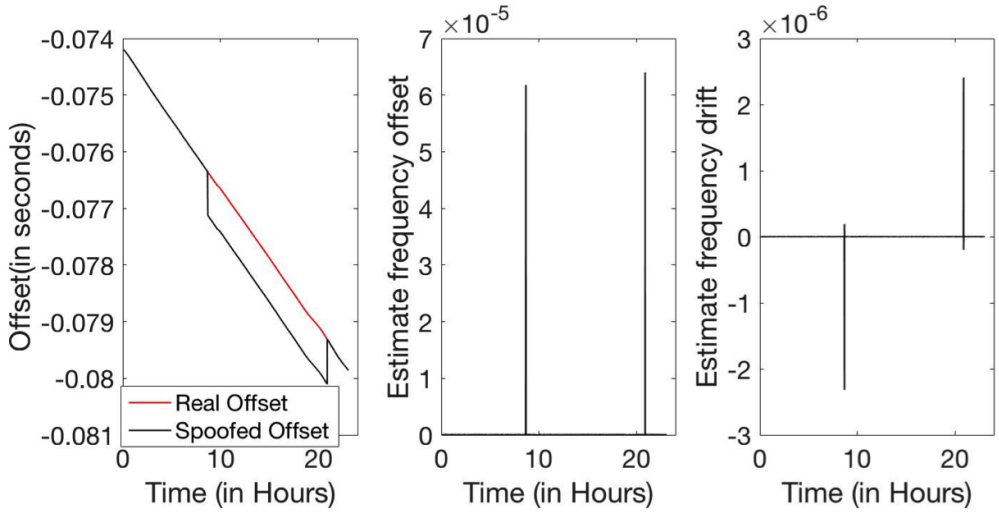

Figure 5.6: Example of a temporal shift injection attack.(a) Clock offset of a free running TCXO with respect to a GPS reference. A temporal shift injection attack is initiated at 5000 seconds. There a sudden jump in the time offset which is corrected at the end of the attack; (b) Estimation of the frequency offset of the local clock with respect to the spoofed GPS signal during before and after the attack. (c) Similar estimation of frequency drift.

Detecting temporal shift injection in clock frequency domain is straightforward as one can notice the sudden overshoot in the estimated frequency offset and frequency drift. Therefore, simple jump detection techniques can be employed for discovering such attacks.

For the meaconing and replay attack we assume that an attack scenario on a trained Phasor Measurement Unit (PMU) as described in [75]. The attack involves the deployment of a simulated gradually increasing delay on GPS signals, which results in an anomalous exaggeration of signal transmission time, and in turn, induces an offset error in the GPS receiver. This attack described by the authors has been

used by other experimental evaluations of fault detection algorithms and provides a well-documented baseline to study the effectiveness of our proposed approach. Figure 5.7(a) illustrates the evolution of the attack starting at 5130 seconds causing a gradual deviation of the GPS trained clock (solid line) against the true reference (dashed line).

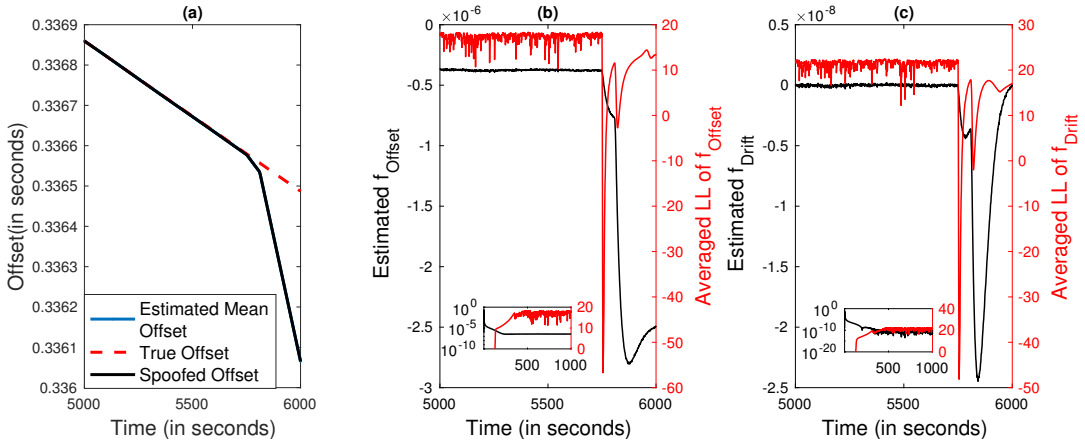

Figure 5.7: Example of a replay and meaconing attack. (a) Clock offset of a free running TCXO with respect to a GPS reference. A spoofing attack is initiated at 5130 seconds. The estimated time offset faithfully follows the spoofed signal as there is no sudden jump in the time offset;(b) Estimation of the frequency offset (black curve) and the averaged log-likelihood of the frequency offset(red curve) of the local clock with respect to the spoofed GPS signal during before and after the attack. The initial start-up and transition period for f_{offset} and the log-likelihood is shown in the inset. (c) Similar estimation of frequency drift (black curve) and the averaged log-likelihood of the frequency drift (red curve) of the local oscillator. The window size (k) for this computation is 128 data points.

To detect the attack, we modeled the TCXOs using the stochastic model presented in Section 5.3. Existing time offset based data level detection techniques can only detect spoofing if there is a discontinuous change in time, caused by a step change in the GPS reference. However, in this attack, the time is delayed slowly making the attack mostly undetectable. Since the process noise measurements of our TCXO were unknown, we used empirical values based on prior literature on clock jitter $q_1 = 10^{-3}, q_2 = 10^{-6}, q_3 = 10^{-9}$. We then design a state space model and use the Kalman filter formulation to detect anomalies in the received GPS signal.

From Figure 5.7, we can see that if we use the simultaneous negative values of averaged log-likelihood of frequency drift and offset as an indication of spoofing attack, the detector can detect the first anomaly at 5752s, (about 10 minutes after the start of the attack). Note that in this particular experiment the spoofing attack was discovered when the cumulative error on the local clock was less than $4\mu s$. This is a relatively small error for some GPS-dependent systems.

5.5 Analysis

5.5.1 Accuracy

The accuracy of the proposed detection technique depends on the noise margin and stability of the hardware clocks. For our experiments, we have used inexpensive temperature controlled crystal oscillators (TCXOs), which provides a degree of immunity to temperature variations. One can use an ensemble of hardware oscillators to create a virtual clock using the system of equations as discussed in section 5.3.1

to reduce this noise. It should be noted that the accuracy and robustness of our detection mechanism requires a prior estimate of the measurement noise $\mathbf{Q}$. Without any prior estimate, for example, if we set the spectral densities q_1, q_2 and q_3 to 1, then this technique detects the attack at 5811 and 5813 seconds and reports a false positive at 3183 seconds.

Moreover, the detection mechanism is dependent on the window size (k) that provides a historical moving average. Larger windows provide better estimates; however, larger window size requires more memory and longer start-up time. For the experiment shown in Figure 5.7, $k = 8$ gives a false alarm rate of 66% which reduces to 20% for $k = 16$ and 0% for $k = 32$% and higher. Furthermore, there is a transient period during startup for the Kalman filter (as shown in the insets of Figure 5.7(b) and (c)) which requires a finite wait period before a consistent estimation of the log-likelihood. Therefore, the weakness of this proposed approach is the start-up period before effective detection is possible. For the example attack provided in this work, it takes about 600 seconds for system start-up.

5.5.2 Computation Cost

The cost of matrix-vector computations for a Kalman filter in the prediction and update step contains computation in the order of $O(\mathbf{D}^2)$, $O(\mathbf{MD})$, and $O(\mathbf{M}^3)$ complexity. The covariance matrices are symmetric, and therefore, Cholesky factorization can be used for maintaining $\mathbf{P}$ in a square-root form. Since the prediction and update step requires the knowledge of only current and previous steps, this construc-

tion has a very low memory complexity. The anomaly detection step has logarithmic complexity which can be simplified by approximating $p(m_{i,n}) \approx \mathbf{e}^{\left(-\frac{(m_{i,n}-\bar{m}_{i,n-1})^2}{\sigma^2_{i,n-1}}\right)}$. The averaging window has a fixed memory requirement which can be lowered by reducing the number of historical data points.

5.5.3 Hardware Overhead

GPS receivers already contain a hardware oscillator which is synchronized using the GPS signals. By turning off the synchronization, it may be possible to convert this clock to a free running oscillator. The synchronization based timing corrections can be performed in software. Another approach is to add a hardware component with embedded free running oscillators to employ the proposed method without altering the GPS receiver design. The computation can be performed using onboard processors in IoT devices or by adding a low power microcontroller that takes GPS derived time as an input and provides the corrected time and spoofing detection capability to the system.

5.5.4 Power Constraints for IoT

It should be noted that using this spoofing detection technique for IoT devices may significantly increase the device's power consumption due to the need for continuous monitoring of the GPS signal. To save power by turning off monitoring periodically, the detector should compute the current frequency states as quickly as possible. Since there is a start-up time for this approach to detect spoofing, one can store

the historical values for the Gaussian mean and variance along with the noise measurements to reduce the startup time. For low power GPS receivers, if the receiver is in sleep mode and starts with a spoofed GPS data from an attacker, then this detection technique may be ineffective due to convergence transients during startup.

5.5.5 Signal Recovery

This detection technique has a-priori knowledge of the correct clock states, and one can calculate an approximated value of the true time offset using the historical clock states. For example, to recover from the example spoofing attack, once the attack is detected, the *approximately* correct value of the time offset can be calculated with the Gaussian posterior mean ($\mathbf{m_{n|n-1}}$) and the historical mean of the predicted values $\bar{\mathbf{m}}_{\mathbf{n,n-1}}$ using:

$$m_{1,n|n} = m_{1,n-1} + \Delta m_{2,n|n-1} + \frac{1}{2}\Delta^2 m_{3,n-1|n} \tag{5.18}$$

$$m_{2,n|n} = \bar{m}_{2,n-1} \tag{5.19}$$

$$m_{3,n|n} = \bar{m}_{3,n-1} \tag{5.20}$$

and then using $m_{1,n|n}$ as the *approximate* value of the true time offset at a time step n.

5.5.6 Hardware Intrinsic Security

Our approach does not depend on networked components for synchronization or attack detection. The hardware oscillator(s), as well as the computational and

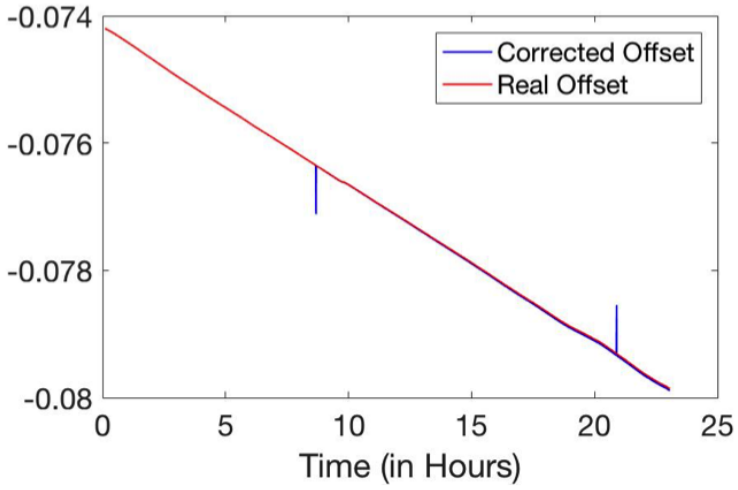

Figure 5.8: Corrected offset using the recovery algorithm for temporal shift injection attack. The correction approach can provide a semi-accurate clock reference during an attack. The accuracy can be improved by using multiple clock sources.

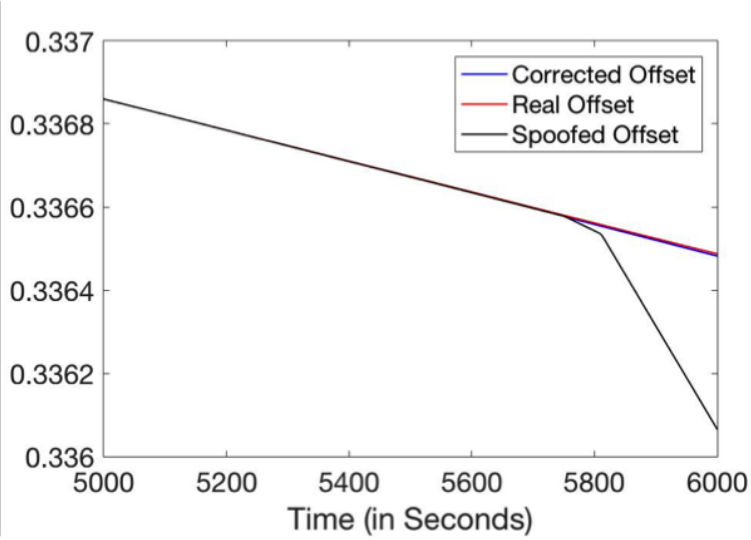

Figure 5.9: Corrected offset using the recovery algorithm for meaconing attack. The correction approach can provide a semi-accurate clock reference during an attack. The accuracy can be improved by using multiple clock sources.

measurement components, are located on the receiver and algorithm updates may be introduced using existing firmware update infrastructure. Since dependency on other networked components can make the system vulnerable to network attacks, the proposed method provides better security and reliability.

5.5.7 Tamper Resistant Design

Tampering with the on-board clock(s) would detectably affect and change the clock frequency states. As the changes are hardware dependent and unique, the design is inherently tamper-resistant.

5.5.8 Comparison with Existing Approaches

As discussed earlier in this chapter, attack detection via comparison of time-offset between the GPS-clock and a local clock is not always reliable, and self-consistent attacks can easily spoof such detection techniques. Furthermore, offset comparison requires time from a single local clock to be synchronized with a trusted clock periodically to keep the readings accurate. As a result, current spoofing detection solutions in the literature on comparing time with a known entity requires a notion of trust to a third party time provider. To avoid these shortcomings, our proposed design measures the clock states of a hardware oscillator with respect to the GPS-time to verify the authenticity of the GPS signal.

It should be noted that commercially available low-cost GPS receivers do not use any of the countermeasures described in the previous section [64]. Most of

the measures are only implemented as prototypes in the lab environment where software-defined radio platforms are used in most of the tracking, analysis, and detection approaches. Implementation cost for practical deployment of these prototypes significantly limits their widespread use. The spoofing detection method presented in this chapter does not require additional RF circuitry or antennas and uses low-cost TCXOs and data analysis to detect attacks. Furthermore, since our approach is a data-level detection mechanism, no change to the receiver architecture is required to integrate our countermeasure as an add-on to existing systems.

5.6 Conclusions

In this work, we present a design for integrating data-level spoofing detection with an existing GPS-based timing system. The design uses single (or multiple) free running oscillators to detect anomalies in the GPS-derived frequency drift and the offset. We demonstrate that this approach can provide fast and accurate detection of GPS spoofing attacks published in the literature. Since GPS spoofing attacks pose a significant threat to IoT systems, including spoofing detection methods such as the method presented in this chapter in future GPS receiver designs will secure future navigation hardware for IoT application.

Chapter 6: Conclusions and Future work

In this dissertation, we have demonstrated several novel ways of lightweight authentication using hardware dependent techniques. As the IoT space becomes larger, new and efficient security protocols will be required to support a wide, distributed low-power networks. Novel methods ensuring security and privacy will be necessary, as well as existing cryptographic techniques needs to be revisited for this purpose. From a hardware engineering point of view– when power and area budget becomes crucial, techniques similar to the ones discussed in this work will be cost effective and energy efficient. We envision many implications and possible future direction of this work.

6.1 Improved Authentication Mechanism using LPN

From the two simple authentication protocols presented in chapter 3, it is evident that memristive hardware can assist simple authentication and secure key storage facilitates. To harness this potential, we present a learning parity in the presence of noise (LPN)-based authentication scheme that is provably secure against passive attacks. This scheme is derived from the Hopper and Blum(HB) authentication protocol that provides simple yet efficient authentication from hard learning problems

[76, 77]. For this design, we will assume the assumption R5 of section 3.2 holds for the memristor crossbar used in the authentication, (*i.e.*, a memristor can be put into a non-recoverable high resistive state using high current flow through the device.)

Theorem 6.1: The physical state (describing whether a memristor can be written) performs a logical AND operation on the bit written to the memristor.

Proof: Assume the physical state describing whether the memristor can be written is denoted by y (i.e., $y = 1$ meaning there will be successful state change if the data to be written is 1, $y = 0$ means there will not be a state change and the memristor will stay at NRHRS). The existence of nor-recoverable high resistive state ensures that there will be memristors with $y = 0$. Then, if the incoming bit is represented by x and the actual value stored after the WRITE operation is z, one can draw the truth table as shown in Table 6.1

Table 6.1: Truth table describing the relation between the memristive state and the data stored

x (Incoming Bit)	y (Memristive State)	z (Memory Content After WRITE)
0	0(NRHRS)	0
0	1(Writeable)	0
1	0(NRHRS)	0
1	1(Writeable)	1

From the truth table, it is evident that, $z = x \wedge y$.

6.1.1 Description of a LPN-Based Authentication Protocol

Let us denote this protocol as Protocol III as this is a successor of Protocol I and II of chapter 3. The protocol uses a memristive crossbar M of size $\ell \times n$. The secret $\mathbf{X} \in \mathbb{Z}_2^{\ell \times n}$ for authentication is distributed using Algorithm 8. For authentication, multiple round of interactive authentication (as presented in table 6.2) is performed.

Assumptions: Assume R5 in 3.2 holds for the memristors used in the authentication.

Algorithm 8 Key Generation and Storage in Memristive Crossbar for LPN based Authentication

procedure $\mathbf{X} \leftarrow KeyGen03(1^\lambda)$
    Sample $\mathbf{X} \xleftarrow{\$} \mathbb{Z}_2^{\ell \times n}$
    return $\mathbf{X}$
procedure $KeySto3(\mathbf{X})$
    for all $i \in \{1, \ldots, \ell\}$ **do**
        for all $j \in \{1, \ldots, n\}$ **do**
            if $\mathbf{X}[i][j] = 0$ **then**, **RESET** $M[i][j]$ to NRHRS.

Enrollment: The verifier saves $\mathbf{X}$ for later authentication, the prover keeps the crossbar M.

Table 6.2: Single round interactive authentication for protocol III

Prover(M, τ)		Verifier$(\mathbf{X})$
		$\mathbf{R} \xleftarrow{\$} \mathbb{Z}_2^{\ell \times n}$
	$\xleftarrow{\mathbf{R}}$	
— $e \in \{0, 1 \| Prob[e = 1] = \tau\}$		
— Write $\mathbf{R}$ in M using the following scheme:		
If $\mathbf{R}[i][j]$=0, **RESET** $M[i][j]$;		
else **SET** $M[i][j]$		
— Read back the corrupted value $\mathbf{C}$ from M using the following scheme:		
If $M[i][j]$=HRS, $\mathbf{C}[i][j] = 0$		
else $\mathbf{C}[i][j] = 1$		
$z := e \oplus hdp(\mathbf{C}, \mathbf{0}^{\ell \times n})$		
	$\xrightarrow{z}$	
		$\mathbf{P} = \langle \mathbf{R} \circ \mathbf{X} \rangle$
		If $z = hdp(\mathbf{P}, \mathbf{0}^{\ell \times n})$ accept

The authentication is performed in t rounds. The verifier finally accepts the prover if the response of the prover was wrong for fewer than $t\tau$ times.

6.1.2 Security Analysis

Let us assume that an attacker ($\mathcal{A}$) tries to learn the secret $\mathbf{X}$ by eavesdropping $(\mathbf{R}, z)$ over multiple rounds. Then it can be shown that this learning problem for the attacker can be reduced to a learning parity with the presence of noise (LPN) problem.

Definition 6.2 (LPN Problem) Assume $\tau \in \mathbb{R}$ is a constant noise parameter where $0 < \tau < 0.5$, $t \in \mathbb{N}$ is the number of samples, $\mathbf{e}$ is a random binary vector such that $\mathbf{e} \xleftarrow{\$} \{\mathbf{x} \in \mathbb{Z}_2^t : wt(\mathbf{x}) \leq \tau t\}$ and $\mathbf{s}$ be a ℓ-bit binary vector (i.e., $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^\ell$). Given a random binary matrix $\mathbf{R} \xleftarrow{\$} \mathbb{Z}_2^{t \times \ell}$, and $\mathbf{z} = \langle \mathbf{R}, \mathbf{s} \rangle \oplus \mathbf{e}$, find an ℓ-bit vector $\mathbf{x}'$ such that $wt(\langle \mathbf{R}, \mathbf{x}' \rangle \oplus \mathbf{z}) \leq \tau t$.

LPN is an NP-hard problem [78]. The LPN problem is also known as the Syndrome decoding problem that tries to find the closest vector to a random linear error-correcting code, which is believed to be exponentially hard[79]. For $\tau > 0$, the BKI algorithm described in [80] gives a subexponential time algorithm that solves the LPN problem in $2^{O(\ell/log\ell)}$ time. Hopper and Blum first proposed a simple authentication protocol (HB) [76] using the LPN problem. The authentication protocol described in table 6.2 can be derived from the HB protocol.

Theorem 6.3 Assume $\mathbf{r} = vec(\mathbf{R})$ denotes vectorization operation that generates a vector $\mathbf{r} \in \mathbb{Z}_2$ of a matrix $\mathbf{R}$. Then, the $hdp()$ calculation by the prover $\mathcal{P}$ returns the parity of $\langle \mathbf{r}, \mathbf{s} \rangle$ where $\mathbf{r} = vec(\mathbf{R})$, $\mathbf{s} = vec(\mathbf{S})$, $\mathbf{R}$ is the random challenge sent by the verifier and $\mathbf{S}$ is the matrix representing the writabilty of each memristor in the crossbar (i.e., if $\mathbf{S}[i][j] = 0$,$M[i][j]$ is at NRHRS).

Proof: From Theorem 6.2, we can see that, after a writing operation the memory crossbar contains the corrupted value $\mathbf{C}$ which is equal to the binary Hadamard product of $\mathbf{R}$ and the physical state ($\mathbf{S}$) of the crossbar (i.e., $\mathbf{C} = \langle \mathbf{R} \circ \mathbf{S} \rangle$). Then, if we consider $\mathbf{c} = vec(\mathbf{C})$, $\sum_i \mathbf{c}[i] \bmod 2$ will be equal to the inner product of $\mathbf{r}$ and $\mathbf{s}$ (i.e., $\sum_i \mathbf{c}[i] \bmod 2 = \langle \mathbf{r}, \mathbf{s} \rangle$). Since $hdp(\mathbf{C}, \mathbf{0})$ defines the parity of the Hamming distance between $\mathbf{C}$ and $\mathbf{0}$, it effectively calculates the parity of the Hamming weight of $\mathbf{c}$ which is equal to $\sum_i \mathbf{c}[i] \bmod 2$. Therefore, $hdp(\mathbf{C}, \mathbf{0}) = \langle \mathbf{r}, \mathbf{s} \rangle$.

From theorem 6.3 it is evident that XORing an error e to the response of the prover ($hdp(\mathbf{C}, \mathbf{0})$), one can construct an LPN problem with the shared secret $\mathbf{S}$ representing the physical state matrix of a memristive crossbar as shown in the authentication protocol at table 6.2 .

This protocol is provably secure against passive attackers. However, it can be shown that the protocol is insecure against active attackers and man-in-the-middle (MIM) attacks. So, we improved the protocol in a similar fashion proposed at [77].

Table 6.3: Single round interactive authentication for protocol III+

Prover($M, \mathbf{Q_v} \in \mathbb{Z}_2^{\ell \times n}, Q_z \in \{0,1\}, \tau$)	Verifier($\mathbf{X}, \mathbf{Q_v} \in \mathbb{Z}_2^{\ell \times n}, Q_z \in \{0,1\}$)
	$\mathbf{V} \xleftarrow{\$} \mathbb{Z}_2^{\ell \times n}$
$\xleftarrow{\mathbf{V}}$	
— $e \in \{0,1 \| Prob[e=1] = \tau\}$	
— $\mathbf{R} \xleftarrow{\$} \mathbb{Z}_2^{\ell \times n}$	
— $\mathbf{Y} = \mathbf{Q_v} \oplus \mathbf{V}$	
— Write $\mathbf{R}$ in M using the following scheme:	
If $\mathbf{R}[i][j]$=0, **RESET** $M[i][j]$;	
else **SET** $M[i][j]$	
— Write $\mathbf{Y}$ in M	
— Read back the corrupted value $\mathbf{C}$ from M using the following scheme:	
If $M[i][j]$=HRS, $\mathbf{C}[i][j] = 0$	
else $\mathbf{C}[i][j] = 1$	
$z := Q_z \oplus e \oplus hdp(\mathbf{C}, \mathbf{0}^{\ell \times n})$	
$\xrightarrow{(z, \mathbf{R})}$	
	$\mathbf{Y}' = \mathbf{Q_v} \oplus \mathbf{V}$
	$\mathbf{P} = \langle \mathbf{R} \circ \langle \mathbf{X} \circ \mathbf{Y}' \rangle \rangle$
	If $z = Q_z \oplus hdp(\mathbf{P}, \mathbf{0}^{\ell \times n})$ accept

6.2 Efficient Memristive Hardware Design and Fabrication

The authentication scheme discussed in the previous section uses an $\ell \times n$ 1T1M memristive crossbar M. The key-storing process described by the procedure $KeySto3(\mathbf{X})$ puts the memristive element $M[i][j]$ to non-recoverable high resistive state (NRHRS) for $\mathbf{X}[i][j] = 0$. This operation makes later LPN-based computation simpler to achieve with the memristive hardware.

The calculation for the $hdp()$ function can easily be supported using finite state machine built with a D-flip-flop, a two input AND gate and a two input XOR gate. From our observation on the effect of variation in simple memristor based security hardware, we recognize the opportunities in the following areas for future research.

1. Novel circuit designs are required for reducing the supply voltage noise across a memristor used in security hardware. Although memory applications can sustain the effect of noise variations and operate reliability due to the simple binary operation, security application that wants to harness the unique properties (such as nonlinearity, physical variability, etc.) of these novel devices memristor to evaluate their designs reliability with variation.

2. New simple error correction techniques need to be developed to fight predictable environmental and operational variations.

3. Novel algorithmic techniques and supporting hardware are required for developing better state read-out and copy operation of memristive devices used for

device security.

Furthermore, we have fabricated and characterized memristive hardware for the experimentation purposes. We find that fabricating reliable and cycle accurate memristive hardware requires extensive processing and fabrication techniques. Hence, future research on the optimized fabrication is required to reduce the cost of memristive hardware.

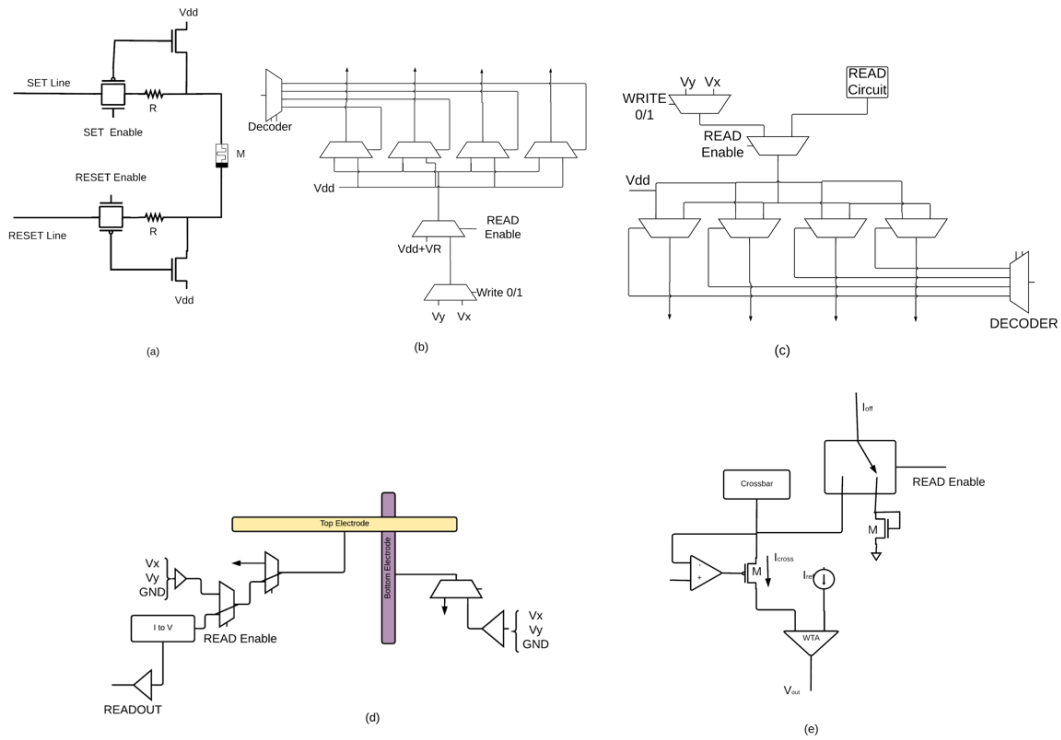

Figure 6.1: Additional driver circuits required for efficient READ-WRITE operation [7]. Sub-figure (a) shows how SET and RESET lines are controlled for setting and resetting the memristor, R represents the line resistance for the SET and RESET line; (b) and (c) represents the SET and RESET line driver designs required for controlling the memristors in the crossbars; (d) shows a how read operation is performed using the driver circuits; and (e) represents the sensing circuit required for readout.

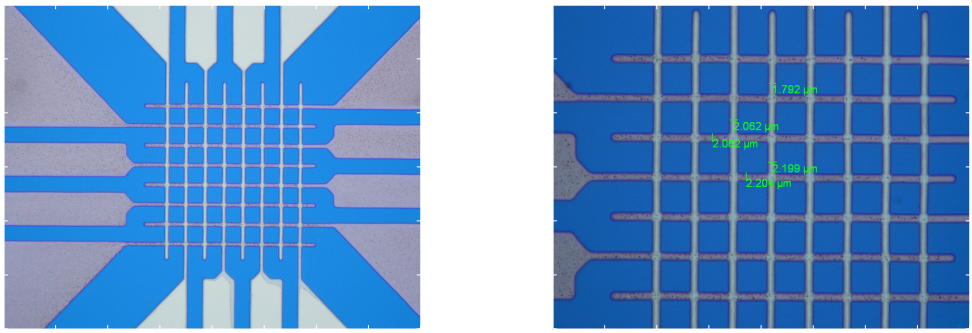

Figure 6.2: Fabricated memristor and its dimensions. The top (light colored) and bottom (dark colored) electrode are seen as the gray crossbars. The width of the top and bottom crossbars are $2\mu m$ on average. In between the top and bottom crossbars, a thin film of HfO_x is deposited using atomic layer deposition.

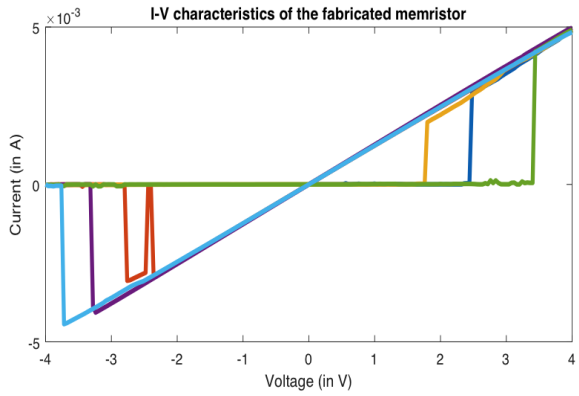

Figure 6.3: Sample I-V curve for the fabricated memristor. Multiple read-write cycle of the memristor is provided. The violet, sky-blue and red lines represent a LRS to HRS transition when the voltage across a memristor at a low resistive state is varied from +4 to -4V. The green, yellow and dark-blue lines represent a HRS to LRS transition when the voltage across a memristor at a high resistive state is varied from -4 to +4V.

6.3 Application of Visual Cryptography in Physical Layer Group Key Agreement Protocols

This dissertation has demonstrated the importance of simple yet secure cryptographic protocols that can take advantage of physical properties. Such can also be applicable to key distribution and key agreement protocols that uses physical nature of the physical layer. For example, Jain et al. [81] have recently developed physical layer group key agreement protocols on a controller area network (CAN) bus that uses the capability of the CAN bus to calculate logical AND over the bus. In this section, we show that the techniques presented in this thesis can be applicable in such physical designs.

In this section, we develop the application of visual cryptographic schemes in secure key storage for physical layer group key agreement protocols on a controller area network (CAN) bus. For this application, we have modified the group key agreement schemes of Plug and Secure (PnS) protocol described by Jain et al. [81]. We have observed that the intrinsic properties of a CAN bus provide the opportunity to apply Visual Cryptographic (VC) primitives quickly and this can be helpful in secure key storage for secrets generated using PnS protocols.

6.3.1 Problem Definition

Assume N users join in a PnS protocol to generate a L-bit secret $\mathcal{K}$. After the protocol finishes, each user will successfully generate $\mathcal{K}$ and use it for future com-

munications within the group. However, compromising one node in a group will reveal the shared key to an attacker, especially if the key is being stored for a longer period. Therefore, a solution is required for distributed key storage which will (1) provide security against attacks on nodes (2) provide fast regeneration of the keys when required.

6.3.2 Solution

To solve this key storage problem, we use the constructs of visual cryptography. First, let's consider a toy example. Assume the case where authenticated nodes A and B are in a group, and they share a b-bit secret $\mathcal{K}$. Now, for longer storage of $\mathcal{K}$, A and B decide to generate a new key $\mathcal{K}_n$ using the following algorithm:

For the i^{th} bit in $\mathcal{K}$

1. If the bit is 0, A and B calculate the output of the hash function $h(\mathcal{K}, i)$. If the output has an even parity then both of them save "0" in the $2i^{th}$ and "1" in the $(2i+1)^{th}$ place in $\mathcal{K}_n$ and if the output has an odd parity then both of them save "1" in the $2i^{th}$ and "0" in the $(2i+1)^{th}$ place in $\mathcal{K}_n$.

2. If the bit is 1, A and B calculate the output of the hash function $h(\mathcal{K}, i)$. If the output has an even parity then A saves "0" in the $2i^{th}$ and "1"in the $(2i+1)^{th}$ place in $\mathcal{K}_n$ and B save "1" in the $2i^{th}$ and "0" in the $(2i+1)^{th}$ place in $\mathcal{K}_n$ in its key. When it results in odd parity, A and B invert their roles.

Now, for an adversary attacking any one of the nodes, it is impossible to get the secret $\mathcal{K}$ looking at the key stored in either A or B.

To regenerate the secret, all A and B has to do is put their stored keys simultaneously on the CAN bus. Due to the intrinsic wired-AND operation on the CAN bus, the secret can be reconstructed easily. For each adjacent pair of bits on the CAN bus, if they are "00", then the corresponding secret bit is reconstructed as 1, and if they are "01" or "10", the corresponding secret is reconstructed as 0.

The problem associated with this solution is that an eavesdropper can also easily deduce the secret by looking at the CAN bus. To prevent such attacks, A and B can store same inverting mask that inverts the bits for some (ideally half of the) bit positions in secret. XORing with a common masking string dedicated to the group can easily accomplish the task.

Based on this discussion above, we provide a detailed description of expanding this scheme to a 2-out-of-n scheme, where n-ECUs shares the secret string $\mathcal{K}$ and at least 2-ECUs are required for successfully regenerate the secret. We define two Boolean matrices B_0 and B_1, and C_0 and C_1 is the set of all the matrices generated by permuting the columns of B_0 and B_1. Then, for n users we pick an integer m such that $\binom{m}{m/2} \geq n$. We can consider a ground set $S$of size m and all subsets of S os size $\frac{m}{2}$. We can design B_0 as a $n \times m$ matrix with each row containing $m/2$ zeros and $m/2$ ones. The i^{th}row of B_1 corresponds to the i^{th} subset of S, i.e., if the j^{th} element belongs to thei^{th} subset then $B_1(i,j) = 1$, otherwise, $B_1(i,j) = 0$[21].

Then at each node, we can do the following. Every node is allocated one distinct row of B_0 (lets denote the row with $B_{0,n}$)and one distinct row of B_1 (lets denote the row with $B_{1,n}$). For i^{th} bit position, the node calculates $y = h(\mathcal{K}, i)\ mod\ m$. Then if the i^{th} bit is zero, it rotates $B_{0,n}$ y times and stores the m bits and if the i^{th}

bit is one, it rotates $B_{1,n}$ y times and stores the corresponding m bits.

For example, consider the case where n=6. We can choose m=4 since $\binom{4}{2} \geq n$. Now consider a ground set $G = \{1, 2, 3, 4, 5, 6\}$ and the subset of G containing two elements are $\{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}$

Then, $B_0 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$ *and* $B_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$

Using the similar construction, Naor and Shamir extended the case for a generating B_0 and B_1 for k-out-of-k scheme. For k nodes, let us consider a ground set G consisting of k elements $g_1, g_2, ...g_k$. We denote all subsets of G with even cardinality as $p_1, p_2, .., p_{2^{k-1}}$ and odd cardinality is denoted by $q_1, q_2, , q_{2^{k-1}}$. The resulting Boolean metrics B_0 and B_1 will have dimensions $k \times 2^{k-1}$. From [21], we can define, $B_0[i,j] = 1$iff $g_i \in p_j$ and $B_1[i,j] = 1$iff $g_i \in q_j$. Then using the same idea as before, each node has each distinct row of B_0 and B_1 and uses a hash function to generate the permuted representation for i^{th} bit. A n-out-of-k scheme can also be generated by extending this scheme as described by Naor and Shamir[21].

Thus we have demonstrated the use of VC for storing the PnS secret in a way so that compromising some node would not reveal the secret to the adversary, and the secret for group key authentication can be stored in the distributed nodes without the requirement of a dealer dealing the secret shares beforehand.

List of Publications

The following publications further encompasses the contribution of this dissertation work.

Book Chapter

1. **Md Tanvir Arafin** & Gang Qu, "Memristor-based Security", To appear in *Security Opportunities by Nano Devices and Emerging Technologies*, CRC Press/ Taylor & Francis, USA.

Journal Publications

1. Mingze Gao, Qian Wang, **Md Tanvir Arafin**, Yongqiang Lyu, & Gang Qu, "Novel Approximate Data Formats for Low Power and Security in the Internet of Things ", *IEEE Computer*, Volume 50, Issue 6, pp.27–34, 2017.

2. **Md. Tanvir Arafin**, & Gang Qu,, "Memristors for Secret Sharing based Authentication," to appear in *IEEE Transactions On Very Large Scale Integration (VLSI) Systems*.

3. **Md. Tanvir Arafin**, Dhanajay Anand, & Gang Qu,, "Hardware Clocks for GPS Spoofing Detection," under preparation.

Conference Publications

1. **Md Tanvir Arafin**, Dhananjay Anand & Gang Qu, "A Low-Cost Secure GPS Spoofing Detector Design for the Internet of Things Applications", In Proceedings of the *27th ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pp.161–166, 2017. **(Best Paper Candidate)**

2. **Md Tanvir Arafin**, Mingze Gao & Gang Qu, "VOLtA: Voltage Over-Scaling Based Lightweight Authentication for IoT Applications", In Proceedings of the *22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 336–341, 2017.

3. Shalabh Jain, Qian Wang, **Md Tanvir Arafin**,& Jorge Guajardo,, "Probing Attacks on Key Agreement for Automotive Controller Area Networks,", To appear in Proceedings of the *Embedded Security on Cars* (ESCAR), 2017.

4. **Md Tanvir Arafin**, Andrew Stanley & Praveen Sharma "Hardware-Based Anti-Counterfeiting Techniques for Safeguarding Supply Chain Integrity", To appear in Proceedings of the *50th IEEE International Symposium on Circuits & Systems (ISCAS)*, 2017.

5. **Md Tanvir Arafin** & Gang Qu, "Secret Sharing and Multi-user Authentication: From Visual Cryptography to RRAM Circuits", In Proceedings of the *26th ACM Great Lakes Symposium on VLSI (GLSVLSI),* pp.169–174, 2016.

6. **Md Tanvir Arafin**, Dhananjay Anand & Gang Qu, "Detecting GNSS Spoofing using a Network of Hardware Oscillators", In Proceedings of the *47th An-*

nual Precise Time and Time Interval Systems and Applications Meeting, pp.74 - 79, 2016.

7. **Md Tanvir Arafin** & Gang Qu, "RRAM Based Lightweight User Authentication", In Proceedings of the *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 139–145, 2015.

8. **Md Tanvir Arafin**, Carson Dunbar, Gang Qu, Nathan McDonald & Lok Yan, "A Survey on Memristor Modelling and Security Applications", In Proceedings of the*16th International Symposium on Quality Electronic Design (ISQED),* pp.440–447, 2015. **(Invited Paper)**

U.S. Patent and Invention Disclosure

1. **Method to Mitigate Transients Based Attacks on Key Agreement Schemes over Controller Area Network**, Patent application number: 62/468,669.

2. **Method to Mitigate Voltage Based Attacks on Key Agreement over Controller Area Network**, Patent application number 62/468,705.

Poster Presentation

1. **Hardware Security for IoT devices**, Amazon Graduate Research Symposium, Seattle, Washington, 2017.

Bibliography

[1] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[2] Dmitri B Strukov, Gregory S Snider, Duncan R Stewart, and R Stanley Williams. The Missing Memristor Found. *Nature*, 453(7191):80–83, 2008.

[3] D Biolek, V Biolkova, and Z Biolek. SPICE Model of Memristor with Nonlinear Dopant Drift. *Radioengineering*, 2009.

[4] Hisham Abdalla and Matthew D Pickett. SPICE Modeling of Memristors. In *International Symposium on Circuits and Systems (ISCAS)*, pages 1832–1835. IEEE, 2011.

[5] Garrett S Rose, Nathan McDonald, Lok-Kwong Yan, Bryant Wysocki, and Karen Xu. Foundations of Memristor Based PUF Architectures. In *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 52–57. IEEE, 2013.

[6] Hyongsuk Kim, Maheshwar Pd Sah, Changju Yang, and Leon O Chua. Memristor-based multilevel memory. In *Cellular nanoscale networks and their applications (CNNA), 2010 12th international workshop on*, pages 1–6. IEEE, 2010.

[7] Bhaswar Chakrabarti, Miguel Angel Lastras-Montaño, Gina Adam, Mirko Prezioso, Brian Hoskins, M Payvand, A Madhavan, A Ghofrani, L Theogarajan, K-T Cheng, et al. A multiply-add engine with monolithically integrated 3d memristor crossbar/CMOS hybrid circuit. *Scientific Reports*, 7:42429, 2017.

[8] Gang Qu and Lin Yuan. Design things for the internet of thingsan eda perspective. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 411–416. IEEE, 2014.

[9] H-S Philip Wong, Heng-Yuan Lee, Shimeng Yu, Yu-Sheng Chen, Yi Wu, Pang-Shiu Chen, Byoungil Lee, Frederick T Chen, and Ming-Jinn Tsai. Metal–oxide rram. *Proceedings of the IEEE*, 100(6):1951–1970, 2012.

[10] Kuk-Hwan Kim, Siddharth Gaba, Dana Wheeler, Jose M Cruz-Albrecht, Tahir Hussain, Narayan Srinivasa, and Wei Lu. A functional hybrid memristor crossbar-array/cmos system for data storage and neuromorphic applications. *Nano letters*, 12(1):389–395, 2011.

[11] Sung Hyun Jo, Kuk-Hwan Kim, Ting Chang, Siddharth Gaba, and Wei Lu. Si memristive devices applied to memory and neuromorphic circuits. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 13–16. IEEE, 2010.

[12] Garrett S Rose, Nathan McDonald, Lok-Kwong Yan, and Bryant Wysocki. A write-time based memristive puf for hardware security applications. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 830–833. IEEE, 2013.

[13] Garrett S Rose, Jeyavijayan Rajendran, Nathan McDonald, Ramesh Karri, Miodrag Potkonjak, and Bryant Wysocki. Hardware security strategies exploiting nanoelectronic circuits. In *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, pages 368–372. IEEE, 2013.

[14] Jeyavijayan Rajendran, Ramesh Karri, and James Bradley Wendt. Nanoelectronic solutions for hardware security.

[15] Yandan Wang, Wei Wen, Hai Li, and Miao Hu. A novel true random number generator design leveraging emerging memristor technology. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pages 271–276. ACM, 2015.

[16] Junwei Sun, Yi Shen, Quan Yin, and Chengjie Xu. Compound synchronization of four memristor chaotic oscillator systems and secure communication. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 23(1):013140, 2013.

[17] Md Tanvir Arafin and Gang Qu. Rram based lightweight user authentication. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '15, pages 139–145, Piscataway, NJ, USA, 2015. IEEE Press.

[18] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[19] George Robert Blakley. Safeguarding cryptographic keys. *Proc. of the National Computer Conference1979*, 48:313–317, 1979.

[20] Douglas R. Stinson. An explication of secret sharing schemes. *Designs, Codes and Cryptography*, 2(4):357–390, 1992.

[21] Moni Naor and Adi Shamir. Visual cryptography. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 1–12. Springer, 1994.

[22] Moni Naor and Benny Pinkas. Visual authentication and identification. In *Annual International Cryptology Conference*, pages 322–336. Springer, 1997.

[23] Roxana Geambasu, Tadayoshi Kohno, Amit A Levy, and Henry M Levy. Vanish: Increasing data privacy with self-destructing data.

[24] Nathan Beckmann and Miodrag Potkonjak. Hardware-based public-key cryptography with public physically unclonable functions. In *International Workshop on Information Hiding*, pages 206–220. Springer, 2009.

[25] Roel Maes. *Physically Unclonable Functions*. Springer, 2013.

[26] Roel Maes and Ingrid Verbauwhede. Physically unclonable functions: A study on the state of the art and future research directions. In *Towards Hardware-Intrinsic Security*, pages 3–37. Springer, 2010.

[27] G Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference*, pages 9–14. ACM, 2007.

[28] Leon Chua. Memristor-the missing circuit element. *IEEE Transactions on Circuit Theory*, 18(5):507–519, 1971.

[29] Leon O Chua and Sung Mo Kang. Memristive devices and systems. *Proceedings of the IEEE*, 64(2):209–223, 1976.

[30] Leon Chua. Resistance Switching Memories are Memristors. In *Memristor Networks*, pages 21–51. Springer, 2014.

[31] Leon O Chua. The Fourth Element. *Proceedings of the IEEE*, 100(6):1920–1927, 2012.

[32] Yogesh N Joglekar and Stephen J Wolf. The Elusive Memristor: Properties of Basic Electrical Circuits. *European Journal of Physics*, 30(4):661, 2009.

[33] Themistoklis Prodromakis, Boon Pin Peh, Christos Papavassiliou, and Christofer Toumazou. A Versatile Memristor Model with Nonlinear Dopant Kinetics. *IEEE Transactions on Electron Devices*, 58(9):3099–3105, 2011.

[34] S Benderli and TA Wey. On SPICE Macromodelling of TiO_2 Memristors. *Electronics Letters*, 45(7):377–379, 2009.

[35] Eero Lehtonen and Mika Laiho. CNN Using Memristors for Neighborhood Connections. In *12th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA)*, pages 1–4. IEEE, 2010.

[36] Matthew D Pickett, Dmitri B Strukov, Julien L Borghetti, J Joshua Yang, Gregory S Snider, Duncan R Stewart, and R Stanley Williams. Switching Dynamics in Titanium Dioxide Memristive Devices. *Journal of Applied Physics*, 106(7):074508, 2009.

[37] John G Simmons. Generalized Formula for the Electric Tunnel Effect Between Similar Electrodes Separated by a Thin Insulating Film. *Journal of Applied Physics*, 34(6):1793–1803, 1963.

[38] Shahar Kvatinsky, Eby G Friedman, Avinoam Kolodny, and Uri C Weiser. Team: Threshold Adaptive Memristor Model. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(1):211–221, 2013.

[39] Alon Ascoli, Ronald Tetzlaff, Fernando Corinto, and Marco Gilli. PSpice Switch-Based Versatile Memristor Model. In *International Symposium on Circuits and Systems (ISCAS)*, pages 205–208. IEEE, 2013.

[40] Ximeng Guan, Shimeng Yu, and HS Philip Wong. A spice compact model of metal oxide resistive switching memory with variations. *IEEE electron device letters*, 33(10):1405–1407, 2012.

[41] Md Tanvir Arafin and Gang Qu. RRAM based lightweight user Authentication. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '15, pages 139–145. IEEE Press, 2015.

[42] Jeyavijayan Rajendran, Garrett S Rose, Ramesh Karri, and Miodrag Potkonjak. Nano-PPUF: A Memristor-based Security Primitive. In *Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 84–87. IEEE, 2012.

[43] James B Wendt and Miodrag Potkonjak. The Bidirectional Polyomino Partitioned PPUF as a Hardware Security Primitive. In *Global Conference on Signal and Information Processing (GlobalSIP)*, pages 257–260. IEEE, 2013.

[44] Jeyavijayan Rajendran, Ramesh Karri, James Bradley Wendt, Miodrag Potkonjak, Nathan R McDonald, Garrett S Rose, and Bryant T Wysocki. Nanoelectronic Solutions for Hardware Security. *IACR Cryptology ePrint Archive*, 2012:575, 2012.

[45] Garrett S Rose, Nathan McDonald, Lok-Kwong Yan, and Bryant Wysocki. A Write-time Based Memristive PUF for Hardware Security Applications. In *Proceedings of the International Conference on Computer-Aided Design*, pages 830–833. IEEE Press, 2013.

[46] Omid Kavehei, Chun Hosung, Damith Ranasinghe, and Stan Skafidas. mrPUF: A Memristive Device Based Physical Unclonable Function. *arXiv preprint arXiv:1302.2191*, 2013.

[47] Md Tanvir Arafin and Gang Qu. Secret sharing and multi-user authentication: From visual cryptography to RRAM circuits. In *Proceedings of the 26th Edition on Great Lakes Symposium on VLSI*, pages 169–174. ACM, 2016.

[48] Anas Mazady, Md Tauhidur Rahman, Domenic Forte, and Mehdi Anwar. Memristor PUFa security primitive: Theory and experiment. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 5(2):222–229, 2015.

[49] Ulrich Rührmair, Christian Jaeger, Matthias Bator, Martin Stutzmann, Paolo Lugli, and György Csaba. Applications of High-Capacity Crossbar Memories in Cryptography. *IEEE Transactions on Nanotechnology*, 10(3):489–498, 2011.

[50] Yevgeniy Dodis, Leonid Reyzin, and Adam 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*, pages 523–540. Springer, 2004.

[51] S Shin, K Kim, and S-M Kang. Memristive xor for resistive multiplier. *Electronics letters*, 48(2):78–80, 2012.

[52] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(9):56–64, 1989.

[53] Gwoboa Horng, Tzungher Chen, and Du-Shiau Tsai. Cheating in visual cryptography. *Designs, Codes and Cryptography*, 38(2):219–236, 2006.

[54] Mohamed Elgebaly and Manoj Sachdev. Variation-aware adaptive voltage scaling system. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 15(5):560–571, 2007.

[55] Nilanjan Banerjee, Georgios Karakonstantis, and Kaushik Roy. Process variation tolerant low power dct architecture. In *Proceedings of the conference on Design, automation and test in Europe*, pages 630–635. EDA Consortium, 2007.

[56] Jie Han and Michael Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *2013 18th IEEE European Test Symposium (ETS)*, pages 1–6. IEEE, 2013.

[57] Rangharajan Venkatesan, Amit Agarwal, Kaushik Roy, and Anand Raghunathan. Macaco: Modeling and analysis of circuits for approximate computing. In *Proceedings of the International Conference on Computer-Aided Design*, pages 667–673. IEEE Press, 2011.

[58] Fatemeh Ganji, Shahin Tajik, and Jean-Pierre Seifert. Why attackers win: on the learnability of xor arbiter pufs. In *International Conference on Trust and Trustworthy Computing*, pages 22–39. Springer, 2015.

[59] James E Stine, Ivan Castellanos, Michael Wood, Jeff Henson, Fred Love, W Rhett Davis, Paul D Franzon, Michael Bucher, Sunil Basavarajaiah, Julie Oh, et al. Freepdk: An open-source variation-aware design kit. In *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*, pages 173–174. IEEE, 2007.

[60] Jae W Lee, Daihyun Lim, Blaise Gassend, G Edward Suh, Marten Van Dijk, and Srinivas Devadas. A technique to build a secret key in integrated circuits for

identification and authentication applications. In *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on*, pages 176–179. IEEE, 2004.

[61] Chris Bonebrake and Lori Ross O'Neil. Attacks on GPS time reliability. *Security & Privacy, IEEE*, 12(3):82–84, 2014.

[62] Todd E Humphreys, Brent M Ledvina, Mark L Psiaki, Brady W OHanlon, and Paul M Kintner Jr. Assessing the spoofing threat: Development of a portable GPS civilian spoofer. In *Proceedings of the ION GNSS international technical meeting of the satellite division*, volume 55, page 56, 2008.

[63] Ali Jafarnia-Jahromi, Ali Broumandan, John Nielsen, and Gérard Lachapelle. Gps vulnerability to spoofing threats and a review of antispoofing techniques. *International Journal of Navigation and Observation*, 2012, 2012.

[64] Markus G Kuhn. Signal authentication in trusted satellite navigation receivers. In *Towards Hardware-Intrinsic Security*, pages 331–348. Springer, 2010.

[65] Nils Ole Tippenhauer, Christina Pöpper, Kasper Bonne Rasmussen, and Srdjan Capkun. On the requirements for successful GPS spoofing attacks. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 75–86. ACM, 2011.

[66] Xichen Jiang. Spoofing GPS receiver clock offset of phasor measurement units. 2012.

[67] Pratap Misra and Per Enge. *Global Positioning System: Signals, Measurements and Performance Second Edition*. Lincoln, MA: Ganga-Jamuna Press, 2006.

[68] Peter H Dana and Bruce M Penrod. The role of GPS in precise time and frequency dissemination. *GPS World*, pages 38–43, 1990.

[69] Todd E Humphreys. Detection strategy for cryptographic gnss anti-spoofing. *Aerospace and Electronic Systems, IEEE Transactions on*, 49(2):1073–1090, 2013.

[70] S.B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 227–234 vol.1, Mar 1999.

[71] David W Allan. Time and frequency(time-domain) characterization, estimation, and prediction of precision clocks and oscillators. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 34(6):647–654, 1987.

[72] Tadayoshi Kohno, Andre Broido, and Kimberly C Claffy. Remote physical device fingerprinting. *Dependable and Secure Computing, IEEE Transactions on*, 2(2):93–108, 2005.

[73] Charles A Greenhall. A review of reduced kalman filters for clock ensembles. *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, 59(3):491–496, 2012.

[74] Simon S Haykin. *Adaptive filter theory*. Pearson Education India, 2008.

[75] Daniel P Shepard, Todd E Humphreys, and Aaron A Fansler. Evaluation of the vulnerability of phasor measurement units to GPS spoofing attacks. *International Journal of Critical Infrastructure Protection*, 5(3):146–153, 2012.

[76] Nicholas J Hopper and Manuel Blum. Secure human identification protocols. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 52–66. Springer, 2001.

[77] Eike Kiltz, Krzysztof Pietrzak, David Cash, Abhishek Jain, and Daniele Venturi. Efficient authentication from hard learning problems. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 7–26. Springer, 2011.

[78] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

[79] ER Berlekamp, RJ McEliece, and HCA van Tilborg. On the inherent intractability of certain coding problems. 1978.

[80] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.

[81] Shalabh Jain and Jorge Guajardo. Physical layer group key agreement for automotive controller area networks. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 85–105. Springer, 2016.