Hardware-based Authentication Applications

Md Tanvir Arafin & Gang Qu

Abstract Authentication is one of the most fundamental problems in computer security. Implementation of any authentication and authorization protocol requires the solution of several sub-problems, such as secret sharing, key generation, key storage, and secret verification. With the widespread employment of the Internet-of-Things (IoT), authentication becomes a central concern in the security of resource constraint internet-connected systems. Interconnected elements of IoT devices typically contain sensors, actuators, relays, and processing and control equipment that are designed with a limited budget on power, cost, and area. As a result, incorporating security protocols in these IoT components can be rather challenging. To address this issue, in this chapter, we discuss hardware-oriented security applications for the authentication of users, devices, and data. These applications illustrate the use of physical properties of computing hardware such as main memory, computing units, and clocks for authentication applications in low power on the IoT devices and systems.

6.1 Introduction

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

Md Tanvir Arafin
Morgan State University, Baltimore, MD 21251
e-mail: mdtanvir.arafin@morgan.edu

Gang Qu
University of Maryland, College Park, MD 20742
e-mail: gangqu@umd.edu

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 standard 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 [1].

Compromising regarding security for low power modules can pose severe threats to the complete system. 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. 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. On the other hand, authentication secrets can be created and distributed using secret sharing schemes to resist 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. Thus, secure classical authentication strategies 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.

Hardware security is an emerging field in computer security 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. As an example of hardware-based emerging security applications, in this chapter, we discuss several lightweight hardware intrinsic security solutions to solve problems associated with authentication of entities.

6.2 Organization

In this chapter, we will discuss several examples of hardware based application that authenticates devices, users, and broadcast signals used for navigation. We start with the backgrounds on the fundamentals of hardware based authentication in section 6.3. In the three subsequent sections we explore three different hardware based authentication applications. Section 6.4 explores the application of approximate computation in authentication technique and presents a device authentication protocol VOLtA. Section 6.5 gives a detailed overview of single and multi-user authentication using novel memory component: the memristor. Finally, Section 6.6 demonstrates the application of crystal oscillators in hardware-based signal authentication applications.

6.2.1 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.

6.3 Background

6.3.1 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.

Single Entity Authentication: For authenticating a single user/entity, 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) [2].

Passwords are the most commonly used entity 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 [2]. 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 [3].

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(CRP) 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 [2].

Secret Sharing and Authentication of Multiple Entities: 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 [4] defines the concept of threshold scheme for secret sharing. This (k,n) threshold scheme (where $k \leq n$) can be described as follows:

Definition 6.1 *: Assume that a secret S 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-entity authentication problem, where at least k authentic users must present their shares to gain access to a system. One may argue that authenticating each user and counting the authenticated users can trivially solve the multi-entity 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 [4–7], this method will reveal the identity of each authenticated user, creating user privacy concerns.

Shamir [4] and Blakley [5] 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 [6]. 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.

6.3.2 Hardware in 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, as discussed in the previous section 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. Hardware based security primitives such as physically uncloneable functions (PUFs) provide some solution to these problems.

Entity Authentication Using Physically Uncloneable Functions Silicon Physically Uncloneable Functions (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 [8–10]. For using PUFs in authenticating a device, one can hash a password with the physically uncloneable 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 needs to attack the system while it is up and running, which is significantly harder [11]. Thus, hardware-based random number generators and PUFs can be useful in solving primary key generation and storage challenges. Moreover, physical randomness in these PUF designs are considered inherently independent and identically distributed. Therefore, learning attacks that focus on modeling the internal operation of an well designed PUF fails. The key generation and authentication model for general PUFs are given below.

Algorithm 6.1 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. **Authentication:** An interactive PUF based authentication protocol is given in Table 6.1

Table 6.1 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.

Hardware-based Classical Secret Sharing

On the other hand, a simple hardware-based secret sharing model was first proposed by Naor and Shamir [7] 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 requires 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 designing 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 [12].

6.4 Hardware-based Authentication using Approximation Errors

Physical variations in hardware have already been widely utilized for security applications in the form of physically unclonable function (PUF). However, for device authentication, PUF is costly regarding hardware and power consumption, bringing an obstacle for its widespread usage in IoT. Therefore, in this section, we will illustrate a device authentication protocol that does not require any additional hardware in the low power Things.

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. Let us 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.

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 [13]

$$\sigma_{\Delta V_t} = A_{\Delta V_t} / \sqrt{WL} \tag{6.1}$$

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 [13]

$$d_{gate} \propto \frac{V_{DD}}{\beta(V_{DD} - V_t)^{\alpha}} \tag{6.2}$$

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 [14–16].

From equation 6.2 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.

6.4.1 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) [16]. It was found that the error probability increases as the number of critical paths that fail to meet the timing constraint increase [16]. 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 [16]. 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.

6.4.2 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 [16]. 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,

$$\mathbf{z_i} = f(\mathbf{x_i}, \mathbf{y_i}, \mathbf{x_{i-1}}, \mathbf{y_{i-1}}) \tag{6.3}$$

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. [17].

6.4.3 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.

M1 Voltage over-scaling can produce process variation dependent errors in the computing unit.
M2 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.
M3 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.
M4 Such model discussed in R3 would be unique for each computational unit since the manufacturing-dependent process variations are random in nature

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 in HSpice platform using the FreePDK 45nm libraries [18]. To introduce process variation in our design, we have used 200 modified NMOS and PMOS models with variable threshold voltages. Gaussian distribution with a ±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 synthe-

sized 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 Figure 6.1) with process variations.

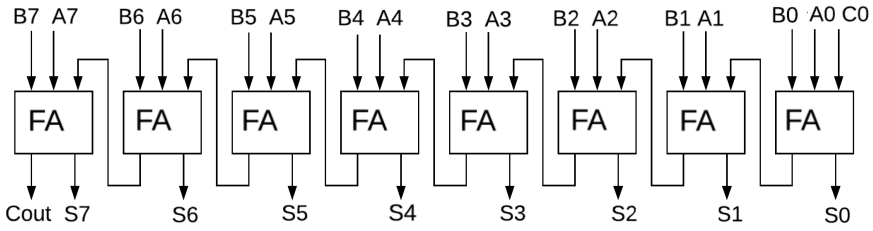

Fig. 6.1 8-bit-ripple carry adder.

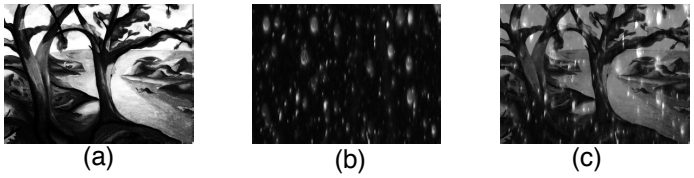

Fig. 6.2 An example of superimposing two images. We have used two grayscale images (a) trees and (b) snowflakes from MATLAB library to generate the superimposed image (c) Snowfall.

From visual observation of Figure 6.3(a), 6.3(b) and 6.3(c), one can clearly notice the effect of voltage over-scaling in this simple image processing application. Figure 6.3(b) and 6.3(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 6.3(d) and 6.3(e), if we plot their differences as done in Figure 6.3(f), one can notice the effect of process variation in the approximately computed result.

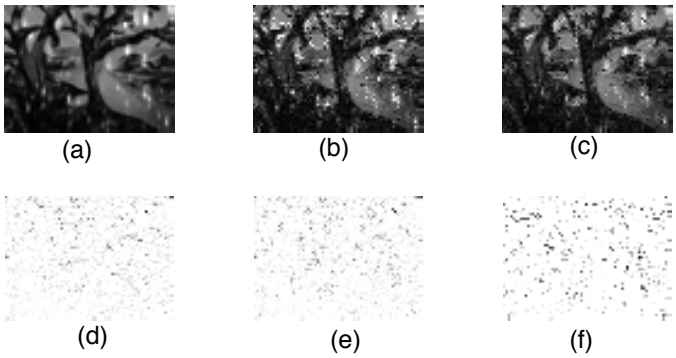

Fig. 6.3 An example of the effect of process variations in voltage over-scaling based computation. In (a) the grayscale image Snowfall is computed using trees and snowflakes without voltage over-scaling using $v_{dd} = 1V$; in (b) and (c) the image is computed under voltage over-scaling using $v_{dd} = 0.4V$ 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 6.3(a), 6.3(b) and 6.3(a),6.3(c) (as done in Figure 6.4) 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.

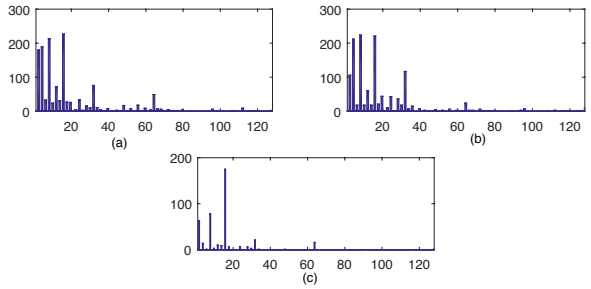

Fig. 6.4 A histogram of the Euclidian distances (a) between the figures 6.3(a) and 6.3(b); (b) between the figures 6.3(a) and 6.3(c); between the figures 6.3(b) and 6.3(c). The x-axis in the sub-figures represents bins of all the possible distance values from 1-128, and the y-axis in the sub-figures represents the number of pixels (in the two corresponding computation of Snowfall images) that share the same distance values.

6.4.4 Authentication Protocol Using Approximation Errors

For single entity authentication, we will assume an interactive protocol VOLtA [19] 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 M1-M4 presented in section 6.4.3. 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 6.2 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 6.2 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.

6.4.5 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 Bob's 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 6.4.1. 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.

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 Alice's 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.

Eavesdropping Attack For eavesdropping attack, Malice eavesdrop on some communications between Alice and Bob and records Bob's 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 Bob's circuit, she can send random string every time for authentication. Therefore, VOLtA would be useful in counteracting such attacks.

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.

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 Alice's 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.

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 $\mathbf{x_2}$.

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.

A detailed discussion on the experimental validation of VOLtA can be found in [19]. In summary, in this section, we introduce a voltage over-scaling based authentication scheme(VOLtA) that uses the random physical variation of a VLSI system. 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.

6.5 Authentication Using Memory Components

In this section, we address the problem of secure and lightweight authentication of single and multiple entities using memory system components. 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.

We consider emerging resistive random access memory(RRAM), also known as memristors, devices for secret sharing based authentication. Memristor-based crossbars are used for designing one of the first memristor-based PUFs called nano-PPUF [20]. 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 [20], $\mathbf{V_C}$ 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 [21] and [22].

6.5.1 Requirements and Utility Functions

Before proposing memristor based entity 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.

From common device features of memristors, we specify our assumptions on the expected device behavior that are needed for the design of the secret sharing based 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.

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.

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.

6.5.2 An example of single entity authentication: Protocol I

For single entity 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.

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 6.3) is performed.

Assumptions: 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 6.3 Key Generation Using Resistive States

procedure $\mathbf{x} \leftarrow KeyGen01(1^{\lambda}, \alpha)$ ▷ $\alpha \in \mathbb{N}$ is the experiment repetition value
  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 $t_{wr,avg}$.
  Assume the key length $\ell = nb$, where $n, b \in \mathbb{N}$. $\ell, t_{wr,avg}, V_{select}, V_{BL}, V_{WL}$ values depend on the security parameter k.
  for j=1 ... α **do**
    RESET R.
    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 $i \in \{1, \ldots, n\}$.
    $\mathbf{X}[j, :] \leftarrow I_1 || I_2 \ldots, || I_n$.
  $\mathbf{x_a}[i] \leftarrow$ majority values of the i^{th} column of $\mathbf{X}$
  Use a b-bit ADC to convert $\mathbf{x_a}$ to a binary string $\mathbf{x}$.
  return $\mathbf{x}$

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 **x**. ϵ is the predetermined error threshold for the authentication.

Table 6.3 Single round interactive authentication for Protocol I

Prover($\mathbf{x}$, s)		Verifier(M, s, ϵ)
		— **RESET** M
		— **Precondition**(M,s)
— Find the current I_i for the precondition value s using **x**		
		— $c \xleftarrow{\$} \{i \in \{1, 2, \ldots n - s\} : n = \ell/b\}$
		— **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 **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 $s \leftarrow s + r$.

Protocol I requires 1T1M memristive cell 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.

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 condition. The non-linearity between the applied voltage and the corresponding resistance can create unique state transitions for a given memristor. When the resistive characteristics are different for two different memristors, different amount of voltage is required for obtaining the same level of resistance [23]. 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.

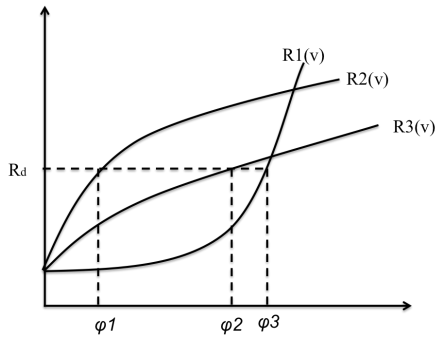

Fig. 6.5 For memristive device with different programming cycle different voltages are required to reach the same resistive level [23].

6.5.3 Security Analysis of Protocol I

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

Theorem 6.1: 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 6.2: 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 6.1: [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 [24] for a secure application. Helper data in error correction can leak information, and therefore, such construction must be carefully designed.

6.5.4 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 [7]:

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

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

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 6.6. 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.

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* [7].

Assumptions: For the authentication scheme described below, we assume that R1-R6 described at 6.5.1 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 6.4 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 $q_1, q_2, \ldots, q_{2^{k-1}}$
  Construct two Boolean matrices $\mathbf{B_0}$ and $\mathbf{B_1}$ of size $k \times 2^{k-1}$ such that, $\mathbf{B_0[i,j]} = 1$ iff $g_i \in e_j$ and $B_1[\mathbf{i,j}] = 1$ 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 6.4 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 6.6	
	— 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 $\mathbf{y}$ 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 $\mathbf{x}$.

Algorithm 6.5 Extraction of Correct Precondition Values

```
1:  procedure y ← ErrCorr(k, M)          ▷ M is a memristive array of length ℓ
2:      for all i ∈ {1, ..., ℓ} 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.

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 6.6 which can be used for reconstructing the shared secret.

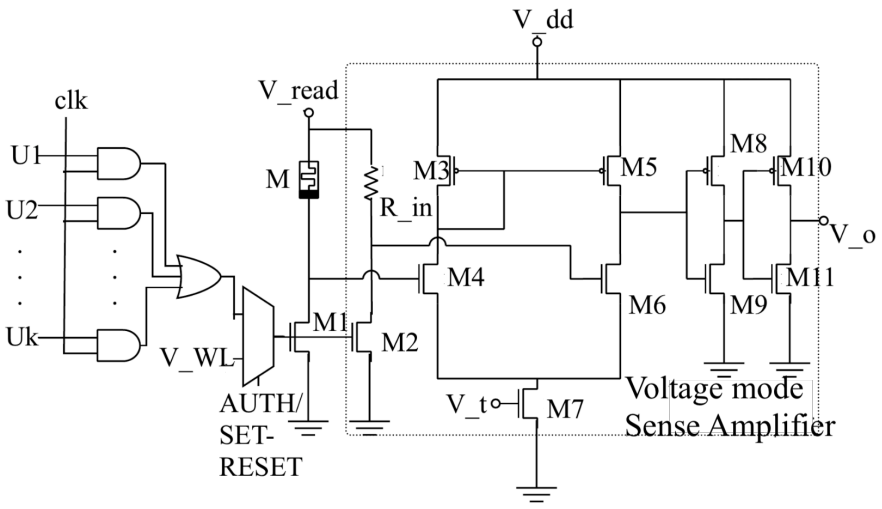

Fig. 6.6 Circuit for applying the pulsed input from k-users and the assisting voltage-mode sense amplifier-buffer circuit for reconstructing the secret.

6.5.5 Security Analysis of Multi User Authentication Protocol

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. [25]. 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.

In summary, 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.

6.6 Authentication and Spoofing Detection Using Hardware Clocks

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). Recently, there have been several demonstrations of weaknesses and vulnerabilities of GPS signals and GPS receivers [26–30]. 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.

In this section, 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.

6.6.1 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 [31].

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:

$$r_{true} = c\, t_{propagation} = \sqrt{(x_t - x_r)^2 + (y_t - y_r)^2 + (z_t - z_r)^2} \tag{6.4}$$

To solve equation 6.4 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:

$$r_{pseudo} = r_{true} - ct_r \tag{6.5}$$

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:

$$t_{sync} = t_{local} + t_r \tag{6.6}$$

In practice, this method provides accuracy in the order of 10 meters in position and nearly $0.1\mu s$ in time [32]. 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.

6.6.2 Hardware Based Signal Authentication and Spoofing Detection

Here, we will demonstrate how 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 6.7.

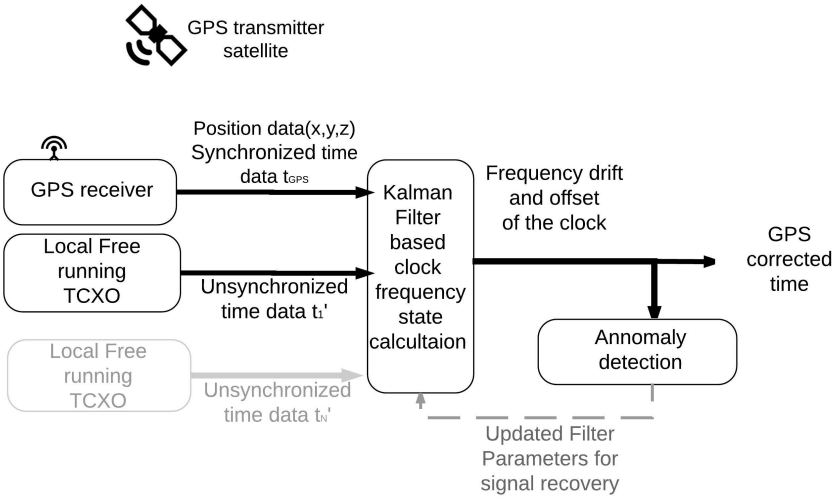

Fig. 6.7 Design of the secure GPS receiver with onboard authentication and spoofing detection mechanism. The receiver is equipped with a single (or multiple) free running temperature controlled oscillator(s). Kalman filter-based state estimation is used for calculating frequency states of this clock. 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.

6.6.3 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.

At a given time t, the deviation of a clock from ideal time can be expressed as [33]:

$$x(t) = x_0 + y_0 t + \frac{1}{2} D t^2 + \epsilon(t) \tag{6.7}$$

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.* [34] first proposed the idea of remote device fingerprinting using the uniqueness of frequency offset of hardware clocks. Since the publication of Khono's work [34], 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.

6.6.4 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 [35]:

$$\frac{dx_1}{dt} = y_1 + w_1; \frac{dy_1}{dt} = D_1 + w_2; \frac{dD_1}{dt} = w_3; \tag{6.8}$$

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 6.8 can be written as [35]:

$$\mathbf{X_n} = \mathbf{F_n X_{n-1}} + \mathbf{W_n} \tag{6.9}$$

$$\mathbf{\Xi_n} = \mathbf{H_n X_n} + \mathbf{V_n} \tag{6.10}$$

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 [35]:

$$\mathbf{F_n} = \begin{bmatrix} 1 & \Delta & \Delta^2/2 \\ 0 & 1 & \Delta \\ 0 & 0 & 1 \end{bmatrix} \tag{6.11}$$

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

$$\mathbf{Q} = \begin{bmatrix} q_1\Delta + q_2\frac{\Delta^3}{3} + q_3\frac{\Delta^5}{20} & q_2\frac{\Delta^2}{2} + q_3\frac{\Delta^4}{8} & q_3\frac{\Delta^3}{6} \\ q_2\frac{\Delta^2}{2} + q_3\frac{\Delta^4}{8} & q_2\Delta + q_3\frac{\Delta^3}{3} & q_3\frac{\Delta^2}{2} \\ q_3\frac{\Delta^3}{6} & q_3\frac{\Delta^2}{2} & q_3\Delta \end{bmatrix} \tag{6.12}$$

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 [35]. 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.

6.6.5 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 $\mathbf{H_n} = [1, 0, 0]$. $\mathbf{V_n}$ 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 [36] is given by the following equations:

Prediction Step:

$$\mathbf{m_{n|n-1}} = \mathbf{F_n m_{n-1|n-1}} \tag{6.13}$$

$$\mathbf{P_{n|n-1}} = \mathbf{F_n P_{n-1|n-1} F_n^T} + \mathbf{Q} \tag{6.14}$$

Update Step:

$$\mathbf{K_n} = \mathbf{P_{n|n-1} H_n^T (H_n P_{n|n-1} H_n^T} + \mathbf{R})^{-1} \tag{6.15}$$

$$\mathbf{m_{n|n}} = \mathbf{m_{n|n-1}} + \mathbf{K_n}(\mathbf{\Xi_n} - \mathbf{H_n m_{n|n-1}}) \tag{6.16}$$

$$\mathbf{P_{n|n}} = \mathbf{P_{n|n-1}} - \mathbf{K_n H_n P_{n|n-1}} \tag{6.17}$$

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 $\mathbb{I}^{3\times3}$.

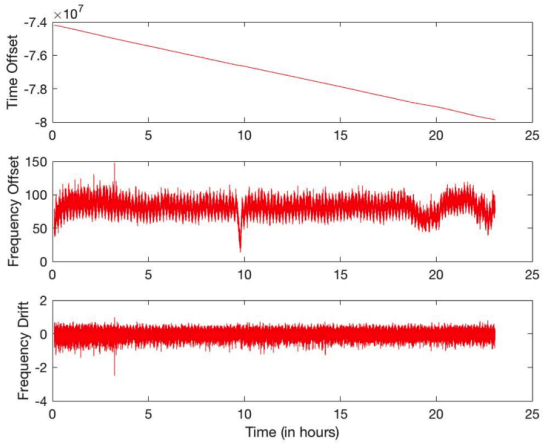

Fig. 6.8 Example time offset, frequency offset and frequency drift for a crystal oscillator with respect to a stable GPS clock.

6.6.6 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 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,

$$\mathbf{S_{n-1}} = \mathbf{H_n P_{n|n-1} H_n^T} + \mathbf{R} \tag{6.18}$$

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

$$p(m_{i,n}|m_{i,n-1},\ldots m_{i,n-k}) = \frac{1}{\sqrt{2\pi\sigma^2_{i,n-1}}} e^{(-\frac{(m_{i,n}-\bar{m}_{i,n-1})^2}{\sigma^2_{i,n-1}})} \tag{6.19}$$

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,

$$z_{i,n} = \alpha z_{i,n-1} + (1-\alpha)\ln(p(m_{i,n})) \tag{6.20}$$

with α as the smoothing factor.

6.6.7 Spoofing Detection and Results

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.

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 6.9. During the attack, the attacker keeps maintains the temporal shift. When the attack ends, there is a sudden jump in the time offset which is represents the recovery of the authentic signal by the receiver.

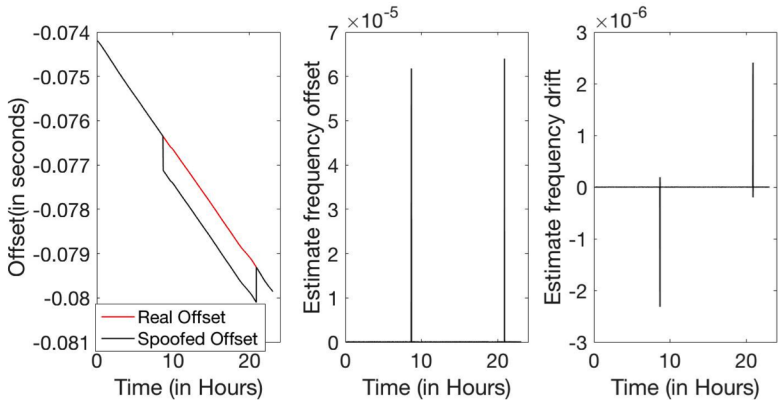

Fig. 6.9 Example of a temporal shift injection attack.(a) Clock offset of a free running TCXO with respect to a GPS reference. A temporal shift injection attack is initiated at 5000 seconds. There a sudden jump in the time offset which is corrected at the end of the attack; (b) Estimation of the frequency 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 as described in [37]. 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 6.10(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).

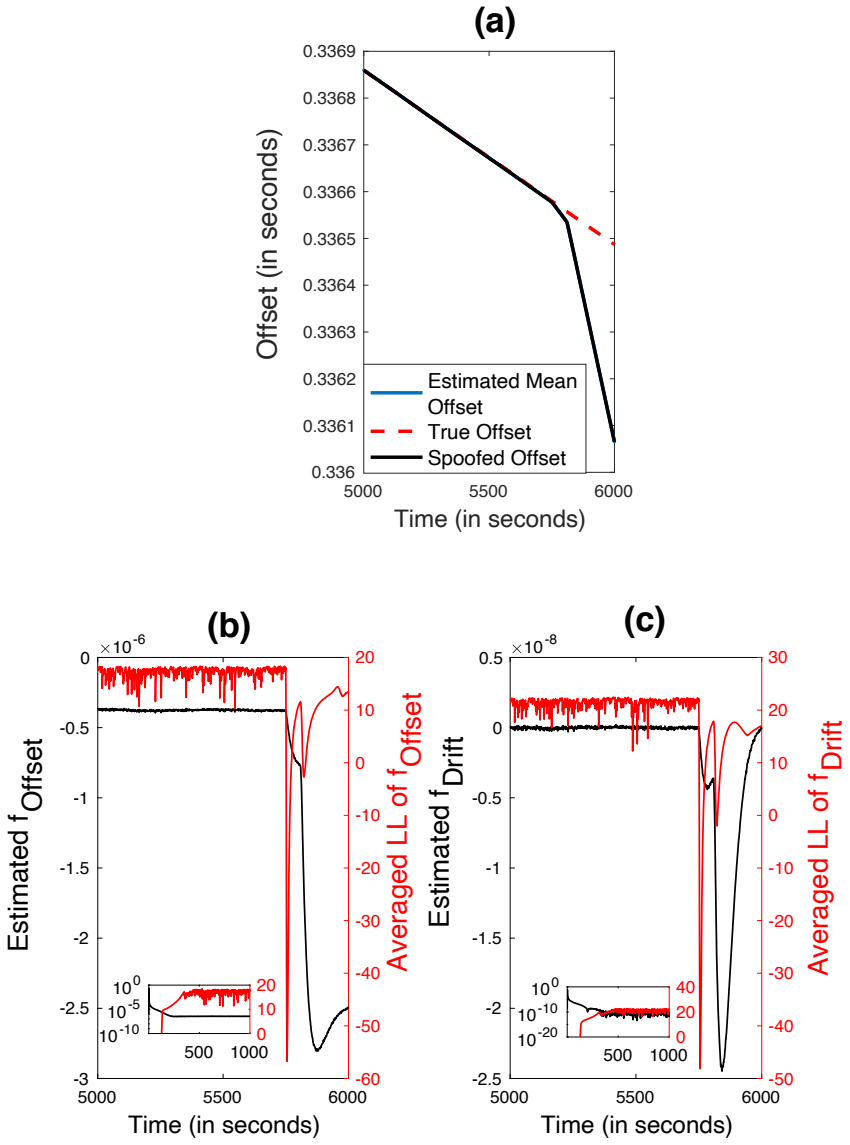

Fig. 6.10 Example of a replay and meaconing attack. (a) Clock offset of a free running TCXO with respect to a GPS reference. A spoofing attack is initiated at 5130 seconds. The estimated time offset faithfully follows the spoofed signal as there is no sudden jump in the time offset;(b) Estimation of 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 6.6.2. 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 6.10, 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.

6.6.8 Analysis of Hardware Dependent Signal Authentication

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 6.6.3 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 $\mathbf{Q}$.

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 construction 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.

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.

In summary, we present a design for integrating data-level spoofing detection with an existing GPS-based timing system in this section. 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.

6.7 Conclusions

In this chapter, we have demonstrated several examples 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.

References

1. G. Qu and L. Yuan, "Design things for the internet of things—an eda perspective," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2014, pp. 411–416.
2. A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.
3. M. T. Arafin and G. Qu, "Rram based lightweight user authentication," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 139–145. [Online]. Available: http://dl.acm.org/citation.cfm?id=2840819.2840839
4. A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
5. G. R. Blakley, "Safeguarding cryptographic keys," *Proc. of the National Computer Conference1979*, vol. 48, pp. 313–317, 1979.
6. D. R. Stinson, "An explication of secret sharing schemes," *Designs, Codes and Cryptography*, vol. 2, no. 4, pp. 357–390, 1992.
7. M. Naor and A. Shamir, "Visual cryptography," in *Workshop on the Theory and Application of Cryptographic Techniques*. Springer, 1994, pp. 1–12.
8. N. Beckmann and M. Potkonjak, "Hardware-based public-key cryptography with public physically unclonable functions," in *International Workshop on Information Hiding*. Springer, 2009, pp. 206–220.
9. R. Maes, *Physically Unclonable Functions*. Springer, 2013.
10. R. Maes and I. Verbauwhede, "Physically unclonable functions: A study on the state of the art and future research directions," in *Towards Hardware-Intrinsic Security*. Springer, 2010, pp. 3–37.

11. G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th annual Design Automation Conference*. ACM, 2007, pp. 9–14.
12. M. Naor and B. Pinkas, "Visual authentication and identification," in *Annual International Cryptology Conference*. Springer, 1997, pp. 322–336.
13. M. Elgebaly and M. Sachdev, "Variation-aware adaptive voltage scaling system," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 5, pp. 560–571, 2007.
14. N. Banerjee, G. Karakonstantis, and K. Roy, "Process variation tolerant low power dct architecture," in *Proceedings of the conference on Design, automation and test in Europe*. EDA Consortium, 2007, pp. 630–635.
15. J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European Test Symposium (ETS)*. IEEE, 2013, pp. 1–6.
16. R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "Macaco: Modeling and analysis of circuits for approximate computing," in *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 2011, pp. 667–673.
17. F. Ganji, S. Tajik, and J.-P. Seifert, "Why attackers win: on the learnability of xor arbiter pufs," in *International Conference on Trust and Trustworthy Computing*. Springer, 2015, pp. 22–39.
18. J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh *et al.*, "Freepdk: An open-source variation-aware design kit," in *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*. IEEE, 2007, pp. 173–174.
19. M. T. Arafin, M. Gao, and G. Qu, "Volta: Voltage over-scaling based lightweight authentication for iot applications," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017, pp. 336–341.
20. J. Rajendran, G. S. Rose, R. Karri, and M. Potkonjak, "Nano-PPUF: A Memristor-based Security Primitive," in *Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2012, pp. 84–87.
21. J. B. Wendt and M. Potkonjak, "The Bidirectional Polyomino Partitioned PPUF as a Hardware Security Primitive," in *Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2013, pp. 257–260.
22. J. Rajendran, R. Karri, J. B. Wendt, M. Potkonjak, N. R. McDonald, G. S. Rose, and B. T. Wysocki, "Nanoelectronic Solutions for Hardware Security." *IACR Cryptology ePrint Archive*, vol. 2012, p. 575, 2012.
23. H. Kim, M. P. Sah, C. Yang, and L. O. Chua, "Memristor-based multilevel memory," in *Cellular nanoscale networks and their applications (CNNA), 2010 12th international workshop on*. IEEE, 2010, pp. 1–6.
24. Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2004, pp. 523–540.
25. G. Horng, T. Chen, and D.-S. Tsai, "Cheating in visual cryptography," *Designs, Codes and Cryptography*, vol. 38, no. 2, pp. 219–236, 2006.
26. C. Bonebrake and L. R. O'Neil, "Attacks on GPS time reliability," *Security & Privacy, IEEE*, vol. 12, no. 3, pp. 82–84, 2014.
27. T. E. Humphreys, B. M. Ledvina, M. L. Psiaki, B. W. OâHanlon, and P. 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*, vol. 55, 2008, p. 56.
28. A. Jafarnia-Jahromi, A. Broumandan, J. Nielsen, and G. Lachapelle, "Gps vulnerability to spoofing threats and a review of antispoofing techniques," *International Journal of Navigation and Observation*, vol. 2012, 2012.
29. M. G. Kuhn, "Signal authentication in trusted satellite navigation receivers," in *Towards Hardware-Intrinsic Security*. Springer, 2010, pp. 331–348.
30. N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful GPS spoofing attacks," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 75–86.

31. P. Misra and P. Enge, *Global Positioning System: Signals, Measurements and Performance Second Edition*. Lincoln, MA: Ganga-Jamuna Press, 2006.
32. P. H. Dana and B. M. Penrod, "The role of GPS in precise time and frequency dissemination," *GPS World*, pp. 38–43, 1990.
33. D. W. Allan, "Time and frequency(time-domain) characterization, estimation, and prediction of precision clocks and oscillators," *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 34, no. 6, pp. 647–654, 1987.
34. T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *Dependable and Secure Computing, IEEE Transactions on*, vol. 2, no. 2, pp. 93–108, 2005.
35. C. A. Greenhall, "A review of reduced kalman filters for clock ensembles," *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, vol. 59, no. 3, pp. 491–496, 2012.
36. S. S. Haykin, *Adaptive filter theory*. Pearson Education India, 2008.
37. D. P. Shepard, T. E. Humphreys, and A. A. Fansler, "Evaluation of the vulnerability of phasor measurement units to GPS spoofing attacks," *International Journal of Critical Infrastructure Protection*, vol. 5, no. 3, pp. 146–153, 2012.