November 2016

Intrinsic Functions for Securing CMOS Computation: Variability, Modeling and Noise Sensitivity

Xiaolin Xu
Electrical and Computer Engineering Department

INTRINSIC FUNCTIONS FOR SECURING CMOS COMPUTATION: VARIABILITY, MODELING AND NOISE SENSITIVITY

A Dissertation Presented

by

XIAOLIN XU

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2016

Electrical and Computer Engineering

INTRINSIC FUNCTIONS FOR SECURING CMOS COMPUTATION: VARIABILITY, MODELING AND NOISE SENSITIVITY

A Dissertation Presented

by

XIAOLIN XU

Approved as to style and content by:

Wayne P. Burleson, Chair

Daniel E. Holcomb, Member

Christof Paar, Member

Krista J. Gile, Member

Prof. Christopher V. Hollot, Department Head
Electrical and Computer Engineering

DEDICATION

To Shuo and our parents.

ACKNOWLEDGMENTS

I would like to express my deep gratitude to all those who made this thesis possible. I cannot overstate my gratitude to my adviser, Professor Wayne Burleson. I appreciate so much the opportunity he gave me to pursue my research under his supervision, without whose encouragement and guidance, this thesis would not have been possible. He is the best adviser and teacher I could have wished for, he mentored me with his enthusiasm, his inspiration, and his efforts to develop and polish my thesis. He has always been a source of inspiration and knowledge throughout my years at UMass.

I wish to thank Professor Daniel Holcomb, for the collaboration during my recent two years at UMass and everything that will follow; for great ideas, insights that make him smile, and advice about research and life. All the discussions we had during the past several years were extremely crucial in developing this thesis. I owe much of my growth as a researcher to him.

I thank Professor Christof Paar for selecting me as the Teaching Assistant (TA) of *Crypto Engineering*, working with him was a great experience that brought together cryptography and computer engineering. Thanks for the unselfish help, insights, feedback, and for making the job search a collaborative effort.

I also want to thank Professor Krista Gile for serving as my dissertation committee member, reading the dissertation and providing valuable insights into this thesis.

During my Ph.D study, I have spent great summer time at Cryptography Research Inc. (*CRI*), hereby I thank Professor Daniel Holcomb again for his recommendation, I also thank Scott Best for the collaboration on super "COOL" industrial hardware security research. I also spent four highly productive months in the WLAN group

at Broadcom Ltd. as an intern, I thank every group member for their help and collaboration.

Finally, this thesis is dedicated to my wife Shuo, and our parents. *So many* thanks for your love, being gracious and understanding, I would have nothing if not for you and our parents.

ABSTRACT

INTRINSIC FUNCTIONS FOR SECURING CMOS COMPUTATION: VARIABILITY, MODELING AND NOISE SENSITIVITY

SEPTEMBER 2016

XIAOLIN XU

B.E., UNIVERSITY OF ELECTRONIC SCIENCE & TECHNOLOGY OF CHINA

M.S., UNIVERSITY OF ELECTRONIC SCIENCE & TECHNOLOGY OF CHINA

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Wayne P. Burleson

A basic premise behind modern secure computation is the demand for lightweight cryptographic primitives, like identifier or key generator. From a circuit perspective, the development of cryptographic modules has also been driven by the aggressive scalability of complementary metal-oxide-semiconductor (CMOS) technology. While advancing into nanometer regime, one significant characteristic of today's CMOS design is the random nature of process variability, which limits the nominal circuit design. With the continuous scaling of CMOS technology, instead of mitigating the physical variability, leveraging such properties becomes a promising way. One of the famous products adhering to this *double-edged sword* philosophy is the Physically Unclonable Functions (PUFs), which extract secret keys from uncontrollable manufacturing variabilities on integrated circuits (ICs). However, since PUFs take advantage of microscopic process

variations, thus many specialized issues including variability, modeling attacks and noise sensitivity need to be considered and addressed.

In this dissertation, we present our recent work on PUF based secure computation from three aspects: variability, modeling and noise sensitivity, which are deemed the foundations of our study. Moreover, we found that the three factors coordinate with each other in our study, for example, the modeling technique can be utilized to improve the unsatisfied reliability caused by noise sensitivity, quantifying the variability can effectively eliminate the impact from noise, and modeling can help with characterizing the physical variability precisely.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Sophisticated semiconductor industry facilitates the design, manufacturing and fabrication of integrated circuits (IC). One significant characteristic of the complementary metal-oxide-semiconductor (CMOS) technology development is the persistent physical scalability and ever-increasing usability. The scaling of CMOS techniques has helped people to integrate more transistors into smaller sized chips, and made it possible to realize ultra-large-scale CMOS design. This advancement favors the development of electronic devices like smart phones, tablets and portable medical devices, but also proposes new challenges. One example is that designers have to face new design limitations like process variations, lower supply voltage, temperature sensitivity and device aging, known as the PVTA issues. From another perspective, this also proposes new requirements for the hardware security study, since malicious hardware devices can now devastate commerce, government operation and even national defense in a larger scale.

To enhance the computation security, many infrastructures like computer or network server have been equipped with hardware security modules, such as a plug-in card, which manages the digital keys for strong authentication or crypto-processing. In contrast, due to the limited silicon or energy budget, strengthening the security of many small embedded devices like implanted medical devices seems not an easy job. As we are stepping into the *Internet of Things (IoT)* era, such small devices are ubiquitous and all connected with the IoT network. In particular, many embedded systems are now carrying sensitive information like user profiles or credentials, thus rendering them

as attractive targets for malicious attackers. Therefore, in the proliferation of small devices in the IoT era, it is urgent for people to properly address the security challenge and protect security-critical information on these devices from various attacks.

While tiny embedded processors are becoming the central processing engines in many portable devices, lightweight cryptographic primitives is also gaining more attention in hardware security research. Since most of the embedded systems pose tight constraints on area, energy and functionality, significant effort has been spent to integrate more transistors and functionalities onto smaller chips. One challenge in this process is that while the CMOS technology scales into nanometer regime, designers have to consider the impact from like design uncertainty, which is introduced by aggressive scalability. One main cause of the design uncertainty is the random dopant fluctuation (RDF) [7], which introduces variabilities in the number of dopant atoms in transistor channels and changes the threshold voltage of a transistor. Based on current fabrication technology, totally eliminating such process variations is still a difficult mission. In this scenario, constructively utilizing such phenomenon becomes a meaningful direction. One well-known product based on this philosophy is Physical Unclonable Function (PUF), which was firstly introduced in [102] [26] [77].

PUF can be used to substitute many conventional applications like key generator or chip identifier. For example, one of the conventional solutions to identify IC is using static identifiers (IDs) stored in non-volatile memory. However, such IDs are usually composed by digital numbers and thus vulnerable to cloning. Due to the uniqueness, physical characteristics was proposed as a new resource to build device IDs, such physical features can be viewed as the fingerprinting of an IC, which has the advantage of robust immutability and resistance to cloning and tampering. While being employed as an identifier, PUF is originally described as a physically obfuscated key [27], and more recently as a weak PUF [31].

As many other electronic devices, PUF design is promoted by the scaled CMOS technology, but also challenged. In this dissertation, we present our contribution on PUF based hardware security in nanometer CMOS regime. More specifically, we focus on advancing reliability, security of PUFs from three perspectives: variability, modeling attacks and noise sensitivity, which are viewed as the foundations for our study.

1.1 Challenges and Opportunities

1.1.1 Variability

Though frequently suspected and challenged, the development of CMOS technology has been continually driven by Moore's Law over the past decades [112] [2]. Unfortunately, as the CMOS technology steps into nanometer scale, as depicted in Fig. 1.1, the fabrication uncertainty of device has become a big problem. One important cause of the uncertainty problem is the inability to set all device parameters to the desired values exactly. The observable characterization of fabrication uncertainty is the *mismatched* device parameters, i.e., the different threshold voltage between transistors [107] [41]. Variability on modern chips can be classified into three categories: *1*) the variabilities between different dies, i.e., the *inter-die* variability have large correlation with each other, which affects all transistors on the same die; *2*) a portion of transistors one the same die are impacted by the *intra-die* variability, thus are correlated over smaller distance; and *3*) random issues like dopant fluctuations affect each transistor randomly.

From a device perspective, CMOS design based on the scaled technology is now affected more by the variabilities like process variations, lower supply voltage, temperature and aging issues, also known as PVTA variations. Since security designs are usually embedded in other circuits, thus are also challenged by the persistent physical scalability and demand for increasing usability. PUF, which was firstly introduced in

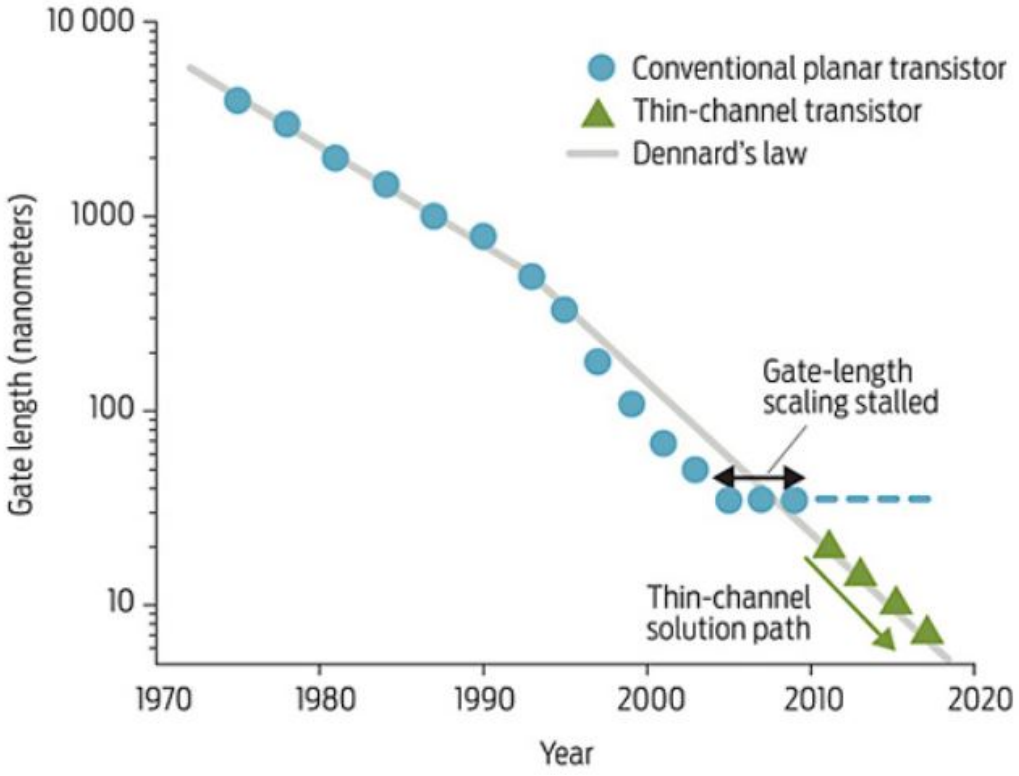


Figure 1.1: Transistor scaling trends [2].

[102], is a well-known example that constructively leverages such physical features for security purpose.

Over the past 15 years, different variants or types of PUFs and applications based on PUFs have emerged, as summarized in Fig. 1.2, see [91] for an overview. They all share the above features of being a disordered structure, possessing physical unclonability, and exhibiting some form of challenge-response mechanism. More specifically, PUF is a function which leverages the physically variabilities like delay variations [56], power-up state of SRAMs [36] and even optical variations [77] [89] [83]. During the last ten years, PUFs have established themselves as an alternative to conventional security approaches [28] [77]. In a nutshell, a PUF is a disordered, at least partly randomly structured physical system. Due to its random structure that is caused by uncontrollable, small-scale manufacturing variations, it is physically

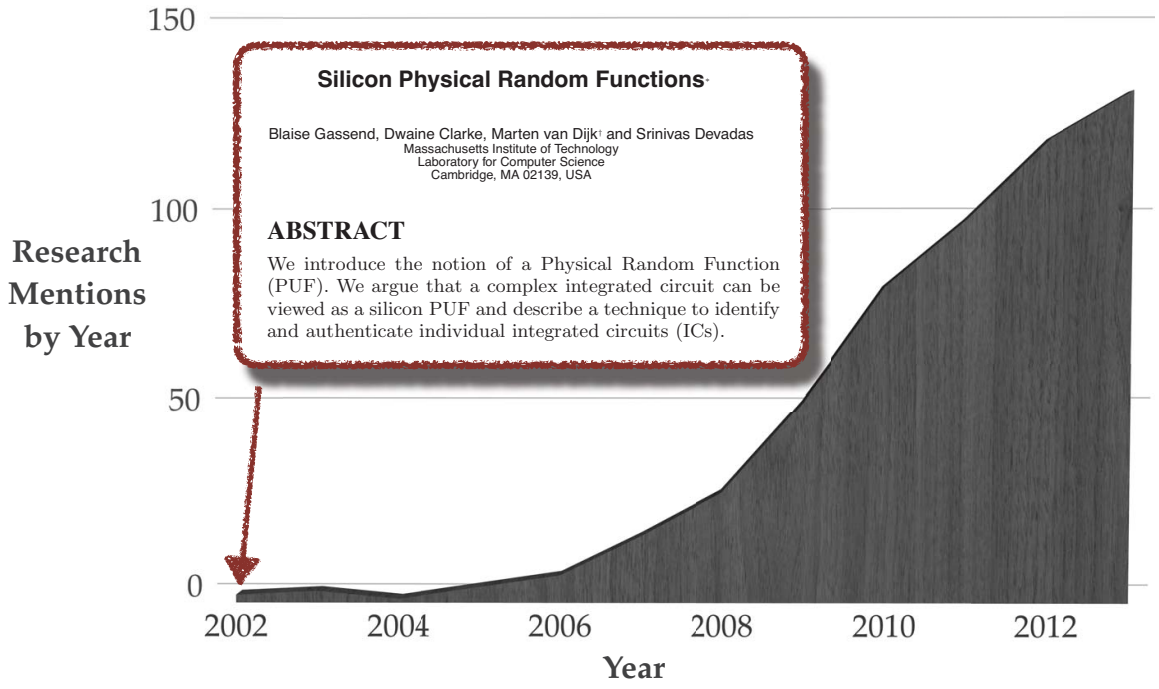


Figure 1.2: PUF is becoming a hot topic in hardware security research during the past decade [91].

unclonable, i.e., no two specimens can be produced that are physically exactly identical. This limitation applies to both the original manufacturer and to other, potentially adversarial, parties. All PUFs have one basic functionality in common, namely some *challenge-response* mechanism: They can be triggered by signals that are commonly denoted as *challenges* C_i. Upon excitation by a challenge C_i, they react by producing a response R_{C_i} that depends on their internal disorder and usually also on the challenge itself. The tuples (C_i, R_{C_i}) are called the *challenge-response pairs (CRPs)* of the PUF.

The two main PUF-types are often denoted as strong and weak PUFs [93] [90]. Strong PUFs are built on customized circuits, which have a very large number of possible challenges, too many to read out all corresponding CRPs in feasible time. Their challenge-response mechanism should be complex in the sense that it is hard to derive unknown CRPs from a set of known CRPs. Strong PUFs are usually employed with a publicly accessible CRP interface, i.e., anyone holding the PUF or the PUF

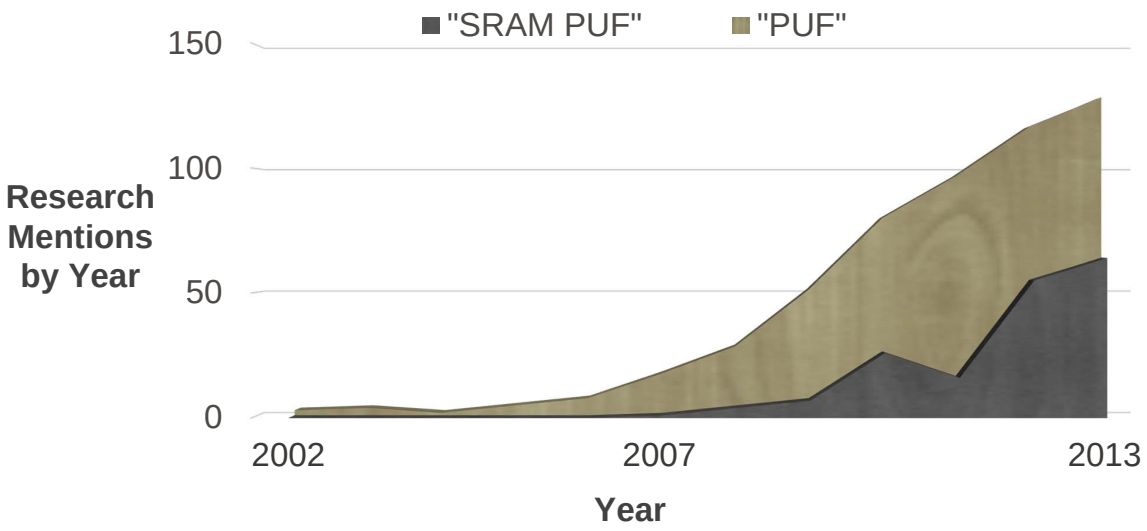
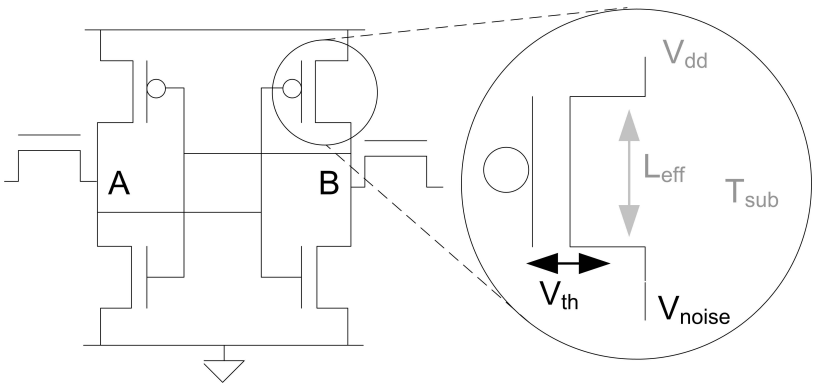


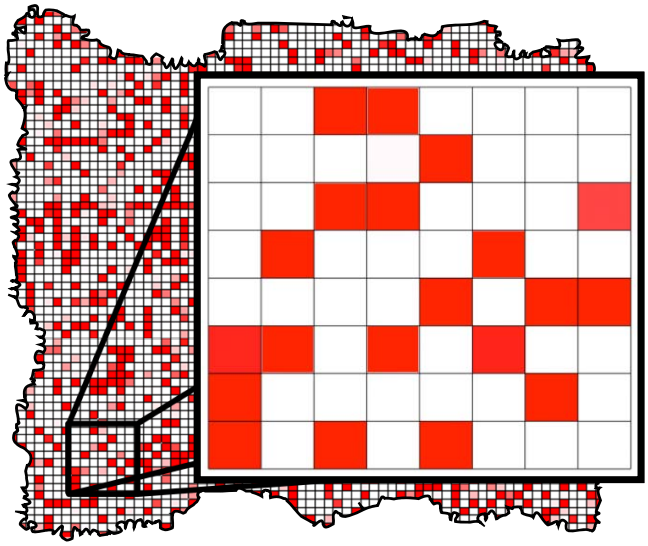
Figure 1.3: SRAM PUF has been gaining a lot of attention ever since it was proposed [91].

embedding hardware can apply challenges and read out responses. The lack of access restriction mechanisms on strong PUFs is therefore a key difference from weak PUFs. In recent years, strong PUFs have turned out to be a very versatile cryptographic and security primitive: First of all, by using a fixed set of challenges, they can be employed for internal key derivation, just like weak PUFs. But they can do more: They can also implement a host of advanced cryptographic protocols, ranging from identification [77] [61] to key exchange [110] [11], key management [45] to oblivious transfer [11], and more recently, as a keyless secure sensor [92].

In contrast, weak PUFs possess essentially a single, fixed challenge C. They are mainly used for internal key derivation in security hardware. The underlying security assumption is that attackers must not be able to access the internal response of the PUF. The most well-known weak PUF example is the SRAM PUF proposed by Holcomb et al. [36], by reading out the power-up state of the SRAM cells [31] [37]. Though invented later than strong PUFs, SRAM PUF, as the most well-known example in weak PUF category, has gained a lot of attention in hardware security



(a) An ideal SRAM cell is a symmetric architecture.



(b) An example fingerprint based on SRAM PUFs.

Figure 1.4: A 64-bit fingerprint based on SRAM PUFs. Reproduced from Holcomb et al. [36] with permission.

study, as Fig. 1.3. Since SRAM circuitry is widely used as storage module, this makes SRAM PUF a ideal platform to generate secret keys or device fingerprint, as shown in Fig. 1.4b, a 64-bit fingerprint based on SRAM PUF.

1.1.2 Modeling

Modeling is a powerful method for human beings to learn the mechanisms of nature, especially to simulate complex theories or systems [24] [48]. In most recent literatures about PUFs, modeling is mainly employed to play a "bad" role: help with

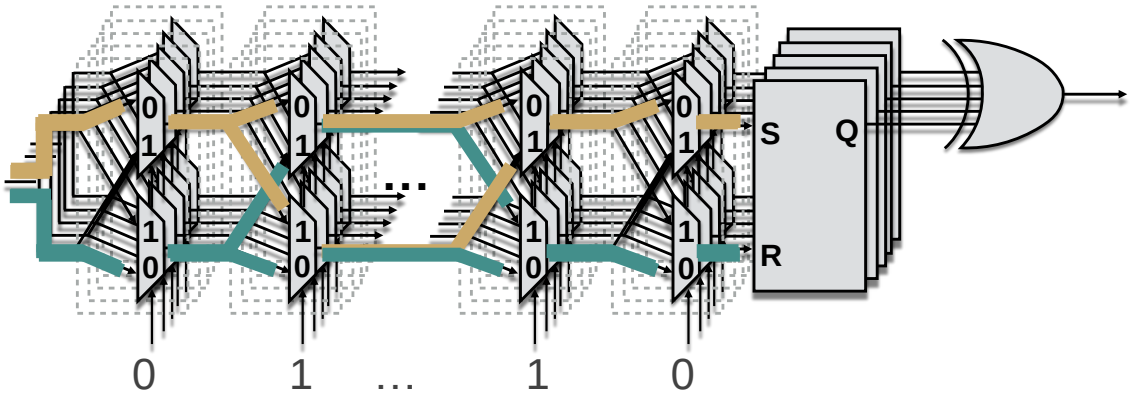


Figure 1.5: XOR Arbiter PUF, that is composed by n Arbiter PUFs in parallel to improve the resistance against modeling attacks [91].

predicting the behavior of PUFs, i.e., attacking PUFs. The usual working flow of *"modeling attacks"* is that, an adversary collects a subset of CRPs, uses them to train a machine learning (ML) algorithm, and later employs the model produced by the ML algorithm to predict unknown CRPs. Even though many possible applications make the secure construction of secure strong PUFs a worthwhile and rewarding research target. Unfortunately, it is a non-trivial one, since a large number of powerful attacks on some first-generation electrical strong PUFs have been published recently, including Machine Learning modeling attacks [93] [90]; side-channel attacks [95] [94]; and also optical characterization techniques [106]. Most of these attacks target the first electrical strong PUF, the so-called Arbiter PUF [28] [77] and variants thereof, for example XOR Arbiter PUFs (Fig. 1.5) and Feed-Forward Arbiter PUFs.

Though ML modeling techniques are challenging the security of strong PUFs, but also making them the most powerful tool to learn the internal variability of PUFs until now. That is, ML modeling technique makes it possible for people to quantify the sub-nanometer or sub-picosecond level physical variabilities. In this thesis, we propose two constructive utilizations of Machine Learning techniques to help with building of robust and reliable PUFs. One of them is simulating the microscopic

process variations of PUFs. Since PUFs are leveraging microscopic process variations, thus it is impossible to measure many important features directly, because that any external probing will introduce bias to PUF circuitry. For example, it is extremely difficult to measure the delay length of Arbiter PUFs, which is of pico-second level with current instruments.

Another constructive usage of computer modeling in this thesis is speeding up the circuit simulation. The behavior of a transistor or circuit is determined by features including transistor size, temperature, supply voltage, etc. Thus it will be a waste of time to repeatedly simulating one interested feature while varying the input patterns. In this thesis, we propose to use machine learning techniques to build up models and speed up the simulation of data retention voltage (DRV) for SRAM circuits, which is $2.2e6$ times faster than using simulation like *Hspice*.

1.1.3 Noise Sensitivity

Many PUF architectures have been explored in literature, including strong PUFs like Arbiter PUF [26] [56] [29], Ring Oscillator (RO) PUF [102], Lightweight PUF [70], and weak PUFs like SRAM PUF [36] [31]. However, since all of the proposed PUFs are built on microscopic physical variability, thus are sensitive to noise. Reliability (see definition in Sec. 1.2.2) is an important metric to evaluate PUFs, which reflects their ability to produce the same response for a particular challenge despite the existence of noise, for weak PUFs like SRAM PUF, reliability refers to the constant power-up states across different environmental conditions. Even though many meaningful applications on PUFs have been proposed, reliability is still a problem that hinders their practical applicability. This is because that environmental noise like power supply (V_{DD}) and temperature (T) variations is ubiquitous, and can easily overcome the process variabilities of a silicon device.

The environmental noise can not be avoided, since even every silicon device itself will dissipate heat after operating for a while. Thus, the output of a PUF depends not only on the challenge ($\mathbf{C}$) but also on transient environmental conditions (e). If denoting the input/output relationship of a PUF p with a function f_p, a PUF operating in different conditions may possibly generate a flipped response (R) to the same challenge vector ($\mathbf{C}$), as expressed in Eq.1.1.

$$R = f_p(\mathbf{C}, e) \tag{1.1}$$

PUFs are mainly proposed to constructing secret keys or serve as a lightweight device identifier. According to [49], the security of most current cryptographic techniques rely on the robustness of secret keys. Thus if using PUFs to identify hardware devices or constructing secret keys, fingerprint observations must be consistent over time and across different environmental conditions. A fundamental concern in PUFs is to minimize the impact of noise and environmental fluctuations while still being sensitive to the microscopic variations that make each device unique. A common way of minimizing the impact of noise and environment is to use differential circuits. Yet small variations in the fingerprint of a device are inevitable, and much effort is spent on error correction of somewhat-unreliable fingerprints or PUF outputs, or adding significant extra circuitry for calibration [74]. However, error correcting codes and calibration circuitry are expensive in terms of the number of raw bits and silicon resource required to create a reliable key, and more so if a large number of errors must be correctable.

In this thesis, we present our work that studies the noise problems on PUFs. We propose a novel clockless sequential PUF structure that performs autonomous majority voting to improve reliability.

1.2 Terminologies of PUF

1.2.1 CRPs

Challenge and response are commonly used terms in computer security, which stand for the information exchanged between two parties, specifically, challenge means the proposed question by one party, while response is the answer provided by the other side. Only while the question is correctly answered, the two parties can authenticate each other, for example, supplying the correct password before logging in. In PUF study, these two terms are borrowed to denote the input and output respectively. As suggested by the names, PUFs can be used in two-party authentication: a set of challenges are applied and the corresponding responses are stored in advance, only the genuine PUF holder can provide the correct responses while being queried. Different PUF architectures support various amount of CRPs, according to the number of usable CRPs, PUFs can be classified into two classes: strong and weak PUFs [33] [93] [90]. For example, the more complex physical variability makes it possible for strong PUFs to have a large number of usable CRPs and allow free querying of responses.

1.2.2 Reliability

Since PUFs leverage microscopic process variations, thus are sensitive to environmental noise like slight temperature or supply voltage fluctuations. Due to the sensitivity, a PUF may not produce consistent response for the same challenge under different environmental conditions. Reliability of a PUF is defined as its capability to reproduce the same responses while digesting the same challenge vectors. This feature is very important in evaluating a PUF design, since most application based on PUFs are related with security or privacy, such as authentication or key generation. A widely used measure for reliability is intra-chip hamming distance as denoted in Eq. 1.2 [99] [51], where the $\mathbf{C} = \{C_0, C_2, C_2 \dots C_{n-1}\}$ stand for a group of n challenge vectors. The intra-chip hamming distance $Dist_{intra}(\mathbf{C})$ computes the proportion of flipped

responses in $\mathbf{R}_{\mathbf{C},e}$ (responses under random environmental conditions e), comparing with that in $\mathbf{R}_{\mathbf{C},ne}$ (responses under nominal environmental condition ne).

$$Dist_{intra}(\mathbf{C}) = HD(\mathbf{R}_{\mathbf{C},ne}, \mathbf{R}_{\mathbf{C},e}) \tag{1.2}$$

1.2.3 Uniqueness

Since PUF utilizes the physical variability from manufacturing, thus is supposed to be distinguishable from each other. The metric measuring this capability of PUFs is uniqueness, that is characterized by the inter-chip Hamming Distance $Dist_{inter}(\mathbf{C})$.

$$Dist_{inter}(\mathbf{C}) = HD(\mathbf{R}^i_{\mathbf{C}}, \mathbf{R}^j_{\mathbf{C}}) \tag{1.3}$$

After digesting the same challenge vectors $\mathbf{C} = \{C_0, C_2, C_2 \dots C_{N-1}\}$, two PUF instances i and j will generate responses $\mathbf{R}^i_{\mathbf{C}}$ and $\mathbf{R}^j_{\mathbf{C}}$ respectively. The uniqueness between these two PUF instances is calculated as Eq. 1.3, where $Dist_{inter}(\mathbf{C})$ calculates the hamming distance between $\mathbf{R}^i_{\mathbf{C}}$ and $\mathbf{R}^j_{\mathbf{C}}$. Based on this definition, there will be totally $N \times (N-1)/2$ uniqueness values for a group of N PUF instances. According to the *central limit theorem*, if N is large enough, the distribution of uniqueness values should be following a *Gaussian* style, and the ideal uniqueness should be close to 50%.

1.2.4 Uniformity

Frequency prediction is one possible attacking method on security systems, in which the attacker collects outputs from known queries in order to build a probability distribution for each output. An ideal security system is supposed to produces each output bit as 0 or 1 with a probability of 0.5, which eliminates the possibility of frequency prediction. For a hardware system like PUF, the probability of generating

0 or 1 is mainly determined by the circuit bias, uniformity is a metric of PUFs that denotes such bias in a PUF design.

$$HW(\mathbf{C}) = \sum_{i=0}^{N-1}(R_i) \tag{1.4}$$

In practical, uniformity is usually denoted by fractional Hamming Weight, i.e., the proportion of "1" responses in a set of responses, as Eq. 1.4. $\mathbf{C} = \{C_0, C_1, C_2 \ldots, C_{n-1}\}$ denotes a group of n challenge vectors applied on PUF, and R_i corresponds to the response for each single challenge C_i. A good PUF is expected to have equal number of 0 and 1 responses, i.e., a uniformity of 0.5, indicating the high unpredictability of the design.

1.3 Background and Motivation

In previous sections, we introduced the basic definition of PUFs and some related terminologies. We referred PUF study from three aspects: *1*) physical variability challenges conventional CMOS design but favors hardware security primitives like PUFs; *2*) modeling attacks is a main threat for PUF based applications, but also a promising tool to study PUFs deeply; and *3*) noise sensitivity limits the practical applicability of PUFs by degrading their reliability. It it notable that, since all of the three directions are about PUFs, thus there is intrinsic link between them, such as the noise sensitivity is the cause of reliability problem. In the following paragraphs, we will review some background information of the three aspects.

A wide variety of PUFs and fingerprints based on custom or pre-existing integrated circuit components have been proposed. The identifying features used by custom designs include MOSFET drain-current [60], timing race conditions [26], and the digital state taken by cross-coupled logic after a reset [101]. IC identification based on pre-existing circuitry is demonstrated using SRAM power-up state [37] [31], and

physical variations of flash memory [78]. Lee et al. [57] derive a secret key unique to each IC using the statistical delay variations of wires and transistors across ICs. Circuit-level techniques for increasing the reliability of SRAM PUFs are explored by Bhargava et al. [10]. An experimental evaluation of low-temperature data remanence on a variety of SRAMs is provided by Skorobogatov [100], and SRAM remanence in RFID has been studied by Saxena and Voris as a limitation to SRAM-based true random number generation [97].

Previous works [109] have used error correction to construct secret keys from noisy PUF sources; however, these approaches are expensive in their required number of gates. Suh et al. use a BCH code to correct 21 errors among 127 raw bits to create a 64-bit key [104]. Guajardo et al. [31] derive a 278-bit secret key from 1,023 bits of power-up SRAM state using a BCH code that can correct up to 102 errors. Maes et al. [67] introduce an SRAM helper data algorithm to mask unreliable bits using low-overhead post-processing algorithms. Recently, Yu et al. [126] proposed the use of index-based syndrome (IBS) coding for deriving reliable key bits from PUF outputs. A notable feature of error correction using IBS coding in PUFs is that the syndrome does not leak information about the encoded bits. Hiller et al. extend IBS coding for SRAM PUFs [35]. Van Herrewege et al. [111] have designed a new lightweight authentication scheme using PUFs that does not require storage of a large number of PUF challenge-response pairs.

Compared to the low cost of the SRAM used for DRV fingerprinting, a relatively significant practical cost may be associated with the generation of the test voltages for characterizing the DRVs. Emerging devices such as computational RFIDs [87] can use software routines to extract DRVs, but as contactless devices they must generate all test voltages on-chip. On-chip dynamic control of SRAM supply voltage is assumed in the low-power literature at least since work on drowsy caches [25]. Supply voltage tuning has also been applied with canary cells to detect potential SRAM failures,

and as a post-silicon technique to compensate for process variation and increase manufacturing yields [75] [121].

Reliability is a bottleneck that limits practical applicability of PUFs. To improve the reliability of PUFs, techniques at system-level [18] [21], model-level [126] [63] [68] and circuit-level [82] have been proposed. Delay based PUF is a big family, including Arbiter PUF [26] [56] [29], RO PUF [102] and Lightweight PUF [70]. Among those PUFs, RO PUF is a special one, that employs relatively fewer delay components. A rich body of work has been published to improve the reliability of RO PUFs, such as selecting RO pairs to maximize the frequency gap [102]. In [84], an aging-resistant RO PUF is proposed to make RO PUFs reliable against aging. While for other PUFs of higher complexity like Arbiter PUFs, few work has been explored except some generic methods [126] [64], such as using error correction codes (ECC) to correct the flipped responses. However, there is more work needed to complete this picture:

- Few existing works take the device aging problem into consideration. Techniques that work well for overcoming transient unreliability may be ineffective for dealing with aging related flips;
- No specialized attention has been paid to improve the reliability of the popular Arbiter PUFs using properties that are specific to the Arbiter PUF circuit structure, instead of using general purpose error correction.

In this we explore the difference between reliable and unreliable PUF CRPs, and the impact of environmental noise and device aging on them. We propose to use the modeling technique that is capable of predicting both types of PUF unreliability. We use the Arbiter PUF as an example to explore the reliability of delay based PUFs. As a well-known member of delay based PUF family, Arbiter PUF has been employed as the basic block to build many complicated PUF architectures, like the Feed-forward Arbiter PUF [55], Lightweight PUF [70], and XOR PUF [102]. Therefore, techniques

that apply to the Arbiter PUF can be applied directly to other members of the delay-based PUF family.

As described in previous sections, the noise sensitivity challenges PUFs reliability. This impact can not be absolutely avoided since PUFs are based on microscopic physical variability, which is easily disturbed by noise. Inspired by the truth that physical variability was a barrier for conventional CMOS design but a chance for hardware security study, constructively utilizing the noise sensitivity feature is a promising direction, especially considering that PUFs own security characteristics.

Machine Learning modeling attacks have been deemed as the main threat to strong PUFs. One possible solution to enhance the security of strong PUFs is leveraging a more complex internal response-generating mechanism, which was supposed to make ML attacks harder. For this reason, alternative silicon architectures have been proposed in recent years. One such alternative is the *"Bistable Ring PUF" (BR PUF)* and its derivative *"Twisted Bistable Ring PUF" (TBR PUF)* [14][15]. While it is still a open problem that whether the proposed BR PUF and TBR PUF is as secure as expected.

1.4 Thesis Contribution

The contribution of this thesis includes:

- Proposing a ranking technique to build reliable PUFs using data retention voltage of SRAM cells, firstly applying Artificial Neural Network (ANN) technique for simulation-free prediction of DRV as a function of temperature, process variation assignments, and transistor sizes [119];

- Developing an analytical models for the BR PUF and the TBR PUF and use these new models in order to apply, for the first time, support vector machines (SVMs) to the BR PUF and the TBR PUF [120];

- Constructively utilizing ML modeling method to improve the reliability of delay-based PUFs, by predicting the noise-sensitive CRPs [116];

- Proposing a on-chip majority voting technique for reliable weak PUF design, in which a sequential majority voting is realized using a self-timed circuit without orchestration by a global clock. [118];

- Propose a failure-based PUF that uses failures induced by controlling the duration of power gating. We demonstrate the approach in simulation using D flip-flop circuits, and show that it reliably produces high quality output bits [117].

1.5 Thesis Outline

Following the introduction in this chapter, we divide the contribution of this thesis into three parts:

- Chapter 2 presents a constructive usage of physical variability: reliable PUF design on SRAM circuits. We demonstrates that SRAM DRV can serve as a basis for reliable identification and key generation. We also propose the first work that applies machine learning for simulation-free prediction of DRV as a function of temperature, process variation assignments, and transistor sizes.

- Chapter 3 demonstrates a high quality PUF can be created by exploiting differences in the failure propensity of instances of identical storage cells, and this approach does not require power-up states to be unbiased.

- Chapter 4 re-examines security of the BR PUF and TBR PUF closely based on FPGA implementations. We propose an analytical model for BR PUF and TBR PUF, and for the first time, support vector machines (SVMs) is employed to model these two PUF architectures.

- Chapter 5 analyzes the unreliability causes of a PUF from two aspects: transient noise and aging; the reliability impact of both aspects are explored respectively. We employ ML modeling method for PUF characterization and utilize the model to filter out the unreliable CRPs for each PUF, to achieve higher reliability.

- Chapter 6 explores hardware reliability techniques for PUF design. A novel clockless sequential PUF structure that performs autonomous majority voting is proposed, in which the external interface is included. Moreover, an internal structure that autonomously performs majority voting to improve reliability is realized.

- Chapter 7 concludes this dissertation.

CHAPTER 2

RELIABLE PHYSICAL UNCLONABLE FUNCTIONS USING DATA RETENTION VOLTAGE OF SRAM CELLS

The PUF studied in this chapter utilizes the variation sensitivity of SRAM Data Retention Voltage (DRV), the minimum voltage at which each cell can retain state. Prior work shows that DRV can uniquely identify circuit instances with 28% greater success than SRAM power-up states that are used in PUFs [39]. However, DRV is highly sensitive to temperature, and until now this makes it unreliable and unsuitable for use in a PUF. In this chapter, we enable DRV PUFs by proposing a DRV-based hash function that is insensitive to temperature. The new hash function, denoted DRV-based Hashing (DH), is reliable across temperatures because it utilizes the temperature-insensitive ordering of DRVs across cells, instead of using the DRVs in absolute terms. To evaluate the security and performance of the DRV PUF, we use DRV measurements from commercially-available SRAM chips, and use data from a novel DRV prediction algorithm. The prediction algorithm uses machine learning (ML) for fast and accurate simulation-free estimation of any cell's DRV, and the prediction error in comparison to circuit simulation has a standard deviation of 0.35 mV. We demonstrate the DRV PUF using two applications – secret key generation and identification. In secret key generation, we introduce a new circuit-level reliability knob as an alternative to error correcting codes. In the identification application, our approach is compared to prior work and shown to result in a smaller false-positive identification rate for any desired true-positive identification rate.

2.1 Introduction

This chapter presents our work that employs Data Retention Voltage (DRV), the minimum supply voltage at which state is retained, as the basis for a new SRAM PUF. Previous work [39] has shown DRV fingerprints to be more informative than power-up SRAM PUFs [31, 37]. The physical characteristics responsible for DRV are imparted randomly to each cell during manufacturing, providing DRV with a natural resistance to cloning. DRVs are not only random across chips, but also have relatively little spatial correlation within a single chip and can be treated in analysis as independent [52]. The proposed technique has the potential for wide application, as SRAM cells are among the most common building blocks of nearly all digital systems.

In this chapter we extend the idea of DRV fingerprinting to create a PUF based on DRV. To overcome the temperature-sensitivity of DRV, we propose a DRV-based hashing scheme that is robust against temperature changes. The robustness of this hashing comes from its use of (reliable) DRV-ordering instead of (less reliable) DRV values. The use of DRV-ordering can be viewed as a differential mechanism at the logical level instead of the circuit level as in most PUFs. To help validate the DRV PUF, we propose a machine learning (ML) technique for simulation-free prediction of DRVs as a function of process variations and temperature. The machine learning model enables the rapid creation of the large DRV data sets required for evaluating the DRV PUF approach. Our approach is furthermore supported using hardware measurement of DRV data.

The contributions of this chapter are as follows:

- We demonstrate that SRAM DRV can serve as a basis for reliable identification and key generation. This finding is supported by DRV characterizations of 20k SRAM cells measured three times at each of three different temperatures.
- We present the first work that applies machine learning for simulation-free prediction of DRV as a function of temperature, process variation assignments, and transistor

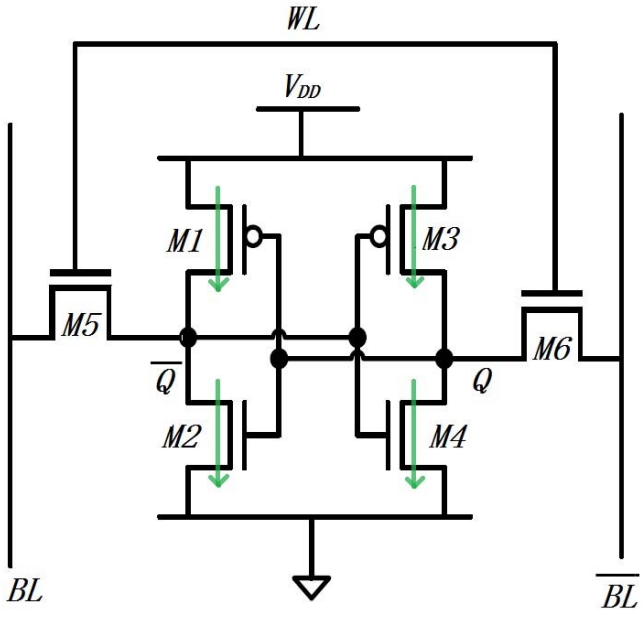


Figure 2.1: A six transistor SRAM cell. Q and $\bar{Q}$ are the complementary state nodes that store a single bit value between cross-coupled inverters implemented by transistors $M1$, $M2$, $M3$ and $M4$. WL is the wordline, and controls access transistors $M5$ and $M6$. BL and $\overline{BL}$ are the complementary bitlines used to read and write the SRAM cell. Arrows denote the direction of current leakage.

sizes. Once the machine learning model is trained, it can predict the DRV of a cell at a given temperature 2.2e6 times faster than circuit simulation, and its prediction error versus circuit simulation has a standard deviation of only 0.35 mV.

2.2 Data Retention Voltage of SRAM

An SRAM cell is commonly implemented in CMOS technology as a six-transistor circuit (Fig. 3.2). When an SRAM cell is in the *standby* condition, its word line (WL) is set low, and the two access transistors ($M5$ and $M6$) are shut off. If the supply voltage is sufficient, two inverters (composed of $M1, M2$ and $M3, M4$) use positive feedback to pull one complementary state node (Q or $\bar{Q}$) high, and the other low. If supply voltage is below DRV, then transistors operate in the sub-threshold (sub-V_{th}) region [81] where they are highly sensitive to variations and may lose state. Such a

loss of state on account of insufficient supply voltage is termed a data retention failure. The voltage at which data retention failures occur in each SRAM cell depends on its asymmetric process variation. Because DRV is randomly assigned to each cell through process variation, the DRV fingerprint of SRAM is a physical fingerprint suitable for use in a PUF.

Since the DRV of SRAM signifies the minimum supply voltage at which cells can store arbitrary state, DRV is usually studied as a lower limit to supply voltage scaling. Most previous literature focuses on cases where the SRAM supply voltage remains safely above DRV. While remaining above DRV, the supply voltage can be adjusted to reduce leakage power [25, 12], compensate for manufacturing variability [75], or compensate for environmental variations [113]. Our work is not concerned with remaining above DRV, but instead with characterizing the DRV of each cell and using this unique variation-sensitive information as part of a PUF.

Fast and accurate DRV analysis is needed to evaluate DRV fingerprinting, and significant research effort has been spent on solving this problem. The default technique for DRV analysis is Monte Carlo circuit simulation. When searching for the DRV of an entire array instead of individual cells, improvements over basic Monte Carlo simulation include the use of importance sampling [22], adaptive sampling [23, 44], and boundary line searching [30]. An overview of several statistical techniques is given by Wang et al. [114]. In Sec. 2.4 of this chapter, we propose a new technique that uses machine learning to predict DRV. This approach differs from the aforementioned statistical approaches in having the goal of predicting the DRV of individual cells, instead of just accurately estimating the failure rate of the entire SRAM using process variation statistics.

Procedure 1 Characterize the DRV fingerprint of a set of SRAM cells.

Input: A set of bit-addressable SRAM cells
Output: v_i^0, v_i^1 {the DRV characterization of cell at address i.}

Let V_{nom} be the nominal supply voltage for the SRAM
Let s_i refer to the logical state of SRAM address i.
for $w = 0, 1$ **do**
  for $i \in SRAM$ **do**
    $s_i \leftarrow w$ {write w to SRAM address}
    $v_i^w \leftarrow v_{min}$ {value used if no retention failure observed}
  end for
  $v_{test} \leftarrow v_{max}$ {initialize test voltage}
  while $v_{test} > v_{min}$ **do**
    lower SRAM voltage from V_{nom} to v_{test}
    remain at voltage v_{test} for time t_{test}
    raise SRAM voltage from v_{test} to V_{nom}
    for $i \in SRAM$ **do**
      if $(s_i \neq w) \wedge (v_i^w = v_{min})$ **then**
        SRAM cell at address i did not retain state w after applying v_{test}*, and* v_{test} *is the first and highest voltage at which this retention failure occurred.*
        $v_i^w \leftarrow v_{test}$
      end if
    end for
    $v_{test} \leftarrow v_{test} - \Delta v$ {try a lower voltage next}
  end while
end for

2.3 DRV Characterization

We characterize the DRV of an SRAM cell at address i with a pair $\langle v_i^0, v_i^1 \rangle$. Each v_i^w represents the highest voltage at which address i will have a retention failure after state w is written to it. In principle, v_i^0 and v_i^1 are real-valued; in practice, we approximate each one using N discrete voltages with a step size of Δv. Proc. 1 presents our characterization procedure. The implementation details of Proc. 1 vary slightly when applied in hardware measurement or simulation as explained in the next two subsections.

The DRV characterization procedure is parameterized by maximum, minimum, and step size for test voltages (v_{max}, v_{min}, and Δv respectively), and by the time (t_{test}) for which each test voltage is applied. Simulations by Nourivand et al. [75] using a procedure similar to Proc. 1 show that a value of 2 ms for t_{test} is sufficient to induce

retention failures. The total time to characterize the DRV of an SRAM cell using Proc. 1 is given by t_{proc} in Eq. 2.1. In the case of simulation, t_{proc} is the simulated time, and the actual runtime for the circuit simulator is many orders of magnitude larger. The frequency of observing different DRVs in hardware measurements and simulation are shown in Fig 4.5.

$$t_{proc} = t_{test} \times \frac{v_{max} - v_{min}}{\Delta v} \tag{2.1}$$

2.3.1 Hardware DRV Measurement

The target platform for DRV fingerprinting is an integrated SRAM block with an adjustable supply voltage, as is sometimes used to compensate for variation [54]. To simplify experiments, our platform mimics this configuration using a dedicated SRAM chip and a separate microcontroller. Figure 2.2 presents the overview of our experimental system. SRAM supply voltages are generated using analog outputs of a Texas Instruments MSP430 F2618 microcontroller [108], and that same microcontroller also orchestrates the timing of the supply voltage changes (per Proc. 1). An op-amp configured as a voltage follower tracks the analog output voltage from the microcontroller and powers the SRAM at the same voltage; the op-amp is used because the analog output of the microcontroller cannot supply enough current to power the SRAM directly. All experiments use instances of SRAM chip AS6C6264 [3] and the DRV characterization parameters are v_{max} = 700 mV, v_{min} = 0 mV, Δv = 2 mV, and t_{test} = 1 s. Thermal tests are conducted inside of a Sun Electronics EC12 Environmental Chamber [105], and an OSXL450 infrared non-contact thermometer [76] with $\pm 2°C$ accuracy is used to verify the temperature.

Note that our experimental platform differs from that used in our previous work [39]. In our previous work, the DRVs of SRAM cells in a microcontroller memory are characterized by repeatedly lowering the microcontroller's supply voltage and

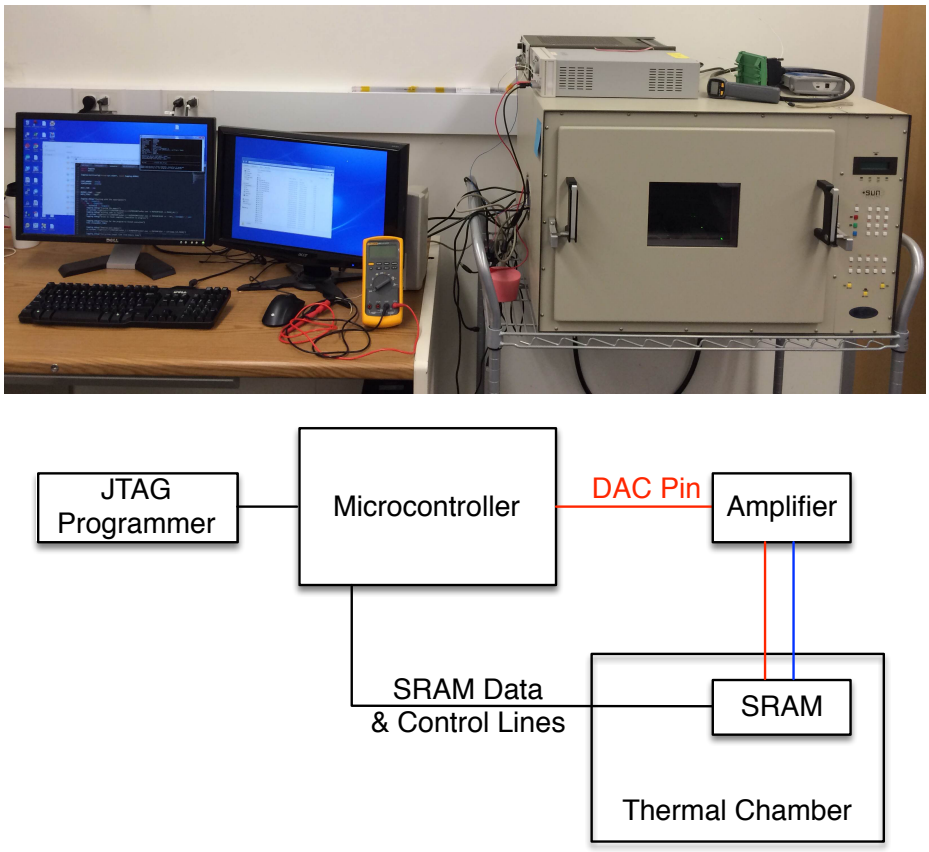


Figure 2.2: Experimental platform used for determining SRAM chip DRV assignments. Photographed in EECS department, at the University of Michigan.

observing the highest voltage that induces a retention failure in each cell. Because the microcontroller's SRAM shares a common supply node with the processing core, the low test voltages used for the characterization cause the core to reset and lose its state. As persistent state is required for the DRV characterization, our experiments used the microcontroller's non-volatile memory to preserve state while the test voltages were applied.

2.3.2 DRV Measurement in SPICE Simulation

Circuit simulation is a second platform for DRV characterization (Proc. 1), and it complements hardware measurements by allowing for DRV exploration under controllable process variations and environmental conditions. In circuit simulation,

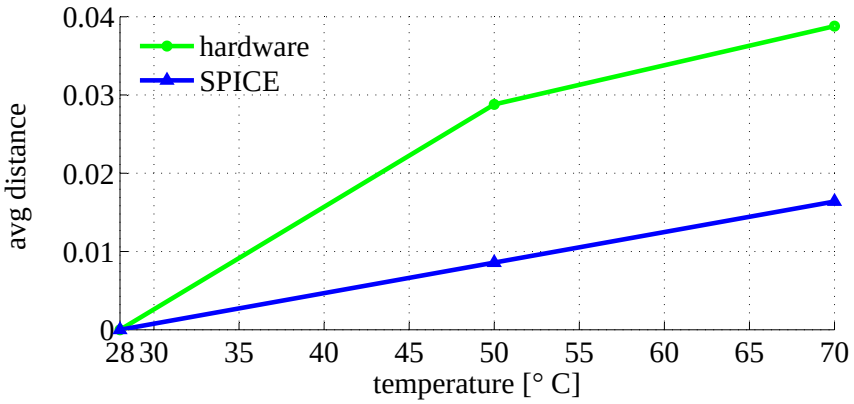
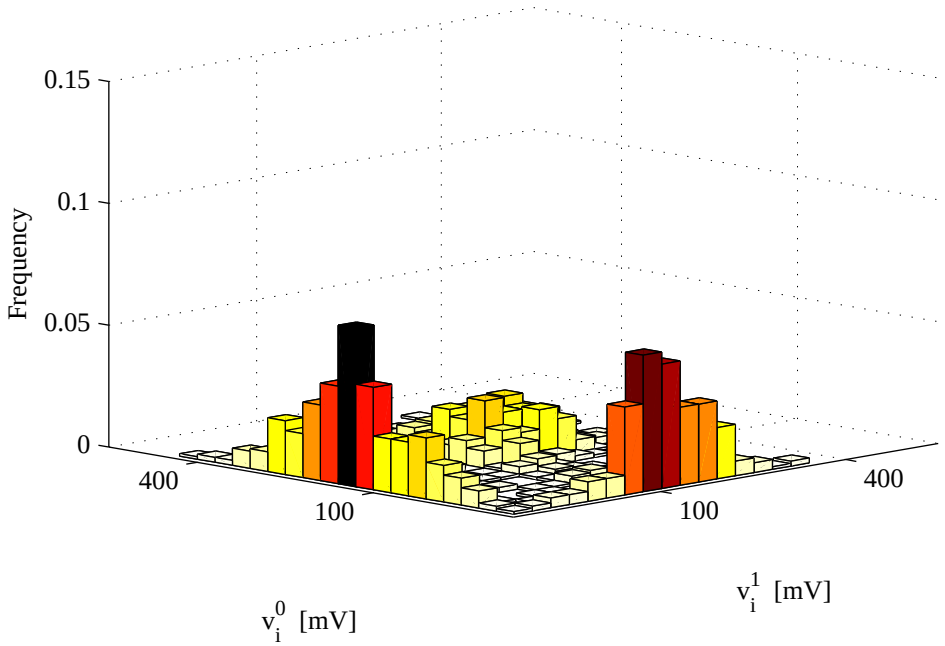


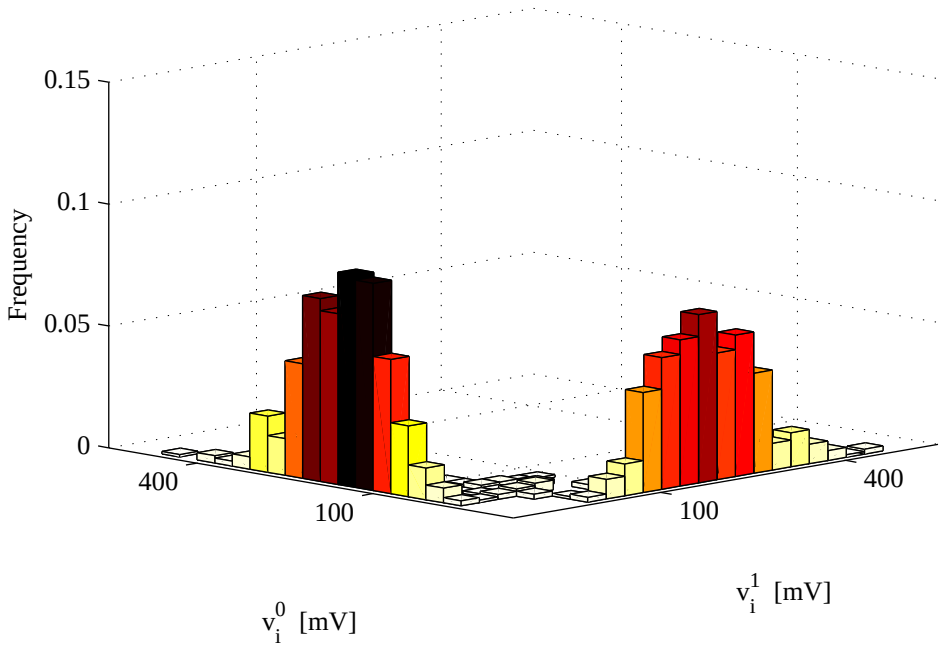
Figure 2.3: Distance (per Eq. 2.3) between two characterizations of the same cell increases as temperature changes.

we use transistor models from the 45 nm predictive technology model [127, 13]. To mimic the random process variations that give each cell its unique DRV, variations are introduced for transistor width W, length L and threshold voltage V_{th}. The International Technology Roadmap for Semiconductors (ITRS) indicates that transistor length should have a 3σ variation that is 10% of the nominal length L [1, 5]. Adopting the same guideline for transistor width, in our simulation the random components of both W and L are normally distributed with a standard deviation that is 3.33% of the nominal W or L value. The standard deviation of threshold voltage is given by Eq. 2.2; a value of 1.8 mV*μm is used for the matching constant $A_{V_{th}}$[79]. The parameter values used when implementing the DRV characterization procedure (Proc. 1) in SPICE are $v_{max} = 500$ mV, $v_{min} = 0$ mV, $\triangle v = 0.1$ mV, and $t_{test} = 2$ ms.

$$\sigma_{V_{th}} = \frac{A_{V_{th}}}{\sqrt{W * L}} \tag{2.2}$$



(a) DRV Joint PDF from hardware measurements



(b) DRV Joint PDF from SPICE

Figure 2.4: The joint probability distribution function over all cells of the two variables (v_i^0 and v_i^1) comprising a DRV characterization. The distribution is determined experimentally using Proc. 1, and shows that a large fraction of cells have the minimum possible value (v_{min}) for either v^0 or v^1, indicating a cell that retains one written state across all test voltages.

2.3.3 Impact of Temperature Variations

DRVs generally increase with temperature [81], and this hinders the reliability of DRV-based fingerprinting. Recalling that each DRV is a point $\langle v_i^0, v_i^1 \rangle$ in 2-dimensional space, an intuitive way to define the distance between two DRVs is to use their distance in this 2-dimensional space (Eq. 2.3). This distance metric is used as the basis for DRV fingerprint matching in our previous work [39]. To demonstrate the impact of temperature, we compute the average distance between two characterizations of the same cell, when one is taken at 28°C and the other at 50°C, or 70°C. As shown in Fig. 2.3, the average distance between the two characterizations increases with the temperature difference.

$$d1(i,j) = \sqrt{\left(v_i^0 - v_j^0\right)^2 + \left(v_i^1 - v_j^1\right)^2} \tag{2.3}$$

Given that DRV fingerprints are intended for use in real-world scenarios without precisely-controlled temperatures, the temperature sensitivity shown in Fig. 2.3 indicates that the distance metric of Eq. 2.3 is prone to unreliability in real-world usage. In Sec. 2.5, we propose a new technique for extracting temperature-invariant information from DRV and demonstrate that this new technique is highly reliable when temperature fluctuates.

2.4 Modeling the DRV of an SRAM Cell

Although the SPICE simulation described in Sec. 2.3.2 is a straightforward and highly accurate approach to characterize the DRV of SRAM cells, it is very time consuming for two reasons. The first reason is that, to find the maximum voltage that induces a failure in each cell, numerous test voltages must be applied (Proc. 1). The second reason is simply that simulating each test voltage is itself very slow. On our experimental machine, equipped with an Intel Xeon E5-2690 processor running at

2.90 GHz with 64 GB of RAM, simulating a single test voltage on a single SRAM cell for 2 ms has a runtime of 0.17 s.

An alternative to iterative SPICE analysis is to predict DRV using a model. Just as the DRV of each SRAM cell is ultimately determined by temperature and the process variations of its transistors, the DRV of an SRAM cell can be formulated as a function of its temperature T and transistor width, length, and threshold voltage (W, L, and V_{th} respectively). Qin et al. [81] provide an analytical model for the DRV of an individual cell as in Eq. 2.4, where DRV_r is the DRV at room temperature, and DRV_f is defined in Eq. 2.5 with $\triangle T$ representing the temperature difference from room temperature. Terms a_i, b_i, and c in Eq. 2.5 are fitting coefficients and their values are determined empirically for each CMOS technology process [81].

$$DRV = DRV_r + DRV_f \tag{2.4}$$

$$DRV_f = \sum_{i=1}^{6} a_i * \frac{\triangle(W_i/L_i)}{W_i/L_i} + \sum_{i=1}^{6} b_i * \triangle(V_{thi}) + c * \triangle T \tag{2.5}$$

Although this model can accurately estimate the DRV of a cell, it has two weaknesses that create the need for a more advanced model:

1. To predict a specified DRV value with Eq. 2.5, the user needs to know the DRV_r for each SRAM cell. This value is not expressed as a function of transistor parameters and can only be calculated through hardware measurement or computationally expensive circuit simulation.

2. Using the same coefficients a_i, b_i and c for different SRAM cells creates estimation errors. In reality, the DRV of different cells increase according to different coefficients depending on their unique process variations. This distinction is especially important in our work, where the unique impact of process variations across cells is critical to the overall work.

2.4.1 Predicting DRV using Artificial Neural Networks

Our approach addresses the two aforementioned weaknesses in Qin's work by using machine learning for DRV prediction. The machine learning algorithm predicts the DRV directly from the parameters that are responsible for determining it, without using expensive circuit simulation. Given that the values of process variation parameters vary over bounded ranges, it follows that the DRVs too fall within a bounded range $[DRV_{min}, DRV_{max}]$. The range of DRVs is manually divided into K classes, each with size $\triangle DRV$ (Eq. 2.6). The use of K classes makes DRV prediction a "multi-classification" problem: for any given feature pattern $\{W_i, L_i, ...V_{th\_i}, T\}$, there is exactly one among K classes corresponding to the correct DRV output.

$$\begin{aligned}[DRV_{min}, DRV_{max}] = \{&[DRV_{min}, DRV_{min} + \triangle DRV) \cup \\ &[DRV_{min} + \triangle DRV, DRV_{min} + 2 * \triangle DRV) \cup ... \\ &... \cup [DRV_{max} - \triangle DRV, DRV_{max}]\}\end{aligned} \tag{2.6}$$

Artificial Neural Network(ANN) is a well-known ML method [40] that is widely used to solve multi-classification problems. Our DRV prediction method specifies the DRV of an SRAM cell as an ANN output class, and indicates the class to which the corresponding SRAM parameter pattern should be assigned to. Our approach collects a group of samples from SPICE, including physical parameters of SRAM circuitry as input and corresponding DRV values (which can be viewed as golden value) as output. An ANN model is later trained based on this data to match input with output. In this process, neurons learn to classify the examples from each class (Fig 2.5). Finally, the hidden neurons dealing with the same class will be combined as one group, so the number of groups corresponds to the number of output classes. Each class has a corresponding surface, which is approximated by the combined neuron groups.

To get data for ANN model training, we use SPICE simulator as the infrastructure for collecting DRV statistics. DRV values are extracted from simulations of 2000 cells across temperature 25℃ to 100℃, with a step size of 1℃. A common problem

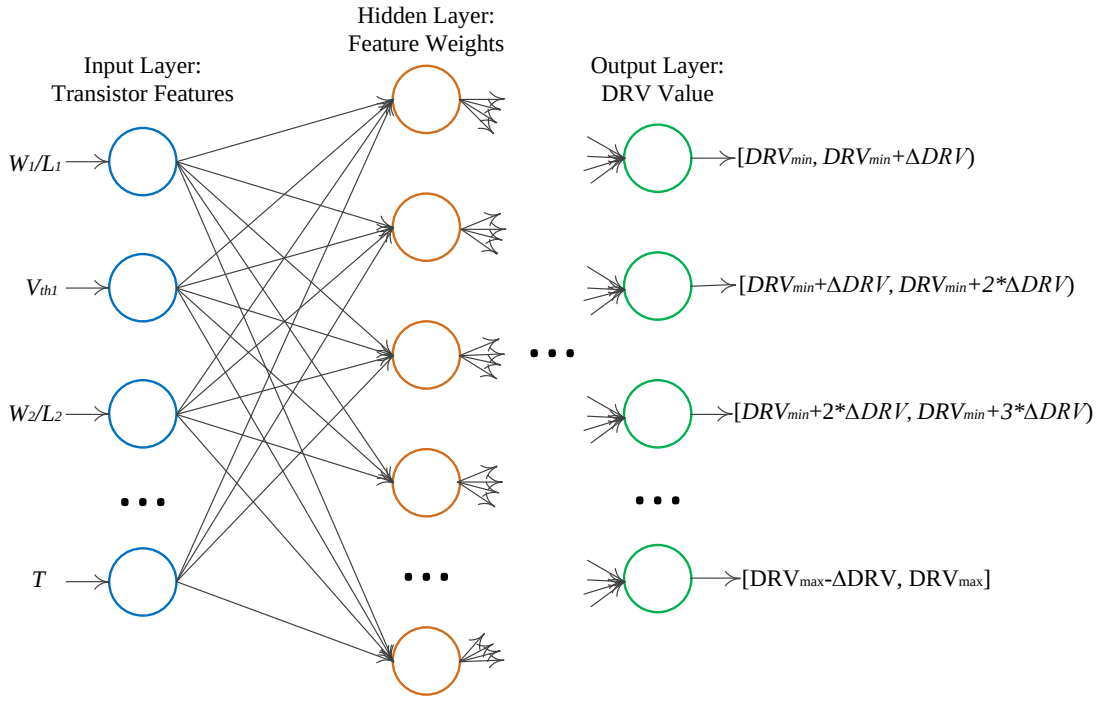


Figure 2.5: Artificial neural network for DRV classification and prediction.

with ML models is overfitting, where a model is trained to perform well on training data but fails to yield similar results upon seeing new data. To avoid overfitting in building the DRV model, we reordered the samples and divided them into three subsets: training (60%) validation (20%) and test (20%). Training set is the data set used for computing the gradient and updating the network weights and biases. The validation set is used to monitor errors during the training process. The validation set and training set error usually decreases during the initial phase of training. The test set error is not used during training, but is used to validate the model performance and compare different models.

2.4.2 Evaluating Accuracy of DRV Prediction

The prediction results of the test subsets are shown as Fig. 2.6. The regression plots display the ANN-predicted DRVs with respect to the golden DRV values collected

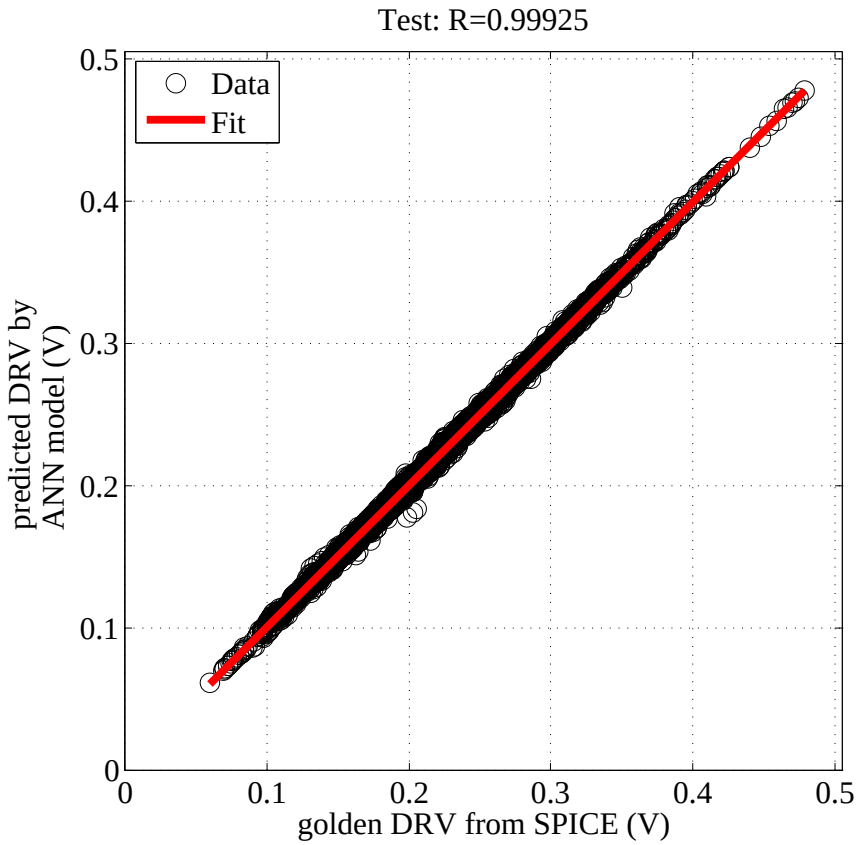


Figure 2.6: Training results based on Neural Network model, across three data sets. R denotes the correlation between golden DRV data from SPICE simulation, and predicted DRV value from our model.

from SPICE simulation. In Fig. 2.6, R denotes the correlation between model outputs and golden values. For a perfect fit, the predicted outputs should be equal to the golden values (the data should fall along a 45 degree line). For our DRV model, there is a high correlation between prediction and output for all data sets.

Before our ANN model in this work, the linear model described in Eq. 2.5 was widely used to model the DRV value of SRAM designs, which can be optimized with Linear Regression (LR) method. LR fits a data model that is linear in the model coefficients. The most common type of LR is a "*least-squares fit*", which can find an optimal line to represent the discrete data points. In a LR model, the

same physical parameters of ANN models are defined as input training features $p = \{p_i, |p_i \in \{W_1/L_1, W_2/L_2, \ldots, T\}\}$. By denoting the linear coefficients with $\theta = \{\theta_0, \theta_1, \ldots, \theta_n\}$, we get:

$$h_\theta(p) = \theta_0 + \theta_1 * p_1 + \cdots + \theta_n * p_n \tag{2.7}$$

where θ stands for the set of coefficients (e.g. a_i and b_i as shown in Eq. 2.5). Each training sample is composed of transistor feature set p and the corresponding golden DRV value DRV_{golden} from SPICE simulation. Based on "*least-squares fit*" rule, the cost function of m training examples can be expressed as:

$$J(\theta) = \frac{1}{2m} \sum_{k=1}^{m} (h_\theta(p^{(k)}) - DRV_{golden}^{(k)})^2 \tag{2.8}$$

where $p^{(k)}$ corresponds to the training features of k^{th} training sample, like the transistor sizes and temperature. To obtain the optimal θ, we applied "Gradient Descent" simultaneously on each coefficient $\theta_j,\ j \in (1, 2 \ldots n)$:

$$\begin{aligned} &Repeat\{ \\ &\quad \theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \\ &\qquad = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(p^{(k)}) - DRV_{golden}^{(k)}) p_i^{(k)} \\ &\quad \} \end{aligned} \tag{2.9}$$

α is the learning rate of linear regression model, $p_i^{(k)}$ is the i^{th} feature of the k^{th} training sample.

To further evaluate the effectiveness of our ANN model, we compare its prediction error to that of the LR model[1] on a randomly chosen data set of size 3500. Fig. 2.7

[1]The linear regression model is trained and optimized with the same three data sets as the neural network model.

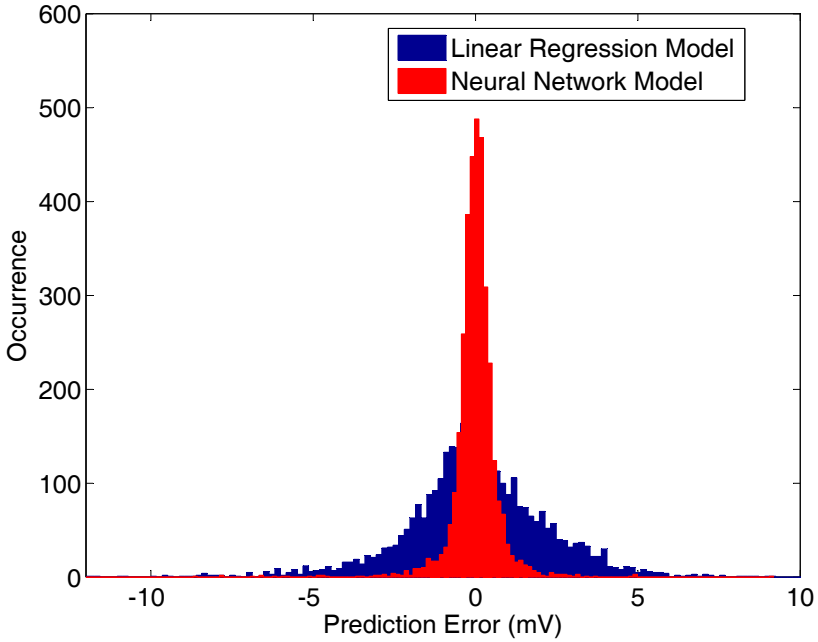


Figure 2.7: *DRV* prediction error for the artificial neural network model and linear regression model. In both cases, error is determined by comparison to SPICE simulated results.

presents the prediction error of these models. The neural network model achieves smaller prediction errors than the linear regression model. The mean μ and standard deviation σ of prediction error for the neural network model are -0.01 mV and 0.35 mV respectively, while those of the linear regression model are 0.041 mV and 0.9 mV. The neural network model outperforms the linear model because the neural network model assigns varied weights and bias to different feature patterns [2], whereas the linear model formulates all input features with the same optimized θ.

[2]This also validates our finding in Fig. 2.8, that different SRAM cells have different DRV growth slopes while temperature is increasing.

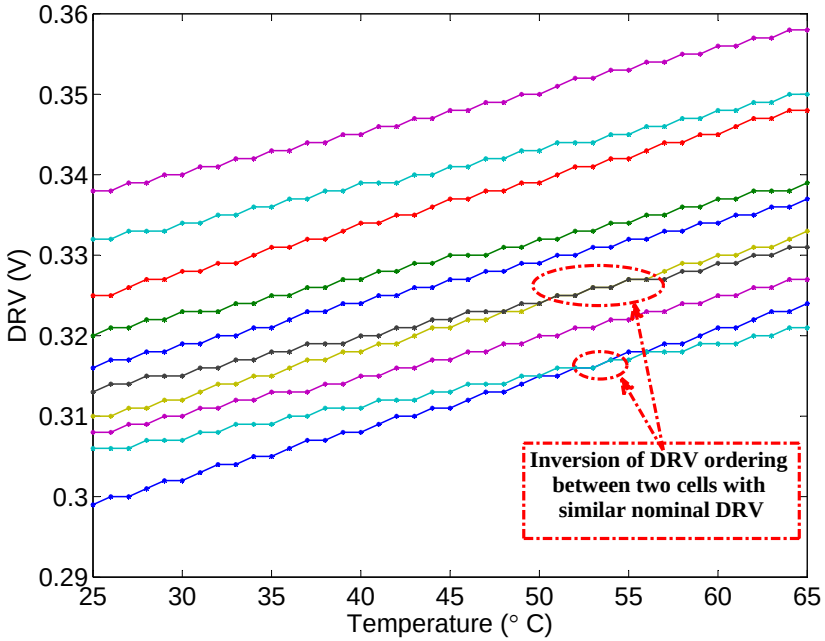


Figure 2.8: DRV of SRAM cells increase linearly with temperature, but the slope varies across cells. The ordering of DRV is largely preserved across temperatures, except for a few cells that switch ordering.

2.5 DRV-based PUF

DRV-based identification or authentication schemes must consider the impact of temperature changes. The DRV of each SRAM cell increases approximately linearly with temperature [81], and the coefficient relating DRV to temperature varies only slightly across cells. Accordingly, the relative ordering of DRVs across cells is more reliable than the values themselves; in other words, the cell with the i^{th} highest DRV will remain roughly the i^{th} highest when temperature changes, even though all DRVs will change in absolute terms. Fig. 2.8 shows the relationship of DRV and temperature, according to machine learning prediction, for 10 randomly chosen SRAM cells. The DRV ordering is preserved across temperature values, except for two pairs of cells that flip their ordering. Any two cells with sufficiently different nominal DRVs have a DRV-ordering that does not change with temperature.

2.5.1 DRV-based Hashing with DH and DH-PREIMAGE

To utilize the robustness of DRV ordering, we propose a hashing scheme with the mapping between challenges and responses defined by the DRV-ordering within SRAM address pairs. In this scheme, a challenge C of length m is a sequence of address pairs $(\langle \bar{c}_0, c_0 \rangle, \ldots, \langle \bar{c}_{m-1}, c_{m-1} \rangle)$, a response R is a bit string $(r_0, \ldots, r_{m-1})$. A DRV measurement D assigns values to each address of an SRAM such that $D(c) = \max(v_c^0, v_c^1)$, where v_c^0 and v_c^1 are the minimum retention voltages after writing the 0 and 1 states to the cell at address c (Proc. 1). Note that a DRV D is a single imprecise observation, and two DRVs produced by the same chip will only match approximately. Procedure DH(D, C) (Proc. 2) hashes a challenge C to a response R. Procedure DH-PREIMAGE(D, R) (Proc. 3) computes a challenge C that reliably hashes to response R on a particular chip. For any DRV assignment D and response R, the relationship $R = \text{DH}(D, \text{DH-PREIMAGE}(D, R))$ holds. Procedures DH and DH-PREIMAGE are the building blocks for key generation and identification applications in Sec. 2.5.2 and 2.5.4.

The DRV-based hashing procedure DH is designed to be resilient to small fluctuations in DRV, and to common-mode DRV shifts such as those caused by temperature. The steps for DH are given in Proc. 2. For each address pair $\langle \bar{c}_i, c_i \rangle$ in the challenge, the corresponding response bit r_i is assigned a 1 if address c_i has the higher or equal DRV, and 0 if address $\bar{c}_i$ has a higher DRV. Procedure DH is made resilient by applying challenges for which the addresses in each pair have vastly different DRVs, so that the inequality at line 2 of Proc. 2 consistently resolves in the same way despite small variations.

Given a DRV D and a desired response R, the role of procedure DH-PREIMAGE is to create a challenge C that will reliably generate R whenever it is applied to the same SRAM that produced D. The steps for DH-PREIMAGE are shown in Proc. 3. For each bit r_i of the desired response, a pair of challenge addresses $\langle \bar{c}_i, c_i \rangle$ is chosen. If

the desired response bit is 0 (1), then $\bar{c}_i$ (c_i) is assigned the address of the cell with the higher DRV. Note that the two addresses chosen for each pair have markedly dissimilar DRVs that are separated by $|D| - m$ positions in DRV ordering (see lines 3 and 5 of Proc. 3), where $|D|$ is the SRAM size and m is the response length. Stated differently, one address in each pair has one of the m highest DRVs in the SRAM, and the other has one of the m lowest DRVs. The DRV dissimilarity within each address pair ensures that the higher DRV can be reliably determined when the challenge is applied in a subsequent call to DH.

A demonstration of the DRV-based hashing is given in Fig. 2.9. According to the depicted DRV assignment D, procedure DH hashes challenge $C = (\langle 1, 10\rangle, \langle 6, 9\rangle, \langle 7, 5\rangle)$ to response $R = (1, 0, 1)$: the first response bit is 1 because address 10 has a higher DRV than address 1, the second response bit is 0 because address 6 has a higher DRV than address 9, and the third response bit is 1 because address 5 has a higher DRV than address 7.

A necessary condition for obtaining a wrong response bit for a challenge address pair is that, when some test voltage is applied, the cell with nominally lower DRV fails, and the cell with the nominally higher DRV does not. In the toy example of Fig. 2.9, given that the DRVs within each pair have a gap of 110 mV, this will only happen in the case of extreme noise or if the supply voltage differs by 110 mV from one cell location to the other. As the cells of an SRAM are powered by the same supply, and given that supply nodes are already designed to avoid local voltage droop, such a large supply voltage difference across cells is uncommon.

The DRV hashing scheme in this chapter is related to index-based syndrome (IBS) coding [126] and ordering-based encoding schemes applied to PUFs with real-valued outputs [69]. Given a group of indexed objects with real-valued measurements, IBS coding encodes each bit to a syndrome that is the index of the maximum or minimum value in the group depending on the bit's polarity. Given noisy measurements of

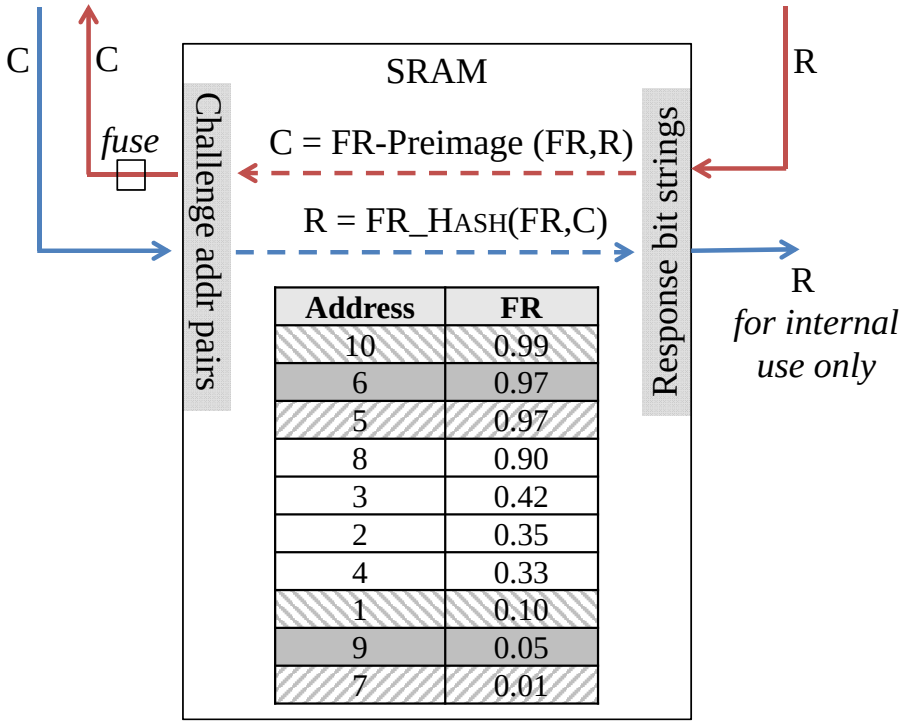


Figure 2.9: Example of DRV-hashing. According to the depicted DRV assignment D, and letting challenge C be $(\langle 1, 10\rangle, \langle 6, 9\rangle, \langle 7, 5\rangle)$, procedure $\mathrm{DH}(D, C)$ produces response $R = (1, 0, 1)$. Similarly, procedure DH-Preimage(D, R), given this response R, would produce as output the same challenge C.

the same indexed objects, the syndrome is decoded by determining whether the measurement it indexes is closer to maximal or minimal in the group. A comparison of the reliability and security of IBS versus other approaches is given by Yu et al. [125]. Variants of IBS coding are also applied to SRAM power-up state PUFs [35]. Procedures DH-Preimage and DH are analogs for IBS encoding and decoding respectively. In addition to using a hashing scheme related to IBS coding, a second reliability enhancing feature is that the SRAM cell pairs are selected to maximize the discrepancy between the values in each pairing. The idea of configuring real-valued PUFs to utilize large discrepancies for enhanced reliability has been proposed previously for ring oscillator PUFs [102, 124], where the identifying feature is oscillator frequency instead of minimum retention voltage.

Procedure 2 $R = \text{DH}(D, C)$: Use DRV assignment D to hash challenge C to response R.

Input: D {DRV assignments for a set of SRAM addresses}
Input: C {sequence of addr pairs $(\langle \bar{c}_0, c_0 \rangle, \ldots, \langle \bar{c}_{m-1}, c_{m-1} \rangle)$}
Output: R {bit string response $(r_0, \ldots, r_{m-1})$ to challenge}

```
for ⟨c̄_i, c_i⟩ ∈ C do
    r_i ← (D(c_i) ≥ D(c̄_i))
end for
return R
```

Procedure 3 $C = \text{DH-PREIMAGE}(D, R)$: Map response R to challenge C using DRV assignment D.

Input: D {SRAM DRVs. $D(a)$ is DRV of cell at address a.}
Input: R {the desired response bit string $(r_0, \ldots, r_{m-1})$}
Output: C {sequence of addr pairs $(\langle \bar{c}_0, c_0 \rangle, \ldots, \langle \bar{c}_{m-1}, c_{m-1} \rangle)$} {sort addresses by DRV. Let a_i denote address such that $D(a_i)$ is i^{th} highest among all addresses}

```
for i ∈ 0..(|R| − 1) do
    if r_i = 1 then
        ⟨c̄_i, c_i⟩ ← ⟨a_{i+|D|−m}, a_i⟩ {c_i gets addr with higher DRV}
    else
        ⟨c̄_i, c_i⟩ ← ⟨a_i, a_{i+|D|−m}⟩ {c̄_i gets addr with higher DRV}
    end if
end for
return C
```

2.5.2 Secret Key Generation

Cryptographic keys must be fully reliable, and this is in conflict with the inherent imprecision of PUFs in sensing the effects of small physical variations. Error correcting codes can bridge the gap from noisy PUFs to reliable keys. With error correction, some number of raw response bits are transformed by helper data into a noisy codeword that is decoded into a reliable key. One example of error correction in weak PUFs is the use of BCH codes with power-up fingerprints [31]. The physical nature of PUFs also allows for circuit-level reliability mechanisms that enable lighter-weight[3] error correcting codes, or in some cases supplant them entirely. Examples of reliability-enhancing circuit techniques are reinforcing variation tendencies with directed aging [74, 10], and

[3] i.e. codes that are cheaper to implement but cannot correct as many errors.

using helper data to mark on each device the response bits that are precharacterized as unreliable [126].

Our key generation using DRV-hashing uses circuit-level reliability enhancement and (optionally) error correcting codes. The steps to implement DRV-based secret key generation for a given SRAM instance are shown in Proc. 4. Lines 1-5 comprise the enrollment process to occur at the manufacturer immediately after fabrication. First, an arbitrary secret key K is chosen and encoded into codeword R; R is the value that should be the response of the DRV PUF in the field later. Next, DH-PREIMAGE is called (line 3) to generate a challenge that will reliably hash to response R on this PUF instance. The challenge is then stored to a one-time-programmable on-chip memory (line 4). Finally, the enrollment process is completed by blowing a fuse to disable the DH-PREIMAGE functionality (line 5). After the enrollment process is completed, the PUF can be used in the field as a secret key. When the stored challenge C is applied to the PUF in the field, it hashes to response R' according to DRV D' (line 6). If D' is similar to the enrollment DRV D (as it will be for the same chip), then the inherent robustness of DH should produce a response R' that exactly matches or closely approximates R. Response R', a possibly noisy version of the original codeword R, is decoded to correct errors and regenerate the enrolled key K (line 7). This key is a secret, chosen by the party that enrolled the DRV PUF, and known only to them. To maintain secrecy of key K, it must only be used as an input to a cryptographic hash, and never be revealed in plain text.

Note the secrecy of the generated key requires that an attacker cannot apply arbitrary challenges to the DRV PUF. If an attacker can apply chosen challenges, then helper data manipulation attacks from other pairing-based PUFs [17] can also expose the secret key from the DRV PUF. It is therefore necessary in the secret key application that the challenge addresses are supplied exclusively from a memory that is not overwritable in the field. Note that non-overwritable challenge addresses

need not be secret. The addresses do not leak information about the key under the assumption that DRVs are independent and identically distributed, as shown in work on index-based syndrome coding [125].

An adversary possessing a chip may somehow modify the test voltages prescribed by Proc. 1 for DRV characterization. This could induce a flawed DRV characterization; for example, if the voltage is never lowered at all, the characterization would wrongly conclude that all cells have a DRV of 0 V since none ever had a retention failure at any test voltage. However, voltage manipulation does not provide any useful information about the DRV-based secret key. Even though it may be possible to learn that a retention failure happens at some particular voltage, no information is leaked unless it can be determined which address in a pair failed, and this is not observable because the output of the DRV PUF is never revealed in the clear.

2.5.3 Reliability

The response bit-error-rate (BER) in key generation experiments depends on the key size and the size of the SRAM used for the DRV PUF. The result of Fig. 2.10 shows the BER[4] of an m-bit key generated by a $2m$-cell SRAM[5]. The response BER decreases as m increases, and does not exceed 1e-5 for any key size larger than 60 bits. The BER decreases as the SRAM size increases, because a larger SRAM tends to have a larger difference between the DRVs of the addresses within each challenge pair. We refer to the DRV difference between the two addresses in each challenge pair as the "DRV gap" of a DRV PUF. Larger DRV gaps indicate more reliable DRV PUF

[4]In BER analysis, error correcting codes are not used so that circuit-level reliability of the proposed hashing scheme can be observed. Error correcting codes would serve to correct the errors that contribute to BER.

[5]An SRAM with $2m$ cells is the smallest SRAM capable of generating an m-bit response, because each response bit is generated using two addresses.

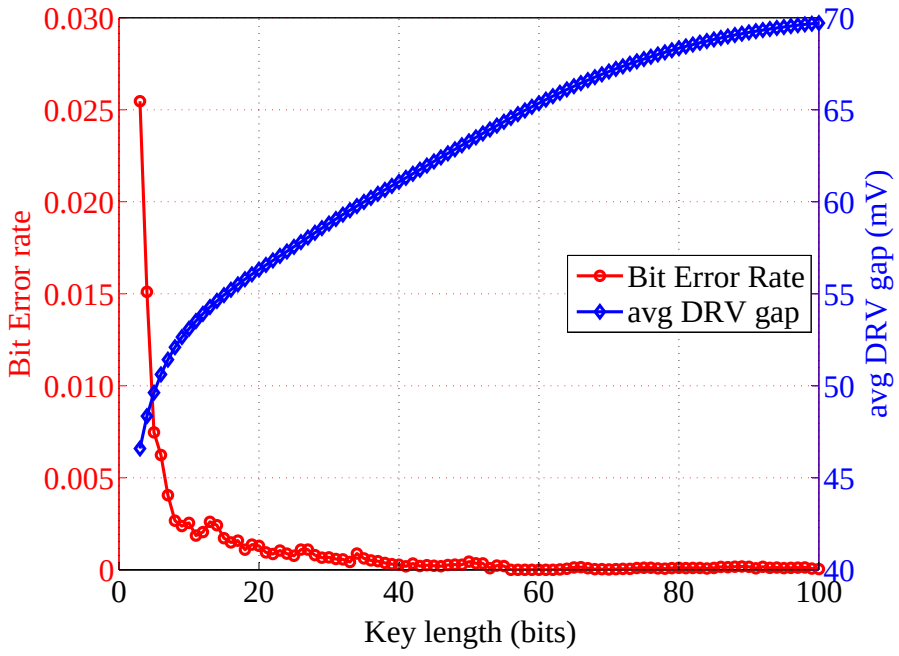
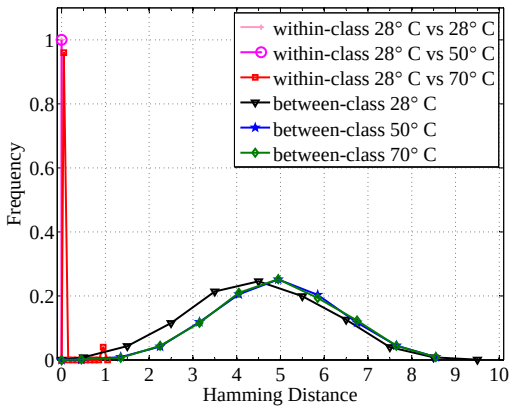


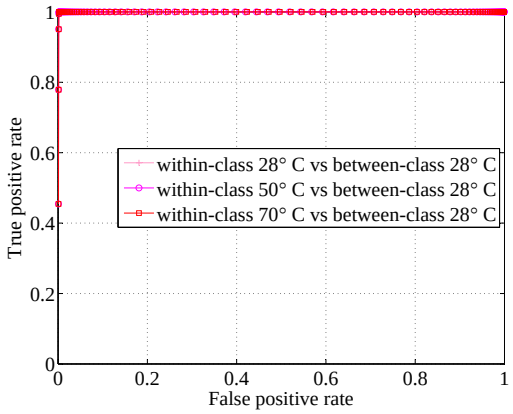
Figure 2.10: When implementing a key of length m using a DRV PUF in SRAM of size $2m$ (per Proc. 4), the response bit error rate (BER) decreases as m increases. The decrease in BER results from an increase in the DRV gap, where "avg DRV gap" represents the absolute DRV difference between $\bar{c}_i$ and c_i, averaged over challenge pairs.

responses, because the determination in DH of which challenge address has the higher DRV will be less error prone.

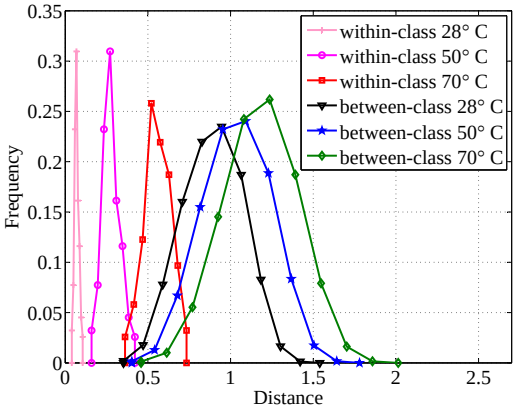
The BER can be further reduced by increasing the size of the SRAM to beyond the minimum of twice the key length m. In this case, the cells with DRVs near the median DRV of the SRAM are not among the m highest nor m lowest, and are therefore not selected by DH-Preimage to be used in the challenge. This further increases the DRV gap to reduce BER. Fig. 2.10 shows experimentally the average DRV gap as a function of SRAM size and key length. The areas of Figure 2.10 with the darkest coloring correspond to the most reliable scenarios for key generation. Therefore, arbitrary robustness can be added directly to the hashing scheme, creating a second reliability knob to be used in concert with, or instead of, error correcting codes.



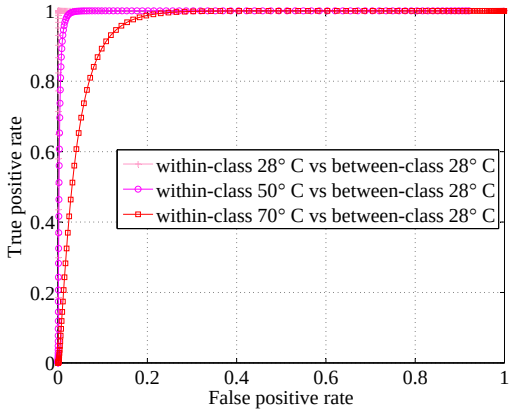
(a) Distribution of between-class and within-class distances according to the metric Response-Distanceleft (Proc. 5)



(b) ROC curves from Response-Distance data at left



(c) Distribution of between-class and within-class distances according to the metric Voltage-Distanceleft (Eq. 2.10)



(d) ROC curves from Voltage-Distance data at left

Figure 2.11: Results showing identification performance using the distance metric Response-Distance (Proc. 5) in upper plots, and performance using Voltage-Distance (Eq. 2.10) in the lower plots. The temperature resilience of DH and DH-Preimage causes Response-Distance to outperform Voltage-Distance, as indicated by lower false positive rates for equivalent true positive rates.

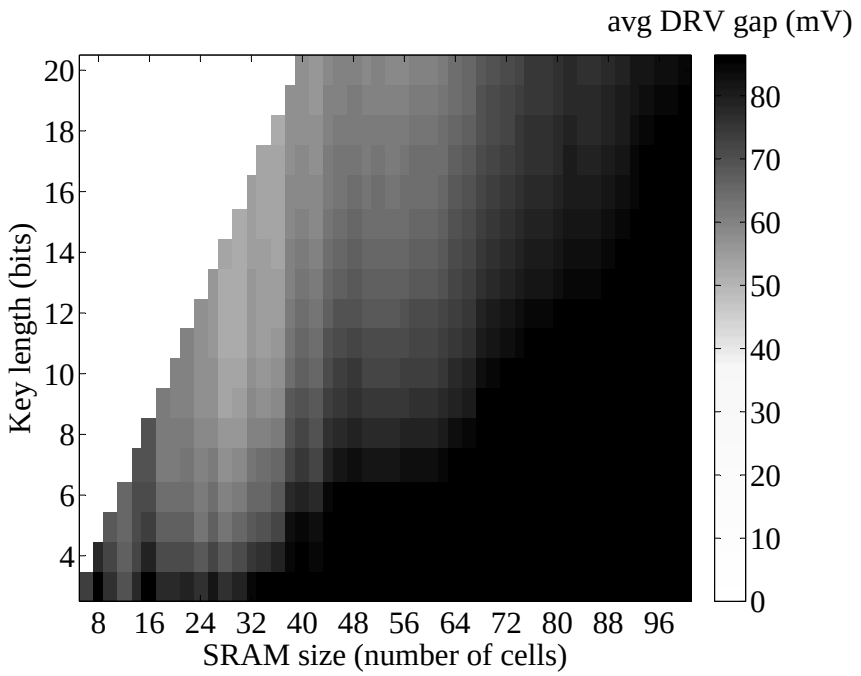


Figure 2.12: When implementing a key of length m using a DRV PUF in SRAM of size $\geq 2m$ (per Proc. 4), there is a clear increase in the average DRV gap as SRAM size increases. Because a larger DRV gap equates to a lower BER, changing the size of SRAM therefore represents a reliability knob for the DRV PUF.

2.5.4 Circuit Identification

Hashing functions DH and DH-Preimage can be used for reliable chip identification, in a way that is similar to their use in key generation. In this application, the DRV of each SRAM is not considered secret, and arbitrary challenges can be applied to the SRAM. The goal of chip identification is to determine whether two DRV characterizations $D1$ and $D2$ are generated by the same SRAM cells. Using a distance metric to quantify dissimilarity between two DRV characterizations, a determination of "same identity" is made whenever the distance between $D1$ and $D2$ is below some matching threshold. Distances between two DRVs from the same cells are denoted within-class, and distances between two DRVs from different cells are denoted between-class. A true positive identification occurs when a within-class distance is below the matching threshold, and a false positive identification occurs

Procedure 4 Use DRV PUF as reliable secret key. Lines 1-4 enroll the PUF and personalize it with key K. Lines 5-6 occur in the field to regenerate K from challenge C.

1: Choose a secret key K
2: $R = \text{Ecc-Encode}(K)$ {for error correction}
3: $C \leftarrow \text{DH-Preimage}(D, R)$ {challenge C is public}
4: Store C to one-time-programmable on-chip memory
5: Disable DH-Preimage {Blow fuse. See Fig. 2.9}
6: $R' \leftarrow \text{DH}(D', C)$ $\{D' \approx D \implies R' \approx R\}$
7: $K \leftarrow \text{Ecc-Decode}(R')$ {Regenerated secret key inside chip}

when a between-class distance is below the matching threshold. Perfect identification is possible when all within-class distances are smaller than all between-class distances, as it is then possible to choose a matching threshold that will produce a true positive identification for all within-class distances without any false positives.

The distance between DRVs $D1$ and $D2$ is computed as Response-Distance($D1, D2$) (Proc. 5). The first step of Proc. 5 is to choose a random response $R1$ (line 1) and generate for $D1$ a challenge C that is the preimage of $R1$ (line 2). Response $R2$ is then obtained by hashing C using $D2$ (line 3). If $D1$ and $D2$ are from the same chip, then the BER analysis of the previous subsection shows that $R1$ and $R2$ will also be similar. To perform identification in a challenging (high-BER) scenario, we use a short key of length 10 and a small SRAM with 20 cells (see Fig. 2.10). The distribution of within-class and between-class distances are shown in Fig. 2.11a, and the receiver operating characteristic (ROC) plot of Fig. 2.11b shows the identification performance for the data. Each ROC curve traces tradeoffs between true positive and false positive identification that can be achieved by changing the matching threshold. The three ROC curves in Fig. 2.11b compare identification of across-temperature within-class distances against between-class distances at a single temperature; these three comparisons are chosen because they are the most challenging identification scenarios, as indicated by the largest overlaps between the two distributions (Fig. 2.11a).

We evaluate the performance of RESPONSE-DISTANCE for circuit identification (Proc. 5) by comparing its ROC curves (Fig. 2.11b) against the ROC curves obtained for the same data by our prior identification scheme [39]. In our prior work, letting the characterization of address i in $D1$ and $D2$ be denoted $\langle v1_i^0, v1_i^1 \rangle$ and $\langle v2_i^0, v2_i^1 \rangle$, the distance between $D1$ and $D2$ is given by VOLTAGE-DISTANCE$(D1, D2)$ (Eq. 2.10). The within-class and between-class distances according to VOLTAGE-DISTANCE have a larger overlap (Fig. 2.11c), and the corresponding ROC curves (Fig. 2.11d) have inferior performance because they admit in all cases a larger false positive identification rate for the same true positive identification rate.

$$\text{VOLTAGE-DISTANCE}(D1, D2) = \sum_i \sqrt{(v1_i^0 - v2_i^0)^2 + (v1_i^1 - v2_i^1)^2} \tag{2.10}$$

Procedure 5 RESPONSE-DISTANCE$(D1, D2)$: Compute distance between responses of two SRAMs when the same challenge is applied to both.

Input: $D1$ {DRVs for chip 1. $D1(a)$ is chip 1 DRV at address a.}
Input: $D2$ {DRVs for chip 2. $D2(a)$ is chip 2 DRV at address a.}
Output: x {the distance between $D1$ and $D2$}
1: Choose randomly $R1 \in \{0,1\}^{|D1|/2}$
2: $C \leftarrow$ DH-PREIMAGE$(D1, R1)$ {note: $R1 = \text{DH}(D1, C)$}
3: $R2 \leftarrow \text{DH}(D2, C)$
4: $x \leftarrow$ HAMMING-DISTANCE$(R1, R2)$
5: **return** x

2.6 Summary

This chapter presents a collection of techniques that allow the data retention voltage of SRAM cells to be used as the basis of a reliable PUF. We propose a machine learning approach for fast and accurate simulation-free prediction of DRV values. We present procedures DH and DH-PREIMAGE for reliable hashing based on DRV measurements, and demonstrate that the reliability of these approaches stems from

their use of DRV-ordering instead of the absolute DRV values that were previously proposed. We use a large data set of DRVs from circuit simulation to train and analyze our DRV-prediction scheme, and use a large data set of DRV measurements from SRAM chips to quantify the reliability of DH and DH-Preimage in key generation and identification applications.

CHAPTER 3

RELIABLE PUF DESIGN USING FAILURE PATTERNS FROM TIME-CONTROLLED POWER GATING

A common PUF mechanism is based on uninitialized power-up states of bistable storage elements, but any bias in these storage elements will reduce their sensitivity to process variations and induce correlated outputs. Alternatively, a high quality PUF can be created by exploiting differences in the failure propensity of instances of identical storage cells, and this approach does not require power-up states to be unbiased. Contrary to previous failure-based PUFs that induce failures by controlling supply voltage explicitly, in this chapter we propose a failure-based PUF that uses failures induced by controlling the duration of power gating. We demonstrate the approach in simulation using D flip-flop circuits, and show that it reliably produces high quality output bits.

3.1 Introduction

Storing static on-chip identifiers (IDs) with non-volatile memory is a conventional method to identify integrated circuits. However, such identifiers are vulnerable to attacks and are trivial to clone. On the contrary, converting physical features of circuitry into unique fingerprints has several advantages and many possible implementations. In literature, circuits that translate physical features into unique IDs are called physical unclonable functions (PUFs) [26, 77], or more specifically within PUF literature as physically obfuscated keys [27] or weak PUFs [31]. Since PUFs are harder to clone, they can lead to more secure chip IDs and keys. The outputs of PUFs are sensitive to

various noise sources and this hinders their practical applicability. Much effort has been spent on correcting the bit errors caused by environmental fluctuations, such as using error correction codes [19], or adding helper circuitry for calibration or temporal majority voting [73].

The power-up state of static random-access memory (SRAM) has been proposed as the basis for PUFs in FPGAs [31] and micro-controllers [36]. However, modern FPGAs initialize or reset the values of all SRAM blocks at time of power up, and this renders SRAM PUFs unsuitable for FPGAs. For building PUFs in FPGAs, Maes et al. [65] propose using the power up states of flip-flops instead of SRAM. Unlike SRAM, flip-flop circuits have biased power up states and can only produce high quality outputs through complicated post-processing methods [65].

Independent of any bias in power up state or other fingerprint, any system with a number of identical storage cells will have some asymmetry in how failure-prone the various cells are. Stated differently, if the cells are subjected to the same conditions, some cells will be more likely to fail than others. This asymmetry can be used as the basis of a fingerprint. Yet, memory cells are designed so that failures will not occur in normal operation, and therefore some steps must be taken to induce the failures. In previous works, failures have been induced by manipulating the supply voltage of SRAM arrays [39, 119], or by lengthening the refresh interval of DRAM arrays [86]. In this work, we propose manipulating the duration of power gating as a way of inducing device-identifying failures and we demonstrate a PUF design based on this principle.

Power gating is an energy-saving technique for embedded systems [42] in which the power supply is effectively turned off for inactive parts of the chip. Power gating is common in modern embedded systems, including the Xilinx 7 series FPGAs [43], and the ARM Cortex-M series processors [46]. An example schematic of power-gated logic is depicted in Fig. 3.1; a PMOS transistor is inserted between the normal supply voltage (V_{DD}) and a virtual supply voltage (V_{DD_V}) that directly powers the logic.

A "high" sleep signal will cut the connection to the virtual power supply so that the gated circuitry will not draw power from V_{DD}. Utilizing power-gated DFFs, we make the following contributions in this work:

- We utilize retention failure counts of storage cells to create a PUF that can be used even when the storage cells are strongly biased in their power-up states;
- We propose a characterization algorithm to extract failure counts and reliably map them to responses in an experimental dataset containing 10,000 DFF cells;
- We present the first technique that extracts identifying information from retention failures of CMOS cells without requiring the ability to explicitly control supply voltage.

The remainder of this chapter is structured as follows: Section 3.2 reviews related work on PUFs. Section 3.3 introduces data retention failures, their measurement based on power-gating and their sensitivity to temperature. Section 3.4 presents our approach and evaluates it on DFFs. Section 3.5 presents directions for future work and concludes this chapter.

3.2 Related Work

Chip identifiers based on features of pre-existing circuits are an economical solution, and many techniques have been proposed for this purpose. Perhaps the most popular such approach exploits the power-up states of SRAM cells as chip IDs [31, 37]. Maes et al. [65] propose using the power-up state of DFFs as on-chip identifiers for FPGAs, but the power-up states of DFFs have significant bias due to design asymmetry.

An alternative to using power-up state of memories is to extract identifiers from retention failure signatures in various types of memories. Identifiers of this type can be extracted from the minimum data retention voltages of SRAM cells [39]. Bacha et al. [8] show that device-identifying retention failures can be observed through the error correction logic of caches on existing processors. Xu et al. [119] propose a key

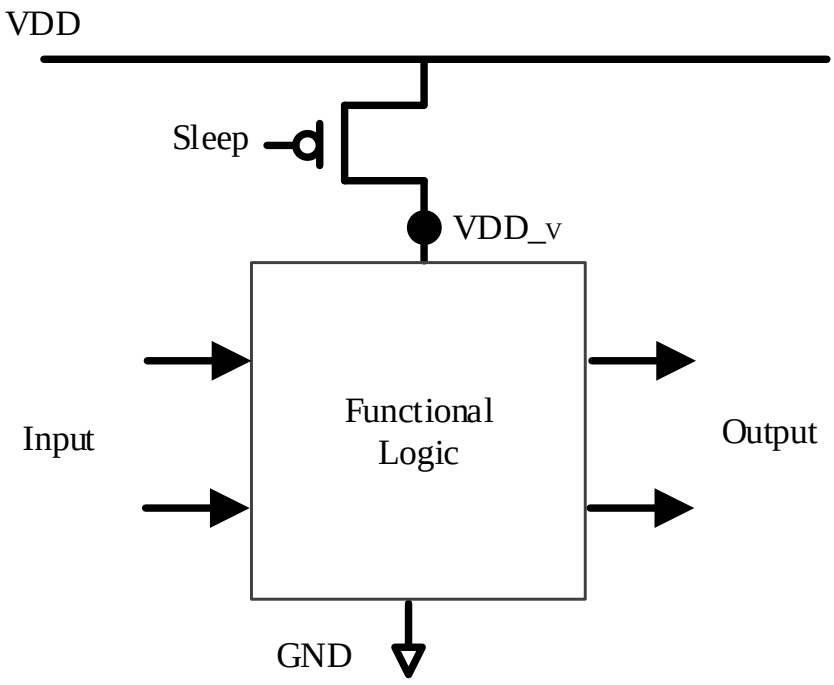


Figure 3.1: Schematic of power gating using a header switch. A PMOS transistor is employed between the supply node (V_{DD}) and the virtual supply node (V_{DD_V}) that directly powers the block. A sleep signal enables and disables the connection between V_{DD} and V_{DD_V}.

generation scheme based on SRAM retention voltages that can produce identifiers which are reliable across temperature variations. Other researchers have proposed DRAM PUFs [88] by using identifying properties of DRAM failures, and several other works have shown that write failures [34] and retention failures in approximate DRAMs under lengthened refresh intervals [85] also produce identifying signatures. Borrowing ideas from failure-based PUFs, and time-controlled failures in DRAM, we propose a time-controlled failure-based PUF for CMOS storage cells that be actuated through power gating.

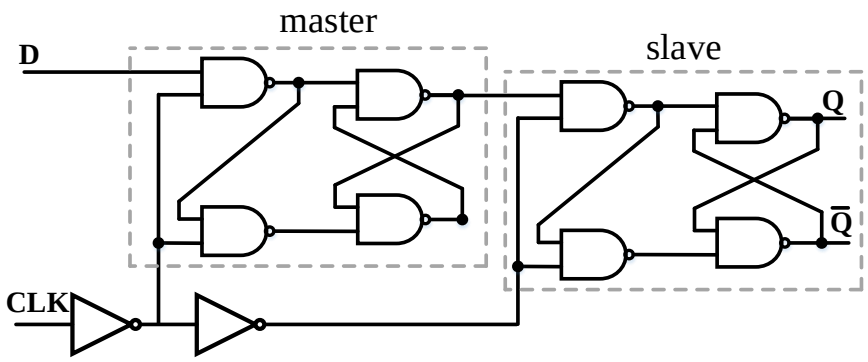


Figure 3.2: Schematic of a positive-edge triggered master-slave D flip flop. Q and $\bar{Q}$ are the complementary state nodes that store a single bit value between cross-coupled NAND gates. The input value D is stored in the master latch when CLK rises. Nearly simultaneously, the slave latch opens to allow the stored signal from the master to propagate through the slave to the output.

3.3 Inducing Failures using Timing

3.3.1 Retention Failures at Low Supply Voltage

CMOS storage cells can fail to retain stored values if the supply voltage is too low. In the case of SRAM cells, the minimum supply voltage at which data can be safely stored is denoted the data retention voltage (DRV). SRAM must therefore remain above DRV to function correctly. While remaining above DRV, the supply voltage can be adjusted to reduce leakage power [25, 12], compensate for manufacturing variability [75], or compensate for environmental variations [113].

Our work borrows the idea of DRV from SRAM literature and applies it data retention of DFF cells. A positive-edge-triggered master-slave DFF cell can be constructed from two D latches as depicted in Fig. 3.2. Like SRAM, DFF cells can fail to retain data when voltage drops too low. Unlike SRAM cells, DFF cells have biased power up states [65]. Fig. 3.3 shows the power-up of 30 DFF cells, and it can be observed that all cells take the same power-up state of $Q = 1$. This bias makes the power-up states of the cells unsuitable as fingerprints. In spite of having

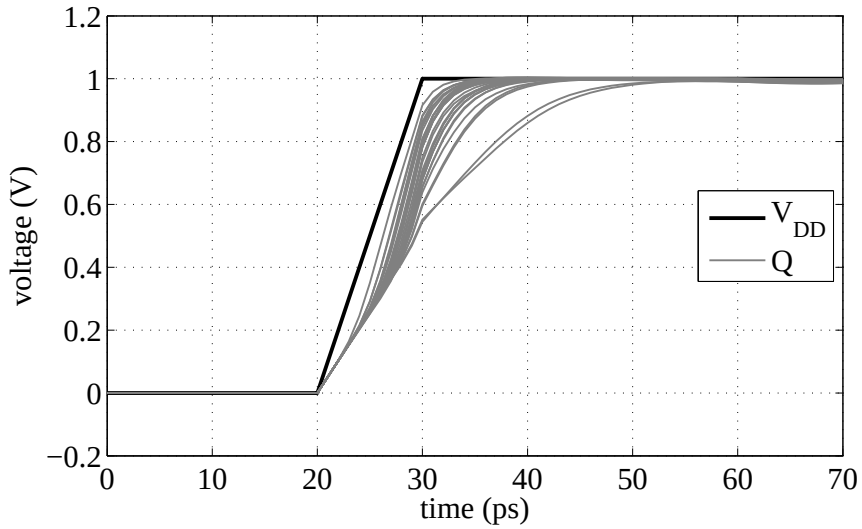


Figure 3.3: Power up simulation of 30 D flip-flop cells showing that all cells power-up to the $Q = 1$ state. This consistent bias makes their power-up state unsuitable as a fingerprint.

indistinguishable power-up states, these cells may experience retention failures at different supply voltages.

3.3.2 Power Gating Duration as a Proxy for Supply Voltage

Finding the voltage at which each cell will fail, as is required for DRV-based fingerprints [39], generally relies on voltage control and this causes the fingerprint generation procedure to be very complicated. Prior work shows that the absolute retention failure voltages are not necessary and that a reliable identifier can be extracted from the relative ordering of the failures [119], but in this prior work the ordering is still extracted by controlling the supply voltage.

Our proposed technique effectively derives ordering of retention failures across cells without explicitly controlling the supply voltage. Instead of voltage control, power-gating duration is the controlled variable that reveals the relative failure ordering of cells. The minimum voltage applied to the cells, which is the direct cause of the retention failures, is simply an effect of the power gating duration. Fig. 3.4 shows

a header switch, or sleep transistor, added between the power supply node V_{DD} and a set of DFF cells. When the gate of the sleep transistor is 0 it will be turned on and the DFFs will operate normally. If the sleep transistor is temporarily turned off by a high gate voltage, node V_{DD_V} will decay and then be recharged once the sleep transistor is turned on again. If V_{DD_V} falls sufficiently low during this time, then some DFFs may lose their stored values. In the simulation, nodes D and CLK of each DFF are held low during power gating.

A longer sleep duration allows V_{DD_V} to reach a lower voltage, and will cause more cells to lose their stored values. In this way, controlling the sleep duration controls the minimum voltage applied to the DFF cells. By observing the failures that occur for each different sleep duration (i.e. at each different minimum V_{DD_V} voltage), it becomes possible to learn which cells are more or less failure prone than others. This idea of controlling power gating duration as a proxy for controlling supply voltage is a key idea of our work. The complexity of circuit required to control timing is trivial compared to that of controlling voltage. Digital-to-time converters can be made small and inexpensive, whereas the ability to control the supply voltage to a particular block of storage cells requires at least a regulator and additional voltage domain.

The simulation result in Fig. 3.5 shows the decay of V_{DD_V} during the sleep time, using a test circuit with 30 DFF cells built with 45nm PTM library [127]. In Fig. 3.5a, a relatively shorter sleep duration is applied and no DFF cells lose their values because V_{DD_V} remains sufficiently high. When the sleep time increases (Fig. 3.5c), a lower voltage is reached and more DFF cells lose their written "0" state and transition to the "1" state when power is restored after gating.

3.3.3 Impact of Temperature

The lowest voltage reached by V_{DD_V} during power gating is related not only to the duration of power gating, but also to the temperature. A high temperature will

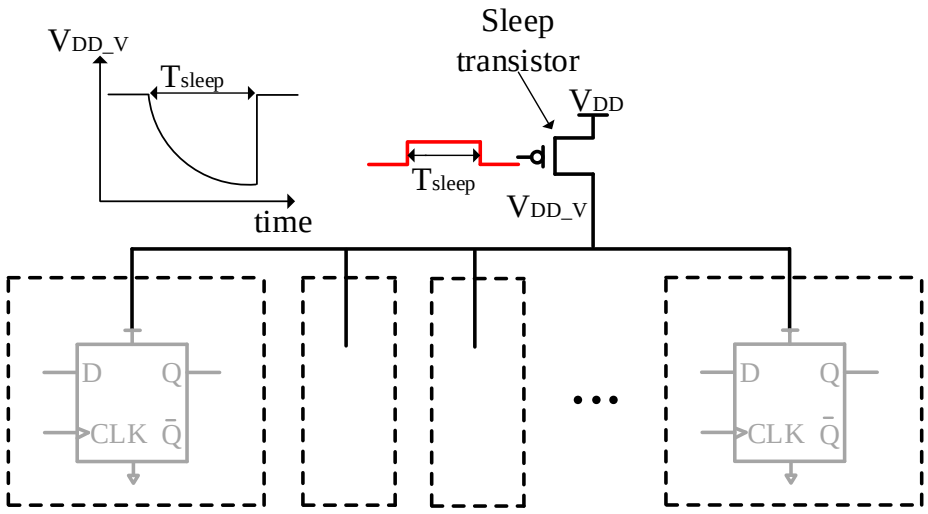
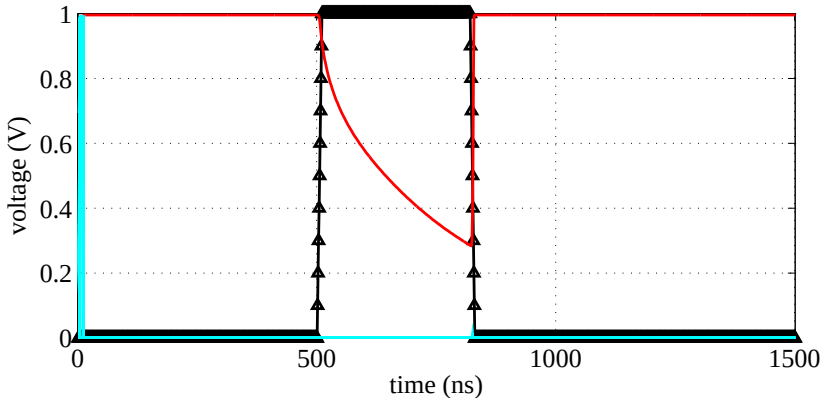


Figure 3.4: A group of DFF cells with a PMOS sleep transistor added for power gating. When the sleep signal is low, the sleep transistor is on, and the DFF cells operate as normal. When the sleep signal is high, V_{DD_V} will decay and may cause cells to lose state. The amount of decay in V_{DD_V} depends on the duration of the sleep signal as well as temperature.

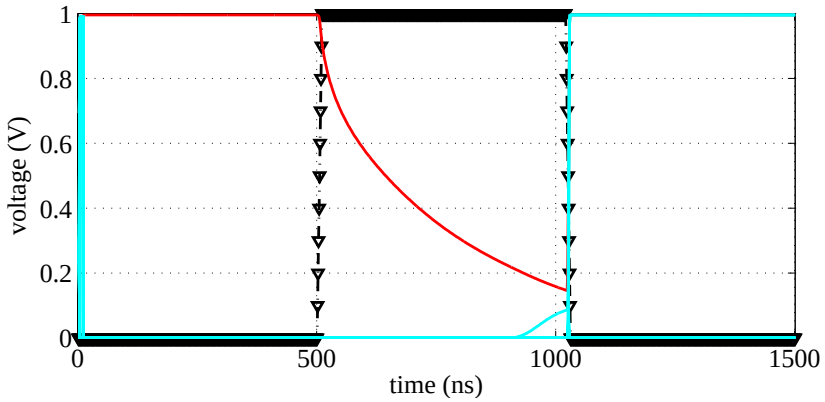
cause V_{DD_V} to decay more quickly. To examine the temperature-sensitivity of V_{DD_V}, we simulate its decay under different temperatures from 0 o 100 Given the same sleep duration, the simulation result in Fig. 3.6 shows that, V_{DD_V} decays faster at higher temperature.

3.4 Extracting A Signature From Failure Rates

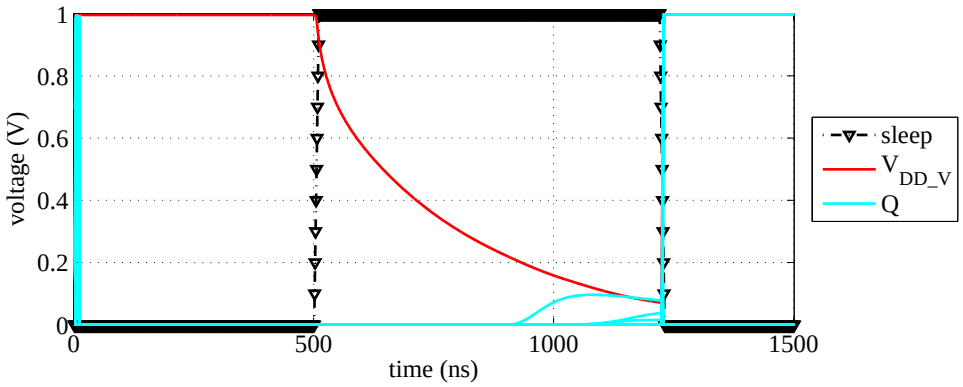
In this section, we extract a fingerprint that is based on failure counts of a collection of storage elements, in our case DFFs. Failure count is measured by adaptively subjecting the cells to a variety of different power gating durations, and counting how many of these power gating trials cause each cell to fail. By observing the number of failures (i.e. the failure count), we learn the failure propensity of each cell relative to the overall population; in a simple case, if one cell fails 8 times while other cells have a median of 3 failures, then it is known that the chosen cell is particularly prone to failures. Yet, we must supply the right conditions to make this determination. If



(a) T_{sleep}=300ns



(b) T_{sleep}=500ns



(c) T_{sleep}=700ns

Figure 3.5: The simulation results of 30 DFF cells during power gating. All of the initial written values to the cells are "0". In Fig. 3.5a, no cells lose their "0" state because of the short sleep duration. In the longer power gating durations of Fig. 3.5b and 3.5c, more cells fail to retain the original states. This is caused by the natural bias in DFF; when the supply voltage decreases, the Q competes with $\bar{Q}$, and a low enough supply voltage leads to a data retention failure.

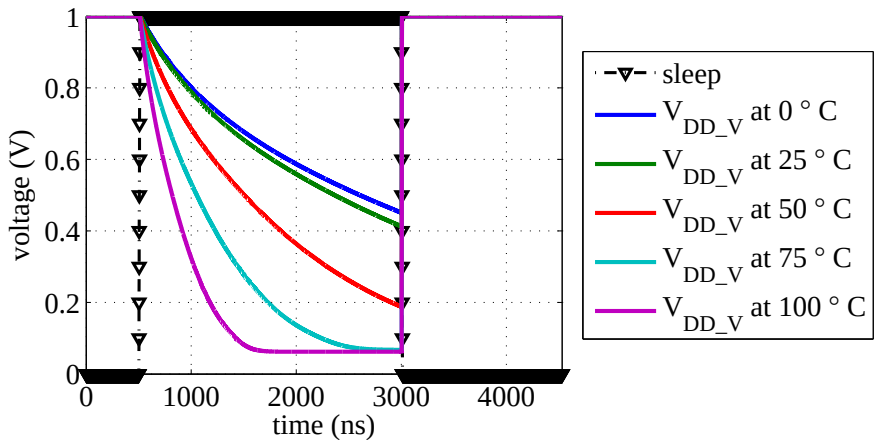


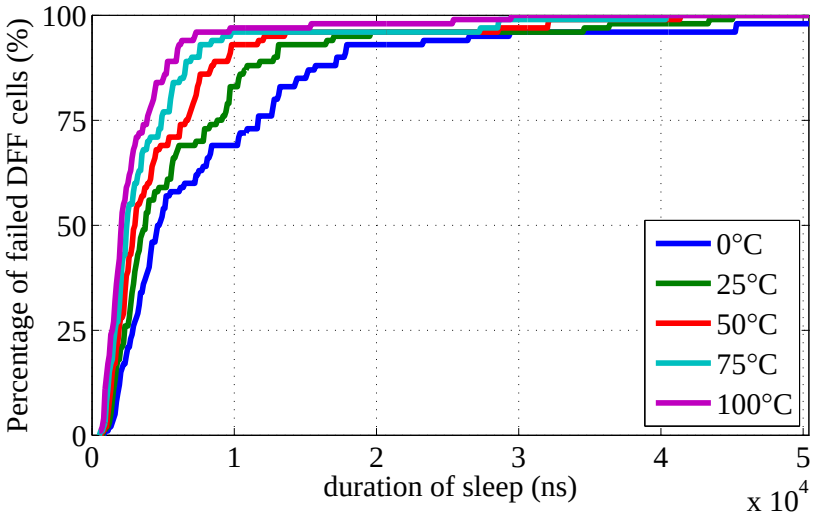
Figure 3.6: Decay of V_{DD_V} during power gating at different temperatures. A higher temperature speeds up the decay of V_{DD_V}.

applying power gating scenarios that cause no cells to fail, or all cells to fail, then nothing is learned. Ideally, we therefore seek to apply conditions that cause half of cells to fail.

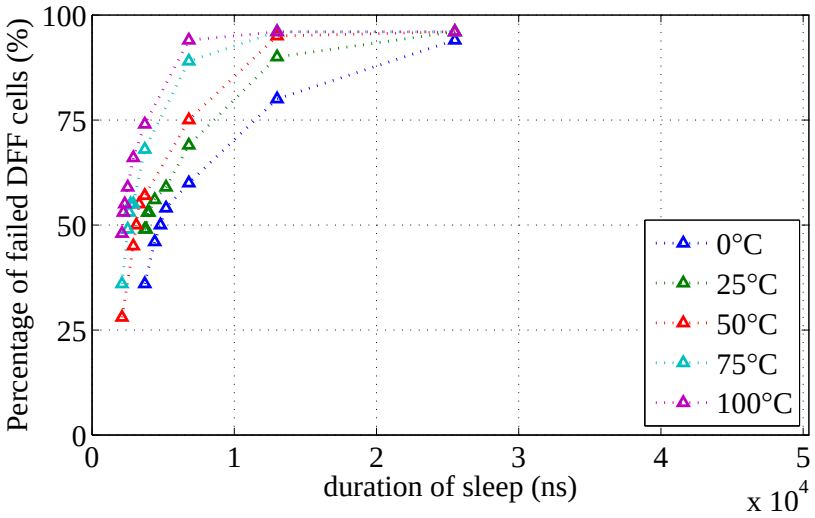
Fig. 3.7a shows the number of cells that fail for different durations of power gating. This data is obtained by linearly sweeping the duration of power gating after writing "0" to 10,000 DFF cells, and counting failures when power is restored. Because the DFF cells are biased toward the "1" state, almost all cells fail when the power gating is long enough. To build DFF PUFs based on the failure count feature, we propose a two-step scheme comprising an enrollment process and a key (response) generation process.

3.4.1 Enrollment Process

Enrollment would happen at the manufacturer when the chip is created, and key generation would occur in the field to produce the secret key. The enrollment phase is described in Proc. 6. Repeatedly, all DFF cells under test are written with 0, then power is gated for some time and subsequently restored, and the cells are checked to see whether they have retained the written value. The duration of power gating is



(a) sleep pulses duration linearly increases



(b) sleep pulse duration controlled by binary search

Figure 3.7: Simulation results showing that the percentage of DFF cells experiencing retention failures varies with the duration of power gating and for different temperatures. Increasing the sleep duration causes more cells to fail. Additionally, increasing temperature speeds up retention failures because V_{DD_V} decays faster at higher temperatures (see Fig. 3.6). The plot at right shows specific sleep durations that are explored by binary search seeking to find the gating duration that causes 50% of cells to fail at each temperature.

Procedure 6 Enrollment phase, characterize and sort the failure count of a set of DFF cells, generate the helper data **H** based on the given key **K**.

Input: **DFF**: a set of indexed DFF cells
Input: An n-bit key $\mathbf{K} = (k_0, k_1, \ldots, k_{n-1})$
Output: $\mathbf{H} = \big(h_0, h_1, \ldots, h_{n-1}\big)$

1: $\mathbf{FC}$ = COUNT-FAILURE(**DFF**)
2: $\mathbf{DFF}_{HFC} \leftarrow \{\forall j \in \mathbf{DFF} \mid FC(j) > 70^{th}\ percentile\}$
3: $\mathbf{DFF}_{LFC} \leftarrow \{\forall j \in \mathbf{DFF} \mid FC(j) < 30^{th}\ percentile\}$
4: **for** $i := 0$ **to** $n - 1$ **do**
5: **if** $(k_i == 1)$ **then**
6: choose h_i at random from $\mathbf{DFF}_{HFC}$
7: remove h_i from $\mathbf{DFF}_{HFC}$
8: **else**
9: choose h_i at random from $\mathbf{DFF}_{LFC}$
10: remove h_i from $\mathbf{DFF}_{LFC}$
11: **end if**
12: **end for**
13: **return H**

varied at each trial, and the total number of failures observed in each cell is counted across all trials. This procedure records the failure count of all DFF cells. and creates two sets of cell indices: $\mathbf{DFF}_{HFC}$ containing the indices of the 30 percent of cells with the most failures, and $\mathbf{DFF}_{LFC}$ containing the indices of the 30 percent of cells with the fewest failures. The cells that are close to the median failure counts are not in either set. The two sets of cell addresses are used to map a chosen key **K** to helper data $\mathbf{H} = \big(h_0, h_1 \ldots h_{n-1}\big)$. The helper data will subsequently be used to regenerate the key. The helper data need not be secret, but an adversary should not be allowed to manipulate helper data.

3.4.2 Key Generation Process

In the field, the helper data and new measurements of failure counts can be used to generate the enrolled key. The task of key generation is to discover, for each cell index in the helper data, whether that cell is unusually prone to failure or unusually resistant to failure relative to the overall cell population. The key generation procedure is given

in Proc. 7. In this scheme, the PUF outputs are chosen by mapping each DFF index in the helper data to a corresponding 0 or 1 output according to whether its failure count is above or below the median failure count of the cells. Note that the helper data is the only state retained from the enrollment, and that all failure counts, and the computed median failure count, are obtained from the new measurements in the field. The relative failure counts obtained during key generation can differ from those used at enrollment, and it is for this reason that the enrollment chooses only cells in the upper or lower 30% of failure counts; as long as a cell that was in the upper (or lower) 30% of failure counts during enrollment remains in the upper (or lower) 50% during key generation, then the output will be correctly generated.

3.4.3 Binary Search

If sleep durations are not carefully chosen, many of the applied sleep durations may yield little discriminating information. For example, a very long sleep duration may cause all cells to fail and this tells nothing about relative failure propensities of cells. Noting that the most discriminating power gating durations are the ones that cause half of cells to fail, we propose to use for the Count-Failure routine in key generation (Proc. 7) a binary search procedure that adapts the gating duration in search of the specific gating duration that induces 50% failures. The number of failures for each cell is counted over the steps of the binary search. For the same cells shown in Fig. 3.7a, the markers on the plot in Fig. 3.7b show the specific power gating durations that are explored by a binary search algorithm at each temperature, and the fraction of cells that fail in each power gating trial. It can be observed that the algorithm targets the 50% failure duration, and thus the data points are clustered around 50% for each temperature. Note two key features of the proposed approach:

Procedure 7 PUF response generation: generate responses **R** from a set of DFF cells using helper data **H**.

Input: **DFF**: a set of indexed DFF cells
Input: $\mathbf{H} = \big(h_0, h_1, \ldots, h_{n-1}\big)$
Output: $\mathbf{R} = \big(r_1, r_2, \ldots, r_{n-1}\big)$

1: $\mathbf{FC} = \textsc{Count-Failure}(\mathbf{DFF})$
2: **for** $i := 0$ **to** $n-1$ **do**
3: **if** $\big(FC(h_i) \geq \underset{\forall j \in \mathbf{DFF}}{\text{median}}\ FC(j)\big)$ **then**
4: $r_i \leftarrow 1$
5: **else**
6: $r_i \leftarrow 0$
7: **end if**
8: **end for**
9: **return R**

- It is adaptive, and for any temperature will find the appropriate gating duration to provide highly discriminating failure counts that separate the most failure-prone cells from the least failure-prone cells.

- It implicitly has characteristics of a majority voting scheme in the sense of being robust to noise-induced upsets during a single power-gating trial.

3.4.4 Experiment Results

We evaluate the reliability of the failure count based DFF PUFs and depict the results in Fig. 6.7. The figure shows, for different number of iterations, the mean PUF reliability across 500 trials and the reliability obtained by comparing data from 25°C to 100°C. For comparison, we show that the same PUF reliability can be obtained by using random durations of power gating, but this requires more trials because it is does not target highly informative trials. In the case of random power gating, the durations are randomly chosen from the range 0 to 50 μ. When the number of iterations increases, the failure counts produced are better able to identify which cells are unusually prone to failure or unusually resistant to failure. Given that power

gating iterations will increase runtime, there exists a tradeoff between runtime and the reliability of the generated keys, but in either binary search or random durations, a larger number of iterations will improve key reliability. To maximize reliability in a small number of trials, binary search is a better choice.

3.5 Summary

In this chapter, we propose a novel method to use power gating to build reliable PUFs from failure signatures in CMOS storage elements such as D flip-flops. The work is notable in its ability to generate high quality identifiers from biased cells, and it performs best when cells are biased. The method exploits power gating as a mechanism to extract a chip-specific ordering of failures across cells, and does so without requiring the expensive voltage control that was needed by previous works in this area. The method is shown to be reliable and can use either random durations of power gating or an efficient adaptive method to characterize failure counts across temperatures.

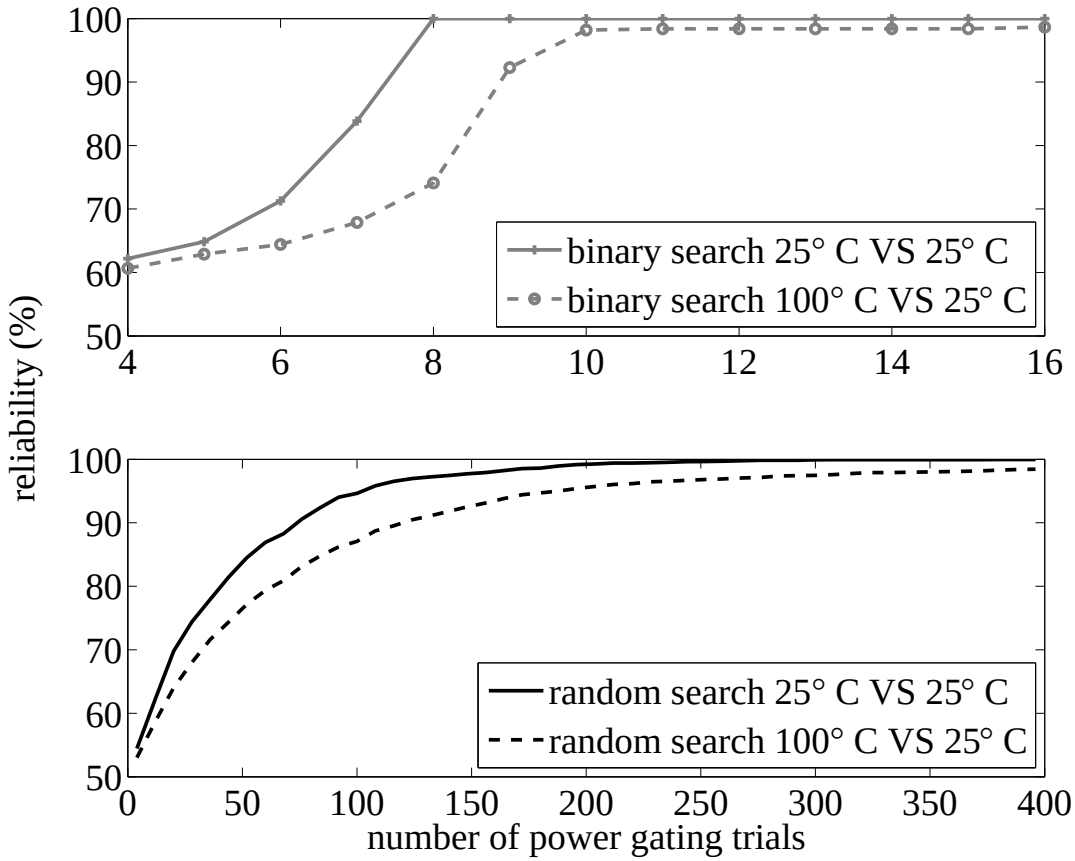


Figure 3.8: Plot shows the reliability of generated keys produced using binary search and random durations of power gating. The key values are enrolled to the cells at 25°C using Proc. 6 and then the keys are generated at 25°C and 100°C using Proc. 7. The binary search is able to produce highly reliable keys using fewer power gating trials because it seeks out gating durations that are most capable of distinguishing the cells that are failure prone from those that are not. The plotted data is averaged over 500 trials.

CHAPTER 4

SECURITY EVALUATION AND ENHANCEMENT OF BISTABLE RING PUFS

Bistable Ring (BR) Physical Unclonable Function (PUF) is a newly proposed but promising hardware security primitive in PUF family. The novel architecture not only makes it capable of generating exponential responses, but also being harder to model and machine learn than the Arbiter PUF. In this chapter, we evaluate the security of Bistable Ring PUF by applying different attacking models and methods, besides the proposed attacks in other literatures, we propose a novel model based on Support Vector Machine (SVM). XORing method is employed to enhance the attack resilience of single BR PUF, we demonstrate that XORed BR PUF is beyond the reach of current known attacking models, thus is secure.

4.1 Introduction

At TRUST 2014, Hesselbarth and Schuster [98] succeeded in revealing some basic vulnerabilities of the BR PUF against ML techniques. They proved that BR PUFs can be attacked by a single layer artificial neural network (ANN) with prediction errors between close to 0% and 20%, varying from hardware instance to instance. Among the 20 FPGA instances examined, 14 could be predicted with errors less than 10%. This puts close limits on the security usability of the BR PUF on FPGAs. Schuster and Hesselbarth subsequently proposed a small design improvement, so-called twisted BR PUFs (TBR PUFs), which they conjectured to possess better security. Using their own ANN algorithm, they were also able to attack TBR PUFs again. However, the

TBR PUF shows average higher prediction errors with respect to ANNs, indicating that TBR PUFs has some improvements over plain BR PUFs. It remained open in [98] whether said improvement is sufficient for secure practical usage of the TBR PUF.

Our Contributions In this chapter, we re-examine the security of the BR PUF and TBR PUF closely, again using FPGA implementations. We thereby make the following new contributions:

- We implement 8 instances of the BR PUF and the TBR PUF on FPGA. To achieve a more comprehensive ML analysis, we implement bitlengths other and larger than 64, namely also 32, 128 and 256. These bitlengths had never before been implemented in silicon and studied in the literature.

- We develop the first analytical models for the BR PUF and the TBR PUF.

- We use these new models in order to apply, for the first time, support vector machines (SVMs) to the BR PUF and the TBR PUF. This more powerful ML-tool drastically improves the ML predication rates relative to previous work. None of our 8 instances has a prediction error exceeding 5%. This result answers the open question of Hesselbarth and Schuster whether certain individual and particularly hard instances of the BR PUF or TBR PUF could be used securely in practice: In our findings, this was not the case.

- We then propose a new, efficient strategy for the secure practical use of the BR PUF: namely the employment of l instances in parallel, whose l outputs are XORed at the end in order to produce one single-bit PUF-response. We call the resulting structure XOR BR PUF. We show that even for small values of l such as 4, this structure cannot be machine learned by our current techniques, while it is still sufficiently stable in practice. This work is the first study of XOR BR PUFs in the literature.

This chapter is organized as follows. Section 4.2 discusses our attacks on the BR PUF, while Section 4.3 details our attacks on the TBR PUF. Section 4.4 suggests the use of XOR BR PUFs and evaluates their performance improvement. Section 4.5 concludes this chapter.

4.2 SVM Attack on BR PUFs

4.2.1 Mechanism of BR PUF

A ring oscillator (RO) is a device composed of an odd number of logically-inverting delay elements. Since the output of the last element is always the logical "NOT" of the first input, an RO will continually oscillate. Derived from the non-settling structure of RO, BR PUF is a ring comprising an even number of inverting cells. Such a design behaves like a memory cell and will fall into one of two possible stable states: either "101010..." or "010101...".

As depicted in Fig. 4.1, a 64-bit BR PUF is composed of 64 stages, where each stage has two inverting delay elements (NOR gates as an example). A challenge vector $C = \{c_1, c_2, \ldots, c_n\}$ selects the NOR gates used in each bistable ring configuration by providing values to the MUX and DEMUX gates of the stages. Since each NOR gate has unique process variation, each different challenge vector creates a unique bistable ring configuration, and in total 2^{64} different configurations can be created. A common "RESET" signal is added to each stage to establish a known "all-0" state before letting the ring stabilize to produce its response. Evaluation of the response begins when "RESET" is released and the ring starts to oscillate through the selected NOR gates. Once a stable state is reached, the outputs of any two adjacent stages will be logical compliments of each other, either "10" or "01". The choice among the two possible stable states of the ring depends on noise and the process variation of the NOR gates used in the ring configuration. Any interconnection node between two stages can be

used as an output port, and in this work we use the half bit-length port to read out the response (Fig. 4.1).

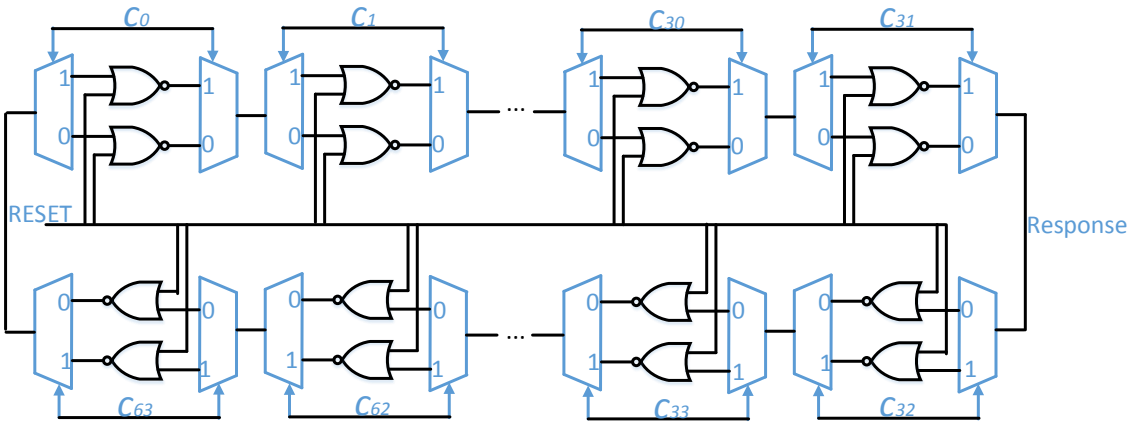


Figure 4.1: Schematic of a single BR-PUF with 64 stages.

4.2.2 Intuition for Modeling BR PUF

The intuition for our modeling attack is that the response can predicted based on a summation of weights. Such an additive model is commonly used for predicting the responses of Arbiter PUFs, where the weights represent stage delays [59]. An additive model has also been used for predicting the resolution of metastability [38], with weights representing the strength with which different cells pull toward a particular outcome. We similarly use an additive model in this work; the weight we associate with each gate represents the difference between its pull-up strength and pull-down strength. The weights are summed across all gates used by a challenge to find the overall favored response for that challenge; a positive sum indicates a preference for the positive response. Note that the summation of weights requires negative and positive polarities because the positive overall response is favored by the pull-up strength of even stages and the pull-down strength of odd stages.

4.2.3 Model

Let the difference between the pull-up and pull-down strength of the top NOR gate in the i^{th} stage be represented by t_i, and in the bottom NOR gate in the i^{th} stage be represented by b_i. The even stages will contribute toward the positive response with strength t_i (or b_i if the challenge bit selects the bottom NOR gate of the stage), and the odd stages will contribute toward the positive response with strength $-t_i$ (or $-b_i$). To account more generally for even-ness or odd-ness, the strength of the i^{th} stage toward the positive response can be written as $-1^i t_i$ if the challenge bit is 0, and $-1^i b_i$ if the challenge bit is 1. For a given 64-bit challenge, the total strength pulling toward the positive response is the summation of 64 t_i and b_i weights.

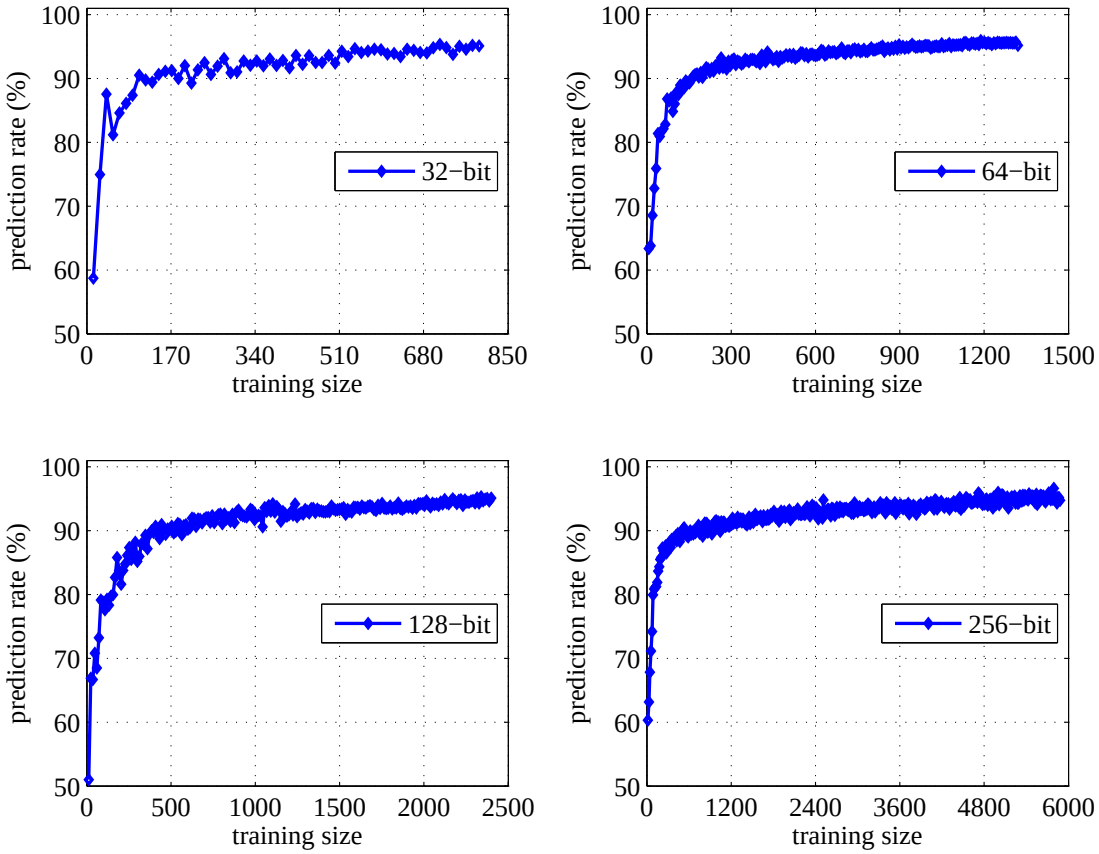


Figure 4.2: Prediction rate of SVM modeling attacks on BR PUFs. When the length of the BR PUF increases, more CRPs are required to train the model to achieve 95% prediction. Note that the scale of the x-axes are not consistent across the subfigures.

For convenience we define α_i and β_i (Eq. 4.1) such that $\alpha_i + \beta_i = -1^i t_i$ and $\alpha_i - \beta_i = -1^i b_i$. This notation allows the pull of the i^{th} stage toward the positive response to be written generally as $\alpha_i + c_i\beta_i$ with challenge bit $c_i \in \{-1, 1\}$. The summed strengths toward the positive response for any challenge vector C is $r(C)$ as shown in Eq. 4.2. According to our formulation, if the t_i and b_i weights were known explicitly, then the response could be predicted by the sign of $R(C)$ (Eq. 4.2).

$$\alpha_i = -1^i \left(\frac{t_i - b_i}{2}\right) \qquad \beta_i = -1^i \left(\frac{t_i + b_i}{2}\right) \tag{4.1}$$

$$R(C) = sgn(\sum_{i=0..n-1} \alpha_i + c_i\beta_i) \tag{4.2}$$

Given that weights are not known, since there are only two possible responses of BR PUFs, based on the model above, we can convert the response prediction of BR PUFs into a classification problem. Support Vector Machines (SVM) are powerful learning tools that can perform binary classification of data, the classification is realized with building a hyperplane separating surface. While digesting the known input and output data sets, the hyperplane separating surface will be curved to minimize the error of predicted values.

Known CRPs are used to train the classifier to predict responses from challenges. In the SVM formulation, first note that the α_i terms in Eq. 4.2 can be discarded because they are constant for a given PUF instance across all challenges. Only $c_i\beta_i$ terms remain, and from these terms the response must be predicted. Given a set of challenges and associated responses, the training examples therefore have as their feature vector an applied challenge $C_j \in \{-1, 1\}^n$ and as their labels the observed response $R(C_j) \in \{-1, 1\}$ to challenge C_j. Note that β_i terms do not appear explicitly in the SVM formulation as the classifier simply works to find the maximum margin hyperplane to separate the challenges into two classes according to their responses.

4.2.4 SVM Attacks on BR PUFs

To explore the effectiveness of SVM attacks, we implemented on a Xilinx Spartan-VI FPGA board, 8 BR PUFs with lengths of 32-, 64-, 128- and 256 bits, and collected 1,000,000 CRPs from each of them (to decrease the impact of measurement noise, all of the final CRPs are formulated by majority voting from 11 repeated measurements). SVM attacks are implemented with a linear kernel to mimic the operation of single BR PUFs (note that to attack XOR BR PUFs, SVM model with a polynomial kernel is utilized, where the poly-order of the model is set as the XORing complexity of BR PUFs). The results of SVM attacks are shown as Fig. 4.2. To demonstrate the relationship between prediction rate and CRPs used for different PUF lengths, we utilize 95% as a threshold prediction rate. It is clear that while the size of BR PUF is increasing, the demand for CRPs is also increasing to build its ML model. However, for any tested size of BR PUF, the SVM modeling attack is successful in predicting responses. This means a single BR PUF is not secure, even if it has a large number of stages.

4.3 Twisted BR PUFs Attack

4.3.1 Model of TBR PUFs

Uniformity, or fractional Hamming weight, is an important feature of PUFs. A good PUF that produces an equal number of 0 and 1 responses will have a uniformity of around 0.50. However, the uniformity of CRPs of BR PUF implementations has been found to be biased in previous work [98] (see also Sec. 4.4.3.3 in this work). To compensate for this drawback, TBR-PUF was proposed in [98]. Compared to the BR PUF, the TBR-PUF has a more compact design; when applying a challenge vector to the TBR PUF, all of its $2n$ inverting elements are used in the ring. By contrast, in the standard BR PUF, half of the NOR gates in the circuit are unused for any given challenge. Taking the TBR PUF in Fig. 4.3 as an example, using challenge bit $c_0 = 1$

or $c_0 = 0$ will change the location of D_0^0 and D_1^0 in the ring, but in either case D_0^0 and D_1^0 will both contribute in some polarity to the response.

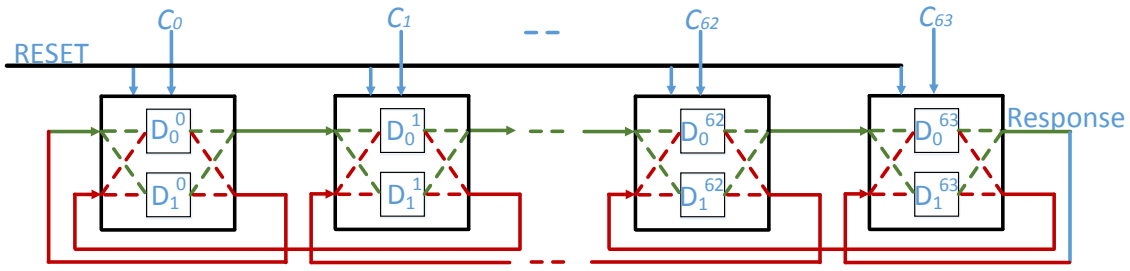


Figure 4.3: Schematic of a single TBR-PUF with 64 stages.

From Sec. 4.2, we know that a ring composed of an even number of inverting elements will stabilize according to the summed strength of the pull-up and pull-down strengths of each gate. The TBR PUF uses pull-up and pull-down strengths of all inverting components in the circuit, but only the polarity (i.e. even-ness or odd-ness) of each element toward the overall ring response changes with the challenge vector. According to the interconnections of the 64-bit TBR PUF, the two NOR gates in the i^{th} stage are the i^{th} and $127 - i^{th}$ element in the overall ring. Because one element is odd in the overall ring, and one is even, the pull-up strength of the top and bottom gates in each stage are working against each other. Therefore, the overall contribution toward the positive response is β_i (Eq. 4.3), or $-\beta_i$ if the i^{th} challenge bit is negative. The overall sum of weights pulling toward the positive response for challenge C is therefore $R(C)$ (Eq. 4.4). Eq. 4.2 and Eq. 4.4 differ only in in the physical meaning of β_i, and in Eq. 4.2 having an additional offset term of $\sum_i \alpha_i$, but in terms of ML modeling they are actually the same identical model. Therefore, the complexity of ML attacks on the TBR PUF is the same as the complexity of attacking the BR PUF.

$$\beta_i = -1^i(t_i - b_i) \tag{4.3}$$

$$R(C) = sgn(\sum_{i=0...n-1} c_i \beta_i) \tag{4.4}$$

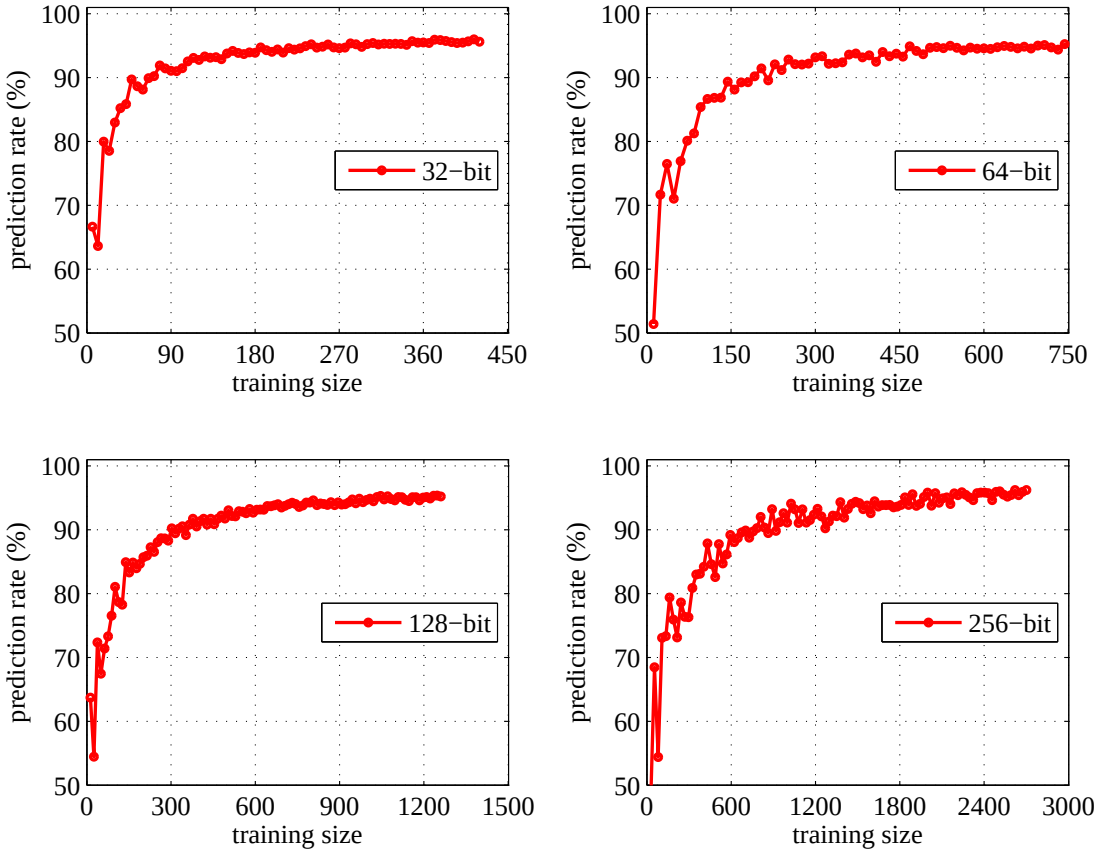


Figure 4.4: Prediction rate of SVM modeling attacks on TBR PUFs of different bit lengths. As in Fig. 4.2, to achieve same prediction rate, a larger PUF requires more CRPs.

4.3.2 SVM Attacks on TBR PUFs

Given that we have shown the model of a TBR PUF to be the same as that of a BR PUF, we can again train an SVM classifier to predict its responses to each challenge. Eight TBR PUFs are implemented with Spartan-VI FPGA boards, and 1,000,000 CRPs are collected from each of them. For each CRP, majority voting over 11 repeated measurements of the response to a challenge are performed in order to reduce the impact of noise.

Following the experiment in Sec. 4.2.4, SVM attacks with polynomial kernel are applied on TBR PUFs of 32-, 64-, 128- and 256 bit-length (the poly-order of the model is set as the XORing complexity). The results in Fig. 4.4 show that the modeling attacks succeed in modeling all different sizes of the TBR PUF, with prediction rate no lower than 95%.

4.4 XORing BR PUFs to Enhance the Security

It is possible using ML to model the behavior of a single strong PUF like the Arbiter PUF [59]. To thwart modeling attacks, an XOR function was proposed as a way to enhance security of Arbiter PUFs [103] and lightweight PUFs [71]. In an XOR PUF, the same challenge vector is applied to l single PUFs in parallel, and their outputs are XORed together to form a one-bit response. XORing is an efficient method to enhance the security of strong PUFs, because the XOR function obfuscates the CRPs of the individual PUFs [103]. Inspired by this idea, we propose to use XOR strategies on BR PUFs to improve their resistance to modeling attacks.

4.4.1 Review of Existing Attacks on XOR PUFs

The addition of XOR functions increases the resistance of strong PUF against modeling attacks. Both the training time and number of CRPs required to train a model increase exponentially with the number of XORed PUFs [93]. Attacking XOR-based Arbiter PUFs with more than five parallel Arbiter PUFs was stated as difficult based on pure ML modeling [90]. Later works devised a more powerful class of hybrid attacks that combine side channels with ML [95, 115]. Power and timing side-channels allow information about the sub-responses (i.e. the responses of single PUFs before the final XOR) of XORed PUFs to be extracted and used to improve the prediction rate of ML models. In light of these hybrid attacks, if the side-channel

No. of XORs	Bit Length	CRPs ($\times 10^3$)	Predict. Rate	Training* Time
2	32	0.8	95%	3 sec
	64	4	95%	10 sec
	128	18	95%	6 mins
	256	—	50.8%	—
3	32	1.2	95%	5 sec
	64	7.2	95%	24 sec
	128	—	50.1%	—
	256	—	50.1%	—
4	32	—	50.1%	—
	64	—	50.3%	—
	128	—	49.8%	—
	256	—	50.1%	—

Table 4.1: The run times and number of CRPs that are required for SVM attacks on the XOR BR PUFs of different sizes. Prediction rates around 50% imply that the SVM model can not break XOR BR PUFs of these complexity. *Note that the training time is greatly determined by the computational systems.

information of BR PUFs can also be measured, then the use of XOR will not be an effective way to enhance the security.

4.4.2 SVM Modeling Attacks on XORed BR PUF

Adopting the model of single BR PUF in Sec. 4.2, for an XOR BR PUF employing l BR PUFs, the XORed response to a challenge C can be described by Eq. 4.5. Note the similarity between this formula and the formula of the single BR PUF (Eq. 4.2). The only modification is that now each stage has l different α and β terms, one for each of the PUFs. The overall response is based on how many of the individual PUFs have a positive response.

$$R(C) = sgn\Big(\prod_{j=0}^{l-1}\big(\sum_{i=0}^{n-1}\alpha_{i,j} + c_i\beta_{i,j}\big)\Big) \tag{4.5}$$

In applying SVM to XOR BR PUF, it is found that we can only break the complexity up to 2 XOR for 128-bit length and 3 XOR for 64-bit length. The number

of CRPs and runtime[1] for SVM modeling attacks against XOR BR PUFs are listed in Tab. 4.1. We can surmise that XOR BR PUFs with 4 or more XORed outputs are beyond the reach of current SVM modeling attacks.

4.4.3 Performance Evaluation of XORed BR PUF

While the basic motivation of XORing BR PUF is to resist modeling attacks, the impact of the XOR on other key metrics must also be considered. In this section, we evaluate the impact of the XOR function on reliability, uniqueness, and uniformity.

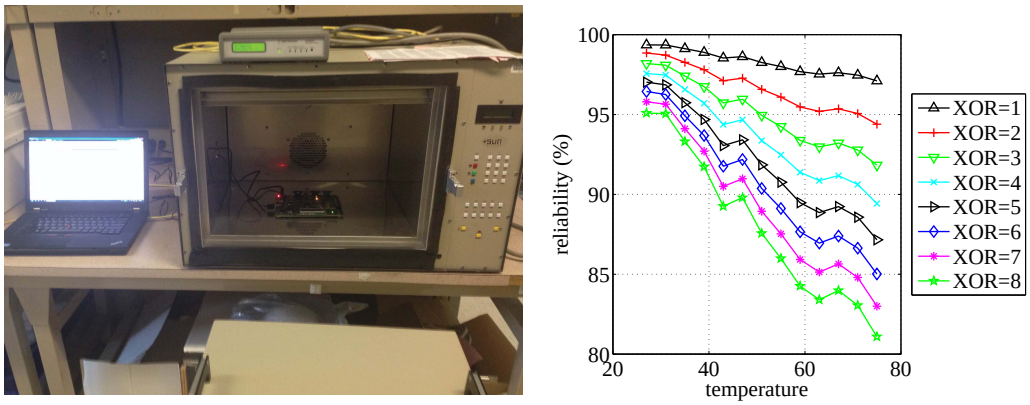
4.4.3.1 Reliability

Reliability is the ratio of consistent CRPs when a PUF is operating in different environment conditions such as temperature. To evaluate the reliability of XOR BR PUFs, 8 BR PUFs are measured across different temperatures between 27°C and 75°C, with a 4°C step, using a Sun Electronics EC12 Environmental Chamber [105] to control the temperature (Fig. 4.5a). Reliability is evaluated by comparing CRPs collected at 27°C to CRPs collected at other temperatures. For a XOR PUF, any unstable sub-response can cause the XORed response to be unreliable. Therefore, the reliability at any temperature will decrease with the number of PUFs that are XORed together (Fig. 6.7). According to the first BR PUF paper [15], an effective solution to solve this problem is employing CRPs that settle down quickly, since those CRPs are less sensitive to noise.

4.4.3.2 Uniqueness

Uniqueness is the capability of a PUF to distinguish itself from other instances. Uniqueness is quantified as the fraction of responses that are different across instances when the same challenges are applied. Thus for m PUF instances, a total of $\frac{m*(m-1)}{2}$

[1]The computer used has a common Intel 3630QM quadcore processor.



(a) experimental platform (b) reliability across different temperatures

Figure 4.5: Evaluating reliability across different temperatures. Because the reliability of each single BR PUF decreases with temperature, the reliability of the XOR BR PUF results degrade significantly.

uniqueness values are obtained. To better explore the uniqueness of XOR BR PUF, we compute its within-class (response flipping by noise, temperature noise here) and between-class uniqueness (response difference between instances), these results are depicted in Fig. 4.6.

4.4.3.3 Uniformity

Uniformity denotes the average response of a PUF, the ideal value of which is 0.5, meaning equal amount of "1" and "0" responses. Uniformity that is far away from 0.5 will have less response entropy and be easier to attack with modeling [123]. In our experiment, the uniformity of a single BR PUF is found to be highly biased, and this phenomenon was also reported in [98] [123]. The XOR function helps to remove this bias.

To validate the uniformity improvement from the XOR function, we collected the CRPs from eight 64-bit BR PUFs from FPGA (without CRP majority voting). It is found that some PUF instances show extreme bias, but XORing more single BR PUFs together decreases response bias (Fig. 4.7).

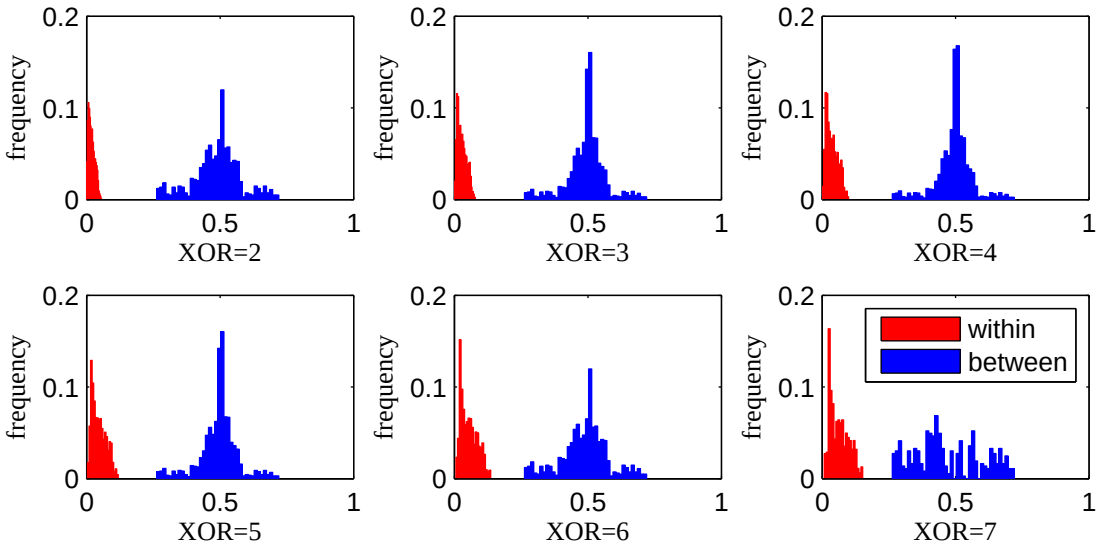


Figure 4.6: The between-class and within-class Hamming distance of XOR PUFs. Even when XORing together more BR PUFs, the within-class and between-class Hamming distances can still be differentiated. The results are based on 8 BR PUFs, thus there is only one 8 XOR BR PUF and no uniqueness is formulated for it.

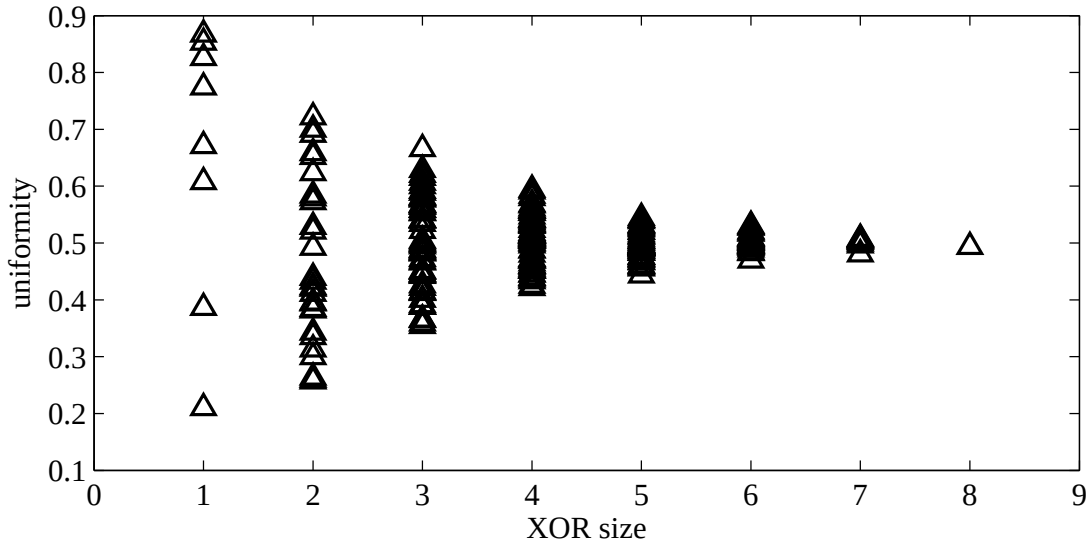


Figure 4.7: The response uniformity of a single BR PUF (represented by "XOR=1" in plot) is highly biased. When more BR PUFs are XORed together, the uniformity is closer to 0.5.

4.5 Summary

In this chapter, we studied two relatively new PUF variants: BR PUF and its derived architecture TBR PUF. Their resilience against ML modeling attacks is explored and it is shown that their response can be predicted with success rate exceeding 95% using reasonable runtime and less than 10k CRPs in all cases. Our work confirms that neither a single BR, nor TBR, PUF is secure. To strengthen the BR PUF against modeling attacks, we proposed and evaluated an XOR BR PUF variant. It is found that XORing 4 or more BR PUFs together produces a behavior that is beyond the modeling capability of current SVM ML attacks, and also improves other key PUF metrics like uniformity. Future work will explore the effectiveness of other modeling attacks, like Evolutionary Strategy and Logistic Regression methods.

CHAPTER 5

USING STATISTICAL MODELS TO IMPROVE THE RELIABILITY OF DELAY-BASED PUFS

In previous chapters, we present the effectiveness of using ML modeling to attack PUF primitives. This chapter addresses two causes of unreliability: transient unreliability caused by environmental noise, and persistent unreliability caused by device aging. To improve reliability, we constructively apply Machine Learning modeling, and use the models to predict and then discard challenge-response pairs (CRPs) that will be unreliable with respect to noise and aging on a given PUF instance. The proposed method provides flexibility to control error rate by deciding what percentage of challenges to discard. Our experiments find that a PUF with nominal reliability of 91% can be made fully reliable by discarding only 20% of challenges.

5.1 Introduction

Though many PUF architectures have been proposed, reliability is still a problem that is common to all PUFs. PUF responses are unreliable because supply voltage (V_{DD}) and temperature (T) variations can overcome the impacts of process variations and flip the outputs. Because a PUF operating in different conditions can generate a different response to the same challenge vector, the PUF output is therefore a function of not only the challenge and process variations, but also of transient environmental conditions.

It is desirable for a PUF to generate highly reliable responses over its lifetime. However, device aging is an important but rarely studied source of unreliability in

PUFs. Unlike environmental noise that temporarily flip PUFs' challenge-response pairs (CRPs) (PUF work more reliably when V_{DD} and T return back to normal), device aging causes a permanent change in the behavior of a PUF. Device aging is usually caused by negative bias temperature instability (NBTI), hot carrier injection (HCI), time dependent dielectric breakdown (TDDB), and electromigration [50] [62]. In this work, we focus on NBTI and HCI in particular. NBTI impacts the threshold voltage of pMOS transistors and decreases the speed of circuitry. Besides varying the threshold voltage of nMOS transistors, HCI also narrows the effective channel length of transistors. To build a comprehensive reliability model, aging must be considered in addition to environmental noise.

To make a PUF highly reliable, unstable CRPs that are easily flipped by environmental noise and aging should not be used. We propose a modeling technique to achieve this goal that requires no extra circuitry other than a normal PUF. Based on our method, a user applies a random set of challenges to a PUF, and obtains responses. From the CRPs, the user trains a Machine Learning (ML) model that can predict the reliability of responses to any challenge. He then uses the model to ensure that he discards unreliable challenges and only applies challenges for which the responses are highly reliable. In the experimental validation with 64−bit Arbiter PUFs (can also be applied to other PUFs), our technique can help to achieve greater than 99% reliability at the cost of only 0.21 seconds of computation time and 1500 training CRPs to build the model. Larger training sets lead to better modes and further improve reliability. Before this work, ML was always utilized to attack PUFs, so our work is novel in using ML modeling constructively to improve reliability. The contributions of this work are as follows:

- We classify the unreliability source of a PUF into two aspects: transient noise and aging; the reliability impact of both aspects are analyzed.

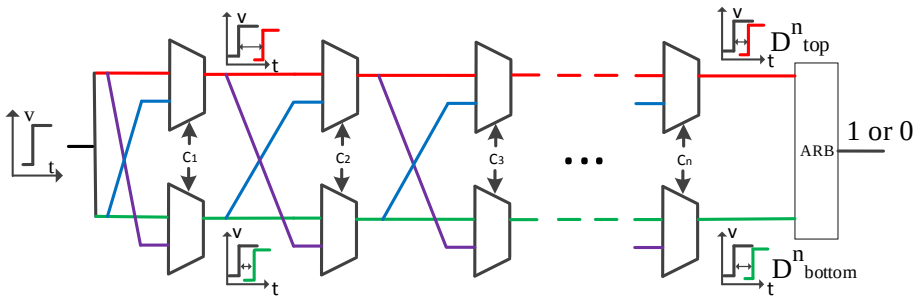


Figure 5.1: Schematic of an Arbiter PUF. Challenge **C** controls the propagation paths of rising edges that gather delay mismatch as they propagate toward the final arbiter.

- We train a ML model for PUF characterization and utilize the model for identifying and filtering out the unreliable challenge vectors for each PUF, allowing higher reliability to be achieved.

The rest of this chapter is organized as follows: Section 5.2 analyzes PUF unreliability caused by environmental noise and aging. Section 5.3 presents the use of ML modeling techniques to filter out unreliable CRPs. Section 5.4 presents experimental validation of the proposed approach. Section 5.5 presents directions for future work and concludes this chapter.

5.2 Analysis of Unreliable CRPs

A $n-$bit Arbiter PUF is composed of n stages, with each stage employing two 2:1 MUXs as depicted in Fig. 5.1. A challenge vector $\mathbf{C} = \{c_1, c_2, \ldots c_n\}$ is applied as the control signals for all stages to configure two paths through the PUF toward the arbiter; at each stage the paths are configured to be either straight or crossing. Thus, a rising edge applied at the input of the first stage gathers the delay mismatch from the paths through each stage while propagating toward the arbiter. The arbiter is a latch that digitizes the response into "1" or "0" by judging which rising edge is the first to arrive.

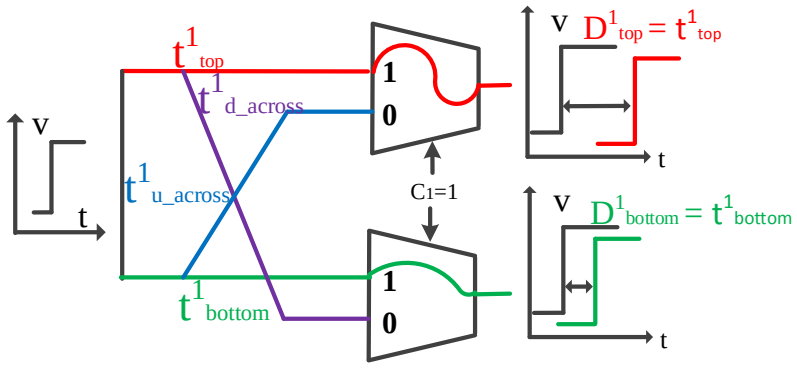


Figure 5.2: Propagation paths through the delay cells of an arbiter PUF.

5.2.1 Challenge and Response Pairs

A CRP can be considered reliable only if a PUF is capable of repeating this input/output relationship, even under different conditions (temperature, supply voltage, aging). Thus, before analyzing the unreliability of a PUF, it is meaningful to explore the relationship between its input and output. In an n-bit Arbiter PUF (Fig. 5.1), we define the propagation delay from the input to the first stage to the top and bottom outputs of the i^{th} stage as D^i_{top} and D^i_{bottom} respectively. The delay mismatch between two delay paths is summed up as the timing difference between D^n_{top} and D^n_{bottom} (Fig. 5.2). By mapping original challenge $c_i \in \{0, 1\}$ into $c_i \in \{-1, 1\}$ we get:

$$\begin{aligned} D^i_{top} &= \frac{1+c_i}{2}(t^i_{top} + D^{i-1}_{top}) \\ &+ \frac{1-c_i}{2}(t^i_{u\_across} + D^{i-1}_{bottom}) \\ D^i_{bottom} &= \frac{1+c_i}{2}(t^i_{bottom} + D^{i-1}_{bottom}) \\ &+ \frac{1-c_i}{2}(t^i_{d\_across} + D^{i-1}_{top}) \end{aligned} \tag{5.1}$$

where $D^0_{top} = D^0_{bottom} = 0$, and t^i_{top}, t^i_{bottom}, $t^i_{u\_across}$, $t^i_{d\_across}$ represent the four possible delays through the i^{th} stage. Denoting the delay difference between top and bottom arbiter inputs as DD, following Eq. 5.1, we get $DD = D^n_{top} - D^n_{bottom}$. The response ($r$) of an Arbiter PUF of n-bit length is therefore determined by the sign of DD (Eq.

5.2).

$$r = \begin{cases} 0, & \text{if } sgn(DD) > 0 \\ 1, & \text{if } sgn(DD) < 0 \end{cases} \tag{5.2}$$

From Eq.5.2, we can deduce that a PUF response is flipped when the sign of DD changes, from positive to negative or vice versa.

5.2.2 Environmental Noise and Aging

5.2.2.1 Environmental Noise

To explore the impact of noise in more detail, we perform simulations on a set of 64−bit Arbiter PUFs. In this simulation, in addition to measuring the arbiter output that depends on DD, we also sampled D_{top} and D_{bottom} to compute DD. Responses and DD values for $50k$ random challenges are collected under $1.1V$ and $28°C$ and used as a golden set. Furthermore, we repeated this collection under altered supply voltage and temperature points (to mimic practical situation, random noise is considered in all simulations). Two noise models: flicker and channel thermal noise are adopted in our simulation. Flicker noise is a frequency dependent noise caused by charge fluctuation in oxide traps, while channel thermal noise reflects voltage fluctuations due to the random motion of electrons. Parameters $fnoimod$ and $tnoimod$ in the BSIM4[1] are used to add flicker and thermal noise respectively;[2] noise is therefore modeled on all transistors in the circuit. The flipped responses are from challenges that, in the golden set, correspond to DD in a small range:

---

[1]BSIM4 model is based on the industry standard efforts of the Compact Modeling Counsel and the BSIM modeling group at UC Berkeley.

[2]the values assigned to the three parameters comprising $fnoimod$ are noia = 6.25e41, noib = 3.125e26, noic = 8.75; the values assigned to the parameters comprising $tnoimod$ are ntnoi = 1, tnoia = 1.5e6 and tnoib = 3.5e6

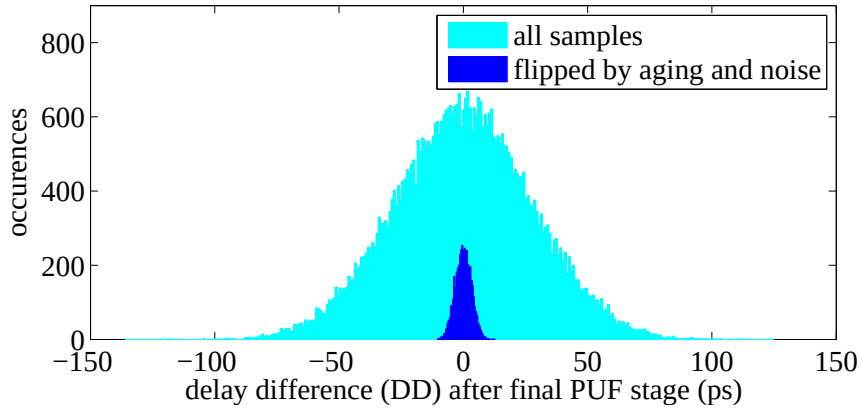


Figure 5.3: Exact delay difference DD of two sets: $50k$ golden samples (colored in cyan), the subset of samples flipped by aging and noise (colored in blue).

$$DD_{umin} \leq DD \leq DD_{umax} \tag{5.3}$$

For CRP data from Arbiter PUF, DD_{umax} therefore denotes the maximum DD among unreliable challenges, and DD_{umin} denotes the minimum DD among unreliable challenges. If only the challenge vectors that satisfy either $DD > DD_{umax}$ or $DD < DD_{umin}$ are applied to the PUF, then the PUF will be reliable and not prone to flipped response bits. For our simulated 64-bit PUFs based on $45nm$ Predictive Technology Model (PTM) [127], it is found that $DD_{umax} - DD_{umin} = 22.6ps$ when considering noise, voltage and temperature fluctuations (Fig. 5.3). However, in practice, it is impossible to directly filter out challenge vectors that induce large values of DD because *1)* DD cannot be measured directly, as the arbiter input signals are not externally observable; and *2)* even if it were possible to measure DD (e.g. using time-to-digital conversion [58]), it is infeasible to measure DD for all 2^n challenges as would be required for an n-stage PUF . As will be shown in Section 5.3, we overcome both of these limitations using a machine learning model to predict DD based on the responses produced at arbiter outputs.

5.2.2.2 Aging

In previous literature, aging is rarely considered as a source of unreliability. Unlike environmental variations, the impact of aging on transistors is permanent. The most common aging effects include NBTI and HCI, both of which can degrade the performance of a circuit. NBTI mainly changes the threshold voltage of transistors, thus causing decreased drive current and lower operating speed. HCI arises as a result of the aggressive scaling of device geometries, most notably for narrowing device channel length. Shorter channel length means higher circuit speed, but this also increases electric fields in the channel. As a result, these fields can damage the gate-oxide interface, resulting in degradation in device performance. The amount of degradation varies across devices and is a function of the switching activity of a device within a circuit. Thus, for a circuit that relies on physical process variations like a PUF, aging can introduce permanent unreliability.

To mimic the aging of Arbiter PUFs, we use Cadence Ultrasim as our infrastructure and simulate a group of Arbiter PUFs across different age spans: year, month, week, day, hour and minute. Using un-aged CRPs as golden values, we measure the reliability degradation due to aging, without the presence of noise or temperature and voltage fluctuations (Fig. 5.4). It can be seen that after only 1 hour of simulated aging, about 2% of responses become unreliable with respect to their golden CRPs. This occurs because some golden CRPs have associated DD values that are close to 0, and are thus susceptible to be flipped by even a small amount of aging. After a year of aging, approximately 8% of responses are unreliable compared with golden CRPs. The range of DD values that were susceptible to flipped responses within a year of aging fall within a range of $DD_{umax} - DD_{umin} = 25.4ps$. This indicates that, as with environmental noise, reliability due to aging can also be mitigated by applying to each chip only those challenges that induce a large DD.

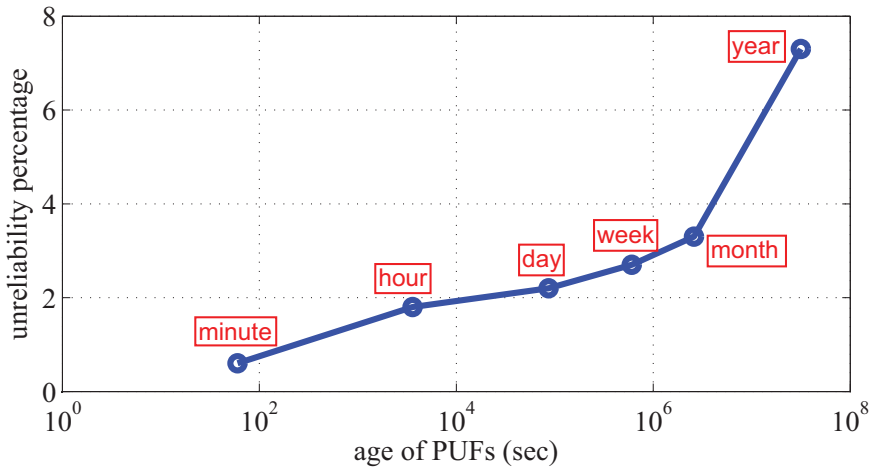


Figure 5.4: Aging introduces unreliability to PUFs, if denoting the CRPs of a new PUF as nominal database, older PUFs becomes more unreliable.

This section has explained that environmental noise and aging degrade PUF reliability and showed that all of the flipped responses are associated with DD closer to 0 in the golden data. The observation that unreliable challenges are associated with small DD values motivates an approach of improving PUF reliability by avoiding the challenges that induce small DD values on each PUF. The first step toward avoiding these challenges is to train a model that can predict DD of challenges.

5.3 Modeling DD of a PUF

Based on the analysis in Sec. 5.2, knowing the DD of each challenge vector makes it possible to get reliable CRPs by avoiding challenges with smaller DD. We treat these smaller-DD as likely to be unreliable and our goal is therefore to identify and avoid applying these challenges. However, as probing inside a PUF to measure DD is not practical, we instead predict DD using ML modeling method, that requires no silicon overhead other than a normal PUF circuit. To get the value of DD, we decompose an Arbiter PUF and mimic the generation process of its responses. Here we firstly define two parameters as:

$$\begin{aligned}\alpha_i &= (t^i_{top} - t^i_{bottom} + t^i_{d\_across} - t^i_{u\_across})/2 \\ \beta_i &= (t^i_{top} - t^i_{bottom} - t^i_{d\_across} + t^i_{u\_across})/2\end{aligned} \tag{5.4}$$

Based on Eq. 5.4, for a given challenge vector $\mathbf{C} = \{c_1, c_2, \ldots c_n\}$, corresponding response generation can be modeled by accumulating the delay mismatches through delay stages as:

$$DD = \alpha_1 k_0 + \cdots + (\alpha_n + \beta_{n-1})k_{n-1} + \beta_n k_n \tag{5.5}$$

where $k_n = 1$ and $k_i = \prod_{j=i+1}^{n} c_j$, reflecting the number of times that the rising edges will change tracks between the i^{th} stage and the arbiter. Thus, knowing the challenge and α_i and $\beta_i, i \in (1 \ldots n)$ makes it possible to compute DD for any challenge. By denoting the delay parameters of an Arbiter PUF with vector $\mathbf{p}_{model} = \{\alpha_1, \alpha_2 + \beta_1, \ldots \alpha_n + \beta_{n-1}, \beta_n\}$, and defining challenge features as $\mathbf{k} = \{k_0, k_1 \ldots k_n\}$, we can formulate the model-predicted DD of each challenge vector as:

$$DD = \langle \mathbf{p}_{model}, \mathbf{k} \rangle \tag{5.6}$$

Though we can not directly extract individual component delays of the Arbiter PUF, we can use ML modeling to predict suitable values for $\mathbf{p}_{model}$. To accomplish this, we use a set of known challenge response pairs to train a Support Vector Machine (SVM) classifier. SVM models are powerful learning tools that can perform binary classification of data, classification is achieved by a linear or nonlinear separating surface in the input space of the data set. SVMs have been widely used in modeling attacks on Arbiter PUFs [59, 90, **?**]. Note that accurate delay prediction is not an explicit objective of the SVM model, as the SVM model only seeks a value of $\mathbf{p}_{model}$ that will accurately predict responses. However, because responses are determined by the sign of DD (Eq. 5.2), a model that is trained to predict responses accurately is also

quite successful in predicting numeric values of DD. Therefore, our approach trains a binary classifier, and then uses the resulting model to predict the delay difference induced by each challenge. In turn, we use the predicted delay difference to infer whether or not a challenge will be reliable if applied to the PUF.

As known, if A is highly correlated with B, the correlation coefficient ($corr$) between them should be close to 1, and one can deduce information about B from A. The correlation can be quantified as:

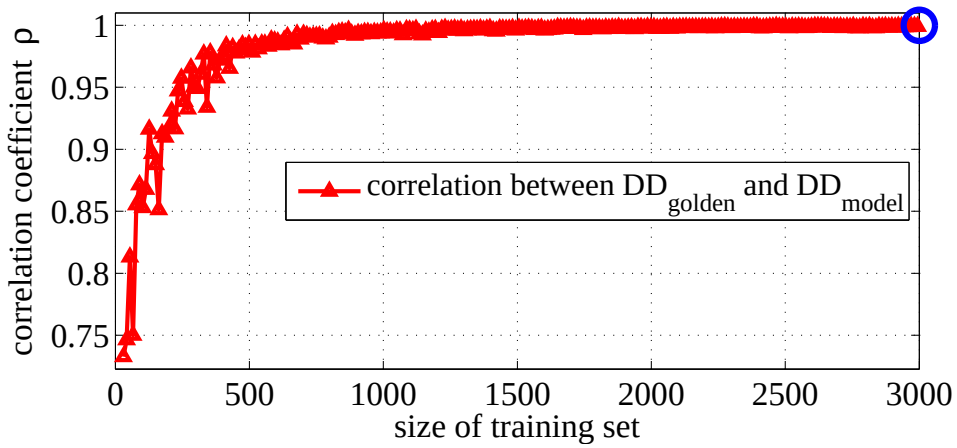
$$corr(A, B) = \frac{E[(A - E(A))(B - E(B))]}{\sigma A * \sigma B} \tag{5.7}$$

In our example, denoting the modeled PUF feature with $\mathbf{p}_{model}$, we hope that the model-predicted DD will be close to the golden value of DD that is measured from simulation. If this model works well for response prediction, then we expect ρ in Eq. 5.8 to be close to 1 when the CRP dataset used to train the model is sufficiently large.

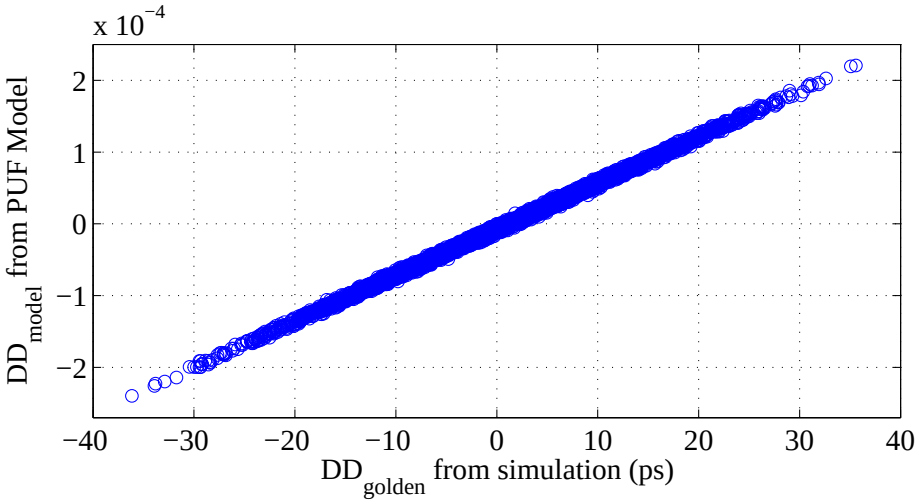
$$\begin{aligned} &corr(DD_{golden}, DD_{model}) = \\ &corr(DD_{golden}, \langle \mathbf{p}_{model}, \mathbf{k} \rangle) = \rho \end{aligned} \tag{5.8}$$

To validate the use of SVM classifier for delay prediction, a set of CRPs and corresponding DD_{golden} values are extracted from SPICE simulation. We collect 5000 CRPs and apply 60% of them as training set for PUF model building, while other 40% CRPs for model validation. We compute the correlation coefficient (ρ) between DD_{golden} (from simulation) and DD_{model}, which is calculated as $\langle \mathbf{p}_{model}, \mathbf{k} \rangle$. Fig 5.5a shows that ρ approaches 1 when applying many CRPs for model training. To present more details about this correlation, random challenges are chosen and DD_{golden} and DD_{model} (trained on 3000 CRPs) are shown for each one (Fig. 5.5b). Although DD_{golden} and DD_{model} are different in scale, the correlation between them is very high. Note that our approach for selecting reliable challenges does not need exact values but only need accurate relative magnitudes of delay differences to infer reliability.

Because the predicted DD values from the model are accurate with respect to the golden values from simulation, we can indeed filter out unreliable challenges of the PUF using DD_{model}.



(a) correlation coefficient ρ increases as training size



(b) good agreement between golden and model-predicted delay difference

Figure 5.5: In (5.5a), the correlation coefficient ρ between golden delay difference and model-predicted delay difference. While the PUF training size is increasing, higher ρ is achieved. In (5.5b), based on the model trained with 3000 CRPs, there is good agreement between DD_{golden} and DD_{model} for 2000 random challenges.

Procedure 8 Use ML model to compute the range of model-predicted delay differences that are likely to be unreliable for a given PUF. Challenges predicted to have delay differences inside this range will not be applied to the PUF, and this will improve the overall PUF reliability. Reliability can be improved by using a larger value of dr to discard more challenges.

Input: A discard ratio dr and a set of challenges and corresponding responses obtained from a single PUF at nominal supply voltage and temperature.

Output: A range $[DD_{min}, DD_{max}]$ of delay differences to consider as unreliable for this PUF.

1: Let $\mathbf{k}$ be the challenges mapped to challenge features (Eq. 5.5 and 5.6)
2: $\mathbf{p}_{model} \leftarrow \mathbf{SVM}(\mathbf{k}$, responses) {train the PUF model}
3: $\mu_p = \text{avg}\langle \mathbf{p}_{model}, \mathbf{k} \rangle$ {mean predicted delay difference}
4: $\sigma_p^2 = \text{var}\langle \mathbf{p}_{model}, \mathbf{k} \rangle$ {variance of predicted delay difference}
5: the distribution of delay differences across all challenges is modeled to be $\mathcal{N}(\mu_p, \sigma_p^2)$
6: DD_{min}=$F^{-1}(0.5 - dr/2) = \mu_p + \sigma_p \Phi^{-1}(0.5 - dr/2)$
7: DD_{max}=$F^{-1}(0.5 + dr/2) = \mu_p + \sigma_p \Phi^{-1}(0.5 + dr/2)$ {delay difference cutoffs based on PUF model $\mathbf{p}_{model}$ and selected challenge features $\mathbf{k}$ (Eq. 5.10)}
8: **return** $[DD_{min}, DD_{max}]$

5.4 Experimental Results

Following the methodology in Sec. 5.3, we use a PUF model to filter out challenge vectors that are potentially unreliable. In this section, we validate this method with noise from environment and aging. Based on the findings in Fig. 5.3, flipped responses are only these satisfying $DD_{umin} \leq DD_{golden} \leq DD_{umax}$. The main algorithm in our method uses an SVM classifier to predict delay cutoffs that will rule out approximately fraction dr of the overall challenge space. However, there are still two questions that need to be answered when applying this method:

1. How many CRPs are necessary to accurately model the DD_{umax} and DD_{umin} cutoffs each PUF?

2. What percentage of CRPs must be avoided to achieve a given level of response reliability?

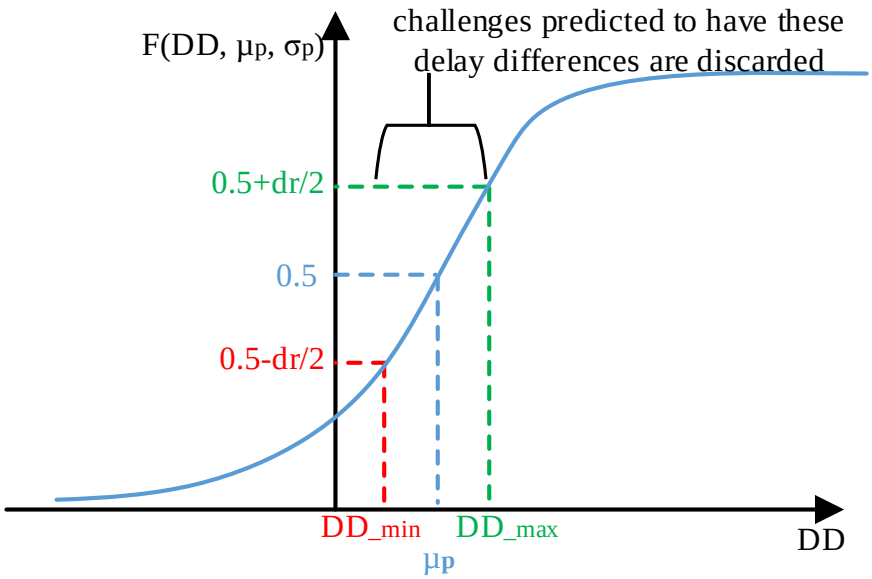


Figure 5.6: Illustration of the cumulative distribution function of delay difference DD. Based on a set of CRPs, we train a PUF model $\mathbf{p}_{model}$ and use it to compute μ_p and σ_p. Then given a discarded ratio dr, we discard the challenges for which the predicted DD is between DD_{min} and DD_{max}.

The first question is addressed using Alg. 8. For a PUF, assuming that its physical features follow Gaussian distribution, the corresponding DD distribution across randomly selected challenges will follow a normal distribution (Fig. 5.3). Our approach characterizes the distribution of delay difference (DD) by experimentally determining the mean and variance. For a set of delay difference values, we build its probability distribution function following Eq. 5.9. The randomness of training set ensures that it envelopes the unreliable range, μ_p and σ_p are defined in Alg. 8. An illustration about how this function works is shown in Fig. 5.6.

$$F(DD, \mu_p, \sigma_p) = \frac{1}{\sigma_p\sqrt{2\pi}} exp^{-\frac{(DD-\mu_p)^2}{2\sigma_p^2}} \tag{5.9}$$

As concluded in Sec. 5.2, the unreliable responses come from challenges generating smaller DD_{golden}, either negative or positive. Thus, even not knowing the exact unreliable range, we can still apply Quantile function to quantify it with Alg. 8. If

denoting $\Phi^{-1}(dr)(dr \in (0,1))$ as the Probit function of standard normal distribution, we determine the range of delay differences to discard using Eq. 5.10 (as steps 6 and 7 in Alg. 8), where dr stands for the ratio of challenges that should be discarded from the CRP database of each PUF. The value of dr provides a tradeoff in which selecting a larger value of dr to discard more challenges will result in more reliable PUF operation, but a smaller space of challenges that can be applied to the PUF.

$$F^{-1}(dr) = \mu_p + \sigma_p \Phi^{-1}(dr) \tag{5.10}$$

5.4.1 Validation under environmental noise

The qualified challenges ($DD_{model} \notin [DD_{min}, DD_{max}]$) are applied on PUFs, and their responses are compared with golden database, the result is shown as Fig. 5.7. With training size increasing, the values learned for DD_{min} and DD_{max} become more accurate, and fewer challenges need to be discarded to achieve the same reliability. The runtime to train the SVM model is modest; when using a training set of 4000 CRPs, training takes only 0.38 seconds.

Applications: As Machine Learning modeling is mostly considered in attacking PUFs, our main concern in this work is to show that this technique can also be applied for constructive purpose. Nevertheless, to use the technique in this work for reliability improvement, one should consider whether an adversary observing the reliable challenges that are applied to the PUF, would be able to use knowledge about which challenges are reliable to train a model that can predict the PUF response. To the best of our knowledge, this is not a question that has yet been addressed in the PUF community, and would be an interesting direction for future work.

5.5 Summary

In this chapter, we analyzed the causes of PUF unreliability. A machine-learning approach is utilized to model the delay difference on a given PUF. Based on the

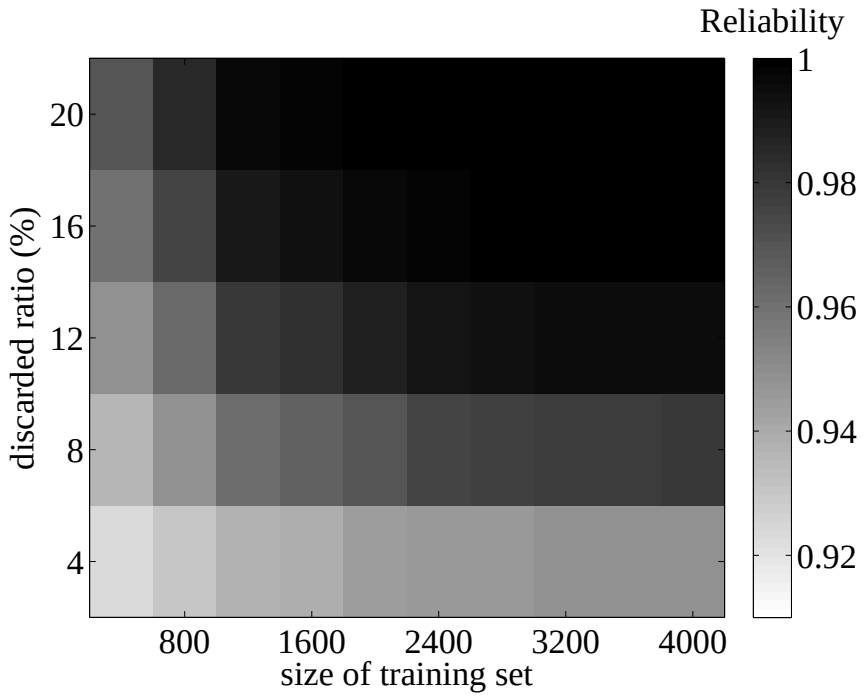


Figure 5.7: Validation under aging and environmental noise, across all of the simulated PUF instances. Trade-off between training size and discarded ratio can be seen in the figure. A larger *dr* is conservative and can compensate for the lower quality delay predictions of a model trained from a smaller training set.

prediction of delay difference from each challenge, we discard challenges that are likely to be unreliable. We validate our technique with a large data set, using both environmental noise and aging impacted simulations. Our method is demonstrated as an effective new technique for reliable PUF operation. Future work will explore the applicability to other PUF architectures.

CHAPTER 6

A CLOCKLESS SEQUENTIAL PUF WITH AUTONOMOUS MAJORITY VOTING

Techniques to ensure the reliability of PUF-based keys have associated costs in energy, area, and time. These costs must be minimized in order for PUFs to find practical application in resource-constrained scenarios. In this chapter, we propose and evaluate a new style of PUF that improves reliability by autonomously performing majority voting. The novelty of this design, and the source of its efficiency, is that the sequential majority voting happens using a self-timed circuit without orchestration by a global clock. We show that the proposed design can be instantiated to achieve different tradeoffs in energy versus bit-error-rate, and area versus latency.

6.1 Introduction

As we predicted in previous chapters, noise is an unwanted issue PUF outputs typically have a bit error rate of a few percent. Circuit level reliability techniques and error correcting codes are therefore used to produce reliable keys from the slightly-unreliable PUFs. In this chapter we propose a PUF that has the external interface of a simple weak PUF, but has an internal structure that autonomously performs majority voting to improve reliability. The specific contributions of this work as follows:

- We describe a novel clockless sequential PUF structure that performs autonomous majority voting.
- We evaluate using circuit simulation the reliability of the proposed PUF across a range of supply voltages and temperatures.

- We explore the tradeoffs of energy and reliability that can be accomplished using different parameterizations of the proposed PUF.

6.2 Related Work

Secret key style PUFs generate n-bit outputs using n-instantiations of logically identical cells, each of which generates a single output bit. The cells used for these PUFs resemble memory cells, and in many cases actually are repurposed memory cells. Such a cell has two stable states. The PUF output bit is generated by driving the cell to an unstable state and letting it transition to a stable state that is determined by process variation of the cell. Perhaps the best known PUF of this sort is based on the power-up state of SRAM cells [37, 31]; at power-up, an SRAM cell with both state nodes discharged becomes unstable and transitions to one of its two stable states. Other examples of cross-coupled PUF cells are ones based on resettable NOR latches [101], the butterfly PUF based on cross-coupled latches [53], and PUFs based on flip-flops [66] or sense amplifiers [9]. Yamamoto et al. [122] propose a PUF based on RS latches with inputs tied together, and this basic circuit is also a component of our proposed PUF. However, the manner in which the RS latch is used in our design differs substantially. Yamamoto's work is based on the location of RS latches with random output bits.

Secret keys used for cryptography must be highly repeatable over time, and given that PUFs are based on physical processes that are susceptible to noise, mechanisms are required to bridge the gap from unreliable circuits to reliable keys. These mechanisms can be broken into circuit-level techniques operating at the level of individual PUF bits, and error correction happening at the word level across many bits in aggregate. The two types of mechanisms can be used together, such that the circuit-level techniques reduce the bit-error-rate, and the reduced number of errors are then corrected using coding.

6.2.1 Reliability Enhancements in Secret Key PUFs

Bit-level reliability enhancements for PUFs decrease the error rate by reducing the impact of noise on the PUF outputs. Cells that are highly biased toward one output are largely immune to the small influence of noise, and bit errors happen in PUF cells that are approximately metastable. Directed accelerated aging techniques measure PUF outputs after manufacturing, and then reinforce cell tendencies using aging to increase the bias of each cell toward its more probable output [10]. Dark-bit masking [6] refers to pre-characterizing cells to identify and then mask out unreliable bits. Dark-bit masking can either be applied as a post-silicon testing step, or performed on-line in the field by evaluating the PUF under various stress conditions [72, 96].

6.2.2 Temporal Majority Voting

Especially relevant to the work in this chapter is temporal majority voting [6, 72]. In majority voting, a cell is evaluated n times in sequence, and the overall output is decided according to which value occurs in a majority of the n trials. Larger values of n improve the reliability of the PUF output. Effectively, majority voting across n trials counteracts the influence of noise that can cause a cell to produce its less likely value in a single trial. For example, consider a cell that has probability 0.1 of producing output value 0, and probability 0.9 of producing output value 1; this cell can be considered as having a error rate of 0.1 and nominal output value of 1. Using p to denote the error rate before majority voting, the error rate of the same bit after majority voting across n output observations is $p_{bit} = \sum_{i=\frac{n+1}{2}}^{n} \binom{n}{i} p^i (1-p)^{n-i}$. The relationship between p_{bit} and p for different values of n is shown in Fig. 6.1. For a bit as described above ($p = 0.1$), the error rate after majority voting is $p_{bit} = 0.028$ for $n = 3$, and $p_{bit} = 0.0027$ for $n = 7$.

Note that majority voting will not improve reliability of a cell that is equally likely to produce either output value ($p = 0.5$), or a cell for which the more likely output

changes according to temperature or another environmental parameter. Reliability of these cells must be addressed through one of the other reliability mechanism such as dark-bit masking, directed aging, or error correcting codes with helper data.

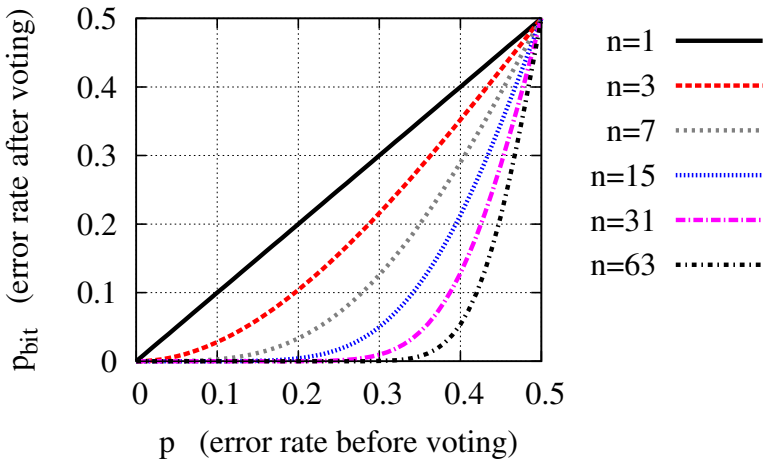


Figure 6.1: The impact of majority voting on reliability of a single bit. For any probability p of an output error in a single trial, p_{bit} is the corresponding probability of having an output error after using majority voting across n trials. Increasing n decreases the probability of error, except for cells with p equal to 0.5.

6.2.3 Error Correction using Helper Data

Reliability enhancements such as majority voting are complementary to the use of error correction. An example of error correction using the code-offset construction [47, 20] is shown in Fig. 6.2. An arbitrary secret key K is chosen for each PUF at enrollment and then encoded to a codeword $C(K)$ that is XORed with PUF response W to generate helper data. The unknowability of W prevents helper data $C(K) \oplus W$ from revealing secret $C(K)$, so helper data need not be protected against invasive attack and can be stored using any non-volatile memory, fuses, or antifuses. To regenerate the secret key in the field, a PUF produces a value W' that may differ slightly from W due to noise or environmental variations. Response W' is XORed with the helper data to produce a version of codeword $C(K)$ that is corrupted by the difference between W and W'. If the number of errors that can be corrected by the chosen code

exceeds the Hamming distance between W and W', then key K is correctly recovered by decoding the noisy codeword. Note that more reliable PUFs will have smaller Hamming distances between W and W', and this reduces the number of errors that must be correctable by the error correcting code.

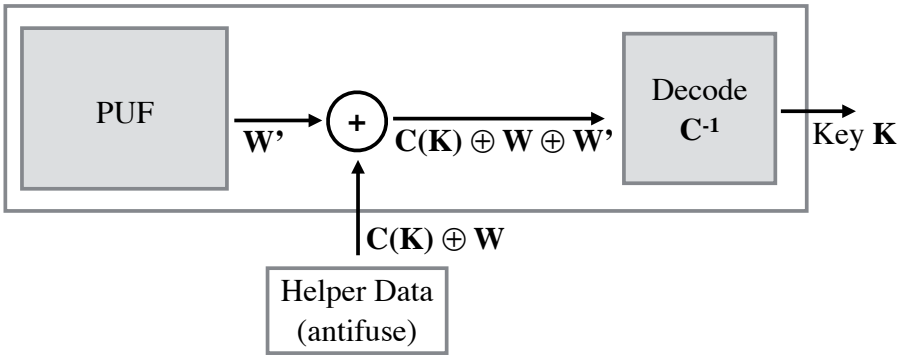


Figure 6.2: PUF-based secret key generation using helper data for error correction. The helper data is generated during a one-time enrollment process and is fixed over the life of the PUF.

6.3 Design of a Clockless Sequential PUF

In this work we propose a clockless sequential PUF that performs majority voting as part of its sequential behavior. Fig. 6.4 shows the proposed PUF design; each bit of PUF output corresponds one instantiation of this design. The design uses two LFSRs to count oscillations on circuit nodes, and the PUF output is determined according to which LFSR is first to reach its final count. Because each oscillation is effectively a PUF observation, producing an output based on oscillation counts constitutes autonomous voting without orchestration by a global clock. In other words, the sequential behavior of the PUF is unrelated to the system clock, and in some configurations can occur within a single system clock. The evaluation of the PUF is initiated by asserting EVAL. When the PUF has completed its sequential evolution, signal READY is asserted, and the output is read from OUT. Note that READY can

be neglected if sufficiently conservative time is waited between asserting EVAL and reading OUT.

6.3.1 Principle of Operation

Each PUF output bit is determined by the variation-induced behavior of an 8-transistor cross-coupled SR latch with its two inputs tied together. As shown in the shaded box in Fig. 6.4a, we denote the input X and the outputs Y1 and Y0. The SR latch is stable only when {X,Y1,Y0} takes the values {0,1,1}, {1,1,0} or {1,0,1}. If these signals have any other combination of values, the latch is unstable and its outputs will immediately transition to a stable latch configuration; for example {0,1,0} would transition to {0,1,1}.

Our design implements feedback around the SR latch to ensure that there are no stable states of the overall PUF circuit during evaluation, and furthermore that there are two possible modes of oscillation. During evaluation, the RUN signal input to the AND3 gate is high, so that X is a logical AND of delayed versions of Y1 and Y0. Because of this logical AND, each stable state {X,Y1,Y0} will lead after a propagation delay to a successor state {X',Y1,Y0}, such that X' = AND(Y1,Y0). These successor states are all unstable and will immediately transition to a stable state and then again propagate back to an unstable state, continuing in this way indefinitely. This constant evolution of state is shown in Fig. 6.3, where all unstable states are shown as unshaded. The variation-sensitive state transition of the PUF is the choice of stable state that follows unstable state {1,1,1}. The state of the circuit will never settle until RUN is deasserted. If the evolution of circuit state tends to follow the upper loop of Fig. 6.3, then node Y1, and OSC1 in Fig. 6.4, will tend to oscillate. If the evolution of circuit state follows the lower loop, then Y0 and OSC0 will oscillate.

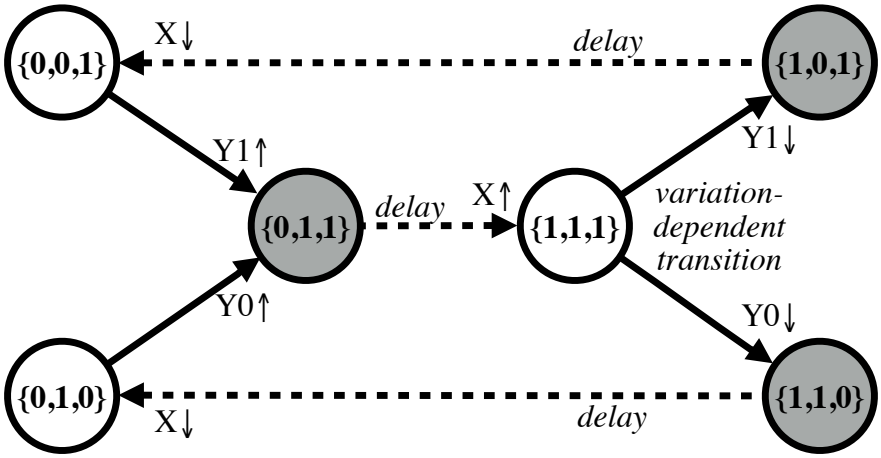


Figure 6.3: The evolution of SR latch state when embedded in the feedback circuit of Fig. 6.4a. The three values marking each state are assignments to {X,Y1,Y0}. The states that are shaded grey would be stable states if not for feedback. The unshaded states occur transiently when latch outputs propagate back to change the value of X. The successor state to {1,1,1} is determined by process variation and, to a lesser extent, by noise.

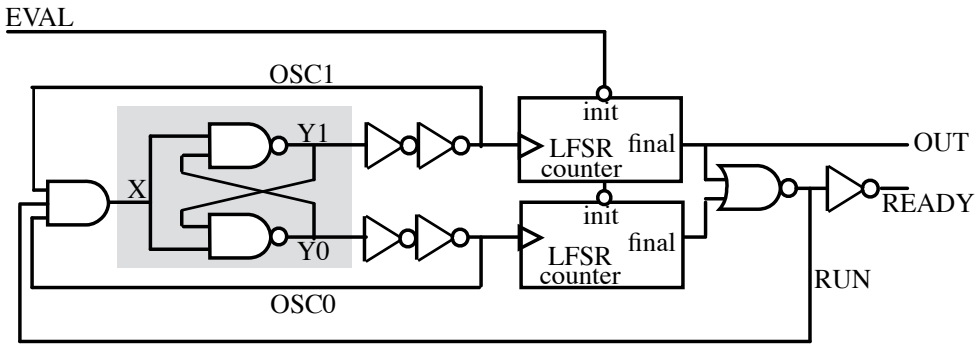
6.3.2 Linear Feedback Shift Register as Counter

To meet strict timing constraints, An LFSR instead of a binary counter is used to count oscillations. In any digital circuit, the delay of the critical path must be less than the clock period, so that the combinational logic will stabilize to its final value before the next clock edge comes. If this condition is not met, then a timing violation is said to occur, and the value that gets stored on the clock edge is unpredictable. In our circuit (Fig. 6.4a), the clock signal to each counter is in reality an oscillating signal from the PUF. The time between successive rising edges on the LFSR clock input is the oscillation period of PUF node OSC1 or OSC0, and is therefore determined by the time for a signal to propagate twice through an AND3 gate, a NAND2 gate, and two inverters. The oscillation periods of these nodes are shorter than the delay of carry chains in adder circuits, so adders cannot be used to count oscillations without causing timing violations. We resolve this concern by using an LFSR, which has a much shorter critical path delay, for the counter. Because the LFSR critical path delay

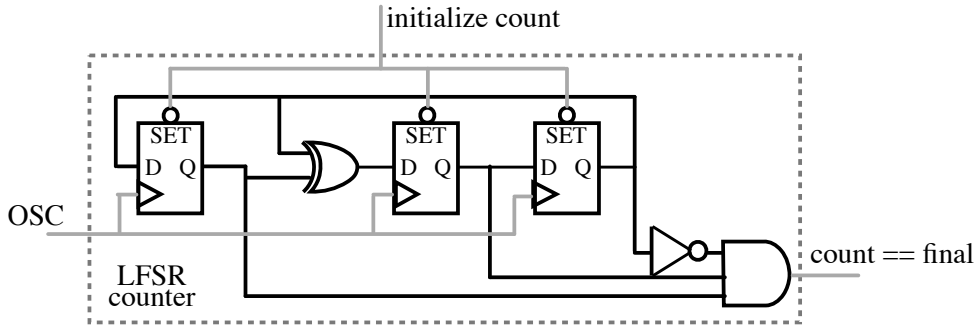
is shorter than the oscillations of the PUF nodes (Fig. 6.8), it can be clocked using the oscillating PUF nodes without causing timing violations. An LFSR has previously been used as a round counter in the the KATAN block cipher by DeCanniere et al. [16]; as in our case here, DeCanniere chooses to use an LFSR for its short critical path, small implementation, and low power. An alternative solution to using LFSR counters is to lengthen the PUF oscillation period by increasing the propagation delay of the feedback structure, but this is avoided because it would incur area and power costs.

The size of the LFSR, in number of bits, is a parameter of the PUF that can be chosen at design time. In Fig. 6.4b we show a 3-bit counter as one particular choice of LFSR size. For any given LFSR size n, a maximum length polynomial must be chosen so that the LFSR will have $2^n - 1$ non-repeating states, and therefore $2^n - 2$ transitions between the initial state and the final state. The implications of LFSR size are discussed in the evaluation sections.

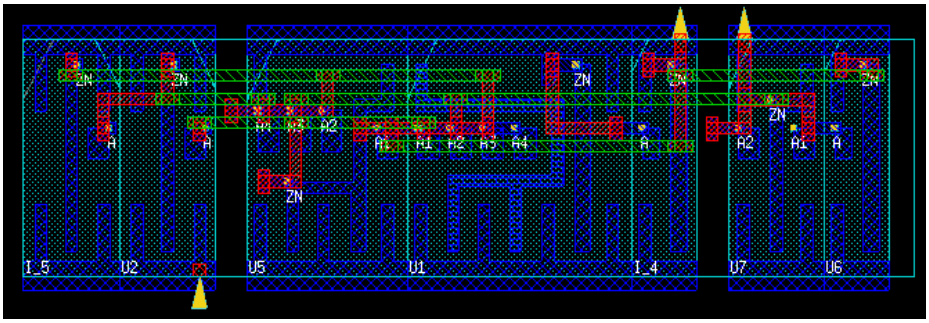
The proposed PUF architecture can be built from an industry gate library, and the synthesized PUF with 3-bit LFSR counters is shown in Fig. 6.4c. The area is 43 μm^2 in a 45nm technology node. The schematics of Fig. 6.4 imply that each PUF has two dedicated LFSR counters, and in this case all PUFs can be evaluated in parallel. It is also possible to design the PUF so that LFSR counters are shared across multiple PUF bits; PUF bits that share counters would be evaluated serially. Sharing of counters or other support circuits would reduce area cost but increase latency due to serialization. In this work, we assume the fully parallel solution, where each PUF has its own dedicated LFSR counters. Note that the area overhead of counters is not unique to our design and in fact any majority voting scheme must similarly have counters.



(a) Overall PUF circuit schematic including two LFSR counters



(b) Detailed schematic view of 3-bit LFSR counter



(c) Layout view of PUF with 3-bit LFSR counters (M1-M4 layers shown)

Figure 6.4: Schematic and layout view of the proposed PUF. The SR latch and feedback circuit together induce an oscillating behavior that favors OSC1 or OSC0. An LFSR advances its state at each rising edge of OSC1 or OSC0 to count the oscillations of that signal. When one LFSR reaches its final count value, RUN is deasserted, and the output is ready to read out.

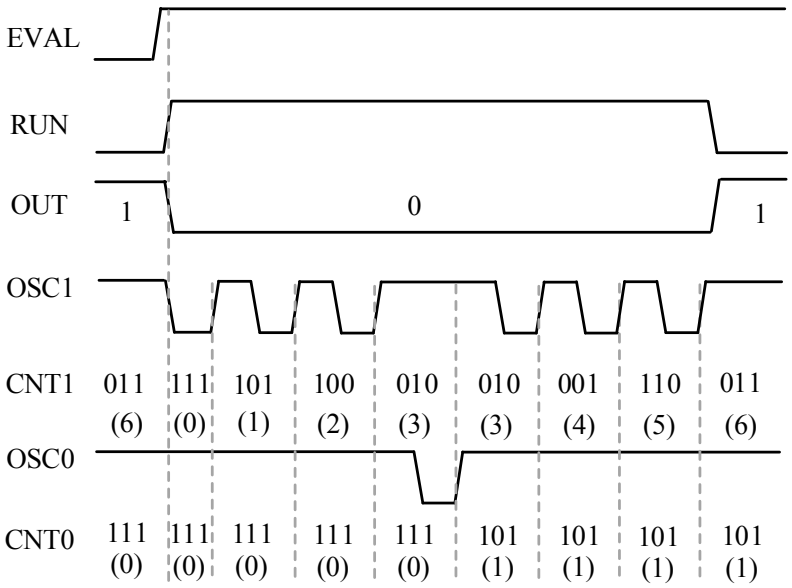


Figure 6.5: The transient waveforms of a single execution of the proposed PUF. The EVAL signal starts the evaluation by initializing the LFSR counter states (see Fig. 6.4), and everything thereafter happens autonomously. Once a counter (counter 1 in this case) reaches its sixth and final state, the oscillation is terminated, and the output is ready. In this example the PUF output is 1 because counter 1 reached its final state.

6.4 Experimental Validation

We evaluate the proposed PUF design by simulating the circuit at different supply voltages and temperatures.

6.4.1 Simulation Methodology

The results in this chapter are obtained from circuit simulation using Synopsys HSPICE, version E-2010.

6.4.1.1 Transistor Models and Sizing

Transistor and interconnect models are from the freely-available Predictive Technology Model (PTM). More specifically, the transistor models are PTM models for a 45 nm process [13]. The circuit behavior is sensitive to the transistors in the inverters and the NAND2 gates in the SR latch, so the relevant parameters for these transistors

are shown in Tab. 6.1. The mean value μ for each variation-impacted parameter is the nominal value in the transistor model. The standard deviation (σ) of threshold voltage ($vth0$) depends on transistor size, and is calculated from Eq. 6.1 using a value of 1.8 $mV\ \mu m$ for A_{VT} [80]. The 3σ of transistor length is set to be 10% of the overall transistor length based on existing literature [4], and this is implemented using transistor parameter lint[1] as shown in Tab. 6.1.

$$\sigma_{VT} = \frac{A_{VT}}{\sqrt{WL}} \tag{6.1}$$

		Sizing		Process Variation			
				vth0 [mV]		lint [nm]	
		W [nm]	L [nm]	μ	σ	μ	σ
NAND2	NMOS	180	50	468	19.0	5	1.7
	PMOS	180	50	-491	19.0	5	1.7
INV	NMOS	90	50	468	26.8	5	1.7
	PMOS	180	50	-491	19.0	5	1.7

Table 6.1: Transistor sizes and process variation parameters.

6.4.1.2 Noise

To evaluate the ability of majority voting to mitigate noise, we adopt two noise models in our simulation: flicker noise and channel thermal noise. Flicker noise is a frequency dependent noise caused by charge fluctuation in oxide traps, and results in fluctuations of both mobile carrier numbers and mobilities in the channel. Channel thermal noise reflects voltage fluctuations due to the random motion of electrons. The BSIM4[2] parameters $fnoimod$ and $tnoimod$ implement flicker and thermal noise

[1] lint, standing for internal length, represents the difference between nominal and effective transistor length

[2] BSIM4 model is based on the industry standard efforts of the Compact Modeling Counsel and the BSIM modeling group at UC Berkeley.

respectively;[3] noise is therefore modeled on all transistors in the circuit. The simulation results of Fig. 6.6 obtained using the described variation and noise models shows the clear separation of between-class and within-class Hamming distances. The large between-class Hamming distance implies high uniqueness, and the relatively small within-class Hamming distance implies high reliability.

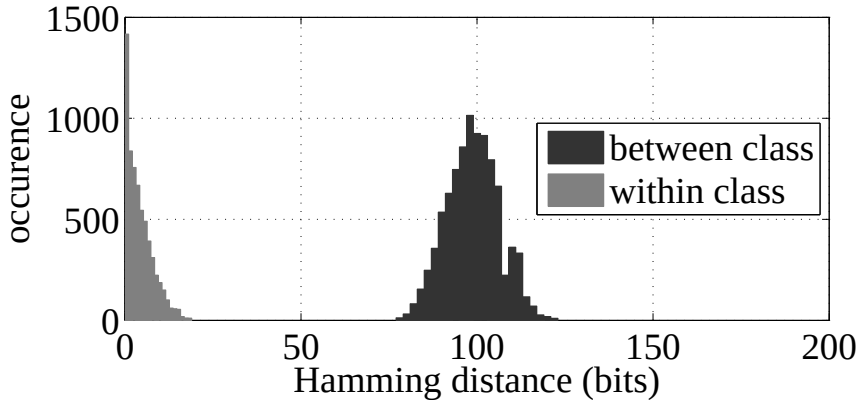
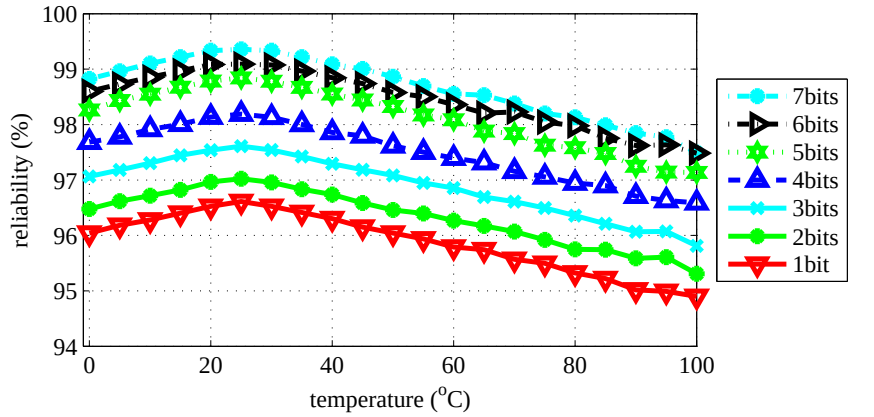


Figure 6.6: Hamming distances of the proposed PUF with 7-bit LFSR counters. Within-class data points are two outputs from the same PUF, and between class are from different PUFs. In addition to noise, the temperature is assigned randomly from the range of 0°C to 100°C in each trial for this plot.

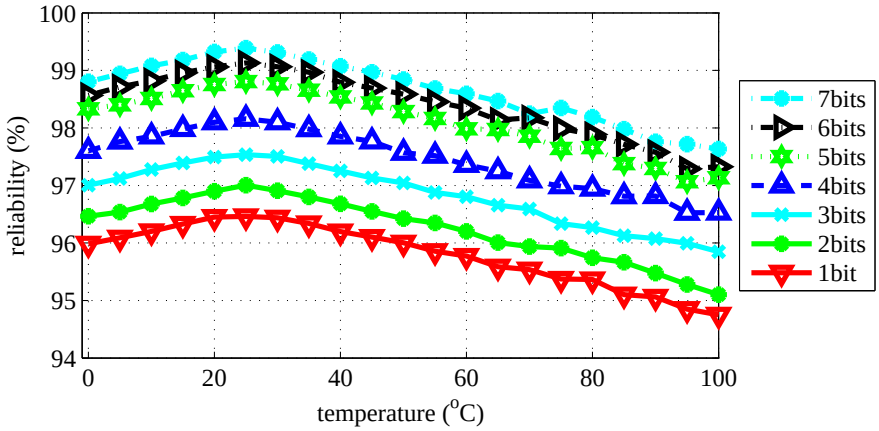
6.4.2 Reliability

The reliability of the proposed PUF is evaluated at the nominal supply voltage of 1.1 V, and at voltages of 0.825 V and 0.55 V (Fig. 6.7). On each plot of Fig. 6.7, each line shows reliability across temperature of PUFs with a given size LFSR; correctness of the PUF output is determined by comparing each output value against an output from the same PUF at 25°C. Increasing the size of the LFSR improves the reliability of the PUF because it increases the number of majority voting trials that occur internally before an output is produced when one of the LFSR counters reaches its final value.

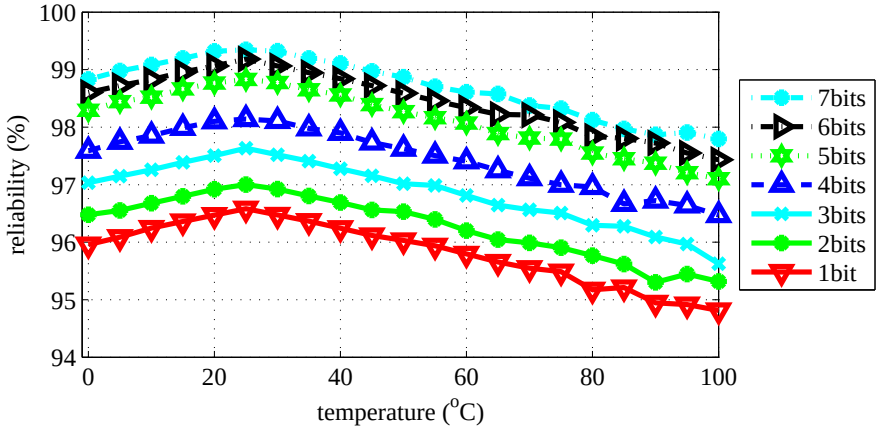
[3]the values assigned to the three parameters comprising *fnoimod* are noia = 6.25e41, noib = 3.125e26, noic = 8.75; the values assigned to the parameters comprising *tnoimod* are ntnoi = 1, tnoia = 1.5e6 and tnoib = 3.5e6



(a) Nominal Vdd of 1.1 V



(b) Reduced Vdd of 0.825 V



(c) Reduced Vdd of 0.55 V

Figure 6.7: Reliability evaluation of the proposed PUF across temperatures for various widths of LFSR counter. The results are averaged based on 1,000,000 PUF instances. For the result labeled as a 1-bit counter, no majority voting is performed.

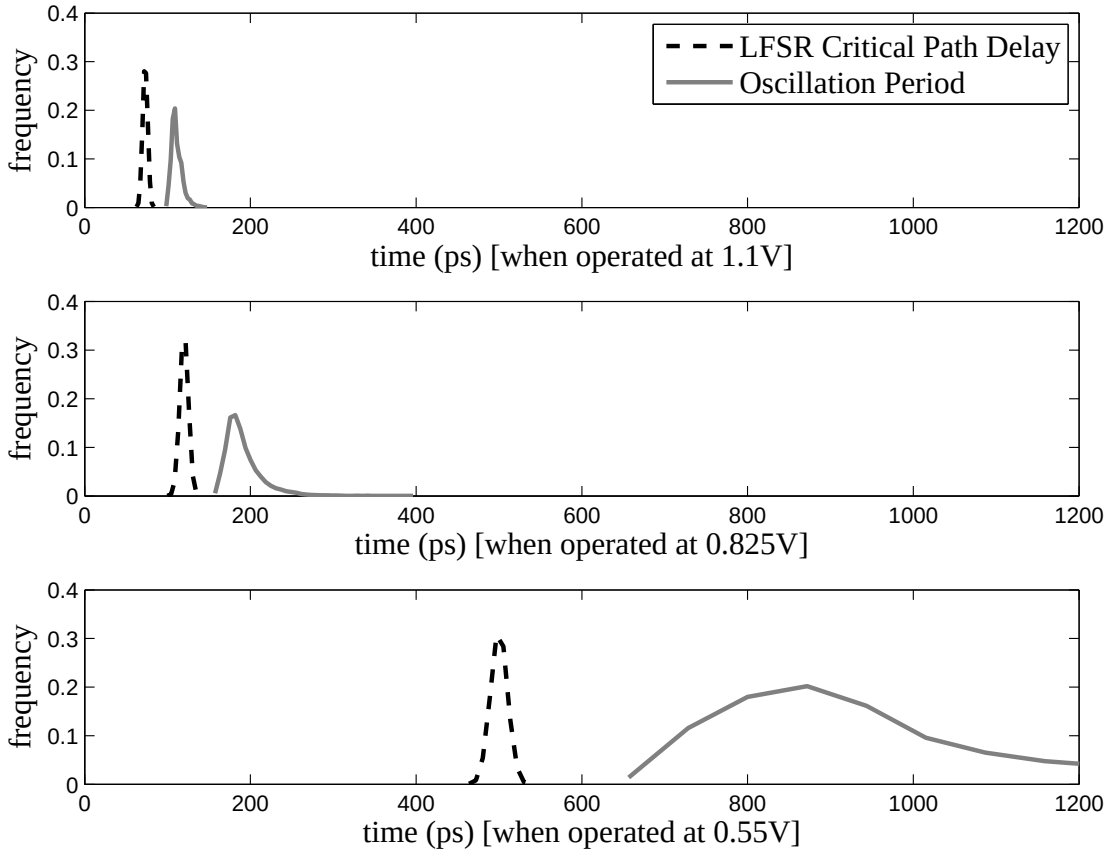


Figure 6.8: The distribution of LFSR critical path delays and PUF oscillation periods at three different supply voltages. Because LFSR critical path delay is shorter that the oscillation period even with process variation, the LFSR circuit can use the PUF oscillations as a clock without having timing violations.

6.4.3 Error Correction

Error correcting codes are used to correct PUF output bit errors to produce reliable secret keys (see Sec. 6.2.3). The number of PUF output bits needed to generate a key is significantly larger than the key size, and this number increases when more errors must be correctable. For example, Guajardo et al. [32] show that reliably generating a 228-bit key requires 6,128 PUF output bits when conservatively assuming a bit-error-rate of 0.16. Our use of majority voting in the proposed PUF reduces bit-error-rate and reduces the number of PUF output bits used to generate a key. There are at least three benefits to reducing the required number of PUF output bits: *1*) the area cost of PUF circuitry is proportional to its number of output bits; *2*) the amount of helper data (see Fig. 6.2) is also proportional to its number of output bits; and *3*) the area and energy cost of the error correction algorithm increases with the number of bits used.

In this section, we quantify how the reliability improvement of the proposed PUF reduces the burden of error correction. We focus on binary BCH codes, which are a set of codes that can be adapted for their block size and for the number of correctable errors in each block. As the number of correctable errors per block increases, the number of useful bits in each block decreases.

For a PUF with bit error rate p_{bit}, we search for the BCH code that uses the minimum number of PUF output bits to generate a 2,048-bit key that will be incorrect fewer than once per billion uses. We find an optimal code for different values of p_{bit} by using exhaustive search of all BCH codes that have block length less than 511 bits. For a BCH code described by parameters b (block size in bits), k (useful key bits per block), and t (correctable errors per block), the following procedure calculates the probability (denoted p_{key}) of producing an incorrect 2,048-bit key. First, the probability (denoted p_{block}) of having an uncorrectable error in a single block is calculated as the probability of having more than t erroneous bits in the block (Eq. 6.2), under

the assumption that bit errors are independent and identically distributed. Then, given that $n_{blocks} = \lceil 2,048/k \rceil$ blocks are needed to generate the entire key, the key error probability p_{key} is calculated as the probability that not all blocks are error-free (Eq. 6.3). If the p_{key} of a particular BCH code is less than 1E-9, then the BCH code is deemed suitably reliable for correcting a bit error rate of p_{bit}, and will be considered optimal if the number of bits used ($b * n_{blocks}$) is fewest among all BCH codes that satisfy the same condition of p_{key} being less than 1E-9.

$$p_{block} = \sum_{i=t+1}^{b} \binom{b}{i} p_{bit}^{i} \left(1 - p_{bit}\right)^{b-i} \tag{6.2}$$

$$p_{key} = 1 - \left(1 - p_{block}\right)^{n_{blocks}} \tag{6.3}$$

Based on the analysis described in the preceding paragraph, Table 6.2 shows the optimal BCH code for the p_{bit} values from different LFSR counter sizes. For each LFSR size, the value of p_{bit} is the bit error rate obtained in the worst-case scenario of 100°C and 0.55 V (see Fig. 6.7c). The improved reliability that comes from increasing LFSR size has a large impact on the number of PUF bits needed to generate the key.

Number of LFSR bits	p_{bit}	Optimal BCH code b	k	t	PUF bits required	Percent of bits saved
1	0.0519	255	47	42	11,220	-
2	0.0468	511	103	61	10,220	8.9%
3	0.0437	511	121	58	8,687	22.6%
4	0.0353	511	157	51	7,154	36.2%
5	0.0290	511	184	45	6,132	45.3%
6	0.0257	511	211	41	5,110	54.5%
7	0.0221	511	229	38	4,599	59.0%

Table 6.2: BCH codes that generate a 2,048-bit key using the minimal number of PUF outputs for different values of p_{bit}. The BCH codes are chosen such that the key will be generated incorrectly less than once per billion uses. The value of p_{bit} in each row corresponds to the observed bit-error-rate at 100°C at 0.55 V supply. Each BCH code block uses b bits to encode k key bits with capability to correct up to t errors.

6.4.4 Energy

Changing the LFSR counter size trades energy-per-bit against reliability. For an n-bit LFSR, a maximal length polynomial is chosen so that the evaluation will end after one counter has experienced $2^n - 2$ rising transitions on its oscillating input. When considering both counters, the total number of oscillations in the evaluation is between $2^n - 2$ and $2(2^n - 2) - 1$. The energy of a PUF evaluation is roughly linear in the number of oscillations and therefore exponential in LFSR size n (Fig. 6.9). Once the LFSR becomes sufficiently large, the small reliability gains from additional majority voting trials come at a high energy cost. In addition to changing the LFSR size at design time, another strategy to reduce energy-per-bit is lowering the operating voltage of the PUF. Note that both increasing LFSR size and lowering supply voltage increase the time required to complete each evaluation of the PUF.

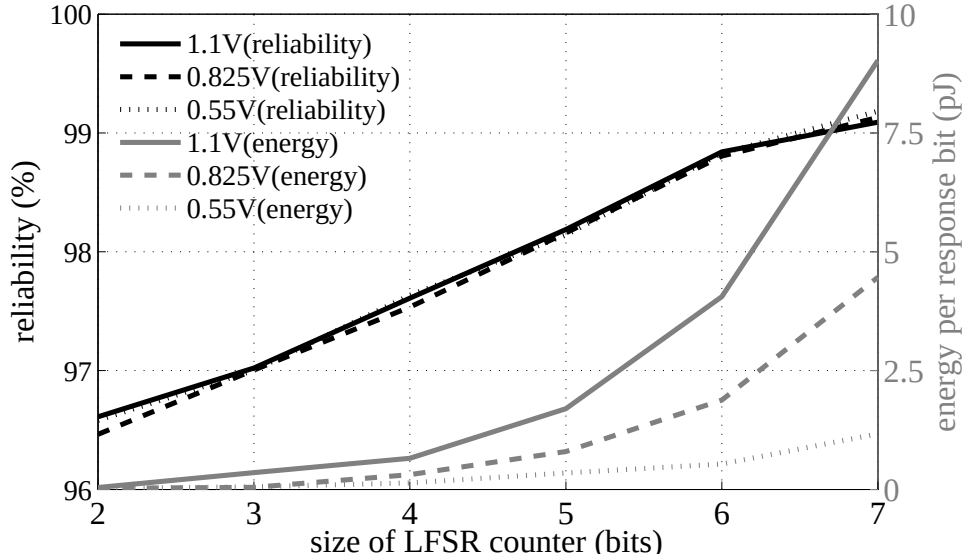


Figure 6.9: The energy-per-response-bit increases roughly linearly with the number of oscillations, which is exponential in the number of LFSR counter bits. Meanwhile, the reliability increases with the size of the LFSR counter due to the increased number of majority voting trials performed.

6.5 Summary

This chapter proposes a novel PUF that uses autonomous majority voting without orchestration by a clock. The proposed design is parameterizable by its LFSR counter size, and this allows reliability to be traded off against energy and area. Our experiments show that the improved reliability of the design translates to as much as 59% savings in the number PUF bits needed for key generation when combined with error correcting codes. An interesting direction for future work is to minimize total energy of PUF key generation when using this design together with both dark bit masking and error correction.

CHAPTER 7

CONCLUSIONS

Driven by Moore's law, semiconductor industry has experienced continuous physical scaling in the past few decades. This advancement greatly facilitates the development of electronic devices like smart phones, tablets, but also proposes new challenges for hardware security researchers. This dissertation presents some of our recent work in advancing secure CMOS computation with intrinsic functions: i.e., the so-called Physical Unclonable Functions (PUFs). Three topics are studied: nanometer-scale process variations, Machine Learning based modeling and noise sensitivity.

Chapter 1 of this dissertation lists the challenge and opportunities in nanometer cryptographic circuit design. We predict that though process variations are detrimental to conventional circuit design, PUFs have been proposed as a viable solution to harness the unpredictable nature of process variations for security applications. Terminologies like *challenge-response pairs (CRPs)*, *reliability*, *uniqueness* and *uniformity* are defined in this chapter.

Since PUFs leverage microscopic process variations, thus are sensitive to environmental noise like slight temperature or supply voltage fluctuations. Due to such sensitivity, a PUF may not produce consistent response for the same challenge under different environmental conditions. To address the reliability of currently proposed PUF primitives, two highly reliable PUF mechanisms on two common seen memory architectures, SRAM and D Flip-flop are proposed in Chapter 2 and 3. To improve the efficiency of our propose PUF mechanisms, we combined algorithm like binary

search, advanced Machine Learning method like Artificial Neural Network (ANN) in these two works.

Bistable Ring (BR) PUF is a newly proposed but promising hardware security primitive in PUF family. In Chapter 4, we studied the ML modeling resilience of BR PUF and its derived architecture TBR PUF. We evaluate the security of Bistable Ring PUF by applying different attacking models and methods, besides the proposed attacks in other literatures, we propose a novel model based on Support Vector Machine (SVM). XORing method is employed to enhance the attack resilience of single BR PUF, we demonstrate that XORed BR PUF is beyond the reach of current known attacking models, thus is secure.

Environmental noise including temperature variance and supply voltage fluctuation is a known cause for PUFs reliability problem. However, device aging as a widely studied issue in hardware deign, was rarely considered in previous literatures on PUFs. Chapter 5 addresses these two causes of unreliability: transient unreliability caused by environmental noise, and persistent unreliability caused by device aging. We constructively apply ML modeling method, and use the models to predict and then discard CRPs that will be unreliable with respect to noise and aging on a given PUF instance.

In Chapter 6, we propose and evaluate a new style of PUF that improves reliability by autonomously performing majority voting. The novelty of this design, and the source of its efficiency, is that the sequential majority voting happens using a self-timed circuit without orchestration by a global clock. We show that the proposed design can be instantiated to achieve different tradeoffs in energy versus bit-error-rate, and area versus latency.

BIBLIOGRAPHY

[1] International technology roadmap for semiconductors (itrs) – front end process, 2004 update.

[2] Ahmed, Khaled, and Schuegraf, Klaus. Transistor wars. *Spectrum, IEEE 48*, 11 (2011), 50–66.

[3] Alliance Memory Inc. 8K X 8 Bit Low Power CMOS SRAM.

[4] Anis, M, and Aburahma, M H. Leakage current variability in nanometer technologies. In *System-on-Chip for Real-Time Applications, 2005. Proceedings. Fifth International Workshop on* (2005), pp. 60–63.

[5] Anis, Mohab, and Aburahma, Mohamed H. Leakage current variability in nanometer technologies. In *System-on-Chip for Real-Time Applications, 2005. Proceedings. Fifth International Workshop on* (2005), IEEE, pp. 60–63.

[6] Armknecht, Frederik, Maes, Roel, Sadeghi, Ahmad-Reza, Sunar, Berk, and Tuyls, Pim. Memory leakage-resilient encryption based on physically unclonable functions. In *Towards Hardware-Intrinsic Security*, Ahmad-Reza Sadeghi and David Naccache, Eds., Information Security and Cryptography. Springer Berlin Heidelberg, 2010, pp. 135–164.

[7] Asenov, Asen. Random dopant induced threshold voltage lowering and fluctuations in sub-0.1 μm mosfet's: A 3-d atomistic simulation study. *Electron Devices, IEEE Transactions on 45*, 12 (1998), 2505–2513.

[8] Bacha, Anys, and Teodorescu, Radu. Authenticache: Harnessing cache ecc for system authentication.

[9] Bhargava, M, Cakir, C, and Mai, K. Attack Resistant Sense Amplifier Based PUFs (SA-PUF) with Deterministic and Controllable Reliability of PUF Responses. *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on* (2010).

[10] Bhargava, Mudit, Cakir, Cagla, and Mai, Ken. Reliability enhancement of bi-stable PUFs in 65nm bulk CMOS. *International Symposium on Hardware-Oriented Security and Trust* (2012).

[11] Brzuska, Christina, Fischlin, Marc, Schröder, Heike, and Katzenbeisser, Stefan. Physically uncloneable functions in the universal composition framework. In *Advances in Cryptology–CRYPTO 2011*. Springer, 2011, pp. 51–70.

[12] Cabe, Adam C, Qi, Zhenyu, and Stan, Mircea R. Stacking SRAM Banks for Ultra Low Power Standby Mode Operation. In *Design Automation Conference* (June 2010).

[13] Cao, YU, Sato, T, Sylvester, D, Orshansky, M, and Hu, C. Predictive technology model.

[14] Chen, Qingqing, Csaba, Gyorgy, Lugli, Paolo, Schlichtmann, Ulf, and Ruhrmair, U. The bistable ring puf: A new architecture for strong physical unclonable functions. In *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on* (2011), IEEE, pp. 134–141.

[15] Chen, Qingqing, Csaba, Gyorgy, Lugli, Paolo, Schlichtmann, Ulf, and Ruhrmair, U. Characterization of the bistable ring puf. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012* (2012), IEEE, pp. 1459–1462.

[16] De Canniere, Christophe, Dunkelman, Orr, and Knežević, Miroslav. KATAN and KTANTAN: a family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems-CHES 2009*. Springer, 2009, pp. 272–288.

[17] Delvaux, J, and Verbauwhede, I. Key-recovery attacks on various RO PUF constructions via helper data manipulation. In *Proceedings of the Conference on Design, Automation and Test in Europe* (2014), European Design and Automation Association.

[18] Delvaux, Jeroen, Gu, Dawu, Schellekens, Dries, and Verbauwhede, Ingrid. Helper data algorithms for puf-based key generation: Overview and analysis.

[19] Devadas, S, and Yu, Meng-Day. Secure and robust error correction for physical unclonable functions. *Design Test, IEEE PP*, 99 (2015), 1–1.

[20] Dodis, Yevgeniy, Ostrovsky, Rafail, Reyzin, Leonid, and Smith, Adam. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing 38*, 1 (2008), 97–139.

[21] Dodis, Yevgeniy, Reyzin, Leonid, and Smith, Adam. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in cryptology-Eurocrypt 2004* (2004), Springer, pp. 523–540.

[22] Dolecek, Lara, Qazi, Masood, Shah, Devavrat, and Chandrakasan, Anantha. Breaking the simulation barrier: Sram evaluation through norm minimization. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design* (2008), IEEE Press, pp. 322–329.

[23] Dong, Changdao, and Li, Xin. Efficient sram failure rate prediction via gibbs sampling. In *Proceedings of the 48th Design Automation Conference* (2011), ACM, pp. 200–205.

[24] Feynman, Richard P. Simulating physics with computers.

[25] Flautner, K, Kim, NS, and Martin, S. Drowsy caches: Simple techniques for reducing leakage power. *International Symposium on Computer Architecture* (2002).

[26] Gassend, B, Clarke, D, and Van Dijk, M. Silicon Physical Random Functions. In *Proceedings of the IEEE Computer and Communications Society* (2002).

[27] Gassend, Blaise. Physical Random Functions. Master's thesis, MIT, USA, 2003.

[28] Gassend, Blaise, Clarke, Dwaine, Van Dijk, Marten, and Devadas, Srinivas. Silicon physical random functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security* (2002), ACM, pp. 148–160.

[29] Gassend, Blaise, Lim, Daihyun, Clarke, Dwaine, Van Dijk, Marten, and Devadas, Srinivas. Identification and authentication of integrated circuits. *Concurrency and Computation: Practice and Experience 16*, 11 (2004), 1077–1098.

[30] Gong, Fang, Yu, Hao, Shi, Yiyu, Kim, Daesoo, Ren, Junyan, and He, Lei. Quickyield: an efficient global-search based parametric yield estimation with performance constraints. In *Design Automation Conference,* (2010), IEEE, pp. 392–397.

[31] Guajardo, J, Kumar, S, Schrijen, GJ, and Tuyls, P. FPGA intrinsic PUFs and their use for IP protection. *Cryptographic Hardware and Embedded Systems* (2007).

[32] Guajardo, Jorge, Kumar, Sandeep S, Schrijen, G-J, and Tuyls, Pim. Physical unclonable functions and public-key crypto for fpga ip protection. In *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on* (2007), IEEE, pp. 189–195.

[33] Guajardo, Jorge, Kumar, Sandeep S, Schrijen, Geert-Jan, and Tuyls, Pim. *FPGA intrinsic PUFs and their use for IP protection*. Springer, 2007.

[34] Hashemian, Maryam S., Singh, Bhanu, Wolff, Francis, Weyer, Daniel, Clay, Steve, and Papachristou, Christos. A Robust Authentication Methodology Using Physically Unclonable Functions in DRAM Arrays. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition* (2015), DATE '15, pp. 647–652.

[35] Hiller, Matthias, Merli, Dominik, Stumpf, Frederic, and Sigl, Georg. Complementary IBS: Application specific error correction for PUFs. *International Symposium on Hardware-Oriented Security and Trust* (2012).

[36] Holcomb, D, Burleson, WP, and Fu, K. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. *Proceedings of the Conference on RFID Security* (2007).

[37] Holcomb, Daniel E, Burleson, Wayne P, and Fu, K. Power-up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *IEEE Transactions on Computers* (2009).

[38] Holcomb, Daniel E, and Fu, Kevin. Bitline PUF: Building Native Challenge-Response PUF Capability into Any SRAM. In *Cryptographic Hardware and Embedded Systems (CHES 2014)* (Sept. 2014), Lejla Batina and Matthew Robshaw, Eds., vol. 8731 of *Lecture Notes in Computer Science*, pp. 510–526.

[39] Holcomb, Daniel E, Rahmati, Amir, Salajegheh, Mastooreh, Burleson, Wayne P, and Fu, Kevin. Drv-fingerprinting: Using data retention voltage of sram cells for chip identification. In *Radio Frequency Identification. Security and Privacy Issues*. Springer, 2013, pp. 165–179.

[40] Hornik, Kurt, Stinchcombe, Maxwell, and White, Halbert. Multilayer feed-forward networks are universal approximators. *Neural Networks 2*, 5 (1989), 359–366.

[41] Horowitz, Mark, Alon, Elad, Patil, Dinesh, Naffziger, Samuel, Kumar, Rajesh, and Bernstein, Kerry. Scaling, power, and the future of cmos. In *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International* (2005), IEEE, pp. 7–pp.

[42] Hosseinabady, Mohammad, and Nunez-Yanez, Jose Luis. Run-time power gating in hybrid arm-fpga devices. In *Field Programmable Logic and Applications (FPL), 2014 24th International Conference on* (2014), IEEE, pp. 1–6.

[43] Hussein, Jameel, Klein, Matt, and Hart, Michael. Lowering power at 28 nm with xilinx 7 series devices.

[44] Jaffari, Javid, and Anis, Mohab. Adaptive sampling for efficient failure probability analysis of sram cells. In *Proceedings of the 2009 International Conference on Computer-Aided Design* (2009), ACM, pp. 623–630.

[45] Jin, Chenglu, Xu, Xiaolin, Burleson, Wayne, Rührmair, Ulrich, and van Dijk, Marten. Playpuf: Programmable logically erasable pufs for forward and backward secure key management.

[46] Johnson, Ian. Supercharging the embedded device: Arm cortex-m7.

[47] Juels, Ari, and Wattenberg, Martin. A fuzzy commitment scheme. In *Proceedings of the 6th ACM conference on Computer and communications security* (1999), ACM, ACM Press, pp. 28–36.

[48] Kelton, W David, and Law, Averill M. *Simulation modeling and analysis*. McGraw Hill Boston, 2000.

[49] Kerckhoffs, Auguste. *La cryptographie militaire*. University Microfilms, 1978.

[50] Khan, Seyab, Hamdioui, Said, Kukner, Halil, Raghavan, Praveen, and Catthoor, Francky. Incorporating parameter variations in bti impact on nano-scale logical gates analysis. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2012 IEEE International Symposium on* (2012), IEEE, pp. 158–163.

[51] Kim, Inyoung, Maiti, Abhranil, Nazhandali, Leyla, Schaumont, Patrick, Vivekraja, Vignesh, and Zhang, Huaiye. From statistics to circuits: Foundations for future physical unclonable functions. In *Towards Hardware-Intrinsic Security*. Springer, 2010, pp. 55–78.

[52] Kumar, Animesh, Qin, Huifang, Ishwar, Prakash, Rabaey, Jan, and Ramchandran, Kannan. Fundamental data retention limits in sram standby – experimental results. In *Quality Electronic Design, 2008. ISQED 2008. 9th International Symposium on* (2008), IEEE, pp. 92–97.

[53] Kumar, S S, Guajardo, J, Maes, R, Schrijen, G J, and Tuyls, P. The butterfly PUF protecting IP on every FPGA. *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on* (2008), 67–70.

[54] Lai, Ya-Chun, Huang, Shi-Yu, and Hsu, Hsuan-Jung. Resilient Self-V_{DD}-Tuning Scheme With Speed-Margining for Low-Power SRAM. *IEEE Journal of Solid-State Circuits 44*, 10 (Oct. 2009), 2817–2823.

[55] Lee, Jae W, Lim, Daihyun, Gassend, Blaise, Suh, G Edward, Van Dijk, Marten, and Devadas, Srinivas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *VLSI Circuits, 2004. Digest of Technical Papers. Symposium on*, pp. 176–179.

[56] Lee, Jae W, Lim, Daihyun, Gassend, Blaise, Suh, G Edward, Van Dijk, Marten, and Devadas, Srinivas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on* (2004), IEEE, pp. 176–179.

[57] Lee, J.W., Lim, Daihyun, Gassend, B., Suh, G.E., van Dijk, M., and Devadas, S. A technique to build a secret key in integrated circuits for identification and authentication applications. In *2004 Symposium on VLSI Circuits. Digest of Technical Papers.* (June 2004), pp. 176 – 179.

[58] Li, Shuo, and Salthouse, Christopher. Digital-to-time converter for fluorescence lifetime imaging. In *Instrumentation and Measurement Technology Conference (I2MTC), 2013 IEEE International* (2013), IEEE, pp. 894–897.

[59] Lim, Daihyun. Extracting secret keys from integrated circuits, MSc Thesis, 2004.

[60] Lofstrom, K., Daasch, W.R., and Taylor, D. IC identification circuit using device mismatch. In *IEEE International Solid-State Circuits Conference. Digest of Technical Papers.* (2000), pp. 372 –373.

[61] Lofstrom, Keith, Daasch, W Robert, and Taylor, Donald. Ic identification circuit using device mismatch. In *Solid-State Circuits Conference, 2000. Digest of Technical Papers, IEEE International* (2000), IEEE, pp. 372–373.

[62] Lorenz, Dominik, Georgakos, Georg, and Schlichtmann, Ulf. Aging analysis of circuit timing considering nbti and hci. In *On-Line Testing Symposium, 2009. IOLTS 2009. 15th IEEE International* (2009), IEEE, pp. 3–8.

[63] Maes, Roel. An accurate probabilistic reliability model for silicon pufs. In *Cryptographic Hardware and Embedded Systems-CHES 2013*. Springer, 2013, pp. 73–89.

[64] Maes, Roel, Tuyls, Pim, and Verbauwhede, Ingrid. Low-overhead implementation of a soft decision helper data algorithm for sram pufs. In *Cryptographic Hardware and Embedded Systems-CHES*.

[65] Maes, Roel, Tuyls, Pim, and Verbauwhede, Ingrid. Intrinsic pufs from flip-flops on reconfigurable devices. In *3rd Benelux workshop on information and system security (WISSec 2008)* (2008), vol. 17.

[66] Maes, Roel, Tuyls, Pim, and Verbauwhede, Ingrid. Intrinsic PUFs from flip-flops on reconfigurable devices. In *3rd Benelux workshop on information and system security (WISSec 2008)* (2008), vol. 17.

[67] Maes, Roel, Tuyls, Pim, and Verbauwhede, Ingrid. Low-Overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs. *Cryptographic Hardware and Embedded Systems* (2009).

[68] Maes, Roel, Tuyls, Pim, and Verbauwhede, Ingrid. A Soft Decision Helper Data Algorithm for SRAM PUFs. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on* (2009), IEEE, pp. 2101–2105.

[69] Maes, Roel, Van Herrewege, Anthony, and Verbauwhede, Ingrid. Pufky: A fully functional puf-based cryptographic key generator. *Cryptographic Hardware and Embedded Systems* (2012).

[70] Majzoobi, Mehrdad, Koushanfar, Farinaz, and Potkonjak, Miodrag. Lightweight secure pufs. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pp. 670–673.

[71] Majzoobi, Mehrdad, Koushanfar, Farinaz, and Potkonjak, Miodrag. Lightweight secure PUFs. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design* (2008), IEEE Press, pp. 670–673.

[72] Mathew, S K, Satpathy, S K, Anders, M A, Kaul, H, Hsu, S K, Agarwal, A, Chen, G K, Parker, R J, Krishnamurthy, R K, and De, V. A 0.19pJ/b PVT-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22nm CMOS. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International* (2014), pp. 278–279.

[73] Mathew, Sanu K, Satpathy, Sudhir K, Anders, Mark A, Kaul, Himanshu, Hsu, Steven K, Agarwal, Amit, Chen, Gregory K, Parker, Rachael J, Krishnamurthy, Ram K, and De, Vivek. A 0.19 pj/b pvt-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22nm cmos. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International* (2014), IEEE, pp. 278–279.

[74] Mathew, Sanu K, Satpathy, Sudhir K, Anders, Mark A, Kaul, Himanshu, Hsu, Steven K, Agarwal, Amit, Chen, Gregory K, Parker, Rachael J, Krishnamurthy, Ram K, and De, Vivek. 16.2 a 0.19 pj/b pvt-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22nm cmos. In *Solid-State Circuits Conference Digest of Technical Papers, IEEE International* (2014), IEEE, pp. 278–279.

[75] Nourivand, Afshin, Al-Khalili, Asim J, and Savaria, Yvon. Postsilicon Tuning of Standby Supply Voltage in SRAMs to Reduce Yield Losses Due to Parametric Data-Retention Failures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1 (2011), 29–41.

[76] Omega Engineering, Inc. *OSXL450 Infrared Non-Contact Thermometer Manual*, Last viewed Oct 27, 2014.

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

[78] Prabhu, P, Akel, A, Grupp, L, Yu, WK, Suh, G, Kan, Edwin, and Swanson, Steven. Extracting Device Fingerprints from Flash Memory by Exploiting Physical Variations. *Proceedings of the 4th International Conference on Trust and Trustworthy Computing* (2011).

[79] Qazi, Masood, Tikekar, Mehul, Dolecek, Lara, Shah, Devavrat, and Chandrakasan, Anantha. Loop flattening & spherical sampling: Highly efficient model reduction techniques for sram yield analysis. In *Proceedings of the Conference on Design, Automation and Test in Europe* (2010), pp. 801–806.

[80] Qazi, Masood, Tikekar, Mehul, Dolecek, Lara, Shah, Devavrat, and Chandrakasan, Anantha. Loop flattening & spherical sampling: highly efficient model reduction techniques for SRAM yield analysis. In *DATE '10: Proceedings of the Conference on Design, Automation and Test in Europe* (Mar. 2010).

[81] Qin, Hulfang, Cao, Yu, Markovic, D, Vladimirescu, A, and Rabaey, J. SRAM leakage suppression by minimizing standby supply voltage. In *5th International Symposium on Quality Electronic Design.* (2004), pp. 55–60.

[82] Qu, Gang, and Yin, Chi-En. Temperature-aware cooperative ring oscillator puf. In *Hardware-Oriented Security and Trust, IEEE 2009.*

[83] Raghavan Kumar, Xiaolin Xu, Wayne Burleson Sami Rosenblatt, and Kirihata, Toshiaki. Physically unclonable functions: A window into cmos process variations. *Circuits and Systems for Security and Privacy* (2016), 183 –244.

[84] Rahman, Tauhidur, Forte, Domenic, Fahrny, Jim, and Tehranipoor, Mohammad. Aro-puf: an aging-resistant ring oscillator puf design. In *Proceedings of the conference on Design, Automation & Test in Europe* (2014), European Design and Automation Association, p. 69.

[85] Rahmati, Amir, Hicks, Matthew, Holcomb, Daniel E, and Fu, Kevin. Probable cause: the deanonymizing effects of approximate dram. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture* (2015), ACM, pp. 604–615.

[86] Rahmati, Amir, Salajegheh, Mastooreh, Holcomb, Dan, Sorber, Jacob, Burleson, Wayne P, and Fu, Kevin. Tardis: Time and remanence decay in sram to implement secure protocols on embedded devices without clocks. In *Proceedings of the 21st USENIX conference on Security symposium* (2012), USENIX Association, pp. 36–36.

[87] Ransford, Benjamin, Clark, Shane, Salajegheh, Mastooreh, and Fu, Kevin. Getting things done on computational RFIDs with energy-aware checkpointing and voltage-aware scheduling. In *USENIX Workshop on Power Aware Computing and Systems (HotPower)* (December 2008).

[88] Rosenblatt, S, Chellappa, S, Cestero, A, Robson, N, Kirihata, T, and Iyer, S S. A Self-Authenticating Chip Architecture Using an Intrinsic Fingerprint of Embedded DRAM. *Solid-State Circuits, IEEE Journal of*, 99 (2013), 1–10.

[89] Rührmair, U, Devadas, Srinivas, and Koushanfar, Farinaz. Security based on physical unclonability and disorder. *Introduction to Hardware Security and Trust* (2011), 65.

[90] Rührmair, U, Sölter, J, Sehnke, Frank, Xu, Xiaolin, Mahmoud, Ahmed, Stoyanova, Vera, Dror, Gideon, Schmidhuber, Jürgen, Burleson, Wayne, and Devadas, Srinivas. Puf modeling attacks on simulated and silicon data. *Information Forensics and Security, IEEE Transactions on* (2013).

[91] Rührmair, Ulrich, and Holcomb, Daniel E. PUFs at a glance. In *Proceedings of the conference on Design, Automation & Test in Europe* (2014), European Design and Automation Association, p. 347.

[92] Rührmair, Ulrich, Martinez-Hurtado, JL, Xu, Xiaolin, Kraeh, Christian, Hilgers, Christian, Kononchuk, Dima, Finley, Jonathan J, and Burleson, Wayne P. Virtual proofs of reality and their physical implementation. In *2015 IEEE Symposium on Security and Privacy* (2015), IEEE, pp. 70–85.

[93] Rührmair, Ulrich, Sehnke, Frank, Sölter, Jan, Dror, Gideon, Devadas, Srinivas, and Schmidhuber, Jürgen. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security* (2010), ACM, pp. 237–249.

[94] Rührmair, Ulrich, Xu, Xiaolin, Sölter, Jan, Mahmoud, Ahmed, Koushanfar, Farinaz, and Burleson, Wayne. Power and timing side channels for pufs and their efficient exploitation. *IACR Cryptology ePrint Archive 2013* (2013), 851.

[95] Rührmair, Ulrich, Xu, Xiaolin, Sölter, Jan, Mahmoud, Ahmed, Majzoobi, Mehrdad, Koushanfar, Farinaz, and Burleson, Wayne. Efficient power and timing side channels for physical unclonable functions. In *Cryptographic Hardware and Embedded Systems–CHES 2014*. Springer, 2014, pp. 476–492.

[96] Satpathy, Sudhir, Mathew, Sanu, Li, Jiangtao, Koeberl, Patrick, Anders, Mark, Kaul, Himanshu, Chen, Gregory, Agarwal, Amit, Hsu, Steven, and Krishnamurthy, Ram. 13fj/bit probing-resilient 250k PUF array with soft darkbit masking for 1.94% bit-error in 22nm tri-gate CMOS. In *European Solid State Circuits Conference (ESSCIRC)* (2014), IEEE, pp. 239–242.

[97] Saxena, Nitesh, and Voris, Jonathan. We can remember it for you wholesale: Implications of data remanence on the use of ram for true random number generation on rfid tags. *arXiv preprint arXiv:0907.1256* (2009).

[98] Schuster, Dieter, and Hesselbarth, Robert. Evaluation of bistable ring PUFs using single layer neural networks. In *Trust and Trustworthy Computing*. Springer, 2014, pp. 101–109.

[99] Škorić, B, Tuyls, Pim, and Ophey, Wil. Robust key extraction from physical uncloneable functions. In *Applied Cryptography and Network Security* (2005), Springer, pp. 407–422.

[100] Skorobogatov, S. Low temperature data remanence in static RAM. Tech. Rep. UCAM-CL-TR-536, University of Cambridge Computer Laboratory, 2002.

[101] Su, Ying, Holleman, J., and Otis, B.P. A digital 1.6 pj/bit chip identification circuit using process variations. *IEEE Journal of Solid-State Circuits 43*, 1 (Jan. 2008), 69 –77.

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

[103] Suh, G. Edward, and Devadas, Srinivas. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference* (2007), DAC '07, ACM, pp. 9–14.

[104] Suh, G.E., O'Donnell, C.W., and Devadas, S. AEGIS: a single-chip secure processor. *IEEE Design & Test of Computers 24*, 6 (Nov.-Dec. 2007), 570 –580.

[105] Sun Electronic Systems, Inc. *Model EC1X Environmental Chamber User and Repair Manual*, 2011.

[106] Tajik, Shahin, Dietz, Enrico, Frohmann, Sven, Seifert, Jean-Pierre, Nedospasov, Dmitry, Helfmeier, Clemens, Boit, Christian, and Dittrich, Helmar. Physical characterization of arbiter PUFs. In *Cryptographic Hardware and Embedded Systems–CHES 2014*. Springer, 2014, pp. 493–509.

[107] Taur, Yuan. Cmos design near the limit of scaling. *IBM Journal of Research and Development 46*, 2.3 (2002), 213–222.

[108] Texas Instruments Inc. MSP430F241x, MSP430F261x Mixed Signal Microcontroller. In *Texas Instruments Application Report* (Jun. 2007, revised Nov. 2012).

[109] Tuyls, Pim, and Batina, Lejla. RFID-tags for anti-counterfeiting. In *Topics in Cryptology - CT-RSA 2006, volume 3860 of LNCS* (2006), Springer Verlag, pp. 115–131.

[110] Van Dijk, Marten E. System and method of reliable forward secret key sharing with physical random functions, Jan. 26 2010. US Patent 7,653,197.

[111] van Herrewege, Anthony, Katzenbeisser, Stefan, Maes, Roel, Peeters, Roel, Sadeghi, Ahmad-Reza, Verbauwhede, Ingrid, , and Wachsmann, Christian. Reverse fuzzy extractors: Enabling lightweight mutual authentication for PUF-enabled RFIDs. In *Financial Cryptography (FC) 2012* (Feb 2012), LNCS, Springer.

[112] Wang, Alice, Calhoun, Benton H, Chandrakasan, Anantha P, Chandrakasan, Anantha, and Chandrakasan, Anantha. *Sub-threshold design for ultra low-power systems*, vol. 95. Springer, 2006.

[113] Wang, Jiajing, and Calhoun, Benton Highsmith. Techniques to Extend Canary-Based Standby VDD Scaling for SRAMs to 45 nm and Beyond. *IEEE Journal of Solid-State Circuits 43*, 11 (2008), 2514–2523.

[114] Wang, Jiajing, Singhee, Amith, Rutenbar, Rob A, and Calhoun, Benton H. Two fast methods for estimating the minimum standby supply voltage for large srams. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 29*, 12 (2010), 1908–1920.

[115] Xu, Xiaolin, and Burleson, Wayne. Hybrid side-channel/machine-learning attacks on PUFs: a new threat? In *Proceedings of the conference on Design, Automation & Test in Europe* (2014), European Design and Automation Association, p. 349.

[116] Xu, Xiaolin, Burleson, Wayne, and Holcomb, Daniel E. Using statistical models to improve the reliability of delay-based pufs. In *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on* (2016), IEEE, pp. 547–552.

[117] Xu, Xiaolin, and Holcomb, Daniel. Reliable puf design using failure patterns from time-controlled power gating. In *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*.

[118] Xu, Xiaolin, and Holcomb, Daniel. A clockless sequential puf with autonomous majority voting. In *Proceedings of the 26th edition on Great Lakes Symposium on VLSI* (2016), ACM, pp. 27–32.

[119] Xu, Xiaolin, Rahmati, Amir, Holcomb, Daniel, Fu, Kevin, and Burleson, Wayne. Reliable physical unclonable functions using data retention voltage of sram cells. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 34*, 6 (June 2015), 903–914.

[120] Xu, Xiaolin, Rührmair, Ulrich, Holcomb, Daniel, and Burleson, Wayne. Security evaluation and enhancement of bistable ring pufs. In *Radio Frequency Identification*, Stefan Mangard and Patrick Schaumont, Eds., vol. 9440 of *Lecture Notes in Computer Science*. Springer International Publishing, 2015, pp. 3–16.

[121] Xu, Xiaolin, Suresh, Vikram, Kumar, Raghavan, and Burleson, Wayne. Post-silicon validation and calibration of hardware security primitives. In *2014 IEEE Computer Society Annual Symposium on VLSI* (2014), IEEE, pp. 29–34.

[122] Yamamoto, Dai, Sakiyama, Kazuo, Iwamoto, Mitsugu, Ohta, Kazuo, Ochiai, Takao, Takenaka, Masahiko, and Itoh, Kouichi. Uniqueness enhancement of PUF responses based on the locations of random outputting RS latches. In *Cryptographic Hardware and Embedded Systems–CHES 2011*. Springer, 2011, pp. 390–406.

[123] Yamamoto, Dai, Takenaka, Masahiko, Sakiyama, Kazuo, and Torii, Naoya. Security evaluation of bistable ring PUFs on FPGAs using differential and linear analysis. In *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on* (2014), IEEE, pp. 911–918.

[124] Yin, CED, and Qu, G. LISA: Maximizing RO PUF's Secret Extraction. *Hardware-Oriented Security and Trust* (2010).

[125] Yu, MD, M'Raihi, David, Devadas, Srinivas, and Verbauwhede, Ingrid. Security and reliability properties of syndrome coding techniques used in puf key generation. In *GOMACTech Conference* (2013), pp. 1–4.

[126] Yu, Meng-Day, and Devadas, S. Secure and Robust Error Correction for Physical Unclonable Functions. *IEEE Design & Test of Computers 27*, 1 (2010), 48–65.

[127] Zhao, Wei, and Cao, Yu. New generation of predictive technology model for sub-45 nm early design exploration. *Electron Devices, IEEE Transactions on 53*, 11 (2006), 2816–2823.