A Study on Modeling of MUX-based Physical Unclonable Functions

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Anoop Koyily

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Master of Science in Electrical Engineering

Advisor: Keshab K. Parhi

April, 2018

Acknowledgements

I would like to acknowledge the support of my adviser, Prof. Keshab K. Parhi and Prof. Chris H. Kim in guiding and providing the necessary direction throughout this work.

I would also like to acknowledge the help of my lab colleagues for providing the help and contribution to this work, and also all those people who directly and indirectly have helped me in pursuing this work.

Abstract

Physical Unclonable Functions (PUFs) are simple circuits that are ideal for hardware security. Typically, they are used for identifying and authenticating integrated circuits (ICs). In this work, we are interested in a class of delay based PUFs which mainly consist of multiplexers. They are known as multiplexer-based PUFs or *MUX PUFs*, for short. We are interested in modelling their structure and then, analyzing their performances.

Our work can be mainly divided into some key contributions. First, we discuss about the different types of MUX PUFs that we deal with in this work. They are the simple or linear configuration, feed-forward configuration and modified feed-forward configuration. We then, present a typical scheme used for the authentication of these PUFs. However, much of the work concentrates on a modified version of the authentication scheme, where instead of storing a look-up table (LUT) of challenge-response pairs (CRP) in the server, we store a set of delay parameters corresponding to the physical attributes of the MUX PUF. These stored parameters are the delay-differences of the MUX stage and the arbiter delay. We show that MUX PUFs can be modelled using an *additive linear delay model*. The additive model helps in the computation of an important parameter, known as *total delay-difference*. Based on the total delay-difference, we can compute two different versions of the output or response: *hard-response*, which is either a '0' or '1' bit and *soft-response*, which can take continuous values between 0 and 1. We formulate models for obtaining both these responses. Various metrics used for the evaluation of PUF performance are discussed. The general lab setup used to collect the required PUF data is also discussed.

Next, we discuss about the various effects of aging on the performance of MUX PUFs. We extend the linear delay model to include the variations in delay parameters due to aging. The model makes certain assumptions about how noise and aging affect the delay chain (consisting of the multiplexers) and the arbiter. We assume that for a fixed set of conditions, the noise can only cause a constant amount of degradation to the performance of an aging PUF. However, aging which is caused due to undesirable changes like negative bias temperature instability (NBTI), hot carrier injection (HCI) and time dependent dielectric breakdown (TDDB) results in a gradual degradation of

performance. That is, the variations due to aging gradually increase with time in contrast to that of noise. In our study, we compare the standalone effects of aging and noise on the PUF. We observe that for the same amount of variation, aging degrades the authentication performance much more than noise. Furthermore, experimental aging data collected from PUFs in our lab suggest that the percent variation in delay parameters can be modelled as a Gaussian distribution. However, there is a small difference in how the percent variations of delay-differences of MUX stages and the arbiter delay are modelled. The former is a zero mean Gaussian, whereas the latter is a positive mean Gaussian with mean and variance both gradually increasing with aging. In addition, the variation in arbiter delay is assumed to be higher than that of delay-differences due to "asymmetric" aging in case of arbiter. This happens under *unequal aging scenario*. Using a Monte-Carlo based simulation for aging, authentication accuracy of the three configurations are studied. We also suggest approaches to improve the authentication accuracy that will increase the lifetime of a PUF. This can be done by either recalibrating the delay parameters or by tuning a threshold based on total delay-difference.

Next, we discuss an entropy based approach that can be used to identify whether a MUX is linear or non-linear. The approach is focused on computing the conditional entropy of responses to a set of predefined challenges. The challenge set consists of randomly chosen challenges and their 1-bit neighbors. The entropy is computed across the responses of two 1-bit neighboring challenges. For non-linear MUX PUFs like feed-forward, the method determines the MUX stages which are controlled by internally generated challenge bits as opposed to external challenge bits. This is based on the observation that the conditional entropy for each of these stages is zero. Also, the number of zero conditional entropy values across the MUX stages provide an upper bound on the number of internal arbiters present in the PUF. With the proposed approach, we observe 100% sensitivity and 100% specificity for identifying non-linearity. Furthermore, we show that the proposed approach requires very less number of stable random challenges (about 50) for successfully determining whether a PUF is linear or not for real chips.

Our next contribution involves a logistic regression based approach to predict the soft-response for a challenge using the total delay-difference as an input. This approach enables us to determine whether a challenge is stable or not. The approach learns

a logistic function based on the total delay-difference which has just 3 parameters. Therefore, this is a simple approach which gives comparable performance against a more complex approach based on artificial neural network (ANN) models. The model demonstrates good sensitivity and precision but poor specificity.

Finally, we discuss a bit-flipping algorithm used to convert the unstable challenges to stable challenges. It is based on the idea that a threshold on the total delay-difference can guarantee stability of challenges. The thresholds can be obtained empirically from the probability distributions of the total delay-difference. A straightforward approach is to discard and issue a new random challenge for authentication if the current challenge is unstable. In this paper, we propose a novel bit-flipping based approach in which we claim that by flipping few bits of the original unstable challenge, we can convert it to a stable one with minimal number of bit-flips. By using the algorithm, we are able to transform the most likely unstable challenges to stable ones, typically with 1 bit-flip for linear and modified feed-forward PUFs and 3 bit-flips for the feed-forward PUFs. These bit-flips correspond to the flips in the XOR-ed challenge. We also compare the computation complexities of best, average and worst-case scenarios for the straightforward and proposed approaches. In terms of number of addition operations, the proposed approach has slightly better average-case performance but much better worst-case performance than the straightforward approach.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Physical unclonable functions (PUFs) [1, 2, 3] are novel security primitives used in applications of hardware security. They are formulated from the notion of physical random functions [2]. They can be implemented using conventional integrated circuit (IC) design techniques. This leads us to a method of identifying and authenticating individual ICs and a means of building secure smartcards. There are many methods available to identify and authenticate ICs. One can embed a unique identifier in an IC to give it a unique identity. This approach can identify the IC, but cannot authenticate it. To enable authentication, one needs to embed a secret key onto the IC. This key needs to remain secret, which means the IC has to be resistant to attacks which may attempt to discover the key. Numerous attack models have been described in the literature [4, 5, 6].

The secret keys in a PUF are due to the uncontrollable process variations that occur during the manufacturing. That is, they are derived from the complex physical properties of the PUF rather than the traditional method of obtaining them from a non-volatile memory. For example, volatile secret keys can be generated just from the random delay characteristics of wires and transistors. Such keys form the *signature* of a hardware like PUF. Because a device like PUF taps into the random variations occuring during the fabrication process, these secret keys are difficult to predict, clone or duplicate. PUFs, therefore, present an efficient and reliable way to generate volatile secrets that only exist in digitial form when they are powered on and running [7].

Various types of PUFs have been discussed in the literature [8, 9]. They can be mainly differentiated based on how the randomness was originally introduced, i.e., intrinsically or extrinsically. The inherent properties of a PUF greatly depends up on this differentiation. Extrinsic-based PUFs have a much greater ability to distinguish between devices and have minimal environmental variation compared to intrinsic-based PUFs. This is because of the underlying principles and parameters which can be directly controlled and optimized for extrinsic-based PUFs. However, intrinsic-based PUFs are more attractive because they can be included in a design without any modification to the manufacturing process. Some of the popular intrinsic-based PUF designs are: delay-based PUFs (which include ring-oscillator (RO PUF) [10], multiplexer-based (MUX PUF) [7, 11]), SRAM-based PUFs [12] etc. In this work, we will be dealing with delay-based MUX PUFs.

The thesis is organized in the following way:

- Chapter 2 discusses about the various types of multiplexer-based PUFs (MUX PUFs) and the schemes used for authentication. We also discuss the formulation of linear delay model, hard and soft-response and also how challenge-response pairs (CRPs) are classified as stable and unstable. We also discuss about the various metrics used to analyze the performance of PUFs. The basic lab setup used for the collection of data is also described.

- Chapter 3 discusses the various effects of aging on PUFs. We propose a Monte-Carlo based simulation model for analyzing the effects of aging.

- Chapter 4 discusses an entropy based approach to detect linearity in MUX PUFs.

- Chapter 5 proposes a logistic regression based model for classification of challenges as stable or not. The model is used to train a response curve for the arbiter.

- Chapter 6 discusses the bit-flipping algorithm that can be used to convert an unstable challenge to a stable one.

- Chapter 7 concludes the paper and discusses possible future directions.

Chapter 2

Multiplexer-based delay PUFs

2.1 Introduction

Just like any other device, a PUF can be modelled in terms of its input-output relationship. The input is termed as *challenge* and output as *response*. This input-output relationship can be termed as a *challenge-response pair*, or CRP for short. For transistors used in PUFs, manufacturing randomness exists due to variations in transistor length, width, gate oxide thickness, doping concentration density, body bias, metal width, metal thickness, and interlevel dielectric (ILD) thickness, and so on [11, 13]. These manufacturing variations lead to a significant amount of variability in PUFs. This helps in the generation of significant number of unique CRPs for each PUF IC. In this work, we mainly deal with the multiplexer-based PUFs.

A multiplexer or MUX PUF is an example of strong PUF [14], which can accommodate many possible challenge-response pairs (CRPs). One of the advantages of a MUX PUF is its simplistic design that enables flexibility in its structure. There are various types of MUX PUFs available in PUF literature [11, 15].

A simple MUX PUF, also called as linear MUX PUF, is shown in Fig. 2.1. It has N multiplexer stages and an arbiter at the end. The response is generated based on the input challenge and the inherent process variations of the PUF chip. The process variations are due to the variations in delay parameters of the multiplexer and arbiter. Control bit for each MUX stage is obtained from the N-bit input challenge. That is, MUXes in each stage act as a switch to either cross or straight propagate the rising

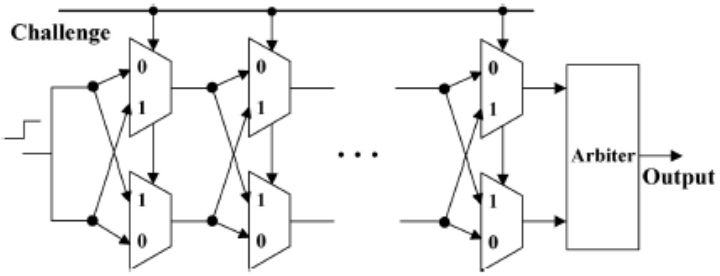


Fig. 2.1: A Linear MUX PUF

edge signal, based on the corresponding challenge bit. Therefore, there exists two paths traversed by the rising edge signal. The arbiter which is usually an SR-latch or flip-flop, outputs a 1-bit response. It translates the analog timing difference between the two paths into a digital value. For instance, if the rising edge signal arrives at the top input of the arbiter earlier than the signal arriving at the bottom input, the output will be '1'; otherwise, if it reaches the bottom path first, the output will be '0'.

Using the skeletal structure of the simple linear MUX PUF (Fig. 2.1), more complex MUX PUF configurations are possible. Two such examples are the feed-forward and modified feed-forward configurations [11, 16], shown in Fig. 2.2. There also exist reconfigurable PUF configurations like logic-reconfigurable or CRP-reconfigurable [11, 17, 18]. Examples of logic-reconfigurable PUFs are overlap, cascade and separate feed-forward structures. They are classified based on the arrangement of internal arbiters in the PUF structure. We will analyze these configurations using an entropy based approach in Chapter 4 [19].

Feed-forward structures shown in Fig. 2.2 have non-linearity added into the original linear MUX PUF configuration. This is done by inserting an additional internal (or intermediate) arbiter. Furthermore, the modified feed-forward configuration proposed in [11] has an additional inter-connection in the intermediate stage. Fig. 2.2 shows an example in which the internally generated challenge bits are at successive stages, N_2 and N_2+1. The main purpose of these configurations is to increase the security of MUX PUFs. Works in [5, 20] analyze the security aspect of these PUFs. In this work, we will analyze and compare the performance of the three configurations discussed so far, i.e., linear, feed-forward and modified feed-forward configurations.

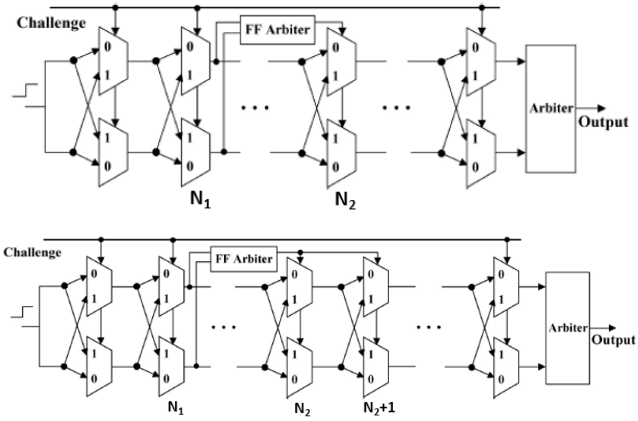


Fig. 2.2: Feed-forward PUF (top), and modified feed-forward PUF (bottom) configurations

2.2 Linear Delay Model

2.2.1 Total Delay-Difference and Hard Response

Previously, we discussed the basic structure that constitutes a linear MUX PUF as shown in Fig. 2.1. The input is an N-bit challenge and output is 1-bit response. A rising edge signal excites two paths at the first MUX stage simultaneously. The actual (two) propagated paths are determined by the N-bit input challenge. After the last MUX stage, the arbiter generates 1-bit response by comparing the arrival times of the two paths at its input. Therefore, it is standard to model a MUX PUF using an additive linear delay model [11, 21, 22], as shown below:

$$r_N = \sum_{i=1}^{N+1} (-1)^{C'_i} \Delta^i = \overbrace{\underbrace{\sum_{i=1}^{N} (-1)^{C'_i} \Delta^i}_{\text{Delay Chain}} + \underbrace{\Delta^{arb}}_{\text{Arbiter}}}^{\text{Additive linear model}} \tag{2.1}$$

$$R = sign(r_N) = \begin{cases} 1, & r_N \geq 0 \\ 0, & r_N < 0 \end{cases} \tag{2.2}$$

In (2.1, 2.2), r_N is the *total delay-difference* and R is the final response bit. $C'_i = \oplus_{j=i}^{N} C_j$ is the XOR-ed challenge bit corresponding to challenge bit C_i, Δ^i is the i^{th} stage delay-difference and $\Delta^{arb} = \Delta^{N+1}$ ($C'_{N+1} = 0$) is the bias corresponding to the arbiter delay [11].

The N MUX stages constitute the *delay chain*. The final response bit, R, takes value '0' or '1' depending on the sign of total delay-difference, r_N. Response, R, is also called as a *hard-response* because it can only take binary values '0' or '1'.

2.2.2 Soft-Response

We can also define the PUF response in terms of a *soft-response*, R_s. Soft-response, R_s, can be defined as the probability of response to be '1' (or '0'), i.e., R_s=$P(R =$ '1'$)$=$\frac{R_1}{M}$, where R_1 is the number of times the response bit is '1' out of M measurements. The probabilistic behaviour of the response can be modelled by considering factors like *environmental noise* and *metastability*.

Environmental noise can be attributed due to changes in supply voltage or temperature. In such a case, the model in (2.1) needs to be modified to include the variations due to noise. Additive noise model is used for this purpose. The modified expression after adding the noise variations to the delay parameters, Δ^i and Δ^{arb}, is shown below:

$$r_N = \sum_{i=1}^{N+1} (-1)^{C'_i} \Delta^i + \sum_{i=1}^{N+1} n_i \tag{2.3}$$

where n_i is the environmental noise contribution in the MUX stages and arbiter.

However, to study the effects of metastability, we need to consider timing parameters of the arbiter like setup and hold time. For a CMOS based latch, valid data must be present at the input for a specified period of time before the clock signal arrives. This is the *setup* time. Also, the data must remain valid for a specified period of time after the clock transition which is its *hold* time. Any change in data signal which occurs between these times will result in output reaching intermediate voltage levels, and remain there for an indefinite amount of time before resolving to either a high or low signal. This stable high or low output state depends upon the process technology, manufacturing and environmental conditions. It is almost impossible to predict the final stable state. More secure configurations like feed-forward show a higher degree of metastability compared to linear or modified feed-forward configurations [23, 24, 25, 26].

Fig. 2.3 shows two possible metastable conditions that can occur in the arbiter. First case (denoted by I) shows the signal states when the clock (or reset, R) arrives later at the arbiter relative to the data, and second (denoted by II) when the clock arrives

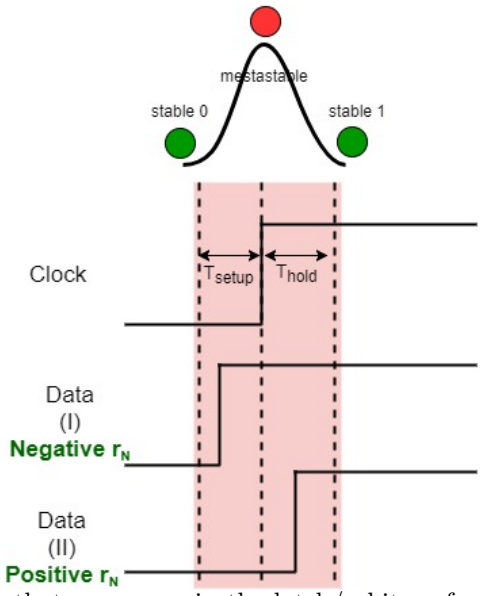


Fig. 2.3: Metastable states that can occur in the latch/arbiter of a MUX PUF in case of two different scenarios

first relative to the data (or set, S). In the former case, the total delay-difference, r_N, is negative and in the latter it takes a positive value. This is analogous to a ball rolling over a hill (due to Gaussian nature of r_N) and each side represents a stable state for the response bit. The top of the hill represents a metastable state [27]. When $|r_N| \lessapprox T_{setup}$ (I) or $|r_N| \lessapprox T_{hold}$ (II) (shaded area in figure), the PUF is in a metastable state.

2.2.3 Stable and Unstable CRPs

Due to the effects of metastability and environmental noise, the PUF response, R, to a given challenge can vary over time. Due to these variations, the soft-response, R_s, to a challenge can take any value in between 0 and 1. We can, therefore, classify the challenge-response pairs (CRPs) as either *stable* or *unstable* based on a threshold defined on the soft-response, R_s. We choose the threshold to be 0.1-0.9, which means if the soft-response, R_s, is between 0.1 and 0.9, the CRP is deemed *unstable*, otherwise it is considered to be a *stable* '0' or '1'. In our lab, we use a soft-response based chip to do such a classification [28]. We will discuss more about the lab setup in the next sections.

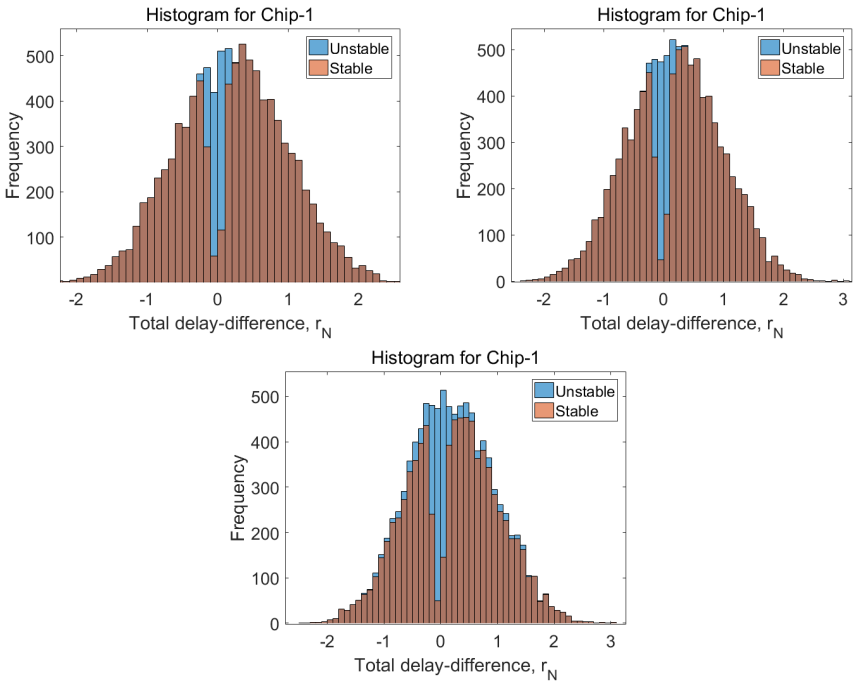


Fig. 2.4: Histogram of total delay-difference, r_N, for linear (top left), modified feed-forward (top right) and feed-forward (bottom) configurations (using ground truth from the chips).

2.2.4 Probability distribution of Total Delay-Difference

Similar to the linear configuration in (2.1, 2.2), expressions for total delay-difference, r_N, and final response bit, R, in case of feed-forward and modified feed-forward configurations can also be obtained. Independent of the PUF configuration, an N-stage MUX PUF has total delay-difference, r_N, distributed as $N\left(\mu_{arb}, 2N\sigma^2 + \sigma^2_{arb} + (N+1)\,\sigma^2_n\right)$, where $N(0, 2N\sigma^2)$ is due to the multiplexer stages, $N(\mu_{arb}, \sigma^2_{arb})$ is due to the arbiter delay and $N\left(0, (N+1)\,\sigma^2_n\right)$ is due to environmental noise. Here, σ^2 is the variance of multiplexer delay, μ_{arb} and σ^2_{arb} are the mean and variance of arbiter delay and σ^2_n is the variance of environmental noise.

Fig. 6.2 shows the distribution of total delay-difference, r_N, with 10,000 random challenges for the three PUF configurations. Observe that the feed-forward configuration has a higher spread of r_N for unstable challenges (blue color). This is due to an increased metastability [26]. For our chips, variance of r_N for unstable challenges in case of feed-forward configuration is atleast 10 times more than the other two configurations. For linear and modified feed-forward configurations, total delay-difference, r_N, of unstable

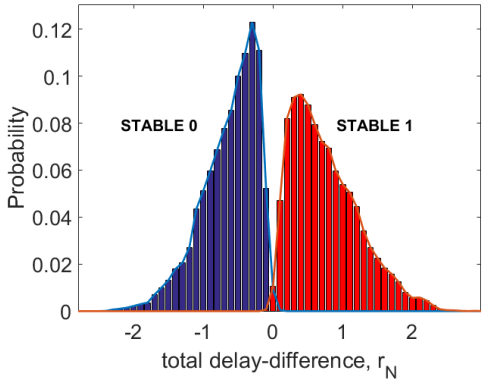


Fig. 2.5: Probability distributions of r_N corresponding to stable 0 and stable 1 response bits for a linear PUF with noise std=5% of std of Δ^i.

challenges are much more closer to 0. As discussed before, the definition of unstable CRPs comes from a thresholding (0.1-0.9) based on the *soft-response* [28]. Feed-forward configurations have about 85% of challenges being stable, compared to about 89-90% in the case of linear and modified feed-forward configurations. Fig. 2.5 shows the probability distribution of r_N corresponding to the stable 0 and stable 1 CRPs for a linear PUF. As can be observed from the figure, there is a small degree of overlap between the distributions. This corresponds to the environmental noise present in (2.3) and represents the error present in the proposed model.

2.3 Authentication of PUFs

PUF devices are used in hardware security for applications of device authentication [29]. A typical PUF authentication scheme is shown in Fig. 2.6 [28]. It can be described in terms of two phases:

- Enrollment phase: A large set of challenge-response pairs (CRPs) is measured from each fabricated chip and stored on a secure server. The stored data is in the form of lookup tables (LUTs) as shown in Fig. 2.6.

- Authentication phase: The server receives an authentication request along with the chip ID from the user and selects a list of random challenges from its database. These challenges are sent to the user, and their responses are sent back to the

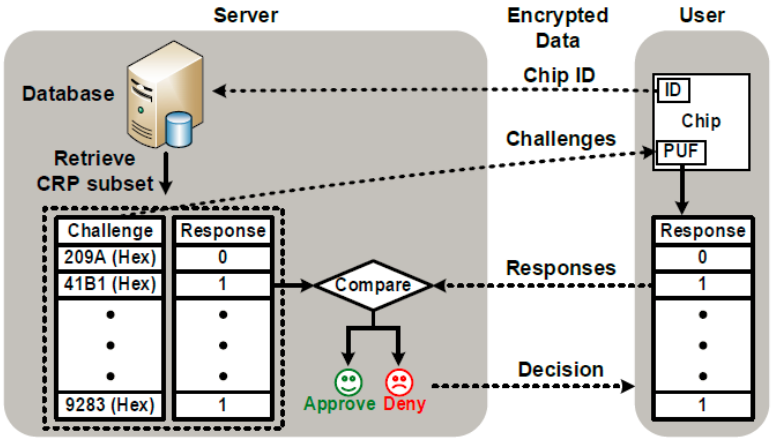


Fig. 2.6: Typical authentication procedure based on chip ID and PUF

server. The user is "granted access" to the PUF if the responses from the chip match to those stored in the LUT.

The CRPs stored in the LUT during the enrollment phase form the *signature* of a PUF. These CRPs are unique for a given PUF chip. However, to maintain a sufficient amount of uniqueness, the LUT can typically contain thousands of CRPs which require a huge amount of server space for storing them. There are other alternative approaches that consume much lesser storage area [5, 6]. Work in [5] discusses the idea of estimating the delay parameters of a MUX PUF and storing them in the server for the computation of final response. The final response obtained from the stored parameters is then compared against that obtained from the PUF when excited by the chosen set of challenges. This method only requires storage space corresponding to the size of MUX PUF in use. We will utilize this approach later in Chapter 3.

For successful authentication of PUFs, we typically define a threshold corresponding to the amount of error that can be tolerated on the response bits. For a given response length (corresponding to the number of CRPs used for authentication) and threshold, we can compute the false positive and false negative rates. False positive rate is the probability that PUF A will be authenticated as PUF B when PUF A produces the same output as PUF B, and false negative rate is the probability that a correct PUF will fail to be authenticated. The former happens when two PUFs generate similar outputs, whereas the latter happens when a PUF fails to generate a consistent output.

An example is discussed in [7] where an error of 10 out of 128 bits is tolerated for a successful authentication. The false positive and false negative rates are calculated to be about 2.1×10^{-21} and 5×10^{-11}, respectively. These are computed based on the inter-chip and intra-chip variations, respectively. False positive depends on the relative uniqueness (or inter-chip variation) of a PUF A with respect to another PUF B. Higher the uniqueness (or inter-chip variation) between the two chips, *lower* is the false positive rate. On the other hand, false negative rate depends on the stability (or consistency) of the responses being generated by the PUF. The responses to certain challenges are more inconsistent than the others. Therefore, having more of such CRPs (i.e., ones with lower stability or higher intra-chip variation) contribute to a *higher* false negative rate. In the previous section, we had discussed about unstable CRPs which possess soft-response values between 0.1 and 0.9. While selecting random challenges for authentication, we prefer to avoid such challenges. However as discussed previously, the probability of encountering such CRPs is low, in the range 10-15%. In case these challenges get selected, we need to ensure that the error in the response bits is within the tolerance limit. In Chapter 6, we will discuss some approaches to deal with unstable CRPs.

2.4 PUF Performance Metrics

There are various metrics that can be used to analyze the performance of different PUF configurations [11, 30, 7, 31]. In previous section, we observed that among the three PUF configurations, feed-forward is the most unreliable due to a higher number of unstable CRPs. In this case, we used reliability as the performance metric. Similarly, we can list other performance metrics as follows:

- *Reliability :* Reliability or intra-chip variation is a measure of reliability of the PUF, which is determined by comparing the digital signatures of the PUF to the same challenge under different environmental conditions. It is the probability that a certain response bit will flip when a given challenge is applied multiple times. It is essentially same as a soft-response, R_s, except it is averaged across a set of CRPs rather than just one CRP. The set of CRPs is the one chosen for authentication. Reliability is computed as $1 - P_{intra}$, where P_{intra} is the intra-chip variation of the PUF.

- *Uniqueness :* Uniqueness or inter-chip variation is determined by comparing the digital signature of a PUF to that of another. All possible chip-combinations should be considered for the comparison. Inter-chip variation, P_{inter}=50% represents the best uniqueness for a PUF. Uniqueness is computed as $1 - |2P_{inter} - 1|$.

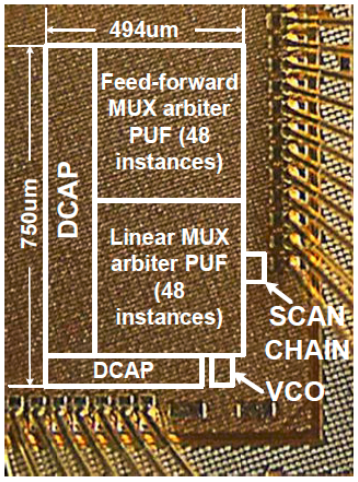
- *Randomness :* Ideally, a MUX PUF is expected to produce unbiased '0' or '1' as the response. Randomness represents the ability of the PUF to output the response as '0' or '1' with equal probability. Therefore, it is equal to 1-$|2P(R = 1)$-1|.

- *Unpredictability :* Unpredictability measures the degree of security possessed by a particular PUF configuration. In other words, it ensures that an adversary cannot efficiently compute the PUF response to an unknown challenge, even if he can adaptively obtain a certain number of other CRPs from the same and other PUF instances. PUF literature discusses various attack based models that analyze the unpredictability of different PUF configurations.

2.5 PUF Setup

In this section, we will discuss about the hardware setup used for our experiments. Our chips implement 32 stage MUX PUF which has approximately 4.3×10^9 challenge choices ($=2^{32}$). The technology used for the fabrication is 32nm IBM HKMG [28]. Fig. 2.7 shows the basic layout of the IC used in the lab. Typical voltage, temperature and other settings like type of arbiter and voltage-controlled oscillator (VCO) frequency are also shown in the table.

Each PUF IC has 48 linear and feed-forward MUX PUF configurations arranged in a 6×16 grid. Top three rows are the linear PUFs and the bottom three are the feed-forward configured PUFs. The feed-forward configuration can be configured as a linear configuration by disabling the intermediate stage. Likewise, it can also be configured as a modified feed-forward configuration by adding an extra inter-connection, as in Fig. 2.2. Top three rows are the linear PUFs and the bottom three are the feed-forward configured PUFs.



Process	32nm
VDD	0.8~1.0V (normal: 0.9V)
Temperature	25°C, 85°C
Ckt. area	0.37 mm^2
PUF type	Linear and Feed-forward MUX arbiter PUF
PUF stages	32
# of PUFs	48 (linear) + 48 (feed-forward)
VCO frequency	<1.4 GHz
Arbiter	SR latch

Fig. 2.7: 32nm chip microphotograph and summary table.

2.5.1 Data Collection

In previous section, we have discussed about the importance of a soft-response based analysis of the PUF output. We use temporal majority voting (TMV) scheme to convert a soft-response to a hard-response. It is essentially used to address the response instability occurring due to factors like metastability and environmental noise. According to this method, the PUF response is read out multiple times and the majority value is taken as the final PUF response [28, 32, 33, 34]. However, the so called *unstable* CRPs have variations in their responses. To overcome this limitation, an authentication strategy based on thresholding of the soft-response is used. As discussed before, the threshold chosen is 0.1-0.9.

In the PUF chip, we take 102,400 measurements of the response to a given challenge. By using an 1/1024 fast on-chip divider, the measurements are then converted to a value between 0 and 1. This is the *soft-response*. For example, if we obtain the response bit as '1' for 100,000 times out of 102,400, the soft-response, R_s, is ~0.977. Furthermore, by applying a threshold of 0.1-0.9, we can consider the soft-response value of 0.977 to be a *stable '1'* bit ($\because$ 0.977>0.9). Using similar steps, we can classify all CRPs to be either *stable '0'*, *stable '1'* or *unstable*. The experimental results show that by selecting the stable CRPs based on soft responses, the PUFs can work reliably under a wider range of VDD and temperature.

Fig. 2.8 shows the PUF design which can collect massive PUF data using an on-chip voltage controlled oscillator running at gigahertz frequencies [28]. The basic idea is to measure the probability of the response being '1' or '0' using an on-chip counter which counts the arbiter outputs, and compare the value with the total number of VCO cycles. The ratio between the two count values is the probability of the response being '0', P(R='0'). The probability of response being '1' can be computed as, P(R='1')=1-P(R='0').

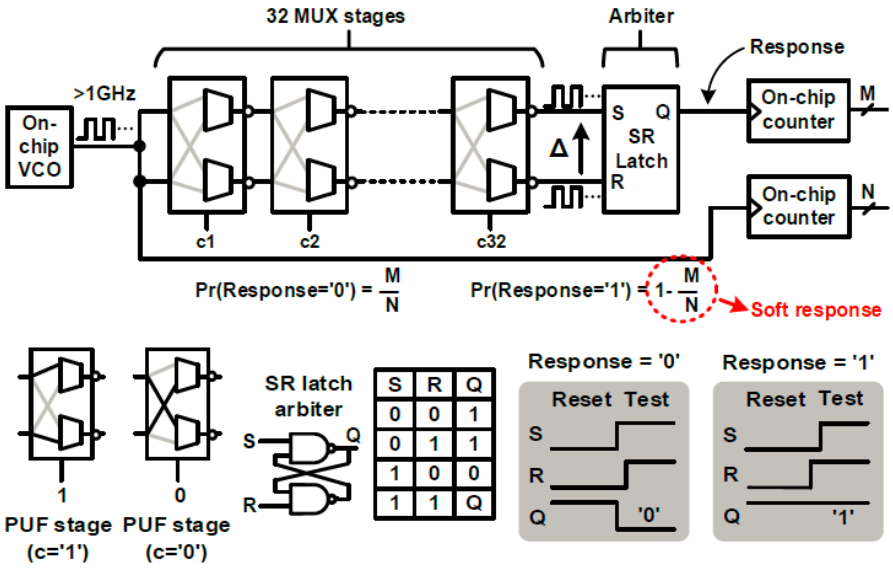


Fig. 2.8: MUX PUF design utilizing an on-chip voltage controlled oscillator circuit and counters to efficiently collect soft response.

Chapter 3

Effect of Aging on MUX-PUFs

3.1 Introduction

The behavior of challenge-response pairs (CRPs) of MUX PUFs can be modeled using various adaptive learning techniques [5, 6, 20]. As mentioned in previous chapter, we are particularly interested in a least mean square (LMS) based approach which has been used to estimate the delay-difference of MUX stages [5]. The estimated model parameters can then be stored in a server database and used for authentication. While similar approaches have been proposed before [6, 35], this method proposes to store the physical parameters of the model as opposed to those of an artificial neural network (ANN) or other non-linear models [20]. Furthermore, the parameters of ANN or other models do not correspond to the intrinsic physical parameters of the chip. The proposed method in [5] can be considered canonic as it is based on the physical parameters that correspond to N multiplexer stages and arbiter(s).

For applications in authentication, we desire PUFs to perform reliably over time. But various uncontrollable factors like environmental noise, metastability and aging degrade the authentication accuracy of a PUF. Environmental noise mainly occurs due to variations in temperature and/or supply voltage; and metastability occurs due to timing violations at the input of arbiter. However, for a given environmental condition, the effect of environmental noise and metastability on the PUF performance is a constant. In contrast, aging affects the reliability of a PUF over an extended period of time.

Aging is mainly caused due to undesirable changes in hardware structure such as

negative bias temperature instability (NBTI), hot carrier injection (HCI) and time dependent dielectric breakdown (TDDB) [36, 37]. NBTI and HCI, in particular, are known to induce progressive slowdown in hardware [37, 38]. This results in a gradual increase in the delays of hardware structures like multiplexers. In the proposed aging model, the increase in delays results in a slowdown effect in terms of the delays of multiplexers and arbiter(s). It is known that the multiplexer delays of a PUF are distributed randomly [21, 22]. We assume that, due to aging, the mean and variance of these delays gradually increase. This has been shown to be a valid assumption in prior aging works on delay-based PUFs [39]. However, our model considers the delay-difference rather than the delays for each MUX stage. We assume that the delay-differences are distributed with zero mean and variance which increases gradually with aging. In addition to the delay stages, the arbiter (which is typically an SR latch) also forms a key component in the functioning of a PUF. In [24, 40], it is shown that for a latch/flip-flop, various timing related factors like setup time, hold time, clock-to-output and data-to-output can increase with aging. This work considers the effect of aging on arbiter in terms of only its propagation delay (or clock-to-output). In [39], it is shown that the arbiter in a delay-based PUF forms an Achilles' heel due to its "asymmetric" aging. Therefore, we can assume that the variations in the arbiter due to aging would be much more significant compared to the stage delay-differences. We will discuss this more in the next parts.

Our work in [41] proposes an aging and noise based model for analyzing MUX-based PUFs. The model simulates conditions close to that for a real chip. The role of such a statistical simulation framework is important in the sense that both existing and new PUF structures can be characterized with respect to aging and noise effects without fabricating chips. Using the model, we investigate how various PUF configurations are affected by aging. One may be led to believe that the delay difference of a MUX stage would not be affected by aging if both the delay paths age by same amount. However, this is not true as the top and bottom path-delays both increase but by different amounts even after applying the same stress condition for the same stress duration [42, 43]. If the top path-delay increases more (or less) than the bottom path-delay, the total delay-difference increases (or decreases) and thus, the final response bit can flip. Intuitively, this explains why the delay difference variation can be modeled as a zero-mean Gaussian

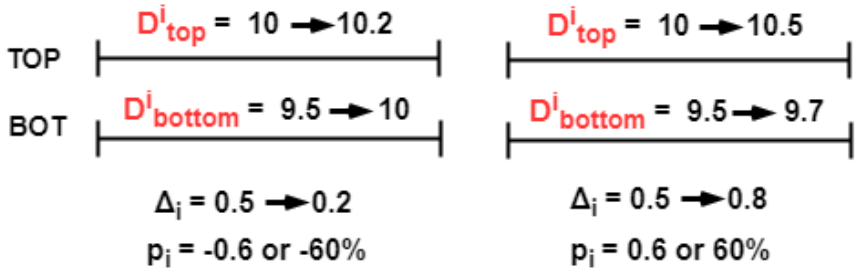


Fig. 3.1: An example showing two scenarios of how the delay-difference of a MUX stage can vary with aging. D^i corresponds to delays of top and bottom multiplexers, Δ^i to the delay difference and p_i to the percentage change in delay-difference. p_i can be both positive or negative with aging.

random variable. An example is shown in Fig. 3.1. In this work, we show that the variations in delay-difference due to aging can be modeled in terms of a ratio between two correlated Gaussian distributions and then, approximated as a zero mean Gaussian distribution with increasing variance. The arbiter delay, on the other hand, is modeled in the same way except the ratio distribution has a positive mean.

A prior aging work on linear PUFs [44] concluded that the effect of aging on PUFs is permanent and, therefore, the unstable (or unreliable) challenges need to discarded. In our work, we argue that, despite irreversible changes in the PUF structure due to aging, it can still be used for authentication by using suitable methods. One approach is to recalibrate the model parameters (i.e., the delay differences and the arbiter delay). The recalibration can be done by using the LMS method described in [5]. However, recalibrating hundreds of devices is not a feasible solution. Another approach for improving the authentication performance is by choosing appropriate thresholds on *total delay-difference* to discard challenges that are unreliable. We investigate these approaches in the upcoming sections.

3.2 Background and Aging Model

3.2.1 Aging model for the delay chain

A linear MUX PUF (shown in Fig. 2.1) has N multiplexer stages and an arbiter at the end. Control bits for each MUX stage are obtained from the input challenge. For MUX stage i, the delay-difference, $\Delta^i = D^i_{top} - D^i_{bottom}$, where D^i_{top} and D^i_{bottom} are the

delays associated with top and bottom multiplexers of the i^{th} stage. We will assume multiplexer delays, D^i_{top} and D^i_{bottom}, to be Gaussian distributed as $N(\mu, \sigma^2)$ [21] (this can be attributed to the manufacturing process variations which tend to be random in nature). Therefore, delay-difference of the i^{th} stage, Δ^i, will be distributed as $N(0, 2\sigma^2)$.

Due to aging, the multiplexer delays of each stage gradually increase, which corresponds to an increase of mean and variance to (μ',σ_a^2). Therefore, the new delay-difference of the i^{th} stage, Δ^i_{aged}, will be distributed as $N(0, 2\sigma_a^2)$ (where $\sigma_a > \sigma$). This is validated in prior aging work [39], where it is shown that the standard deviation of delay-difference increases with aging. The new delay-difference of the i^{th} stage for an aged PUF can be expressed as:

$$\Delta^i_{aged} = \Delta^i \left(1 + \frac{\Delta^i_{aged} - \Delta^i}{\Delta^i}\right) = \Delta^i \left(1 + p_i\right) \tag{3.1}$$

where p_i is the *percent delay-difference variation* in the i^{th} stage. p_i is basically the ratio of two correlated Gaussian distributions, $N(0, 2(\sigma^2 + \sigma_a^2 + 2\rho\sigma\sigma_a))$ and $N(0, 2\sigma^2)$ (where ρ is the correlation coefficient between Δ^i and Δ^i_{aged}). As the variance of Δ^i increases with aging, σ_a^2 term in the variance of p_i will start to dominate. Therefore, the variance of these approximate distributions is to an extent proportional to σ_a^2. Prior work in [45] suggests some approximate distributions for correlated ratio distributions in terms of Gaussian, t-distribution etc. Also, in [46] a ratio of two Gaussians has been used to model true random number generators.

Using aging data collected from test chips for upto 10 hours, we observe that the ratio distributions have a good fit with *t-distribution*. Fig. 3.2 shows ratio distributions for recovery times of 2 and 6 hours using a voltage based stress test, where the voltage was increased from a nominal value of 0.9V to 1.8V at 25°C. However, the data collected was only for 3 chips and corresponds to 3x32=96 samples for 32-stage MUX PUFs. This is insufficient in order to obtain a precise distribution of p_i.

For our model, we adopt a Gaussian approximation for the ratio distribution (as t-distribution is a good approximation for Gaussian when dealing with small sample sizes). An example is ratio distribution, p_i, with standard deviation=0.05 (or 5%). In prior work [39], a 5% standard deviation in delay-difference roughly corresponds to 2 years of aging.

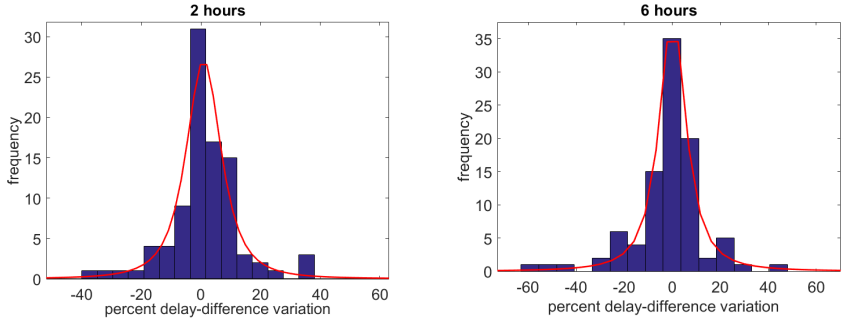


Fig. 3.2: Ratio distribution with *t-distribution* fit for data collected from 3 chips for 2 hours (left) and 6 hours (right) of aging. X axis is the percent delay-difference variation.

By using the model in (3.1) with a set of initial Δ^i (sampled from a Gaussian distribution with standard deviation, $\sqrt{2}\sigma$) and p_i (sampled from a ratio distribution with a given standard deviation), we can generate various instances of aged delay-difference, Δ^i_{aged}, for a MUX stage i. Note that σ is obtained from the values of Δ^i estimated from the test chip using LMS in [5].

3.2.2 Aging model for arbiter

The arbiter at the end of the delay chain, is modeled in a slightly different manner. We model it in terms of its delay, Δ^{arb}, which accounts for its propagation delay. However, being a delay element, it can only take positive values unlike delay-difference and, therefore, always has a positive mean. An aged Δ^{arb} can be expressed similar to (3.1) as:

$$\Delta^{arb}_{aged} = \Delta^{arb}\,(1+q) \tag{3.2}$$

where q is the percent change in arbiter delay. The only difference is that q is Gaussian distributed with a positive mean. Similar to the delay-difference, the mean and variance of q increase with aging. However for simplicity, we assume that mean and variance are related as $\mu_q^2{=}3\sigma_q^2$ (assuming q to be a uniform random variable between 0 and $2\mu_q$). This means the mean, μ_q varies more rapidly than the standard deviation, σ_q.

We now summarize the assumptions made for the proposed aging model:

- For a given environmental condition, the effect of environmental noise on the PUF performance is a constant. That is, over time the value of σ_n can be assumed to be constant. This is true even when the PUF is undergoing gradual change due

to the aging process.

- Variance of the delay parameters, for instance, delay-difference, $2\sigma^2$ and arbiter delay, σ^2_{arb}, increase gradually with aging. The model considers them in the form of variance of percentage distributions, p_i and q.
- The variance of the percentage change in arbiter delay, q, increases much more rapidly than that of delay-difference, p_i. This means for a given amount of aging, we can assume $\sigma_{arb} > \sigma$.

3.3 Aging Data

A lot of the aging based simulation methods discussed in literature are based on the idea of accelerated aging [47, 48]. The effects of aging on a hardware like PUF can be accelerated using various stress tests like voltage, temperature. Specifically for our lab chips, we use voltage induced stress test for aging related measurements. In this test, the VDD voltage is increased from a nominal value of 0.9 to 1.8 V for a specific duration of time that corresponds to the amount of aging. Data is collected for up to 10 hours of voltage stress from six 32-bit MUX-PUFs. As discussed in the previous chapter, the PUF outputs are based on their soft-response.

3.3.1 Monte-Carlo Simulation

For a given amount of aging, we generate 1000 PUF instances using Monte-Carlo simulation with percentage delay paramaters, p_i and q, sampled from Gaussian distributions with certain standard deviation. The (initial) delay-differences of individual MUX stage, Δ^i, and arbiter delay, Δ^{arb}, are taken from the model obtained from the actual chip [5]. Gaussian noise with a fixed standard deviation, σ_n, is added to the linear delay model. To simulate aging over a period of time, standard deviations of p_i and q are varied while keeping the standard deviation of noise fixed.

3.4 Authentication Performance with Noise

The model in (2.3) includes the effect of environmental noise to the final response bit. For a given environmental condition (like fixed temperature, voltage supply etc), the

noise is modeled as a Gaussian distribution with fixed variance. Unlike aging, noise is static and, therefore, the degradation in PUF performance is fixed. Fig. 2.5 shows the probability distribution of r_N for stable 0 and stable 1 response bits in the presence of noise with standard deviation=5% std. of Δ^i.

We can quantify the overlap between two distributions in terms of metrics like Jensen-Shannon (JS) [49] or Henze-Penrose (HP) divergence [50]. Jenson-Shannon (or JS) divergence is a symmetric form of Kullback-leibler (or KL) divergence [51]. JS divergence between two distributions P and Q is defined as:

$$JS(P||Q) = \frac{1}{2}(KL(P||R) + KL(Q||R)), \quad where\ R = \frac{1}{2}(P+Q) \tag{3.3}$$

where *KL(.)* corresponds to KL divergence. We found KL divergence to be sensitive to probabilities close to 0 which deemed it unsuitable. HP divergence is computed by randomly choosing total delay-difference values, r_N, from a sorted set of r_N. The set of r_N should have equal number of stable 0 and stable 1 CRPs. HP divergence is equal to $1 - \frac{R}{N}$, where R is the number of differing classifications in the chosen set of size N. JS divergence takes values between 0 to 1, whereas HP divergence takes values between 0.5 to 1.

For a good PUF reliability, we desire a higher divergence between the stable 0 and stable 1 probability distributions. In Fig. 3.3 (**dashed lines**), we show how the two metrics vary for different amounts of noise. The x-axis is the standard deviation of noise (σ_n) represented as a percentage of the standard deviation ($\sqrt{2}\sigma$) of delay-difference, Δ^i. We observe that as the amount of noise increases, the overlap between the distributions also increases and, therefore, the divergence value decreases. Out of the three PUF configurations, feed-forward is the most affected by noise.

3.5 Authentication Performance with Aging

As mentioned before, we assume that the model parameters, i.e., delay-differences and the arbiter delay, have been estimated and are stored in the server database. These stored parameters are estimated for an un-aged PUF instance. But with time as the

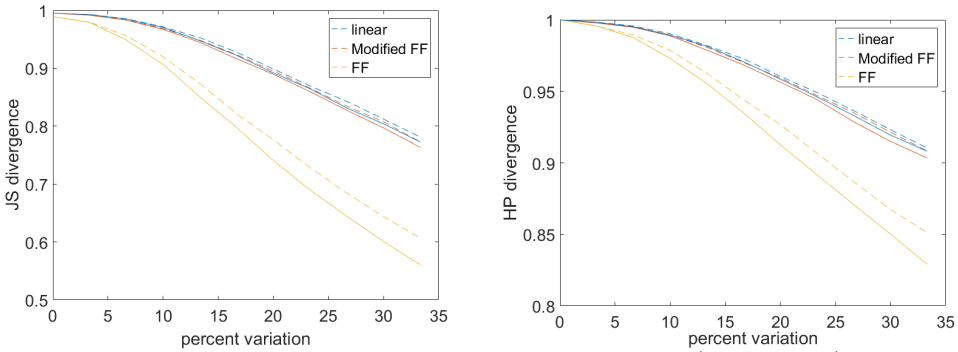


Fig. 3.3: Divergence metric comparisons in case of noise alone (dashed line) and aging alone (solid line) scenarios with (left) Jensen-Shannon divergence (right) Henze-Penrose divergence.

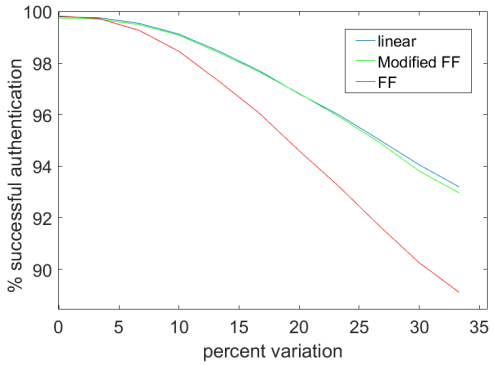


Fig. 3.4: Percentage of successful authentications by assuming equal variance for the delay chain and arbiter due to aging and in presence of 5% std of noise.

PUF starts to age, the delay-differences start to vary gradually and therefore, these stored parameters (or even a CRP look-up table or other adaptive parameters) become outdated. The result is a decrease in the percentage of successful authentications, as shown in Fig. 3.4. An authentication is considered *successful* if the responses to the challenges match with their expected values (which is obtained from the stored model parameters or from a CRP look-up table).

The percentage of successful authentication at 0% standard deviation, i.e., for an unaged PUF, depends upon the amount of environmental noise added to the model. The authentication accuracy is close to 100%. Fig. 3.4 shows the percentage of successful authentications under equal aging scenario for the delay chain and arbiter, i.e., under the assumption of equal variation for both. Note that unless otherwise mentioned, the standard deviations for percentage variation, p_i and q will be assumed to be equal for

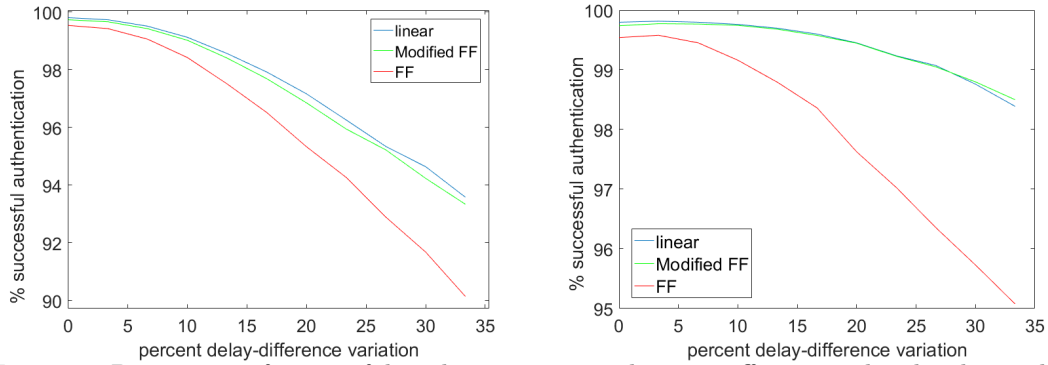


Fig. 3.5: Percentage of successful authentications with aging effect considered only on delay chain (left) and only arbiter (right) in presence of 5% noise std.

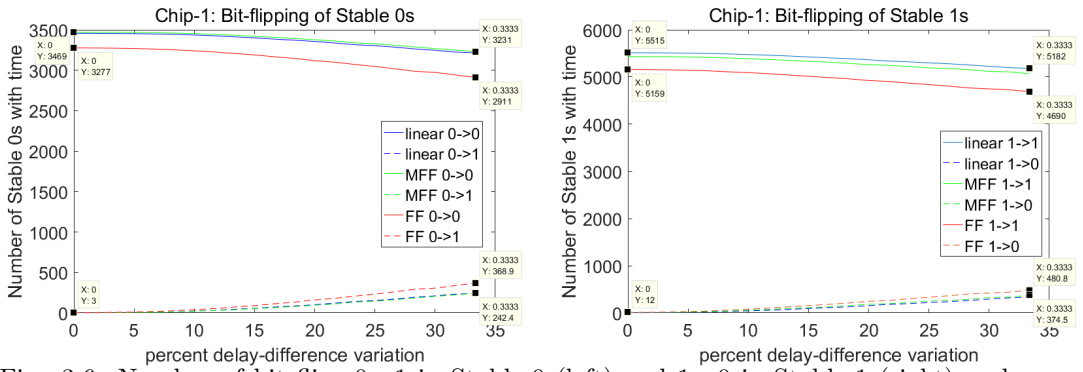


Fig. 3.6: Number of bit flips 0→1 in Stable-0 (left) and 1→0 in Stable-1 (right) under equal aging scenario.

a given amount of aging. However, as discussed before, we expect a higher variance for arbiter than the delay chain. Fig. 3.5 shows the authentication accuracies for aging effects considered separately on the delay chain and arbiter. We can observe that the performance degradation due to aging in arbiter is significant.

Tables 6.1, 6.3 show the percentage of successful authentications under equal and unequal aging conditions of the arbiter and delay-difference. We also compare the standalone effects of aging and noise on the performance of MUX PUFs. From Table 6.1, we observe that under equal aging condition, both noise and aging affect the authentication accuracy in a similar manner. That is, the performance is only slightly degraded in the case of aging alone. From Table 6.3, we observe that under the assumption that the variation in arbiter delay, q, is considerably more (i.e., 20%) than in delay-difference, p_i, the performance degradation in the case of aging alone is prominent than noise alone.

Table 3.1: Percentage Successful Authentication under equal aging scenario; $STD(q) = STD(p_i)$

% STD		No Noise	Noise STD=5%	Noise STD=10%	Noise STD=20%
Linear	**Original**	0.9993	0.9980	0.9930	0.9729
	5%	0.9981	0.9967	0.9917	0.9714
	10%	0.9927	0.9911	0.9860	0.9674
	20%	0.9697	0.9683	0.9639	0.9487
MFF	**Original**	0.9985	0.9974	0.9921	0.9710
	5%	0.9977	0.9963	0.9911	0.9698
	10%	0.9923	0.9906	0.9854	0.9661
	20%	0.9690	0.9675	0.9629	0.9486
FF	**Original**	0.9982	0.9954	0.9863	0.9528
	5%	0.9955	0.9927	0.9837	0.9523
	10%	0.9842	0.9817	0.9728	0.9450
	20%	0.9463	0.9442	0.9387	0.9187

Table 3.2: Percentage Successful Authentication under unequal aging scenario; $STD(q) = STD(p_i)$+20%

$\%STD(p,q)$		No Noise	Noise STD=5%	Noise STD=10%	Noise STD=20%
Linear	(0,20)	0.9961	0.9941	0.9892	0.9704
	(5,25)	0.9914	0.9898	0.9843	0.9657
	(10,30)	0.9825	0.9805	0.9761	0.9594
	(20,40)	0.9568	0.9556	0.9526	0.9409
MFF	(0,20)	0.9960	0.9944	0.9891	0.9694
	(5,25)	0.9912	0.9896	0.9848	0.9663
	(10,30)	0.9827	0.9815	0.9761	0.9592
	(20,40)	0.9566	0.9561	0.9528	0.9391
FF	(0,20)	0.9795	0.9773	0.9700	0.9441
	(5,25)	0.9672	0.9654	0.9591	0.9354
	(10,30)	0.9506	0.9494	0.9433	0.9248
	(20,40)	0.9136	0.9124	0.9096	0.8937

Fig. 3.6 shows the number of bit flips for Stable-0 and Stable-1 response bits with aging. The top curves (solid line) in each figure show a decrease in the number of stable-0s (or stable-1s) with aging. The bottom curves (dashed line) show an increase in the number of flipped bits corresponding to stable-0s (or stable-1s). For a fixed level of aging, we observe that the number of bit-flips is the highest for the feed-forward configuration. For example, for 33% standard deviation of delay-difference variation, the percentage of Stable-0 bit-flips for linear/modified feed-forward is roughly $\frac{242}{3469}$=6.97% and for feed-forward is $\frac{369}{3277}$=11.3%. Similarly, for a 33% standard deviation of percent delay-difference variation, the percentage of Stable-1 bit-flips for linear/modified feed-forward is roughly $\frac{374}{5515}$=6.78% and for feed-forward is $\frac{481}{5159}$=9.32%. We can observe that the number of bit-flips Stable-0→1 is more than Stable-1→0. This is evident from

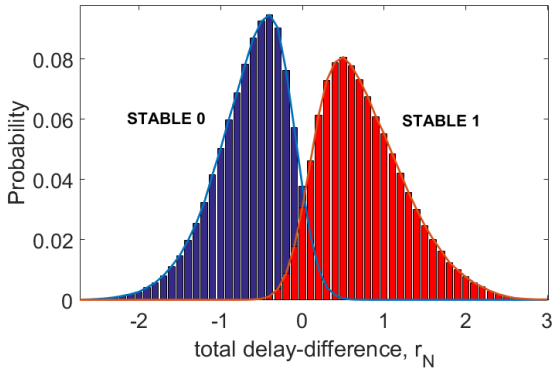


Fig. 3.7: Probability distributions of stable 0 and stable 1 response bits with aging with 30% delay variation for a linear PUF

the fact that the mean of total delay-difference, r_N, (i.e., μ_{arb}) increases with aging. Also, note that for an unaged PUF instance (i.e., 0% variation), the response bits are slightly skewed towards bit '1'. This is because the overall mean of the distribution in Fig. 6.2 is non-zero ($= \mu_{arb}$).

We observed that with aging, a feed-forward configuration is more prone to bit flips and therefore, has more authentication failures than the other two configurations. This is because any bit-flip in the intermediate response bit (or the internal challenge bit) affects the final response bit much more significantly than in the case of modified feed-forward or linear configuration. In case of modified feed-forward, the interconnection between consecutive stages (almost) negates the effect of internal challenge bits on the final response.

Similar to the case of noise, the overlap between the stable-0 and stable-1 distributions would increase with aging (as shown in Fig. 3.7). Fig. 3.3 (**solid lines**) shows how both the divergences (JSD and HPD) change with aging. We observe that both aging and noise affect the authentication performance in a similar manner. Even so, for the same amount of percent variation, the performance is slightly worse for aging. The difference is mainly due to the way the arbiter ages with time.

3.6 Temporal Properties of unstable CRPs

In this section, we discuss about how aging affects the (initial) unstable CRPs. Note that these CRPs are unstable in an un-aged PUF due to the effects of metastability or environmental noise. Here, we are interested in studying the variations in the total delay-difference, r_N, of these CRPs due to aging. Fig. 3.8 shows how the variance of r_N for unstable challenges varies with time. As observed, the variance for all the three PUF configurations increases with aging. Also, we had observed that stable challenges possess higher values of total delay-difference, r_N. Hence, it is logical to think that some of these unstable challenges would become stable. To validate this, we choose a threshold for r_N (say, α=0.4σ, where σ is the standard deviation of total delay-difference, r_N, for *un-aged* PUF shown in Fig. 6.2). We then observe how the percentage of unstable challenges with $|r_N|>\alpha$ varies with aging. This is shown in Fig. 3.8.

We can observe that the percentage of unstable challenges with $|r_N|>\alpha$ increases with time (or aging). For linear and modified feed-forward configurations, the percentage remains near 0 till about 7% standard deviation of delay-difference variation (p_i or q). This means that, in the initial periods of aging, there is hardly any increase in the number of unstable challenges with $|r_N|>\alpha$. This is due to the choice of α (=0.4σ), which essentially guarantees the stability of these CRPs. Feed-forward configuration, on the other hand, has higher ($\sim$25%) percentage of unstable challenges with $|r_N|>\alpha$, even in the un-aged case. This indicates that for feed-forward, α=0.4σ is not an optimal value of r_N for guaranteeing stability. We choose a value of α=2.4σ empirically for this case.

Furthermore, from Fig. 3.8 we can say that for linear and modified feed-forward configurations, about 25% of (initially) unstable CRPs have become stable for a 33% standard deviation of percent delay-difference. Note that 25% of unstable CRPs is roughly equal to $\frac{25}{100}$x10%$\approx$2.5% of total CRPs. In case of feed-forward, such claims can only be made at higher values of α like 2.4σ. However at this threshold, we observe that only a very small percent (<0.1%) of unstable CRPs become stable. Nonetheless for the three configurations, unstable CRPs with $|r_N|>\alpha$ can be re-used for authentication in an aged PUF.

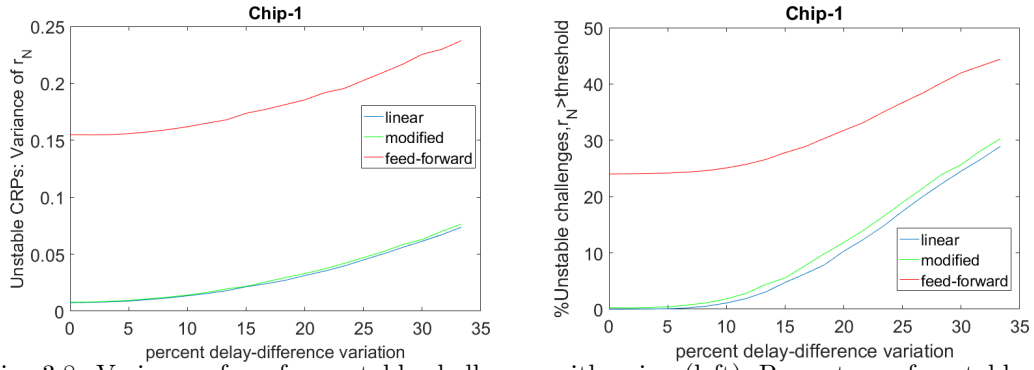


Fig. 3.8: Variance of r_N for unstable challenges with aging (left), Percentage of unstable challenges with $|r_N|>\alpha$, where $\alpha=0.4\sigma$ (right).

3.7 Recalibration and Threshold Tuning

Previously in Fig. 3.4, we had discussed about the authentication failures occurring due to aging. This happens when the (aged) delay-differences start to vary from the ones stored in the server database. A simple solution for this problem is to re-estimate the new delay-differences using the LMS technique described in [5]. For the LMS estimation, a sufficient number of CRPs (<2000) will give us a prediction accuracy close to 100% in case of linear MUX PUFs. Note that other more complex MUX PUF configurations can be reconfigured as a linear configuration by the use of fuses. However, a disadvantage of the approach is that with the increasing number of PUF devices in the market, the number of devices needing recalibration will be very high. Therefore, the approach will prove to be costly and is not feasible in practise.

We propose an alternative approach to improve the reliability by selecting appropriate thresholds (say, β) on the total delay-difference, r_N, values of an un-aged PUF. A desired value of β will correspond to the total delay-difference, r_N, of CRPs that are immune to aging related bit-flips.

Fig. 3.9 shows the percentage of error (or failure) in authentications against varying threshold, β. Note that, *100-(% error)* is equal to the percentage of successful authentications. From the plot, it is easy to observe that CRPs with a higher total delay-difference, r_N, value are more immune to bit-flips. Therefore, a logical choice for optimal β is to choose it as high as possible. For example, for linear configuration β=0.3 is a "good" choice of threshold for up to a standard deviation of p_i=q=16.7% (equal

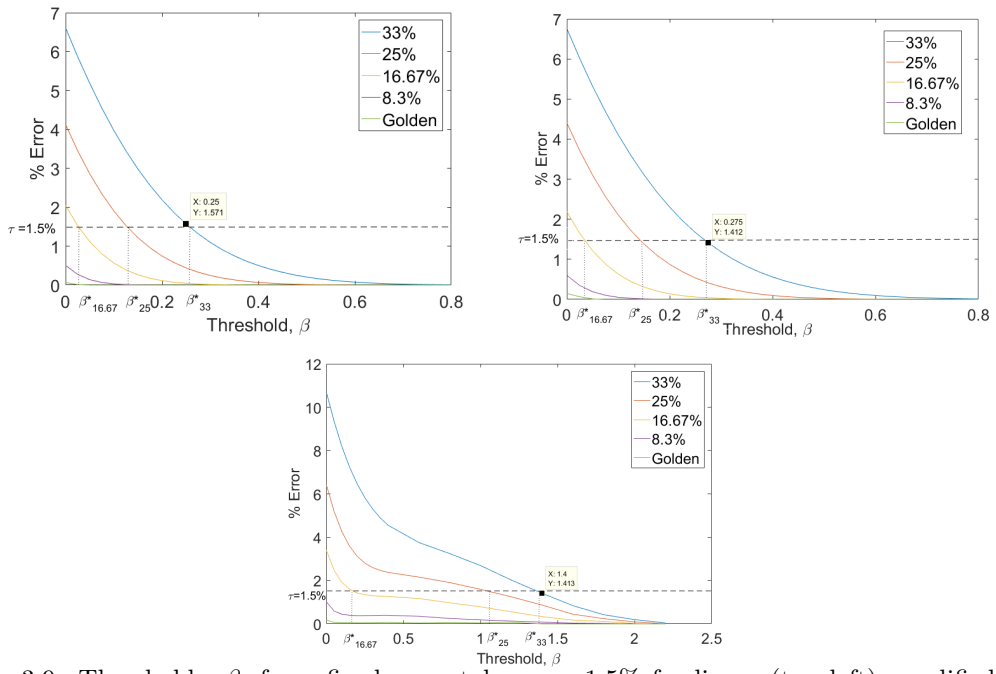


Fig. 3.9: Thresholds, β, for a fixed error tolerance=1.5% for linear (top left), modified feed-forward (top right), and feed-forward (bottom).

aging scenario). It corresponds to low bit-flips ($\sim$0.02%) for 16.7% standard deviation of p_i and q. In previous discussion on unstable CRPs, we observed that α=$0.4\sigma\approx$0.3 (σ=0.75 is the standard deviation of r_N for our chip) guaranteed the stability of CRPs for up to about 15% standard deviation of p_i and q. That is, the same value of threshold, β=α=0.3, serves two different purposes: First, for guaranteeing the stability of CRPs (specified by α) and second, to improve the unreliability caused due to the aging effects (specified by β). For both cases, CRPs corresponding to total delay-difference, $|r_N|>\alpha$ or β are the same and can be termed as *highly stable CRPs*.

In the case of modified feed-forward, the performance is quite similar to that of linear configuration, which is expected. Therefore, the thresholds, α and β ($=0.4\sigma$) are the same. For the case of feed-forward, we observe that the threshold value, β, required for good reliability is much higher. In this case, β=$2.4\sigma\approx$1.8=α (for up to 33% standard deviation of p_i) is a desirable choice. In general for a 33% standard deviation of p_i and q, threshold values of 0.8σ, 0.8σ and 2.4σ are good choices for linear, modified feed-forward and feed-forward configurations, respectively.

The thresholds, β, chosen previously correspond to the cases when the tolerance to error in authentication is very low (close to 0%). However in practical authentication scenarios, we can tolerate a certain amount of error in the responses. For example, prior work in [7] considers a tolerance of 10 bits for a 128-bit response. Fig. 3.9 shows thresholds, β, corresponding to a tolerance level, τ, of about 1.5%. This corresponds to a succesful authentication of 98.5%. As can be observed for linear configuration, threshold $\beta^*_{16.67}$ is much less than $0.4\sigma = 0.3$ obtained for the low tolerance scenario. This is true for the other configurations as well. Hence, in a practical authentication scenario, thresholds lower than 0.8σ, 0.8σ and 2.4σ are good enough for maintaining the required level of reliability for the three configurations. Furthermore, the threshold, β, *vs* percentage successful authentication (or % error) curve in Fig. 3.9 can be learnt using a polynomial fit of order 3 or more.

3.8 Discussion

We observed from our results that the reliability (or intra-chip variation) of a PUF decreases with aging. The effect of aging on other metrics can be summarized below:

- *Uniqueness*: Also called inter-chip variation, is the ability of a PUF to produce outputs that are significantly different from other PUFs. From our simulations of synthesized PUFs, we observe that for at least 70% permutation of the chips, uniqueness increases with aging. This is intuitive because of the fact that aging is a random process due to its Gaussian nature.
- *Randomness*: It is the ability of a PUF to produce unbiased '0' and '1' response bits. From Fig. 3.6, we observed that even for an un-aged PUF, the response is slightly skewed towards bit '1' due to positive arbiter delay. Furthermore, with aging as observed the number of bit flips from Stable-0→1 is much higher than Stable-1→0. This further reduces the randomness present in the PUF output.

For countering the unreliability in the stable CRPs due to aging, we suggested an approach to tune a threshold, β, based on the total delay-difference, r_N. An un-aged PUF has about 85-90% stable CRPs (Fig. 6.2). That is, for a 32-bit un-aged PUF, we have more than 2^{31} stable CRPs. However with aging, the number of stable CRPs

decreases (Fig. 3.6). With the threshold tuning method, the number of stable CRPs depends on the choice of threshold, β. For values of β equal to 0.8σ, 0.8σ and 2.4σ for linear, modified feed-forward and feed-forward configurations, we get about 42%, 42% and 2% stable CRPs, respectively. This corresponds to about 2^{31}, 2^{31} and 2^{26} number of stable CRPs. For a practical authentication scenario, these numbers are still considered significant.

3.9 Conclusion

This work discusses the impact of aging on linear and non-linear MUX PUF configurations. We observe that certain structures (like feed-forward) are much more significantly affected by aging than the others. We also observed that the arbiter can largely dictate how a MUX PUF performs with time. Also, under equal aging scenario of delay chain and arbiter, the effects of aging are similar to that of noise. However, under unequal aging scenario, the degradation due to aging is much more significant than noise. Approaches to improve the reliability by estimating new model parameters or by tuning a threshold based on the total delay-difference were discussed. The threshold depends on the amount of error that can be tolerated in the authentication scheme. It was further observed that the authentication accuracy of feed-forward PUFs is degraded by 3% if the number of MUX stages increases from 32 to 64 under equal aging scenario. Furthermore, the effect of arbiter aging due to asymmetry is lessened as the number of MUX stages increases.

Chapter 4

Entropy based analysis of MUX PUF

4.1 Introduction

In this work, we consider the same three MUX PUF configurations: linear, feed-forward and modified feed-forward as shown in Figs. 2.1, 2.2. We discussed that a simple MUX PUF is *linear* in nature because its response can be computed using a linear delay model. They act as the backbone for more complex configurations such as feed-forward and modified feed-forward. The feed-forward and modified feed-forward configurations (shown in Fig. 2.2) have intermediate arbiter(s) making the structure *nonlinear* in nature. The external challenge bits corresponding to the MUXes controlled by the output of internal arbiters are unused. For example, in the modified feed-forward configuration, the external challenge bits at stages N_2 and N_2+1 are unused. There exist more complex feed-forward based structures like overlap, cascade and separate as shown in Fig. 4.1 that contain different arrangements of internal arbiters [11].

In this work [19], we propose a novel approach based on Shannon entropy to determine whether a given MUX PUF is linear or not. The approach exploits the fact that a nonlinear MUX PUF has at least one internally generated challenge bit. This observation is valid for logic-reconfigurable MUX PUFs like feed-forward and MUX/DeMUX topologies [11]. The approach can also be extended to other types of non-linear MUX PUFs like XOR-arbiter or lightweight PUFs [7, 52, 53]. The analysis described in this

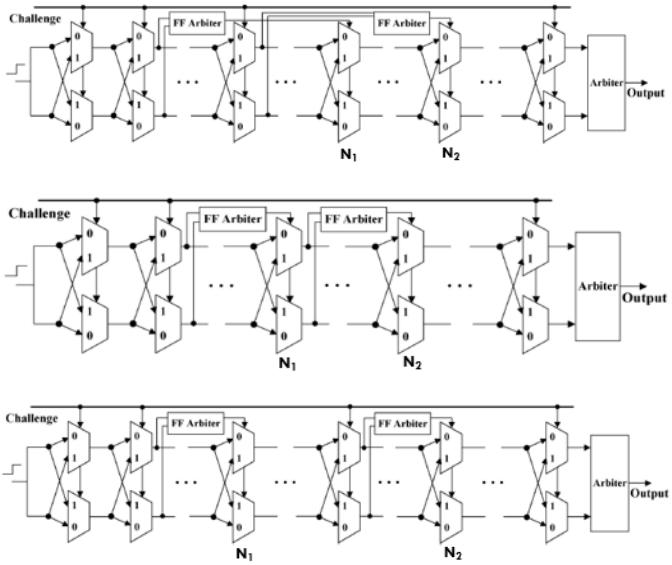


Fig. 4.1: Variants of feed-forward structures: (top) overlap, (middle) cascade, (bottom) separate

work, however, focuses on feed-forward based configurations.

In [31, 54], entropy based metrics for evaluating the unpredictability of a PUF was discussed. The method requires a tester to have access to PUF responses of certain predefined challenges. The challenge set consists of a set of randomly chosen challenges along with their k-bit neighbors. Conditional entropy is then computed by considering the response to a random challenge and challenges that are its k-bit neighbors. Our work adopts a modified version of the approach for determining non-linearity in a MUX PUF. We compute the conditional entropy of response with respect to its 1-bit neighbors that correspond to each stage of the MUX PUF. The entropy measure in this case will indicate the amount of information conveyed by each MUX stage. This will then help us determine the stages that are controlled by the internally generated challenge bits. For example, we can determine the location of N_2 and N_2+1 in case of modified feed-forward configuration.

4.2 Unpredictability and Mutual Information

In the PUF literature, there are various evaluation metrics that are used to analyze the performance of PUFs. One such metric is *unpredictability* [55] which gives us an

indication about the security of a PUF. Given the knowledge of a set of CRPs, it measures the ease with which one can predict the response to a given challenge. Higher the unpredictability, more difficult it is to the predict the response.

One of the approaches used to measure unpredictability is by computing the entropy [31, 54] associated with CRPs. Min-entropy and Shannon entropy based metrics can be used for this purpose [56]. Min-entropy indicates how many bits in the PUF response are uniformly random, whereas Shannon entropy corresponds to an average measure of randomness. Therefore, to measure unpredictability for the overall CRP set, Shannon entropy is more suitable. More specifically, the approach used in this paper is based on *conditional* Shannon entropy.

We know that conditional entropy given by $H(X|Y)$ is another way of measuring *mutual information*, $I(X,Y)$, between two variables X and Y. They are related as $I(X,Y) = H(X) - H(X|Y)$. For our purposes, we will assume X and Y to be PUF responses to two different challenges. Mutual information, $I(X,Y)$, measures the amount of information the knowledge of response Y indicates about response X and vice versa. For a highly secure PUF, we desire $I(X,Y)$=0. This corresponds to the case when X and Y are independent of each other. For a MUX PUF, responses X and Y can take values as bit 0 or 1. That is, the cardinality of X and Y, $|X|$=$|Y|$ is 2. Therefore, $H(X|Y) \leq H(X) \leq log_2(|X|) = 1$.

4.3 Entropy based method

This section briefly describes the entropy based approach adopted from [31]. The aim is to compute $H(X|Y)$ using the responses collected for a predefined set of CRPs.

Let us assume that X corresponds to the response to challenge C_x and Y to C_y. Challenge C_x is chosen at random, whereas C_y is chosen such that the Hamming distance between them, $HD(C_x, C_y)$, is 1 bit. Each C_x will have N 1-bit neighbors C_y, where N is the number of bits in the challenge. Conditional entropy between two random variables X and Y is defined by:

$$H(X|Y) = -\sum_{(x \in X, y \in Y)} p(x,y)\ log_2\left(\frac{p(x,y)}{p(y)}\right) \tag{4.1}$$

For our purposes, X and Y are the PUF responses corresponding to (C_x,C_y) pair. For an N-bit MUX PUF, we follow the following procedure to compute the entropy values:

- Create the challenge space consisting of (C_x,C_y) pairs. This is done by first choosing a random set of challenges as C_x. Then, for each C_x, we select N challenges, C_y, that are 1-bit apart, i.e., $HD(C_x, C_y)$=1 bit.
- Obtain PUF responses X and Y corresponding to each (C_x,C_y) pair.
- Compute entropy as in (4.1) by considering only one of the N (C_x,C_y) pairs at a time. For instance, consider (C_x,C_y) pairs with 1-bit difference at the n^{th} stage, where n can be between 1 and N. That is for an N-stage PUF, we can compute N such conditional entropies, $H(X|Y)$. Each such $H(X|Y)$ can be termed as *bit-wise entropy* corresponding to the n^{th} stage.

Conditional entropy in (4.1) involves computation of joint probability $p(x, y)$ and marginal probability $p(y)$. For our specific case, they can be computed as follows:

- Conditional entropy corresponds to responses X and Y for challenges that are 1-bit apart at an n^{th} stage. Therefore, the summation in the computation of $H(X|Y)$ is over unique pairs of responses, $(X = x, Y = y)$. As x and y are 1-bit values, only four such unique combinations are possible.
- $p(x, y) = \frac{\#(x,y)}{T}$, where $\#(x, y)$ is the number of instances of (x, y) and T is the total number of random challenges, C_x.
- Similarly, $p(y)=\frac{\#y}{T}$, where $\#y$ is the number of instances of y.

4.4 Data Setup for Entropy analysis

We experimented the proposed approach on 9 PUFs that are fabricated across 2 chips. For the entropy-based analysis, we need to generate a new challenge space rather than use a set of random challenges. The generated challenge space consists of 1000 random challenges and their 1-bit neighbors. For a 32-bit MUX PUF (discussed in Section 2.5), a random challenge can have 32 1-bit neighbors. This makes the total size of challenge set as 1000×(1+32)=33,000 challenges.

As discussed in Section 2.5.1, we use a soft-response based chip in our lab. For the proposed approach, we use challenges (or CRPs) that are stable, i.e., whose soft-response values, R_s, are greater than 0.9 or less than 0.1. All the other challenges that are unstable are discarded. This is done so because the final response bit to these (unstable) challenges after 0.1-0.9 thresholding is unreliable and therefore, would result in an inaccurate computation of entropy.

4.5 Bit-wise Entropy results

Conditional Shannon entropy is computed as in (4.1) by using the CRPs collected for predefined set of challenges. We collect responses to 1000 random challenges and their 1-bit neighbors. For a 32-bit MUX PUF, each random challenge, C_x, has 32 neighbors, C_y, such that $HD(C_x, C_y)$=1. That is, during the computation of bit-wise or conditional entropy we will obtain 32 different values corresponding to the stages of MUX PUF. Figs. 4.2 and 4.3 show the bit-wise entropy plot for linear and non-linear MUX PUFs, respectively.

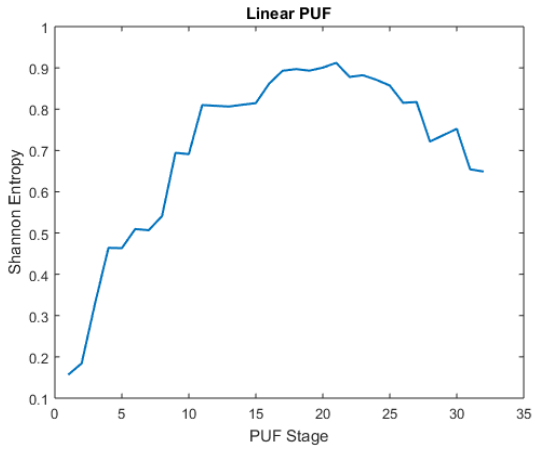


Fig. 4.2: Bit-wise entropy for linear or standard PUF with 1000 random stable challenges

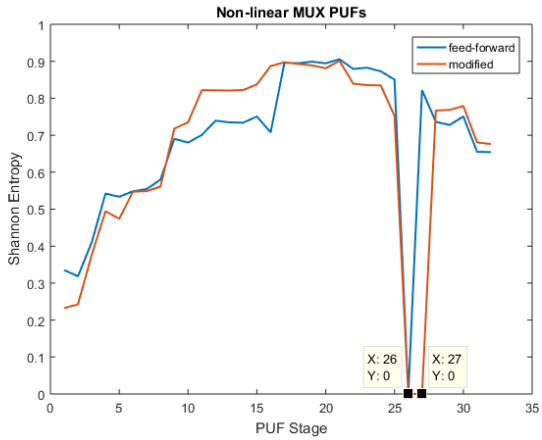


Fig. 4.3: Bit-wise entropy for feed-forward and modified feed-forward PUF configurations with 1000 random stable challenges

The entropy plots of non-linear MUX PUF have a distinctive feature. In the case of modified feed-forward configuration, we observe zero bit-wise entropy at stages N_2=26 and N_2+1=27 and for the feed-forward configuration, we observe zero entropy value at stage N_2=26. In both these cases, the zeros in the entropy plot correspond to the MUX stages being controlled by the internally generated challenge bits. Entropy $H(X|Y)$ being zero indicates that the mutual information, $I(X,Y)$, corresponding to these stages is very high, i.e., equal to 1. This is because $I(X,Y) = H(X) - 0 = log_2(2) = 1$. $H(X)$ takes the maximum value of 1, because the challenges, C_x, are selected at random and therefore, $p(X)\approx 0.5$.

A higher value of bit-wise entropy corresponds to a higher degree of randomness or security for those MUX stages. In other words, certain MUX stages contribute more towards the security of the PUF than others. In general, we observe a higher value of entropy corresponding to the middle stages of the PUF.

In Figs. 4.2 and 4.3, we have used 1000 random challenges for computing the conditional entropy. However in our experiments, we observed that for obtaining a 'good' bit-wise entropy curve, we do not actually need these many random challenges. In Fig. 4.4, we show that about 50 random challenges (and its 1-bit neighbors, therefore a total of 50*33=1650 instead of 1000*33=33,000 challenges) are sufficient to compute an accurate value of bit-wise entropy. The observations are consistent across different tested

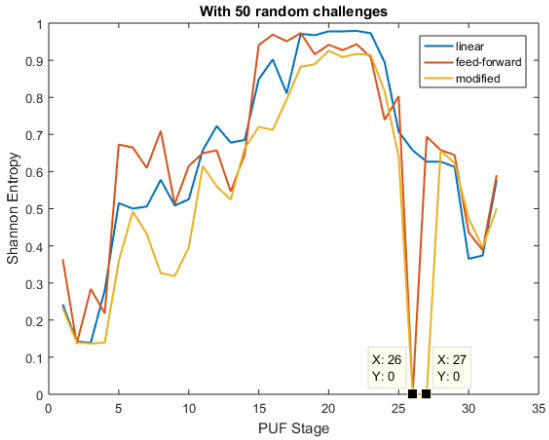


Fig. 4.4: Bit-wise entropy for linear, feed-forward and modified feed-forward PUF configurations with 50 random stable challenges

PUFs. Therefore, we claim a 100% sensitivity and 100% specificity for determining non-linearity in these PUFs.

In Fig. 4.5, we show bit-wise entropy curves obtained for feed-forward configurations with two arbiters, namely, overlap, cascade and separate feed-forward structures (shown in Fig. 4.1). These PUFs are synthesized in software using the additive linear delay model explained in Section 2.2. In the model, delay-difference values are sampled from the probability distribution functions (pdf) obtained from the chips [5]. These PUFs also contain 32 MUX stages, and are configured as follows: (i) Overlap - 1^{st} arbiter between stages 9 and 20, 2^{nd} arbiter between stages 16 and 25; (ii) Cascade - 1^{st} arbiter between stages 9 and 15, 2^{nd} arbiter between stages 16 and 25; (iii) Separate - 1^{st} arbiter between stages 9 and 15, 2^{nd} arbiter between stages 20 and 25.

4.6 Discussion

In previous sections, we have shown that by using the entropy based approach, one can determine whether a MUX PUF is linear or non-linear. The zero entropy values in the plot correspond to the MUX stage(s) or bit(s) that is unused in a challenge. That is, responses X and Y to challenge pair (C_x,C_y), C_y being a 1-bit neighbor to the random challenge C_x at a particular MUX stage, will be the same. $(X = x, Y = y)$ can take values as either (0,0) or (1,1) and therefore, $p(X = x|Y = y)$=1. This results in zero

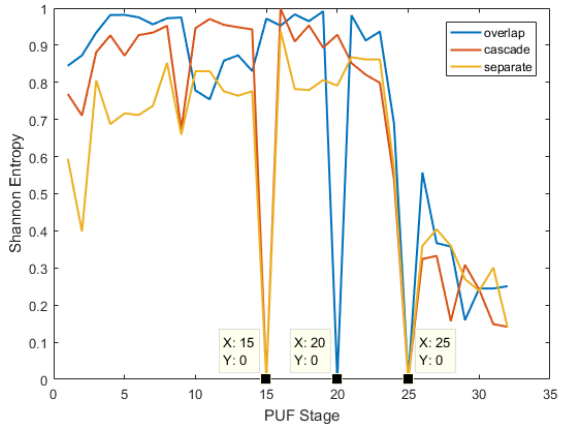


Fig. 4.5: Bit-wise entropy for synthesized feed-forward structures: overlap, cascade and separate with 50 random stable challenges.

entropy at these stages.

Another observation of interest is the number of zeros in the entropy plot. The number of zeros gives an upper bound on the number of internal arbiters present in a non-linear MUX PUF. As an example, if there are K zeros in the plot, one can say that the number of internal arbiters is anywhere between 1 and K. In a sense, this gives away crucial information about the internal structure of the PUF. Therefore, by using such an approach it is possible for an attacker to reverse engineer non-linear based MUX PUFs. Another observation of interest for an attacker might be that certain MUX stages (corresponding to ones with higher entropies) are more critical for predicting the responses. These stages are critical because they convey much less information about output than the others. Such an information theoretic approach is important for the hardware security analysis of various MUX based PUFs.

In this work, we have mainly focused on feed-forward based configurations. Nonetheless, the discussed approaches are also valid for other MUX PUF configurations like MUX/DeMUX or XOR-arbiter PUFs. For MUX/DeMUX PUFs, the approach will help determine the position of stages with select signal. For XOR-arbiter PUFs, the approach will help determine the MUX stages where challenge bits are internally generated across different PUFs. The entropy based approach can also be easily extended for analysis of more secure PUF configurations with higher bit-widths like 64 or 128 MUX stages. Such approaches have also found relevance in ring-oscillator (RO) PUFs [57].

4.7 Conclusion

In this work, we have proposed an entropy based test to determine whether a MUX PUF is linear or non-linear. Based on the analysis, we can obtain an upper bound on the number of internal arbiters present in a non-linear MUX PUF. The approach has significance in reverse engineering applications and in determining the security aspects of a PUF. From an information theoretic perspective, we can determine the MUX stages that contribute more to the security of a PUF.

Chapter 5

Predicting Soft-Response of MUX PUFs via Logistic Regression of Total Delay-Difference

5.1 Introduction

In previous chapters, we discussed about the linear delay model used to model MUX based PUFs. In Section 2.2, we used the model to compute the total delay-difference, r_N, of the delay chain consisting of N multiplexer stages and an arbiter at the end. The arbiter is modelled using its propagation delay and a sign response curve that decides the final response bit to be bit '0' or '1'. Furthermore in Section 2.2.2, we discussed how the PUF response can also be viewed as a soft-response instead of a hard-response. This work discusses models that can be used to compute soft-response to a given challenge for MUX PUFs.

In prior works [6, 20, 28], approaches based on artificial neural network (ANN) models have been proposed to predict the soft-responses from input challenges. Even though these models are able to classify and predict the responses with sufficiently high accuracy, they suffer from large computational complexity. In our proposed framework,

the model of the PUF is assumed to be stored in the server. The server then determines whether the challenge is stable or not by computing the soft response. However, computing thousands of ANN model parameters is not desirable due to significant area and power consumption. In this work [58], we propose a simple logistic regression based approach where the model parameters (which are the delay-differences of MUX stage and arbiter delay) and 3 other parameters are stored in the server. The logistic regression is used to learn the 3 parameters from chip data. We consider logistic function of the form:

$$F(x) = \frac{F_\infty}{1 + e^{-k(x-x_0)}} \tag{5.1}$$

where x is the independent variable, x_0 is the threshold, k is the flatness parameter and the dependent variable, $F(.)$, is the soft-response. In our model, x corresponds to total delay-difference, r_N and can be computed using the delay parameters of the MUX PUF. Learning the logistic (or sigmoid) function in (5.1) involves estimating parameters F_∞, k and x_0.

5.2 Response curve

In Section 2.2, we discussed about the sign response curve (2.2) used to obtain the final response bit, R. As per the response curve, the response bit is '1' if the total delay-difference, r_N, is positive and '0' if it is negative. Fig. 5.1 shows the *sign(.)* response function whose output is a *hard-response*.

In Section 2.2.2, we discussed about the effects of metastability and environmental noise. Due to environmental noise, the delay parameters of the multiplexers and arbiter vary as shown in (2.3). Furthermore due to metastability, the final response bits can vary on multiple measurements. The combined effect is represented in terms of a *soft-response*.

5.2.1 Sigmoid response curve

We discussed in previous chapters that the delay parameters like delay-difference of the multiplexers and the arbiter delay can be estimated using an LMS based method [5]. These estimated parameters are assumed to be stored in the server database. The value

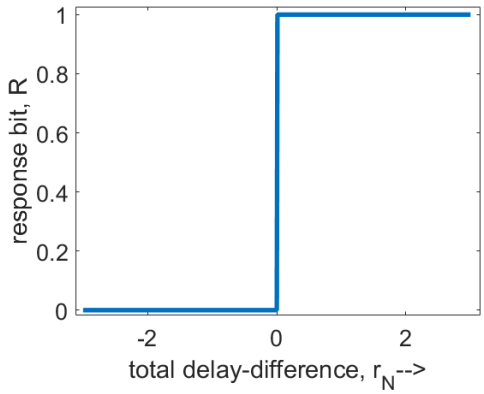


Fig. 5.1: Sign response curve for incorporating the output as hard-response

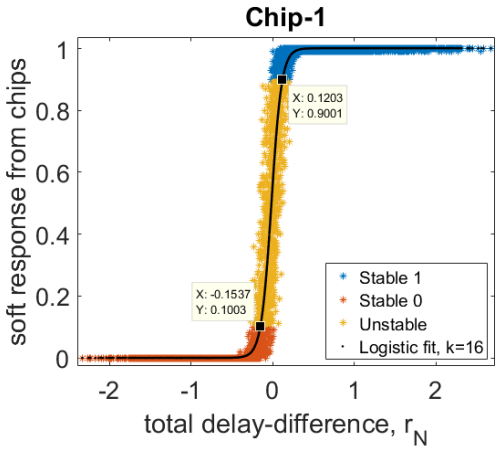


Fig. 5.2: Plot of soft-response, R_s and total delay-difference, r_N as obtained from silicon chip and fitting a sigmoid function (solid line) to the result with k=16.

of total delay-difference, r_N, can then be computed using (2.1).

Fig. 5.2 shows the plot of soft-response obtained from the chip for a linear MUX PUF against total delay-difference, r_N, computed in the server. Note that the computed value of total delay-difference, r_N, does not include the effects of noise and metastability. In the figure, CRPs that are stable-0, stable-1 and unstable according to the 0.1-0.9 threshold (from ground truth) are shown using different colors. The solid line in the plot shows the sigmoid or logistic fit to the chip data.

Fig. 5.3 shows the plot of soft-response for non-linear MUX PUF configurations. The estimated parameters of the sigmoid function for the three configurations are shown in Table 6.2. The table shows the result for PUFs in two different chips, denoted as

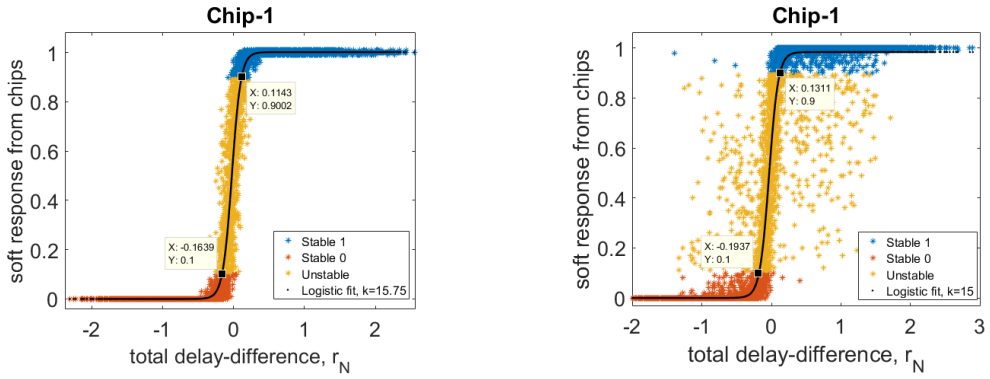


Fig. 5.3: Plot of soft-response, R_s, and total delay-difference, r_N, as obtained from silicon chip for (left) modified feed-forward and (right) feed-forward configurations. The sigmoid function fits lead to k=15.75 and 15, respectively.

Table 5.1: Logistic regression parameters for learning the response curves for different PUF configurations with 10,000 challenges

PUF configs		k	F_∞	x_0
Linear	PUF-1	15.997	1.001	-0.0165
	PUF-2	19.377	1.002	-0.0087
MFF	PUF-1	15.773	1.001	-0.0244
	PUF-2	19.160	1.000	-0.0186
FF	PUF-1	15.032	0.983	-0.0344
	PUF-2	18.921	0.988	-0.0257

PUF-1 and PUF-2. It is important to note that in case of feed-forward configuration, the value of k obtained by the fit is skewed by the noisy observations. The value of k decides the flatness of the response curve and is characteristic of the arbiter and noise variations in the MUX PUF. Note that if $|r_N|<\frac{1}{k}$, the soft-response takes value between 0.27-0.73. This corresponds to a set of unstable CRPs. For our purposes, however, we have chosen a much more stringent threshold of 0.1-0.9. Table 5.2 shows 95% confidence intervals for each of the estimated parameters k, F_∞ and x_0. We can observe that the parameters estimated for linear configuration have higher confidence compared to the other two configurations.

The spread (or noise) in Figs. 5.2 and 5.3 (also seen in Table 5.2) is related to how much effect the environmental noise or metastability have on the final response bit. In

Table 5.2: Confidence interval values for the three logistic regression parameters of the form $k \pm \mu(k)$, where $\mu(k)$ is the 95% confidence interval

PUF configs		$\mu(k)$	$\mu(F_\infty) \times 10^{-5}$	$\mu(x_0) \times 10^{-5}$
Linear	PUF-1	0.0104	7.187	4.722
	PUF-2	0.0142	8.002	4.319
MFF	PUF-1	0.0105	7.546	4.831
	PUF-2	0.0163	9.322	5.011
FF	PUF-1	0.0244	18.52	12.44
	PUF-2	0.0324	18.36	10.20

case of linear configuration, the spread is much less and can be seen in the form of a thick line around the logistic fit. However in the case of feed-forward configuration, the spread is much more significant due to an increased effect of metastability and noise on the final response bit [26]. Whenever the challenge bit at N_2 flips (Fig. 2.2), it may cause the final response bit to flip as well if r_N was already close to 0. However, in case of modified feed-forward, the interconnections in its structure, i.e., N_2 and $N_2 + 1$, almost nullify the effect of the intermediate stage. Therefore, its parameter values and their confidence intervals are similar to that of linear configuration.

Using the estimated parameters for the logistic function, we can now compute the predicted soft-responses. Table 5.3 shows the prediction accuracy of the logistic function. For a given challenge and stage delay-differences, we can obtain the total delay-difference, r_N, using (2.1). The soft-response value is then computed using the logistic function and compared against the threshold 0.1-0.9 for determining whether it is stable or not. The last column in the table shows the precision for the classification of CRPs as stable or unstable. Precision is the percentage of stable CRPs correctly predicted by the function. We will discuss this using two examples. In the case of linear PUF-1, the confusion matrix is as shown below:

		Predicted		
		Stable	Unstable	
Actual	Stable	8620	357	**8977**
	Unstable	114	909	
		8734		

Table 5.3: Comparison of number of stable CRPs obtained from the ground truth and prediction; and percentage of stable CRPs which were correctly predicted (precision) by the logistic function

PUF configs		**No. of Stable CRPs**	**Predicted**	
			No. of Stable CRPs	**% Stable**
Linear	PUF-1	8977	8734	98.69%
	PUF-2	9174	8980	98.59%
MFF	PUF-1	8917	8639	98.55%
	PUF-2	9129	8953	98.06%
FF	PUF-1	8451	8511	94.05%
	PUF-2	8832	8958	95.01%

We can observe that sensitivity = $\frac{8620}{8620+357}$=96%, specificity = $\frac{909}{909+114}$=89%, precision = $\frac{8620}{8620+114}$=98.69%. Sensitivity and specificity measure the proportion of stable and unstable CRPs, respectively, that are correctly identified as such. In case of feed-forward PUF-2, the confusion matrix is shown below:

		Predicted		
		Stable	Unstable	
Actual	Stable	8511	721	**8832**
	Unstable	447	321	
		8958		

We observe that sensitivity=96.4%, specificity=42%, precision=95.01%. We observe that the sensitivity and precision of feed-forward configuration are comparable to that of linear. However, specificity of feed-forward is significantly worse than that of linear configuration.

5.3 Discussion

In previous sections, we used logistic regression to learn the relation between soft-response, R_s, and total delay-difference, r_N. From the results in Table 6.2, we observe that two parameters of the logistic function, F_∞ and x_0 are almost equal to 1 and 0, respectively. Therefore, the response curve can be expressed simply in terms of the

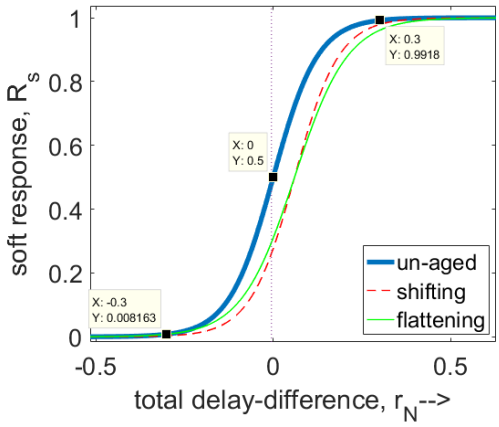


Fig. 5.4: The effect of change in arbiter delay parameters: change in Δ^{arb} shifts the response curve with k=1; change in T_{setup} or T_{hold} flattens the response curve

parameter k:

$$R_s = F(r_N) \approx \frac{1}{1 + e^{-kr_N}} \tag{5.2}$$

It seems possible to calculate the value of k from arbiter's timing parameters, T_{setup}, T_{hold} and noise variations in the PUF due to their inverse relation. This is because the response curve (characterized by k) essentially has the same functionality as cumulative distribution function (cdf) of a Gaussian random variable modelled by total delay-difference, r_N. This would reduce the training effort needed to learn the logistic function.

Another application of the proposed function for the response curve is to study the effect of aging in MUX PUFs discussed in Chapter 3. Due to aging, the hardware delay parameters like Δ^{arb}, T_{setup} and T_{hold} gradually increase. Any change in arbiter delay, Δ^{arb}, shifts the response curve to right (value of x_0 increases) and change in T_{setup} or T_{hold} flattens the response curve (value of k decreases). These effects are shown in Fig. 5.4.

5.4 Conclusion

This paper has demonstrated that soft response can be obtained from a hard PUF using logistic regression. The logistic regression function has been trained using the experimental chip data obtained from a soft PUF [28].

Chapter 6

Bit-Flipping Algorithm to convert Unstable to Stable Challenges

6.1 Introduction

This chapter considers the scenario where the server stores model parameters, as opposed to challenge-response pairs in a look-up table (LUT). These model parameters correspond to the delay-difference of MUX stages and arbiter delay. As discussed earlier, they can estimated by using adaptive learning techniques as in [5]. However, certain challenges do not always output the same response. That is, they have variations in their responses. These CRPs are referred to as *unstable*. Before the server issues a challenge for authentication, it needs to ensure that the chosen challenge is stable. To determine whether a challenge is stable or not, we make use of *total delay-difference* as a metric [41]. This metric can be computed based on the stored model parameters. The stability of a challenge is decided based on a threshold on total delay-difference. If the challenge is not stable, the server can then either discard and issue a new random challenge, or modify the unstable challenge to a stable one by flipping few bits. In this work, we propose a bit-flipping algorithm for the latter approach in case of three MUX PUF configurations: linear, feed-forward and modified feed-forward [11]. The algorithm determines the minimum number of bits that should be flipped, so that the new challenge is guaranteed to be stable. Furthermore, we compare the computational complexity of the proposed algorithm to the straightforward approach based on discarding unstable

challenges.

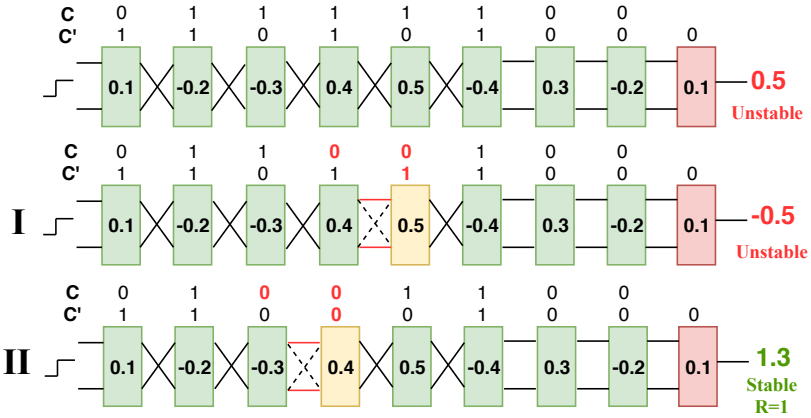


Fig. 6.1: Example showing two possible bit-flipping approaches for an 8-bit linear PUF. Threshold on magnitude of total delay-difference is **0.6**. **Red** color indicates an unstable response, and **green** indicates stable response. Approach II is proposed in this paper: 1 XOR bit-flip leads to 2 bit-flips in the final challenge.

The key contributions of this work include: an approach to transform an unstable challenge to a stable one by flipping only a few bits and analysis of computational complexities involved with the approach compared to the straightforward approach. Fig. 6.1 illustrates the effect of flipping a bit in XOR-ed challenge, and its effect on the total delay-difference and the stability of the response. While it may seem that flipping bit corresponding to *highest magnitude of delay-difference* is a good strategy, we show that this is not always the case. Such an example is shown in Approach I where the response is unstable. Approach II shows our proposed approach. In the example, the proposed approach selects bit-4 for flipping instead of bit-5 as in the case of Approach I. We will explore how these stages are selected in next sections. Furthermore, we also show that the proposed approach has *much better* worst-case and *slightly better* average-case performance in terms of the number of addition operations than the straightforward method.

6.2 Total Delay-Difference based thresholding

Fig. 6.2 shows the probability distribution of total delay-difference, r_N, obtained using the model in (2.3). Non-linear configurations are assumed to have N_1=16, and N_2=26

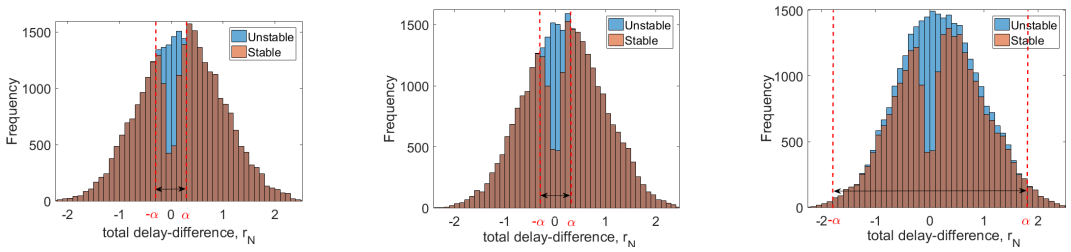


Fig. 6.2: Histogram of total delay-difference, r_N, obtained from synthesized models using 10,000 challenges for (left to right) linear, modified feed-forward and feed-forward configurations. Noise has std. equal to 10% std. of delay-difference, Δ^i.

for a 32-bit MUX PUF. The model parameters, Δ^i, used for computing r_N are sampled from Gaussian distributions with a variance of 0.02 [5]. The CRPs are classified as stable or unstable by using the model in (2.3). We assume the noise standard deviation to be 10% of standard deviation of delay-difference, Δ^i. We chose a threshold of 0.1-0.9 for the classification. This means that if the soft-response is greater than 0.9 or less than 0.1, the response is considered stable, otherwise it is considered to be unstable. From the distribution, we can observe that feed-forward configuration has a higher spread of unstable CRPs compared to the other two configurations. For the threshold 0.1-0.9, we observe that the percentage of stable CRPs for linear or modified feed-forward configuration is about 89-90% compared to about 85% for the feed-forward configuration. Therefore, about 10-15% of random challenges are unstable.

6.2.1 Authentication Scheme

In a typical PUF authentication scheme, there are two main phases: enrollment and authentication. In the enrollment phase, thousands of reference CRPs are measured and stored in the server. In the authentication scheme, the user is granted access if the chip responses to the chosen challenges match with those stored in the LUT (computed using a Hamming distance). However, due to variations in the response, all the responses might not match with the reference set [41]. Hence, we define an *error tolerance* or threshold corresponding to an acceptable amount of mismatch between the responses [7, 28]. An example is discussed in [7], where an error tolerance of 10 out of 128 bits is used. This corresponds to about 7.8% error in the responses. But as discussed previously, about 10-15% of random challenges can be unstable. Therefore,

the Hamming distance between the responses can be lower than that required for a successful authentication as the percentage of unstable challenges is higher than the error tolerance. This will result in a false negative scenario, where a legitimate user fails to be authenticated. Therefore, there is a necessity to ensure that the chosen set of challenges are stable.

The model parameter based authentication scheme requires computation of total delay-difference, r_N, as a metric instead of soft-response. The server, therefore, requires additional resources for storing the model parameters and for the computation of r_N metric described in (2.1). The computation of r_N is essentially $N + 1$ add operations. Furthermore, this scheme only requires storage for the $N+1$ model parameters compared to thousands of CRPs in the previous method. The model parameters can be stored in the server by using a fixed-point representation. The word-length for the representation will depend on the precision required for the computations. Nonetheless, the word-length will be much less than $N + 1$, as in the case of a CRP LUT.

6.2.2 Thresholding Total Delay-Difference

Just like soft-response (with threshold 0.1-0.9), total delay-difference, r_N, also needs to be thresholded for the classification of CRPs as stable or unstable. The threshold decides whether a challenge drawn at random by the server is likely to be stable or not. Fig. 6.2 shows how thresholds, α, can be empirically chosen. The aim is to choose a value of r_N that guarantees the stability of CRPs. A good example is a value greater than *three-sigma* of unstable CRPs. As shown in figure, we choose thresholds of 0.4σ, 0.4σ and 2.4σ for linear, modified feed-forward and feed-forward configurations, respectively [41]. Here, σ is the standard deviation of the overall total delay-difference, r_N. After thresholding, if the challenge turns out to be unstable, the server can then either draw a new challenge and test again, or attempt to flip few bits according to the bit-flipping algorithm described in later section.

6.3 Software model, Algorithm and Results

6.3.1 Software Model

We analyze our proposed methodology using 6 *synthesized* MUX PUFs. We consider the three PUF configurations, namely linear, feed-forward and modified feed-forward. The PUFs are assumed to have 32 MUX stages. The delay-difference of MUX stages and arbiter delay are sampled from Gaussian distributions [5]. PUF synthesis is done in software using the linear delay model described in (2.1), (2.2). In next sections, we will the discuss the straightforward and bit-flipping approaches.

6.3.2 Straightforward Approach: Discarding Unstable CRPs

For the three configurations, we discussed the thresholds, α, on r_N required to guarantee the stability of CRPs. That is, a CRP is considered stable if $|r_N| \geq \alpha$. An unstable challenge according to the threshold, α, is discarded and a new one is issued and tested for stability. However, it is possible, though unlikely, that a new issued challenge is always unstable. This is the *worst-case* scenario, where the number of computations required is *infinity*. The *best-case* scenario happens when the first issued challenge is stable. We will denote the number of computations required for the expression $|r_N| \geq \alpha$ as t. For a linear PUF, $t = N + 2$ add operations. This is because r_N requires $N + 1$ computations and the comparison can be considered as an add operation. Hence, the best case requires t operations. For the *average-case* analysis, we need to consider the *probability of a random challenge being stable*. We will denote this probability as p. As discussed earlier, p can take values between 0.85-0.9 depending on the PUF configuration. The number of computations can thus, be expressed as $pt + p(1-p)2t + p(1-p)^2 3t + ... = \frac{t}{p}$. For every new challenge, total delay-difference, r_N, needs to be computed and therefore, requires t add operations for each attempt. However, the number of attempts is not bounded and therefore, n can go up to infinity.

6.3.3 Bit-flipping Approach: Converting Unstable to Stable

In this section, we discuss the algorithm used to convert a likely unstable challenge to stable one. The idea is to increase the magnitude of total delay-difference, r_N, to be

Algorithm 1 Linear PUFs: Algorithm for mapping challenge, C, to its closest stable one, C_f

$\sigma \leftarrow$ standard deviation of r_N distribution
$\alpha \leftarrow 0.4\sigma$, threshold
$\Delta^i \leftarrow$ delay-difference per stage
$J \leftarrow$ Pre-computed set of size s based on the sign of total delay-difference, r_N
$C' \leftarrow [C'_1, C'_2, ..C'_i.., C'_N]$, input XOR challenge
$C_f \leftarrow [C_1, C_2, ..C_j.., C_N]$, new challenge

1: **if** $\sum_{i=1}^{N+1} |\Delta^i| \geq \alpha$ **then**
2: Compute r_N
3: Choose bits $\{j\} \in J$
4: **for** $1 : s$ **do**
5: **if** $|r_N| < \alpha$ **then** break;
6: flip bit C'_j
7: Compute r_N
8: $C_j \leftarrow C'_j \oplus C'_{j+1}$
9: **else**
EXIT. *Conversion not possible when the delay differences, Δ^i, are very small.*

greater than threshold, α. We achieve this by flipping minimal number of bits in the original challenge.

There are many possible solutions for the bit-flipping approach. Fig. 6.1 shows two such approaches. Approach I considers the bit-flip in the position corresponding to highest magnitude of delay-difference, Δ^i. A bit-flip at position i changes the value of previously computed total delay-difference, r_N, by $2\Delta^i$. In the example shown in Fig. 6.1, the bit-flip at the 5^{th} stage changes the previous value of r_N from 0.5 to -0.5. However, the value is still less than the threshold of 0.6, thus, deeming the new challenge as *unstable*.

We suggest an Approach II, where the bit-flips are allowed only at certain stages. Algorithm 1 shows the proposed approach for a linear PUF. Bits to be flipped are identified based on the sign of total delay-difference, r_N. Remember from (2.1) that r_N depends on the sum of factors, $r_N(i) = (-1)^{C'_i}\Delta^i$. That is, $r_N = \sum_{i=1}^{N} r_N(i) + \Delta^{arb}$. If the sign of r_N is positive, we choose set of bit locations i corresponding to negative values of $r_N(i)$ and vice versa. In addition, these bit locations are stored according to decreasing order of magnitude of delay-difference, $r_N(i)$. We will denote the set storing these bit locations as J which has a size of s. For example, set J for Fig. 6.1 contains entries $[4, 3, 8, 1]$ of size $s = 4$. The number of iterations in the algorithm is limited by the size, s, which usually averages $N/2$ for an N-bit MUX PUF. Therefore, s is also the maximum number of possible bit-flips in C'.

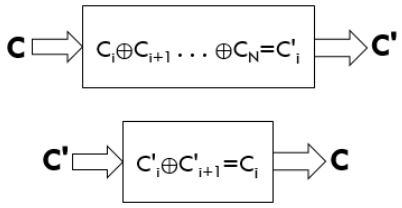


Fig. 6.3: Relation between C and C' for a linear configuration

Fig. 6.3 shows the relation between a challenge, C, and its XOR-ed challenge, C'. For linear MUX PUFs, there exists a one-to-one mapping between C and C' specified by the XOR operation. Note that we flip the bits in XOR-ed challenge, C', rather than C itself. *One* bit flip in C' will result in *two* bit-flips in C, as $C_i = C'_i \oplus C'_{i+1}$ and the XOR operation is commutative in nature.

In the proposed method, the computation of r_N in each iteration involves just 3 additions and 1 bit-shift operation. Shift operation is due to the factor $2\Delta^i$ that gets added to the previously computed value of r_N. 3 add operations are due to the bit-flip, addition of $2\Delta^i$ factor and comparison with threshold, α. In comparison, straightforward approach requires t add operations in each iteration because of a new random challenge.

In the unlikely scenario, the sum of the magnitudes of delay-difference does not exceed the threshold, i.e., $\sum_{i=1}^{N+1} |\Delta^i| < \alpha$, the bit-flipping logic would not help. This can only happen when a PUF has very small delay-differences, Δ^i, or the threshold, α, is too large. The *worst-case* scenario happens when either a stable challenge is found in the s^{th} (or last) attempt or when the bit-flipping logic fails as mentioned. The number of computations in this case is $t + 3s$ additions and s shift operations. Unlike the straightforward approach, the worst-case computation is not infinite. For the *best-case* scenario when stable challenge is found in the first attempt, the bit-flipping algorithm is not needed and therefore, the computation is just t add operations. However, the number of computations for *average-case* scenario is slightly different. After a bit-flip, the new challenge cannot be considered as random and is more probable to be stable, say, with a probability of p_1, where $p_1 > p$. For example, the average number of add operations for the second attempt is $p_1(1-p)(t+3)$. However, for our computations, we assume that $p_1 \approx p$ and therefore, can be expressed as $pt + p(1-p)(t+3) + p(1-p)^2(t+6) + ... + p(1-p)^s(t+3s) = t[1-(1-p)^{s+1}] + \frac{3(1-p)}{p}[1-(1-p)^s(1+ps)]$. We can

simplify this further by assuming nominal values of p and s. For a high value of p and $s \approx 16$ ($=N/2$), it is approx. equal to $t + \frac{3(1-p)}{p}$. The number of shift operations is given by $0 + p(1-p)1 + p(1-p)^2(2) + ... + p(1-p)^2 s = \frac{1-p}{p}[1-(1-p)^s(1+ps)] \approx \frac{1-p}{p}$. Table 6.1 shows the comparison of computational complexities between the two approaches.

Note that extra computations are required for computing XOR-ed challenge, C', from the original challenge, C, in (2.1) and to compute the new challenge, C_f, from flipped version of XOR-ed challenge, C'. For an N-bit PUF, one may implement the former by using the relation $C'_i = C'_{i+1} \oplus C_i$, which requires N additions. This conversion is required for both the approaches. However, the latter conversion from C' to C_f is only required for the bit-flipping approach. This can be done by using the relation $C_i = C'_i \oplus C'_{i+1}$. A flip in bit C'_i affects C_i and C_{i+1}. Therefore, the conversion needs two add operations for every bit-flip. This conversion from C' to C_f is not included inside the loop in algorithm because of one-to-one mapping between C and C'. However, we will discuss that this is not the case for non-linear PUF configurations.

The approach for bit-flipping in the case of feed-forward and modified feed-forward is similar to Algorithm 1. However, it is slightly more involved due to the presence of internal arbiter. Note that in the case of feed-forward, challenge bits at N_2 cannot be flipped as they are internally generated. Therefore, the set J *cannot* contain the stage N_2 as possible choice for bit-flipping. This means that due to the existence of loop from N_1 to N_2, XOR-ed challenge bits of C' are not only dependent on the challenge bits of C, but also dependent on the internal challenge bit at N_2 computed using the delay-differences from stages 1 to N_1. Depending on the location of bit-flip in C', it can also affect the computation of internal challenge bit and therefore, result in two bit-flips in C'. This requires the total delay-difference at stage N_2, r_{N_2}, to be computed inside the loop. Computation of r_{N_2} requires $N_1 + 1$ add operation. This further increases

Table 6.1: Comparison of computational complexities for *a challenge* between Straightforward and Bit-flipping approach.

Straightforward			**Bit-flipping**				
best	**average**	**worst**	**best**	**average**		**worst**	
				add	shift	add	shift
t	$\frac{t}{p}$	∞	t	$t + \frac{k_1(1-p)}{p}$	$\frac{k_2(1-p)}{p}$	$t + k_1 s$	$k_2 s$

Table 6.2: Values of k_1 and k_2 for computation complexities shown in Table 6.1 for the three configurations

Configuration	t	k_1	k_2
Linear	$N+2$	3	1
FF	N_1+N+3	$\sim N_1$	2
Modified FF	N_1+N+3	$\sim N_1$	3

Algorithm 2 Feed-forward PUFs: Algorithm for mapping challenge, C, to its closest stable one, C_f

$\sigma \leftarrow$ standard deviation of r_N distribution
$\alpha \leftarrow 2.4\sigma$, threshold
$\Delta^i \leftarrow$ delay-difference per stage
$J \leftarrow$ Pre-computed set of size s based on the sign of total delay-difference, r_N
$C' \leftarrow [C'_1, C'_2, ..C'_i.., C'_N]$, input XOR challenge
$C_f \leftarrow [C_1, C_2, ..C_j.., C_N]$, new challenge

1: **if** $\sum_{i=1}^{N+1} |\Delta^i| \geq \alpha$ **then**
2: Compute r_N
3: Choose bits $\{j\} \in J$
4: **for** $1 : s$ **do**
5: **if** $|r_N| < \alpha$ **then** break;
6: flip bit C'_j
7: Compute r_{N_2} (*If applicable*)
8: Compute r_N
9: $C_j \leftarrow C'_j \oplus C'_{j+1}$
10: **else**
EXIT. *Conversion not possible when the delay differences, Δ^i, are very small.*

the number of computations for each iteration. The exact number of computations depends on the position of the bit being flipped. However, we can generally express it in terms of N_1. The number of shift operations in each iteration can be either 1 or 2 in case of feed-forward, and 1 or 3 in case of modified feed-forward. We denote the a constant number of additions and shifts in each iteration as k_1 and k_2, respectively. Table 6.1 shows the number of computations required for a challenge. The value of constants k_1 and k_2 for the PUF configurations are shown in Table 6.2. The value of t for computing total delay-difference, r_N, and comparing against α is $N_1 + N + 3$ for both the configurations. Note that the empirically chosen threshold, α, is much higher ($=2.4\sigma$) in case of feed-forward configuration [41]. As discussed before, this is due to a larger spread in values of r_N for unstable challenges, as shown in Fig. 6.2.

We will discuss the bit-flipping approaches shown in Figs. 6.1 (Approach II) and 6.4.

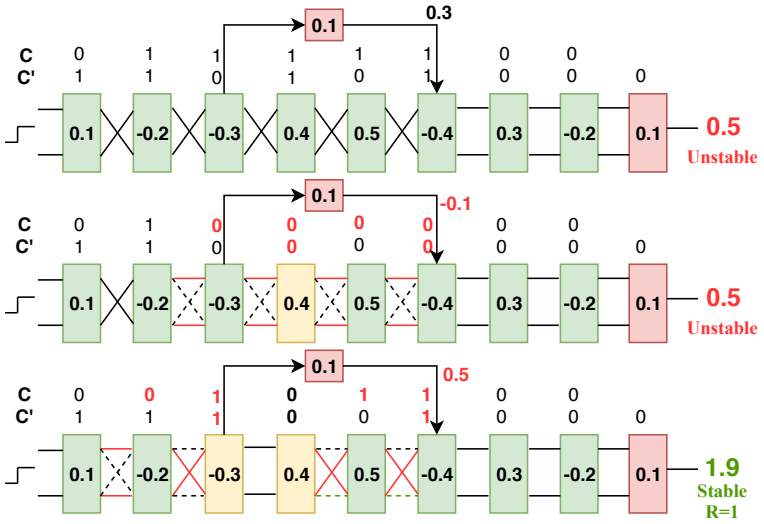


Fig. 6.4: Example of bit-flipping algorithm for an 8-bit feed-forward PUF with threshold on total delay-difference=**1.8**: 2 XOR bit-flips lead to 3 bit-flips in the final challenge

Example: We consider an 8-bit PUF and assume the delay differences, Δ=[0.1,-0.2,-0.3,0.4,0.5,-0.4,0.3,-0.2], arbiter delay, Δ^{arb}=0.4 and threshold, α=0.6 for linear and 1.8 for feed-forward configuration.

Linear: We consider the input challenge, C='**01111100**'. XOR-ed challenge will be C'='**11010100**'. Therefore, $r_N(i)$ = [-0.1,0.2,-0.3,-0.4,0.5,0.4,0.3,-0.2]. Using model in (2.1), computed r_N=**0.5**. As r_N is positive, J=**[4,3,8,1]**. Iteration-1: As $|r_N|<\alpha$, i.e., $|0.5|<0.6$, challenge is *unstable* and **bit-4** in C' needs to be flipped. New C'='110**0**0100' and r_N= **1.3**. Iteration-2: As $|r_N|>\alpha$, i.e., $|1.3|>0.6$, exit loop as stable challenge is obtained. Using relation shown in Fig. 6.3, the new challenge C_f is'01**00**1100'. Number of bit-flips in C' and C are 1 and 2, respectively.

Feed-forward: Again the same challenge, C='**01111100**'. XOR-ed challenge will be C'='**11010100**'. However, the internally generated challenge bit at stage 6 is computed using total delay-difference at N_2, r_{N_2} = 0.3. This indicates internal challenge bit as '1', which is the same as the external bit in this case. Therefore, $r_N(i)$ = [-0.1,0.2,-0.3,-0.4,0.5,0.4,0.3,-0.2]. Using model in (2.1), computed r_N=**0.5**. As r_N is positive, J=**[4,3,8,1]**. Iteration-1: As $|r_N|<\alpha$, i.e., $|0.5|<1.8$, challenge is *unstable* and **bit-4** in C' needs to be flipped. New C'='110**0**0100' and r_N= **0.5**. Iteration-2: As $|r_N|<\alpha$, i.e., $|0.5|<1.8$, challenge is *unstable* and **bit-3** in C' needs to be flipped. New C'='11**1**0100'

and r_N= **1.9**. Iteration-2: As $|r_N|>\alpha$, i.e., $|1.9|>1.8$, exit loop as stable challenge is obtained. The new challenge C_f is'0**010**1100'. Number of bit-flips in C' and C are 2 and 4, respectively.

Table 6.3: Performance of bit-flipping algorithm for synthesized 32-bit MUX PUFs with 10,000 random challenges

Performance Metrics		**PUF 1**	**PUF 2**
Linear	Uniqueness	100%	100%
	%Stability-Before	90.89%	90.7%
	%Stability-After	100%	100%
	mean XOR bit flips, C'	1	1
	max XOR bit flips, C'	1	1
	mean bit flips, C	2	2
	max bit flips, C	2	2
Feed-forward	Uniqueness	99.98%	99.97%
	%Stability-Before	84.28%	85.41%
	%Stability-After	99.43%	97.61%
	mean XOR bit flips, C'	3.27	3.18
	max XOR bit flips, C'	7	6
	mean bit flips, C	6.55	6.38
	max bit flips, C	15	13
Modified FF	Uniqueness	100%	100%
	%Stability-Before	90.09%	89.57%
	%Stability-After	99.93%	99.8%
	mean XOR bit flips, C'	1	1
	max XOR bit flips, C'	1	1
	mean bit flips, C	3.02	3.01
	max bit flips, C	4	4

6.3.4 Results

Table 6.3 shows the performance of the bit-flipping approach with 10,000 randomly chosen challenges. We can evaluate the performance using the following metrics:

- *Uniqueness* gives an indication of the percentage of unique challenges after the

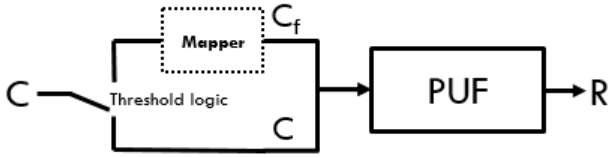


Fig. 6.5: Proposed PUF methodology for authentication

mapping. This can be an important security metric. Any value other than 100% indicates the existence of some many-to-one challenge mapping for the algorithm. We observe that most of the challenges after the mapping are unique. However, in case of feed-forward, a higher threshold means lower number of available stable challenges after the conversion. There are only 2% of challenges above the threshold and therefore, the bit-flipping algorithm may map to some of the challenges more than once.

- *Stability* is a key metric of interest in this paper. We observe the percentage of challenges that are stable before and after the mapping. Before the mapping, about 10-15% of challenges were unstable. With the proposed approach, we have drastically reduced the percentage of unstable challenges to less than 1%.

- *Bit-flips* We observe that in the case of linear and modified feed-forward configuration, we typically need just 1 bit-flip in the XOR-ed challenge, C'. For feed-forward, however, we need higher number of bit-flips.

6.4 Discussion

In this section we discuss the implications of our proposed approach. We propose an architecture shown in Fig. 6.5. The bit-flipping logic can be thought of as a mapping operation. The mapper is only needed when the total delay-difference, r_N, is less than the threshold, α. For the values of threshold suggested in this paper, the mapper is used for about 30% of the challenges in case of linear and the modified feed-forward configurations. For feed-forward, due to its high threshold, it is used for almost 98% of the challenges. However, we showed that the worst-case computations for the proposed approach is much better than the naive approach based on discarding unstable challenges. And the average computations for the approach are slightly better when $t > k_1$.

This is a valid assumption for MUX PUFs with large number of multiplexers, like 32 or 64-bit PUFs.

In the case of a scenario where the highest delay-difference value is greater than threshold, i.e., $max(|\Delta^i|) > \alpha$, one bit-flip in the XOR-ed challenge would guarantee stability. This is because one bit-flip changes the total delay-difference, r_N, value by $2\Delta^i$. So, one may also choose the value of threshold just below the highest expected value of delay-difference. This would reduce the number of iterations.

6.5 Conclusion

This paper has presented a novel bit-flipping approach for authenticating PUF devices where challenge-response pairs do not need to be stored. In the proposed approach, the server only needs to store model parameters extracted after the chip is fabricated. These parameters are further used to compute total delay-difference which is used to decide the stability of a challenge and if unstable, convert it to a stable one by flipping few bits. Future work can be directed towards more efficient implementations for different MUX PUF configurations.

Chapter 7

Conclusion and Future Direction

This thesis presents various approaches for the modelling of MUX-based PUFs. These are novel approaches based on the statistical analysis of the lab data.

The analysis of the PUF hardware using linear delay model and its extension for the analysis of the effect of aging are important contributions of this work. Though the model makes some assumptions, they are justified based on experimental data and prior works. The aging model can be further extended by including the variations due to other delay parameters like setup and hold times of the arbiter. Variations due to all the parameters, i.e, delay chain consisting of the multiplexers, the arbiter and noise contribute the overall variance of the total delay-difference. This variance is directly related to the flatness parameter k for the sigmoid function corresponding to soft-responses. The sigmoid curve can also be used to learn the classification of stable '0' and stable '1' instead of soft-responses. The accuracy of such a model would be higher than one obtained from all CRPs.

The entropy approach to detect linearity of different PUF configurations can be further extended to analyze their unpredictability. A higher value of entropy corresponds to a higher level of security. Futhermore, the appraoch can be used to reverse engineer MUX based PUFs. The zeros obtained the entropy curve correspond to the location of internal challenge bit(s) or output of intermediate arbiter(s). However, the stages corresponding to the input of arbiter is fairly unknown. In case of modified feed-forward configuration discussed in this work, where inter-connection exists between N_2 and $N_2 + 1$, the location of input of arbiter is not significant for determining the final

response bit. This makes such configurations more susceptible to attacks.

References

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

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

[3] Daihyun Lim, Jae W Lee, Blaise Gassend, G Edward Suh, Marten Van Dijk, and Srinivas Devadas. Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(10):1200–1205, 2005.

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

[5] SV Sandeep Avvaru, Chen Zhou, Saroj Satapathy, Yingjie Lao, Chris H Kim, and Keshab K Parhi. Estimating delay differences of arbiter PUFs using silicon data. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 543–546. IEEE, 2016.

[6] Gabriel Hospodar, Roel Maes, and Ingrid Verbauwhede. Machine learning attacks on 65nm arbiter PUFs: Accurate modeling poses strict bounds on usability. In *2012 IEEE international workshop on Information forensics and security (WIFS)*, pages 37–42. IEEE, 2012.

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

[8] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Francois-Xavier Standaert, and Christian Wachsmann. A formalization of the security features of physical functions. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 397–412. IEEE, 2011.

[9] Farinaz Koushanfar, Saverio Fazzari, Carl McCants, William Bryson, Matthew Sale, Peilin Song, and Miodrag Potkonjak. Can EDA combat the rise of electronic counterfeiting? In *Proceedings of the 49th Annual Design Automation Conference*, pages 133–138. ACM, 2012.

[10] Abhranil Maiti and Patrick Schaumont. Improved ring oscillator PUF: an FPGA-friendly secure primitive. *Journal of cryptology*, 24(2):375–397, 2011.

[11] Yingjie Lao and Keshab K Parhi. Statistical analysis of MUX-based physical unclonable functions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(5):649–662, 2014.

[12] Jorge Guajardo, Sandeep S Kumar, Geert Jan Schrijen, and Pim Tuyls. FPGA intrinsic PUFs and their use for IP protection. In *CHES*, volume 4727, pages 63–80. Springer, 2007.

[13] Hongliang Chang and Sachin S Sapatnekar. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. In *Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, page 621. IEEE Computer Society, 2003.

[14] Ulrich Rührmair, Heike Busch, and Stefan Katzenbeisser. Strong PUFs: models, constructions, and security proofs. In *Towards hardware-intrinsic security*, pages 79–96. Springer, 2010.

[15] Ulrich Rührmair, Jan Sölter, Frank Sehnke, Xiaolin Xu, Ahmed Mahmoud, Vera Stoyanova, Gideon Dror, Jürgen Schmidhuber, Wayne Burleson, and Srinivas Devadas. PUF modeling attacks on simulated and silicon data. *IEEE Transactions on Information Forensics and Security*, 8(11):1876–1891, 2013.

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

[17] Yingjie Lao and Keshab K Parhi. Novel reconfigurable silicon physical unclonable functions. In *Proceedings of Workshop on Foundations of Dependable and Secure Cyber-Physical Systems (FDSCPS)*, pages 30–36. Citeseer, 2011.

[18] Yingjie Lao and Keshab K Parhi. Reconfigurable architectures for silicon physical unclonable functions. In *Electro/Information Technology (EIT), 2011 IEEE International Conference on*, pages 1–7. IEEE, 2011.

[19] Anoop Koyily, Chen Zhou, Chris H Kim, and Keshab K Parhi. An entropy test for determining whether a MUX PUF is linear or nonlinear. In *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*, pages 1–4. IEEE, 2017.

[20] SV Sandeep Avvaru, Chen Zhou, Chris H Kim, and Keshab K Parhi. Predicting hard and soft-responses and identifying stable challenges of MUX PUFs using ANNs. In *2011 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 934–937. IEEE, 2017.

[21] Zouha Cherif Jouini, Jean-Luc Danger, and Lilian Bossuet. Performance evaluation of physically unclonable function by delay statistics. In *New Circuits and Systems Conference (NEWCAS), 2011 IEEE 9th International*, pages 482–485. IEEE, 2011.

[22] Zaur Tariguliyev and Berna Ors. Reliability and security of arbiter-based physical unclonable function circuits. *International Journal of Communication Systems*, 26(6):757–769, 2013.

[23] Vladimir Stojanovic and Vojin G Oklobdzija. Comparative analysis of master-slave latches and flip-flops for high-performance and low-power systems. *IEEE Journal of solid-state circuits*, 34(4):536–548, 1999.

[24] Cícero Nunes, Paulo F Butzen, André I Reis, and Renato P Ribas. A methodology to evaluate the aging impact on flip-flops performance. In *2013 26th Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–6. IEEE, 2013.

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

[26] Konstantinos Markantonakis and Keith Mayes. Errata to: Secure smart embedded devices, platforms and applications. In *Secure Smart Embedded Devices, Platforms and Applications*, pages E3–E14. Springer, 2014.

[27] Ashirwad Bahukhandi. Lecture notes for advanced logic design and switching theory, January 2002.

[28] Chen Zhou, Saroj Satapathy, Yingjie Lao, Keshab K Parhi, and Chris H Kim. Soft response generation and thresholding strategies for linear and feed-forward MUX PUFs. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, pages 124–129. ACM, 2016.

[29] Srinivas Devadas, Edward Suh, Sid Paral, Richard Sowell, Tom Ziola, and Vivek Khandelwal. Design and implementation of puf-based" unclonable" RFID ICs for anti-counterfeiting and security applications. In *RFID, 2008 IEEE International conference on*, pages 58–64. IEEE, 2008.

[30] Charles Herder, Meng-Day Yu, Farinaz Koushanfar, and Srinivas Devadas. Physical unclonable functions and applications: A tutorial. *Proceedings of the IEEE*, 102(8):1126–1141, 2014.

[31] Christian Wachsmann and Ahmad-Reza Sadeghi. Physically unclonable functions (PUFs): Applications, models, and future directions. *Synthesis Lectures on Information Security, Privacy, & Trust*, 5(3):1–91, 2014.

[32] Sanu K Mathew, Sudhir K Satpathy, Mark A Anders, Himanshu Kaul, Steven K Hsu, Amit Agarwal, Gregory K Chen, Rachael J Parker, Ram K Krishnamurthy, and Vivek De. 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 (ISSCC), 2014 IEEE International*, pages 278–279. IEEE, 2014.

[33] Anastacia Alvarez, Wenfeng Zhao, and Massimo Alioto. 14.3 15fj/b static physically unclonable functions for secure chip identification with¡ 2% native bit instability and 140× inter/intra PUF hamming distance separation in 65nm. In *Solid-State Circuits Conference-(ISSCC), 2015 IEEE International*, pages 1–3. IEEE, 2015.

[34] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Berk Sunar, and Pim Tuyls. Memory leakage-resilient encryption based on physically unclonable functions. In *Towards Hardware-Intrinsic Security*, pages 135–164. Springer, 2010.

[35] Yansong Gao, Damith Chinthana Ranasinghe, Gefei Li, Said F Al-Sarawi, Omid Kavehei, and Derek Abbott. A challenge obfuscation method for thwarting model building attacks on PUFs. *IACR Cryptology ePrint Archive*, 2015:471, 2015.

[36] Dinesh Ganta and Leyla Nazhandali. Study of IC aging on ring oscillator physical unclonable functions. In *Fifteenth International Symposium on Quality Electronic Design*, pages 461–466. IEEE, 2014.

[37] John Keane, Xiaofei Wang, Devin Persaud, and Chris H Kim. An all-in-one silicon odometer for separately monitoring HCI, BTI, and TDDB. *IEEE Journal of Solid-State Circuits*, 45(4):817–829, 2010.

[38] Abhishek Tiwari and Josep Torrellas. Facelift: Hiding and slowing down aging in multicores. In *2008 41st IEEE/ACM International Symposium on Microarchitecture*, pages 129–140. IEEE, 2008.

[39] Naghmeh Karimi, Jean-Luc Danger, Florent Lozach, and Sylvain Guilley. Predictive aging of reliability of two delay PUFs. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 213–232. Springer, 2016.

[40] Vikram G Rao and Hamid Mahmoodi. Analysis of reliability of flip-flops under transistor aging effects in nano-scale CMOS technology. In *2011 IEEE 29th International Conference on Computer Design (ICCD)*, pages 439–440. IEEE, 2011.

[41] Anoop Koyily, Satya Venkata Sandeep Avvaru, Chen Zhou, Chris H Kim, and Keshab K Parhi. Effect of aging on linear and nonlinear MUX PUFs by statistical modeling. In *Design Automation Conference (ASP-DAC), 2018 23rd Asia and South Pacific*, pages 76–83. IEEE, 2018.

[42] Sangwoo Pae, Jose Maiz, Chetan Prasad, and Bruce Woolery. Effect of BTI degradation on transistor variability in advanced semiconductor technologies. *IEEE Transactions on Device and Materials Reliability*, 8(3):519–525, 2008.

[43] Stewart E Rauch. Review and reexamination of reliability effects related to NBTI-induced statistical variations. *IEEE Transactions on Device and Materials Reliability*, 7(4):524–530, 2007.

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

[45] George Marsaglia. Ratios of normal variables and ratios of sums of uniform variables. *Journal of the American Statistical Association*, 60(309):193–204, 1965.

[46] Yingjie Lao, Qianying Tang, Chris H Kim, and Keshab K Parhi. Beat frequency detector–based high-speed true random number generators: Statistical modeling and analysis. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(1):9, 2016.

[47] Abhranil Maiti, Logan McDougall, and Patrick Schaumont. The impact of aging on an fpga-based physical unclonable function. In *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, pages 151–156. IEEE, 2011.

[48] Meng-Day Mandel Yu, David MRaihi, Richard Sowell, and Srinivas Devadas. Lightweight and secure puf key storage using limits of machine learning. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 358–373. Springer, 2011.

[49] Dominik Maria Endres and Johannes E Schindelin. A new metric for probability distributions. *IEEE Transactions on Information theory*, 49(7):1858–1860, 2003.

[50] Norbert Henze and Mathew D Penrose. On the multivariate runs test. *Annals of statistics*, pages 290–298, 1999.

[51] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

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

[53] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. Lightweight secure PUFs. In *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, pages 670–673. IEEE, 2008.

[54] Stefan Katzenbeisser, Ünal Kocabaş, Vladimir Rožić, Ahmad-Reza Sadeghi, Ingrid Verbauwhede, and Christian Wachsmann. PUFs: Myth, fact or busted? a security evaluation of physically unclonable functions (pufs) cast in silicon. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 283–301. Springer, 2012.

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

[56] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[57] Weiqiang Liu, Yifei Yu, Chenghua Wang, Yijun Cui, and Máire O'Neill. RO PUF design in fpgas with new comparison strategies. In *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, pages 77–80. IEEE, 2015.

[58] Anoop Koyily, Chen Zhou, Chris H Kim, and Keshab K Parhi. Predicting soft-response of MUX PUFs via logistic regression of total delay difference. In *Circuits*

and Systems (ISCAS), 2018 IEEE International Symposium on, page To Appear. IEEE, 2018.