Attack-Resistance and Reliability Analysis of Feed-Forward and Feed-Forward XOR PUFs

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

Satya Venkata Sandeep Avvaru

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

Prof. Keshab K. Parhi

May, 2019

Acknowledgements

I would like to express my sincere gratitude to my adviser Prof. Keshab Parhi for his support, patience and invaluable guidance through the course of this research. I would like to thank Prof. Chris Kim for his feedback and for providing the data that initiated this work. I would also like to thank them along with Prof. Antonia Zhai, the thesis committee, for taking their time to review my thesis.

I am extremely grateful to my parents for their unconditional love and their belief in me.

I would like to extend my gratitude towards my current and past lab mates, Sandhya Koteshwara, Bhaskar Sen, Nanda Kumar Unnikrishnan, Anoop Koyily and Lulu Ge for their encouragement and suggestions. I am also very thankful for my friends, Krishna Sandeep Prata, Ankit Moldgy and Anirudh Reddy Ravula and Teja Dasari for their ongoing company, friendship and support.

Abstract

Physical unclonable functions (PUFs) are lightweight hardware security primitives that are used to authenticate devices or generate cryptographic keys without using non-volatile memories. This is accomplished by harvesting the inherent randomness in manufacturing process variations (e.g. path delays) to generate random yet unique outputs. A multiplexer (MUX) based arbiter PUF comprises two parallel delay chains with MUXs as switching elements. An input to a PUF is called a *challenge vector* and comprises of the select bits of all the MUX elements in the circuit. The output-bits are referred to as *responses*. In other words, when queried with a challenge, the PUF generates a response based on the uncontrollable physical characteristics of the underlying PUF hardware. Thus, the overall path delays of these delay chains are random and unique functions of the challenge.

The contributions in this thesis can be classified into four main ideas. First, a novel approach to estimate delay differences of each stage in MUX-based *standard arbiter PUFs*, *feed-forward PUFs* (FF PUFs) and *modified feed-forward PUFs* (MFF PUFs) is presented. Test data collected from PUFs fabricated using 32 nm process are used to learn models that characterize the PUFs. The delay differences of individual stages of arbiter PUFs correspond to the model parameters. This was accomplished by employing the *least mean squares* (LMS) adaptive algorithm. The models trained to learn the parameters of two standard arbiter PUF-chips were able to predict responses with 97.5% and 99.5% accuracy, respectively. Additionally, it was observed that perceptrons can be used to attain 100% (approx.) prediction accuracy. A comparison shows that the perceptron model parameters are *scaled* versions of the model derived by the LMS algorithm. Since the delay differences are challenge independent, these parameters can be stored on the server which enables the server to issue random challenges whose responses need not be stored. By extending this analysis to 96 standard arbiter PUFs, we confirm that the delay differences of each MUX stage of the PUFs follow a Gaussian probability distribution.

Second, artificial neural network (ANN) models are trained to predict hard and soft-responses of the three configurations: standard arbiter PUFs, FF PUFs and MFF

PUFs. These models were trained using silicon data extracted from 32-stage arbiter PUF circuits fabricated using IBM 32 nm HKMG process and achieve a response-prediction accuracy of 99.8% in case of standard arbiter PUFs, approximately 97% in case FF PUFs and approximately 99% in case of MFF PUFs. Also, a probability based thresholding scheme is used to define *soft-responses* and artificial neural networks were trained to predict these soft-responses. If the response of a given challenge has at least 90% consistency on repeated evaluation, it is considered *stable*. It is shown that the soft-response models can be used to filter out unstable challenges from a randomly chosen independent test-set. From the test measurements, it is observed that the probability of a stable challenge is typically in the range of 87% to 92%. However, if a challenge is chosen with the proposed soft-response model, then its portability of being stable is found to be 99% compared to the ground truth.

Third, we provide the first systematic empirical analysis of the effect of FF PUF design choices on their reliability and attack resistance. FF PUFs consist of feed-forward loops that enable internally generated responses to be used as select-bits, making them slightly more secure than a standard arbiter PUFs. While FF PUFs have been analyzed earlier, no prior study has addressed the effect of loop positions on the security and reliability. After evaluating the performance of hundreds of PUF structures in various design configurations, it is observed that the locations of the arbiters and their outputs can have a substantial impact on the security and reliability of FF PUFs. Appropriately choosing the input and output locations of the FF loops, the amount of data required to attack can be increased by 7 times and can be further increased by 15 times if two intermediate arbiters are used. It is observed adding more loops makes PUFs more susceptible to noise; FF PUFs with 5 intermediate arbiters can have reliability values that are as low as 81%. It is further demonstrated that a *soft-response thresholding strategy* can significantly increase the reliability during authentication to more than 96%.

It is known that *XOR arbiter PUFs* (XOR PUFs) were introduced as more secure alternatives to standard arbiter PUFs. XOR PUFs typically contain multiple standard arbiter PUFs as their components and the output of the component PUFs is XOR-ed to generate the final response. Finally, we propose the design of feed-forward XOR PUFs (FFXOR PUFs) where each component PUF is an FF PUF instead of a standard

arbiter PUF. Attack-resistance analysis of FFXOR PUFs was carried out by employing artificial neural networks with 2-3 hidden layers and compared with XOR PUFs. It is shown that FFXOR PUFs cannot be accurately modeled if the number of component PUFs is more than 5. However, the increase in the attack resistance comes at the cost of degraded reliability. We also show that the soft-response thresholding strategy can increase the reliability of FFXOR PUFs by about 30%.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Research overview

The rapid development of computing hardware has provided the software flexibility to enable convenient mobile data processing. Indeed, devices such as smartphones have become a unified platform capable of conducting financial transactions and storing a user's private information. Due to the mobile nature of electronic devices, privacy and security are pressing concerns especially in cases where an adversary can gain physical access to the devices. Traditional security measures involve storing a secret key in a non-volatile memory such as an electrically erasable programmable read-only memory (EEPROM) or a static random-access memory (SRAM) which can be expensive in terms of their area and power consumption. Also, the secrecy is difficult to uphold in practice. With embedded devices becoming more ubiquitous, there is a requirement for low-power, low-area and low-cost hardware security alternatives.

Physical unclonable functions (PUFs) are light-weight, low-cost hardware security primitives that can be used to securely authenticate devices or generate cryptographic keys without the involvement of non-volatile memories. This is accomplished by harvesting inherent randomness in manufacturing process variations of integrated circuits (ICs) to create secret keys. PUFs were introduced in the very beginning of the twenty first century, first as physical one-way functions [2], then as physical random functions [3] and finally as physical unclonable functions or PUFs. Since then, numerous PUF realizations have been proposed [4] [5]. Based on whether they are typically used for

authentication or key generation, PUFs can be broadly categorized as "strong PUFs" or "weak PUFs". One of the fundamental differences between strong PUFs and weak PUFs is that weak PUFs support a small number of unique challenges while strong PUFs can process a large number of challenges. As a result, strong PUFs make it unfeasible for an adversary to access all the challenge-response pairs (CRPs) in a limited time. Arbiter PUFs are an example of strong PUFs.

In this work, we analyze and evaluate the performance of various arbiter PUF structures that exploit the manufacturing variability in gate delay as the source of unclonable randomness. Unclonablity, reliability, uniqueness and randomness are fundamental characteristics of PUFs [6] [7]. Unclonablity ensures resilience against attacks that can replicate the behavior of a PUF. Reliability is a measure of robustness against noise and environmental variations. Uniqueness ensures a PUF produces unique responses compared to other PUFs with identical design and layout while randomness ensures that responses of a PUF are not biased towards a 0 or a 1.

This work mainly focuses on attack-resistance and reliability analyses. As discussed in [8], possible attacking strategies on PUFs can be classified into 3 kinds: Prediction attacks, Reverse engineering attacks and Collision attacks. A Reverse engineering attack attempts to learn the behavior of a PUF by studying the input-output relation between several challenge response pairs. The knowledge of the PUF architecture and the amount of CRPs available usually have a significant effect on the feasibility of an attack. Several reverse engineering attack strategies have been proposed in the past. The authors in [9] have demonstrated the vulnerability of arbiter PUFs to modeling attacks and executed an attack strategy using a machine learning technique. In [10] and [11], responses from silicon data have been used to model attacks on arbiter PUFs using machine learning algorithms. Despite this limitation, they are used as building blocks in variants like XOR arbiter PUFs [12], lightweight secure PUFs [13], and composite PUFs [14] because of their low area overhead and availability of a large set of challenge-response pairs (CRPs). Feed-forward arbiter PUFs and XOR arbiter PUFs were introduced as more secure alternatives to standard arbiter PUFs. However, the added security comes at the cost of degraded reliability, i.e., they are more susceptible to noise. In this thesis, we study the attack resistance and reliability of various configurations of standard arbiter PUFs, Feed-forward arbiter PUFs and XOR arbiter PUFs.

1.2 Outline of the thesis

- Chapter 2 describes the designs and the mathematical models of standard, feed-forward and modified feed forward arbiter PUFs. It also presents methods to estimate the physical parameters, that characterize these PUFs, using challenge-response pairs.
- In Chapter 3, artificial neural network models to predict hard and soft-responses are presented and their implications are discussed.
- Chapter 4 studies the effect of feed-forward loops placement on attack-resistance and reliability of feed-forward PUFs with multiple loops.
- Chapter 5 proposes and analyzes feed-forward XOR PUFs in terms of their attack-resistance and reliability.
- Chapter 6 presents the main observations derived in the thesis and briefly discusses the future directions.

Chapter 2

Estimating Delay Differences of Standard and Feed-Forward Arbiter PUFs

2.1 Introduction

In this chapter, a novel approach to estimate the delay difference of each MUX stage in a PUF using measured data from PUF chips fabricated in 32 nm process is presented. A total of six PUF chips were fabricated and 96 MUX PUF circuits were implemented on each chip. The layout was identical for each PUF circuit. Challenge-Response test data are used to model the delay differences of each MUX stage using a simple *least mean square (LMS)* adaptive filtering algorithm. The advantage of this approach is that the delay differences of arbiter PUFs can be estimated without using any sophisticated machine learning techniques. Additionally, parameters involved in feed-forward and modified feed-forward configurations are observed. It is shown that the delay differences indeed follow a Gaussian distribution which is a standard assumption. It is also shown that the delay differences of the various stages of the PUF circuits in each chip and among all chips are unique. These delay differences have not been estimated in any prior analysis of MUX PUFs. The fact that the delay differences belong to the same Gaussian PDF has also not been confirmed from test data before. An approach to

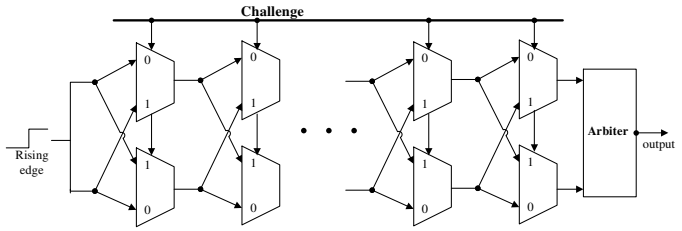


Figure 2.1: Structure of a standard arbiter based PUF.

estimate *arbiter delays* utilizing these delay differences for feed-forward and modified feed-forward PUFs is also presented in this chapter.

2.2 Background

2.2.1 Arbiter Based PUFs

Arbiter PUFs [15] [9] are delay based PUFs that use an arbiter circuit in order to compare path delays of the circuit. Fig. 2.1 illustrates the basic structure of an arbiter based MUX PUF. In a MUX PUF, the delay difference between the two possible paths of a MUX stage is different for each stage in a chip. The challenge is typically a randomly chosen binary vector whose length is same as the number of stages in the circuit. It can be observed that there are two possible paths that are excited by the rising edge and hence a race condition is established to reach the output. The choice of the challenge vector affects the individual path delays at each stage of the PUF and hence the overall path delays. The function of the arbiter circuit is to determine which rising edge arrives first and assign the output, i.e., the response to 0 or 1, accordingly.

A standard arbiter PUF can be characterized by an *additive linear delay model* [15] [13] [16]. An additive delay model is based on the assumption that the total delay in a path is sum of the delays due to elementary components. This further implies that the ultimate difference in path delays can be modeled as sum of individual delay differences at each stage. The individual delay difference at the ith stage, denoted by Δ_i, depends on the ith bit of the challenge vector. Thus, the response of an arbiter based PUF which is based on the overall delay difference between the paths is influenced by the corresponding challenge vector. If C_i denotes the challenge bit at the i-th stage of an

arbiter PUF with N stages, the overall delay difference Δ can be computed as

$$\Delta = \sum_{i=1}^{N} (-1)^{X_i} \Delta_i, \tag{2.1}$$

where each X_i is computed as a cumulative XOR of the successive challenge bits, i.e., $X_i = C_{i+1} \oplus C_{i+2} ... \oplus C_N$ for $i = 1$ to $N - 1$ and $X_N = 0$. The response R is 0 or 1 depending on the sign of the overall delay difference.

$$R = \begin{cases} 1, \Delta \geq 0 \\ 0, \Delta < 0. \end{cases} \tag{2.2}$$

Like any physical circuit, PUFs are also subject to random noise. As a result of uncertainty due to noise and manufacturing processes there may be setup-hold time violations in the arbiter circuit leading to meta-stable outputs. The output or the response in this case is referred to as an *unstable response*.

2.2.2 Feed-Forward PUFs

The ability to model arbiter PUFs as linear models makes it susceptible to modeling attacks where an attacker tries to build a software clone of the PUF. As a way to make arbiter PUFs more secure, *feed-forward PUFs* (FF PUFs) have been proposed [15] [9] [16]. Structure of a simple feed-forward arbiter PUF is depicted in Fig. 2.2. In this case, an additional arbiter, called *intermediate arbiter*, is used to determine the outputs of one of the intermediate stages which is then used as a select bit for one of the later stages. Note that multiple internal arbiters can also be used. This improves security by introducing non-linearity into the model making it more complex since the additive linear delay model is no longer valid. Moreover, this introduces uncertainty as the locations of the internal feed-forward loops are hidden to the adversary making it more difficult to build an accurate model. Multiple such feed-forward loops can be used to make it more complex.

2.2.3 Modified Feed-Forward PUFs

One of the drawbacks of using FF PUFs is that *reliability* is degraded as compared to a standard arbiter PUF of same size [7]. Reliability is the ability to produce a constant

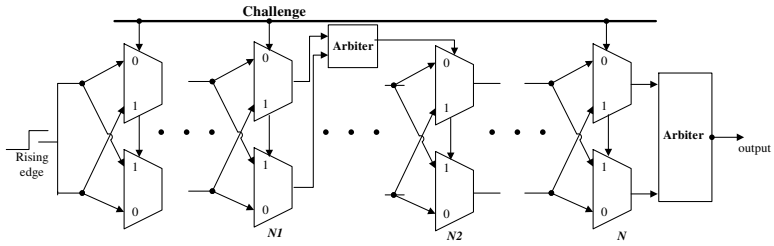


Figure 2.2: Structure of a feed-forward PUF with one loop (FF PUF).

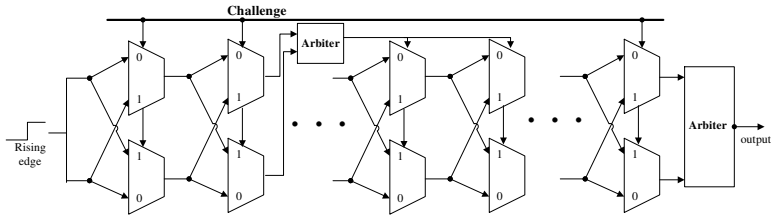


Figure 2.3: Structure of a Modified feed-forward PUF (MFF PUF).

response to the same challenge under different environmental conditions. To improve reliability, a *modified feed-forward PUF* (MFF PUF) structure shown in Fig. 2.3 was proposed in [7]. In this configuration, the arbiter output from an intermediate stage is used as a challenge bit for two consecutive later stages.

2.2.4 Least Mean Square (LMS) Algorithm

Gradient descent or the *method of steepest descent* [17] is an adaptive optimization algorithm used to minimize (or maximize) a given function, referred as the *cost function*. Starting from an arbitrary *tap-weight vector* ($\mathbf{w}$), the solution improves iteratively. In each iteration, the weight vector is adjusted in the direction of the steepest descent or the direction opposite to the gradient of the cost function. Ultimately, under appropriate conditions, the solution converges to the *Wiener solution*. The *LMS algorithm* is based on an approximation of gradient descent where an instantaneous estimate of the gradient computed from available data is used. Therefore, the LMS algorithm is essentially a tool to estimate the parameters (called weights) which optimally express a given input-output relation by minimizing a so called cost function.

In this context, the objective is to minimize the mean squared error between the

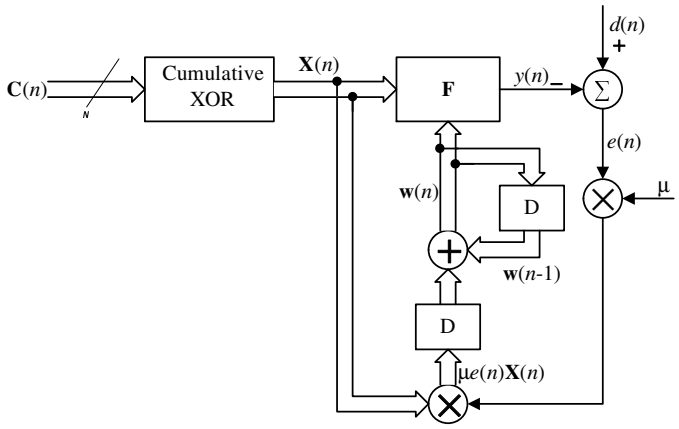


Figure 2.4: Block diagram of the LMS algorithm

predicted response and the desired response. In other words, we estimate the delay differences by using challenges and responses as inputs and outputs of the desired model. As described in the previous section, an arbiter PUF is a delay based model and can be characterized by the additive linear delay model. These parameters, i.e., the delay differences at each stage or the values of Δ_i's, are considered as the weights of an adaptive filter. Fig. 2.4 shows the block diagram which illustrates the sequence of steps involved in estimating these weights based on the LMS algorithm from a given set of CRPs [18]. The input vector is concatenated with a '1' to account for the bias. So, the tap-weight vector is of size $N + 1$ if the MUX has N stages. The filter output $y(n)$ is then computed and compared to the desired response bit $d(n)$. The difference $e(n)$ is then used to update the weights.

2.2.5 Single Layer Perceptrons

Artificial Neural Networks (ANN) were used to model the functionality of PUFs. These models were then used to predict responses. We need to provide sufficiently large amount of data to a machine learning algorithm, in order to learn a predictive model. The data in this case are the available CRPs. The simplest version of an Artificial Neural Network is called a *single layer perceptron* or, simply, a perceptron. A perceptron is the basic processing element in any ANN and defines a hyper-plane which can be used to divide the input space into two groups [19]. This can therefore be applied to train a binary

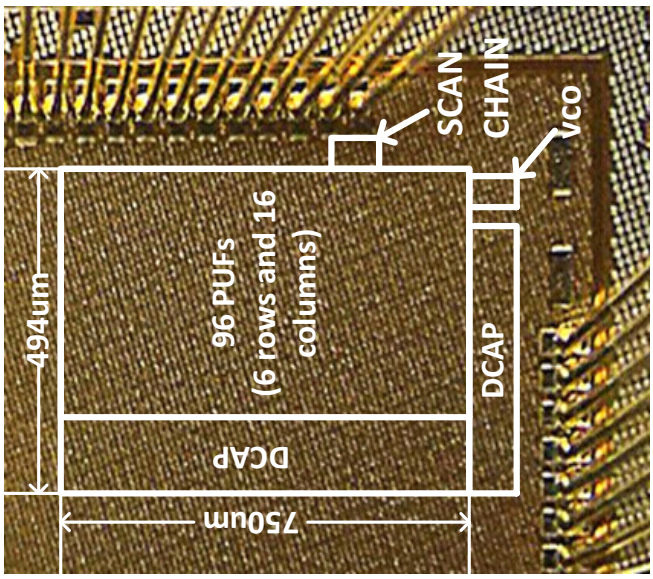


Figure 2.5: Die photo of the PUF [1].

classifier. In this work, perceptron structures with a hard-limit transfer function were implemented to train a software model of standard arbiter PUFs.

2.3 Experimental Setup

The PUFs under study were implemented in 32nm HKMG test chips in three different configurations: standard arbiter based PUF, FF PUF (with one loop) and MFF PUF. Each chip contains 96 PUF circuits and six such chips were tested. Each PUF can be configured as a linear PUF, FF PUF, and MFF PUF by programming two control bits [1]. The die photo is shown in Fig. 2.5. The PUFs thus manufactured were observed to exhibit a high degree of uniqueness and randomness [1]. For the purpose of estimating delay differences, 96 standard arbiter PUFs on a chip were tested and their corresponding CRPs were extracted. Two standard arbiter PUFs are analyzed to predict hard-responses and soft-responses. Four PUFs (two from each chip) from each configuration are analyzed. Each PUF is a 32-stage MUX based PUF, i.e., multiplexers are used as switching elements in each of the 32 stages. An on chip voltage-controlled oscillator (VCO) and counter are used to measure responses reliably and efficiently. A

set of 10,000 and 20,000 unique challenges were randomly generated in order to evaluate standard arbiter PUFs and FF PUFs, respectively.

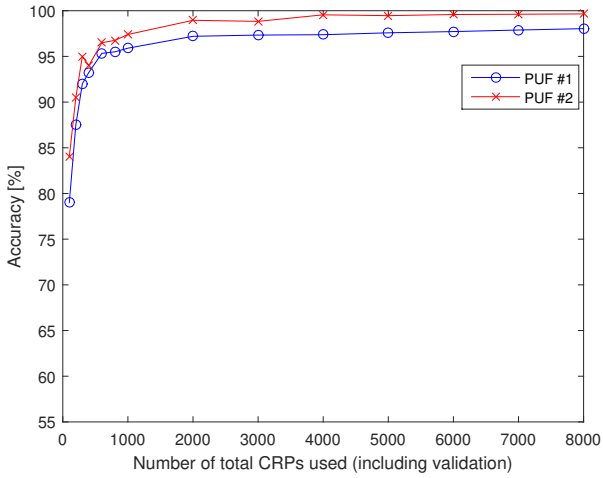
2.4 Estimating the Physical Parameters of Standard Arbiter PUFs

2.4.1 Model Accuracy

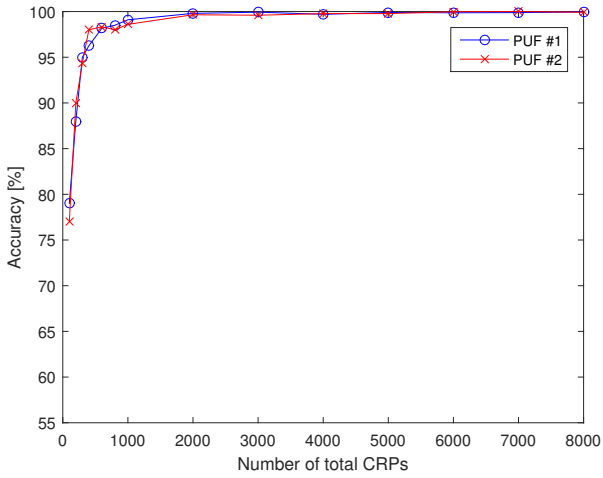
As we have a 32 stage PUF, each challenge is a 32-bit vector. So, 33 tap-weights are defined, including a weight for the bias term, and are initialized to random values between 0 and 1. The number of CRPs used for training the model is varied in order to observe its effect on model accuracy and model parameters. In order to obtain reliable estimates, a five-fold cross validation scheme is used to validate the trained models. This means that 80% of the available data is used to train the model and the remaining 20% is used to validate it, and this is repeated for five times, until all the data are tested. The response of the model is decided to be either '0' or '1' based on whether the output is positive or negative. These model responses are compared with the true responses to compute *classification accuracies*. The average accuracy across the five folds is considered as accuracy of the model.

This chapter considers modeling of the MUX PUF using perceptron and the LMS algorithm. The results of the two modeling approaches are compared. The LMS algorithm applies the gradient descent with adaptive learning rate to train the network. Mean square error is used as the cost function. The data are divided into 5 folds out of which, 3 folds are used for training while one fold is used for validation and the remaining fold is used to test the accuracy of the model. The validation fold is required in order to test the convergence of the network and decide when to stop training. These are implemented using MATLAB neural network toolbox.

Fig. 2.6(a) shows how the accuracy of the models based on the LMS varies with the number of CRPs available. Fig. 2.6(b) shows the variation of testing accuracy of models trained using single layer perceptron, as a function of number of CRPs. Each figure has two plots corresponding to the data from PUFs on two different chips.



(a) LMS



(b) Perceptron

Figure 2.6: Prediction accuracy of a model based on LMS and perceptron plotted against total number of CRPs.

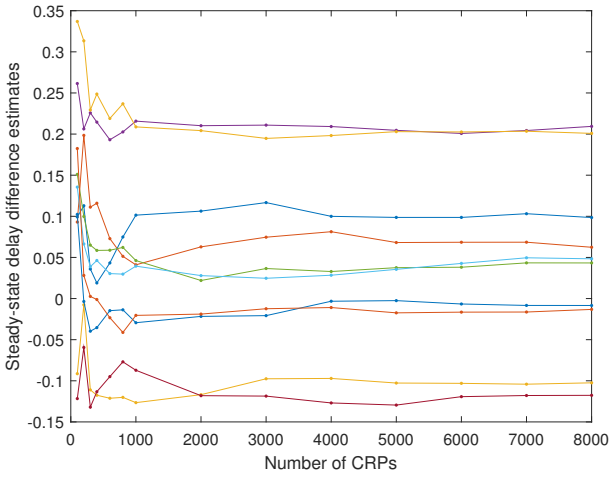
2.4.2 Convergence of the Estimated Values

The Δ values cannot be considered as estimates of parameters of a PUF if they are not reliable and consistent. In order to validate the reliability of these estimates, the PUFs have been modeled multiple times and convergence of the estimates has been verified. Initially they are assigned a random value between 0 and 1. As the training progresses, they converge to their true values. Fig. 2.7(a) shows how the estimated delay differences change as more challenges are used for training. The values oscillate initially and gradually attain a stable value. Fig. 2.7(b) shows the model parameters estimated using the perceptron algorithm as a function of number of CRPs used. Initially, the model parameters *vary* considerably with change in challenges but they gradually stabilize as the CRPs increase in number. Moreover, the delay differences estimated using LMS algorithm have been observed to be scaled versions of the model parameters of perceptron.

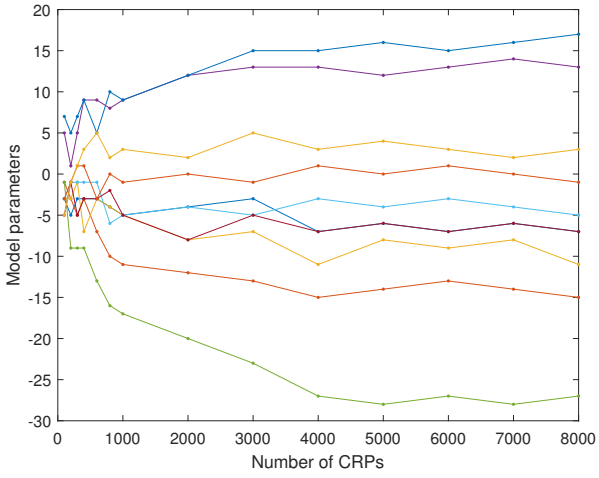
Fig. 2.8 presents a comparison of estimated delay differences using both techniques. Each value plotted corresponds to delay difference at a given stage (represented on the X-axis). A scale factor of -100 has been multiplied to the LMS estimates. Fig. 2.8 clearly illustrates a one-to-one correspondence between the models. Thus, delay differences can be estimated from either the LMS model or the perceptron model.

2.4.3 Distribution of the Delay Parameters

Besides the susceptibility of arbiter PUFs to modeling attacks, the ability to characterize the PUFs and estimate their individual delay differences at each stage is very significant. 96 PUFs were fabricated on a chip and 1000 CRPs were recorded for each of these PUFs. Based on these CRPs, PUF parameters (Δ values) were estimated and recorded. Fig. 2.9 shows the distribution of the estimated delay differences for 96 PUFs from one chip at the 17th stage. It has been observed that at every stage, the delay difference values follow a similar Gaussian distribution. In a prior statistical analysis of MUX based PUFs [7], the delays were modeled as an independently identically distributed (i.i.d.) random variable. The histogram in Fig. 2.9 validates those assumptions.



(a) LMS



(b) Perceptron

Figure 2.7: Estimated model parameters plotted against number of CRPs used for modeling PUF-1 from chip 1

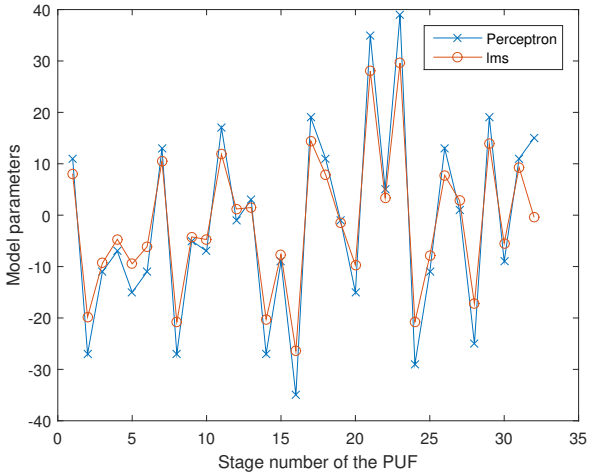


Figure 2.8: Comparison of parameters estimated using LMS and perceptron based modeling of PUF-1 from chip 1

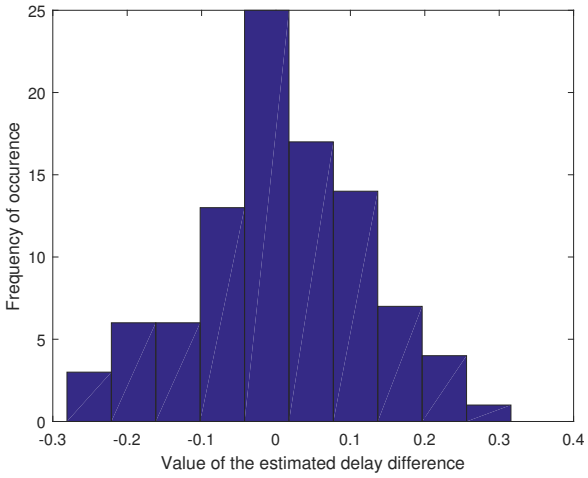


Figure 2.9: Distribution of estimated delay differences at the 17th stage

2.5 Estimating the Intermediate Bias of Feed-Forward PUFs

The knowledge of knowing individual delay differences opens new possibilities. One of them is the ability to learn other unknown physical characteristics of delay based PUFs. The standard arbiter PUFs studied earlier were modified into feed-forward and modified feed-forward structures as shown in Fig. 2.2 and Fig. 2.3. As a result, we now have the knowledge of individual delay differences of FF PUFs. Using these values, we claim that we can mathematically model different configurations of FF and MFF PUFs. But the only missing piece of information is the path delays caused by the feed-forward loops and the intermediate arbiters. We propose an approach to empirically estimate the effect of these values based on CRPs of the FF PUFs.

In case of the FF PUFs, there is a loop which connects the output of 16th stage to 26th stage, i.e., the challenge bit input of the 26th stage is now replace by the intermediate output from the 16th stage. We know that the first 15 stages still act as a standard arbiter PUF whose Δ values are known. Therefore according to the additive linear delay model, we can compute the intermediate output R_{16} as

$$R_{16} = sign(\sum_{i=1}^{16}(-1)^{X_i}\Delta_i + B_{int}), \tag{2.3}$$

where $X_i = C_{i+1} \oplus C_{i+2} ... \oplus C_{16}$, for i=1 to 15 ($X_16 = 0$) and B_{int} is the intermediate bias that we need to estimate. Then we can use the additive linear delay model, described in (2.1) and (2.2) to model FF PUF with 26th challenge bit input C_{26} replaced with R_{16}. Similarly in order to model MFF PUFs, we replace C_{26} and C_{27} with R_{16}. Thus, these formulations of feed-forward configurations have a direct correspondence to the structure of their circuit.

The unknown value R_{16} is estimated by assuming search range is between -1 to +1. So, for all values between -1 and +1 the final outputs for a set of challenges of the PUF is computed using the model described above. For this purpose, we use a data set of 18000 randomly chosen challenges with stable responses. These outputs are then compared to the ground truth to calculate the accuracy of the model. The value of the intermediate bias (B_{int}) is chosen as the one with maximum model accuracy. Moreover, a five-fold cross validation is used to test the reliability of the estimated value of B_{int} on a independent test set. A pseudo code for this algorithm is presented in Algorithm.

1.

Algorithm 1 Estimating intermediate bias of feed-forward PUFs

Input: CRPs (C, R), Δ values
Output: B_{int}
for B_{int} = values between -1 and +1 **do**
  for each challenge vector C in the training set **do**
    Compute R_{16} according to (2.3)
    Replace C_{26} with R_{16}
    Replace C_{27} with R_{16} (In the case of modified feed-forward configuration)
    Compute R according to (2.1) and (2.2)
  end for
  Compare with ground truth and compute accuracy
end for
$B^*_{int} \leftarrow B_{int}$ with maximum accuracy **return** B^*_{int}

2.5.1 Results

12000 stable challenge responses pairs were used to analyze two PUFs in multiple configurations. 10000 CRPs were used for estimation and 2000 CRPs were used to test the estimated bias. As mentioned earlier, the values of overall bias which encompasses the effect of the final arbiter were estimated to be -0.1931 and 0.0245 for PUF-1 and PUF-2, respectively. The accuracies for each assumed value in the range +1 and -1 is plotted in Fig. 2.10. As the effect of intermediate arbiter is same for both feed-forward and modified feed-forward, the value of B_{int} should be the same. As expected, similar analysis of modified feed-forward configuration of the same PUFs resulted in the same estimates as depicted in Fig. 2.11. It can be observed that there is only one peak and the accuracy drops as we go away from the optimal value.

The values of the internal bias, B_{int}, are estimated as 0.2525 for PUF-1 and 0.1313 for PUF-2. These estimated values were tested on independent test sets based on five-fold cross validation. Four folds were used for estimation and one fold for testing. The test accuracies and the estimates for the five folds are shown in Table 2.1 to demonstrate consistency of the approach.

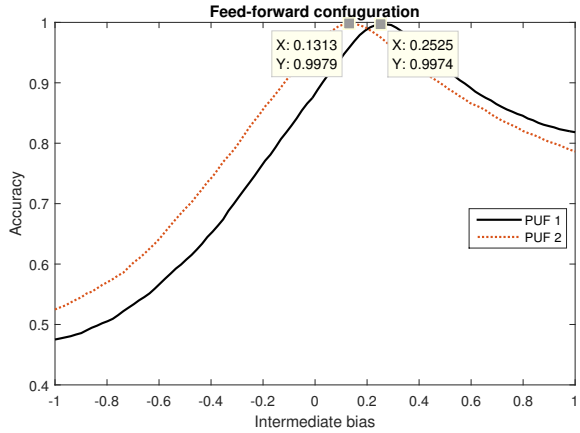


Figure 2.10: Accuracy of intermediate bias in the range -1 to +1 for feed-forward configuration.

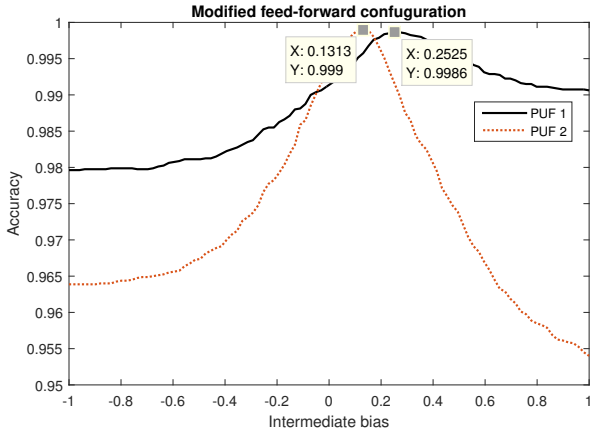


Figure 2.11: Accuracy of intermediate bias in the range -1 to +1 for modified feed-forward configuration.

2.6 Discussion and Conclusion

Even though predicting responses of an arbiter PUF has been studied previously, the LMS algorithm based approach establishes that sophisticated machine learning techniques are not required for creating a model for a MUX PUF. As we can observe from Fig. 2.6(b), by exploiting a slightly more complex structure of neural networks, we

Table 2.1: Estimated Intermediate Bias Values and the Corresponding Test Accuracies.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean
Feed-Forward PUF-1						
B_{int}	0.2525	0.2525	0.2525	0.2525	0.2525	0.2525
Acc.	0.9971	0.9958	0.9975	0.9979	0.9967	0.9970
Feed-Forward PUF-2						
B_{int}	0.1313	0.1313	0.1313	0.1313	0.1313	0.1313
Acc.	0.9967	0.9983	0.9979	0.9971	0.9992	0.9978
Modified Feed-Forward PUF-1						
B_{int}	0.2525	0.2525	0.2525	0.2525	0.2525	0.2525
Acc.	0.9975	0.9983	0.9983	0.9992	0.9975	0.9982
Modified Feed-Forward PUF-2						
B_{int}	0.1313	0.1313	0.1313	0.1313	0.1313	0.1313
Acc.	0.9975	0.9992	1.0	0.9983	0.9988	0.9988

are able to predict the response almost with certainty. Though the LMS approach is not 100% accurate, it provides a significant accuracy of around 97.5% for one PUF and around 99.5% for the other. A notable advantage of this approach, as compared to machine learning methods, is its simplicity. It has also been observed that LMS method requires considerably less training time per iteration as compared to the above mentioned ANNs.

Usually, a large set of CRPs corresponding to each chip is stored in a server for the purpose of authenticating the ICs. But storing these Δ values (and the intermediate arbiter bias) instead of CRPs provides certain benefits. First, the storage memory requirement is considerably reduced. Moreover, it is impractical to store all the possible CRPs since the number increases exponentially as the size (number of stages) of the PUF increases. Storing the Δ values enables the server to verify the responses of an arbitrary subset of challenges on demand. These model parameters can also be used to

choose preferable (more reliable) challenges which will be discussed in the later chapters.

Chapter 3

Predicting Hard and Soft-Responses of Feed-Forward PUFs using ANNs

3.1 Introduction

The ability to model arbiter PUFs as linear models makes it susceptible to modeling attacks where an attacker tries to build a software clone of the PUF. As a way to make arbiter PUFs more secure, FF PUFs have been proposed [15] [16]. As described in Chapter 2, an additional arbiter (called intermediate arbiter) is used in FF PUFs to determine the response in one of the intermediate stages. This *intermediate response* is then used as a challenge bit for one of the later stages. Note that multiple internal arbiters can also be used to improve security by introducing non-linearity. This chapter studies the unpredictability of PUFs by adopting a black-box approach to model standard, FF and MFF arbiter PUFs using ANNs. Unpredictability is estimated in terms of number of CRPs required to train an accurate model of the PUF. Most of the literature on modeling arbiter PUFs is based on simulations [20]. Even though real data is utilized in some studies [10] [21] [11], they are confined to standard arbiter PUFs. In this work, we present models to predict responses of the three types of arbiter PUFs based on silicon data [1] [15] [16].

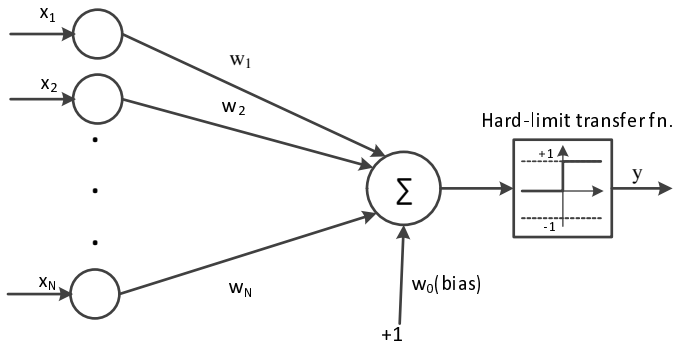
Like any physical circuit, PUFs are also subject to random noise. As a result of uncertainty due to noise and manufacturing processes there may be setup-hold time violations in the arbiter circuit leading to meta-stable outputs. The output or the response in this case is referred to as an *unstable response*. The issue of unreliable responses and possible counter-measures were discussed in [22]. One of the drawbacks of using FF PUFs is that *reliability* is degraded as compared to a standard arbiter PUF of same size [7]. Reliability is the ability to produce a constant response to the same challenge under different environmental conditions. To improve reliability, MFF PUFs were proposed in [7].

This chapter also presents models that are able to accurately predict, for a given challenge, probability of its response being a '1'. This probability is called a *soft-response* and these models are called soft-response models. To the best of our knowledge, this is the first time models are proposed to predict soft-responses. Storing hard-response models in the server has been addressed in prior literature [23] [18]. However, we show that the soft-response models can be stored in the server to detect and discard *unstable responses*. Using these soft-response models, the probability of choosing a stable challenge increases from 89% to more than 98%. Additionally, we show that these soft-response models achieve high accuracy and hence can be used to predict response-bits.

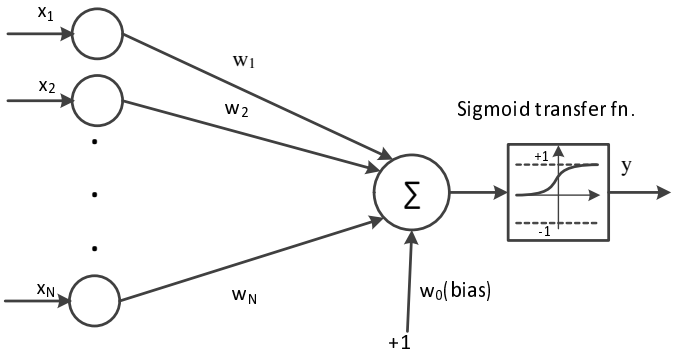
3.2 Artificial Neural Network Models

Machine learning (ML) techniques are computer algorithms used to construct complex input-output mappings. In other words, we use machine learning algorithms to *learn* a model from a given subset of inputs and outputs which makes machine learning a natural approach to model PUFs. In particular, as the response is either 0 or 1, building a predictor is same as building a *binary classifier*. Several ML algorithms have been used in the past to model PUFs. We implement *Artificial Neural Networks* (ANNs) which are mathematical structures inspired from neural connectivity in human brain. The inputs and outputs in this case correspond to challenge vectors and responses, respectively.

Single layer perceptron (SLP) is the simplest version of an ANN and is the basic processing element of any ANN. A perceptron is used to construct a generalized linear



(a) Perceptron with hard-limit transfer function as the activation function



(b) Perceptron with sigmoid transfer function as the activation function

Figure 3.1: Structure of single layer perceptron.

model [24], i.e., a hyperplane in the space of input vectors, of the form given by

$$y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x}), \tag{3.1}$$

where $\mathbf{x}$ is the feature vector derived from inputs and f is called the *activation function*. The model parameters $\mathbf{w}$ are trained using a *training algorithm* such that a *cost function* is minimized. The structure of a perceptron can be observed in Fig. 3.1. Either the activation function shown in Fig. 3.1(a) or Fig. 3.1(b) can be used depending on whether we need a continuous or a discrete output.

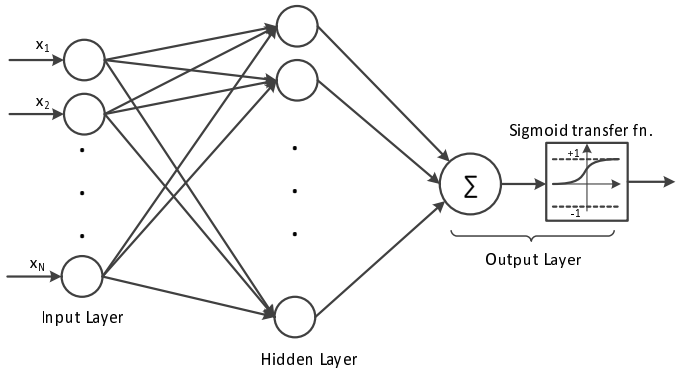
In situations where a linear model is no longer valid, more complex structures called *multilayer perceptrons* (MLPs) are required. Multilayered neural networks are able to

approximate an arbitrary non-linear model by using multiple perceptrons as building blocks. Optimized parameters of the model are calculated by adapting the model iteratively until the cost function reaches a desired value (ideally zero). The model is usually trained using a form of gradient descent based algorithm called *error backpropagation* [24].

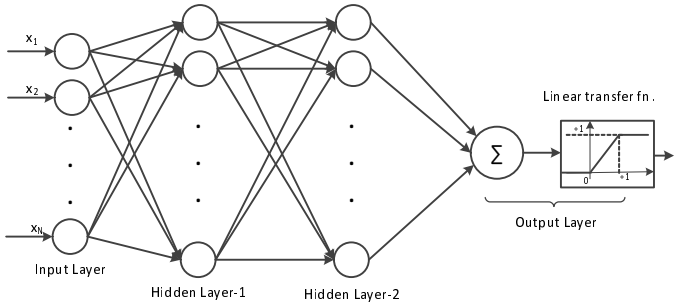
Perceptrons have been used to predict reponses of a linear PUF whose structure is depicted in Fig. 3.1. In case of standard arbiter PUFs (linear PUFs), a hard-limit activation function is used for hard-response prediction and for soft-response prediction a sigmoid activation is used to achieve a continuous output. In case of FF PUFs, we implement MLPs with one hidden layer to predict hard-responses and a sigmoid transfer function is used at the output. Equation for the sigmoid transfer function $f(.)$ is given by $f(y) = 1/(1 + exp(-y))$. The final outputs are thresholded to be 1 if it is more than 0.5 and 0 otherwise. Predicting soft-responses of FF PUFs is a harder problem and requires more complex models. To accomplish this, MLPs with two hidden layers have been employed. In the case of predicting soft-response of FF PUFs, the output layer uses a linear transfer function. As the outputs are probability values, the model outputs are restricted to the range between 0 and 1 during the testing phase. Architecture of MLP with one hidden layer used for hard-response prediction and MLP with two hidden layers used for soft-response prediction of the FF configurations are depicted in Fig. 3.2(a) and 3.2(b), respectively. In all the cases, hidden neurons comprise a hyperbolic tangent sigmoid activation function. Equation for the hyperbolic tangent sigmoid transfer function $g(.)$ is given by $g(y) = 2/(1 + exp(-2y)) - 1$. In order to train the neural networks, a variant of the backpropagation algorithm called *resilient backpropagation* (RProp) [25] has been used for its efficiency.

3.3 PUF Implementation and Data Extraction

The PUFs under study were implemented in 32nm HKMG test chips in three different configurations: standard arbiter based PUF, FF PUF (with one loop) and MFF PUF. Each PUF can be configured as a linear PUF, FF PUF, and MFF PUF by programming two control bits [1]. Each PUF is a 32-stage MUX based PUF. A set of 10,000 and 20,000 unique challenges were randomly generated in order to evaluate standard arbiter PUFs



(a) MLP with one hidden layer used for hard-response prediction of FF and MFF PUFs. The hidden units contain a bias term as an input (not shown above) and have a hyperbolic tangent sigmoid activation function.



(b) MLP with two hidden layers used for soft-response prediction of FF and MFF PUFs. The hidden units contain a bias term as an input (not shown above) and have a hyperbolic tangent sigmoid activation function.

Figure 3.2: Structure of multilayer perceptrons.

and FF PUFs, respectively. About 100,000 repetitive tests were performed for each challenge and the number of 0's and the number of 1's were counted. The soft-response probabilities are based on these 100k measurements. As environmental variations may cause changes in the PUF responses, all the measurements were collected at a source voltage of 0.9 volts and an ambient temperature of 25 degrees Celsius. As a convention, 90-10 thresholding is used to assess stability of the responses. For instance, if the number of 1's is less than 10%, the response is considered a stable 0, and if it is greater than 90% it is considered a stable 1. If the value is between 10 and 90, it is considered an unstable response. Models computed to predict the hard-responses are trained based on stable responses. For each PUF, the corresponding data set is randomly split into 5 equally sized subsets and five-fold cross validation was employed. 3 of the 5 folds were used for training while one fold was used for validation and the remaining fold was used for testing. This was repeated until the entire data set, i.e., all the five folds, were tested.

3.4 Predicting Hard-Responses

Solution to a classification problem involves finding out the best model from a chosen set of models called *hypothesis space*. While dealing with standard arbiter PUFs, the hypothesis space is the set of all linear models or the set of all 33 dimensional hyperplanes. This is because it is established that all standard arbiter PUFs follow the additive linear delay model. In this case, we implement single layer perceptrons with hard-limit transfer function, trained according to *the perceptron algorithm* [24].

In case of the other two configurations, i.e., FF and MFF PUFs, the additive linear delay model - a linear hypothesis space - is no longer valid. Therefore we employ multilayer perceptrons with one hidden layer to train these models. The hidden layer has 30 hidden units. The positions of FF loops are assumed to be unknown to the attacker and cumulative XOR-ed challenges [15] [16] [11] are used as inputs even in this case. The neural networks were adapted to minimize mean square error using *resilient backpropagation* (RProp) [25] as the training algorithm. RProp was chosen for its faster and more stable convergence as compared to standard backpropagation. All hidden neurons have hyperbolic tangent activation functions and the output neuron has

Table 3.1: Results of the Hard-Response Models

Configuration	Chip	PUF	Mean Acc.	Std. Dev.	No. of CRPs
Std.	1	1	99.8%	0.11%	1500
	2	1	99.7%	0.22%	1500
FF	1	1	97.14%	1.35%	10200
		2	96.8%	0.94%	10200
	2	1	96.7%	0.84%	9000
		2	96.8%	0.49%	9600
MFF	1	1	98.32%	0.18%	7800
		2	99.7%	0.26%	7200
	2	1	98.6%	0.18%	7800
		2	98.35%	0.39%	7200

a sigmoid activation. The final outputs are thresholded to be 1 if it is more than 0.5 and 0 otherwise.

3.4.1 Results

Two standard arbiter PUFs have been investigated. For each PUF, models have been trained by increasing the amount of available stable CRPs in each case. For every model, mean accuracy of the five-folds along with its corresponding standard deviations were observed. It was eventually observed that both the PUFs can be modeled almost with certainty (99.8% accuracy) by using 1200 CRPs for training.

Four PUFs (two from each chip) in FF configuration and four PUFs in MFF configuration are analyzed. For each PUF, multiple models were computed by increasing the size of the data set from 2000 CRPs to 17000 CRPs, i.e, the number of CRPs used for training is increased from 1200 to 10200 (60% of the total). In each case, the five-fold cross validation scheme results in five sub-models. Mean of the classification accuracies is used as evaluation metric for each model. The corresponding standard deviations are also observed to verify convergence of the sub-models. In the case of FF PUFs, on

average, all models reach a maximum accuracy of about 97% by using approximately 10000 CRPs for training. High values of accuracy accompanied by very small standard deviation indicates robustness of the models. In the case of MFF PUFs, all PUFs attain a maximum accuracy of approximately 98.5% while reaching a significant value of 98% at only 6000 CRPs. These results for all configurations are summarized in Table 3.1. It can be observed that MFF PUFs are less secure than FF PUFs in spite of the additional non-linearity.

3.5 Predicting Soft-Responses

The objective of a soft-response model is to predict the probability of response being 1, $P(R = 1)$. In this case, the output is no longer a single bit response but a real number between 0 and 1. In other words, this is a regression problem instead of a binary classification problem and hence requires more accurate modeling. Moreover, unstable responses are also taken into consideration while predicting soft-responses. Models are evaluated based on *mean absolute error (MAE)* of the test set and accuracy is defined as 1−MAE.

In case of standard arbiter PUF, we employed a single layer perceptron with *sigmoid* transfer function at the output. In case of FF PUFs, MLPs with one hidden layer were not able to attain accuracies of more than 90%. To this end, we used two-hidden layer MLPs with 30 and 10 units in the first and second hidden layers, respectively. These MLPs were trained using resilient backpropagation. All hidden neurons have hyperbolic tangent activation functions and the output neuron has a linear activation. As the outputs are probability values, the model outputs are restricted to the range between 0 and 1 during the testing phase. Mean and standard deviation of the accuracy values for 5 folds were computed while increasing the training size from 1000 CRPs to 18000 CRPs. Correspondingly the number of CRPs required for training was increased from 600 to 10800 as three of the five folds were used for this purpose.

3.5.1 Results

Soft-response prediction accuracies of the two standard arbiter PUFs are displayed in Fig. 3.3(a). Y-axis represents the mean test accuracy of five folds and X-axis represents

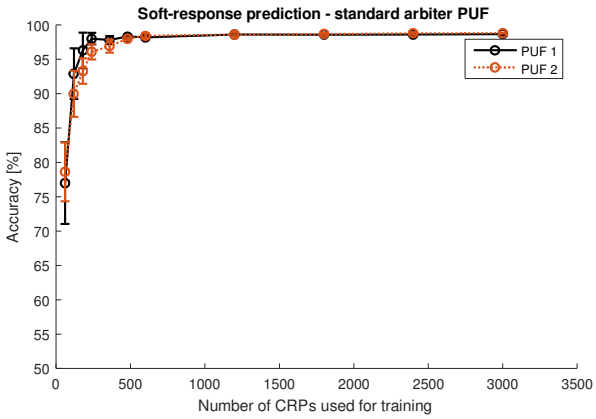
the number of CRPs used for training or three folds. Accuracy is evaluated based on mean absolute error. Vertical bar at each data-point represents the standard deviation of accuracy values across the five folds. It is observed from the plot that both the PUFs achieve a high accuracy of approximately 98.5% accuracy using only 500 CRPs and attain a steady-state accuracy of 98.8% eventually. It can also be noted that along with increased accuracy standard deviation for the five sub-models becomes closer to zero. Fig. 3.3(b) shows that model outputs closely resembles the ground truth.

Results of the four FF PUFs are illustrated in Fig. 3.4. While the accuracy values differ among the PUFs, all models attain more than 95% accuracy. Models of PUF-1 and PUF-2 on the first chip have 95.77% and 96.6% accuracy, respectively, while the models of the PUFs on chip-2 achieve 95.3% accuracy. Overall, we can say that soft-responses of FF PUFs can be predicted with an accuracy of 95-96% using 10,000 training samples.

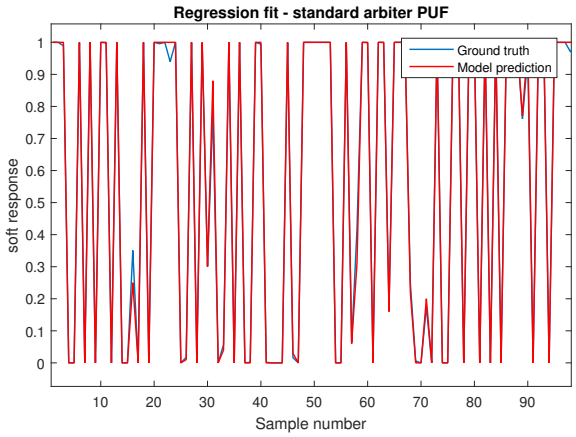
The soft-response models of MFF PUFs are evaluated as shown in Fig. 3.5. On chip-1, models of PUF-1 and PUF-2 eventually attain maximum accuracies of 96% and 97.4%, respectively. In case of chip-2, both the PUFs attain 96.2%. A subset of model outputs for a FF PUF and a MFF PUF are plotted in Fig. 3.6 along with the ground truth as a reference. This depicts the similarity between predicted and actual soft-responses.

3.6 Identifying Unstable Responses

During a typical authentication process, a randomly chosen set of challenges are tested and Hamming distance of the response string is used for validation. A large portion of these challenges could lead to unstable responses. 8-13% of the responses have been observed to be unstable according to the test data. The ability to tell beforehand if a given challenge can produce a stable response could be useful. In a 32-bit challenge, the total number of challenges is 2^{32} or approximately 4.3 billion. Based on experimental data, the percent of stable challenges was found to be about 90%. Thus, the number of stable challenges is approximately 3.86 billion. The unstable challenges are unreliable for authentication. We claim that the predicted soft-responses using our models are accurate enough to validate stability of a response without actually testing them on the

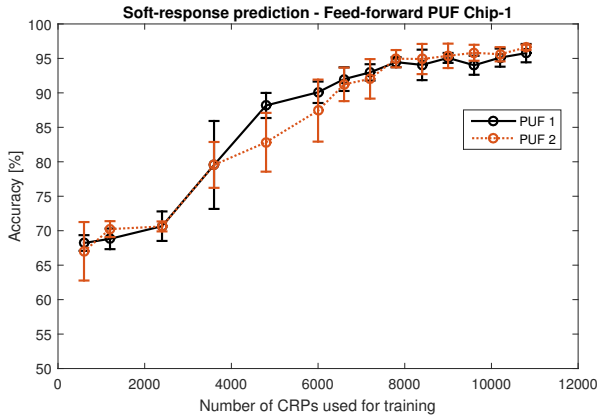


(a) Accuracy vs. number of CRPs used for training.

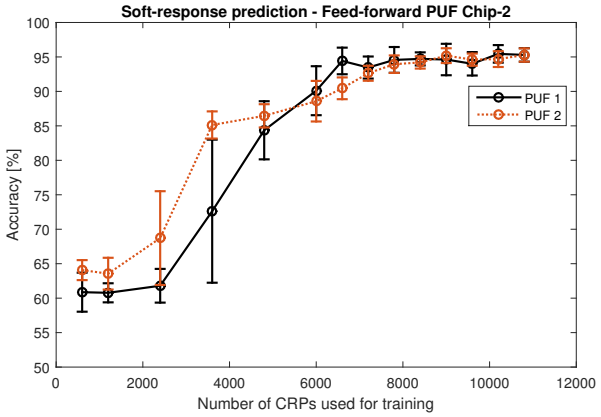


(b) A subset of model outputs and ground truth responses

Figure 3.3: Soft-response model accuracy for std. arbiter PUFs.

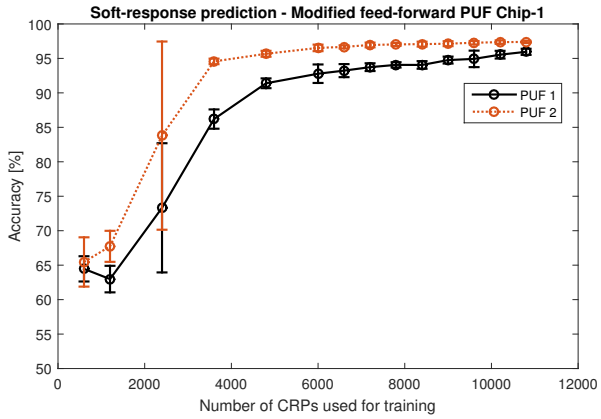


(a) Chip-1

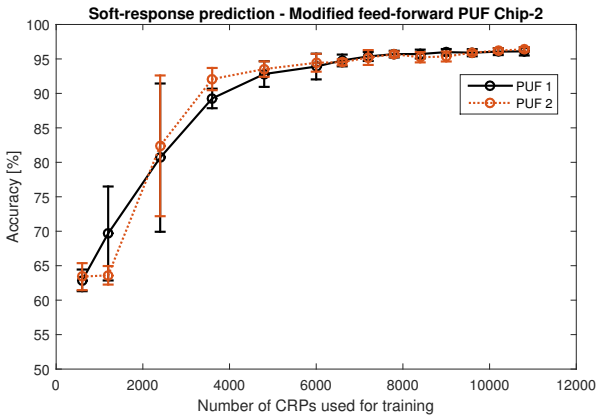


(b) Chip-2

Figure 3.4: Soft-response model accuracy of FF PUF models vs. number of CRPs used for training.

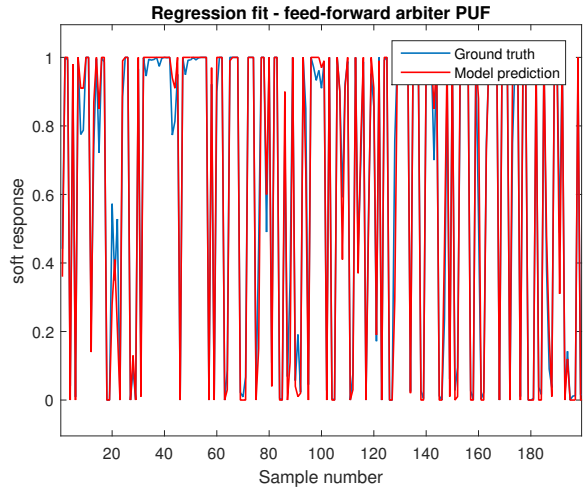


(a) Chip-1

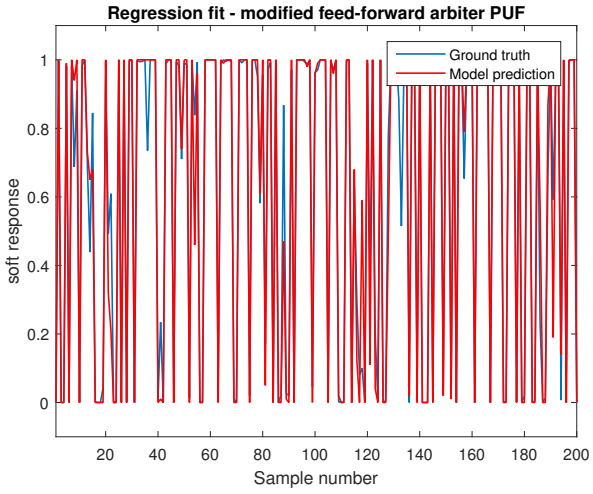


(b) Chip-2

Figure 3.5: Soft-response model accuracy of MFF PUF models vs. number of CRPs used for training.



(a) FF PUF



(b) MFF PUF

Figure 3.6: Soft-response model outputs and ground truth of FF and MFF PUFs for 200 challenges.

Table 3.2: Percent of Stable Responses in the Test Set Before and After the Elimination

Config.	Chip/ PUF	Original set		New set	
		No. of CRPs	% Stable	# CRPs Chosen	% Stable
Std.	1/1	1000	91.4%	905	99.7%
	2/1	1000	91.7%	909	99.1%
FF	1/1	4000	86.9%	3162	98.2%
	1/2	4000	87.25%	3374	98.3%
	2/1	4000	89.52%	3301	98.7%
	2/2	4000	90.85%	3424	98.8%
MFF	1/1	4000	86.95%	3293	97.4%
	1/2	4000	89.75	3536	98.3%
	2/1	4000	91.62	3393	98.9%
	2/2	4000	92.2	3487	99.3%

circuit.

For a given challenge, the first step is to use the models to determine the soft-response which is then used to determine if the response is unstable based on 0.1-0.9 thresholds. This allows the server to issue unstable challenges but not use them for authentication. This approach enhances the security of the PUF. Table 3.2 provides a comparison between proportion of stable responses in the original data and proportion of stable responses in the new set, after the elimination process. Moreover, we argue that these soft-response models can be simultaneously used to predict hard-responses by simply thresholding the predicted soft-responses. It is to be noted that the results are based on evaluation over an independent test set whose sizes are reported in the table (3rd column). For instance, consider PUF-1 in the first chip in FF configuration. 4000 independent random challenges are examined out of which only 86.9% are stable according to the test measurements. Using the soft-response model 3162 (out of the

Table 3.3: Summary of ANN Models Used

Type of PUF	Type of response	#Hidden neurons	Intermediate activation	Final activation
Std.	Hard	0	-	Hard-limit
	Soft	0	-	Sigmoid
FF	Hard	30	tanh	Sigmoid
	Soft	30, 10	tanh	Linear
MFF	Hard	30	tanh	Sigmoid
	Soft	30, 10	tanh	Linear

4000) are found to be stable. By comparing with the ground truth, it turns out that 98.2% of these 3162 CRPs are in fact stable. Using this approach, the proportion of unstable CRPs in the new set falls to less than 3%. This enables the server to issue and verify the response for any randomly chosen challenge at will by storing these models.

3.7 Conclusion

Since a typical application of a PUF is to generate cryptographic keys, unpredictability is a vital security property of any PUF. One way to compare PUFs in terms of their practical unpredictability is to build modeling attacks and compare the ease of attack [26]. In terms of security, it is observed that Standard $<$ MFF $<$ FF; however, this is valid for the specific configurations evaluated and may not be valid for arbitrary configurations. We have shown that hard and soft-response models can be trained for linear and nonlinear PUFs. The ANN models used for training of various PUFs are summarized in Table 3.3. The soft-response models can be used to eliminate unstable challenges and substantially increase the proportion of stable CRPs from about 90% to 99%. Furthermore, these models can be used to predict the stable hard-responses by thresholding the soft-response model outputs, thereby enabling the server to test any random challenge without having to store a large set of CRPs.

Chapter 4

Effect of Loop Positions on Attack-Resistance and Reliability of Feed-Forward PUFs

4.1 Introduction

FF PUFs containing one to five intermediate arbiters [16] [7] are considered in this chapter. Several prior studies [10] [11] [27] [28] have demonstrated vulnerability of FF PUFs to attacks. However, the structure of FF PUFs in these studies was chosen arbitrarily and focused on FF PUFs with loops cascaded with each other or placed separately from each other. In this work, we explore various configurations and determine how changing the location of feed-forward loop inputs and outputs affects the security and reliability. Also, most of the prior studies employ evolutionary strategies to learn a predictive model of a FF PUF. The limitation of this approach is that the attacker is assumed to have the knowledge of the PUF design, i.e., number of feed-forward loops and the location of their inputs and outputs. We do not make any such assumptions in this work and incorporate a black-box approach. We also do not focus on the class of attack strategies that use side-channel information [21] [28] [29].

Unpredictability, reliability and uniqueness are fundamental characteristics of PUFs. Unpredictability ensures resilience against cloning attacks and reliability is a measure

of robustness to environmental noise. Uniqueness makes sure that the outputs of PUFs with identical design and layout produce unique responses. A major limitation of FF PUFs is that reliability is degraded in comparison with standard arbiter PUFs of same size [7]. Reliability is the ability to produce a constant response to the same challenge under different environmental conditions. To improve the reliability, a modified FF PUF structure was proposed in [7]. It has been shown in [27] that modified FF PUFs have a degraded security in spite of having two loops which accounts for more non-linearity. One of the goals of this work is to see if changing the FF loop location could be an effective countermeasure to improve the security of FF PUFs. Additionally, we propose an entropy based metric to estimate the unpredictability of a PUF structure. It is based on the idea of measuring entropy of challenge and response deviations between standard and FF-PUFs. Since, standard arbiter PUFs are known to be predictable, we believe that higher degree of entropy is an indication of higher unpredictability.

We also empirically estimate the reliability of FF PUFs and how it is affected by the choice of feed-forward loop positions. While FF PUFs have been analyzed earlier, *no prior study* has addressed the effect of loop positions on the security and reliability. It is shown that the locations of the arbiters and their outputs can affect the security and reliability of the FF PUF. Additionally, we incorporate a *soft-response thresholding* strategy [27] to identify *stable challenges* and show that reliability can be increased significantly for authentication.

4.2 Feed-Forward PUF Structures

Due to their lack of uniqueness, FF PUFs with one loop are not considered in this study [30]. However, FF PUFs with two loops are considered. In case of *double-loop FF PUFs* there is one intermediate arbiter with two outputs, i.e., the output of the intermediate arbiter is used to replace the select bit of two later stages as shown in Fig. 4.1. The output of the intermediate arbiter located at stage N_1 is fed to stages N_2 and N_3. If the two output stages are adjacent, i.e., if $N_3 = N_2 + 1$, it is referred to as a *modified FF PUF* [7].

Alternatively, we could generate the two intermediate responses from two different arbiters. We denote the input and output stage locations of the first FF loop as N_{11} and

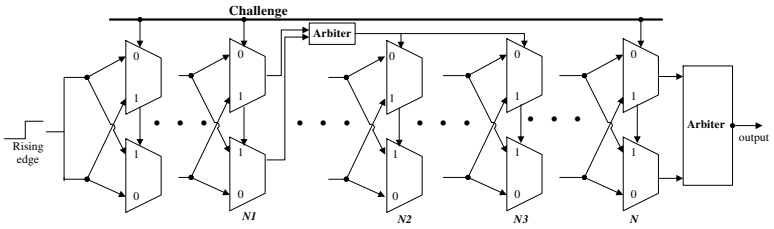
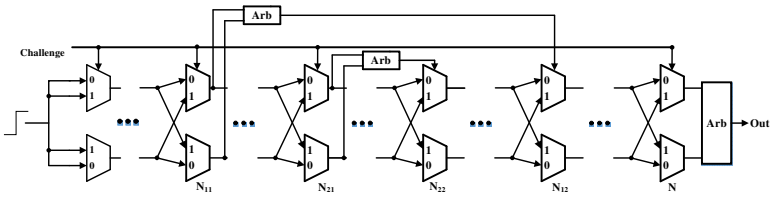


Figure 4.1: Double-loop feed-forward PUF. Intermediate arbiter is located at stage N_1. The intermediate output is fed to stages N_2 and N_3.

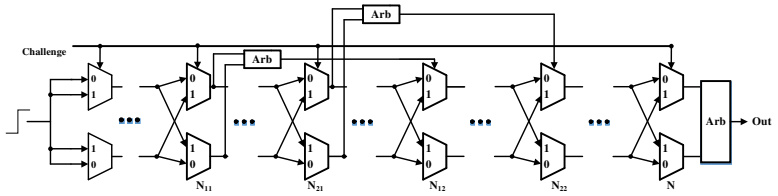
N_{12}, respectively. The input and output positions of the second loop are denoted as N_{21} and N_{22}, respectively (see Fig. 4.2). Depending on the relative positions of these two FF loops, there are four possible configurations: FF overlap, FF cascade, FF separate [31] and FF nested. If one loop is enclosed by the other loop, it is referred to as FF *nested* configuration. If there is at least one stage overlapped between the two loops it is called *feed-forward overlap* structure. If the output stage of the first loop is same as the input stage of the second loop, it is called *feed-forward cascade* structure. When there is at least one stage separation between the output of the first loop and input of the second loop it is called *feed-forward separate*. Cascade and overlap configurations with two loops were shown to exhibit poor uniqueness properties and hence we do not consider these structures in this work. The structures of nested and overlap configurations are depicted in Fig. 4.2.

4.3 Reliability Definition

Depending on the challenge bits, the path delays of the two paths of an arbiter PUF can be similar. Under these circumstances, as a result of uncertainty due to noise and manufacturing processes there may be setup-hold time violations in the arbiter leading to meta-stable outputs. The susceptibility of a PUF to these effects can be characterized by reliability. Reliability of a PUF gives an estimate of how consistent the response is for a given challenge under noise. For each of thr PUFs examples considered, 100 noisy responses are generated. To emulate the effect of environmental and measurement noise, Gaussian noise is added to the delay difference parameter at every stage. To compute the stability of a PUF circuit, the same challenge is provided as input under different values



(a) Feed-forward nested.



(b) Feed-forward overlap.

Figure 4.2: Feed-forward PUFs with two intermediate arbiters.

of noise and the associated responses are recorded. Reliability metric is mathematically defined in terms of *intra-chip variation* (or intra-chip Hamming distance) [6]. Assume R is an n-bit response vector used as a reference and the noisy response-vectors are denoted by R'. An empirical estimate of intra-chip variation, P_{intra}, is given by,

$$P_{intra} = \frac{1}{m} \sum_{i=1}^{m} \frac{HD(R, R'_i)}{n} \tag{4.1}$$

where m is the number of noisy response-vectors that represent different environmental conditions. Reliability can be computed as $(1 - P_{intra}) \times 100\%$.

4.4 Simulation Details

Variants of the additive linear delay model are used to simulate several FF PUF configurations. As mentioned in previous literature, i.i.d. standard normal distributions can be used to model delay difference values at each stage of the PUF [11]. It is known that a Gaussian distribution with non-zero mean and unit variance captures the effect of arbiter [18]. So, we sample arbiter delays from a Gaussian distribution with mean 0.1 and variance 1. We believe that the non-zero mean emulates the effect of arbiter bias. Additionally, noise has been added to all the simulation models to make them more

realistic and to observe its effect. An additive Gaussian noise with zero mean and 5% *noise level* is added to delay difference parameters (Δ_i) at each stage and to the arbiter delay (Δ_{arb}) [7] [32]. Noise level is defined as the ratio of the standard deviation of the noise to the standard deviation of delay differences at each stage.

We consider two FF PUF configurations: one intermediate arbiter with two outputs (double-loop), and two intermediate arbiters each with its own output (two loops). Feed-forward arbiter PUFs with one intermediate arbiter and two intermediate arbiters are shown in Figs. 4.1 and 4.2, respectively. In case of double-loop configurations, the input stage (N_1) is considered to be 15. The 25th, 35th and 45th stages are considered as the candidates for the first output stage of the FF loop (N_2) and the second output stage (N_3) is varied from $N_2 + 1$ to 64 in increments of one position. For configurations with two intermediate arbiters, the first loop is kept constant with $N_{11} = 15$ and $N_{12} = 45$. The second intermediate arbiter is placed at stage $N_{21} = 30$ and the output location (N_{22}) is changed from 31 to 64. If N_{22} is between 31 and 44, it is considered as a FF nested configuration and if it is between 45 and 64, it is considered as an overlap configuration.

4.5 Security Analysis

FF PUF structures presented above are considered for the security analysis. To assess the unpredictability, PUFs in each configuration are attacked using machine learning. For a given configuration, 10 PUF instances (PUFs with identical design) are simulated to ensure consistency in the results. For each PUF instance, multilayered perceptrons with one hidden layer are trained to accurately predict the responses. 20,000 CRPs each are used for validation and testing while varying the number of CRPs required for training. The hidden layer comprised of 50 neurons and 80 neurons for structures with one intermediate arbiter and two intermediate arbiters, respectively. Since an attacker can easily identify the noisy challenges, we only used the CRPs that have at least 90% consistency in the presence of noise. As more data are used for training, the prediction accuracy naturally increases until it reaches a maximum value after which it gets saturated. Convergence of training, validation and testing errors are examined to avoid over-training. All the models achieved more than 92% accuracy. Therefore, 8%

tolerance level is chosen as the threshold to decide if a model is accurate. Minimum Number of CRPs (approximated to the nearest multiple of 5000) required to reach 92% test accuracy is considered as the evaluation metric to compare *unpredictability* of various configurations studied.

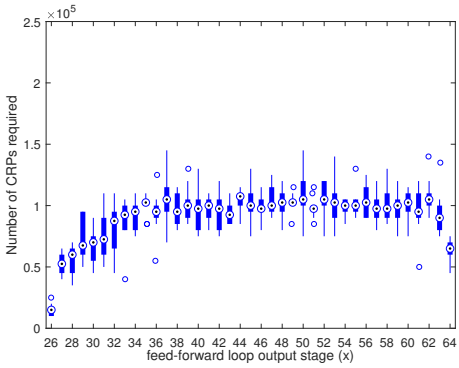
4.5.1 Feed-Forward PUFs with One Intermediate Arbiter

Double loop PUFs with $N_1 = 15$ have been trained with varying values of the first output stage (N_2) and the second output stage (N_3). The number of CRPs required to attain 92% accuracy has been recorded for 10 instances in each design. For a given value of N_1 and N_2, box-plots illustrating the effect of changing the value of N_3 are shown in Fig. 4.3. In Fig. 4.3(a), it can be observed that as the value of N_3 increases, training data size required to train an accurate model increases until a certain position, seen to be the 36th stage in this case. After the 60th stage, as we get closer to the output stage, we need less training information. We can further observe that by just selecting a better choice of feed-forward loop positions we can increase the number of required CRPs to train an accurate model by more than 7 times: from 15,000 ($N_2 = 25$ and $N_3 = 26$ - modified FF PUF) to 107,500 ($N_2 = 25$ and $N_2 = 50$).

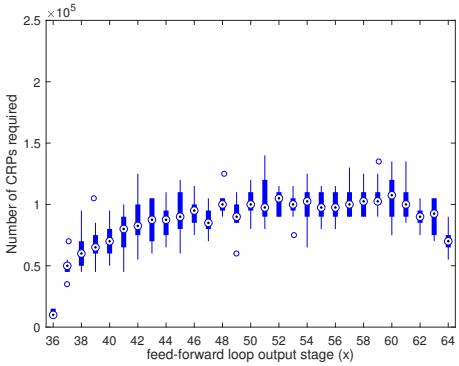
A similar "inverted-U" trend can be observed in all three cases. It can be observed that the change in output position of the feed-forward loop has significant effect on the security of the PUF. These results indicate that the feed-forward loop output stage should be chosen as far as possible from the input location of the loop as well as the final stage for better attack-resistance.

4.5.2 Feed-Forward PUFs with Two Intermediate Arbiters

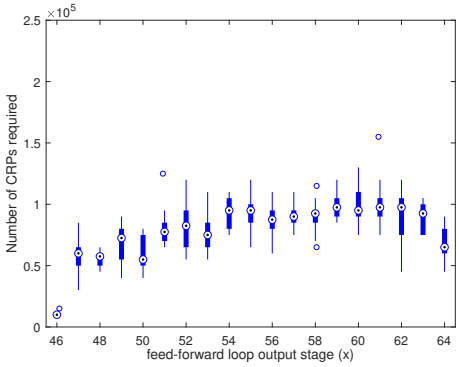
ANN models for feed-forward overlap (Fig. 4.2(a)) and nested (Fig. 4.2(b)) configurations are trained using multilayered perceptrons with one hidden layer. These models were relatively more complex and required 80 neurons instead of 50 to reach good prediction accuracies. The size of the training data required to attain 92% accuracy was measured. The effect of changing the second FF loop output stage on this value is shown in Fig. 4.4(a) and Fig. 4.4(b) for nested and overlap structures, respectively. It can be noted that the observed values imply that the PUFs with 2 intermediate arbiters are



(a) N_1=15, N_2=25 and N_3=x.



(b) N_1=15, N_2=35 and N_3=x.



(c) N_1=15, N_2=45 and N_3=x.

Figure 4.3: Double-loop PUFs. Minimum number of CRPs required to predict with 92% accuracy *vs.* position of the FF loop output (N_3). Each box represents 10 PUFs.

more secure than double-loop PUFs. More importantly, similar to the previous case, we can note that the output positions closer to other FF loop inputs/outputs require relatively less training data.

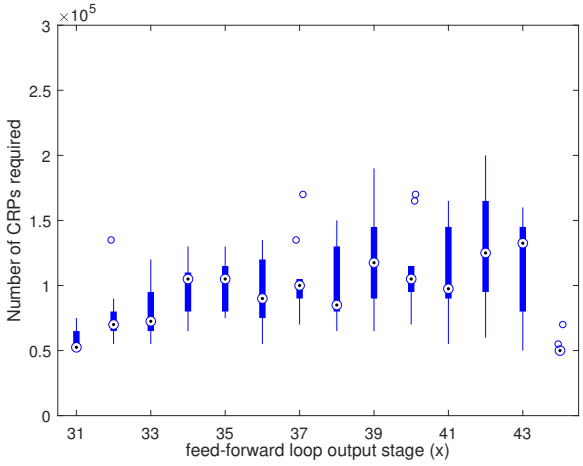
4.6 Entropy of Challenge-Response Deviation Metric

Unpredictability of a PUF is a measure of complexity of the input output mapping between inputs and ouputs of a predictive model. It is known that linear PUFs can be easily learned [10] [11] [18]. So, the further the function of a FF-PUF deviates from the function of a linear PUF, the harder it becomes to learn an accurate predictive model. With this premise in mind, we propose an entropy based unpredictability metric to estimate the effect of FF loop position for a given FF structure. The *entropy of challenge-response deviation* metric is based on two terms: the *entropy of the XOR-ed challenge deviation* and the *entropy of the response deviation.*

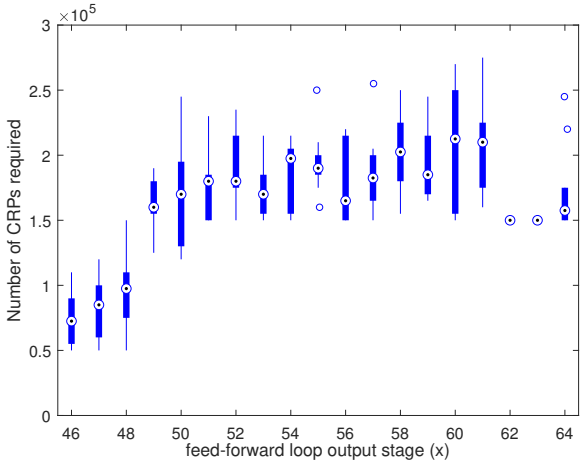
4.6.1 Definition

The first metric, δ^X_{dev}, is based on comparing the input vector of a given FF-PUF (X_{FF}) to that of a linear PUF (X_{lin}) with same delay difference parameters. It is known that a standard MUX PUF can be modeled as a linear function of cumulative XOR-ed challenges (X) as described in (2.1). Assume a standard MUX PUF is converted to a feed-forward PUF such that the intermediate output replaces the challenge-bit of stage N_2. For a given challenge, there are only two possible ways in which this conversion can alter the values of a certain bit in X. Case-1: The intermediate arbiter output is same as the challenge-bit at N_2 and none of the XOR-ed challenge-bits is affected. Case-2: The intermediate arbiter output is different from the challenge-bit. This leads to a more complex mapping between the input vector X and the response, which makes it harder to learn a predictive model.

Let us define a Bernoulli random variable X_{dev} that indicates whether a given input-bit (X_i) deviates (flips) from its original value due to change in the structure from linear to feed-forward. Entropy quantifies the degree of randomness in X_{dev}. For a given challenge (C), the probability of deviation can be computed by using normalized Hamming distance between X_{lin} and X_{FF}, i.e., $P^X_{dev} = \frac{HD(X_{lin}, X_{FF})}{N}$ where N is the



(a) Nested configuration. N_{11}=15, N_{12}=45, N_{21}=30 and N_{22}=x



(b) Overlap configuration. N_{11}=15, N_{12}=45, N_{21}=30 and N_{22}=x

Figure 4.4: Nested and Overlap FF PUFs. Minimum number of CRPs required to predict with 92% accuracy *vs.* position of the feed-forward loop output stage (N_{22}). Each box represents 10 PUFs.

number of stages. The *XOR-ed challenge deviation* metric (δ^X_{dev}) is defined as the mean entropy of X_{dev} over a set of challenges. This can be expressed as

$$\delta^X_{dev} = \frac{1}{M} \sum_{C \in \mathcal{M}} -P^X_{dev} log_2(P^X_{dev}) - (1 - P^X_{dev}) log_2(1 - P^X_{dev}), \tag{4.2}$$

where $\mathcal{M}$ is a randomly chosen subset of challenges and M is the number of challenges.

Similarly, a metric can be defined to measure the dissimilarity between the response signatures of a linear PUF and a FF-PUF. Define a Bernoulli random variable R_{dev} which indicates whether or not the response of the FF-PUF is different from that of the linear PUF for the same challenge. The probability of response changing can be computed as $P^R_{dev} = \frac{HD(R_{lin}, R_{FF})}{M}$ where R_{lin} and R_{FF} are the response signatures of linear and FF-PUFs, respectively and M is the number of bits in the response signature. The *response deviation metric* (δ^R_{dev}) is then computed as the entropy of R_{dev}. This can be expressed as

$$\delta^R_{dev} = -P^R_{dev} log_2(P^R_{dev}) - (1 - P^R_{dev}) log_2(1 - P^R_{dev}), \tag{4.3}$$

The unpredictability metric (δ_{dev}) is the average value of δ^X_{dev} and δ^R_{dev}, i.e., $\delta_{dev} = \frac{\delta^X_{dev} + \delta^R_{dev}}{2}$.

4.6.2 Results

All configurations evaluated for the security analysis are considered here. δ_{dev} is computed for 100 instances for each of the double-loop, nested and overlap configurations. The set $\mathcal{M}$ consists of 10,000 randomly chosen challenges. δ_{dev} plotted as a function of varying FF loop output stage is presented in Fig. 4.5(a) for the case of double loop PUFs. Fig. 4.5(b) depicts the results for nested and overlap configurations. Each value is the mean of 100 instances. All the 100 instaces were observed to have similar values: the variance of δ_{dev} for these 100 PUF instances is observed to be in the order of 10^{-6}. In other words, the metric is invariant to the parameters of the PUF and only depends on the design. It can be observed that δ_{dev} increases as the value of FF loop output stage is increased upto a certain point and then starts decreasing as we get closer to the final stage. The peak value is attained when the output position is chosen away from the final stage and other inputs/outputs of FF loop(s). The inverted-U trend seen

here roughly explains the results in Fig. 4.3 and Fig. 4.4. Thus, this metric can be used to estimate and compare the effect of changing FF loop positon on the security of FF-PUFs. It is to be noted that this metric is limited to estimating and comparing the unpredictability among different PUFs with same structure but varying FF loop locations. It cannot be used to compare arbitrary FF-PUF structures and hence cannot be considered as a universal metric for unpredictability of FF-PUFs.

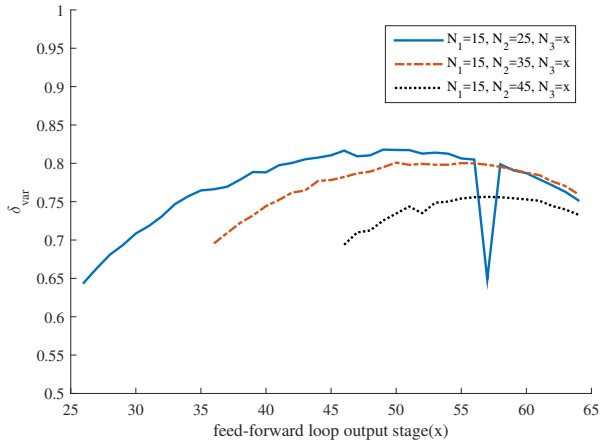
4.7 Reliability Analysis

4.7.1 Effect of Changing FF Loop Output Location

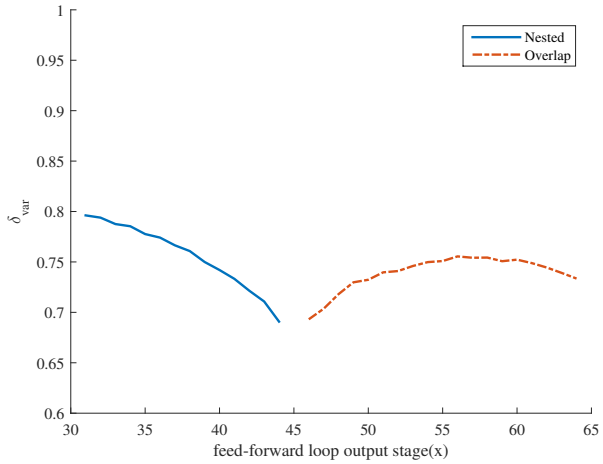
For each configuration, i.e., for a given PUF design, 100 PUF instances are evaluated using 1000 CRPs. The mean and standard deviation across the 100 PUF instances are recorded. For double loop configurations, reliability as a function of feed-forward loop output position is shown in Fig. 4.6(a). It can be seen that the reliability decreases as the value of N_1 or N_2 is increased. The mean reliability decreases approximately linearly as the second output stage (N_2) is pushed further towards the right. The standard deviation values were observed to be always less than 0.2%. Reliability values of nested and overlap FF PUFs are shown in Fig. 4.6(b). The expected linearly decreasing trend is observed again. This trend is in accordance with previous statistical analysis [7].

4.7.2 Soft-Response Thresholding

When the same challenge is applied to a PUF multiple times, it may not result in the same output due to the effect of noise. As a consequence of the path delays in the circuit, some challenges are more prone to generate an inconsistent response. By identifying those challenges, we can increase the reliability during authentication. For a given challenge, soft-response is defined as the probability that the response is 1 ($Pr(R = 1)$) under environmental variations. Empirically, the soft-responses were computed by applying the same challenge 100 times under the presence of noise. As a convention, we use 90% as a threshold to measure the consistency of a response. That is, if the response is 0 or 1 in at at least 90% of the cases, it is considered a *stable response*. Otherwise, it is referred to as an *unstable response*. Therefore if $Pr(R = 1)$ is less than 10%, the

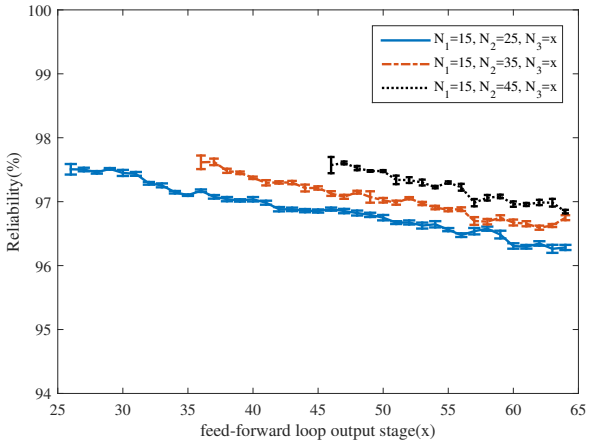


(a) Double-loop PUFs.

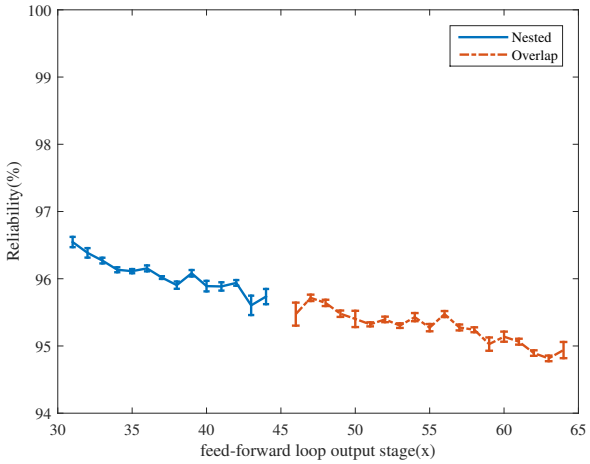


(b) Nested and overlap PUFs. N_{11}=15, N_{12}=45, N_{21}=30 and N_{22}=x

Figure 4.5: δ_{dev} plotted as a function of varying FF loop output stage for double-loop, nested and overlap configurations. The X-axis represents the position of the feed-forward loop output stage. Each point is a mean value of 100 PUFs.



(a) Double-loop PUFs.



(b) Nested and overlap PUFs. N_{11}=15, N_{12}=45, N_{21}=30 and N_{22}=x

Figure 4.6: Mean reliability plotted as a function of varying FF loop output stage for double-loop, nested and overlap configurations. The error bar represents standard deviation. All values are computed across 100 PUFs.

response is considered a stable 0, and if it is greater than 90% it is considered a stable 1. If the value is between 10% and 90%, it is considered an unstable response. Soft-responses of several FF PUFs were computed by assigning each challenge 100 times. A subset of challenges was then chosen by thresholding the soft-responses using 90-10 thresholding. The challenges that generate stable responses are referred to as *stable challenges*. Thus, stable challenges can be identified and used to increase reliability of authentication.

4.7.3 Adding More FF Loops

We know that adding more loops and more internal arbiters adds complexity to the PUF models making them more resistant to modeling attacks. But this comes at the cost of degraded reliability. So, we extend the above reliability analysis to analyze FF PUFs with more than 2 loops. FF PUFs with multiple loops and varying number of intermediate arbiters were simulated in overlap and nested configurations and their mean reliability values were computed. Further, soft-response thresholding is applied to these configurations under different noise levels (5%, 10% and 15%) and the reliability was computed before and after thresholding i.e., using 1000 randomly chosen challenges and by using 1000 stable challenges. These results are presented in Tables 4.1-4.3.

In case of FF PUFs with a single intermediate arbiter, placing the FF loop outputs closer to each other results in higher reliability compared to spreading them apart. In spite of adding more loops, the change in reliability is not significant since the intermediate response is generated by the same arbiter. Therefore, multiple intermediate arbiters were used to generate independent intermediate responses. We know that multiple arbiters can be configured in overlap or nested fashion (see Fig. 4.2). In general, it can be noticed that adding more loops makes the circuit more susceptible to noise which leads to less reliability and less proportion of stable challenges. However, less proportion of stable challenges is not directly a concern as the total number of available challenges is huge for arbiter PUFs and increases exponentially with the number of stages. For 64-bit PUFs, assuming a meager 10% of the challenges are stable, the server still has 1.8×10^6 trillion (10% of 2^{64}) stable challenges. For nested and overlap FF PUFs, the results show that reliability can be improved by replacing the FF loops with *modified* FF loops. For example, FF PUF with 5 intermediate arbiters and

5 loops in overlap configuration has 84.8% reliability (15% noise level). If the 5 loops are replaced by modified FF loops, i.e, the same output is provided to two consecutive stages, the reliability increases to 92%. These results also illustrate that soft-response thresholding can significantly increase the reliability to more than 96% in every case. Considering nested configuration with 5 loops, we can see that mean reliability is 81.6% (15% noise) and 57% of the challenges are stable. By choosing these stable challenges for authentication, reliability can be increased to 96.3%.

4.8 Discussion and Conclusion

An empirical analysis of the effect of loop positioning in FF PUFs on their and attack-resistance and reliability is presented in this chapter. We show that location of the loop can have significant effect on attack-resistance. In case of double loop PUFs, the number of CRPs required by an attacker to train an accurate model can be increased from 15,000 ($N_2 = 25$ and $N_3 = 26$ - modified FF PUF) to 107,500 ($N_2 = 25$ and $N_3 = 50$), i.e., more than 7 times, by just changing the location of feed-forward loop output stage (N_3). Similar observations can be made for nested and overlap configurations. In general, output stages for FF loops should be chosen away from the input stages and other output stages to attain better security characteristics. We believe that FF PUFs can be used as a better alternative to standard arbiter PUFs as components of an XOR PUF since FF PUFs are inherently nonlinear and a linear approximation similar to the cases of standard XOR PUFs or Interpose PUFs is not applicable [11] [33]. The above observations can play an important role in appropriately choosing the design of FF PUFs for better reliability and security.

We also propose an entropy based metric to determine the more secure FF loop positions by comparing the inputs and outputs of a FF-PUF to that of a linear PUF. This enables designers to enhance the security of FF-PUFs by choosing better locations for input and output stages of FF loops without having to train attack models. The drawbacks of this metric are that it is a rough estimate and also is limited to comparing FF-PUFs with similar structures but different FF loop positions. Thus, it cannot be considered as a universal unpredictability metric. The following conclusions can be made regarding the reliability of FF PUFs. First, as FF output stage is chosen further

away, the value of reliability decreases. This can be attributed to the fact that FF loop output has more impact on the final response as it is positioned closer the final stage. Second, as more intermediate arbiters are added, the PUFs tend to be more susceptible to noise in general. Third, replacing FF loops with modified FF loops can increase the reliability. The loss of reliability can be addressed by identifying unstable challenges. We show that applying soft-response thresholding can effectively increase the reliability to more than 96%. Since this is based on simple thresholding, the added complexity is negligible.

Table 4.1: Percent of Stable Challenges and Reliability Before and After Thresholding for FF PUFs with One Intermediate Arbiter. Threshold = 90%.

Num. of Arb	Num. of Loops	FF loop locations	Noise Level = 5%			Noise Level = 10%			Noise Level = 15%		
			% Stable	Rel. Before	Rel. After	% Stable	Rel. Before	Rel. After	% Stable	Rel. Before	Rel. After
1*	2	$15 \rightarrow 25, 26$	94.70	97.62	99.60	89.58	95.35	99.16	84.60	93.07	98.73
1	2	$15 \rightarrow 25, 30$	94.15	97.37	99.55	88.59	94.90	99.04	83.18	92.43	98.60
1	4	$15 \rightarrow 25, 26, 30, 31$	94.54	97.50	99.59	89.34	95.24	99.15	84.25	92.94	98.61
1	4	$15 \rightarrow 25, 35, 45, 55$	93.10	96.98	99.47	86.46	93.99	98.89	80.06	91.06	98.25
1	8	$15 \rightarrow 25, 26, 35, 36, 45, 46, 55, 56$	94.25	97.45	99.56	88.71	94.99	99.07	83.43	92.60	98.62
1	8	$15 \rightarrow 25, 30, 35, 40, 45, 50, 55, 60$	93.12	96.91	99.50	86.53	93.95	98.91	80.15	91.11	98.26
1	16	$15 \rightarrow 25, 26, 30, 31, 35, 36, 40,$ $41, 45, 46, 50, 51, 55, 56, 60, 61$	93.96	97.31	99.54	88.11	94.60	99.06	82.51	92.21	98.53

* The intermediate arbiters are connected to modified FF loops.

Table 4.2: Percent of Stable Challenges and Reliability Before and After Thresholding for FF PUFs with Multiple Intermediate Arbiters in Overlap Configuration. Threshold = 90%.

Num. of Arb	Num. of Loops	FF loop locations	Noise Level = 5%			Noise Level = 10%			Noise Level = 15%		
			% Stable	Rel. Before	Rel. After	% Stable	Rel. Before	Rel. After	% Stable	Rel. Before	Rel. After
2	2	$10 \rightarrow 20; 15 \rightarrow 30,$	92.15	96.49	99.37	84.88	93.11	98.71	78.18	90.37	98.11
2*	4	$10 \rightarrow 20, 21; 15 \rightarrow 30, 31$	94.47	97.53	99.55	89.30	95.19	99.14	84.26	92.89	98.65
3	3	$10 \rightarrow 20; 15 \rightarrow 30; 25 \rightarrow 40$	90.47	95.63	99.22	81.92	92.02	98.46	74.29	88.58	97.73
3*	6	$10 \rightarrow 20, 21; 15 \rightarrow 30, 31$ $25 \rightarrow 40, 41$	94.15	97.37	99.56	81.92	91.97	98.45	83.74	92.64	98.61
4	4	$10 \rightarrow 20; 15 \rightarrow 30; 25 \rightarrow 40$ $35 \rightarrow 50$	88.23	94.68	99.07	78.10	90.21	98.13	69.29	86.32	97.26
4*	8	$10 \rightarrow 20, 21; 15 \rightarrow 30, 31;$ $25 \rightarrow 40, 41; 35 \rightarrow 50, 51;$	93.66	97.18	99.52	78.10	90.39	98.16	82.94	92.33	98.53
5	5	$10 \rightarrow 20; 15 \rightarrow 30; 25 \rightarrow 40;$ $35 \rightarrow 50; 45 \rightarrow 55$	86.39	93.81	98.90	75.20	89.07	97.86	65.81	84.76	96.87
5*	10	$10 \rightarrow 20, 21; 15 \rightarrow 30, 31;$ $25 \rightarrow 40, 41; 35 \rightarrow 50, 51;$ $45 \rightarrow 55, 56$	93.17	96.94	99.49	87.43	94.36	99.00	82.23	92.01	98.43

* The intermediate arbiters are connected to modified FF loops.

Table 4.3: Percent of Stable Challenges and Reliability Before and After Thresholding for FF PUFs with Multiple Intermediate Arbiters in Nested Configuration. Threshold = 90%.

Num. of Arb	Num. of Loops	FF loop locations	Noise Level = 5%			Noise Level = 10%			Noise Level = 15%		
			% Stable	Rel. Before	Rel. After	% Stable	Rel. Before	Rel. After	% Stable	Rel. Before	Rel. After
2	2	$25 \rightarrow 45; 30 \rightarrow 40$	91.38	96.11	99.32	83.56	92.72	98.63	76.59	89.59	98.04
2*	4	$25 \rightarrow 45, 46; 30 \rightarrow 40, 41$	94.43	97.48	99.57	89.21	95.18	99.16	84.20	92.89	98.66
3	3	$20 \rightarrow 50; 25 \rightarrow 45; 30 \rightarrow 40$	88.89	94.92	99.10	79.34	90.83	98.36	71.23	87.36	97.53
3*	6	$20 \rightarrow 50, 51; 25 \rightarrow 45, 46;$ $30 \rightarrow 40, 41$	94.04	97.30	99.52	88.64	94.94	99.10	83.55	92.56	98.60
4	4	$15 \rightarrow 55; 20 \rightarrow 50; 25 \rightarrow 45;$ $30 \rightarrow 40$	85.97	93.75	98.95	74.45	88.83	97.91	64.92	84.71	97.02
4*	8	$15 \rightarrow 55, 56; 20 \rightarrow 50, 51;$ $25 \rightarrow 45, 46; 30 \rightarrow 40, 41;$	93.72	97.15	99.48	88.17	94.71	99.00	82.97	92.36	98.56
5	5	$10 \rightarrow 60; 15 \rightarrow 55; 20 \rightarrow 50;$ $25 \rightarrow 45; 30 \rightarrow 40;$	82.68	92.24	98.63	68.74	86.29	97.41	57.62	81.61	96.34
5*	10	$10 \rightarrow 60, 61; 15 \rightarrow 55, 56;$ $20 \rightarrow 50, 51; 25 \rightarrow 45, 46;$ $30 \rightarrow 40, 41$	93.48	97.05	99.50	87.83	94.52	98.99	82.57	92.19	98.51

* The intermediate arbiters are connected to modified FF loops.

Chapter 5

Feed-Forward XOR PUFs: Attack-Resistance and Reliability Analysis

5.1 Introduction

This chapter evaluates XOR PUFs in terms of their resilience against attacks (security) and their resilience against noise (reliability). Authors in [10, 11, 34, 35] have presented methods to attack XOR PUFs using machine learning. One major issue with XOR PUFs is that their models can be expressed as linear decision boundaries [11] which makes it easier to learn an accurate model. Also, evolutionary strategy based attacks were presented in [34, 36] which are based on the assumption that the structure of each component PUF, i.e., the number of model parameters are known to the attacker. A black-box approach is adopted to modeling XOR PUFs where the attacker does not have any access to any side-channel information. Additionally, the existing studies [10,11,34,35,37] are only limited to XOR PUFs with standard arbiter PUFs as elements, called *standard XOR PUFs*. In this chapter, we simulate and analyze XOR PUFs with FF PUFs as elements, called *feed-forward XOR PUFs* (FFXOR PUFs), and assess their security and reliability in comparison with standard XOR PUFs. Since FF PUFs are inherently nonlinear, a linear approximation similar to the cases of standard XOR PUFs

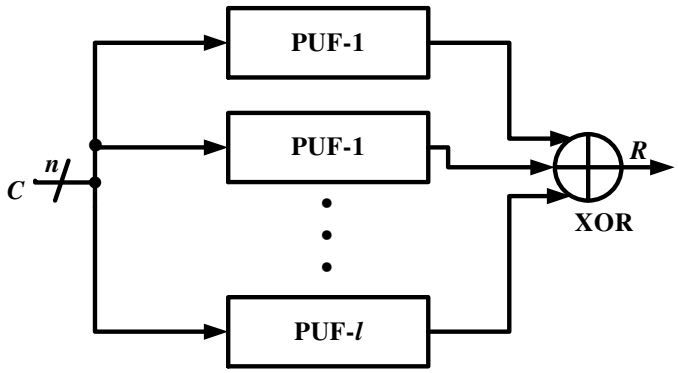


Figure 5.1: XOR arbiter PUF. Each component could be either standard or FF PUF.

or Interpose PUFs [33] is not applicable. Artificial neural networks (ANNs) [24] are used to learn the models and the security of a PUF is estimated in terms of the amount of data an attacker requires to learn an accurate predictive model. It is important to note that our work is limited to modeling attacks and does not focus on side-channel attacks [38–40].

5.2 XOR Arbiter PUFs

To make arbiter PUFs less susceptible to modeling attacks, the idea of XORing outputs of multiple arbiter PUFs to generate the final response was suggested in [12]. The structure of an XOR PUF with l levels, i.e., with l MUX PUFs as *elements* is depicted in Figure 5.1. The same n-bit challenge vector C is provided as input to all the component n-stage arbiter PUFs. XORing the intermediate responses adds non-linearity to the system making it more difficult to learn a model.

5.3 Setup

For security analysis, XOR PUFs and FFXOR PUFs with number of levels $l = 2$ to $l = 8$ are simulated and their neural network models are trained. 32-stage MUX PUFs were used as the components for each XOR PUF. FF PUFs containing a feed-forward loop (see Figure 2.2) whose intermediate response is computed at the 15th stage (N_1) and is fed into the 25th stage (N_2) were used as components of FFXOR PUFs. XOR

PUFs with $l \geq 8$ were found to be attack resistant. For each of these configurations, data for 10 PUF instances were generated. For each instance, 1.2 million challenge-response pairs (CRPs) are extracted. 1 million CRPs are used to train the model and 100,000 CRPs are used for testing and validation each.

For reliability analysis, noisy responses were generated by adding Gaussian noise with varying standard deviation to the delay difference at each stage. As a result, the distribution is modified to $\Delta_i \sim \mathcal{N}(0,1) + \mathcal{N}(0,(\sigma_n)^2)$. Here, σ_n denotes the standard deviation of the noise. Once again, XOR PUFs with 2 levels to 8 levels were evaluated for the reliability analysis. To emulate the effect of environmental and measurement noise, Gaussian noise with zero mean and a standard deviation corresponding to 5%, 10% and 15% noise level is added to the delay difference parameters (Δ_i) at each stage and to the arbiter delay (Δ_{arb}) [7,32]. *Noise level* is defined as the ratio of the standard deviation of the noise to the standard deviation of delay differences at each stage. For a given design configuration, 100 PUF instances were generated. 1000 randomly chosen unique challenges are used to extract a 1000-bit response-vector for each instance. 100 noisy response vectors are generated for each of the 100 PUF instances for the purpose of computing reliability. FFXOR PUFs with multiple loops were also analyzed.

5.4 Security Analysis

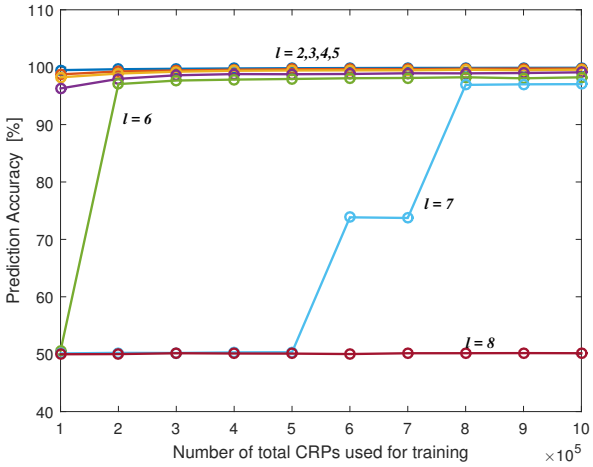
For every PUF instance, several models were trained by increasing the size of the training set from 100k to 1M and the prediction accuracy on an independent test-set is reported. The randomly chosen test challenges are kept consistent across all the models to have an unbiased comparison. For configurations with standard arbiter PUFs as components, mutlilayer perceptrons with 2 hidden layers were implemented with 60 and 30 neurons in each hidden layer, respectively. Since ANNs with 2 hidden layers were unable to accurately model FFXOR PUFs, 3 hidden layer structures with 120, 30 and 15 neurons, respectively, were trained. The training and testing errors are compared for each model to make sure that there is no over-training.

5.4.1 Results

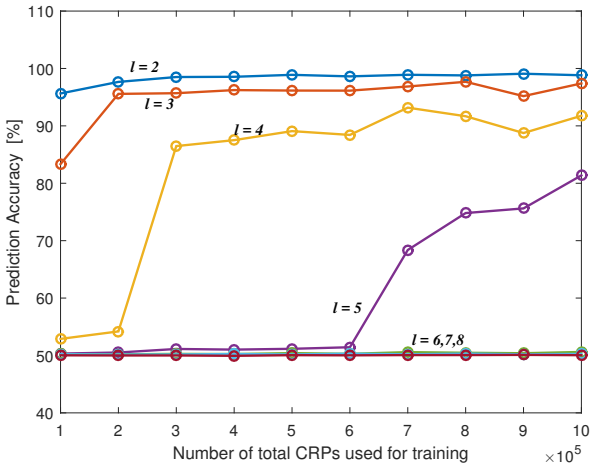
Prediction accuracies of standard arbiter XOR PUFs were evaluated by increasing the size of the training set from 100,000 CRPs to 1 million CRPs. Figure 5.2(a) and Figure 5.2(b) show how prediction accuracies of attack models vary with respect to training size for standard and FFXOR PUFs, respectively. The number of levels is varied from $l = 2$ to $l = 8$. As expected, the prediction accuracy increases with increase in the training size and smaller XOR PUFs are easier to model as compared to higher number of XOR inputs (say $l = 7, 8$). Each accuracy value is the median of 10 PUF instances. In case of the standard XOR PUFs, it can be observed that for $l = 2$ to $l = 7$, models can be trained with more than 95% accuracy. For $l = 8$, the model is stuck at 50% accuracy. In case of FF PUFs, $l = 2, 3, 4$ achieve more than 90% accuracy while the models for PUFs with $l \geq 5$ are not as accurate. The maximum accuracy values attained using 1 million CRPs as a function of number of component PUFs in the circuit are presented in Figure 5.3. The model accuracies of standard and feed-forward XOR PUFs can be observed. Each value is presented in the form of a box-plot of 10 instances.

5.4.2 Discussion

It can be observed that FFXOR PUFs require significantly more resources to attack as compared to standard XOR PUFs. This follows from the fact that the number of ANN parameters required to train standard arbiter PUFs is $33 \times 60 \times 30 \times 1 = 59,400$ and for FF PUFs, it is $33 \times 120 \times 30 \times 15 \times 1 = 1,782,000$. Note that the input layer has 33 parameters, including the bias, since the input size is 32 bits. This costs more computational resources and processing time for the attacker. This difference is expected to be more significant when dealing with XOR PUFs with more than 128 or 256 bit-challenges. The results in Figure 5.3 show that the standard XOR PUFs can be trained with more than 95% prediction accuracy up to 7 levels while FFXOR PUFs with 5 levels only attain a maximum prediction accuracy of 80% and the models for more than 5 levels are just as good as random guessing (50% accuracy). These observations offer an important insight that more secure XOR PUFs that have much less area overhead can be designed by replacing standard arbiter PUFs with FF PUFs as elements. This is especially important as practical PUFs are expected to be lighter,



(a) Standard XOR arbiter PUFs



(b) Feed-forward XOR arbiter PUFs

Figure 5.2: Prediction accuracy *vs.* training size of ANN models for standard XOR PUFs and FFXOR PUFs. The number levels is varied from $l = 2$ to $l = 8$. The FF PUFs contain one loop from $N_1 = 15$ to $N_2 = 25$. The number of stages (N) is 32.

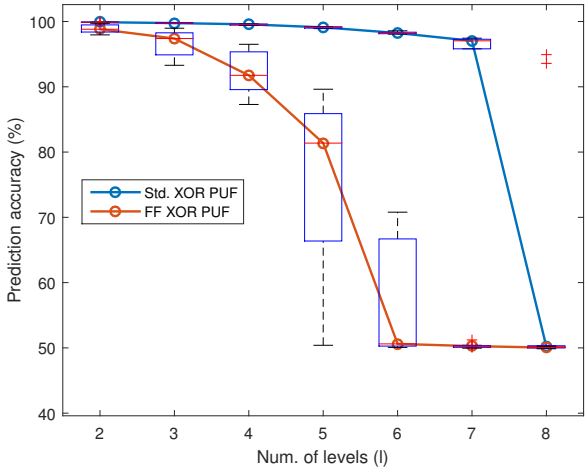


Figure 5.3: Prediction accuracy *vs.* number of levels (l) for 32-stage standard XOR PUFs and FFXOR PUFs. Accuracy values are presented as a box-plot of 10 instances.

i.e., have low hardware resources thus requiring less computations and lower power consumption.

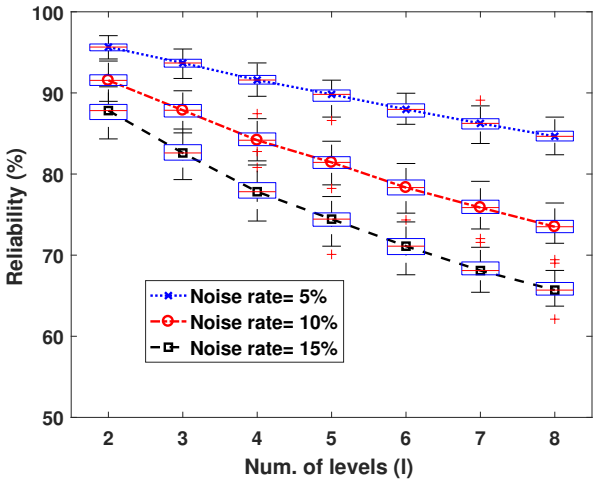
Logistic regression based attack strategies as shown in [41] are not valid for FFXOR PUFs since the component PUFs are not linear. Even though we know that alternatives such as evolutionary strategies or multi-layer perceptrons can be used to train FF PUFs, it has been shown that training FF PUFs is much harder compared to training standard arbiter PUFs [11,27]. For example, a 128-bit standard arbiter PUF requires 2.10 seconds while a FF PUF requires 3:15 hours using evolutionary strategy [11]. In recent work by Becker, it has been shown that XOR PUFs can be attacked by using evolutionary strategies [34]. However, this method may not be applicable to attack FFXOR PUFs unless the internal arbiter output locations are known to the attacker. The attacker could potentially examine all possible FF loop positions and identify the design via trial and error. But this requires a significant increase in the attack time. For FFXOR PUFs composed of N-stage FF PUFs as components, there are $\frac{N(N-1)}{2}$ possible FF loop placements for *one* loop. Therefore, the number of machine learning runs increases by 496 times for 64 bits and 2016 times for 128 bits. In general, for FF PUFs with K loops, it would increase by a factor of $\binom{N}{K+1}$. It is worth noting that reliability based machine learning attacks have been shown to outperform other machine learning attacks on XOR

PUFs. This is because they utilize a *divide-and-conquer approach* which reduces the number of model parameters from $l(N+1)$ to $N+1$, for XOR PUFs with l components of N-stages each. But, the divide-and-conquer approach used in reliability based attacks is dependent on the fact that reliability of a response bit depends equally on each of the component PUFs [34]. But this would not be the case if FF PUFs with non-identical structures are used as components, making them more attack resistant. It has been suggested in [40] that machine learning attacks on XOR PUFs can be made even harder by using different challenges for each component PUF.

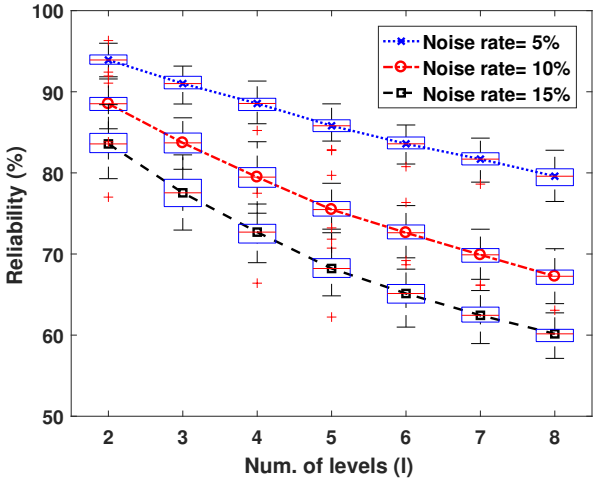
5.5 Reliability Analysis

5.5.1 FFXOR PUFs with Single-loop FF PUFs

Reliability of XOR PUFs up to 8 levels are computed by adding noise to each of the 32 stages in the component PUFs. The component PUFs are FF PUFs with a FF loop placed between stages $N_1 = 10$ and $N_2 = 20$. For each case, i.e., for each value of l, 100 PUF instances with the same circuit design are simulated. For each instance, 100 noisy 1000-bit response signatures are generated. Intra-chip variation is computed (see equation (4.1)) and the change in reliability with increase in the number of XOR levels is shown in Figure 5.4. It can be observed that it decreases as the number of levels increases and FFXOR PUFs have less reliability compared to standard XOR PUFs of the same size (see Figure 5.4). Moreover, the reliability values of FFXOR PUFs drop at a slightly higher rate with respect to the value of l. These values can be used to extrapolate the trend to estimate how reliability scales with the increase in the number of arbiter PUFs used. This demonstrates that reliability of XOR PUFs can be as low as 60% (for $l = 8$). It has been shown in [7] that reliability of an N-stage FF PUF depends on $\arctan(\sqrt{\frac{N_2-1}{N-N_2+1}})$. We considered 64-bit feed-forward PUFs (N = 64) with the intermediate output measured at the 10th stage (N1 = 10) and feeding into the 40th stage (N2 = 40) and verified that they have similar reliability to 32-stage PUFs shown in Figure 5.4 . Note that the value of $\arctan(\sqrt{\frac{N_2-1}{N-N_2+1}})$ is equal to 0.88 and 0.895 for the 32-bit and the 64-bit FF PUFs, respectively.



(a) Standard XOR PUFs



(b) Feed-forward XOR PUFs

Figure 5.4: Reliability *vs.* number of levels for 32-stage standard and FFXOR PUFs.

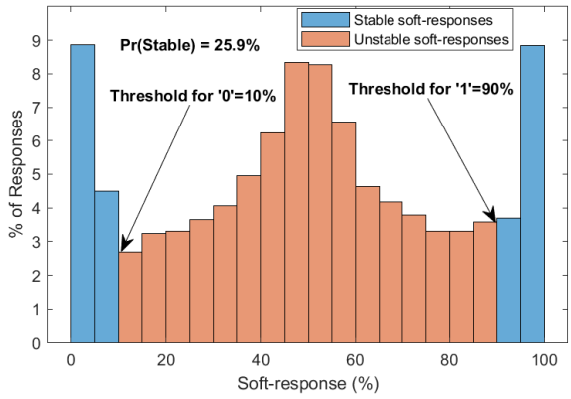


Figure 5.5: Histogram of soft-responses of a 64-stage FFXOR PUF showing stable and unstable responses. $l = 8$ and noise level = 10%.

5.5.2 Soft-response Thresholding

Soft-responses were defined in Chapter 3. For a given challenge, soft-response is defined as the probability that the response is 1 ($Pr(R = 1)$) under environmental variations. Empirically, the soft-responses were computed by applying the same challenge 100 times under the presence of noise. As a convention, we use 90% as a threshold to measure the consistency of a response. That is, if the response is 0 or 1 in 90% of the cases, it is considered a *stable response*. Otherwise, it is referred to as an *unstable response*. Therefore if $Pr(R = 1)$ is less than 10%, the response is considered a stable 0, and if it is greater than 90% it is considered a stable 1. If the value is between 10% and 90%, it is considered an unstable response.

In case of XOR PUFs, a fundamental limitation is that a large portion of the challenges could lead to unstable responses, resulting in low reliability as illustrated in Figure 5.4. For a FFXOR PUF with 8 levels, the soft-response values associated with 20,000 challenges are shown in Figure 5.5. It can be observed that 25.9% of the responses are stable. We know that soft-responses can be computed by repetitive measurement of responses. Soft-responses of the 8-level FFXOR PUF were computed by assigning each challenge 100 times. A subset of challenges were then chosen by thresholding the soft-responses using 90-10 thresholding. The challenges that generate stable responses

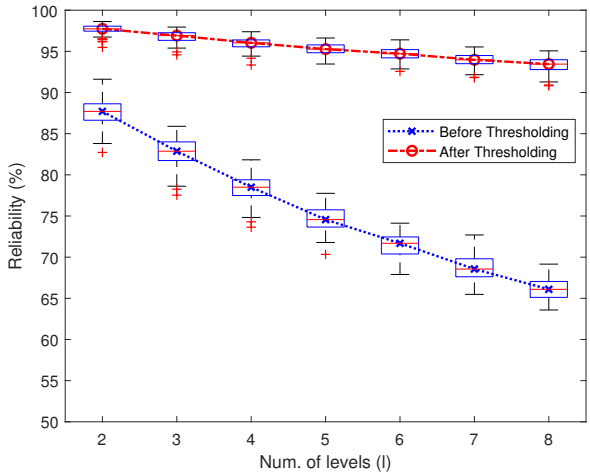


Figure 5.6: Reliability *vs.* number of levels(l) for 64-stage FFXOR PUFs before and after thresholding. Threshold = 90%. $N_1 = 10, N_2 = 40$ and Noise level = 10%.

are referred to as *stable challenges*. Instead of using a set of 1000 randomly chosen challenges, 1000 stable challenges were used to compute the reliability. Reliability values of a FFXOR PUF before and after thresholding are presented in Figure 5.6. It shows that reliability of XOR PUFs during authentication can be significantly increased by thresholding the soft-responses and identifying stable challenges. For example, the reliability of an 8-level FFXOR PUF can be increased from 66% to 93.5%. The resultant reliability achieved by thresholding depends on the value of the threshold used. If the threshold is reduced the number of stable challenges will increase but this may cause a degradation in reliability. The effect of varying the threshold value on the reliability of 8-level FFXOR PUFs is shown in Figure 5.7 for different noise levels.

5.5.3 FFXOR PUFs with Multi-loop FF PUFs

As a counter-measure to reliability of FF PUFs compared to standard arbiter PUFs, a *modified FF PUF* (MFF PUF) structure was proposed in [7]. The idea is to have the intermediate arbiter output feeding into two consecutive stages instead of one. FFXOR PUFs with MFF PUFs as the components were simulated and their reliability was computed. Furthermore, we extend the analysis to FFXOR PUFs consisting of FF

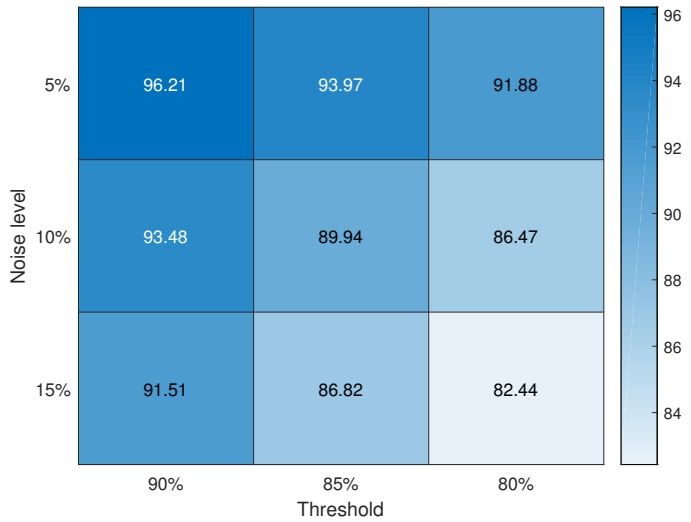


Figure 5.7: Reliability of 8-level FFXOR PUFs after thresholding for different thresholds and noise levels.

PUFs with multiple loops. Component FF PUFs where the output from an intermediate stage is used as the challenge bit to multiple MUX stages were considered. The results of reliability analysis of 8-level FFXOR PUFs up to 8 loops is presented in Tables 5.1. In particular, the proportion of stable challenges and the reliability values computed before and after thresholding are presented for various structures under different noise levels. These values were computed using 90% threshold and each value is the median of 100 PUF instances. The results show that MFF PUFs can be used to attain reliability close to standard XOR PUFs. In case of PUFs with one intermediate arbiter, having 1,3 or 5 loops results in less reliability compared to having even number of loops. The reliability and the proportion of stable challenges can be substantially low, 53% (approx.) and 1-3%, respectively, when multiple intermediate arbiters are used. Two such examples can be noticed in the Table. However, having MFF loops leads to better reliability. The results confirm that the soft-response thresholding strategy can be employed to significantly increase the reliability of response signatures to at least 89%. It is also important to note that XOR PUFs have a huge set of challenges to choose from, i.e., 2^N challenges for N-bit PUFs. For 64-bit XOR PUFs, if we assume that a mere 1% of the challenges are stable, the server still has 1.8×10^5 trillion (1% of 2^{64}) stable challenges.

5.6 Conclusion

FFXOR PUFs are better alternatives to standard XOR PUFs, in terms of their attack-resistance. Standard XOR PUF and FF XOR PUFs with 32-bit challenges, up to 8 XOR levels are evaluated using multi-layer perceptrons and the results are compared. The divide-and-conquer approach used in reliability based attacks is dependent on the fact that reliability of a response bit depends equally on each of the component PUFs [34]. This would not be the case if the component PUFs are non-identical, i.e., these components could be different FF PUF structures with different number of arbiters and loops. The effect of number of XOR levels and the number of FF loops on the reliability of FFXOR PUFs under different noise levels is presented and a soft-response thresholding strategy is demonstrated as an effective counter-measure to degraded reliability. Reliability values as low as 52% were increased to 89%.

Table 5.1: Percentage of Stable Challenges and Reliability Before and After Thresholding. Threshold = 90%.

No. of Arbs/ Loops	FF Loops Locations	Noise Level = 5%			Noise Level = 10%			Noise Level = 15%		
		% Stable	Rel.(%) Before	Rel.(%) After	% Stable	Rel.(%) Before	Rel.(%) After	% Stable	Rel.(%) Before	Rel.(%) After
0/0	-	65.5	84.63	97.36	40.69	73.48	95.26	23.88	65.67	93.45
1/1	$10 \rightarrow 40$	52.1	78.65	96.2	24.73	66.09	93.45	10.63	58.93	91.26
1/2	$10 \rightarrow 40, 41$ (MFF)	63.78	83.98	97.21	38.94	72.63	94.95	22.62	65.18	93.09
1/2	$10 \rightarrow 30, 40$	58.45	81.35	96.75	32.07	69.67	94.25	16.54	62.4	92.41
1/3	$10 \rightarrow 30, 40, 50$	52.14	78.66	96.26	24.65	65.99	93.41	10.5	59.07	91.34
1/4	$10 \rightarrow 30, 31, 40, 41$	62.74	83.27	97.18	37.70	72.21	94.82	21.45	64.58	92.86
1/4	$10 \rightarrow 20, 30, 40, 50$	54.81	79.92	96.43	27.73	67.61	93.76	12.85	60.16	91.75
1/5	$10 \rightarrow 20, 30, 40, 50, 60$	52.34	78.74	96.15	24.61	66.37	93.37	10.6	59.03	91.28
1/6	$10 \rightarrow 30, 31, 40, 41, 50, 51$	61.81	83.12	97.1	36.64	71.73	94.75	20.61	64.26	92.7
1/6	$10 \rightarrow 30, 35, 40, 45, 50, 55$	56.52	80.49	96.52	29.44	68.44	94	14.35	61.11	91.96
1/8	$10 \rightarrow 20, 21, 30, 31, 40, 41, 50, 51$	61.43	82.72	96.96	35.76	71.52	94.59	19.74	63.93	92.72
1/8	$10 \rightarrow 20, 25, 30, 35, 40, 45, 50, 55$	54.82	79.67	96.51	27.55	67.48	93.79	20.73	60.39	91.68
4/4	$10 \rightarrow 20; 15 \rightarrow 30; 25 \rightarrow 40$ $35 \rightarrow 50$	33.47	70.40	94.66	9.98	58.74	91.44	2.72	53.42	89.5
4*/8	$10 \rightarrow 20, 21; 15 \rightarrow 30, 31;$ $25 \rightarrow 40, 41; 35 \rightarrow 50, 51;$	57.73	81.14	96.75	33.09	70.04	94.30	18.18	63.02	92.56
4/4	$15 \rightarrow 55; 20 \rightarrow 50; 25 \rightarrow 45;$ $30 \rightarrow 40$	26.17	66.92	94.01	6.16	56.21	90.89	1.45	52.48	88.80
4*/8	$15 \rightarrow 55, 56; 20 \rightarrow 50, 51;$ $25 \rightarrow 45, 46; 30 \rightarrow 40, 41;$	58.01	81.26	96.85	33.36	70.18	94.52	18.36	63.32	92.58

* The intermediate arbiters are connected to modified FF loops.

Chapter 6

Conclusion and Future Directions

Physical unclonable functions exploit intrinsic physical properties of integrated circuits to enhance the security of devices by replacing or complimenting the traditional cryptographic techniques. Their practical relevance has lead to a substantial rise in the interest making PUFs a hot topic in the field of hardware security. A variety of PUFs have been proposed in the literature starting from optical PUFs [2] and arbiter PUFs [3] to the emerging nanotechnology based PUFs [42]. Even though this makes it difficult to provide a formal unifying definition of PUFs, there are certain fundamental necessary properties [5]. This thesis studies two such properties: unpredictability and reliability, in the context of arbiter PUFs. Unpredictability is a relaxed form of unclonability, i.e., if one can predict the outcome of a PUF for a random challenge, only by observing a set of CRPs, it is easy to build a mathematical clone. Reliability ensures consistency of the responses for repeated evaluation of a challenge. We prove that the inherent physical parameters of standard arbiter PUFs, FF PUFs and MFF PUFs can be accurately estimated and ANN models can be trained to predict their hard and soft-responses [27]. We show that these models can be used to choose reliable challenges for authentication. We also provide important insights on the impact of FF PUFs' design choices, i.e., the number of loops and location of the loops, on their attack-resistance and reliability [43].

XOR PUFs were considered the most secure version of arbiter PUFs since the number of challenge-response pairs required by an attacker increases exponentially with the number of component PUFs [11] [35]. But this observation was discredited later [34].

To this end, we propose and evaluate FFXOR PUFs as a viable, potentially more secure, alternative to the existing XOR PUFs [44]. We believe that state-of-the-art attack methods like CMA-ES [34] are not directly applicable to FFXOR PUFs and also that they can be made more resilient by using non-identical component PUFs such as a FF PUFs with different designs. A drawback of our analysis is that the attack-resistance evaluation is based only on ANNs and does not include side-channel attacks. Future work can be focused on studying the security of FFXOR PUF using other attack strategies. Even though FF XOR PUFs and some versions of MXPUFs [45] are shown to be secure to some extent, it is known that silicon PUFs are still vulnerable to physical attacks [40] such as photon emission analysis. Hence, the emerging area of nanotechnology based PUFs can be an exciting new direction for PUFs [42].

An entropy based metric was presented in Chapter 3 but it is specific to the PUF configurations considered. There has been some effort to formalize security evaluation of PUFs theoretically in the past [26] [46] [47]. However, the metrics are either insufficient or specific to a certain type of PUF and hence cannot be generalized. A universal theoretical metric or model to estimate unpredictability can be very useful to compare the security of different PUFs without undergoing an attack process.

Several solutions have been proposed to increase the reliability of PUFs using error-correcting codes [48]. However they either incur large area and computational costs [49] or leak security information [50] [51]. In this work, we demonstrate that soft-response thresholding is a simple yet effective counter-measure to the degraded reliability of FF and FFXOR PUFs. The limitation of this approach is that in some cases, a large number of measurements might be required for a single authentication.

This thesis provides interesting insights on security and reliability of arbiter PUFs. However, the task of realizing secure and reliable PUFs with low energy and area overhead is still an active research challenge.

References

[1] 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 International Symposium on Low Power Electronics and Design (ISLPED)*, Aug 2016.

[2] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical one-way functions. *Science*, 297(5589):2026–2030, 2002, http://science.sciencemag.org/content/297/5589/2026.full.pdf.

[3] 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, 2002.

[4] C. Herder, Meng-Day Yu, F. Koushanfar, and S. Devadas. Physical unclonable functions and applications: A tutorial. In *Proceedings of the IEEE*, pages 1126–1141, Aug 2014.

[5] Roel Maes and Ingrid Verbauwhede. *Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions*, pages 3–37. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[6] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. A systematic method to evaluate and compare the performance of physical unclonable functions. *IACR Cryptology ePrint Archive*, 2011:657, 2011.

[7] 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, May 2014.

[8] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. Testing techniques for hardware security. In *Proceedings of IEEE International Test Conference (ITC)*, pages 1–10, Oct 2008.

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

[10] G. Hospodar, R. Maes, and I. Verbauwhede. Machine learning attacks on 65nm arbiter PUFs: Accurate modeling poses strict bounds on usability. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 37–42, Dec 2012.

[11] U. Rührmair, J. Sölter, F. Sehnke, Xiaolin Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas. PUF modeling attacks on simulated and silicon data. *IEEE Transactions on Information Forensics and Security*, 8(11):1876–1891, Nov 2013.

[12] G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *44th ACM/IEEE Design Automation Conference*, June 2007.

[13] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Lightweight secure PUFs. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 670–673, Nov 2008.

[14] D. P. Sahoo, S. Saha, D. Mukhopadhyay, R. S. Chakraborty, and H. Kapoor. Composite PUF: A new design paradigm for physically unclonable functions on FPGA. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 50–55, May 2014.

[15] Blaise Gassend, Daihyun Lim, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Identification and authentication of integrated circuits: Research articles. *Concurrency and Computation: Practice & Experience - Computer Security*, 16(11):1077–1098, sep 2004.

[16] D. Lim, J.W. Lee, B. Gassend, G.E. Suh, M. van Dijk, and S. Devadas. Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(10):1200–1205, Oct 2005.

[17] Simon Haykin. *Adaptive Filter Theory (3rd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[18] S. V. 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 *Proceedings of Design, Automation and Test in Europe (DATE)*, pages 543–546, Mar 2016.

[19] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2nd edition, 2010.

[20] Georg T. Becker. On the pitfalls of using arbiter-PUFs as building blocks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(8):1295–1307, Aug 2015.

[21] Jeroen Delvaux and Ingrid Verbauwhede. Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 137–142, June 2013.

[22] X. Xu, W. Burleson, and D. E. Holcomb. Using statistical models to improve the reliability of delay-based PUFs. In *IEEE Computer Society Annual Symposium on VLSI*, pages 547–552, July 2016.

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

[24] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[25] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591, 1993.

[26] 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 *Proceedings of 14th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 283–301, Sep 2012.

[27] S. V. S. Avvaru, C. Zhou, C. H. Kim, and K. K. Parhi. Predicting hard and soft-responses and identifying stable challenges of MUX PUFs using ANNs. In *60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 934–937, Aug 2017.

[28] Raghavan Kumar and Wayne Burleson. Side-channel assisted modeling attacks on feed-forward arbiter PUFs using silicon data. In Stefan Mangard and Patrick Schaumont, editors, *International workshop on Radio Frequency Identification*, pages 53–67. Springer International Publishing, 2015.

[29] X. Xu and W. Burleson. Hybrid side-channel/machine-learning attacks on PUFs: A new threat? In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, March 2014.

[30] A. Ayling, S. V. S. Avvaru, and K. K. Parhi. Not all feed-forward MUX PUFs generate unique signatures. In *Proc. IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2019.

[31] Y. Lao and K. K. Parhi. Reconfigurable architectures for silicon physical unclonable functions. In *IEEE International Conference on Elctro/Information Technology*, pages 1–7, May 2011.

[32] A. Koyily, S. V. S. Avvaru, C. Zhou, C. H. Kim, and K. K. Parhi. Effect of aging on linear and nonlinear mux pufs by statistical modeling. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 76–83, Jan 2018.

[33] Phuong Ha Nguyen, Durga Prasad Sahoo, Chenglu Jin, Kaleel Mahmood, Ulrich Rührmair, and Marten van Dijk. The interpose PUF: Secure PUF design against state-of-the-art machine learning attacks. *IACR Cryptology ePrint Archive*, 2018:350, 2018.

[34] George T. Becker. The gap between promise and reality: On the insecurity of XOR arbiter PUFs. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 235–255, September 2015.

[35] C. Zhou, K. K. Parhi, and C. H. Kim. Secure and reliable XOR arbiter PUF design: An experimental study based on 1 trillion challenge response pair measurements. In *54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2017.

[36] George T. Becker. On the pitfalls of using arbiter-PUFs as building blocks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(8):1295–1307, Aug 2015.

[37] Fatemeh Ganji, Shahin Tajik, and Jean-Pierre Seifert. Why attackers win: On the learnability of XOR arbiter PUFs. In Mauro Conti, Matthias Schunter, and Ioannis Askoxylakis, editors, *Trust and Trustworthy Computing*, pages 22–39, Cham, 2015. Springer International Publishing.

[38] Ahmed N Mahmoud, Ulrich Rührmair, Mehrdad Majzoobi, and Farinaz Koushanfar. Combined modeling and side channel attacks on strong pufs. *IACR Cryptology ePrint Archive*, 2013:632, 2013.

[39] Ulrich Rührmair, Xiaolin Xu, Jan Sölter, Ahmed Mahmoud, Mehrdad Majzoobi, Farinaz Koushanfar, and Wayne Burleson. Efficient power and timing side channels for physical unclonable functions. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES*, pages 476–492, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

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

[41] 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*, CCS '10, pages 237–249, New York, NY, USA, 2010. ACM.

[42] Y. Gao, D. C. Ranasinghe, S. F. Al-Sarawi, O. Kavehei, and D. Abbott. Emerging physical unclonable functions with nanotechnology. *IEEE Access*, 4:61–80, 2016.

[43] S. V. S. Avvaru and K. K. Parhi. Effect of loop positions on reliability and attack resistance of feed-forward pufs. In *Proc. IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2019.

[44] S. V. S. Avvaru and K. K. Parhi. Feed-forward XOR PUFs: Reliability and attack-resistance analysis. In *Proc. ACM Great Lakes Symposium on VLSI*, May 2019.

[45] Phuong Ha Nguyen, Durga Prasad Sahoo, Chenglu Jin, Kaleel Mahmood, and Marten van Dijk. MXPUF: Secure puf design against state-of-the-art modeling attacks. *IACR Cryptology ePrint Archive*, 2017:572, 2017.

[46] F. Armknecht, R. Maes, A. Sadeghi, F. Standaert, and C. Wachsmann. A formalization of the security features of physical functions. In *IEEE Symposium on Security and Privacy*, pages 397–412, May 2011.

[47] Frederik Armknecht, Daisuke Moriyama, Ahmad-Reza Sadeghi, and Moti Yung. Towards a unified security model for physically unclonable functions. In Kazue Sako, editor, *Topics in Cryptology - CT-RSA 2016*, pages 271–287, Cham, 2016. Springer International Publishing.

[48] M. Yu and S. Devadas. Secure and robust error correction for physical unclonable functions. *IEEE Design Test of Computers*, 27(1):48–65, Jan 2010.

[49] Mudit Bhargava and Ken Mai. An efficient reliable puf-based cryptographic key generator in 65nm CMOS. In *Proceedings of the Conference on Design, Automation & Test in Europe*, DATE '14, pages 70:1–70:6, 3001 Leuven, Belgium, Belgium, 2014. European Design and Automation Association.

[50] Jeroen Delvaux and Ingrid Verbauwhede. Attacking PUF-based pattern matching key generators via helper data manipulation. In Josh Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, pages 106–131, Cham, 2014. Springer International Publishing.

[51] Srinivas Devadas and Meng-Day (Mandel Yu. Recombination of physical unclonable functions. 03 2010.