8-2018

Improving Security and Reliability of Physical Unclonable Functions Using Machine Learning

Yuejiang Wen
Clemson University, yuejiaw@g.clemson.edu

Improving Security and Reliability of Physical Unclonable Functions using Machine Learning

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Electrical Engineering

---

by
Yuejiang Wen
May 2018

Accepted by:
Dr. Yingjie Lao, Committee Chair
Dr. Rajendra Singh
Dr. William R. Harrell

Abstract

Physical Unclonable Functions (PUFs) are promising security primitives for device authentication and key generation. Due to the noise influence, reliability is an important performance metric of PUF-based authentication. In the literature, lots of efforts have been devoted to enhancing PUF reliability by using error correction methods such as error-correcting codes and fuzzy extractor. However, one property that most of these prior works overlooked is the non-uniform distribution of PUF response across different bits.

This wok proposes a two-step methodology to improve the reliability of PUF under noisy conditions. The first step involves acquiring the parameters of PUF models by using machine learning algorithms. The second step then utilizes these obtained parameters to improve the reliability of PUFs by selectively choosing challenge-response pairs (CRPs) for authentication. Two distinct algorithms for improving the reliability of multiplexer (MUX) PUF, i.e., *total delay difference thresholding* and *sensitive bits grouping*, are presented. It is important to note that the methodology can be easily applied to other types of PUFs as well. Our experimental results show that the reliability of PUF-based authentication can be significantly improved by the proposed approaches. For example, in one experimental setting, the reliability of an MUX PUF is improved from 89.75% to 94.07% using *total delay difference thresholding*, while 89.30% of generated challenges are stored. As opposed to *total delay difference thresholding*, *sensitive bits grouping* possesses higher efficiency, as it can produce reliable CRPs directly. Our experimental results show that the reliability can be improved to 96.91% under the same setting, when we group 12 bits in the challenge vector of a 128-stage MUX PUF.

Besides, because the actual noise varies greatly in different conditions, it is hard to predict the error of of each individual PUF response bit. This wok proposes a novel methodology to improve the efficiency of PUF response error correction based on error-rates. The proposed method first

obtains the PUF model by using machine learning techniques, which is then used to predict the error-rates. Intuitively, we are inclined to tolerate errors in PUF response bits with relatively higher error-rates. Thus, we propose to treat different PUF response bits with different degrees of error tolerance, according to their estimated error-rates. Specifically, by assigning optimized weights, i.e., 0, 1, 2, 3, and infinity to PUF response bits, while a small portion of high error rates responses are truncated; the other responses are duplicated to a limited number of bits according to error-rates before error correction and a portion of low error-rates responses bypass the error correction as direct keys. The hardware cost for error correction can also be reduced by employing these methods. *Response weighting* is capable of reducing the false negative and false positive simultaneously. The entropy can also be controlled. Our experimental results show that the *response weighting* algorithm can reduce not only the false negative from 20.60% to 1.71%, but also the false positive rate from 1.26×10^{-21} to 5.38×10^{-22} for a PUF-based authentication with 127-bit response and 13-bit error correction. Besides, three case studies about the applications of the proposed algorithm are also discussed.

Along with the rapid development of hardware security techniques, the revolutionary growth of countermeasures or attacking methods developed by intelligent and adaptive adversaries have significantly complicated the ability to create secure hardware systems. Thus, there is a critical need to (re)evaluate existing or new hardware security techniques against these state-of-the-art attacking methods. With this in mind, this wok presents a novel framework for incorporating active learning techniques into hardware security field. We demonstrate that active learning can significantly improve the learning efficiency of PUF modeling attack, which samples the least confident and the most informative challenge-response pair (CRP) for training in each iteration. For example, our experimental results show that in order to obtain a prediction error below 4%, 2790 CRPs are required in passive learning, while only 811 CRPs are required in active learning. The sampling strategies and detailed applications of PUF modeling attack under various environmental conditions are also discussed. When the environment is very noisy, active learning may sample a large number of mislabeled CRPs and hence result in high prediction error. We present two methods to mitigate the contradiction between informative and noisy CRPs.

At last, it is critical to design secure PUF, which can mitigate the countermeasures or modeling attacking from intelligent and adaptive adversaries. Previously, researchers devoted to hiding PUF information by pre- or post processing of PUF challenge/response. However, these

methods are still subject to side-channel analysis based hybrid attacks. Methods for increasing the non-linearity of PUF structure, such as feedforward PUF, cascade PUF and subthreshold current PUF, have also been proposed. However, these methods significantly degrade the reliability. Based on the previous work, this work proposes a novel concept, noisy PUF, which achieves modeling attack resistance while maintaining a high degree of reliability for selected CRPs. A possible design of noisy PUF along with the corresponding experimental results is also presented.

Acknowledgments

First and foremost, I would like to thank my advisor Prof. Yingjie Lao, for his cardinal and continuing encouragement, tremendous guidance, throughout my M.S. study at the Clemson University. In my study and research, he has served as an incredible inspiration and talented mentor. In my life and career development, he sincerely and continuously gives me a lot of guidance and advice. As a student of him, I learned not only the expertise in my area of study and research , but also some important skills in research and thinking. I achieved my study goal for pursuing a master at Clemson University.

My sincere thank also goes to my research group. I am grateful to Xiaojia Wang, Joseph Clements, Bingyin Zhao, Ling Qiu, Ankur Sharma, Weihang Tan, Sarah Fulmer at Clemson University, for their collaborations and valuable feedback to my research/study. I also would like to thank Xiurui Zhao, Xingchen Shao, Yimin Lei, Zuo Zhou, Tingzhao Huang, Shengjie Xu for their support during my life and study at Clemson University.

I also would like to thank Prof. Rajendra Singh, Prof. William Harrell at Clemson University, for their support as members of my M.S. committee and kind help throughout my graduate study. I learned from their immense knowledge and experiences. I am fortunate to be a student of them.

Last but not the least, I am forever grateful to my mother, father, sister, grandpa, grandma for their deep love, for thoughtful care and continuous encouragement. Without the encourage and support from them, I may not enter this program and get the degree.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, semiconductor devices are becoming an essential part of everyday's life, which range across smart phone, intelligent vehicle, IoTs (Internet-of-Things), and healthcare system. Personalized data stored on these devices should be protected from external theft, attacks or falsification. As a result, security has emerged as a critical design challenge for integrated circuits (ICs). Besides, due to the globalization of modern digital design and fabrication, more and more counterfeit ICs are damaging the development of the semiconductor industry and threatening the security of end-user systems. Therefore, robust hardware security mechanisms for IC design without significant overheads in power and silicon area are highly desired.

1.1 Physical Unclonable Functions

Physical unclonable functions (PUFs) are promising security primitives that generate high-entropy and tamper-resistant chip-unique signatures by exploiting the uncontrollable randomness in the inherent process variations. Analogous to a human fingerprint, PUFs are powerful tools for authentication and cryptographic applications. PUF provides an attractive alternative to conventional security methods in standard digital system by extracting signatures from the complex physical phenomenon rather than storing them in a non-volatile memory. These signatures can then be used for many hardware security applications including authentication, IC metering and obfuscation [1–3]. Various types of PUFs such as optical PUF, coating PUF, silicon PUF, butterfly PUF, SRAM-PUF, and memristor PUF [4–8] have been invented. PUFs are powerful tools for chip

authentication and identification, as the PUF response is unique and unpredictable for each IC. In the literature, authentication using PUF response bits as secret keys has been explored in many previous works [9–12]. PUF or PUF-based application needs to be lightweight to keep it attractive in resource-constrained systems. Various types of PUFs have been invented in the literature [4–7]. Among these PUFs, the MUX PUF [7] is of great interest, as it is simple and compact ($O(2^N)$ CRPs for an N-stage PUF) and is ideal for area-constrained platforms. A MUX PUF, which consists of N stages of MUX pairs and an arbiter. The response is generated according to the input challenge vector and the delay difference of each MUX pair. The detail MUX PUF model is introduced in 2.1

1.2 Reliability of PUF

Reliability is an important metric of PUF performance, which is characterized by comparing the Hamming distances of digital signatures generated by a same challenge, however, under different environmental conditions. Ideally, a PUF should always be able to regenerate the same response for a given challenge; otherwise, even the authentic device might fail to pass the authentication. Unfortunately, the MUX PUF is not 100% stable, as temperature variations, voltage variations or even aging effect could lead to inconsistent responses [13].

A number of methods have been employed to mitigate the errors in PUF response, which include error correcting codes (ECC), fuzzy extractor, index -based sydrome (IBS) [14], and ring weight algorithm (RWA) [15]. Most of these methods assume identical error-rate across different bits in a PUF response and introduce redundancies to correct errors within a certain limit.

In the context of PUF reliability enhancement methods, one property that has been addressed little attention entails the non-uniform error rate distribution of PUF responses across different challenges. The probability that a response will differ is dependent not only on the selected challenge but also on the underlying PUF model. For example, the response of a MUX PUF is determined by the racing result between the top and the bottom paths that are generated by the challenge. Intuitively, the output with a smaller total delay difference is more likely to flip. In this wok, we take advantage of this property in developing algorithms for improving the reliability of PUF-based authentication. We present two effective and cost-efficient algorithms by using machine learning techniques.

1.3 Efficient Error Correction of PUF Responses

Indeed, a PUF response dependents on multiple factors, including physical property of the PUF, the challenge, the manufacturing process variation, and the environmental noise. The manufacturing process variation is random during the process but is deterministic after fabrication, which is also the reason that PUF can be used for establishing chip ID. However, the actual magnitudes of the manufacturing process variation sampled by different challenges can vary. Thus, the error-rates of the response bits generated by different challenges for a same PUF might also be different even under a same environmental condition.

This wok considers the non-uniformity in PUF response in developing an algorithm for improving the error-correcting methods of PUF-based authentication. Different from previous works, this work treats different response bits with different degrees of error tolerance. For example, if a bit has a relatively high error-rate, we will correct the error in this bit as the error is more likely to come from the environmental noise instead of malicious attacks. However, if a bit in a PUF response with a very low error-rate flips, the authentication should be failed since it has a high possibility that the inquired PUF is not an authentic PUF. Taking these properties into considerations, we present an algorithm to improve the PUF response error correction by using the predicted error-rate of each PUF response bit, i.e., *response weighting*. *Response weighting* can simultaneously reduce the false positive rate and the false negative rate of the PUF-based authentication. The main concept is to minimize the error-rate variance of the weighted PUF response across all the bits. The optimized weights can be solved by Generalized Reduced Gradient (GRG) algorithm. To the best of our knowledge, this wok is the first work to utilize the non-uniform error-rate of response bits of different PUFs to improve the efficiency of PUF response error correction.

1.4 Challenge from Attacker

Security and uniqueness are essential to a PUF's performance against attacks. In the past decades, numerous hardware security techniques have been proposed. A strong security management for digital systems should encompass the entire lifecycle and all aspects given that adversaries can exploit the weakest link to compromise the systems. The main challenge of protecting hardware devices is the adversary may find ways to physically tamper with devices without leaving a trace, misleading the user to believe that the hardware is authentic and trustworthy. In addition, the

embedded, distributed, unsupervised, and physically exposed nature of modern electronic systems, such as Internet of Things (IoT) and cyber-physical systems (CPS), has made non-invasive or semi-invasive attacks on hardware devices as critical threats. If an entry point can be found from the hardware perspective, the adversary can easily cascade an attack down the whole system. Motivated from these facts, numerous hardware security techniques have been proposed in past years. However, the incentive for defeating them also increases.

Chapter 2

Enhancing PUF Reliability by Machine Learning

2.1 MUX PUF Model

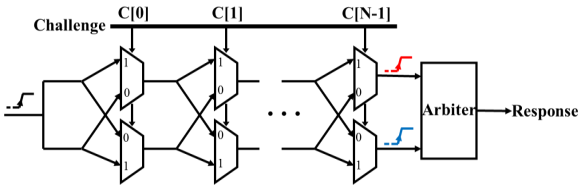


Figure 2.1. A MUX PUF.

Fig. 2.1 illustrates the design of a MUX PUF, which consists of N stages of MUX pairs and an arbiter. The response is generated according to the input challenge vector and the delay difference of each MUX pair. The model has been extensively studied in [16]. Each single MUX can be expressed as an independent identically distributed (i.i.d.) random variable D_i, modeled by a Gaussian random variable $N(\mu, \sigma^2)$, where μ represents the mean and σ represents the standard deviation of the delay of each MUX. As a result, the total delay of the N stages is modeled by $N(N\mu, N\sigma_s^2)$. According to [16], the delay difference after the last stage can be expressed as

$$r_N = \sum_{i=1}^{N} (-1)^{C'_i} \Delta_i + \sum_{i=1}^{N} n_i, \tag{2.1}$$

where $C'_i = \oplus_{j=i+1}^{N} C_j$, $C'_N = 0$, and Δ_i, n_i represents the delay difference and environmental noise of each stage, respectively. Finally, the output bit is generated by $R = sign(r_N)$. If $r_N \geq 0$, R is equal to 1; otherwise, if $r_N < 0$, R will be 0. Therefore, it can be concluded that smaller total delay difference, i.e., $\left|\sum_{i=1}^{N} (-1)^{C'_i} \Delta_i\right|$, leads to higher error probability in the response.

2.2 PUF Reliability Enhancement

In the literature, conventional error-correcting codes such as BCH codes [13], index-based syndrome coding [14], soft-decision decoding [17], and pattern matching [18] have been used to correct response errors in a PUF-based authentication scheme. However, these methods incur large area/power overhead if implemented on hardware, which may inhibit the benefit of achieving lightweight chip signature generation by using compact PUF circuits. Besides, a number of works have been devoted to improving the reliability under certain conditions. For example, repetition test was used to mitigate temperature and voltage variations in PUF responses [19]. A solution was presented in [1] to improve PUF reliability by recognizing stable and unstable arbiters for a given challenge. Recently, a probability based response generation scheme has been proposed [20], which can achieve higher reliability of PUF-based authentication as well.

2.2.1 Estimating PUF Parameter

In the literature, several works that are related to obtaining parameters of PUF circuit model have been proposed for PUF based applications. Modeling attack and parameter-based authentication are two examples.

2.2.1.1 Modeling Attack on PUF

Since manufacturing process variation becomes deterministic after fabrication, it is possible to learn the PUF inner structure to compromise its security. Modeling attack and statistical analysis show that PUFs can be mathematically modeled and replaced by a software program to compute response for future challenges. We detail the modeling results of MUX PUFs in [16]. The total delay difference before the arbiter of an N-stage MUX PUF can be expressed as $r_N = \sum_{i=1}^{N} (-1)^{C'_i} \Delta_i + \sum_{i=1}^{N} n_i$, where $C'_i = \oplus_{j=i+1}^{N} C_j$, $C'_N = 0$, and Δ_i and n_i represent the delay difference and environmental noise for each stage, respectively. The final output is generated by

$R = sign(r_N)$. If $r_N \geq 0$, R is equal to 1; otherwise, if $r_N < 0$, R will be 0, when skew effect is not considered, as shown in Fig. 2.1.

Modeling attack is a major threat to PUF security in which an adversary attempts to derive a numerical model to predict responses to arbitrary challenges with a high probability after collecting a subset of PUF CRPs. Machine learning methods have been used to perform modeling attacks to learn the PUF circuit parameters [21,22]. Algorithms such as support vector machine (SVM), logistic regression (LR), and evolution strategies (ES) all can successfully estimate the delay differences of each stage (i.e., Δ_i) of a MUX PUF after collecting a number of CRPs [21, 23]. Consequently, it is feasible to use the model and parameters to predict the corresponding response with very high accuracy for any future challenge.

2.2.1.2 Parameter-based Authentication

Parameter-based authentication is a new concept in server-based remote PUF authentication [10–12]. In a typical remote authentication protocol, a central server interrogates the remote system by providing a challenge to the PUF and comparing the received response with CRPs stored in its database. Based on the protocol, a new concept of parameter-based authentication has been discussed in [10–12]. As opposed to prior works, the server only stores the parameters of a PUF instead of a large number of CRPs during the enrollment phase of the parameter-based authentication scheme. Similar to modeling attacks, the parameters can be obtained by machine learning algorithms, given the fact that the designer knows the PUF model. In the authentication phase, the received PUF response is compared to the model outputs that are emulated according to the challenge and pre-acquired parameters instead of stored CRPs. This method can significantly reduce the memory requirement for server-based PUF authentication.

Similarly, it is also possible to estimate the error-rate of each PUF response bits through PUF circuit model by using machine learning methods.

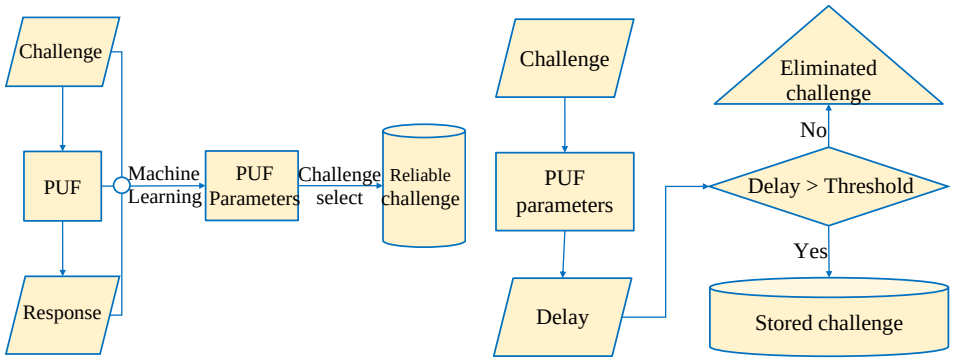
2.3 Proposed Methodology

We propose a novel two-step mechanism for selecting challenge that produces insensitive response to enhance the reliability of PUF-based authentication: PUF parameters learning and challenge selection. For a certain challenge, the response is determined by the physical PUF function.

However, the influence of environmental noise may cause inconsistencies in the expected CRPs. By using the PUF model and parameters trained by machine learning, we will be able to predict the error probability of a PUF response by incorporating noise into the PUF model. Consequently, if we can eliminate the challenges that generate responses with relatively high error probabilities, the reliability can obviously be improved. The general methodology is shown in Fig. 2.2(a). In this work, we use MUX PUF to illustrate PUF reliability enhancement algorithms, which can also be easily applied to other types of PUFs.

Fig. 2.1 illustrates the design of a MUX PUF. The delay difference after the last stage can be expressed as Equation 2.1. And the output bit is generated by $R = sign(r_N)$. If $r_N \geq 0$, R is equal to 1; otherwise, if $r_N < 0$, R will be 0. Therefore, it can be concluded that smaller total delay difference, i.e., $\left|\sum_{i=1}^{N}(-1)^{C'_i}\Delta_i\right|$, leads to higher error probability in the response.

2.3.1 Algorithm 1: Total Delay Different Thresholding



(a) Methodology

(b) Algorithm 1: total delay difference thresholding

Figure 2.2. The flows of proposed methodology and algorithms.

This method improves the MUX PUF reliability by setting a total delay difference *threshold* for selecting PUF challenges. We first estimate the parameters of a MUX PUF by using machine learning algorithms. We then use the obtained parameters to calculate the expected total delay difference for a given challenge. We will only use the challenges which generate total delay difference of a MUX PUF that is greater than the preselected *threshold* (i.e., $\left|\sum_{i=1}^{N}(-1)^{C'_i}\Delta_i\right| > threshold$) for authentication, as shown in Fig. 2.2(b). A similar thresholding method is independently developed in [24, 25]. Algorithm 1 summarizes the detailed steps.

Algorithm 1 Total Delay Difference Thresholding

1. Obtain parameters of the MUX PUF, i.e., Δ_i.
2. For a given challenge, calculate total delay difference $(\sum_{i=1}^{N}(-1)^{C'_i}\Delta_i)$ based on the parameters obtained from Step 1, and compare the total delay difference with pre-defined threshold.
3. If $\left|\sum_{i=1}^{N}(-1)^{C'_i}\Delta_i\right| > threshold$, store the challenge for authentication; otherwise, eliminate the challenge.

2.3.2 Algorithm 2: sensitive bits grouping

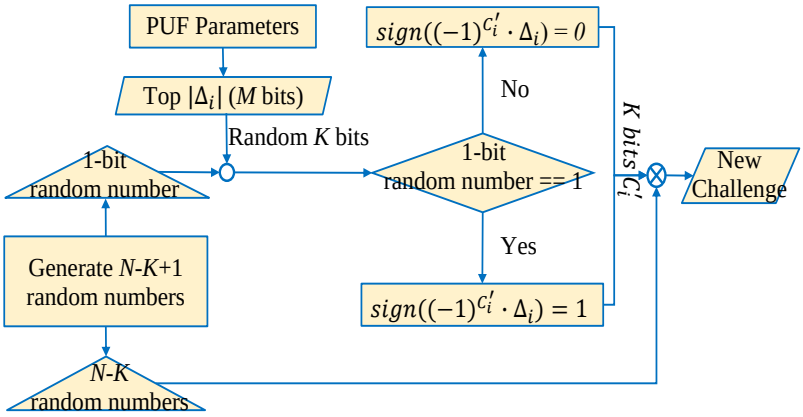


Figure 2.3. The proposed algorithm 2: sensitive bits grouping.

An alternative approach for enhancing PUF reliability in the parameter-based authentication scenario is to group sensitive challenge bits, as shown in Fig. 2.3. For example, we can randomly select K bits from top M bits with the largest absolute values of Δ_i, i.e., $|\Delta_i|$, where the length of the challenge vector is N. We can then group these K challenge bits into 1-bit entropy. According to the value of this 1-bit random number, we adjust the values of C' so that $(-1)^{C'_i}\Delta_i$ of these K challenge bits are all positive or all negative. For example, we may assume the 1-bit random number is 0 without loss of generality. In this case, C' will be 1 for stages with positive Δ_i and 0 for stages with negative Δ_i; while if the 1-bit random number is 1, C' will be 0 for stages with positive Δ_i and 1 for stages with negative Δ_i, respectively. This method can be classified into 3 cases: 1) $K = M$ (i.e., select all top K bits); 2) $K < M < N$; 3) $K < M = N$ (i.e., randomly select K bits without sorting). The full steps are described in Algorithm 2.

Total delay difference thresholding is a straightforward approach that post-processes the generated challenges to eliminate the challenges that do not meet the threshold requirement. However, *sensitive bits grouping* generates challenges that can produce reliable response directly, which can be considered as a pre-processing method. Therefore, *sensitive bits grouping* is more efficient in

generating challenges for PUF-based authentication, since not all the challenges generated by *total delay difference thresholding* will eventually be used. Note that both of these two methods would not increase the false positive rate, i.e., the probability of false authentication of wrong PUFs.

Algorithm 2 Sensitive Bits Grouping

1. Obtain parameters of the MUX PUF, Δ_i.
2. Sort the absolute values of all the stage delay differences of the MUX PUF, $|\Delta_i|$.
3. Randomly group K bits from the top M bits with largest $|\Delta_i|$ ($K < M \leq N$).
4. Generate $N - K + 1$ random numbers.
5. Use $N - K$ bit random numbers directly as the challenges of the un-grouped challenges and the rest 1-bit random number as the entropy for the grouped K bits from Step 3 that if 1-bit random number is 1, $sign((-1)^{C'_i}\Delta_i)$ will be 1 (i.e., C' will be 0 for stages with positive Δ_i and 1 for stages with negative Δ_i); while 1-bit random number is 0, $sign((-1)^{C'_i}\Delta_i)$ will be 0 for all the grouped stages.

2.4 Results of PUF Reliability Enhancement

2.4.1 Experimental Setup

In our experiment, we simulate parameters and manufacturing process variations in the MUX PUF model according to prior theoretical and experimental results in [1, 16, 26]. We vary the environmental variations to examine the performances of the proposed algorithms.

2.4.2 Results

2.4.2.1 Parameters Modeling

Fig. 2.4 presents the accuracy of simulated model of a 64-stage MUX PUF by using linear SVM. It can be seen that the accuracy increases as the increase of the number of samples used for training. Furthermore, the magnitude of noise will also affect the accuracy of machine learning. For example, more samples are needed to achieve a 95% accuracy under a larger environmental variation.

2.4.2.2 Total delay difference thresholding

We compare the PUF reliability of before and after using *total delay difference thresholding* systematically. Our experimental results show that the method can effectively improve the reliability. Meanwhile, we can maintain a considerably high selection ratio of challenge subsets. For example,

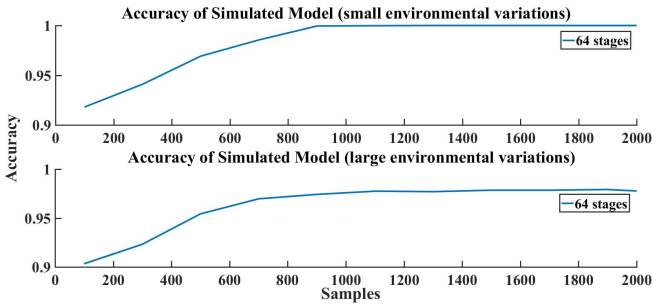


Figure 2.4. Accuracy of simulated model of a 64-stage MUX PUF.

the reliability is improved from 89.75% to 94.07%, while 89.30% of generated challenges are stored, as shown in Fig. 2.5(a) and Fig. 2.5(b). We can further improve the reliability to 99.60% by setting a larger threshold. However, in this case, the percentage of stored challenges is reduced to 76.65%. Table 2.1 presents the results under a relatively large environmental variation. It can be seen that the method can still achieve significant improvement of the PUF-based authentication. However, due to the large noise, challenge selection ratios will be low.

Table 2.1. Reliability of PUF with different selection thresholds (under a large environmental variation)

Stages	Original		Threshold $0.13 * \sqrt{N}\sigma_s$		Threshold $\sqrt{N}\sigma_s$	
	Reliability	Selection Ratio	Reliability	Selection Ratio	Reliability	Selection Ratio
64	72.41%	100%	80.29%	81.47%	95.23%	27.22%
128	73.03%	100%	80.22%	79.80%	95.30%	23.37%
192	72.38%	100%	80.33%	78.09%	94.55%	27.28%
256	71.88%	100%	80.25%	80.17%	95.28%	29.37%

2.4.2.3 Sensitive bits grouping

Fig. 2.5(c) presents the total delay difference distribution after using Algorithm 2 with $K = 12$, $M = 16$, and $N = 128$. Results show that *sensitive bits grouping* can also significantly improve the PUF reliability from 89.75% to 96.91%. In addition, *sensitive bits grouping* provides more configurability than *total delay difference threshold*, as we can adjust the parameters (i.e., K and M) appropriately based on the reliability requirement of a specific application. Table 2.2 presents the performances of *sensitive bits grouping* with different values of K and M, where $N = 64$.

For case 1 that we always select the top K challenge bits with the largest $|\Delta_i|$ (i.e., $K = M < N$), it can be seen from Table 2.2 that the reliability increases with the increase of K. For example, the reliability of the 64-bit PUF can be improved to over 99% from 72.25% when $K = M = 32$. Note

that in this case, the number of possible challenges will be reduced from 2^N to 2^{N-K+1}. For case 3 that we randomly select K challenge bits without sorting (i.e., $K < M = N$), the reliability also increases with the increase of K. Furthermore, for a same K, case 1 can achieve higher reliability, while case 3 will have more possible challenges. The reliability enhancement of case 2 is in between the performances of case 1 and case 3. Therefore, we can conclude that reliability increases as K increases for a certain M and decreases as M increases for a certain K, where $K \leq M$. Note that the number of possible challenges has the opposite trends.

Table 2.2. Reliability of PUF before/after Algorithm 1, 2 with $N = 64$.

Algorithms		Reliability
Original		72.25%
Algorithm 2, case 1	$K = M = 8$	77.10%
	$K = M = 16$	80.52%
	$K = M = 32$	99.01%
Algorithm 2, case 2	$K = 8$ $M = 12$	74.61 %
	$K = 12$ $M = 16$	78.86 %
	$K = 24$ $M = 32$	96.08%
	$K = 32$ $M = 48$	100%
Algorithm 2, case 3	$K = 8$ $M = 64$	73.50%
	$K = 16$ $M = 64$	75.57%
	$K = 32$ $M = 64$	100%

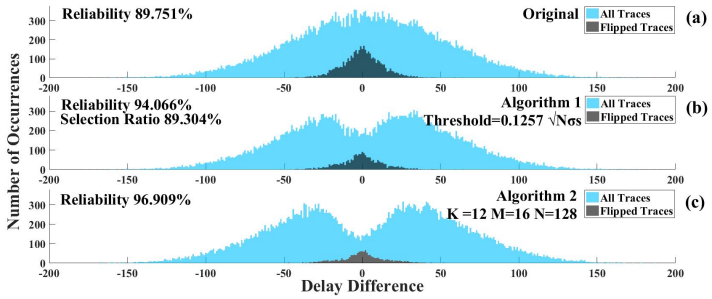


Figure 2.5. Total delay difference distributions of (a) before using the proposed algorithms, (b) after using Algorithm 1, and (c) after using Algorithm 2.

Chapter 3

Enhancement of PUF Response Error Correction Efficiency

We propose a novel error-rate based methodology to enhance the efficiency of PUF-based authentication, which is consist of two main steps, i.e., PUF response error-rate estimation and PUF response processing. The method for PUF response processing is presented in this section, which can be applied on either the enrollment phase or the authentication phase. PUF response error-rate estimation should be performed on the enrollment by using machine learning methods. Based on the estimated error-rate, the information required for PUF response processing can be sent to the PUF along with the challenge or can be used on the server after collecting the raw PUF response from the chip. The proposed methodology is extremely suitable for parameter-based authentication, as the parameters for the PUF model have already been computed which can be directly used to estimate the response error-rate given a certain PUF challenge. The syndrome or helper data of the error-correcting methods should be computed based on the sequence after PUF response processing. The notations used in this work are listed in Table 3.1.

3.0.1 PUF Response Error Correction

One major issue for these PUF-based applications is the reliability, since environmental variations may fluctuate the PUF response. Conventional error-correcting codes such as BCH codes [13] and low-density parity check (LDPC) codes, and fuzzy extractors have been employed to improve

Table 3.1. Notations Used in the Thesis

Notation	Explanation
N	bit-length of the PUF response
T	maximum number of correctable bits in the error-correcting method
N_t	number of bits truncated
N_d	number of bits duplicated by
N_b	number of bits that bypass the error-correcting method
w	weight of each PUF response bit
N_{wi}	number of bits whose weights are equal to i

the reliability of PUFs. The main concept of these techniques is to introduce redundancy into the PUF response by generating a syndrome or helper data based on an authentic response. In a typical PUF error correction setting, the error correcting syndrome or helper data is computed based on this response during the enrollment stage. The syndrome or helper data is public information which is later sent to the PUF along with the challenges. To re-generate the same PUF output during the authentication step, the PUF first produces a response from the circuit and then uses the syndrome or helper data to correct the errors in the circuit output, if the number of errors is within the tolerable range. In such as way that the PUF can consistently reproduce the output as in the initialization step if the device is authentic.

Besides of using ECC or fuzzy extractor, other error-correcting methods for PUF-based authentication have also been exploited in the prior literature. The method of utilizing majority voting on reducing errors has been demonstrated in [27]. The use of repetition codes along with conventional syndrome generation using XOR masking has been proposed for PUFs in [28]. Soft-decision encoders and decoders have also been employed to correct PUF response errors [14]. Furthermore, IBS [14], RWA [15] and pattern matching [29] have also introduced to improve reliability of PUF.

3.0.2 Limitations of Prior Works

In fact, the idea of using ECC for PUF based authentication is adopted from communication systems where ECC is used for transmitting digital data over unreliable communication channels. Many communication channels are subject to channel noise that has a certain error probability or signal-to-noise ratio (SNR), and thus ECC can help correct the errors introduced during transmission from the source to a receiver. In most of these applications, each bit among the encoded message is assumed to have a same error-rate. In other words, the error tolerance for each individual bit is identical. ECC will be able to correct noisy message whose number of errors is less than a certain

threshold.

However, for the PUF based authentication, the assumption that each response bit has the identical error-rate is no longer valid. In practice, the actual PUF response is generated based on both the sample of manufacturing process variation and the sample of environmental noise. It is important to note that the sample of the manufacturing process variation is deterministic after fabrication, which is dependent on the challenge and physical characteristics of the PUF circuit. Thus, even if we assume the effect of environmental noise on each PUF response bit is consistent, the error-rate still varies since the sample of manufacturing process variation is different. It is desired to increase manufacturing process variation to improve the uniqueness, as well as to reduce the environmental noise to improve the reliability. In fact, several works have been developed to improve PUF reliability by enhancing the effect of manufacturing process variation. For example, we can select ring oscillator pairs with large count differences to generate response in a ring oscillator PUF [13] or we can only use stable memory cells for signature generation in a SRAM PUF [8, 30–32]. These methods lead to challenge-response pairs (CRPs) with very low error-rate essentially by choosing samples with large magnitude samples of manufacturing process variation. It can also be proved from another perspective that the error-rate or the reliability of each PUF response bit is different.

What worth to be noticed from the previous methods is that, we overlooked some important properties that can be potentially leverage, to improve the efficiency of PUF response error correction by directly employing the existing ECC to correct errors in PUF response. For example, it is meaningless to correct very noisy (close to random) bits in a PUF response. Furthermore, applying error correction mechanism to stable PUF response bits obviously incurs unnecessary overheads. Other error-correcting methods proposed for PUF based applications in the literature, such as fuzzy extractor, pattern matching, IBS and RWA [15], also assign identical error tolerance to each PUF response bit, which do not take the non-uniformity of PUF response bit error-rate into consideration as well. Additionally, hardware implementations of these methods often incur significant area, power, and delay overheads, which scale up quickly as the number of bits of correction increases [16].

Therefore, it is possible to improve the performance by treating each PUF response bit differently based on the error-rate [33]. Intuitively, we can discard or assign a large degree of error tolerance to a PUF response bit if its error-rate is very high (i.e., the sampled magnitude of the manufacturing process variation is very small), so that the reliability can be improved. At the same time, we can assign low or even zero error tolerance to a PUF response bit with a very low error

rate which should be stable under different environmental conditions if the PUF is authentic.

3.0.3 False Negative/Positive Rate

The objective of error-correcting techniques for PUF based authentication is to mitigate the intra-chip variation of an authentic PUF response during the authentication step. However, by accepting the responses whose Hamming distant to the enrolled PUF response are below a certain threshold, the number of responses that can pass the authentication is also increased, which will degrade the security. For instance, number of responses that can be authenticated is increased to $N+1$ if one error is allowed for an N-bit PUF response. Basically, the application of error-correcting techniques for PUF-based authentication exhibits trade-offs between the false negative rate and the false negative rate, which are defined below:

False negative rate (FN): the possibility of a trustable chip fails the authentication process due to environmental variation.

False positive rate (FP): the possibility of a untrustable chip is wrongly authenticated or a random guess attack passes the authentication.

Note that the relationship of the false negative rate and the reliability of a PUF based authentication can be given by

$$FN = 1 - Reliability \tag{3.1}$$

We need to reduce the value of FN to improve the reliability. FP is also a measurement related to the security of PUF-based authentication. Ideally, we would like to achieve very low values for both FN and FP to improve the performance of the authentication. The value of FN will decrease as the increase of the error-correcting capability. However, FP will increase, if more errors can be corrected. The value of FP for an N-bit PUF response with T-bit error correction can be given by

$$FP = \frac{\sum_{i=0}^{T} \binom{N}{i} - 1}{2^N} \tag{3.2}$$

.

3.1 Error-Rate Estimation

The extracted response of a PUF is determined by the physical PUF function and the applied challenge. However, the environment and measurement noise may cause inconsistencies in the expected CRPs. Further, the noise varies greatly in different conditions. It is hard to accurately predict the actual error of each PUF response. By using the PUF model and parameters obtained through the similar procedures in modeling attack and parameter-based authentication, we will be able to predict the error-rate ranking of each PUF response bit under a certain environmental variation.

We demonstrate the basic idea by use of the multiplexer (MUX) PUF. The same concept can be easily adopted to other types of PUFs as well.

3.2 Linear Approximation of Error Probability

3.2.1 MUX PUF Response Error-Rate Estimation

The extracted response of a PUF is determined by the physical PUF function and the applied challenge. The environmental noise also leads to inconsistencies in PUF response. However, since the noise could vary significantly under different environmental conditions, it is impossible to predict the actual error of each PUF response bit precisely. Instead, we will be able to predict the error-rate ranking of each PUF response bit by using the PUF parameters obtained through the similar procedures as in modeling attack and parameter-based authentication.

We demonstrate the basic idea with the multiplexer (MUX) PUF, which can also be applied to other PUFs with slight modifications. A MUX PUF is presented in 2.1. The key observation from Equation 2.1 is that a larger $|r_N|$ leads to a smaller error rate if environmental noise is stable. As a result, we will be able to rank the error rate of each PUF response bit by using the values of $|r_N|$.

We plot the relationship between error rate and $log_2(\frac{max(|r_N|)}{|r_N|})$ under different environmental conditions, as shown in Fig. 3.1. It can be seen that the error rate is linearly proportional to $log_2(\frac{max(|r_N|)}{|r_N|})$, when error rate is in the range of [5%, 40%]. Since i) using highly unreliable PUF response bits for authentication is meaningless and ii) applying error correction mechanism to stable PUF response bits obviously incurs unnecessary overheads, the PUF response bits with error rates

in the medium range are exactly the ones we want to correct errors. In fact, in the proposed method, highly unreliable bits are eliminated and stable bits are directly used for authentication, which will be discussed in next section. Therefore, we can use the linear property to estimate the relative values of error rates for different response bits by using $log_2(\frac{max(|r_N|)}{|r_N|})$.

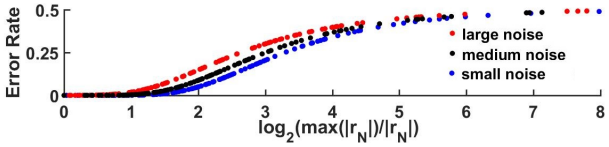


Figure 3.1. Error rate versus $log_2(\frac{max(|r_N|)}{|r_N|})$.

3.3 Response Processing

3.3.1 Response Truncating

As far as we know, if errors in a PUF response exceed the capability of the error-correcting method, the chip would not pass authenticated. Thus, it is neither efficient nor desirable to use PUF response bits with very high error-rates for authentication, which will significantly increase the false negative rate and the design overhead due to the requirement of a stronger error correction technique. Meanwhile, the false positive rate will also be increased, if we increase the error-correcting capability. These high error-rate bits can be discarded by using the methods such as challenge selection [34] to preselect only stable bits for authentication. However, we could still exclude these response bits for authentication to improve the false negative rate by PUF response processing.

Response truncating method utilizes the pre-obtained error-rate rankings and truncates the response bits with very high error-rates before error correction. Fig. 3.2 shows an example. Without loss of generality, we rank the response bits in descending order with respect to the estimated error-rate. In this example, we only truncate the bit with the highest error-rate, i.e., b_{10}. As a result, the remaining response bits will have relatively low error-rates, which could reduce the overall false negative rate. Consequently, a simpler error-correcting method can be applied before authentication to reduce the design complexity, i.e., the parameters *(input length, maximum number of correctable bits)* of the error-correcting method (N, T) can be reduced to $(N - N_{w0}, T)$. Note that the cost of most error-correcting methods decreases with the decrease of both N and T. For example, the area is linearly proportional to the number of tolerated error bits for the design of the BCH codes in [35].

Furthermore, we can also reduce T in the error-correcting method to further reduce the cost of the hardware complexity, while maintaining a similar value of the false negative rate. However, this method may waste several PUF response bits and increase the false positive rate. The expression of FP for the case $(N - N_{w0}, T)$ can be given by

$$FP = \frac{2^{N_{w0}} \sum_{i=1}^{T} \binom{N-N_{w0}}{i}}{2^N}, \tag{3.3}$$

which can be shown to be greater than the original value of FP without response truncating. It can be derived that FP will increase as the increase of N_{w0}, which exhibit a trade-off between the false positive rate and the false negative rate.

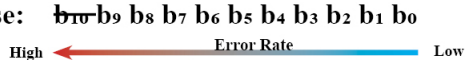


Figure 3.2. An example for response truncating.

3.3.2 Response Duplicating

If a PUF response bit with a very low estimated error-rate flips, we would suspect that the received PUF response might come from malicious adversaries. Therefore, we need to assign low degrees of error-tolerance to these bits.

The main idea of the *response duplicating* method is to increase the input length to the error-correcting block by duplicating low error-rate bits, as illustrated by the example in Fig. 3.3. In this example, two bits with lowest error-rates out of the eight PUF response bits are duplicated. As a result, each error in the two bits with lowest error-rates will lead to two errors in the input of the error-correcting block. Thus, the false positive rate can be reduced, if the error-correcting capability remains the same. The value of FP (false positive) after *response duplicating* can be expressed as:

$$FP = \frac{\sum_{i=0}^{N_{w2}} \sum_{j=0}^{T-2i} \binom{N_{w2}}{i} \binom{N-N_{w2}}{j} - 1}{2^N} \tag{3.4}$$

For example, by duplicating 2 bits with lowest error as shown in Fig. 3.3, with 2-bit error contestability, the value of FP can be reduced from 0.0537 to 0.0371.

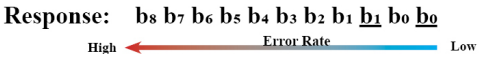


Figure 3.3. An example for duplicating algorithm.

3.4 Response Weighting

As we described above, *response truncating* can reduce the false negative rate, while *response duplicating* can lower the false positive rate. It is possible to combine these two methods to reduce FN and FP simultaneously while achieving an efficiently and reliable error-correcting scheme for PUF based authentication.

In fact, both of these two methods are equivalent to assigning weights to different PUF response bits according to the estimated error-rate ranking. *Response truncating* gives zero weights to bits with highest error-rates, as these bits are dropped before error correction and eliminated for authentication. *Response duplicating* doubles the weight of each duplicated low error-rate bits. We can also duplicate the bits with low error-rates multiple times, i.e., assign different integral weights. Higher weights should be applied to bits with lower estimated error-rates. Furthermore, it is also not necessary to input very reliable response bits which almost never flip under normal environmental condition to the error-correcting block. If the error-rates of these bits are no longer negligible under a very noisy environment, the original PUF authentication with all the response bits would also fail as the number of errors should surpass the error-correcting capability. Therefore, the false negative would not increase if we use these very reliable bits for authentication directly. This property also means we can simplify the error-correcting method by reducing the length of the input by $N_{w\infty}$ (assume the input length still satisfies the minimum requirement for a certain T), while these low estimated error-rate bits can bypass the error-correcting method to be directly used for authentication. In fact, this method is also equivalent to assigning infinite weights to PUF response bits with extremely low error-rate. Thus, it can be noted that N_{w0}, N_{w2}, and $N_{w\infty}$ are the same as N_{w0}, N_{w2}, and $N_{w\infty}$, respectively.

We generalize these methods to a so-called *response weighting* approach. Specifically, each response bit is assigned a weight according to the estimated error-rate ranking in the PUF response, ranging from 0, 1, 2, 3, ..., to ∞. A simple example is shown in Fig. 3.4. The first step is to select very high error-rate response bit(s) to truncate, i.e., b_8 in this example, and very low error-rate response bit(s) to bypass the error-correcting block, i.e., b_0. Then, we assign different weights to the

remaining bits, according to the estimated error-rate ranking of these bits. Higher weights will be assigned to bits with lower error-rates. We can maintain a same input length of the error-correcting block by choosing the weights appropriately. In this example, we assign weight of 2 to b_2 and b_1 , which have the lowest error-rates in the remaining bits. The length of the resulting sequence is still 9. Therefore, no modification to the error-correcting block is needed in this case.

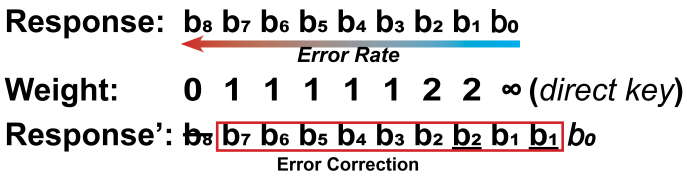


Figure 3.4. An example for PUF response weighting.

Clearly, the manufacturing process variation of the PUFs will be more significant than the environmental variation, under a normal condition; otherwise, PUFs would not be considered as promising security primitives for chip-unique ID generation. In other words, it is safe to say that most of the PUF response bits will have relatively low error-rates, while there might be only a few bits have considerable high error-rates in a PUF response. This property can also be verified by our experimental results as shown in Fig. 3.5. For this 127-bit PUF response, even the noise is increased to 15% correspondingly, there are still 43 bits that are very stable, i.e., error rates are less than 0.3%, where there are only 11 bits with error rates larger than 20%. Thus, we should use a higher value of N_b, compared to N_t.

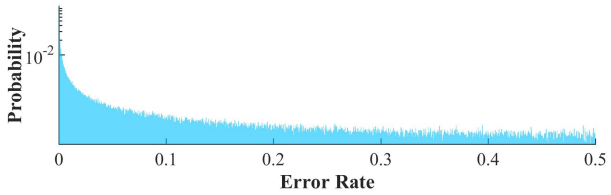


Figure 3.5. The error rate distribution (under a same noise).

A detailed example is illustrated in Fig. 3.6, where $N = 127$, $T = 13$, $N_t = 4$, $N_{w1} = 95$, $N_{w2} = 16$ $N_b = 12$. Note that $N_t + N_{w1} + N_{w2} + N_b$ should be equal to N. The length of the sequence after the *response weighting* is $N_{w1} + 2N_{w2} = 127$, which is the same as original PUF response length. For the 127-bits PUF, only 4 PUF response bits with the highest error-rates are truncated ($N_t = 4$); 12 bits with the lowest error-rates are directly used for authentication without any error correction ($N_b = 12$), while the next 16 bits with relatively low error-rates are duplicated

and combined with the rest of PUF response bits for error correction ($N_{w2} = 16$). Our experimental results show that the false negative can be reduced from 6.5% to 1.5% by using the proposed method. The expression for the value of FP for this method is shown in Equation 3.5:

$$FP = \frac{2^{N_{w0}} \sum_{i=0}^{N_{w2}} \sum_{j=0}^{T-2i} \binom{N_{w2}}{i} \binom{N_{w1}}{j} - 1}{2^N} \tag{3.5}$$

It can be calculated that the false positive rate can be reduced from 12.59×10^{-22} to 9.15×10^{-22} as well by employing the proposed *response weighting*.

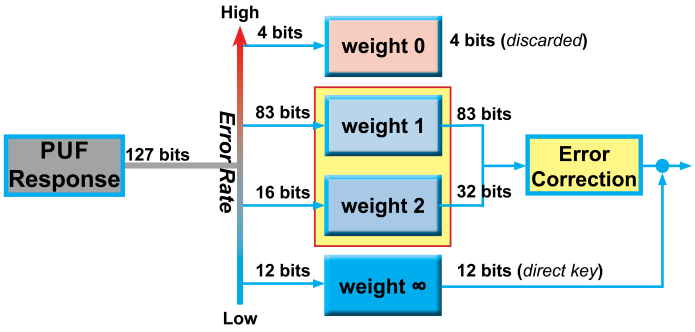


Figure 3.6. A detailed example of PUF response weighting.

The objective is to assign weights to different PUF response bits so that each weighted bit has similar error-rate. Therefore, the resulting sequence will have similar characteristics as the messages transmitted over a communication channel ,which can fully utilize the computational power provided by the error-correcting methods. However, since we can only assign integral or infinite weights which is indeed an integer linear programming (ILP) problem, the solution cannot guarantee that all the weighted response bits will have the exact same weighted error-rate. The objective of the problem is equivalent to minimize the variance of the weighted error-rate, i.e., $Var(w_i e_i)$, where w_i and e_i represent the weight and the estimated error-rated of response bit i, respectively. Note that the error-rate for each PUF response bit can be estimated under a certain environmental condition and can be scaled with the variance of the noise, as solution is only dependent on the relative values or orders of the error-rate. The ILP problem can be formulated as:

$$
\begin{aligned}
\{w_1, w_2, w_3 ...\} = \quad & \arg \min_{w_i} \{Var(w_i * e_i)\} \\
\text{subject to} \quad & \sum_{i=1}^{N} w_i = N \\
& w_i \geq 0 \\
& w_i \in \mathbb{Z} \text{ or } \infty
\end{aligned}
$$

Since ILP falls into the category of NP-hard, we can use heuristic algorithms such as tabu search [36], hill climbing [37], and simulated annealing [38] to solve the problem to obtain the optimized weights. Theoretically, we can assign arbitrary weights to different PUF response bits according the error-rate ranking. However, a large integral weight would significantly reduce the entropy. Therefore, we should set an upper bound U to the weight. Weights in the solution exceed U will be set as infinite.

3.5 Algorithm

The overall algorithm can be summarized in Algorithm 3.

Algorithm 3 Efficient PUF Error Correction through Error-Rate Prediction

1. Obtain PUF parameters through machine learning techniques.
2. Calculate the error-rate for every response bit based on the PUF model and the obtained PUF parameters.
3. Assign weights to the PUF response bits according to the estimated error-rates. Optimized weights can be solved by an ILP problem.
4. Process the PUF response based on the weights.
5. Compute the syndrome or helper data of the error-correcting block based on the resulting sequence from Step 4, which will later be used to correct errors during the authentication phase. Store the resulting sequence for authentication.

Table 3.2. False Negative Rates of PUF-based Authentication before and after Employing *Response Truncating*

N	T	N_{w0}	False Negative Rate		
			V_{intra}=5.5%	V_{intra}=7.8%	V_{intra}=8.6%
127	13	0	2.97%	20.60%	31.46%
		5	0.21%	6.22%	11.29%
127	12	0	5.76%	30.25%	41.78%
		5	1.21%	10.03%	17.27%
127	11	0	10.59%	40.02%	53.63%
		5	2.52%	15.26%	25.91%
127	10	0	18.08%	52.08%	65.51%
		5	3.53%	23.37%	36.02%

Table 3.3. False Negative Rates of PUF-based Authentication before and after Employing *Response Duplicating*

$N + N_{w2}$	T	False Negative Rate					
		V_{intra}=5.5%	7.8%	8.6%	10.2%	12.1%	15.0%
$127 + 0$	13	2.98%	20.89%	31.31%	51.64%	74.21%	92.95%
$127 + 8$	13	2.98%	20.89%	31.31%	51.64%	74.21%	92.95%

3.6 Experimental Results

We continue to use MUX PUFs in our experiments. We simulate parameters and manufacturing process variations in the MUX PUF model based on prior theoretical and experimental results in [1, 16, 26]. We use linear regression (LR) to estimate the delay difference of each MUX stage. According to Equation (2), we can calculate the total delay difference for a given challenge, i.e., r_N. Then, we can rank the error rates of the PUF response bits by using the reverse order of the absolute values of r_N, i.e., $|r_N|$. We vary the environment variations and parameters in the proposed method to examine the performances of the proposed algorithms under different environmental conditions, which are characterized by the PUF intra-chip variations.

We examine the performances of different weight assignments, which are summarized in Table 1. Here, we set the weight limit $U = 3$ so that only 5 possible weights can be assigned, i.e., 0 (truncate), 1 (keep same), 2 (duplicate once), 3 (duplicate twice) and $\ldots \infty$ (no correction and directly used as key). The weights obtained by solving the optimization problem (i.e., $N_{w0} = 9$, $N_{w2} = 7$, $N_{w3} = 14$, $N_{w\infty} = 26$) are also included.

It can be concluded from Table 1 that the false negative rate can be significantly reduced by using the proposed methods. In addition, it can be observed that the optimal weights lead to the lowest false negative rate. It can also be seen from Fig. 3.7 that the false negative rate decreases as the increase of error correctability, T. We plot the relationship between false negative rate and T as shown in Fig. 3.7. However, the area complexity of the error-correction also increases, since the costs of most error-correcting methods increase with the increase of both N and T. For example, the area is linearly proportional to the number of tolerated error bits for the design of the BCH codes in [35]. Therefore, we will be able to reduce the cost of the error-correction hardware implementation by employing the proposed method. According to our experimental results, we can reduce the error-correctability from 13 to 8 to save the hardware cost, while maintaining similar false negative rates.

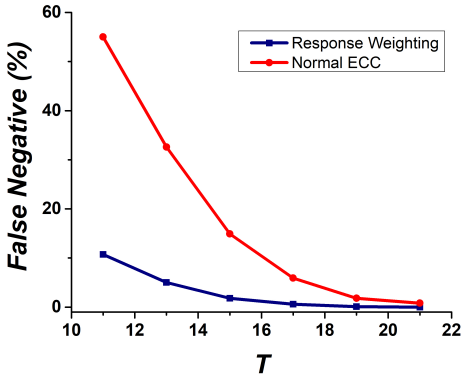


Figure 3.7. False negative rate versus ECC correctability T.

The false positive rate will also be affected by using the proposed method. For example, when $N_{w0} = 4$, $N_{w2} = 10$, $N_{w3} = 3$, $N_{w\infty} = 12$), both the false negative rate and the false positive rate will be reduced, compared to the conventional error-correction. However, the optimal weights lead to slightly larger false positive rates, as the objective of the optimization problem is to minimize the false negative rate instead of the false positive rate. As a result, the specific weight assignment strategy should be chosen carefully based on the application requirement.

Table 3.4. False negative/positive rates of PUF-based authentication before and after employing *response weighting* with different weights

N	T	N_{w0}	N_{w2}	N_{w3}	$N_{w\infty}$	False Negative Rate			False Positive Rate
						$V_{intra} = 5.5\%$	7.8%	8.6%	
127	13	0	0	0	0	2.97%	20.60%	31.46%	1.26×10^{-21}
		5	5	0	0	0.36%	5.72%	11.73%	1.42×10^{-20}
		4	16	0	12	0.50%	8.26%	14.59%	5.38×10^{-22}
		4	10	3	12	0.56%	7.72%	14.59%	1.08×10^{-21}
		4	6	5	12	0.16%	6.72%	12.60%	1.70×10^{-21}
		9	7	14	26	0.04%	1.71%	4.14%	3.42×10^{-21}
127	10	0	0	0	0	18.08%	52.08%	65.51%	1.34×10^{-24}
		5	5	0	0	3.73%	22.77%	35.32%	1.92×10^{-23}
		4	16	0	12	5.19%	28.51%	41.45%	1.28×10^{-24}
		4	10	3	12	5.25%	28.58%	41.74%	2.24×10^{-24}
		9	7	14	26	0.61%	9.42%	16.98%	1.22×10^{-23}
127	8	0	0	0	0	40.76%	75.96%	83.49%	8.44×10^{-27}
		5	5	0	0	12.52%	46.10%	59.51%	1.41×10^{-25}
		4	16	0	12	16.17%	52.79%	64.57%	1.40×10^{-26}
		4	10	3	12	16.35%	51.94%	65.02%	2.21×10^{-26}
		9	7	14	26	3.41%	22.47%	36.73%	1.73×10^{-25}

It can be concluded from Equation (4) that the false positive rate decreases with the increase of $N_{w\infty}$, since the security will be improved if more bits are assigned zero error-tolerance. Fig. 3.8 shows the relationship between the false negative and $N_{w\infty}$ when $T = 20$. It can be seen that the

false negative rate remains the same when the $N_{w\infty}$ is less than 10 for a 127-bit PUF response. This is due to the fact that these bits are very stable so that use these bits directly as the key would not fail the authentication. However, false negative rate could increase when the value of $N_{w\infty}$ becomes relatively large, e.g., 40, which will affect the performance of the PUF-based authentication adversely.

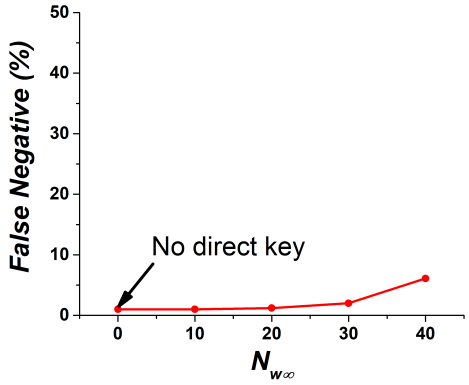


Figure 3.8. False negative rate versus infinity weight (by-passing) $N_{w\infty}$ ($T = 20$).

We also plot the relationship between the false negative rate and the value of N_{w0}, as shown in Fig. 3.9. Note that N_{w2} needs to be increased by N_{w0} to maintain the same PUF response length. We can see that the false negative rate can be significantly reduced by increasing N_{w0}, as more unreliable bits are eliminated. However, the total combinations of the weighted PUF response will decrease.

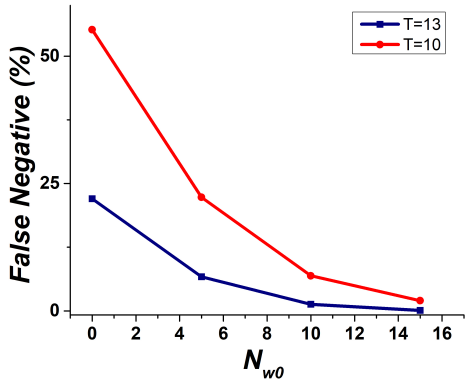


Figure 3.9. False negative rate versus weight 0 (truncating) N_{w0}.

In summary, by using the proposed *response weighting*, the false negative rate and the false positive rate can simultaneously be reduced by assigning the weights to different PUF response bits

appropriately, while the lowest false negative rate can be obtained by solving the integer quadratic programming problem as formulated in Equation (3).

Chapter 4

Modeling Attack PUF by Active Learning

Security and uniqueness are essential to a PUF's performance against attacks. In the past decades, numerous hardware security techniques have been proposed, which include authentication based hardware protection approaches [39], logic obfuscation [40], secure manufacturing and testing [41, 42], side-channel countermeasures [43, 44], and methods for detecting malicious hardware Trojan insertions [45, 46]. Of these, PUF is a promising security primitive that extracts secrets from complex properties of a physical material rather than storing them in a non-volatile memory.

With the rapid advancement of hardware security techniques, however, the incentive for defeating them also increases. Various effective countermeasures or attacking methods targeting these hardware protection methods have also been developed [26, 47–50]. The main challenge of hardware protection is that the adversary may access to the devices physically, which has made non-invasive or semi-invasive attacks on hardware devices as critical threats [51]. On the other hand, machine learning has been quite pervasive in a wide range of applications in recent years, such as robotics, natural language processing, healthcare, economics, and marketing [52]. Although such algorithms have already been applied to the security field, mostly as a means of attack [21, 23]. Unsurprisingly, they are quite powerful making them particularly threatening to existing hardware security techniques. Modeling attack using machine learning algorithms on PUFs for example can compromise most PUFs that previously were deemed "strong" [21].

To maintain the competitive edge requires retooling hardware security techniques with these start-of-the-art attacking methods. In this work, we study the applications of active learning, a subfield of machine learning, on PUF modeling attacks. While active learning is well established in software or network security [53–56], it is rarely used in hardware security, with IC camouflaging [57] and SAT-based deobfuscation [58] the only hardware security related applications. Even though various machine learning methods have been applied to attack different types of PUFs in the literature, modeling attack by active learning remains unexploited. We argue that passive learning methods might fail to capture the true costs of attacks on PUF carried out by *intelligent and adaptive adversaries*. The main contribution of this wok is to leverage the advantages of active learning to improve the efficiency of modeling attacks, which is of great importance in practice if modeling attack requires a cost that is proportional to the number of test vectors (i.e., PUF CRPs). Besides, we also describe the strategies for selecting parameters employed in active learning, which are specific to the applications of PUF modeling attacks under different environmental conditions.

4.1 Active Learning

Active learning is a subfield of machine learning in which data with the highest uncertainty are trained, while avoiding sampling data with labels implicitly determined, as opposed to passive learning that selects samples randomly [53, 56]. Uncertainty sampling and query-by-committee are two popular query strategies. Uncertainty sampling [56] measures the confidence of the classifier on candidate instances, which adaptively selects the instances with the least confidence for training. Query-by-committee [56] measures the agreement among a committee of classifiers and then selects the instances with the highest disagreement, it is very effective in reducing the amount of training data required by having highly uncertain instances annotated as training samples. For example, active learning techniques can reduce the annotation costs of parse selection by 73% [59]. Active learning is also well motivated in security applications where data are abundant but labels are few or expensive to obtain (e.g., intrusion detection/spam filter) [54, 59, 60]. Companies such as Google are already using active learning approaches to minimize the instance for tasks such as labeling malicious advertisements and phishing pages [61].

4.2 PUF Modeling Attack Using Active Learning

The modeling of each response bit can be considered as a multi-input binary classification problem. For a MUX PUF, the output label is determined by the sign of the total delay difference r_N before the arbiter. Therefore, $r_N = 0$ or the $sign()$ function ideally should be modeled as the class boundary by a modeling attack. Each challenge essentially corresponds to a sample in the distribution of total delay difference. In other words, each CRP has different entropy, e.g., challenge with a smaller absolute value of total delay difference is closer to the arbiter boundaries and is more informative. As a result, we can leverage the advantages of active learning to improve the efficiency of PUF modeling attacks, as opposed to using passive sampling with randomly selected CRPs.

4.2.1 Active Learning Scenarios

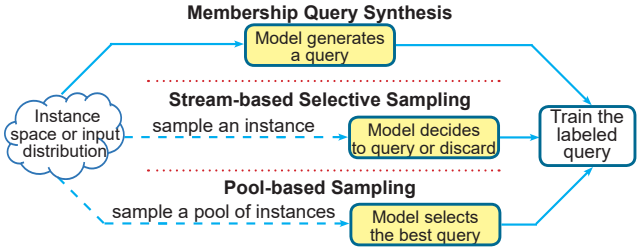


Figure 4.1. Scenarios of active learning. 1) Membership query synthesis: the learner generates label to query; 2) Stream-based selective sampling: the learner decides whether to query or discard an instance after drawing it randomly from the distribution; 3) Pool-based sampling: queries are selectively drawn from the pool by a model.

In the context of PUF modeling attack, we usually consider the adversarial threat model that the adversary can query the PUF oracle for response to a challenge $c \in Q$, where Q is a pool of candidate challenges and is polynomially bounded. The adversary then needs to predict the response for a future challenge that is not in Q. This threat model fits in a scenario where an adversary can only eavesdrop the CRPs being used for authentication and try to use the collected CRPs to reconstruct the PUF model. In this case, the challenges used for training are usually assumed given or randomly selected in prior works. However, if we consider a stronger adversarial threat model that the adversary can query arbitrarily, randomly selecting the CRPs for training obviously is not optimal. An *intelligent and adaptive* adversary would use the current information in each iteration to obtain a lower prediction error with a bounded number of CRPs. We can also consider the scenario that the number of query is unbounded but each query requires a cost. In this

case, it is important to improve the efficiency of modeling attacks.

Fig. 4.1 shows the three most common scenarios. The scenarios of stream-based selective sampling and pool-based sampling fit the threat model of PUF modeling attack perfectly, as responses are usually difficult or expensive to obtain, but challenges are ample. In this work, we consider the pool-based sampling active learning scenario for PUF modeling attack. The adversary tries to sample the most informative or efficient instances from a given pool of challenges to query for the response and then use them for training.

4.2.2 Sampling Strategies

4.2.2.1 Uncertainty Sampling

The simplest and most commonly used query framework in active learning is uncertainty sampling. In this framework, an active learner queries the instances whose labels are the least certain. Various heuristic uncertainty metrics have been proposed to select the samples in active learning. Since PUF modeling attack only involves binary classification problems, most of these uncertainty metrics (e.g., least confidence, marginal sampling, and entropy) may have a similar performance. In this work, we use marginal sampling to perform the uncertainty sampling. The score for each challenge c is computed by $C_M = \arg \min_c \{|P_\theta(R_0|c) - P_\theta(R_1|c)|\}$, where $P_\theta(R_0|c)$ and $P_\theta(R_1|c)$ represent the probabilities of the response bit (0 or 1) for challenge c according to the current PUF model. In each iteration, we always select the CRP with the highest score among the remaining candidates in the pool.

4.2.2.2 Estimated Error-Rate as the Uncertainty Metric

We can also develop a PUF-specific uncertainty metric for active learning. For example, by using machine learning algorithms, we can estimate the delay difference of each stage of MUX PUF after collecting a number of CRPs. Then we will be able to rank the error-rate (i.e., $1 - \mathit{Reliability}$) of each sample according to the estimated total delay difference, $|r_N|$. The error-rate of each PUF response bit is similar to the uncertainty metric, which can be used for selecting samples in active learning. Essentially, a challenge leads to a smaller $|r_N|$ would have a higher value of the uncertainty metric.

4.2.2.3 Query-by-Committee

An alternative yet more theoretically-motivated sampling strategy framework is query-by-committee (QBC) [62]. The QBC approach involves maintaining a committee of models on the same labeled set, but represent competing hypotheses. Each committee member then votes on the labelings of the candidates. The most informative query is considered to be the instance which they most disagree. In the literature, various ensemble learning methods have also been invented [63]. In this work, we examine the performance of QBC strategy with entropy query-by-bagging (EQB) learning method [64], which combines entropy sampling and query-by-bagging.

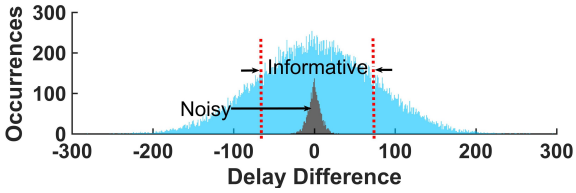


Figure 4.2. A MUX PUF total delay difference distribution.

4.2.3 Noisy PUFs

Noisy PUFs are similar to noisy oracles, which may present in both adversarial and non-adversarial settings. More details about noisy PUF are studied in Chapter 5. Intuitively, conventional active learning is not advantageous for learning noisy PUFs. As shown in Fig. 4.2, CRPs in the regions that are more informative also have higher error rates. Thus, active learning tends to select a large amount of mislabeled CRPs in a noisy condition, which consequently leads to high prediction error. Therefore, it is important to study the implications between informative instances and mislabeled instances.

Table 4.1. Prediction error (%) of active learning with different sampling strategies (under different environmental variations)

Sampling strategy	Var_{intra}: 3.5%					Var_{intra}: 13.5%				
	Number of CRPs in training set					Number of CRPs in training set				
	350	550	1250	1750	6000	350	550	1250	1750	6000
Random	11.50	8.10	6.12	5.90	4.30	17.62	15.18	12.42	11.70	11.12
Marginal	8.85	6.30	4.77	4.62	4.53	16.53	13.60	12.00	11.85	11.80
EQB	9.35	5.90	4.75	4.33	4.20	15.37	13.12	12.10	11.59	11.98
Error rate-based	7.28	5.52	4.07	3.43	2.70	11.60	10.85	10.27	10.32	10.38

We describe two query strategies for noisy PUFs.

4.2.3.1 PUF response Majority Voting

The first method is to use majority voting to improve the quality of PUF CRPs under a noisy condition, before employing the active learning. The label is determined by the majority after querying the same instance K times. However, this method would increase the number of queries per CRP from 1 to K. In other words, if the total number of queries are bounded, the number of CRPs will be reduced to $\frac{1}{K}$ of the original number. Therefore, the value of K needs to be selected appropriately to guarantee sufficient samples as well as high quality data.

4.2.3.2 Eliminate Unreliable CRPs by thresholding

Instead of always using the least confident CRP (also with the highest error rate) for training, we can use the T-th least confident CRP to reduce the impact of mislabeled responses. In this case, the most $(T-1)$ unreliable CRPs will be excluded from training, where the reliability or the error rate is predicted by the current learned PUF model, which is similar to several prior works on improving PUF reliability [31, 32]. For example, a machine learning based method was proposed in [22] to enhance the PUF reliability by setting a lower-bound threshold of estimated $|r_N|$ for the CRPs. Similar methods can be applied to eliminate unreliable CRPs as a pre-processing step. As a result, the prediction error can be reduced. By using the estimated error rate as the uncertainty metric, we can establish an absolute value of threshold T for the estimated total delay difference, i.e., only the CRPs with $|r_N| > T$ will be used for training. We can also use a percentage threshold $T\%$ in eliminating unreliable CRPs if given a pool (assume the candidate CRPs in the pool are randomly drawn). We can always select the $\lceil T\% \times S_P \rceil$-th least confident CRP to query and train in each iteration, where S_P is the size of the remaining candidate pool.

4.3 Experiments

In this section, we present the experimental results of incorporating active learning techniques into modeling attack on a 64 stage MUX PUF. In addition, the proposed methods can also be applied to other kinds of PUFs. We employ the pool-based active learning scenario and randomly generate a sufficient number of CRPs as the sampling pool. Among them, 200 CRPs are randomly selected as the initial training set, while 6000 CRPs are used for testing. We examine the performance of marginal sampling (MS), entropy query-by-bagging (EQB), error-rate based sampling (ER)

under different environmental conditions (intra-chip variations Var_{intra} = 3.5% and 13.5%) along with the proposed strategies for noisy PUFs. We use support vector machine (SVM) to train the instances with an iteration step of 20, i.e., the PUF model and the uncertainty metric are updated after every 20 CRPs.

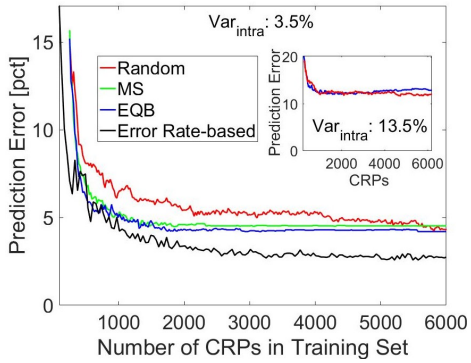


Figure 4.3. Prediction error of different sampling strategies.

4.3.1 Sampling Strategies

The performances of different methods with cross-validation are shown in Fig. 4.3. It can be seen that the prediction errors of modeling attack on a MUX PUF vary with different sampling strategies. All of these active sampling strategies show superior performance compared to random (passive) sampling. For example, after trained with 1250 CRPs, the prediction error of random sampling is only 6.12%, while the prediction errors of MS, EQB, and ER are 4.77%, 4.75%, 4.07%, respectively. To obtain a prediction error of 5% (Var_{intra} = 3.5%), 2790 CRPs are required in passive learning, while only 811 CRPs are required in active learning. The random sampling performs better compared to MS for large datasets, as MS could be negatively influenced by information overlap [65] and noisy information. The prediction errors at several data points are summarized in Table 4.1.

4.3.2 Active Learning under Noisy Conditions

As we discussed above, the performance of active learning will be degraded if the oracle is too noisy. Our experiments show consistent results. When the noise is large (e.g., Var_{intra} = 13.5%), the EQB actually perform poorer than the random sampling, as shown in the inset of Fig. 4.3. We investigate the trade-offs between active learning efficiency and prediction error by using the proposed approaches.

4.3.2.1 PUF Response Majority Voting

We use the PUF response voting approach to pre-process the CRPs and then employ EQB as the active learning sampling strategy. Fig. 4.4 shows the results under $Var_{intra} = 3.5\%$ with $K = 1, 3$, and 5. It can be seen that the prediction error is improved after implementing majority voting on CRPs. If we consider a same number of queries (number of queries = number of CRPs $\times$ K), the majority voting pre-processing step still exhibit advantages when number of queries is 3000, i.e., the prediction errors are 4.37%, 4.30% and 3.26% for $K = 1, 3$, and 5, respectively. Note that the performance of majority voting will be degraded, if the number of CRPs is too small. Thus, the value of K should be chosen according to the environmental conditions and the limit of query numbers.

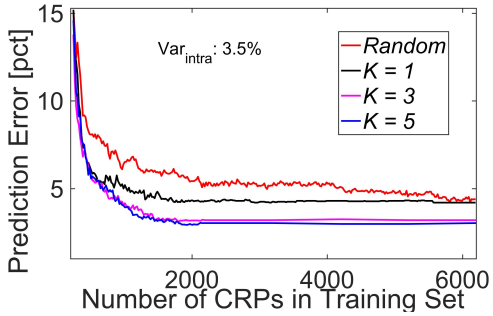


Figure 4.4. Prediction error of PUF response majority voting.

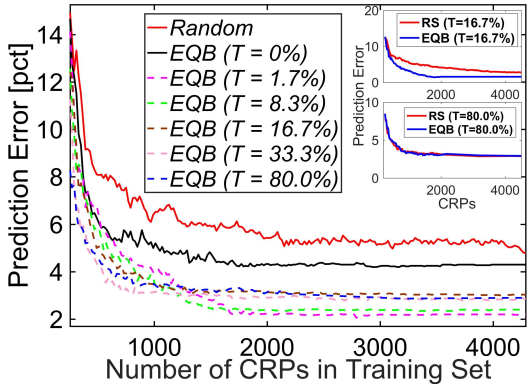


Figure 4.5. Prediction error of different threshold values.

4.3.2.2 Eliminate Unreliable CRPs by Thresholding

An alternative method is to delete the least confident CRPs in the candidate pool by using the proposed thresholding methods to improve data quality. The confidence is computed by the uncertainty metric in different sampling strategies or by using the estimated total delay difference.

The results for $Var_{intra} = 3.5\%$, which is a typical error rate of PUF applications, are shown in Fig. 4.5. It can be observed from Fig. 4.5 that the prediction error reduces from 4.42% to 2.43% after deleting 1.7% CRPs with the least confidence in the candidate pool. In this case, most of the mislabeled data are deleted. However, if we further increase the threshold, the final prediction error will increase, as the most informative CRPs are also deleted. Therefore, when using active learning for PUF modeling attack under noisy environmental conditions, the threshold T should be selected appropriately. On one hand, if the threshold is too large, the remaining CRPs will not be very informative which would lead to poorer learning performance. On the other hand, if the threshold is too low, there are still a large number of error-injected CRPs in the candidate pool, which would reduce the learning efficiency as well as the modeling accuracy. The value of T should increase with the environmental noise.

Chapter 5

Noisy PUF

5.1 PUF Security

With the rapid advancement of hardware security techniques, however, the incentive for defeating them also increases. Various effective countermeasures or attacking methods targeting these hardware protection methods have also been developed [26,47–50]. On the other hand, machine learning has been quite pervasive in a wide range of applications in recent years, such as robotics, natural language processing, healthcare, economics, and marketing [52]. Many modeling attack techniques were developed, as presented in 2.2.1.1, including some state-of-art modeling attack techniques. For example, the active learning can be used to modeling attack MUX PUF with limited resources [66]. Unsurprisingly, they are quite powerful making them particularly threatening to existing hardware security techniques. As we discussed in previous chapters, modeling attack using machine learning algorithms on PUFs for example can compromise most PUFs that previously were deemed "strong" [21].

Along with these attacking methods, various countermeasures have also been developed in recent years to improve the resistance of PUF against modeling attacks: 1) The first approach is to increase the learning complexity by adding non-linearity, including feed-forward PUF (FF-PUF) [67], XOR PUFs [5,13,68–70], non-linear mirror current based PUF [71], subthreshold current array PUF [72], double-layer strong PUF [73], coin flipping PUF [74], etc. However, these methods would significantly degrade the reliability under noise environmental conditions. 2) The second category uses the concept of obfuscation, which masks the challenge, response or some other information

of a PUF to improve its resistance against modeling attacks [75, 76]. Protocol-level access control techniques that limit the direct exposure of CRP information have also been proposed, such as controlled PUF [26] and server-managed CRP lockdown protocol [77]. 3) Recently, several methods for developing cryptographically-secure PUFs have been proposed, which integrates emerging cryptographic techniques such as learning parity with noise (LPN) and learning with errors (LWE) [78,79]. However, these methods suffer from very significant overhead.

Besides machine learning techniques, side-channel attack (SCA) has also been utilized to infer the secret information of a PUF. For example, SCA can directly infer the value of each path before the XOR operation in a parallel XOR PUF [80]. Consequently, modeling attack can be applied to each PUF and hence the overall XOR PUF is compromised. Furthermore, this hybrid SCA/ML attack is also extremely effective in compromising obfuscation based techniques, since it could obtain the internal values of these PUF circuit so that the original PUF is isolated from pre/post processing components [26, 80–82].

Researchers have developed several methods to protect PUF from side-channel attacks by reducing the correlation coefficient between data and physical signal variations. A masked advanced encryption standard PUF was proposed for hardware authentication against hybrid SCA [83]. In [84] a logic style that has constant power consumption and a place and route approach have been used to protect IC. The lightweight PUF unitizes parallel PUF structures to make PUF resistant to circuit faults, reverse engineering and other security attacks.

5.1.1 Contribution

According to the discussion above, one promising direction for developing efficient yet secure PUFs entails increasing the complexity of the underlying PUF circuit (i.e., the first category). However, most of these prior methods essentially trade reliability for security. For instance, the noise on each branch of a parallel XOR PUF contributes to the overall reliability, which makes the PUF extremely sensitive to environmental conditions. Furthermore, a small hamming distance between multiple PUF evaluations may result in very large Hamming distance after pre/post-processing (e.g., Hash function). Theoretically, if we further increase the nonlinearity or the effect of noise (i.e., approaching to a true random number generator), the challenge-response behavior would not be possible modeled, since it is dependent on the unpredictable environmental noise. However, these outputs are not consistent under different evaluations, which cannot be used as chip signature.

In this section, we propose a novel concept, noisy PUF, which achieves high security and high reliability simultaneously. The concept is based on observations that only a small subset of reliable CRPs are required in most applications and the reliability varies among different CRPs. In other words, a modeling attack resistant yet unreliable PUF is still usable, if the designer can find out which CRPs have sufficient degrees of reliability. The proposed methodology is extended from prior works on reconfigurable PUFs [85], which have mechanism to update the challenge-response pairs unpredictably.

5.2 Methodology

The overall methodology for the proposed noisy PUF is summarized in Fig. 5.1. We utilize the one-time-programmable (OTP) memory [86] to control the configure data of a reconfigurable PUF. After fabrication, the default values of the OTP memory configure the PUF into a reliable PUF, which is similar to conventional PUFs that are reliable but easy to model. The PUF is modeled using machine learning based on our previous works [22, 66]. The parameters of the reliable PUF circuit are obtained. In most cases, the modeling attack can achieve nearly 100% accuracy with a small number of CRPs under a stable test environment, especially with the help of active learning as we described in [66].

Then, the PUF is reconfigured using the OTP by the designer or though a trusted vendor. It has been reported in prior works that if certain randomness properties can be met, PUFs can be impervious to general machine learning attacks [5, 73, 75, 78]. Therefore, reconfigurable PUF designs that can configure a conventional reliable PUF to one of these structures with sufficient randomness can be adopted in the proposed methodology. The objective is to create sufficient amount of inherent randomness and produce highly nonlinear model to prevent modeling attacks from deriving the model precisely with a reasonable number of CRPs. For example, we can configure a linear MUX PUF to highly nonlinear feedforward PUFs with a large number of feedforward paths by using the reconfigurable blocks developed in [85]. Most of the CRPs generated by the reconfigured PUF will be unstable, which could make the modeling attack much different to perform. Thus, the reconfigured PUF could achieve high resistance to modeling attacks during application.

However, in the proposed scheme, it is still feasible for the designer to obtain several reliable CRPs by using the circuit parameters from the reliable CRPs before configuration. Note these

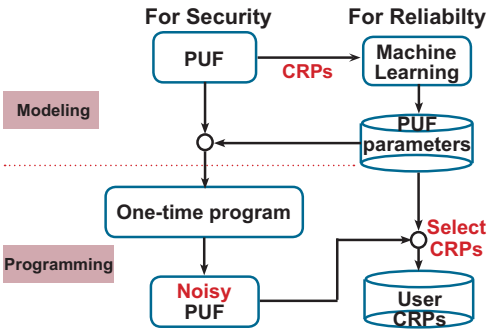


Figure 5.1. The configuration flow of the proposed noisy PUF.

information are not public, which are extremely difficult to infer during application as the adversary can only access to the noisy PUF. The OTP prevents the adversary from configuring the PUF to the reliable PUF again. In the proposed methodology, only these reliable CRPs are used for authentication. Although the environmental noise can vary significantly, it is fair to assume a similar noise level as common TRNGs. Therefore, it is essential to analyze the underlying model to obtain a lower boundary of the required inherent randomness under this assumption. The modeling attack resistance should relate to the number of non-linear components in a PUF design (e.g., ways of XOR PUF or number of reconfigurable components). In this work, we use the delay based arbiter PUFs as an example to illustrate the proposed methodology. For example, as shown in Fig. 5.2 that the delay difference of different CRPs in an arbiter PUF follows a nonuniform distribution, the reliability of each CRP varies. Therefore, although the overall reliability of the reconfigured noisy PUF will be very low, the selected reliable CRPs (e.g., CRPs that are far away from the decision boundary in this case) can still maintain a high degree of reliability.

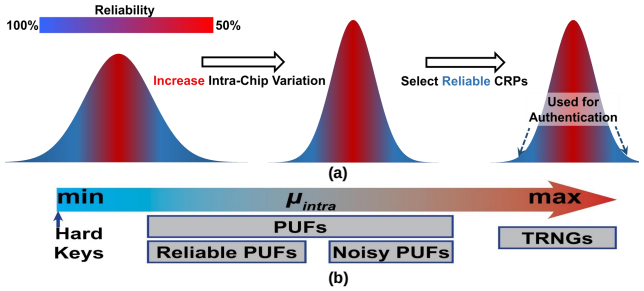


Figure 5.2. (a) The concept of noisy PUF: increase the intra-chip variation so that the data for modeling attack are too noisy to form an accurate model, while only a small portion of the reliable CRPs are used for authentication. (b) Situating the noisy PUFs.

We situate the proposed noisy PUF in Fig. 5.3.

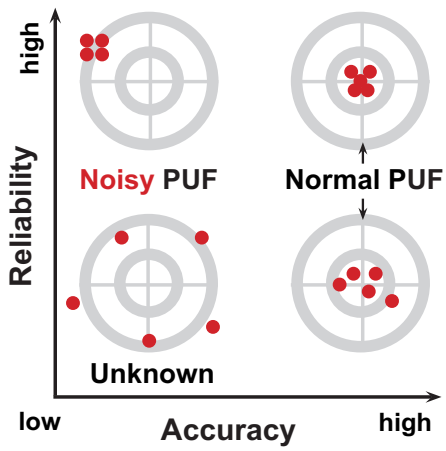


Figure 5.3. The noisy PUF reliability *vs* modeling attack accuracy.

5.3 Case Study: Reconfigurable Delay-Compensation PUF

A possible method for increasing the inherent randomness of a PUF is to compensate the total delay difference of aa arbiter PUF under a specific challenge. In particular, we propose to build the delay compensation scheme based on the design of reconfigurable feed-forward PUF, as shown in Fig. 5.4. The delay difference of each stage can be estimated by using machine learning techniques on the reliable linear arbiter before the reconfigurable. Then, the designer can configure each reconfigurable feed-forward path accordingly to reduce the magnitude of total delay difference, i.e., reduce the reliability under the same environmental condition. For example as shown in Fig. 5.4, if the accumulated delay difference from the *0*-th to the f-th stages is positive/negative (i.e., the top path faster/slower than the bottom path), a feed-forward path is used as the path selection bit of the (f+d)-th stage, which selects the paths such that the top path between the (f+*1*)-th and the (f+d)-th stages is the slower/faster than the bottom path. Consequently, the total delay difference is compensated. This method not only introduces non-linearity by adding feed-forward path, but also amplifies the effect of noise by reducing the magnitude of total delay difference. A considerable number of reconfigurable feed-forward paths needs to be inserted into the original arbiter PUF to meet the randomness requirement. The starting and ending stages of each reconfigurable feed-forward path can be chosen arbitrarily by the designer. In our experiment, we always use reconfigurable feed-forward paths with length d.

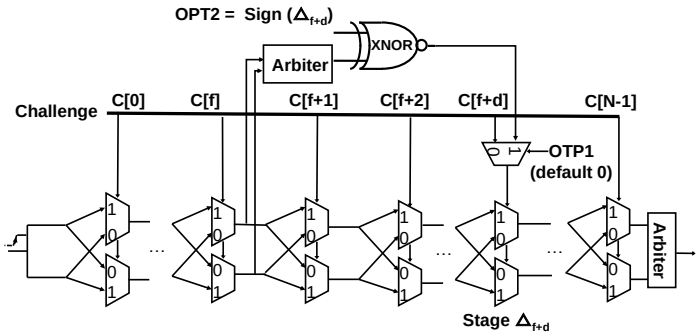


Figure 5.4. One reconfigurable feed-forward path of the proposed reconfigurable delay-compensation PUF.

5.3.1 Reconfiguration

There are two phases of the reconfigurable PUF: designer phase and user phase, which correspond to reliable PUF and noisy PUF, respectively. The proposed reconfigurable algorithm is shown below. We continue to take the reconfigurable feed-forward path in Fig. 5.4 as the example. In the designer phase, both OTP1 and OTP2 are in the default state, 0, which configure the PUF as a linear arbiter PUF. Based on the estimated delay difference of each stage by using machine learning, the designer can estimate the total delay difference after each stage for a given challenge. In the proposed scheme, we program the OTP memory such that $OTP1 = 1$, $OTP2 = sign(\Delta_{f+d})$. Thus, in the user phase, the PUF is reconfigured into a noisy feed-forward PUF, with the path selection bit of the *(f+d)*-th stage equals to $sign(r_f) \odot sign(\Delta_{f+d})$.

Algorithm 4 Delay-compensation PUF Configuration

Designer Phase:
1. Default the values of all the one-time-programable (OTP) memories as 0;
2. Obtain parameters of the MUX PUF, Δ_i;
3. Select reliable CRPs;
4. Program OTP memory $OTP1 = 1$, $OTP2 = sign(\Delta_{f+d})$, accordingly;

User Phase:
1. Reliable CRPs are used for authentication.

After reconfiguration, since the delay difference of each stage is unchanged, the designer can still use these parameters but with the updated model to estimate the overall total delay difference for a given challenge. We will only use the challenges which generate the magnitude of total delay difference that is greater than a preselected *threshold* for authentication.

5.3.2 Experimental Results

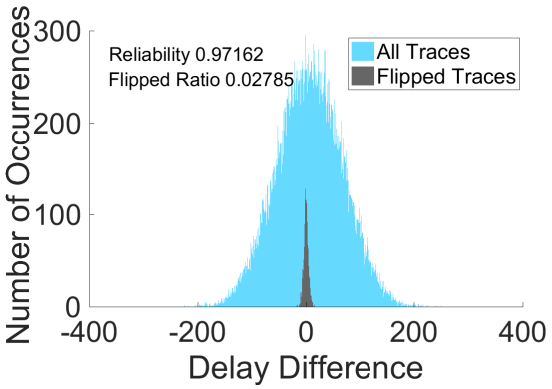
Fig. 5.5(a) and Fig. 5.5(b) show the delay difference distribution in the designer phase and the user phase of a reconfigurable delay-compensation PUF with 64 stages and 4 reconfigurable feed-forward paths. It can be seen that the reliability is decreased from 97.2% to 82.4% after reconfiguration. However, if we only use reliable CRPs, we can still achieve a very high reliability of 98.8% as shown in Fig. 5.5(c).

5.3.3 Attack using Random CRPs of Delay-compensation PUF

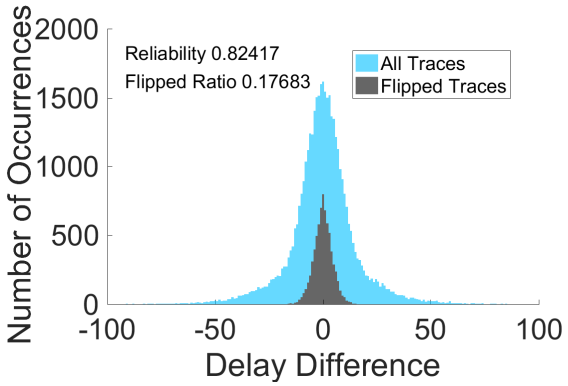
Table 5.1. Prediction Error Rate of the normal PUF and the noisy PUF under the same environmental conditions (N=64 bits)

Phase	d (Distance)	Attack	Prediction Error Rate				Reliability
			450	1000	2000	5000	
Designer	NA	Active Learning	7.4%	5.2%	4.6%	4.5%	97.2%
Attacker	4	Active Learning	27.8%	24.8%	23.3%	23.8%	98.8%
	6	Active Learning	12.6%	10.4%	10.1%	10.0%	99.0%

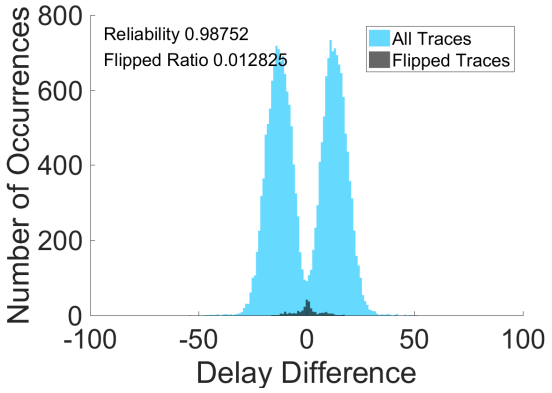
To verify the the effectiveness of the proposed reconfigurable delay-compensation PUF, both conventional passive learning and active learning techniques mentioned in Chapter 4 are employed. Fig. 5.6(a), Fig. 5.6(b), and Fig. 5.6(c) show the accuracy of modeling attack on the reliable PUF, noisy PUF compensated every 4 and 6 stages, respectively. It can be seen that the resistance to modeling attack is significantly increased by the noisy PUF, while reliable CRPs still yield high reliability. We also test the accuracy of the estimated model with only reliable CRPs. In our experiments, we use random/reliable CRPs as the training set, while other reliable CRPs are considered as the test set. Fig. 5.7(a) and Fig. 5.7(b) show the modeling attack accuracies using random CRPs and reliable CRPs, respectively. It can be seen that the accuracy is still relatively low. Under this circumstance, for example, an ECC with 5% $\sim$ 10% error-correcting capability can be integrated into the authentication. Thus, the authentic noisy PUF with the reliable CRPs can pass the authentication, while the modeled PUF would fail. Therefore, we can conclude that this PUF is secure yet functional.



(a) Delay difference distribution for the reliable PUF

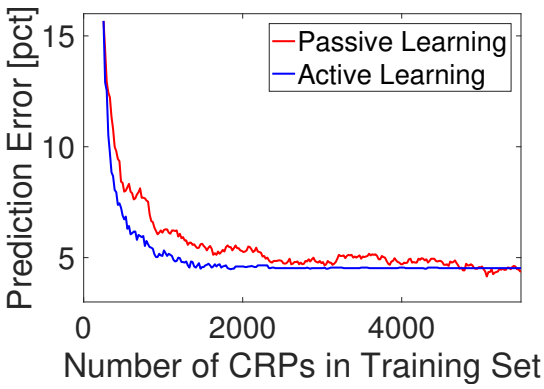


(b) Delay difference distribution for the noisy PUF ($d = 4$)

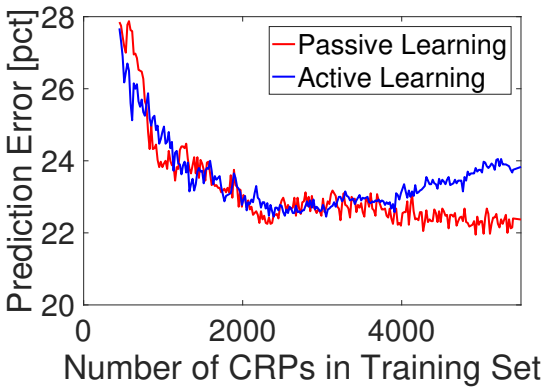


(c) Delay difference distribution for the noisy PUF ($d = 4$) with only reliable CRPs

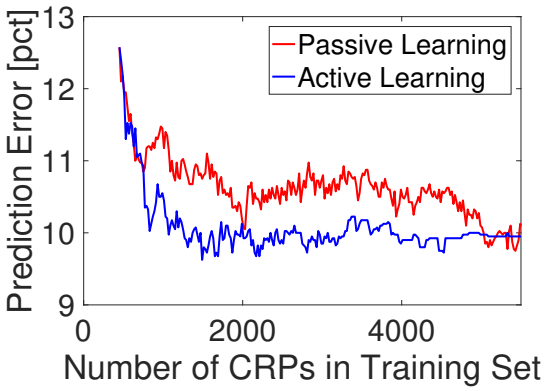
Figure 5.5. The delay distribution of proposed reconfigurable delay-compensation PUF.



(a) Normal PUF

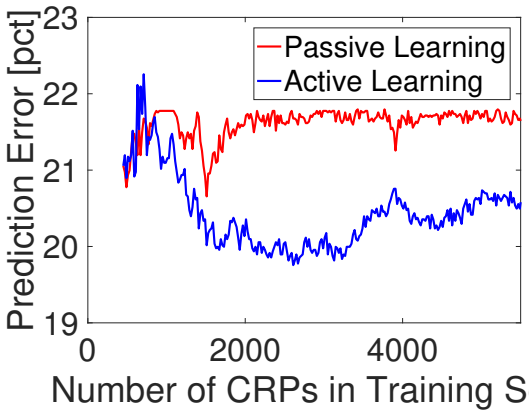


(b) Delay-compensation PUF (compensated every 4 stages)

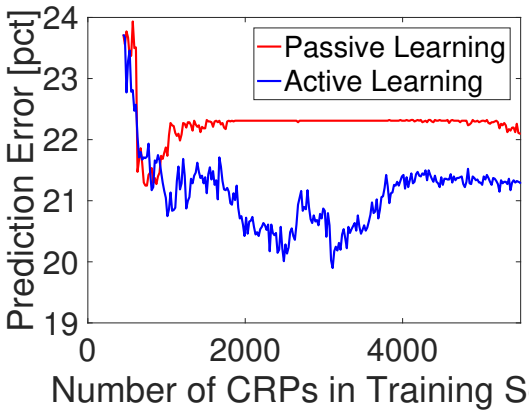


(c) Delay-compensation PUF (compensated every 6 stages)

Figure 5.6. The prediction error of the proposed delay-compensation PUF.



(a) Model trained by random CRPs and tested by selected reliable CRPs (delay compensated every 4 stages)



(b) Model trained by reliable CRPs and tested by reliable CRPs (delay compensated every 4 stages)

Figure 5.7. The prediction error of the proposed delay-compensation PUF.

Chapter 6

Conclusions and Discussion

Physical Unclonable Functions (PUFs) are promising physical security primitives. This thesis focuses on the improvement of reliability of PUF-based authentication, efficient error correction on PUF responses, using active learning on PUF modeling attack, and the design of a novel concept of secure PUF.

This thesis has presented a novel methodology to improve PUF reliability by using machine learning. The methodology has demonstrated on MUX PUF. And two algorithms are developed, i.e., *total delay difference thresholding* and *sensitive challenge grouping*, to enhance MUX PUF reliability. Experimental results have been presented to validate the effectiveness of the proposed algorithms. Then, a novel methodology is proposed to incorporate the non-uniformity of PUF response error-rates across different bits into PUF response error correction to improve the performance of PUF-based authentication. The error-rate of each individual response bit is obtained through PUF model parameter estimation by using machine learning techniques. Several methods have been proposed, i.e., *response truncating*, *response duplicating*, and *response weighting*. The performances of the proposed methods have been studied. We have shown that the *response weighting* would be able to reduce both false positive rate and false negative rate for a PUF-based authentication scheme, compared to conventional error-correcting method. We have also demonstrated several examples of integrating the estimated error-rate consideration into the design of the error-correcting block, which could enhance the overall security and reliability of PUF-based authentication as well. This thesis has also presented a novel framework for incorporating active learning techniques into hardware security field. The active learning techniques is evaluated by modeling attack the PUF un-

der different environment conditions. We demonstrate that active learning can significantly improve the learning efficiency of PUF modeling attack. The sampling strategies and detailed applications of PUF modeling attack under various environmental conditions are also discussed. The proposed framework leverage the advantages of active learning to improve the efficiency of modeling attacks, which is of great importance in developing secure PUFs.

Finally, this work has presented a novel concept of secure PUF, the noisy PUF. Based on the circuit parameters obtained from PUF modeling, we reconfigure PUFs using the OTP before configuration, which makes PUF relatively unstable and hard to model. Meanwhile, it is feasible for the designer/user to obtain several reliable CRPs by using the circuit parameters from the reliable CRPs before configuration, thus, the reliability of PUF-based authentication is still relatively high. A possible case, delay-compensation PUF is presented. The results show that delay-compensation PUF has good resistance to passive/active learning, and it still has relatively high reliability.

6.1 Future Work

Better reliability and security with low overhead are the goals of PUF design. Future work will focus on exploring new designs of the proposed noisy PUF. One possible design is to add highly non-linear blocks into the reconfigurable PUF, which will be bypassed before reconfiguration. In addition, theoretically-proved secure cryptography such as learning-with-error (LWE) can also be incorporated into the proposed noisy PUF methodology.

Bibliography

[1] B. Gassend, D. Lim, D. Clarke, M. Van Dijk, and S. Devadas, "Identification and authentication of integrated circuits," *Concurrency and Computation: Practice and Experience*, vol. 16, no. 11, pp. 1077–1098, 2004.

[2] F. Koushanfar, "Hardware metering: A survey," in *Introduction to Hardware Security and Trust*. Springer, 2012, pp. 103–122.

[3] J. B. Wendt and M. Potkonjak, "Hardware obfuscation using PUF-based logic," in *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 2014, pp. 270–277.

[4] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of ACM Conference on Computer and Communications Security*, 2002, pp. 148–160.

[5] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 670–673.

[6] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Proceedings of Cryptographic Hardware and Embedded Systems (CHES 2007)*, 2007, pp. 10–13.

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

[8] P. Koeberl, Ü. Kocabaş, and A.-R. Sadeghi, "Memristor PUFs: a new generation of memory-based physically unclonable functions," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 428–431.

[9] K. B. Frikken, M. Blanton, and M. J. Atallah, "Robust authentication using physically unclonable functions," in *Proceedings of Information Security Conference (ISC)*, 2009.

[10] M. Rostami, M. Majzoobi, F. Koushanfar, D. S. Wallach, and S. Devadas, "Robust and reverse-engineering resilient puf authentication and key-exchange by substring matching," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 1, pp. 37–49, 2014.

[11] M.-D. Yu, D. M'Raïhi, I. Verbauwhede, and S. Devadas, "A noise bifurcation architecture for linear additive physical functions," in *Proceedings of IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2014, pp. 124–129.

[12] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching," in *Proceedings of IEEE Symposium on Security and Privacy Workshops (SPW)*, 2012, pp. 33–44.

[13] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th annual Design Automation Conference*, 2007, pp. 9–14.

[14] M.-D. Yu and S. Devadas, "Secure and robust error correction for physical unclonable functions," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 48–65, 2010.

[15] W. Yan, F. Tehranipoor, and J. A. Chandy, "PUF-based fuzzy authentication without error correcting codes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016.

[16] Y. Lao and K. K. Parhi, "Statistical analysis of MUX-based physical unclonable functions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 5, pp. 649–662, 2014.

[17] R. Maes, P. Tuyls, and I. Verbauwhede, "A soft decision helper data algorithm for sram pufs," in *2009 IEEE International Symposium on Information Theory*, 2009, pp. 2101–2105.

[18] Z. Paral and S. Devadas, "Reliable and efficient puf-based key generation using pattern matching," in *Proceedings of International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2011, pp. 128–133.

[19] L. Lin, S. Srivathsa, D. K. Krishnappa, P. Shabadi, and W. Burleson, "Design and validation of arbiter-based pufs for sub-45-nm low-power security applications," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1394–1403, 2012.

[20] C. Zhou, S. Satapathy, Y. Lao, K. K. Parhi, and C. H. Kim, "Soft response generation and thresholding strategies for linear and feed-forward MUX PUFs," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, 2016, pp. 124–129.

[21] U. Rührmair, J. Sölter, F. Sehnke, X. 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*, vol. 8, no. 11, pp. 1876–1891, 2013.

[22] Y. Wen and Y. Lao, "Enhancing PUF reliability by machine learning," in *Procedings of 2017 International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.

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

[24] X. Xu, W. Burleson, and D. E. Holcomb, "Using statistical models to improve the reliability of delay-based pufs," in *Proceedings of 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2016, pp. 547–552.

[25] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper data algorithms for PUF-based key generation: Overview and analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 889–902, 2015.

[26] G. T. Becker and R. Kumar, "Active and passive side-channel attacks on delay based PUF designs," *IACR Cryptology ePrint Archive*, vol. 2014, p. 287, 2014.

[27] M. Majzoobi, F. Koushanfar, and S. Devadas, "FPGA PUF using programmable delay lines," in *Proceedings of 2010 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2010, pp. 1–6.

[28] C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls, "Efficient helper data key extractor on FPGAs." Springer, 2008, pp. 181–197.

[29] Z. Paral and S. Devadas, "Reliable and efficient PUF-based key generation using pattern matching," in *Proceedings of 4th IEEE International Conference on Hardware-Oriented Security and Trust (HOST)*, 2011.

[30] G. Selimis, M. Konijnenburg, M. Ashouei, J. Huisken, H. de Groot, V. van der Leest, G.-J. Schrijen, M. van Hulst, and P. Tuyls, "Evaluation of 90nm 6t-SRAM as physical unclonable function for secure key generation in wireless sensor nodes," in *Proceedings of 2011 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2011, pp. 567–570.

[31] M. Hiller, M.-D. Yu, and G. Sigl, "Cherry-picking reliable PUF bits with differential sequence coding," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 9, pp. 2065–2076, 2016.

[32] M. Hofer and C. Boehm, "An alternative to error correction for SRAM-like PUFs," *Cryptographic Hardware and Embedded Systems, CHES 2010*, pp. 335–350, 2010.

[33] Y. Wen and Y. Lao, "Efficient puf error correction through response weighting," in *61th IEEE International Midwest Symposium on Circuits and Systems, MWSCAS 2017*. IEEE, 2017.

[34] Y. Gao, H. Ma, G. Li, S. Zeitouni, S. F. Al-Sarawi, D. Abbott, A.-R. Sadeghi, and D. C. Ranasinghe, "Exploiting PUF models for error free response generation," *arXiv preprint arXiv:1701.08241*, 2017.

[35] Y.-M. Lin, H.-C. Chang, and C.-Y. Lee, "Improved high code-rate soft BCH decoder architectures with one extra error compensation," *IEEE Transactions on very large scale integration (VLSI) systems*, vol. 21, no. 11, pp. 2160–2164, 2013.

[36] F. Glover and M. Laguna, *Tabu Search*. Springer, 2013.

[37] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing bayesian network structure learning algorithm," *Machine learning*, vol. 65, no. 1, pp. 31–78, 2006.

[38] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi *et al.*, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[39] R. S. Chakraborty and S. Bhunia, "Hardware protection and authentication through netlist level obfuscation," in *Proceedings of the 2008 International Conference on Computer-Aided Design*. IEEE Press, 2008, pp. 674–677.

[40] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 83–89.

[41] K. Xiao, D. Forte, and M. M. Tehranipoor, "Efficient and secure split manufacturing via obfuscated built-in self-authentication," in *Procedings of 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2015, pp. 14–19.

[42] J. J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 1259–1264.

[43] T. Güneysu and A. Moradi, "Generic side-channel countermeasures for reconfigurable devices," in *Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems CHES*. Springer, 2011, pp. 33–48.

[44] A. Cui, Y. Luo, and C.-H. Chang, "Static and dynamic obfuscations of scan data against scan-based side-channel attacks," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 363–376, 2017.

[45] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE design & Test of Computers*, vol. 27, no. 1, 2010.

[46] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *Procedings of International Workshop on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2008, pp. 51–57.

[47] H. Eldib, C. Wang, M. Taha, and P. Schaumont, "Quantitative masking strength: quantifying the power side-channel resistance of software code," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1558–1568, 2015.

[48] U. Guin, M. Tehranipoor, D. DiMase, and M. Megrdichian, "Counterfeit ic detection and challenges ahead," *ACM Special Interest Group on Design Automation (SIGDA)*, vol. 43, no. 3, pp. 1–5, 2013.

[49] N. F. Ghalaty, B. Yuce, and P. Schaumont, "Analyzing the efficiency of biased-fault based attacks," *IEEE Embedded Systems Letters*, vol. 8, no. 2, pp. 33–36, 2016.

[50] J. Delvaux and I. Verbauwhede, "Fault injection modeling attacks on 65 nm arbiter and RO sum PUFs via environmental changes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 6, pp. 1701–1713, 2014.

[51] D. Nedospasov, J.-P. Seifert, C. Helfmeier, and C. Boit, "Invasive PUF analysis," in *Procedings of Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. IEEE, 2013, pp. 30–38.

[52] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[53] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of machine learning research*, vol. 2, no. Nov, pp. 45–66, 2001.

[54] M. Ghassemi, A. D. Sarwate, and R. N. Wright, "Differentially private online active learning with applications to anomaly detection," in *Proceedings of the 2016 Workshop on Artificial Intelligence and Security*. ACM, 2016, pp. 117–128.

[55] Y. Li and L. Guo, "An active learning based TCM-KNN algorithm for supervised network intrusion detection," *Computers & security*, vol. 26, no. 7, pp. 459–467, 2007.

[56] S. Tong, *Active learning: theory and applications*. Stanford University, 2001.

[57] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin, and D. Z. Pan, "Provably secure camouflaging strategy for IC protection," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017.

[58] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Appsat: Approximately de-obfuscating integrated circuits," in *Procedings of 2017 International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2017, pp. 95–100.

[59] J. Baldridge and M. Osborne, "Active learning and the total cost of annotation." in *Procedings of Empirical Methods in Natural Language Processing (EMNLP)*, 2004, pp. 9–16.

[60] W. Liu and T. Wang, "Online active multi-field learning for efficient email spam filtering," *Knowledge and Information Systems*, vol. 33, no. 1, pp. 117–136, 2012.

[61] D. Sculley, M. E. Otey, M. Pohl, B. Spitznagel, J. Hainsworth, and Y. Zhou, "Detecting adversarial advertisements in the wild," in *Proceedings of the 17th International Conference on Knowledge Discovery and Data Mining*. ACM, 2011, pp. 274–282.

[62] P. Melville and R. J. Mooney, "Diverse ensembles for active learning," in *Proceedings of the 21th International Conference on Machine Learning*. ACM, 2004, p. 74.

[63] B. Settles, "Active learning literature survey," *University of Wisconsin, Madison*, vol. 52, no. 55-66, p. 11, 2010.

[64] D. Tuia, F. Ratle, F. Pacifici, M. F. Kanevski, and W. J. Emery, "Active learning methods for remote sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 7, pp. 2218–2232, 2009.

[65] S. Chakraborty, V. Balasubramanian, and S. Panchanathan, "Generalized batch mode active learning for face-based biometric recognition," *Pattern Recognition*, vol. 46, no. 2, pp. 497–508, 2013.

[66] Y. Wen and Y. Lao, "PUF modeling attack using active learning," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5.

[67] Y. Lao and K. K. Parhi, "Reconfigurable architectures for silicon physical unclonable functions," in *Proceedings of IEEE International Conference on Electro Information Technology*, 2011, pp. 1–7.

[68] M.-D. M. Yu, D. MRaihi, R. Sowell, and S. Devadas, "Lightweight and secure PUF key storage using limits of machine learning," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2011, pp. 358–373.

[69] Q. Ma, C. Gu, N. Hanley, C. Wang, W. Liu, and M. O'Neill, "A machine learning attack resistant multi-PUF design on FPGA," in *Design Automation Conference (ASP-DAC), 2018 23rd Asia and South Pacific*. IEEE, 2018, pp. 97–104.

[70] T. Machida, D. Yamamoto, M. Iwamoto, and K. Sakiyama, "Implementation of double arbiter PUF and its performance evaluation on FPGA," in *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*. IEEE, 2015, pp. 6–7.

[71] R. Kumar and W. Burleson, "On design of a highly secure PUF based on non-linear current mirrors," in *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 38–43.

[72] M. M. Kalyanaraman, "Highly secure strong PUF based on nonlinearity of MOSFET subthreshold operation," 2012.

[73] Y. Pang, H. Wu, B. Gao, D. Wu, A. Chen, and H. Qian, "A novel PUF against machine learning attack: Implementation on a 16 Mb RRAM chip," in *Electron Devices Meeting (IEDM), 2017 IEEE International*. IEEE, 2017, pp. 12–2.

[74] Y. Tanaka, S. Bian, M. Hiromoto, and T. Sato, "Coin flipping puf: A novel PUF with improved resistance against machine learning attacks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2018.

[75] A. Vijayakumar, V. C. Patil, C. B. Prado, and S. Kundu, "Machine learning resistant strong PUF: Possible or a pipe dream?" in *Hardware Oriented Security and Trust (HOST), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 19–24.

[76] L. Santiago, V. C. Patil, C. B. Prado, T. A. Alves, L. A. Marzulo, F. M. França, and S. Kundu, "Realizing strong PUF from weak PUF via neural computing," in *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1–6.

[77] M.-D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede, "A lockdown technique to prevent machine learning on PUFs for lightweight authentication," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 146–159, 2016.

[78] C. Jin, C. Herder, L. Ren, P. H. Nguyen, B. Fuller, S. Devadas, and M. van Dijk, "FPGA implementation of a cryptographically-secure PUF based on learning parity with noise," *Cryptography*, vol. 1, no. 3, p. 23, 2017.

[79] C. Herder, L. Ren, M. van Dijk, M.-D. Yu, and S. Devadas, "Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 1, pp. 65–82, 2017.

[80] A. Mahmoud, U. Rührmair, M. Majzoobi, and F. Koushanfar, "Combined modeling and side channel attacks on strong PUFs." *IACR Cryptology ePrint Archive*, vol. 2013, p. 632, 2013.

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

[82] J. Delvaux and I. Verbauwhede, "Side channel modeling attacks on 65nm arbiter pufs exploiting cmos device noise," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2013, pp. 137–142.

[83] W. Yu and J. Chen, "Masked aes puf: a new PUF against hybrid SCA/MLAs," *Electronics Letters*, vol. 54, no. 10, pp. 618–620, 2018.

[84] K. Tiri and I. Verbauwhede, "A VLSI design flow for secure side-channel attack resistant ICs," in *Design, Automation and Test in Europe, 2005. Proceedings*. IEEE, 2005, pp. 58–63.

[85] Y. Lao and K. K. Parhi, "Reconfigurable architectures for silicon physical unclonable functions," in *Electro/Information Technology (EIT), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–7.

[86] E. Terzioglu, G. I. Winograd, and M. C. Afghahi, "One-time-programmable memory," Mar. 24 2009, uS Patent 7,508,694.