ADVERSARIAL SELECTION OF CHALLENGE-RESPONSE PAIRS AS A DEFENSE AGAINST STRONG PUF MODELING ATTACKS

Horácio Lima França

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Carlos Eduardo Pedreira
Charles Bezerra do Prado

Rio de Janeiro
Setembro de 2019

ADVERSARIAL SELECTION OF CHALLENGE-RESPONSE PAIRS AS A DEFENSE AGAINST STRONG PUF MODELING ATTACKS

Horácio Lima França

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Carlos Eduardo Pedreira, Ph.D.

Dr. Charles Bezerra do Prado, D.Sc.

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. Flávio Luis de Mello, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2019

França, Horácio Lima

Adversarial Selection of Challenge-Response Pairs as a defense against Strong PUF Modeling Attacks/Horácio Lima França. – Rio de Janeiro: UFRJ/COPPE, 2019.

IX, 27 p.: il.; 29, 7cm.

Orientadores: Carlos Eduardo Pedreira

Charles Bezerra do Prado

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2019.

Referências Bibliográficas: p. 24 – 27.

1. Aprendizado de máquina adversarial. 2. Segurança cibernética. 3. PUF. I. Pedreira, Carlos Eduardo *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

SELEÇÃO ADVERSARIAL DE *CHALLENGE-RESPONSE PAIRS* COMO DEFESA CONTRA ATAQUES DE MODELAGEM A *PUFS*

Horácio Lima França

Setembro/2019

Orientadores: Carlos Eduardo Pedreira
Charles Bezerra do Prado

Programa: Engenharia de Sistemas e Computação

Neste trabalho, apresentamos métodos para melhor proteger os mecanismos de autenticação incorporados em hardware conhecidos como *Physically Unclonable Functions* (PUFs). Esses mecanismos usam as características físicas únicas dos chips nos quais estão inseridos para criar um conjunto de respostas que se mostraram vulneráveis a ataques de modelagem por aprendizado de máquina. As técnicas desenvolvidas aqui são focadas no uso do Aprendizado de Máquina Adversarial para selecionar as cadeias binárias usadas para as operações de autenticação em questão, comumente conhecidas como *Challenge-Response Pairs*, para proteger os dispositivos que utilizam esses *PUFs* de terem suas credenciais de autenticação copiadas. O resultado desta pesquisa é uma série de métodos que se aplicam a diferentes cenários que reduzem a precisão de possíveis ataques de modelagem em até 19%.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ADVERSARIAL SELECTION OF CHALLENGE-RESPONSE PAIRS AS A DEFENSE AGAINST STRONG PUF MODELING ATTACKS

Horácio Lima França

September/2019

Advisors: Carlos Eduardo Pedreira
Charles Bezerra do Prado

Department: Systems Engineering and Computer Science

In this work, we present methods to further secure authentication mechanisms embedded in hardware known as Physically Unclonable Functions (PUFs). These mechanisms use the unique physical characteristics of the chips they are embedded in to create a set of responses were found to be vulnerable to Machine Learning modelling attacks. The techniques developed herein are focused on using Adversarial Machine Learning to select the binary strings used for the authentication operations in question, commonly known as Challenge-Response Pairs, in order to protect the devices using these PUFs from having their authentication credentials copied. The result of this research are a series of methods that apply to different scenarios that reduce the accuracy of possible modelling attacks in up to 19%.

Contents

List of Figures viii

List of Tables ix

1 Introduction 1

1.1 Motivation . . . 1
1.2 Objectives and Contributions . . . 2
1.3 Text Structure . . . 3

2 Background 4

2.1 Physical Unclonable Functions . . . 4
2.1.1 History . . . 4
2.1.2 Function . . . 4
2.1.3 Vulnerabilities . . . 6
2.2 Ensemble Machine Learning . . . 6
2.2.1 History . . . 6
2.2.2 Adaptive Boosting . . . 7
2.2.3 Relevance to current work . . . 7
2.3 Adversarial Machine Learning . . . 8
2.3.1 History . . . 8
2.3.2 Adversarial Machine Learning in Cybersecurity . . . 9
2.3.3 Relevance to current work . . . 10

3 Methodology 11

3.1 Attack model . . . 11
3.2 Single group with poor internal recognition . . . 12
3.2.1 Proposed Scenario and Objective . . . 12
3.2.2 Proposed Solution . . . 12
3.3 Single group with good internal recognition and poor external recognition . . . 13
3.3.1 Proposed Scenario and Objective . . . 13
3.3.2 Proposed Solution . . . 14

3.4 Multiple groups with poor external recognition 14
3.4.1 Proposed Scenario and Objective 14
3.4.2 Proposed Solution 15

4 Setup and Results **17**
4.1 Experimental Setup 17
4.1.1 System Setup 17
4.1.2 Data Setup 17
4.2 Results 18
4.2.1 Single group with poor internal recognition 18
4.2.2 Single group with good internal recognition and poor external recognition 20
4.2.3 Multiple groups with poor external recognition 21

5 Conclusion & Future Works **22**
5.1 Final Considerations 22
5.2 Future Works 23

Bibliography **24**

List of Figures

1.1 Number of IoT devices in operation (in billions) [1]. 2

2.1 Diagram showing the circuit structure of an Arbiter PUF [2]. 5
2.2 Results obtained by Vijayakumar et al. [3] in their research 8

3.1 Examples of the selection performed in a) Single group with poor internal recognition, b) Single group with good internal recognition and poor external recognition [4] 13
3.2 Selection performed in phases [4] 15

4.1 Scores for the Single Group with Poor Internal Recognition Technique. The red line represents the accuracy of the models trained with the tainted data sets (*Selected*), while the blue line represents the accuracy of a model trained with random $100,000$ CRPs (*Control*). [4] 19
4.2 Accuracy charts for the Single Group with Poor External Recognition and Good Internal Recognition Technique. The red lines represent the accuracies of the models trained with algorithmically selected data sets and the blue lines represent the accuracies of models trained with random CRPs. [4] 20

List of Tables

4.1 Results of the multiple groups with poor external recognition technique 21

Chapter 1

Introduction

In this chapter, we explore the motivation behind this dissertation and give a brief summary of what is proposed as a solution. After that, we describe the structure of the rest of the text.

1.1 Motivation

Physical Unclonable Functions (PUFs) [2] are mechanisms embedded in processors to assist in tasks such as secret key generation and authentication in devices with relatively few computational resources. They do this by using the unique characteristics of the chips they are built into to generate outputs based on inputs in the form of binary strings. The combination of input and output in these cases are called challenge-response pairs, or CRPs. The amount of CRPs a PUF can support determines what kind of PUF it is. If it can respond to several challenges it is a strong PUF and will generally be used for authentication. On the other hand, if it only has a few CRPs it is classified as a weak PUF, and will be used for secret key generation.

The amount of devices belonging to the so-called Internet of Things (IoT) has been increasing steadily [1], as can be seen in Fig.1.1. These devices tend to be lightweight and have very few resources put into security because of that. Several attacks have been performed by exploiting these weaknesses [5], so many are trying to improve these devices' security. One of the ways found to secure some of these devices was through the use of PUFs [6, 7], and any further security that could be applied to these devices would be very beneficial.

One of the ways attackers have found to bypass a strong PUF's authentication system is to perform what is called a modelling attack [8]. This attack method consists of intercepting challenges sent from a server to a PUF and their corresponding responses, storing them and then using that data to train a Machine Learning Model. That model can then used by the attacker to spoof a PUF's responses to challenges

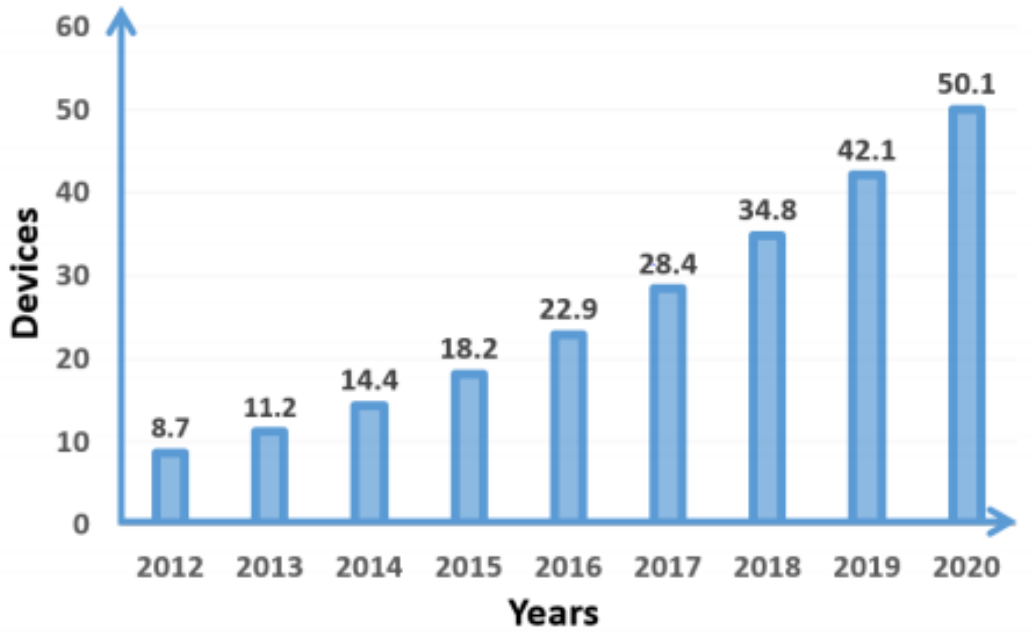

Figure 1.1: Number of IoT devices in operation (in billions) [1].

that it hasn't been explicitly exposed to in the same manner as the targeted PUF would.

1.2 Objectives and Contributions

Most research into increasing the security provided by PUFs focuses mainly on the topology of the circuit or the measurement made to generate responses [9]. The intention behind this project is to find a way to enhance a PUF's security without altering any of its structural properties. In order to follow through with this task, the general idea behind this work is to take inspiration from Adversarial Machine Learning attacks to find a way to hinder attempts to replicate a PUFs responses.

We seek to reach our goals by devising a method with which we can use Machine Learning to create datasets that aren't as vulnerable to modelling attacks as a PUF would be ordinarily. With this research, we hope to create methods to perform a selection of CRPs from a PUF that will increase its resistance to modelling attacks in different scenarios. Said scenarios, that will be described in greater detail further on, are:

- Single group with poor internal recognition
- Single group with good internal recognition and poor external recognition
- Multiple groups with poor external recognition

In these cases when we refer to Internal recognition we are discussing the accuracy an attacker will have when testing a model with the data they have and external recognition refers to the accuracy an attacker would have when attempting to spoof the targeted PUF.

If successful this project would provide a powerful tool to further secure systems using PUF technology for authentication purposes. This is significant because PUFs are generally used in light-weight and resource constrained devices such as those commonly referred to as belonging to the Internet of Things (IoT), and they have been targeted for massive attacks recently. Furthermore, our solution is implemented on the server's side, not requiring resources from the less computationally powerful IoT devices.

1.3 Text Structure

In chapter 2 we explain the relevant background to the methods we have developed. Specifically, PUFs and how they function, Adaptive Boosting (AdaBoost) and Adversarial Machine Learning. In chapter 3 we specify the attack model we have adopted and describe the different methods we have developed and scenarios we have considered to create them.

In chapter 4 we explain how we tested the techniques and the results we obtained from the experiments designed. We also explain how we devised and tested a countermeasure to one of the techniques referenced earlier. In chapter 5 we discuss our final considerations and ponder possible contributions that could be made in the future.

Chapter 2

Background

In this chapter we detail the concepts that serve to describe the basic concepts that serve as a basis for our work. The first section describes PUFs and how they function, the second focuses on Ensemble Machine Learning and, finally, in the third section we explore Adversarial Machine Learning.

2.1 Physical Unclonable Functions

2.1.1 History

The first Physical Unclonable Function (PUF) was designed by Pappu et al. [10] in 2002, known at the time as Physical One-Way Functions. They were devised with inspiration in one-way functions that researchers in the field of cryptography were already using in an algorithmic manner. Their core idea was to reproduce those one-way functions in a simple and low-cost manner.

Over time, researchers have developed several different types of PUF, using varied strategies to extract some form of unique identifying quality. Some of the first, and the type we are more concerned with, were CMOS PUFs[11]. They use the small differences present in the mass production of transistors for their operations. Among those PUFs there are those based on delays within a chip's circuitry, or Delay PUFs, such as the Arbiter PUF, the Glitch PUF and the Ring Oscillator PUF. Another type of CMOS PUF would be the Memory PUF, that observe memory cells stabilizing from an induced unstable state into either 0 or 1.

2.1.2 Function

PUFs are simple mechanisms embedded into chips used to provide low-cost security for devices with few computational resources and low processing power. They do this by exploiting the unique physical characteristics of the chips they are embedded

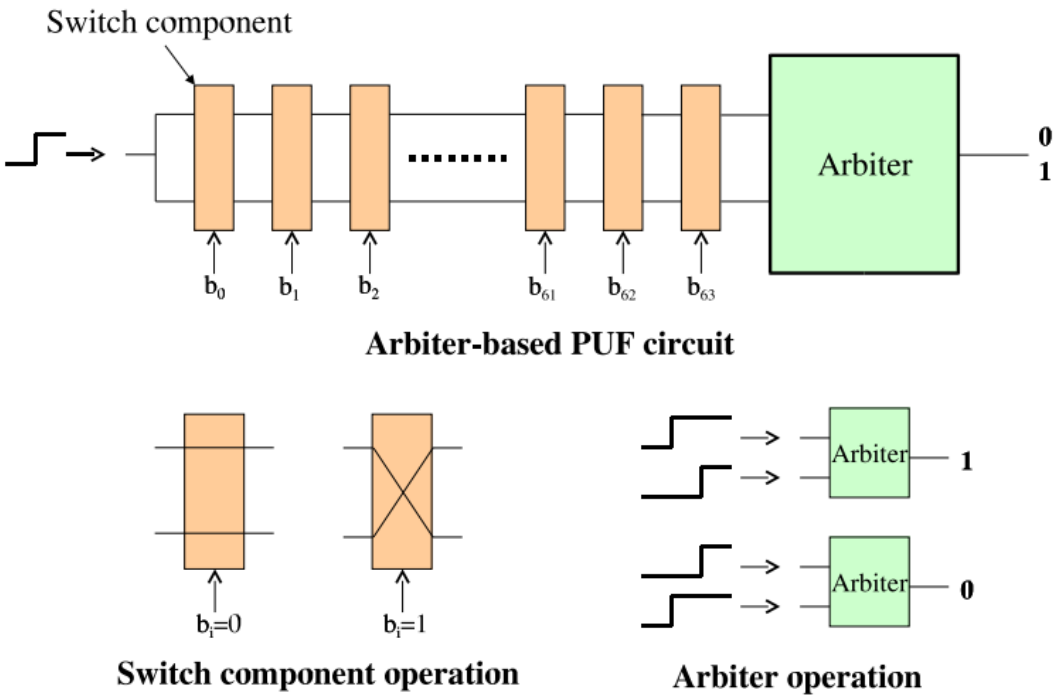

Figure 2.1: Diagram showing the circuit structure of an Arbiter PUF [2].

in to create signals that can either be used to identify a certain device or serve as a cryptographic key for it to use in conjunction with other applications. Because these signals are tied to small idiosyncrasies of specific components they are incredibly hard to reproduce, as the word "Unclonable" suggests.

PUFs are commonly separated into two categories related to how many CRPs they support: Strong or Weak [12]. Weak PUFs have relatively few CRPs, but are still resistant enough to environmental condition changes to have very stable outputs. They are used to generate cryptographic keys because those can be kept secret and there doesn't need to be enough CRPs that an attacker wouldn't be able to ascertain what the response for each challenge would be from simple observation. Strong PUFs, in addition to maintaining a certain resilience to environmental condition change, support many CRPs. The amount is so great that the PUF can be used for authentication purposes, with the server sending challenges to the PUF and checking to see if it returns the correct response, without repeating CRPs.

A relevant example of a strong, but quite simple, PUF is the Arbiter PUF [2]. The PUF is composed of a series of switches that have two paths running through them and an arbiter at the end. It takes a challenge in the form of a string of bits with a length equal to the amount of switches it has. Each bit of the challenge is then used to control the the switches, by indicating if it should maintain the paths as they are or switch them, as can be seen in Fig.2.1. The response is given by the arbiter at the end of these paths by evaluating witch of the currents arrived

at the arbiter first. The switches in the path interfere with the time it takes to arrive at the arbiter because of very small differences in the material that composes them can create smaller or greater delays depending on how it has been set by its corresponding input bit. What makes each arbiter PUF useful as an authentication and identification device is the fact that those small differences in the delays of the switches tend to be unique to each PUF due to the small imprecisions of the fabrication process.

2.1.3 Vulnerabilities

The problem with the approach used to create the Arbiter PUF was that although a human observer would, most likely, never be able to reliably predict the PUF's responses, but a Machine Learning model would. The linear structure of the Arbiter PUF is specially susceptible to this kind of attack. This is because the Arbiter PUFs design puts a lot of influence in the last few switches, given that, depending on how long their delays are, they can determine what the output will be regardless of what was done in the earlier switches of the PUF.

The attacks performed by exploiting vulnerability described earlier consist of a variant of man-in-the-middle attack. In essence, an attacker could listen in on the communication between a server and a device with a PUF and record the CRPs to train a Machine Learning model. That model can then be used to predict the response the PUF accurately enough to seem to be the device with the PUF to the server.

In order to better protect the pattern inherent in the PUF, researchers would try to alter the PUF's structure. One of the solutions found was to use several Arbiter PUFs in parallel and connecting the arbiter responses of each of them to an XOR component, this is known as an XOR Arbiter PUF [13]. Another solution uses the delays in identical Ring Oscillators in order to identify a PUF [14].

2.2 Ensemble Machine Learning

2.2.1 History

Ensemble Machine Learning techniques use multiple machine learning models in conjunction to produce results that are better than a single model would provide. The way the several machine learning models' results are aggregated varies from selecting one response randomly, to calculating the average of said models' results (used mainly for regression) or using the response of each model as a vote and using the most voted response (used in classification).

The potential applications of Ensemble Machine Learning models are as numerous as there are for individual Machine Learning models. It has been used to classify protein in food [15], speech pitch prediction depending on someone's accent [16] and even diagnosing diabetes [17]. In an example slightly more relevant to this work, there is even a precedent for applications in cybersecurity [18].

2.2.2 Adaptive Boosting

The specific Ensemble Machine Learning technique used in this project was Adaptive Boosting (referred to from now on as AdaBoost). AdaBoost creates several estimators iteratively and each time a new one is trained an array of weights associated with each entry in the training data set is updated, increasing the weights corresponding to the entries that weren't classified correctly and decreasing the weights of the ones corresponding to entries that were. The most common weak estimators used by AdaBoost are Decision Trees.

The part of the training algorithm for AdaBoost [19] that adjusts the weights of individual samples takes place as follows: First, an array with p positions, where p is the amount of entries in the training set, each of these positions filled with $1/p$. Following that, a first weak model is trained, and a new array of weights is created to reflect whether or not a sample was correctly classified or not according to the following equation: $w_{it} = w_{it-1} exp(\alpha_t y_i F(x_i))/Z_t$, where α_t is the weight of the classifier t and equals $(log((1-\epsilon_t)/\epsilon_t))/2$ and $w_i t$ is the weight of sample x_i, the class of which is y_i, at iteration t, ϵ_t is the error of weak estimator F and Z_t is a normalizing constant that insures that $\sum_{i=1}^{p} w_{it} = 1$. The next estimator will then train itself making adjustments to its structure proportionally to the weights of each sample.

The inference step of AdaBoost uses the results of all its weak estimators and averages them, using the result (for regression) or counts them as votes (for classification). This approach is believed to mitigate overfitting, which improves accuracy, or at least keeps it from decreasing. The weights of each individual sample also provide insight into how easily they are classified given the current dataset.

2.2.3 Relevance to current work

In a study by Vijayakumar et al. [3], it was found that among several types of machine learning models Gradient Boosting was the most effective in cracking strong PUFs. In their work they compared Bagging, Support Vector Machines, Linear Regression and Gradient Boosting, a variant of Adaptive Boosting, and found that the latter out-performed all the others. In some of the simpler cases, as shown in fig.2.2, the difference can be of 15% accuracy between Gradient Boosting and the

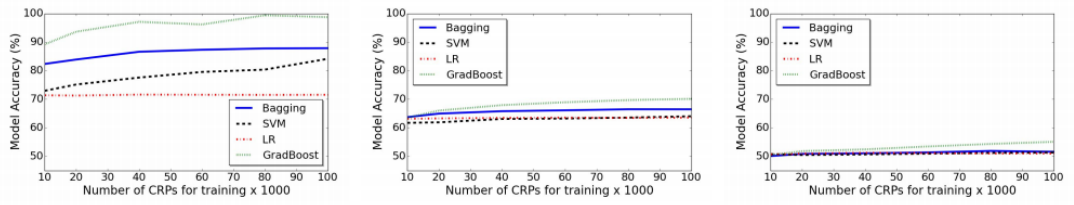

Figure 2.2: Results obtained by Vijayakumar et al. [3] in their research

next best model's performance.

Adaptive Boosting also makes an interesting piece of information explicit in the form of the weights associated with each entry in the training data. Since these weights increase when the corresponding sample is misclassified and and decreases when they are correctly classified, it is fair to make the assumption that the higher a sample's weight is the harder it is to classify. Armed with this knowledge it is theoretically possible to perform an Adversarial Machine Learning countermeasure to the PUF modelling attacks detailed earlier.

2.3 Adversarial Machine Learning

2.3.1 History

Over the past few years, machine learning has been adopted by an increasing number of developers to implement solutions for a very wide variety of fields. Among those fields, security posed a new challenge [20]: since a key aspect of security research is discovering breaches in supposedly secure technology, some researchers decided to study ways to deliberately cause machine learning models (or at least the ones used for intrusion detection) to fail. This approach started to be applied to any situation in which a machine learning model can be affected by two or more players with different goals, effectively taking what would have been a simple optimization problem and transforming it into an attempt to reach a Nash Equilibrium. A Nash Equilibrium [21] is an term from game theory that refers to a situation in which competing agents make decisions considering the decisions of opposing agents instead of simply what will benefit themselves the most.

Adversarial Machine learning has been used on more than one occasion to cause Deep Neural Networks trained to recognize objects in images to mislabel them by applying algorithmically generated filters to those images [22]. There have been projects that fool intrusion detection systems by masking malicious communication as regular traffic and by finding a way to train the model used by said system to identify the intent of the traffic in such a way that it can't be correctly classified [23].

The concept also inspired the creation of Generative Adversarial Networks (GANs) [24]. This system uses two neural networks, one for classification and another for generation. The classification network is initially trained to identify examples of a certain class while the generation network to create examples that would be considered of the class the classification network was trained to identify.

2.3.2 Adversarial Machine Learning in Cybersecurity

As was stated before, much of the work concerning Adversarial Machine Learning came from security researchers. In some cases, they were attempting to bypass intrusion detection systems [23, 25]. In others, researchers tried to hide malicious software form detection [26]. The approaches varied so much that some form of classification was required.

Taxonomy of attacks

Attacks on machine learning models can take various forms and have different objectives. In order to organize them a taxonomy has been developed by Biggio et al. [27]. The first aspect observed is the influence of the attack. In this case, the attack can be Causative if it creates a vulnerability within the model in order to be exploited later, for example, by tainting the data that is used to train the model. Alternatively, the attack is Exploratory if it finds vulnerabilities already present in the model to exploit.

Another aspect classified by this taxonomy is the type of security violation the attack will cause. This categorization is meant for attacks targeting a model that controls access to a system. An attack can cause an Integrity violation if its goal is to pass off malicious samples as legitimate ones. Meanwhile, an attack that simply tries to cause the model to misclassify all samples is trying to cause an Availability violation.

The last category listed in this taxonomy is the specificity of the attack, in other words, whether the attack is trying to affect a specific class or not. An attack can be Targeted if it is attempting to misclassify samples of a specific class or classes. If there is no limitation on what classes the samples to be misclassified are from, then the attack is Indiscriminate.

Beyond these taxonomic classifications, there have been other, more specific classes to describe the most common types of attack. For example, one of the most common types is the Evasion attack [28] is one that tries to hide something malicious, such as malware or spam emails, as legitimate. Another fairly common type of attack is the Poisoning attack [23], in which the attacker exploits the need of a system with a machine learning model to retrain itself to feed the model with

samples, generally outliers, that would lead it to misclassify other samples in the future.

2.3.3 Relevance to current work

The technique proposed in this dissertation is difficult to classify using this taxonomy given that it is not an attack, strictly speaking. If we were to attempt to do so, the most accurate would be Causative, because in all the scenarios to be presented we cause a machine learning model to misclassify samples by interfering with the training data for a machine learning model, and Indiscriminate, because we aim to create misclassification with no regard to the classes themselves since they hold no inherent semantic value.

The intended goal of this project is to use Adversarial Machine learning to select a subset of CRPs from a PUF that would not expose its internal patterns enough that a Machine Learning model would not be able to have a high enough accuracy to replicate said PUF's responses. In order to perform such a selection, we propose using the weights generated by the training algorithm of an Adaptive Boosting model to gauge how easily each CRP is classified within the given data set. In the following sections we will provide explanations for how we used said weights to select CRPs for different scenarios in order to increase PUF security in different situations.

To the extent of our knowledge, the only other project with a similar approach was done by Wang et al. [29]. Their solution involves finding ways to signal that a certain CRP is being transmitted with the expectation that the response would be inverted as a way to poison a potential attacker's machine learning model. This project differs by not altering the data being transmitted in any way.

Chapter 3

Methodology

In this chapter we describe what the attack model being considered is and then expound the techniques developed to select CRPs in order to impede a Machine Learning modelling attack in three different scenarios.

3.1 Attack model

For a complete understanding of how a modelling attack is performed, first one must understand how CRPs are collected and stored. This process occurs during production, taking a chip with a PUF, probing that PUF with a set number of challenges and finally recording the responses they generate. After that, the recorded CRPs are stored in a server to be used when the device with the corresponding PUF is in operation.

The modelling attacks referred to previously use a man-in-the-middle [30] methodology, intercepting the challenges sent by the server and their respective responses sent from the PUF. Those CRPs are then stored and when the attacker has accumulated what they consider to be enough to train a Machine Learning model. Once said model is trained it can be used to respond as if it were the PUF itself.

For the purpose of the creation of our methods and experiments, we assume an attacker can monitor communications between the server and the PUF for any amount of time necessary to acquire the required quantity of CRPs. We also do not take into consideration the reliability of the CRPs in question and assume they are accurate, meaning, we do not evaluate the possible changes that may have occurred over time to said CRPs, be it by changes in the environmental conditions surrounding it or by degradation due to the passage of time.The issue of reliability may be approached in further research.

3.2 Single group with poor internal recognition

3.2.1 Proposed Scenario and Objective

The first scenario imagined, which is the most straightforward, is one where the owner of the PUF simply doesn't want to concern themselves with the possibility of an attacker to perform a machine learning modelling attack on it. This approach is mostly passive, as it doesn't rely on any kind of automated reaction by the server. This would be applied in cases in which an owner of the PUF doesn't want to concern themselves with when an attacker may attempt something, and just allow the server and PUF to run without any kind of intervention.

The objective in this case is to select a set of CRPs that has what we refer to as Poor Internal Recognition. Internal Recognition, in this context, refers to the accuracy of a Machine Learning model, after being trained with a specially selected set of CRPs, when confronted with samples from the same selection of CRPs. The intended result of this is that the attacker will not be able to train a model capable of "cracking" the target PUF with the CRPs available.

3.2.2 Proposed Solution

The selection technique for this scenario (demonstrated in Fig.3.1(a)) is rather simple:

- Use all available CRPs belonging to a PUF to train an Adaptive Boosting model.
- Record the weights attributed to each of the samples in the training data for the last iteration of training, when the final weak discriminator is trained.
- Finally, use the CRPs associated to the highest weights to authenticate the target PUF.

The basic assumption behind the development behind this technique is that the samples with the highest weights are the hardest to classify (within the given training data set). This assumption comes from the fact, as stated earlier, the weights increase as new discriminators misclassify the associated sample. There is, however, a possibility that whatever noise caused these samples to be misclassified could have it's own pattern. This pattern in the noise could improve accuracy of the trained ML model, so the selected subset should be large enough to introduce a large variety of CRPs and counterbalance that pattern.

The reason this technique, if functional, fulfils the requirements of the proposed scenario is that the attacker will not, presumably, be able to achieve a Machine

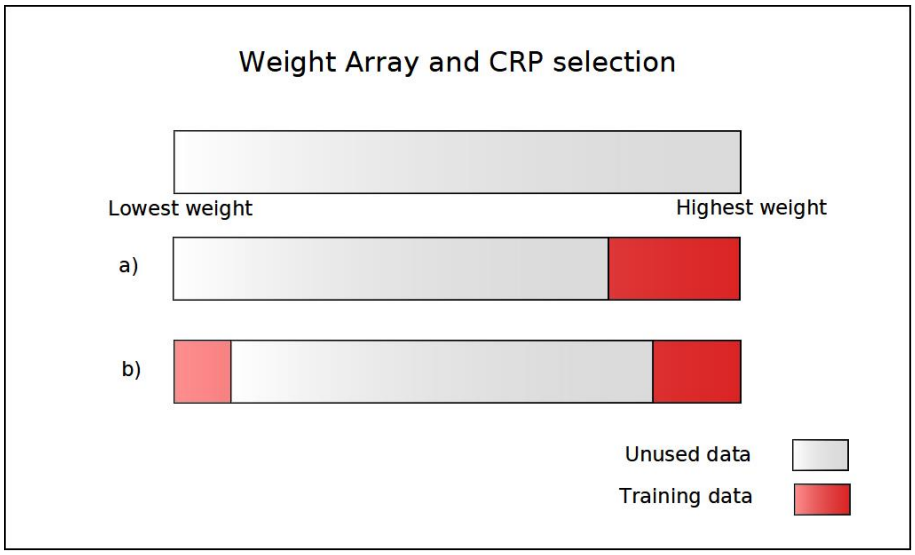

Figure 3.1: Examples of the selection performed in a) Single group with poor internal recognition, b) Single group with good internal recognition and poor external recognition [4]

Learning model that is able to crack the PUF. This is because we can assume that the attacker will not have access to CRPs outside of the selected set and the model would have diminished accuracy as a result. All things considered, this technique could impose a serious limitation on the amount of CRPs so other techniques were developed.

3.3 Single group with good internal recognition and poor external recognition

3.3.1 Proposed Scenario and Objective

In this scenario, we assume the owner of the PUF has access to an intrusion detection mechanism and can react in case an attacker is perceived to be "listening in" on the communication between a PUF and server. In this case, it would be more interesting to only apply counter-measures to an attack as it's happening. The imagined mechanism would use a special selected subset to sabotage an attacker's Machine Learning model when the presence of one is detected.

The technique's goal is to create a subset of CRPs that, if used to train a Machine Learning model, would yield seemingly great accuracy when tested with samples from the subset (good internal recognition) but display poor accuracy when tested with samples from outside the subset (poor external recognition). Such a subset would allow the attacker to train and test a Machine Learning model that is seemingly capable of cracking a PUF but in reality would fail to be sufficiently

accurate to do so. This subset can be considerably smaller than the one in the previous scenario, since it is only used when the attacker is observing the communication between the PUF and the server.

3.3.2 Proposed Solution

As with the previous technique, we begin by ordering the CRPs by their corresponding weights. After that, we take some of the CRPs highest weights and, unlike the former solution, some of the CRPs with the lowest weights to form the desired subset. When a potential attacker is detected spying on the communications between the server and PUF, the selected subset is used for authentication. A visual demonstration can be seen in Fig.3.1(b).

We expect this technique to create a subset that, when used to train a Machine Learning model, said model will have good results when testing it but not be able to crack the PUF when other CRPs are used. We assume this result will occur because while the CRPs with low weights will provide a very clear pattern for a modelling attack to seem accurate, the CRPs with high weights will make the ML model diverge from the real pattern found in the PUF's unique structure. There is cause to believe that the high-weight CRPs won't lower the accuracy of the internal recognition because there is a smaller group of samples and the pattern within the noise that caused them to have higher weights will be more easily classified.

We devised this technique for this scenario because we expect that using the selected subset to train a Machine Learning model will not result in a mechanism that can crack a PUF, even though it would appear to when testing with data from said subset. While this technique would allow the owner of a PUF to use a large amount of CRPs and having the added layer of security provided by adversarial selection, it does count on the attacker using a set number of CRPs to train and test their modelling attack. In order to remedy this, we sought after a method that could replicate the effects of this technique while also being extendable over time, which lead us to the creation of the following technique.

3.4 Multiple groups with poor external recognition

3.4.1 Proposed Scenario and Objective

The previous technique is limited not only by the amount of time it can deceive an attacker but also by counting on the said attacker using a set amount of CRPs to train their model. To remedy this, the method proposed in this section applies

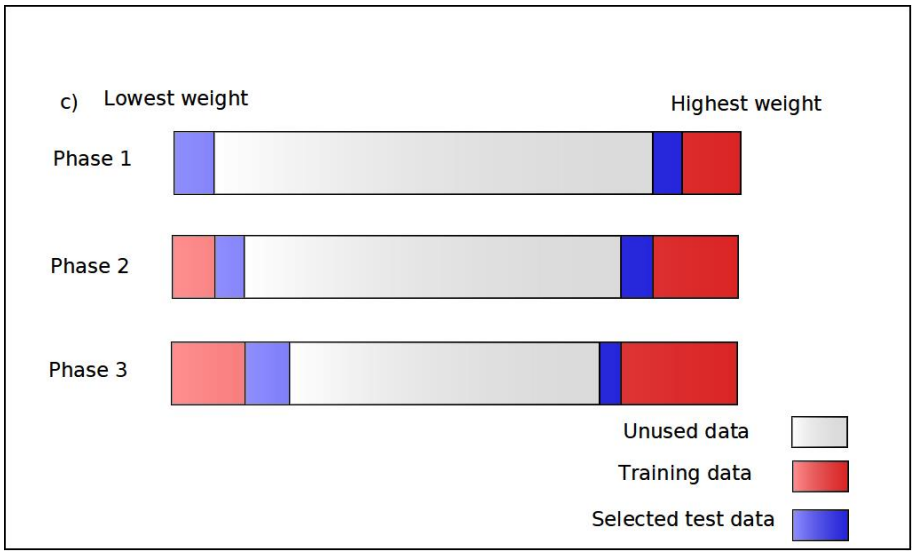

Figure 3.2: Selection performed in phases [4]

to the same situation described in the previous one: an intrusion detection system would be able to warn the PUF owner of the presence of an attacker observing the communication between a server and a PUF. This new method, however, should be able to deceive an attacker for a longer amount of time.

The technique presented here should be able to maintain the attacker in the desired state of confusion continuously by using several selected subsets instead of only one. The challenge, in this case, is to select groups of CRPs that would be released for use one after the other in such a manner that when an attacker uses all released groups to train a Machine Learning model, said model will yield results that suggest it would crack a PUF when tested with samples from the released groups but poor results when tested with the CRP subset released following that. The models trained with these cascading subsets of CRPs would also have to demonstrate poor accuracy when tested against a random selection of CRPs, since there would be, necessarily, a finite amount of said groups.

3.4.2 Proposed Solution

In order to perform the selection of subsets we use a similar approach to the previous one, taking samples from a combination of those associated to the highest and lowest weights found when training an AdaBoost model with the available CRPs. The difference in this case is that the selection is performed in phases, with the proportion of high-weight and low-weight CRPs being adjusted for each one of those phases individually. The optimal proportions for each phase were found empirically through experimentation and the amount of CRPs in each subset is the same.

As an example of the procedure we can observe Fig.3.2. In it the CRPs marked

in red are those we assume the attacker will use as training data and those marked in blue are the ones they would have to correctly classify in order to crack the targeted PUF. In the first phase all of the data available for training is from the highest weight CRPs while the CRPs being transmitted during their attempted attack is a combination of CRPs from the high and low weights. In the next step, we consider that the attacker has recorded the CRPs marked as blue in Phase 1 and is now available for training, and therefore marked in red in Phase 2, while a new batch of CRPs is going to be marked as blue. This new batch is also a combination of high and low weight CRPs, that will be incorporated into the training data in phase 3.

The intention behind this approach is to use the essence of the previous technique, the combination of CRPs with high and low weights associated to them, and parcel the effect in smaller groups. In this case there is the added complication of not only selecting one subset that has good internal recognition and poor external recognition in general, but it is also necessary to take into account that the external subset will also be selected and, presumably, will be incorporated into the internal subset. In order to accommodate this particular characteristic of the issue at hand, the proportion of high-low weight CRPs in each phase tend to skew heavily to one end in one phase, and to the other in the next. This bias is decreased over time, tending towards a balance in high and low weight CRPs.

This technique was applied to this scenario because it aims to provide the protection of the previous method for a longer amount of time. It also isn't as vulnerable during its functioning, because the phases can be smaller than the separate group in the previous scenario, making it less likely that the attacker will attempt an attack during a period where their model would be able to perform a successful spoof. If functional this technique should prove to be a significant improvement over the previous one.

Chapter 4

Setup and Results

In this chapter we go through the data and environment used to test the techniques described in the previous chapter and then present the results of those tests.

4.1 Experimental Setup

4.1.1 System Setup

The computer used for the experiments used an Intel Core i5-5200U 2.20GHz processor, and 8 Gigabytes of memory. The operating system used was a pre-installed version of 64-bit Windows 10. Said computer was used because it was owned by the author and no special machine was needed for these experiments.

All experiments were scripted in Python 3, version 3.6 specifically. The main libraries used were scikit-learn, numpy and matplotlib. Scikit-learn was used specifically because it has many Machine Learning models already prepared and it is open-source, allowing for the necessary alterations for this project.

The Machine Learning model used in these experiments was an Adaptive Boosting classification model with 200 iterations. The weak classifiers used by the AdaBoost model are Decision Trees with a maximum depth of 4. Decision trees were used in this instance because their structure is particularly adequate to replicate the inner-workings of Arbiter PUFs.

4.1.2 Data Setup

For the following experiments, we generated 10 different datasets and performed each experiment on each of them and the results reported are the average of those results. Each dataset contained 1,000,000 CRPs used for training and from which subsets are selected and another 100,000 CRPs used for testing for external results. The training data set is as large as it is because even though Arbiter PUFs are relatively

easy to model, there is still a need for copious amounts of CRPs to perform that modelling.

In order to better represent the data contained in the CRPs we use a filter that emphasises the relation between the bits that compose a challenge, called a Parity filter. This filter is applied by converting every 0 to a -1 and then, starting from the last bit, redefine the number at that position by multiplying the original number by all of the numbers after it. This filter helps express the dependence the arbiter PUF response has on the linear structure of said PUF.

In the following experiments, whenever a test is made using k-fold [31] cross-validation, k = 10. This number allows for a great deal of the samples to be used while not compromising the feasibility of the experiment. Whenever we refer to control tests in the following sections we are describing results obtained by taking a completely random selection of samples from the training data, with no biases whatsoever.

4.2 Results

4.2.1 Single group with poor internal recognition

In this experiment we used the k-fold method to evaluate the accuracy of two models, one trained with a control training set and another with a data set selected using the method proposed for the first scenario described in the previous chapter. The selected datasets all have 100,000 CRPs, what differentiates them from each other is the initial pool of CRPs they were randomly selected from. There is only one relevant data point on the control side in this case because there is no variation possible, only selection of 100,000 CRPs selected randomly from the original training dataset. The selected datasets had from 100,000 to 500,000 samples, in 100,000 intervals.

As can be seen in Fig. 4.1 the best results were obtained using the dataset with 300,000 CRPs, obtaining the lowest accuracy at 74.167%. The results also demonstrate the expected: that the larger the amount of samples available the less effective the technique is, given that a larger pool of data gives the model more chance to properly discern the PUF's internal patterns. There is also an interesting phenomenon when looking at the results for smaller amounts of CRPs, they also have higher accuracies when compared to the lowest at 300,00 samples. The most likely explanation is that when constrained to the hardest samples to classify when taking the entire dataset into consideration, the Machine Learning model was able to find a pattern within them.

One weakness of this approach is the fact that the attacker will probably be aware it is being used, this gives them an opportunity to prepare a more effective attack.

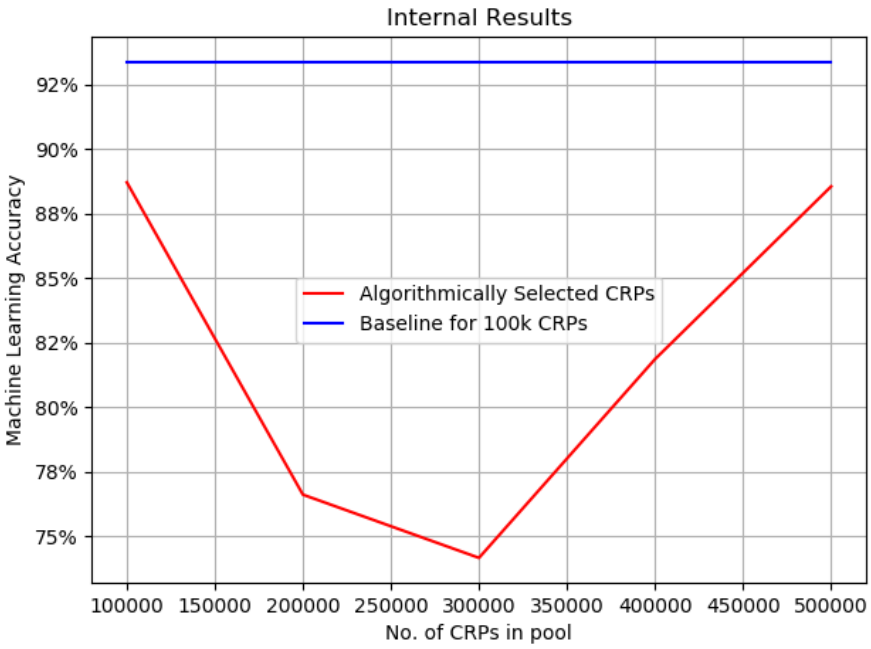

Figure 4.1: Scores for the Single Group with Poor Internal Recognition Technique. The red line represents the accuracy of the models trained with the tainted data sets (*Selected*), while the blue line represents the accuracy of a model trained with random 100,000 CRPs (*Control*). [4]

In order to test how an attacker could try to improve their odds of successfully cracking a PUF in the face of this technique being used by using a similar method. In order to do so, we devised two different selection approaches: one using the CRPs with the lowest weights associated to them, the other was using a random selection biased by the weights associated to each CRP.

We tested these methods on the dataset that had the most accuracy when training a Machine Learning model with a dataset selected from a pool of 300,000 CRPs selected using the Single group with poor internal recognition technique. First, we separated this new data set in a form similar to k-fold with k = 10, resulting in ten different permutations of 30,000 samples set aside for testing and 270,000 samples for training. After that, we used the same technique to generate weights for each CRP in the original datasets for the new training samples. Following that, we used the methods proposed for improving accuracy earlier and tested them all using the 30,000 samples that weren't used for training in each permutation. The results of the fold that had the highest accuracies was 76.803% using random selection, 66.923% using the lower-weighted CRPs and 84.413% when using the biased random selection.

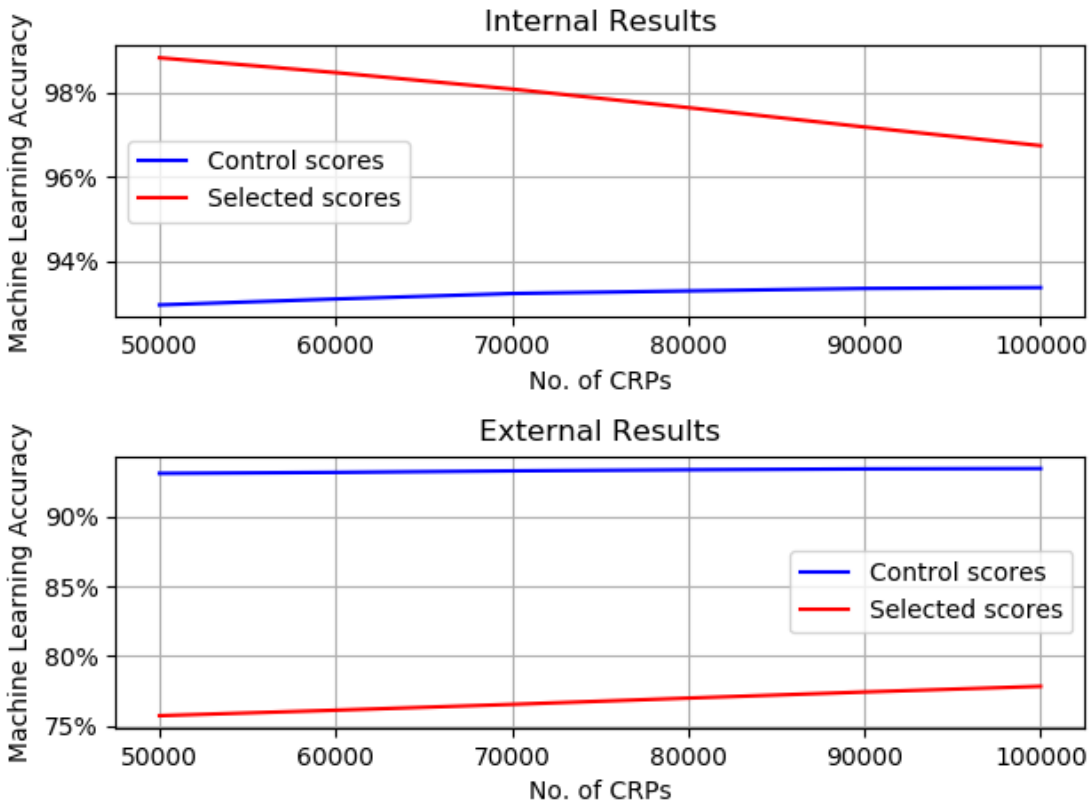

Figure 4.2: Accuracy charts for the Single Group with Poor External Recognition and Good Internal Recognition Technique. The red lines represent the accuracies of the models trained with algorithmically selected data sets and the blue lines represent the accuracies of models trained with random CRPs. [4]

4.2.2 Single group with good internal recognition and poor external recognition

For this experiment we tested control and selected (by the Single group with good internal recognition and poor external recognition technique) datasets using the k-fold method and the separate testing datasets with 100,000 CRPs. The training dataset sizes vary from 50,000 samples to 100,000 in 10,000 intervals. The proportion of high-weight/low-weight CRPs in the selected training datasets was 60%/40%.

As can be seen in Fig. 4.2, when tested by performing k-fold validation (referred to in the figure as "Internal Recognition") the specially selected subset appears to provide a model that appears to crack the hypothetical PUF, achieving accuracies above 96% for all tests, higher than even the control datasets. On the other hand, while the external results for the control datasets maintain similar numbers to the internal results, the selected datasets performed much worse, ranging between 75% and 78%. This indicates that the technique is working as expected, creating a dataset that has good internal recognition and poor external recognition.

Table 4.1: Results of the multiple groups with poor external recognition technique

Phase	High-weight CRP Proportion		TGA	FGA	IAA
	Training	Testing			
1	1.00	0.2	50.839%	49.471%	96.608%
2	0.60	0.3	80.767%	93.071%	96.256%
3	0.50	0.7	85.371%	85.025%	96.056%
4	0.55	0.5	86.385%	86.951%	93.428%

4.2.3 Multiple groups with poor external recognition

This experiment is more complicated than the rest, reflecting the increased complexity of the technique it is testing. First, we set aside groups of CRPs selected using the technique described in subsection 3.4.2, by taking combinations of samples from the highest and lowest weights. Then we train a Machine Learning model with the first selected group and test it using the group that is scheduled to be used next, a separate testing dataset and by performing k-fold validation. After that the selected group that was used for testing is added to the training data and the process is repeated with the next selected group used for testing. This procedure is repeated until the last selected group is used for testing. The experiments were performed with 5 groups of 50,000 CRPs each. The best proportions for each selection was determined empirically.

The results for the tests described earlier are displayed on Table 4.1. The Phases column refers to what step in the testing the results refer to. The High-weight CRP Proportion columns represent how much of the dataset was selected from the samples associated with high weights in the training and the specially selected testing datasets. The TGA, or Test Group Accuracy, column displays the accuracy of the Machine Learning model when tested using the separate test dataset. The FGA, or Following Group Accuracy, column displays the accuracy of the Machine Learning model when tested using the following group. Finally, the IAA, or Internal Average Accuracy, is the accuracy of the Machine learning model using k-fold validation.

As observed in Table 4.1, for the most part the technique seems to keep the accuracies within the desired range. The IAA is constantly at a point that would be considered cracking a PUF, while the TGA is consistently lower. The only exception is the FGA, that at phase 2 (with the first 2 groups used for training and the third being tested) is above the target accuracy and would crack a PUF if used in this way in an actual use case.

Chapter 5

Conclusion & Future Works

In this chapter we conclude this dissertation with our final thoughts and suggestions of future works that could expand on this one.

5.1 Final Considerations

The results achieved by this project demonstrate that there is potential for improving PUF security by using adversarial selection, a result generally sought after by altering a PUF's topology or measurement methodology for outputs. As can be observed in the previous chapter, our methods cause significant reduction in the accuracy of Machine Learning modeling attacks, that the PUF used for testing would ordinarily be vulnerable to. This approach developed here is appealing because it could be used on existing PUFs, even those already in operation.

The selection method seemed to excel specifically in cases where only one subset was being created. The Single group with poor internal recognition technique performed as designed and would be a useful resource if applied in the industry, in spite of the constraint on the amount of CRPs available for use. The Single group with good internal recognition and poor external recognition method also had adequate results, regardless of the limitations of its implementation.

In spite of the gains presented here, there are still shortcomings in these selection techniques. Beyond the problems mentioned earlier, the Multiple groups with poor external recognition method wasn't able to fulfill its purpose for the entirety of its test. Although this is a problem, it also presents an interesting opportunity for more research, and the development of new methods, better suited for the task.

5.2 Future Works

There are a few potential paths to making the Multiple groups with poor external recognition method functional. For instance, different selection criteria could be devised to generate the subset for each phase. Another possibility would be to vary the amount of samples in each of the selected subsets. In any case, there is definitely room for improvement when it comes to that technique.

The research presented here only used the Arbiter PUF as a source of CRPs, using other types of PUF could be fruitful. Considering that the Arbiter PUF is the least secure strong PUF, using the techniques developed here with more robust PUFs would provide a marked increase in security in their use. There is no reason to believe the contrary, but only after testing can we be certain.

Other Machine Learning models would also be a valuable addition to this study. While it is not expected that the results would be very different, verifying that these methods have similar impact on the training of other Machine Learning models would grant some degree of certainty. Beyond that, it is also possible to use those other models to create something similar to the weight array used in this project to assist in the selection methods. For example, a Support Vector Machine (SVM) determines what distance a sample is from the function drawn by its classifier, this information could be used in the same way as we use the weights generated by AdaBoost.

Bibliography

[1] BURHAN, M., REHMAN, R. A., KIM, B.-S., et al. "IoT Elements, Layered Architectures and Security Issues: A Comprehensive Survey", *Sensors*, v. 18, 08 2018. doi: 10.3390/s18092796.

[2] LEE, J., LIM, D., GASSEND, B., et al. "A technique to build a secret key in integrated circuits for identification and authentication applications". In: *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on*, pp. 176–179, June 2004.

[3] VIJAYAKUMAR, A., PATIL, V. C., PRADO, C. B., et al. "Machine learning resistant strong PUF: Possible or a pipe dream?" In: *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 19–24, May 2016.

[4] FRANÇA, H. L., PRADO, C. B., PATIL, V. C., et al. "Defeating Strong PUF Modeling Attack via Adverse Selection of Challenge-Response Pairs". In: *2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pp. 25–30, Dec 2018. doi: 10.1109/AsianHOST.2018.8607165.

[5] WURM, J., HOANG, K., ARIAS, O., et al. "Security analysis on consumer and industrial IoT devices". In: *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 519–524. IEEE, 2016.

[6] BRAEKEN, A. "PUF based authentication protocol for IoT", *Symmetry*, v. 10, pp. 352, 08 2018. doi: 10.3390/sym10080352.

[7] YILMAZ, Y., GUNN, S. R., HALAK, B. "Lightweight PUF-Based Authentication Protocol for IoT Devices". In: *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*, pp. 38–43, July 2018. doi: 10.1109/IVSW.2018.8494884.

[8] RÜHRMAIR, U., SEHNKE, F., SÖLTER, J., et al. "Modeling Attacks on Physical Unclonable Functions". In: *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pp. 237–249, New York, NY, USA, 2010. ACM. ISBN: 978-1-4503-0245-6.

[9] VIJAYAKUMAR, A., KUNDU, S. "A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics". In: *Design, Automation Test in Europe Conference Exhibition (DATE), 2015*, pp. 653–658, March 2015.

[10] PAPPU, R., RECHT, B., TAYLOR, J., et al. "Physical One-Way Functions", *Science*, v. 297, n. 5589, pp. 2026–2030, 2002.

[11] ADAMES, I. A. B., DAS, J., BHANJA, S. "Survey of emerging technology based physical unclonable funtions". In: *2016 International Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 317–322, May 2016. doi: 10.1145/2902961.2903044.

[12] HERDER, C., YU, M., KOUSHANFAR, F., et al. "Physical Unclonable Functions and Applications: A Tutorial", *Proceedings of the IEEE*, v. 102, n. 8, pp. 1126–1141, Aug 2014. ISSN: 0018-9219. doi: 10.1109/JPROC.2014.2320516.

[13] ZHOU, C., PARHI, K. K., KIM, C. H. "Secure and reliable XOR arbiter PUF design: An experimental study based on 1 trillion challenge response pair measurements". In: *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2017. doi: 10.1145/3061639.3062315.

[14] SAHOO, S. R., KUMAR, S., MAHAPATRA, K. "A novel reliable and aging tolerant modified RO PUF for low power application", *Analog Integrated Circuits and Signal Processing*, Aug 2018. ISSN: 1573-1979. doi: 10.1007/s10470-018-1317-z. Disponível em: <`https://doi.org/10.1007/s10470-018-1317-z`>.

[15] TAN, A. C., GILBERT, D., DEVILLE, Y. "Multi-class protein fold classification using a new ensemble machine learning approach", *Genome Informatics*, v. 14, pp. 206–217, 2003.

[16] SUN, X. "Pitch accent prediction using ensemble machine learning". In: *Seventh international conference on spoken language processing*, 2002.

[17] HAN, L., LUO, S., YU, J., et al. "Rule Extraction From Support Vector Machines Using Ensemble Learning Approach: An Application for Diagnosis of Diabetes", *IEEE Journal of Biomedical and Health Informatics*, v. 19, n. 2, pp. 728–734, March 2015. ISSN: 2168-2194. doi: 10.1109/JBHI.2014.2325615.

[18] YERIMA, S. Y., SEZER, S., MUTTIK, I. "High accuracy android malware detection using ensemble learning", *IET Information Security*, v. 9, n. 6, pp. 313–320, 2015.

[19] MEIR, R., RÄTSCH, G. "An introduction to boosting and leveraging". In: *Advanced lectures on machine learning*, Springer, pp. 118–183, 2003.

[20] PAPERNOT, N., MCDANIEL, P. D., GOODFELLOW, I. J., et al. "Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples", *CoRR*, v. abs/1602.02697, 2016. Disponível em: <http://arxiv.org/abs/1602.02697>.

[21] MYERSON, R. B. "Refinements of the Nash equilibrium concept", *International journal of game theory*, v. 7, n. 2, pp. 73–80, 1978.

[22] KURAKIN, A., GOODFELLOW, I. J., BENGIO, S. "Adversarial examples in the physical world", *CoRR*, v. abs/1607.02533, 2016. Disponível em: <http://arxiv.org/abs/1607.02533>.

[23] BIGGIO, B., DIDACI, L., FUMERA, G., et al. "Poisoning attacks to compromise face templates". In: *6th IAPR Int'l Conf. on Biometrics (ICB)*, Madrid, Spain, 06/2013 2013.

[24] GOODFELLOW, I. J. "NIPS 2016 Tutorial: Generative Adversarial Networks", *CoRR*, v. abs/1701.00160, 2017. Disponível em: <http://arxiv.org/abs/1701.00160>.

[25] BIGGIO, B., CORONA, I., MAIORCA, D., et al. "Evasion attacks against machine learning at test time". In: Blockeel, H. (Ed.), *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, v. 8190, pp. 387–402. Springer-Verlag Berlin Heidelberg, 2013.

[26] KOLOSNJAJI, B., DEMONTIS, A., BIGGIO, B., et al. "Adversarial Malware Binaries: Evading Deep Learning for Malware Detection in Executables", *CoRR*, v. abs/1803.04173, 2018. Disponível em: <http://arxiv.org/abs/1803.04173>.

[27] BIGGIO, B., FUMERA, G., ROLI, F. "Security evaluation of pattern classifiers under attack", *IEEE Transactions on Knowledge and Data Engineering*, v. 26, pp. 984–996, 04/2014 2014.

[28] WAGNER, D., SOTO, P. "Mimicry Attacks on Host-based Intrusion Detection Systems". In: *Proceedings of the 9th ACM Conference on Computer*

and Communications Security, CCS '02, pp. 255–264, New York, NY, USA, 2002. ACM. ISBN: 1-58113-612-9. doi: 10.1145/586110.586145. Disponível em: <http://doi-acm-org.ez29.capes.proxy.ufrj.br/10.1145/586110.586145>.

[29] WANG, S.-J., CHEN, Y.-S., LI, K. S.-M. "Adversarial Attack Against Modeling Attack on PUFs". In: *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, pp. 138:1–138:6, New York, NY, USA, 2019. ACM. ISBN: 978-1-4503-6725-7. doi: 10.1145/3316781.3317761. Disponível em: <http://doi.acm.org/10.1145/3316781.3317761>.

[30] AZIZ, B., HAMILTON, G. "Detecting Man-in-the-Middle Attacks by Precise Timing". In: *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, pp. 81–86, June 2009. doi: 10.1109/SECURWARE.2009.20.

[31] Y., B., Y., G. "No Unbiased Estimator of the Variance of K-Fold Cross-Validation", *Journal of Machine Learning Research*, v. 5, pp. 1089–1105, Sept 2004. ISSN: 1532-4435.