A Dissertation

entitled

Hardware Security Design, and Vulnerability Analysis of FPGA based PUFs to Machine Learning and Swarm Intelligence based ANN Algorithm Attacks

by

Ahmed Oun

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the

Doctor of Philosophy Degree in Engineering

Dr. Mohammed Niamat, Committee Chair

Dr. Ahmad Y Javaid, Committee Member

Dr. Weiqing Sun, Committee Member

Dr. Junghwan Kim, Committee Member

Dr. Richard Molyet, Committee Member

Dr. Amy Thompson, Interim Dean
College of Graduate Studies

The University of Toledo
May 2022

An Abstract of

Hardware Security Design, and Vulnerability Analysis of FPGA based PUFs to Machine Learning and Swarm Intelligence based ANN Algorithm Attacks

by

Ahmed Oun

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Doctor of Philosophy Degree in Engineering

The University of Toledo
May 2022

With the increasing trend of outsourcing the fabrication process of Integrated Circuits (ICs) to foreign foundries, hardware security threats have significantly increased. Of particular concern is the infiltration of the IC supply chain with compromised and counterfeit chips by untrusted and dubious foundries. In recent years, the use of programmable devices such as Field Programmable Gate Arrays (FPGAs) has rapidly increased. The increased deployment of these devices in mission-critical computing systems such as communication networks, smart grids, defense equipment, and internet of things; has led hackers to continually devise new techniques to breach the security of these devices. Of serious concern is the implantation of a spurious circuitry, known as a Trojan, to steal or degrade the function of the chip. These tampered chips can subsequently act as 'spy chips' for collecting confidential data by adversaries and hackers. To counter such attacks, a chip designer can embed additional security layers in these devices using Physical Unclonable Functions (PUFs). Although PUFs are supposed to be unclonable and unbreakable, researchers have found that they are vulnerable to Machine Learning (ML)

attacks. From a subset of challenge-response pairs (CRPs), the remaining CRPs can be effectively predicted using different machine learning algorithms.

This research presents a comprehensive vulnerability analysis of different FPGA based PUFs to various Swarm Intelligence (SI) based ANN Algorithms (SI) attacks; namely, Dragonfly Algorithm (DA), Gravitational Search Algorithm (GSA), Cuckoo Search Algorithm (CS), Particle Swarm Optimization (PSO), and the Grey Wolf Optimizer (GWO) algorithms. These algorithms are used to build Artificial Neural Network models to analyze the vulnerability of the different PUFs for modeling attacks. The training algorithms adjust the weights and biases of the ANN to obtain the highest response prediction accuracy by finding their optimum set. To the best of our knowledge, swarm intelligence-based algorithms have not been used in studying the vulnerability of PUFs to ANN-based attacks. The results show that the swarm intelligence algorithms produce better response prediction accuracies results (71.1% - 99.3%) when compared to other well-known ML algorithms. Amongst the various SI and ML algorithms, the GWO algorithm performs the best in predicting the CRPs.

This research further focuses on using different machine learning classifiers attacks, namely: Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), K-Nearest Neighbor (KNN), kernel Support Vector (Kernel SVM), Naive Bayes Classifier (NB). Different Artificial Neural Networks (ANN) based models to study the vulnerability of PUFs are used by modeling the challenge-response pairs. The ANN models are implemented using four different optimization techniques; namely: Root Mean Square Propagation (RMSprop), Adaptive delta (Adadelta) learning rate method for gradient descent, Adaptive Moment Estimation (Adam), and Nesterov-accelerated Adaptive

Moment Estimation (Nadam). The challenge-response data obtained from different PUFs are trained using various modeling algorithms. The results show that the ANN-based algorithms produce better response prediction accuracies results (68.0% - 94.1%) when compared with other ML algorithms.

Two different novel XOR-ROPUFs capable of thwarting various machine learning modeling attacks, and enhancing the security of the PUFs, are also designed. The proposed designs are implemented on Xilinx Artix-7 FPGAs. These PUFs generate an ‘n’ bit response for an ‘n’ bit challenge (n x n); the new response is an ‘n’ bit vector so that the prediction accuracy is calculated based on predicting the number of bit strings (n x n) for different challenges. The new PUF structures drastically reduces the prediction accuracy of CRPs to 24.1%.

For the remaining part of this research work, an authentication scheme is proposed for the security of IoT systems using a lightweight XOR-ROPUF. The proposed management scheme carries out the authentication between the verification authority, the authentication server**,** and the IoT devices to ensure data congeniality and integrity. The proposed XOR-ROPUF based scheme implements a low-cost device authentication solution for identifying the trusted hardware, securing communication among the devices using a lightweight system, and reducing the risk of authentication vulnerability.

Dedicated to my loving and caring parents, my devoted wife, my lovely children and my entire family for their everlasting love, undying faith, trust, support, patience, and encouragement.

Acknowledgements

I would like to express my deep gratitude and appreciation to my advisor, Dr. Mohammed Niamat, for his supportive recommendations, guidance, and encouragement that conducted this research to a successful conclusion and accomplished. I would also like to thank my respected committee members for their critical comments, suggestions and for helping develop this dissertation. My sincere thanks to my wife, children, parents, and family for their ultimate love and continuous support, understanding, and sacrifices throughout this Ph.D. journey. Finally, I wish to acknowledge and give thanks to all my friends and lab mates for their friendship, encouragement, and support, which make my Ph.D. journey a wonderful experience.

Thank you All

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI ..... Artificial Intelligence
APUFs ..... Arbiter Physical Unclonable Functions
ASICs ..... Application Specific Integrated Circuits

CLB ..... Configurable Logic Blocks
CMOS ..... Complementary Metal Oxide Semiconductor
CRP ..... Challenge-Response Pair
CS..... Cuckoo Search Algorithm

FPGA .....Field Programmable Gate Array

GA.....Genetic algorithms
GSA.....Gravitational Search Algorithm
GWO .....Grey Wolf Optimizer

ICs .....Integrated Circuits
IP .....Intellectual Property

LFSR .....Linear Feedback Shift Register
LUT .....Look-Up Table
LR .....Logistic Regression

NIST .....National Institute of Standards and Technology
NB.....Naive Bayes

ROPUFs .....Ring Oscillator Physical Unclonable Functions
RF.....Random Forest
RTL .....Register Transfer Level
SA .....Swarm Algorithms
SVM.....Support Vector Machine

VHDL .....VHSIC Hardware Description Language
VLSI .....Very Large Scale Integration

Chapter 1

Introduction

In recent years, the use of programmable devices such as Field Programmable Gate Arrays (FPGAs) and custom designed Application Specific Integrated Circuits (ASICs) have increased rapidly. The increased deployment of these devices in mission critical computing systems include, but are not limited to, communication networks, smart grids, defense equipment, and internet of things, has led hackers to continually devise new techniques to breach the security of these devices. Examples of such attacks include disabling or degrading the function of these chips in systems like radars and missiles. Other attempts include implanting malicious electronic circuitry in the chips, known as Trojans, to steal vital information for cyber-attacks. These tampered chips can subsequently act as 'spy chips' by collecting confidential data for adversaries and hackers. To counter such attacks, chip designers have embedded additional layers of security in these devices [1,2]. Although researchers have long tried to secure hardware-based systems with both software and hardware-based approaches, this research explicitly focuses on techniques based on hardware-oriented security and trust [3,4]. These approaches mainly involve generation of unique hardware-based cryptographic keys in the form of Challenge-Response Pairs (CRPs). In order to generate hardware-based unique keys, different structures of physical

unclonable functions (PUFs) have been proposed in the past [5,6]. Essentially, a PUF utilizes manufacturing process variation, which is an inherent property of silicon chips, to generate unique and unclonable CRPs. Amongst the different types of PUFs available, the delay-based PUFs are widely studied in CMOS-based silicon devices. The most investigated PUFs on silicon-based devices are the Ring Oscillator PUF (ROPUF) and the Arbiter PUF (APUF). Most of the delay-based PUFs are strong candidates for not only ASICs but also for FPGAs [7,8]. The significant advantage of using PUFs as security measures is that it does not require on-chip memory to generate and store keys; thus, it eliminates the use of on-chip memory for the security of the hardware-based system. Another very significant feature of the PUF is that the keys generated by the PUF are device specific. Further, the keys change with the specific location and placement of the PUF inside the chip, since they depend on the random manufacturing process variations [9,10]. It should be noted that the behavior of PUFs rely on the random manufacturing process variations related to several components that are used to construct it. These components are sometimes linearly interrelated to the number of CRPs. Because of these limitations and linearity, an attacker may try all challenges and know the corresponding responses within an extended period of time [6]. This kind of brute force approach, however, generally fails because of the time required and because of the fact that the exact location of the PUF is unknown. It is further complicated in FPGAs, since the location of a PUF mapped onto an FPGA, unlike an ASIC, can be frequently changed by the designer by changing the bit-stream file.

PUF produces a device-specific unique response for a given challenge. This property of the PUF makes it suitable for different applications including, authentication,

IP protection, random key generation, remote attestation, and secured supply chain, etc. Once the CRPs can be predicted by an attacker; as a consequence, the whole concept of cryptographic primitive for hardware security applications, including PUF as an authenticator, is in jeopardy. Though PUFs are considered unclonable, researchers have shown that they are vulnerable to machine learning-based modeling attacks. An attacker can perform different types of attacks, including side-channel attacks, cloning, reverse engineering, Probably Approximately Correct (PAC) based attacks, and eavesdropping for predicting the CRPs [11-15]. Side-channel based attacks can be performed by monitoring the voltage, current, and power values during runtime. If an attacker wants to authenticate using PUF CRPs without getting any access to the PUF, the attacker would be able to do so if the attacker has the responses available for the challenges, which can be done by eavesdropping on some of the CRPs. Hackers can eavesdrop by using MITM attacks by recording the network data packets and extracting the information of the CRPs when the system is in operation. Thus, after acquiring a set of CRPs, a PUF can be modeled using machine learning. Side-channel based attacks can be performed by monitoring the voltage, current, and power during runtime. PUFs have been successfully attacked using machine learning algorithms such as Logistic Regression, Probably Approximately Correct learning, Evolutionary Strategy, Quick Sorting, etc. [13-15]. In Rührmair et al.'s research [13], the authors used quick sorting for modeling RO PUFs. In J. Delvaux's work [14], the authors performed modeling attacks on APUF, PolyPUF, OB-PUF, RPUF, LHS-PUF, and PUF FSM protocols. The Probably Approximately Correct (PAC) learning algorithm has been used for predicting ROPUF CRPs in [15]. In their work, the number of CRPs required to learn the models is on a scale of ten thousand which is high for an attacker to obtain

from the CRP set. Fault injection-based modeling attacks on APUFs are performed in [16]. In this attack model, an attacker must have physical access to the PUF. Logical Approximation and Global Approximation attacks are performed on different structures of Arbiter PUFs using ANN methods of RMSProp and Gradient Descent Optimizer [17]. In this technique, the number of CRPs required is also high. Different side channel-based modeling attacks have been performed in [18,19], which also requires physical access to the PUF device. Authors in [20] performed deep learning attacks on a Double Arbiter PUF. In their work, the authors performed Logistic Regression-based deep learning attacks. The number of CRPs required to perform such an attack is very high and requires more than a million pairs of CRPs that are difficult for an adversary to obtain in order to attack the PUF.

Genetic Algorithms have also been used to predict CRPs for the ROPUF [21,22]. In the Genetic Algorithm-based modeling, CRPs are generated by crossover, mutation, and then the attacks are performed, which is not consistent for different models of ROPUFs. Mathematical modeling of different PUFs including the Arbiter PUF and the Ring Oscillator PUF has been performed in [23]. In this work, the authors describe a mathematical model for the ROPUF and perform Logistic Regression-based modeling attacks on the Arbiter PUF and the DCMUX PUF. In this approach, the drawback is that the CRPs depend on the different structures of the ROPUF. ML-based modeling attacks on a small set of CRPs using different optimizations including RMSprop, Adam, Nadam, etc., have been performed in [24]. However, the prediction accuracy needs improvement.

Different metaheuristics algorithms exist in the literature to solve optimization problems. Metaheuristics algorithms can be classified into different categories including, Evolutionary, Physics-based, and Swarm Intelligence-based Algorithms. The Genetic

Algorithm (GA) is the most popular Evolutionary based algorithm proposed by [25], which works on an initial random solution and optimizes the solution based on generations and mutations. Other popular Evolutionary based algorithms are Genetic Programming (GP) [26], Evolutionary Strategies (ES) [27], Differential Evolution (DE) [28] etc. The popular physics-based algorithms is the Gravitational Search Algorithm (GSA) [29] which works based on the law of gravity, and the best solution is reached after the iteration can produce specific agents that achieve certain fitness. Ultimately, the heavier; the mass is, the closer the optimum points will be. Other physics-based algorithms include Big-Bang Big-Crunch (BBBC) [30], Central Force Optimization (CFO) [31], Galaxy-based Search Algorithm (GbSA) [32], Gravitational Local Search (GLSA) [33], Charged System Search (CSS) [34] etc. Swarm Intelligence (SI) based algorithms are a subset of the bio-inspired algorithms. SI is a nature-inspired algorithm produced by a group of animals or birds acting together, and the algorithm is based on how these animals act or behave to adapt to the different scenarios occurring in their surroundings [35]. In Particle Swarm Optimization (PSO) the particles chase the position of the best particle and reach their own best position so that the overall best solution of the swarm is obtained [36]. Other popular swarm intelligence-based algorithm includes Ant Colony Optimization (ACO) [37], Cuckoo Search (CS) [38], Grey Wolf Optimizer (GWO) [39], etc., which are inspired by hunting and searching behavior.

In 2014, Mirjalili et al. introduced the Grey Wolf Optimizer (GWO), which is a metaheuristic algorithm that simulates the hierarchical superiority-based hunting mechanism of grey wolves for hunting down prey. This arrangement benefits them to preserve stability and support each other throughout hunting. Wolves have a strict social hierarchy consisting of the alpha (α), beta (β), delta (δ), and omega (ω) wolves [39]. The

GWO algorithm takes these features of Grey Wolf to search optimized solution of a problem utilizing exploitation and exploration; therefore, in the searching process, the best solution position can be comprehensively estimated by three solutions. Thus, the algorithm can significantly decrease the probability of falling into the local optimum. The properties of metaheuristics algorithms have motivated their use to solve different engineering problems such as embedded systems, electric power system [40], scheduling Energy Storage Unit problems [41], communication network and Distributed Compressed Sensing (DCS) problem [42]. Hence, the research on the swarm intelligence optimization algorithms has an academic advantage and practical importance.

1.1 Motivation

The design of delay-based Physical Unclonable Function (PUF) on Field Programmable Gate Arrays (FPGAs) has developed rapidly because of their accelerated reconfigurability, parallelism, and low cost which overcome the limitations of traditional cryptography. Therefore, hackers continually devise different types of specialized attacks to breach and counterfeit these devices' security and piracy. The attackers implementing various types of machine learning attacks in order to emulate PUF’s behavior will lead to expanded research in hardware-oriented security and trust to secure the system from such growing threats by deploying additional security mechanisms for FPGA-based systems using Physical Unclonable Function.

1.2 Research Objectives

- A novel implementation of Artificial Neural Network-based modeling attacks on various PUFs using different Swarm Intelligence algorithms namely, Dragonfly Algorithm (DA), Gravitational Search Algorithm (GSA), Cuckoo Search Algorithm (CS), Particle Swarm Optimization (PSO), and the Grey Wolf Optimizer (GWO) algorithms. To the best of our knowledge, these algorithms have not been used in studying the vulnerability of PUFs to ANN-based attacks.
- Machine Learning Vulnerability analysis of different delay-based PUF using different ML classifiers: Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), K-Nearest Neighbor (KNN), kernel Support Vector (Kernel SVM), Naive Bayes Classifier (NB) classifiers.
- Use of Artificial Neural Network based modeling attacks on various PUFs using different optimizers, namely, Root Mean Square Propagation (RMSprop), Adaptive delta (Adadelta) learning rate method for gradient descent, Adaptive Moment Estimation (Adam), and Nesterov-accelerated Adaptive Moment Estimation (Nadam) to study their vulnerability to theses machine learning attacks.
- Design a novel XOR-ROPUF capable of thwarting machine learning attacks. This design is more secure from hackers who can use both invasive and non-invasive techniques to steal the CRP data.
- Design of a unique delay-based PUF structure that combines an Arbiter PUF with a XOR- ROPUF for thwarting machine learning attacks.

- Development of an authentication scheme for the security of IoT systems using a lightweight XOR-ROPUF. The proposed management scheme carries out the authentication between the verification authority, the authentication server, and the IoT devices to ensure data congeniality and integrity.

1.3 Dissertation Outline

This dissertation is organized as follows:

- **Chapter 2:** This chapter provides a brief overview and background information about our research, including the manufacturing process variations, different types of physical unclonable functions, and further modeling attacks and security.
- **Chapter 3:** The chapter presents a novel implementation of Artificial Neural Network-based modeling attacks on various PUFs using different Swarm Intelligence algorithms to study their vulnerability against these attacks.
- **Chapter 4:** The chapter presents different Machine Learning classifiers (ML) and Artificial Neural Networks (ANNs) based modeling attacks are used against different silicon-based PUFs to test and analyze their vulnerability to these attacks.
- **Chapter 5:** This chapter introduces the design and implementation of a novel XOR-ROPUF capable of thwarting machine learning attacks.
- **Chapter 6:** The chapter presents a unique design of a hybrid delay-based FPGA PUF resistant to machine learning attacks. The technique combines challenge-response pairs from an Arbiter PUF with an XOR-based Ring Oscillator to generate responses that are less vulnerable to machine learning modeling attacks.

- **Chapter 7:** This chapter introduces an authentication scheme for the security of Internet of Things Devices using a lightweight XOR-ROPUF. The proposed management scheme carries out the authentication between the verification authority, the authentication server, and the IoT devices to ensure data congeniality and integrity.
- **Chapter 8:** presents conclusions and potential future work.

Chapter 2

Research Background

2.1 Manufacturing Process Variations

During the various manufacturing steps of integrated circuits, random variations take place due to the variety of oxide thickness, mask variation, process temperature, and pressure, which is termed as manufacturing process variation [43]. Process variation is an inherent property of an IC, and it varies from chip to chip. Process variation differs across individual devices even if the IC class is the same, and they are manufactured from the same wafer. This results in propagation delay in integrated circuits. Process variation increases with the shrinking of device size. By using the process variation property of integrated circuits, the physical unclonable function is designed to produce unique challenge-response pairs [43]. The important parameters which control the process manufacturing variations in an IC are listed in Table 2.1 [44].

Table 2.1: Process variation parameters.

Environment	Gate	Manufacturing	MOS
Voltage	Delay (D)	Nd= Doping	Threshold (Vt)
Temperature	Current (Isub)	tox= thickness	Capacitance (Cg)
Ageing		C=chemical	length (Leff)

2.2 Physical Unclonable Functions (PUFs)

Physical unclonable functions (PUF) is a circuit constructed on a semiconductor device that produces a unique signature using manufacturing process variation to generate unique and unclonable challenge-response pairs. PUFs are unpredictable and unclonable to implement hardware-based security schemes in devices which security primitives for hardware-oriented security applications due to their instance-specificity based on physical properties [45]. PUFs are used to extract unique signatures from silicon-based chips, which can be used for chip authentication and producing unclonable cryptographic keys. This property of the PUF makes it suitable for different applications, including authentication, IP protection, random key generation, remote attestation, secured supply chain, etc. Based on the design principles, randomness and construction properties, PUFs can be classified into different basic categories mainly classified into either Intrinsic PUFs [46] or Non-Intrinsic PUFs [47].

PUFs for which the randomness elements are explicitly introduced are termed non-intrinsic PUFs. The randomness of non-intrinsic PUFs is internally evaluated using specific

equipment. Optical PUFs and Coating PUFs are instances of non-intrinsic PUF types. On the contrary, in intrinsic PUFs, the evaluations are performed internally and mainly use the intrinsic manufacturing process variations of ICs to extract unique and random signatures to authenticate silicon chips. Therefore, its random state properties are introduced during the manufacturing processes. The intrinsic introduced PUFs, including Delay based PUFs, Arbiter PUFs (APUFs) [48,49] Butterfly PUFs [50], and Ring Oscillator PUF [5]. From the different types of PUFs, delay-based PUFs are widely studied in CMOS-based silicon devices. Delay-based PUFs response depends on the difference in the signal's propagation delay between two delay paths. These propagation delays occur due to the process variations during the manufacturing process. The most notable of these delay-based is the Ring Oscillator PUF (ROPUF) and arbiter PUF (APUF) [5]. Figure 2-1 demonstrates PUF's overall classification.

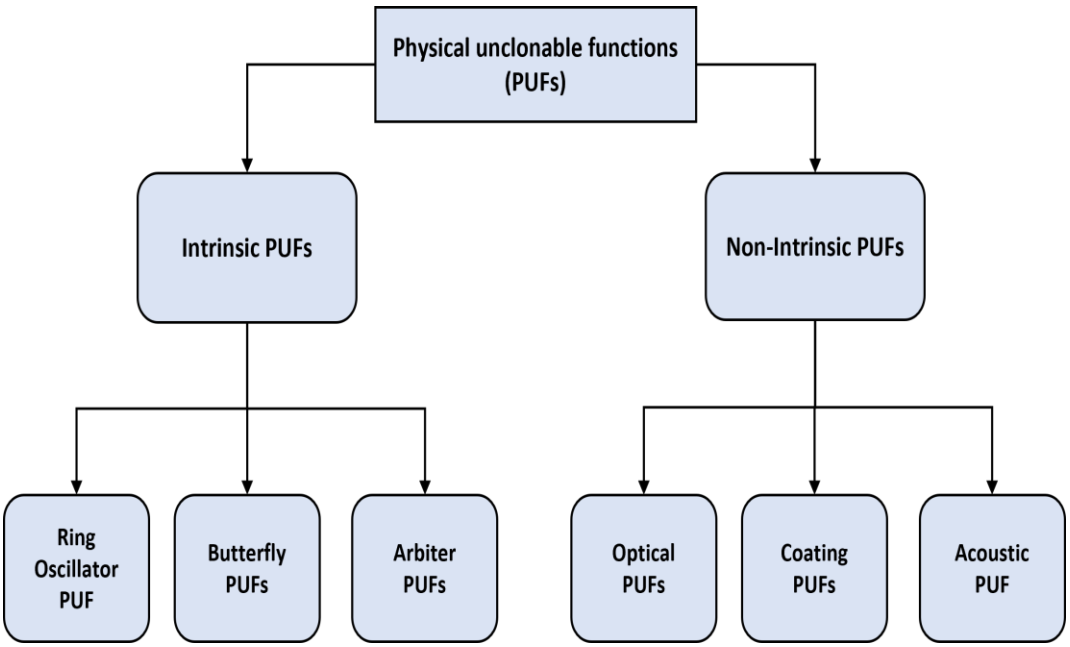

Figure 2-1: PUF Classification

2.2.1 Optical PUF

Optical PUFs are non-electronic technologies presented in [51,52]. Optical PUFs are established on transparent media as one-way physical functions and random optical reflection patterns. These PUFs are based on the random fiber structure of a sheet of paper or the random reflection of the optical properties of the characteristic medium. The PUF uses a unique pattern that mounts when spreading laser against a transparent object to generate a random PUF signature. The design mainly relies on optical microstructure produced by combining refractive glass spheres in a small transparent epoxy plate. Figure 2-2 represents the implementation and essential operation of Optical PUF [46]. Due to the extensive design and system complexity, it is clear that Optical PUFs are infrequent and have limited use.

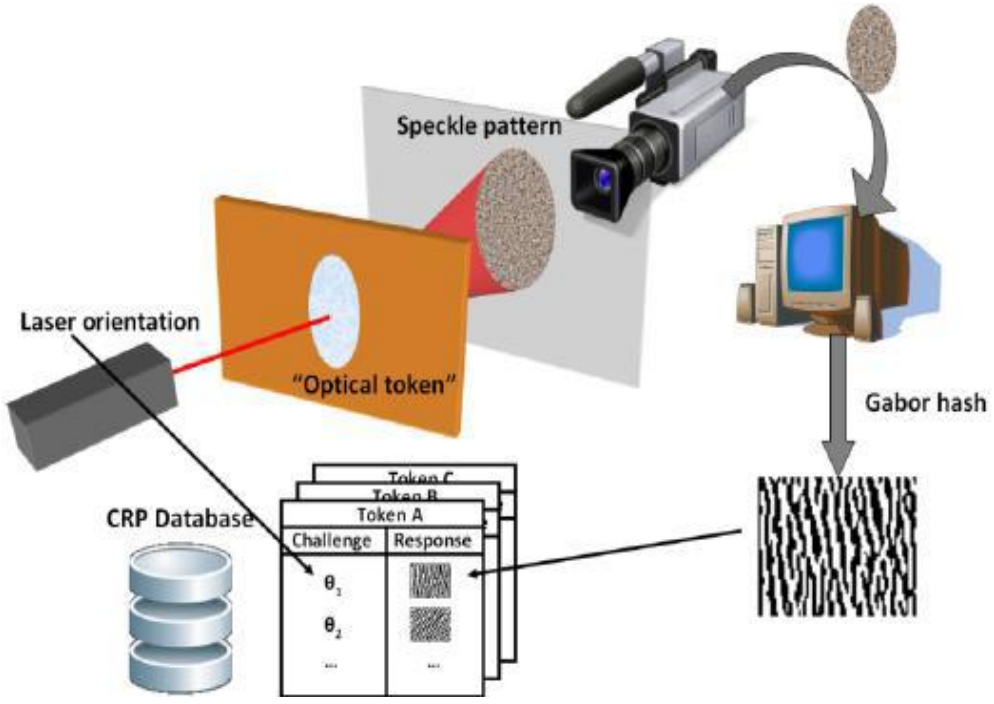

Figure 2-2: Optical Physical Unclonable Function [46]

2.2.2 Coating PUF

Coating PUFs is a non-intrinsic PUF proposed by Tuyls et al. and shown in Figure 2-3 [53]. Coating PUF works on the internal randomness of the capacitance measurement using sensors located on the top IC metal layer. The randomness is explicitly introduced using a passive dielectric coating sprayed on top of the sensors.

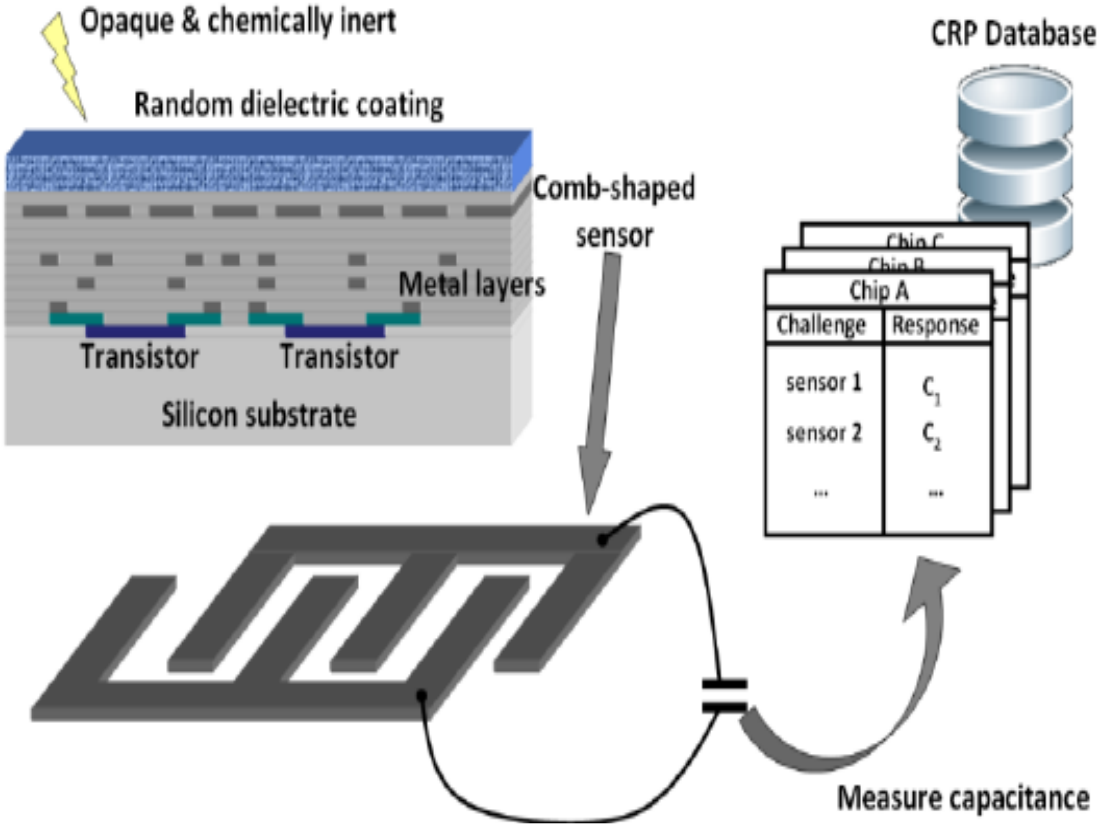

Figure 2-3: Coating Physical Unclonable Function [53]

2.2.3 Ring Oscillator PUF

Ring Oscillator Physical Unclonable Functions (ROPUFs) are silicon-based PUFs that produce a random response [5]. Ring Oscillator PUF (RO-PUF) design relies on delay loops, which can be produced using an odd number of ring oscillators; moreover, due to existing manufacturing variations for each ring oscillator, a unique set of frequencies will be provided. Therefore, the output bit will be generated by the random selection of a ring oscillator pair using multiplexers and compared. As shown in Figure 2-4, it is challenging to predict or clone the extracting unique signatures produced by a challenge-response mechanism for ROPUF design.

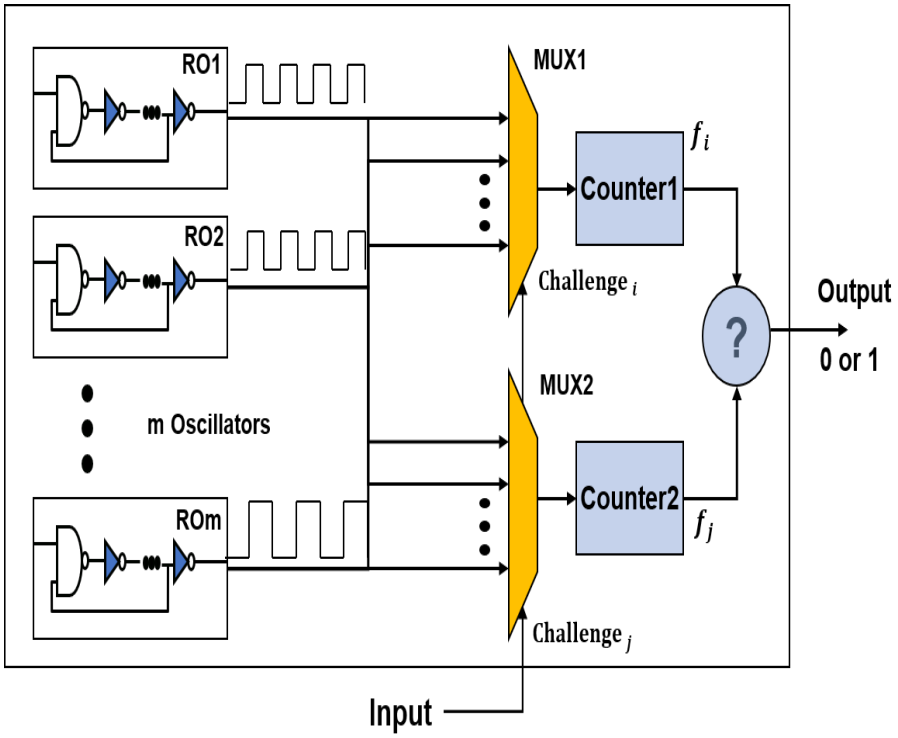

Figure 2-4: Ring Oscillator PUF Circuit.

In order to obtain and produce “0” or “1” output for the ROPUF circuit, a pair of ROs that are mapped at different locations of the chip produces different frequencies ($f_a \; and \; f_b$); these two frequencies ($f_a \; and \; f_b$) must be compared and the higher frequency the will determined the out bit. The measured process variations should be unique for each ROPUF. A response bit (r_{ab}) is produced by the significant comparison of these two frequencies ($f_a \; and \; f_b$) using a simple comparison method as follows in equation 2.1:

$$r_{ab} = \begin{cases} 1, & if \; f_a \; > \; f_b \, , \\ 0, & otherwise. \end{cases} \tag{2.1}$$

2.2.4 Arbiter PUF

In [49], Lee et al. introduced the basic structure of the Arbiter PUF, which is producing a sort of race among two delay paths with an arbiter at the end. An Arbiter PUF generates binary keys using the manufacturing process variation of integrated circuits. In APUF, a rising edge signal travels through two paths simultaneously. Due to process variation, one path travels quicker than the other and generates binary keys as shown in Figure 2-5. The Challenge bits consist of K external bits (C1= b1.b2…..bk) for K number of stages. For challenge bits C1, C2, Cn we get a response of R1, R2… Rn. A D-latch flip-flop is used as an arbiter to determine which response signal reaches the arbiter first. The arbiter is constructed at the end of the design with one input connected to the clock and the other connected to the input. The arbiter is used to select the faster signal that is the one that reaches the arbiter first. From the design of the Arbiter PUF, researchers have developed different models.

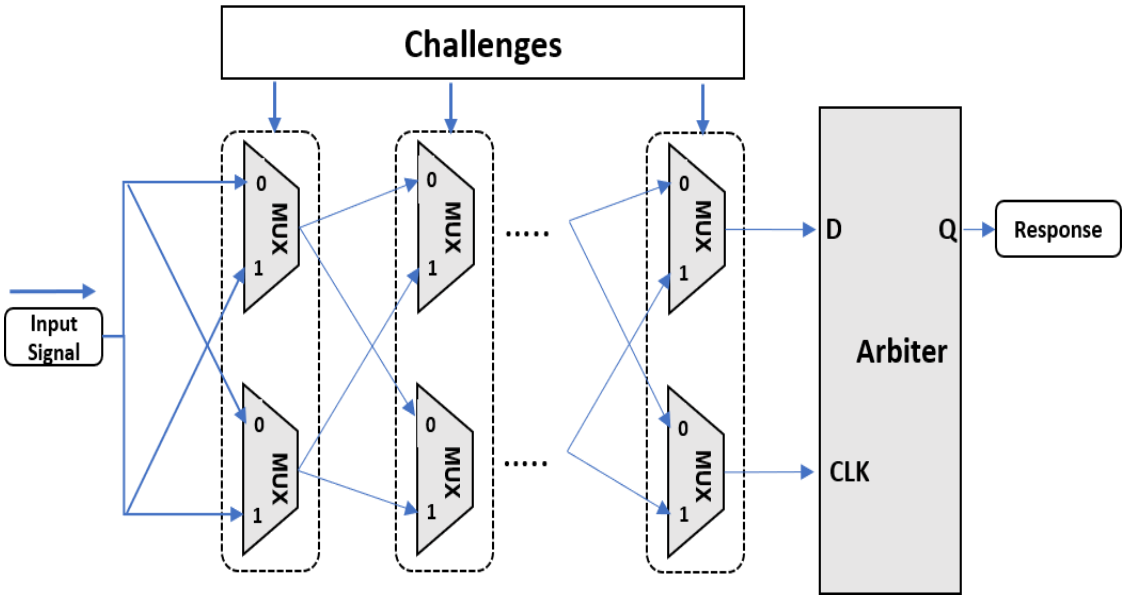

Figure 2-5: An Arbiter PUF Delay Design.

2.2.5 Butterfly PUF

The first structure of the BPUF was introduced in [50]. The butterfly structure displays utilize logically stable and unstable states of a flip flop (latches). A BPUF consists of a pair of cross-coupled latches, flipflop memory cells, as shown in Figure 2-6. These latches can have unpredictable startup values, and the functioning of Butterfly PUF does not require power up the device. The BPUF exploits and operates the random assignment of a stable state with the use of clear/preset functionality of latches. The circuit settles to one of the stable states, which is done by holding one latch in preset while holding the other in clear mode by an excitation signal. The final state is determined by the physical mismatch between the latches and the cross-coupling interconnect, deciding which stable state the PUF will settle on.

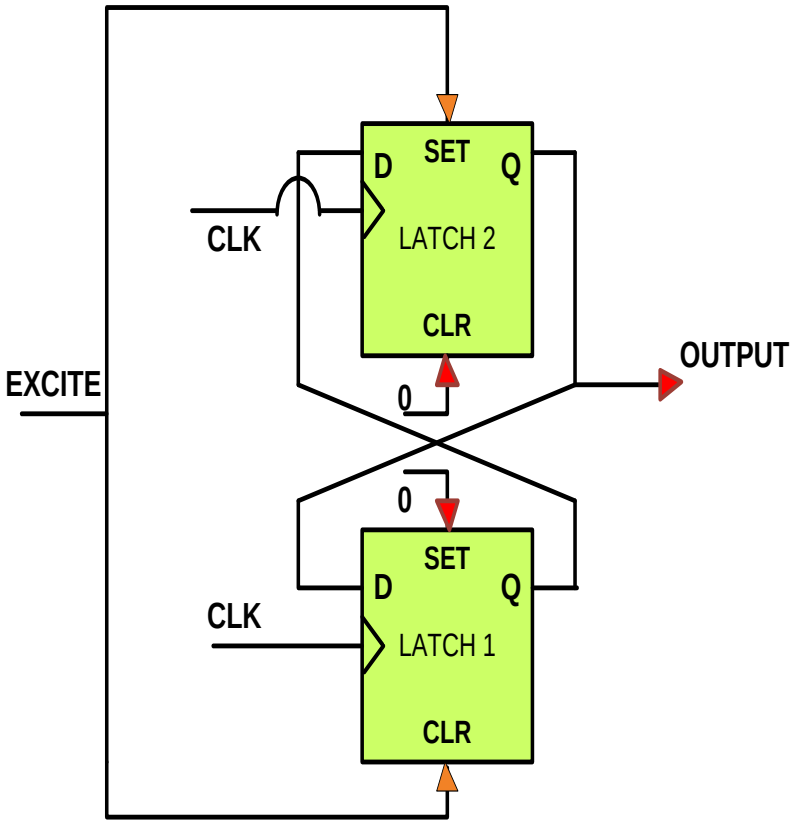

Figure 2-6: Butterfly Physical Unclonable Function (BPUF).

2.3 Challenge and Response Mechanism on PUFs

Physical Unclonable Functions (PUFs) are physical structures embodied on silicon chips that take advantage of delay characteristics of manufacturing process variations to extract chip-unique secret keys for cryptographic applications. As shown in Figure 2-7, a PUF acts as a function which applies an n-bit challenge and produces an k-bit response.

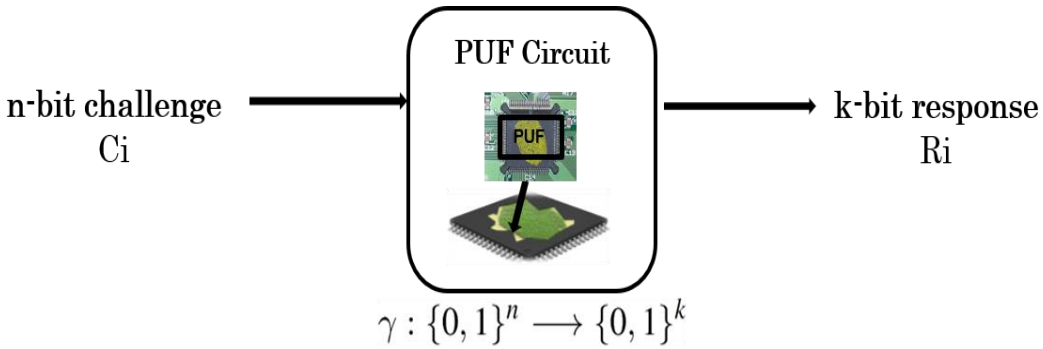

Figure 2-7: Challenge-Response Pairs of a PUF design.

Even though the same PUF design is replicated on n-chips, because of manufacturing variations, PUFs generate unique and random responses that vary from chip to chip. Hence, challenge-response mapping is unique to each chip and can be uniquely authenticated. Randomly selected challenge and response pairs (CRPs) are generated for each chip and stored by trusted parties in a secure database system. These CRPs are regenerated later and compared to the stored one in order to authenticate the identity of these chips. Device authentications are done by applying some of the stored challenges against chips under tests to generate CRPs. The generated CRPs are compared with the stored CRPs in the database. Chips can only be considered as authentic if the generated CRPs are identical to the stored CRPs.

2.4 Modeling Attacks and Security

The general use of programable platforms in different application levels, aerospace, image processing, communication, etc., has raised the issue of their security. The attackers may launch any attacks and steal the original data in various methods. Most of the fabrication places are outside the country where the design has been mapped to reduce the

manufacturing process cost. With the help of hardware Trojans, the attacker can steal the data, threatening personal and national security. The hardware attacks are extensive and can be categorized into different Levels such as invasive, non-invasive attacks, and Machine Learning attacks [47,54,55]; the attackers may try to launch any of these attacks.

2.4.1 Invasive Hardware Attacks

Invasive hardware attacks on a silicon chip are expensive because of equipment cost, knowledgeable attackers, and time. These attacks cause tamper proof of the attack to obtain unauthorized access to an electronic system. There are many types of Invasive hardware attacks such as Overbuilding, Cloning, Physical Tampering Attacks, and Reverse Engineering.

2.4.1.1 Reverse Engineering

Reverse engineering is one example of invasive attacks known as reverse or back engineering attacks to obtain the initial design from the design results. An attacker typically tampers silicon chips to extract information and understand their functionality to perform reverse engineering. Third parties can obtain the results of the existing logic design by cycling all the possible inputs and analyzing results through an exhaustive test bench which can be difficult and time-consuming.

2.4.1.2 Overbuilding

Overbuilding is a design theft that is a severe issue because it is a risky invasive attack. The contractor can do overbuilding manufacturing chips and builds more than the

required number, which is a breach of contract. Overbuilt chips' performance is the same as the original chips because they are precisely similar in functionality and build, making it difficult to follow in the market.

2.4.1.3 Cloning

Cloning is the process of replicating the original design function and making a model of it. The design cloning may lack the essential characteristics of the original design due to a shortage of understanding of the functionality. The replicated design saves the third party all the associated costs because there is no development expense involved in the cloning, and therefore they can make a significant profit. These counterfeit cloned chips are a substantial threat that can produce competing products to develop effective countermeasures.

2.4.1.4 Physical Tampering Attacks

Physical Tampering Attacks attempt to gain unauthorized access to an electronic system by an offender. The attacker tries to access stored cryptographic keys to perform potential physical tampering with malicious goals. Tampering Attacks might be performed through different techniques, such as reverse engineering and Spoofing. However, an adversary can tamper with a silicon chip and store its secret keys to extract the operation details of the design.

2.4.2 Non-Invasive Hardware Attacks

Non-invasive attacks like side-channel attacks refer to the information provided by

implementing cryptographic hardware. Power consumption and time delays are examples of side-channel analysis. The leakage of information provides hints that are useful to an attacker. Whenever a hardware fault exists, it will lead to a constructive processing error, and a diversity of attacks exist.

2.4.3 Machine Learning Attacks

Machine learning attacks are the most common attacks recently used to emulate the PUF's behavior. A software-based design can imitate the challenge-response pair's relationship of a hardware-based PUF design against many evolving hardware security primitives. The assumption is that an adversary can illegally obtain some of the CRPs to get additional details regarding the remaining CRPs to clone the behavior of a specific PUF structure using different algorithms, including support vector machine, genetic algorithm, logistic regression, and PAC. Figure 2-8

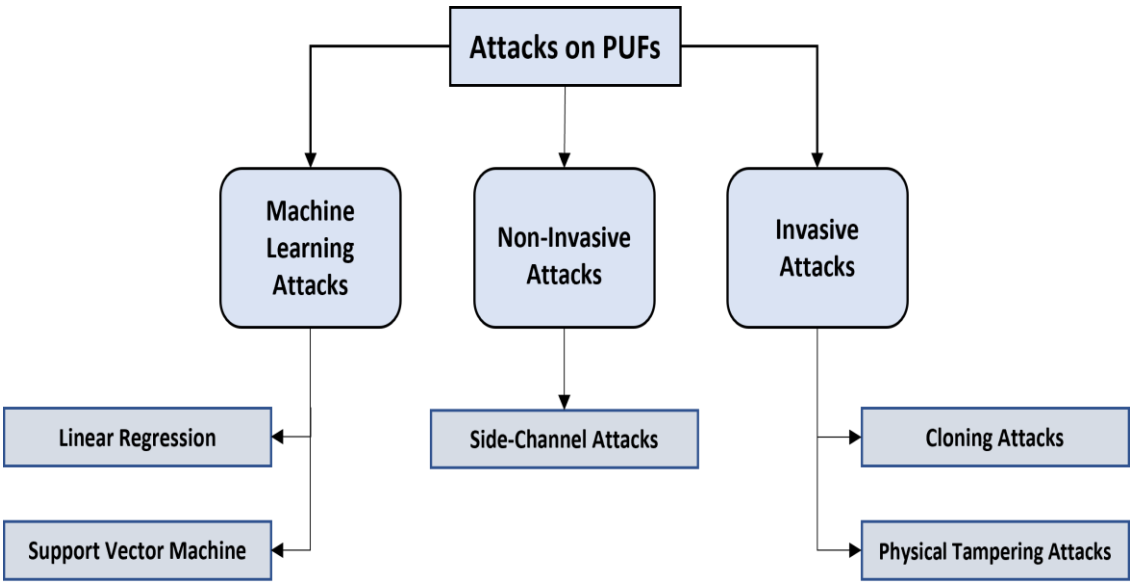

Figure 2-8: Attacks on PUF

2.5 Data Collection and Preprocessing

Data collection and preprocessing involve different techniques to convert the raw data into an application framework or structure before processing with a learning algorithm. These techniques generally have an impact on machine learning implementation. Data collection is essential and involves everything from data assembling to cleaning and preprocessing. Data cleaning eliminates the duplicative observations from the dataset, consisting of replication records or features. The performance of machine learning involves different steps in identifying the problem evaluation. Taking care of missing data and cleaning the data is an important step, and imputation replaces missing data with substituted values. Typically, it is undesirable to have any missing data in our data set for the simple reason that it can cause some errors when training the machinery model, and therefore it must be handled. If the dataset is not ready or unavailable for ML implementation, the data collection needs to be accomplished first.

2.6 Feature Selection Algorithms

Feature selection is an essential attribute for datasets collection, and a subset of original features can accurately represent a problem. Feature selection can identify this subset and the most appropriate features for the problem. This implementation applies prior learning of the dependent and independent parameters of the problem to choose the smallest subset of features that contain the most predictive capability to facilitate the analysis and enhance the performance of the algorithms. For some machinery models, avoid some features being dominated by others so that the model does not even consider

the dominated features. The usage of feature scaling will allow putting all our features on the same scale to optimize the accuracy of your model predictions. The machinery models can readily work with the numerical values; therefore, any nominal features should convert to numerical features. Moreover, all the features transform to the same range of values using scalers to help most machinery models consider the entire features. Different feature selection algorithms were used on the dataset to extract the informative and expected features:

2.6.1 Principal Component Analysis (PCA)

Principal component analysis (PCA) is a technique that analyzes a dataset for reducing its dimensionality, which means decreasing the complexity of the model. The technique forms new uncorrelated variables that maximize variance and simplify the model while keeping relevance and performance [56]. Sometimes the datasets have hundreds of features; therefore, extracting fewer independent variables explains the variance and simplifies the model interpretability. PCA extracts the important information from the dataset represents it as a set of new principal components, and the analysis involves extracting linear composites of variables. PCA maximizes the total variance to examine different patterns [57].

2.6.2 Linear Discriminant Analysis (LDA)

The linear discriminant analysis (LDA) is a primary data analysis technique proposed initially by R. Fisher [58]. Fisher's linear discriminant analysis is a conception to find a linear combination of features that divides classes of objects or possibilities. LDA is

often used in object recognition for feature extraction; it lowers the dimensional space by maximizing the distance between the projected means of the classes and finds an optimal linear transformation that can be used for classification and dimensionality reduction of the provided features [59]. Linear Discriminant Analysis (LDA) defines a subspace of lower dimensions where data points of the actual problem are separable; therefore, variance in any separate data set guarantees maximal separability, which allows for fast and massive processing of data samples [60]. LDA is not an interdependence technique; the measurements are made on independent variables for each observation. The comparable method is discriminant correspondence computation when handling categorical independent variables [61].

2.6.3 Kernel PCA

Kernel principal component analysis (kernel PCA) is a nonlinear extension of PCA that extracts features efficiently in the dataset using the nonlinear feature extraction method [62]. Kernel PCA is associated with kernel function in the feature space to explain the dataset into a more increased dimensional space [63]. kernel PCA feature extractor can be developed in the feature space by computing all the kernel functions in advance and then implementing feature extraction using these kernel functions [64].

Chapter 3

Analysis of Swarm Intelligence based ANN Algorithms for Attacking PUFs

In this chapter, Artificial Neural Networks (ANNs) using swarm intelligence-based modeling attacks are used against different silicon-based PUFs to test their resiliency to these attacks. Amongst the swarm intelligence algorithms, the Dragonfly Algorithm (DA), Gravitational Search Algorithm (GSA), Cuckoo Search Algorithm (CS), Particle Swarm Optimizer (PSO), and the Grey Wolf Optimizer (GWO) are used. The attacks are extensively performed on different types of PUFs. Also, this chapter presents a brief review and explains the basic concept of swarm intelligence techniques that mimic the behavior of the natural collective system.

3.1 Introduction

Swarm Intelligence (SI) is a cooperative system based on a group of agents that achieve a common goal by cooperating according to their behavior and system organization. The fundamental concept behind swarm intelligence techniques is the replication of the behavior of the natural collective system [65]. Amongst the many available swarm intelligence algorithms, Dragonfly Algorithm (DA), Gravitational Search

Algorithm (GSA), Cuckoo Search Algorithm (CS), Particle Swarm Optimization (PSO), and the Grey Wolf Optimizer (GWO) algorithms are frequently used. These algorithms, in general, are simple and computationally efficient. Specifically, these algorithms have higher viability, robustness, stability, and search efficiency, and have a fast convergence rate [66]. Moreover, these algorithms are able to optimize a vast search space with a fixed size population to solve different complex design optimization problems. DA, GSA, CS, PSO, and GWO algorithms are discussed next.

3.2 Dragonfly Optimization Algorithm

The Dragonfly Algorithm (DA) is an Intelligent Swarm Optimization Algorithm [67]. The algorithm is modeled on the behavior of dragonflies for hunting and emigration. Hunting swarm behavior is a structure of a small group of dragonflies moving nearby and abruptly changing the steps, which is considered as static swarm behavior. The dragonfly's lifecycle includes two main milestones: nymph and adult. Migratory swarm behavior is a massive number of dragonflies flying in one direction over long distances. It is also known as dynamic swarming, as showing in Figure 3-1 [68]. Dynamic swarms and Static swarms represent the exploitation and exploration capabilities of the Dragonfly Algorithm. The behavior of dragonflies follows the principles of separation, alignment, cohesion, distraction from the enemies, and attraction towards the food. Similarly, the swarm movement of the dragonfly is determined by five different operators: separation, alignment, cohesion, attraction towards food sources, and distraction towards enemy sources [68].

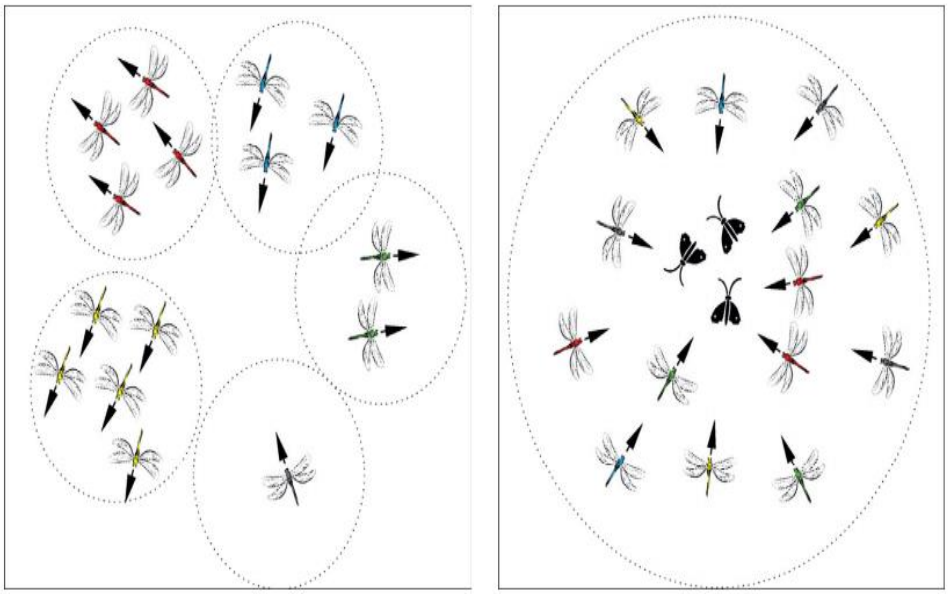

Figure 3-1: (a) Dynamic swarming, (b) Static swarming.

3.2.1 DA Algorithm

The mathematical implementation of DA for updating the position and velocity of dragonfly separation, alignment, and cohesion coefficients are calculated using equations (3.1-3.3) [68]

$$S_i = -\sum_{j=1}^{N} X - Xj \tag{3.1}$$

$$A_i = \frac{\sum_{j=1}^{N} Vj}{N} \tag{3.2}$$

$$C_i = \frac{\sum_{j=1}^{N} Xj}{N} - X \tag{3.3}$$

where, X_j and V_j correspond to the position and velocity of each individual. X corresponds to the position of the current individual, and N denotes the number of neighborhoods individuals. Attraction towards food source F_i and distraction from enemies E_i are calculated using equations (3.4) and (3.5)

$$F_i = X^+ - X \tag{3.4}$$

$$E_i = X^- + X \tag{3.5}$$

where X is the position of the current individual and X^+ denotes the food source and X^-denotes the enemy source. The step vector shows the direction of the movement of the dragonflies and is defined with the following equations (3.6) and (3.7)

$$\Delta X_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\Delta X_t \tag{3.6}$$

where the coefficient weights are separation (s), alignment (a), cohesion (c), food (f), enemy (e), inertia weight (w) and $(S_i, A_{i,} C_i, F_i, E_i)$ represent the i-th individual coefficient weights.

$$X_{t+1} = X_t + \Delta X_{t+1} \tag{3.7}$$

If there is no dragonfly in the neighborhood radius, the position of the dragonfly is updated using Levy Flight equation [69] as given in equation (3.8). This improves the randomness, chaotic behavior, and global search capability of dragonflies.

$$X_{t+1} = X_t + Levy\ (d)X_t \tag{3.8}$$

The fitness function is then evaluated based on the updated position and velocities. The position updating process is continued until the stop condition is met.

The algorithm shown below represents the Dragonfly optimization (DA) pseudo code [68]:

Algorithm 1: Dragonfly Optimization Algorithm

1. Initialize the population of the Dragonflies **Xi.**
2. Initialize the step vectors **ΔXi.**
3. **while** *t < Max of Iterations do*
4. Calculate the fitness values of dragonflies.
5. Update $\mathbf{F_i}$ and $\mathbf{E_i}$
6. Update **w**, **s**, **a**, **c**, **f**, and **e**
7. Calculate **S**, **A**, **C**, **F**, and **E**
8. Update neighboring radius.
9. **if** a dragonfly has at least one neighboring dragonfly
10. Update velocity
11. Update position
12. **else**
13. Update position
14. **end if**
15. Check and correct the new positions based on the
16. boundaries of variables
17. **t=t+1**
18. **end while**

3.3 Gravitational Search Algorithm

Gravitational Search Algorithm (GSA) is a swarm optimization technique proposed by Rashedi et al. based on gravity concepts and different masses' interaction [29]. In this algorithm, the solutions of different agents' populations interact with one another via the

theory of Newtonian gravity force and the laws of motion. The solution's performance is measured by different masses. Due to the force of gravity, the masses are dragged towards each other, which creates a global movement of all objects approaching the objects with heavier masses. The exploration step is when the mass moves towards a heavier mass, and the exploitation is when heavier masses move slowly. Accordingly, each mass can convey information with different masses and see their situation through the gravitational force.

3.3.1 GSA Algorithm

The mass's position compares itself to a problem's solution; then, the best solution is achieved with the heavier agent. The initial population is generated randomly, and the position of the agents are defined as:

$$X_i = (x_i^1, \dots, x_i^d, \dots, x_i^n) \qquad \text{for } i = 1, 2, \dots, N \tag{3.9}$$

where x_i^d presents the position of i^{th} agent in the d^{th} dimension. The gravitational search algorithm sets the initial value of the constant G:

$$G(t) = G_0 e^{-\propto t/T} \tag{3.10}$$

where G_0 and $\propto$ is initialized at the beginning of the iteration and T is the total number of iterations. The agents update the velocity and the position according to these equations:

$$v_i(t+1) = rand_i \times v_i(t) + a_i(t) \tag{3.11}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{3.12}$$

where $rand_i$ is a uniform random variable in the interval [0, 1].

This random number is used to give a randomized characteristic to the search. The total force acting on agent i at iteration t, was calculated as follows:

$$F_i^d(t) = \sum_{j \in K_{best,} j \neq i} rand_j F_{ij}^d(t) \tag{3.13}$$

Where K_{best} represents the set of k agents with best fitness and biggest mass. The pseudo code for the GSA is shown below [29]:

Algorithm 2: Gravitational Search Algorithm

---

1. Objective function $f(x), x = (x_{1,} x_{2,} \dots x_{d,})^T$
2. Initialize the population of n agents x_i
3. **while** *t < Max of Iterations* **do**
4. Evaluate the fitness for each agent
5. **for** each searching
6. Update the $G(t)$, $best(t)$,$worst(t)$and $M_i(t)$
7. **end for**
8. Calculation of the total force in different directions.
9. Calculation of acceleration and velocity.
10. Updating agents' position.
11. **t=t+1**
12. **end while**
13. Return the best solution

3.4 Cuckoo Search Algorithm

Cuckoo search algorithm (CS) is a nature-inspired optimization algorithm proposed by Yang in 2009 to solve optimization problems based on the cuckoo bird's breeding behavior search approach of laying its eggs in the best host nest [38]. The CS algorithm is

based on the brood parasitism of cuckoo birds. The species lay their eggs in other host bird nests to be brooded by the proxy mother bird and use the host bird assistance to hatch their eggs. The hatching probability of similar eggs to the host bird's eggs is high. In some cases, the other bird recognizes the different eggs, so they throw the eggs away, destroy them, and even leave their nests to build another one in a distinct location. The CS algorithm uses better solutions to substitute not-so-good solutions in the nests. As a result, it can enhance search capabilities to improve the relationship between exploration and exploitation.

3.4.1 CS Algorithm

The CS algorithm is performed through the following three rules: first, each cuckoo bird chooses a random nest to lay only one egg; second, the best nests with a good quality of eggs will carry over for the next population; third, a host bird can detect a different egg with a probability of pa ∈ [0, 1] for a constant number of available host nests. Hence, the host bird may either throw the different eggs or leave the nest and build a new one. One of the essential CS features is Lévy flights to generate new candidate solutions rather than a simple random walk. The following Lévy flight is performed to generate new solutions $x^{(t+1)}$ for the i^{th} cuckoo:

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus \text{Lévy } (\lambda) \tag{3.14}$$

where α > 0 is the step size. The product ⊕ means entry-wise multiplications. The Lévy step size probability distribution is represented by:

$$\text{Lévy} \sim u = t^{-\lambda}, \quad (1 < \lambda \leq 3) \tag{3.15}$$

which has an infinite variance with an infinite mean. The pseudo-code of CS algorithm [38]:

Algorithm 3: Cuckoo Search Algorithm

1. Objective function $f(x), x = (x_{1,} x_{2,} \dots x_{d,})^T$
2. Initialize the population of n host nests x_i
3. **while** *t < Max of Iterations* **do**
4. Get a cuckoo randomly by Lévy flights
5. Evaluate its fitness f_i
6. Randomly choose n nest f_j
7. **If** ($f_i > f_j$)
8. Replace j by the new solution
9. **End if**
10. Abandon a fraction of p_a of worse nests and build new ones at new locations via Lévy flights
11. Keep the best solutions
12. Rank the solutions and find the current best
13. **t=t+1**
14. **end while**
15. Return the best solution

3.5 Particle Swarm Optimization

In 1995, Eberhart and Kennedy proposed the Particle Swarm Optimization (PSO), which mimics social behavior and search techniques of a swarm of animals, or a flock of birds, or a school of fish, when they adapt their environment to search for their food [36]. The particles communicate and share their information to find the optimum path to reach its food sources. The shortest path followed is the particle's best position. Based

on the current positions of the local and global positions in the search space, each particle identifies and updates its position until the global-optimum position is achieved. Figure 3.2 shows how the particle changes its position within the search space to obtain food [70].

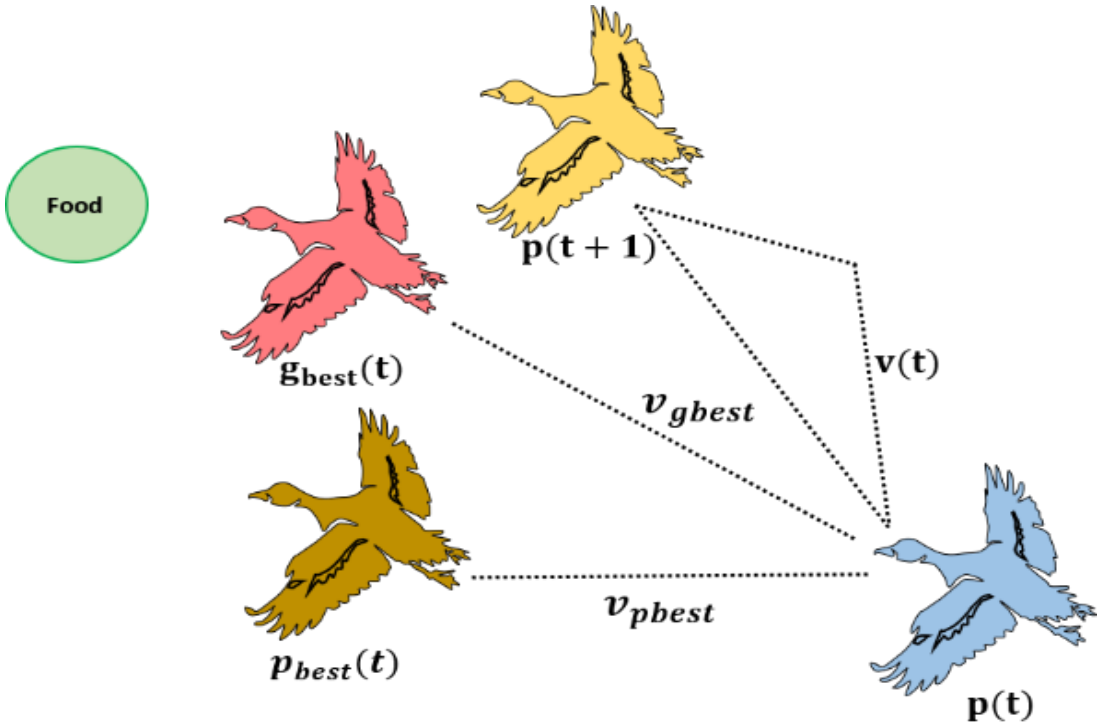

Figure 3-2: Update Particle position and velocity in the search space.

3.5.1 PSO Algorithm

Particle movements affect all other individuals within the group. Each one of them has its position and velocity defined by equation 3-16, which presents the best position achieved with respect to all neighbor's best position.

$$p(t+1) = p(t) + v(t+1) \tag{3.16}$$

Here p(t+1) denotes the updated location of the particle in the swarm, g_{best} defines the global best, p(t) represents the current location of the particle in the swarm, and v(t+1) is the new velocity of the particle in the swarm based on the location of the g_{best}. Based on the current velocity and position of each particle, its own best position pbest and the entire population's best position gbest, the particle's new velocity and position can be determined as:

$$v(t+1) = \omega * v\,(t) + c_1 * r_1 * [p_{best}(t) - p(t)] + c_2 * r_2 * [g_{best}(t) - p(t)] \qquad (3.17)$$

where p_{best} is the best position of the particle, g_{best} is the best position of the swarm, v(t) is the current velocity, and r1, r2 are random numbers from uniform distribution. Both c1, c2 are acceleration coefficients and ω is the inertia weight. The pseudo-code of the Particle Swarm Optimization (PSO) algorithm is shown below [70]:

Algorithm 4: Particle Swarm Optimization Algorithm

---

1. Initialization Particle's Position
2. Initialization Particle's velocity
3. Calculate the fitness values of each particle
4. **while** *t < Max of Iterations* **do**
5. Update the position according to Equation 3.16
6. Update the velocity according to Equation 3.17
7. Choose the particle having the best fitness value as the g-best
8. Compare P-best of each particle with g-best of swarm
9. **t=t+1**
10. **end while**
11. Return **g-best particle**

3.6 Grey Wolf Optimization

In 2014 Mirjalili et al. introduced Grey Wolf Optimizer (GWO), which is an algorithm that explains Grey Wolf's hierarchical hunting-mechanism based on how wolves obey a strict social hierarchy [39]. This pattern maintains the stability and assists other wolves during the hunt. The complete wolf pack must follow the orders of the wolf with the most durability and fighting ability. Figure 3-3 shows the classification of the social hierarchy in a grey wolf pack levels consists of the alpha (α), beta (β), delta (δ) and omega (ω) wolves:

1) Alpha (α): The leader of the pack, at the top of the hierarchy, is mostly responsible for making decisions because it considered the most qualified wolf among the pack.

2) Beta (β): The adviser wolf at the second level in the hierarchy, which helps the alpha in decision-making or other pack activities. A beta follows the leader's directions to maintain discipline over the pack.

3) Delta (δ): Stand at the third level in the hierarchy, Delta follows the orders of alpha and beta, but dominate and lead the omegas.

4) Omega (ω): The lowest level in the grey wolf social hierarchy, omega wolves, always follow the commands of all the other dominant wolves in the social hierarchy.

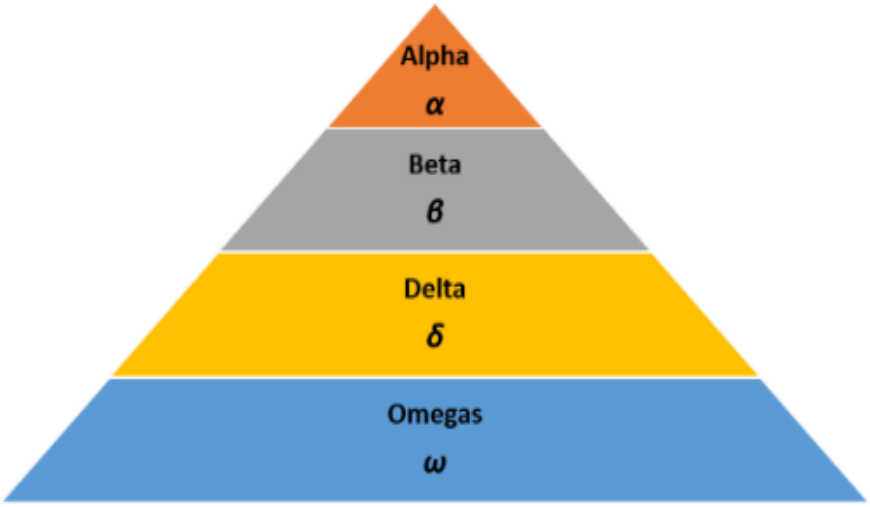

Figure 3-3: Grey Wolf Social Hierarchy.

3.6.1 GWO Algorithm

The hunting behavior of the GWO algorithm is guided by the three wolves α, β, and δ, while the ω wolves follow them. Figure 3-4 illustrates how the position can be updated in the search space for the three wolves. Alpha is the closest location in the search space to prey X_α, which is considered as the first best wolf, X_β is the second-best location for beta wolf, and delta is the third best wolf location X_δ. The rest of the pack, omega wolves, will update their positions according to alpha, beta, and delta positions. The locations of wolves are updated as follows:

$$\vec{D} = \left|\vec{C}.\ \overrightarrow{X_P}(t) - \vec{X}(t)\right| \tag{3.18}$$

$$\vec{X}(t+1) = \overrightarrow{X_P}(t) - \vec{A}\,.\vec{D} \tag{3.19}$$

$$\vec{A} = 2\vec{a}\,.\vec{r}_1 - \vec{a} \tag{3.20}$$

$$\vec{C} = 2\,.\ \vec{r}_2 \tag{3.21}$$

where t represents iteration, $\vec{A}$ and $\vec{C}$ are coefficient vector, $\overrightarrow{X_P}$ is the prey position vector, $\vec{X}$ is the wolf positions, $\vec{a}$ is the linear coefficient, $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ are random vectors located in the scope [0, 1]. The calculation of distances between the position of current individual and individual of alpha, beta, and delta are:

$$\vec{D}_\alpha = \left|\vec{C}_1.\ \vec{X}_\alpha - \vec{X}\right| \tag{3.22}$$

$$\vec{D}_\beta = \left|\vec{C}_2.\ \vec{X}_\beta - \vec{X}\right| \tag{3.23}$$

$$\vec{D}_\delta = \left|\vec{C}_3.\ \vec{X}_\delta - \vec{X}\right| \tag{2.24}$$

where $\vec{X}_\alpha$, $\vec{X}_\beta$, $\vec{X}_\delta$ are the position vector, $\vec{C}_1$, $\vec{C}_2$, $\vec{C}_3$ are randomly generated vectors, $\vec{X}$ represents the position vector of current individual. Therefore, the mathematical models for grey wolf hunting are calculated by:

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 . \vec{D}_\alpha \tag{3.25}$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 . \vec{D}_\beta \tag{3.26}$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 . \vec{D}_\delta \tag{3.27}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{3.28}$$

where $\vec{A}_1$, $\vec{A}_2$, $\vec{A}_3$ are randomly generated vectors.

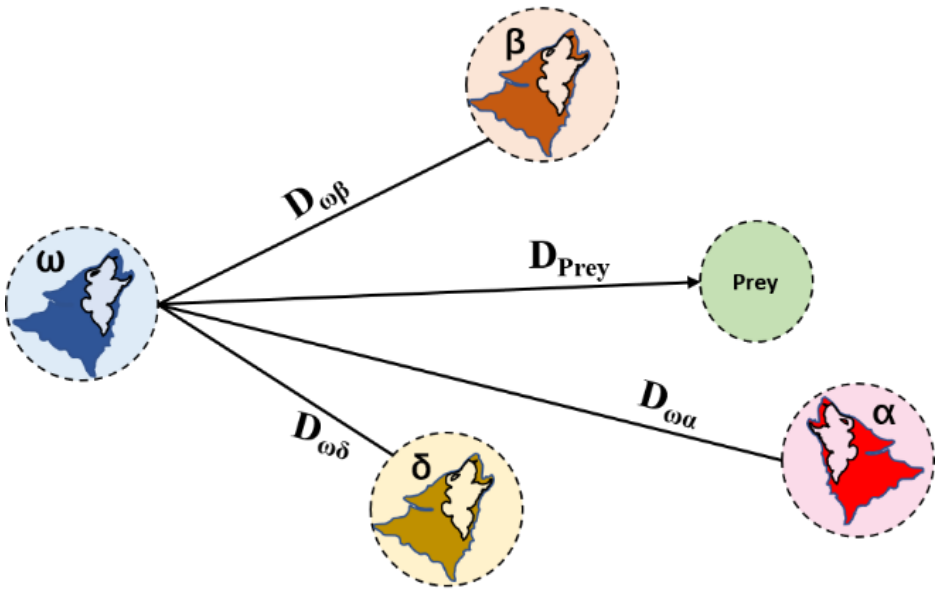

Figure 3-4: Wolves Position surrounding the prey.

The pseudo code for the GWO is shown below [39].

Algorithm 5: Grey Wolf Optimization Algorithm

---

1. Initialize the population of the Grey Wolves
2. Initialize for **a**, $\mathbf{A}$, and $\mathbf{C}$
3. Calculate the fitness values of each wolf $\mathbf{X}_{\alpha}$, $\mathbf{X}_{\beta}$, and $\mathbf{X}_{\delta}$
4. **while** *t < Max of Iterations* **do**
5. **for** each searching wolf
6. Update position
7. **end for**
8. Update **a**, $\mathbf{A}$, and $\mathbf{C}$
9. Calculate the fitness values of all wolves
10. Update the positions of $\mathbf{X}_{\alpha}$, $\mathbf{X}_{\beta}$, and $\mathbf{X}_{\delta}$
11. **t=t+1**
12. **end while**
13. Return $\mathbf{X}_{\alpha}$

3.7 Proposed Method

During modeling the vulnerability of the PUFs, researchers have used many techniques such as Fault injection-based modeling attacks, Genetic Algorithm, Genetic Programming, and Evolutionary Strategies to model PUF CRPs. From their results, it has been observed that these algorithms require a large number of CRPs to model PUF characteristics; moreover, the attacker may need to have physical access to the PUFs. The motivation for choosing swarm intelligence algorithms is that SI algorithms have fewer parameters than evolutionary methods to adjust which makes them flexible, robust, and distributive. SI algorithms are easy to implement, more reliable for finding solutions to many complex problems, and converge faster than other algorithms.

Also, SI optimizers maintain a large search space of candidate information throughout the iterations. Furthermore, the mathematical model's implementation mechanism is very well developed to avoid local optimization and improve performance, making it easier to combine with practical engineering problems. The Swarm Optimization algorithms are used to build ANN models to analyze the vulnerability of the different PUFs described earlier for modeling attacks. These training algorithms adjust the weights and biases of the ANN until the highest response prediction accuracy can be obtained by finding the optimum set of weights and biases. Based on the objective (loss/error) function for the SI algorithms, the weights are adjusted in each iteration in order to minimize the loss/error function that is used in neural networks to minimize the training error.

The Mean Square Error function (MSE), which is the most commonly used parameter for the evaluation of the neural network, is defined in equation (3.29).

$$Mean\_Square\_Error = \frac{1}{n}\sum_{i=1}^{n}\left(Y_{exp} - Y_{obs_i}\right)^2 \tag{3.29}$$

where the performance of the network is evaluated based on the difference between the predicted responses (Y_{obs_i}) and the actual responses (Y_{exp}). The average MSE obtained from all training samples is based on the best solutions of the previous iteration. While the current weights and biases of the neural network are updated, the MSE gradually decreases; therefore, after enough iterations, the algorithms can achieve the best solution. For a challenge vector $C = [C_1, C_2, \dots, C_m]$ of size m, Configurable Ring Oscillator, Ring Oscillator PUF, Inverter based ring oscillator, and the Arbiter PUF will generate a response vector $R = [r_1, r_2, \dots, r_m]$. The CRPs are fed to the ANN network, where each bit of the challenge vector represents one neuron, and the response bit is the outcome of the neural network. For modeling of the PUFs, it is assumed that an attacker gets hold of a small set of CRPs $(C, r) = [(C_1, r_1), (C_2, r_2) \dots, (C_m, r_m)]$, which can be modeled by the ANN-based models using swarm optimization DA, GSA, CS, PSO and GWO algorithms to predict the remaining set of CRPs. Algorithm 6 outlines the steps of how the model trains the Artificial Neural Network based on Swarm Intelligence Algorithms.

Algorithm 6: Training ANN using SI Algorithms

1. Initialize all the parameters of Swarm Algorithm
2. Constricting of ANN learning structure
3. Initialize the weights and biases of the Neural Network
4. **while** *t < Max of Iterations* **do**
5. Map the Challenges Vector $C=[C_1,C_2,\ldots,C_m]$ into the input layer of ANN
6. Calculate the predicted response R
7. Compare the predicted response R to the actual response R
8. Calculate the new global optimum value of weights and biases using swarm algorithm
9. Update the weights and biases of the ANN using Swarm Intelligence optimizes
10. **t=t+1**
11. **end while**
12. **Return** weights, biases, and predicted accuracy

3.8 Computational Complexity Analysis

The computational complexity of the proposed algorithm (Algorithm 6) accounts for the execution time of the algorithm based on its structure. For Algorithm 6, the computational complexity for each step can be described as follows:

- The computational complexity of initialization of the weights and biases is O(N × dim) time, where N represents the population size and dim represents the dimension of the problem.
- In step 5 (mapping the challenge vector) of the proposed algorithm, for each iteration, the challenge vector is mapped into the input layer of ANN with constant computational complexity of O(1); the iteration loop technically runs in O(Iter); therefore, the final time complexity for mapping is O(Iter).
- The calculation of predicting the response and comparing it with the actual response in steps 6 and 7 have a computational complexity of O(Iter × L), where L is the total number of training CRPs.
- For step 8, in each iteration the computational complexity represents the calculation of the new global optimum value of weights and biases O(Iter × N).
- Step 9 represents the updated weights and biases values with computational complexity of O(Iter × N × dim).
- Since the total number of iterations is not more than IterMax, the total time complexity is: O(IterMax) + O(IterMax × L) + O(IterMax × N) +O(IterMax × N × dim) + O(dim × N).
- As seen from the above analysis, the proposed algorithm's computational complexity depends on the size of population (N), dimension (dim), and iterations (Iter).

3.9 Experimental results analysis of swarm intelligence based model attacks

To analyze the vulnerability of the various PUFs to ANN based attacks using different Swarm Intelligence algorithms, a subset of the randomly chosen CRPs is used as the training set. An accuracy score evaluates the attack resistance in terms of the percentage of successful response predictions [71]. The Swarm Intelligence algorithms used to train the ANN network to predict PUF CRPs are implemented using Python 3.5 (64 Bit) frameworks. For training the CRPs, a 2.3 GHz PC with 16 GB RAM and 2GB Graphics card is used. The response prediction accuracy is determined by using cross-validation of ten blocks K-fold method [72]. One of these ten partitions is used as the test set, while the other nine cumulatively serve as the training set. The ANN learning network structure used in the experiment is a 3- Multi-Layer Perceptron (MLP) with 33 nodes in the hidden layer. It is observed that the ANN method dramatically improves the learning rate for the different PUFs. Different parameters and hyperparameters used in ANN for training and prediction of the CRPs are listed in Table 3.1.

Table 3.1: Parameters values used.

Parameters	Values
Number of training CRPs	30000
Number of k folds	10
Loss Functions	MSE
Number of nodes in hidden layer	33
Hidden Layer Activation Function	ReLU
Output Layer Activation Function	Sigmoid
Learning rate	0.01

In this chapter, we describe how the Swarm Intelligence algorithms are used to train the ANN. However, for these algorithms to reach their maximum performance and achieve the best results, proper settings of the initial parameters are required. The parameters chosen for SI algorithms to simulate the DA, GSA, CS, PSO, and GWO algorithms are selected based on references [29], [38], [39], [70], and are given in Table 3.2. ANN-based models are trained for 1000 iterations and the algorithms are tested with an initial population of individuals in the range of 5-150. However, no improvement in prediction accuracy is achieved by increasing the number of individuals to more than 100; therefore, the number of individuals is kept at 100.

Table 3.2: Initial parameters set in Swarm Algorithms.

Algorithm	Parameter	Default Value
DA	Upper DA Value	100
	Lower DA Value	-100
	Iterations	1000
	Number of Agents	100
	Dimension	595
CS	Detection probability (p_a)	0.25
	Step length control	0.01
	Number of iterations	1000
	Number of bird nests	100
	Dimension	595
GSA	Initial gravitational constant (G_0)	100
	Constant values initialization (α)	20
	Number of iterations	1000
	Population Size	100
	Dimension	595
PSO	Cognitive influence (C1)	2
	Social influence (C2)	2
	Inertia weight (ω)	[0.2, 0.9]
	Number of iterations	1000
	Number of particles	100
	Dimension	595
GWO	Decreases linearly ($\vec{a}$)	[2, 0]
	Vector contains random values ($\vec{A}$)	[-2$\vec{a}$, 2$\vec{a}$]
	Vector contains random values ($\vec{C}$)	[0, 2]
	Number of iterations	1000
	Number of wolfs	100
	Dimension	595

Table 3.3 lists experimental results for the prediction accuracy of the different PUFs against the DA, GSA, CS, PSO, and GWO Swarm Intelligence algorithms attacks using a different number of challenge-response pairs 10K and 30K CRPs [71,73]. From the table, it is evident that the PUF structures are vulnerable to Swarm Intelligence-based model attacks with prediction accuracies ranging from 71.1% - 99.3%. The maximum prediction accuracies obtained by Swarm Intelligence-based model attacks is 99.3% for configurable PUF using GWO based model. Also, it is found that the prediction accuracies are much better for all the PUFs when we increase the number of CRPs for training the different models. In the case of 10K CRPs, the maximum prediction accuracy was 88.3%, but when we increase the number of the CRPs up to 30K, the maximum prediction accuracy is 99.3.3%

Table 3.3: Swarm Intelligence-based Prediction Accuracy for Different PUFs.

Number of CRPs	**10,000**					**30,000**				
Type of PUF	**DA %**	**GSA %**	**CS %**	**PSO %**	**GWO %**	**DA %**	**GSA %**	**CS %**	**PSO %**	**GWO %**
ROPUF	79.6	78.6	79.9	80.1	81.9	91.5	92.6	93.9	94.3	95.9
Configurable ROPUF	85.7	85.3	85.2	86.5	88.3	94.4	95.9	96.5	97.4	99.3
Inverter ROPUF	71.9	71.5	72.0	73.5	75.0	89.6	90.2	90.7	92.2	93.7
Arbiter PUF	73.1	71.1	73.3	74.2	76.1	90.1	91.1	91.8	92.3	94.1

Figures 3-5 – 3-8 show plots of the prediction accuracies versus the number of iterations for the different PUFs models, respectively. It can be concluded from these plots that the prediction accuracy of the GWO algorithm performance is higher than the other algorithms. Also, the plots show that the GWO converges fast. Figure 3-9 shows the loss function of the different swarm-based algorithms. It is observed from the figure that the GWO converges faster than the other algorithms.

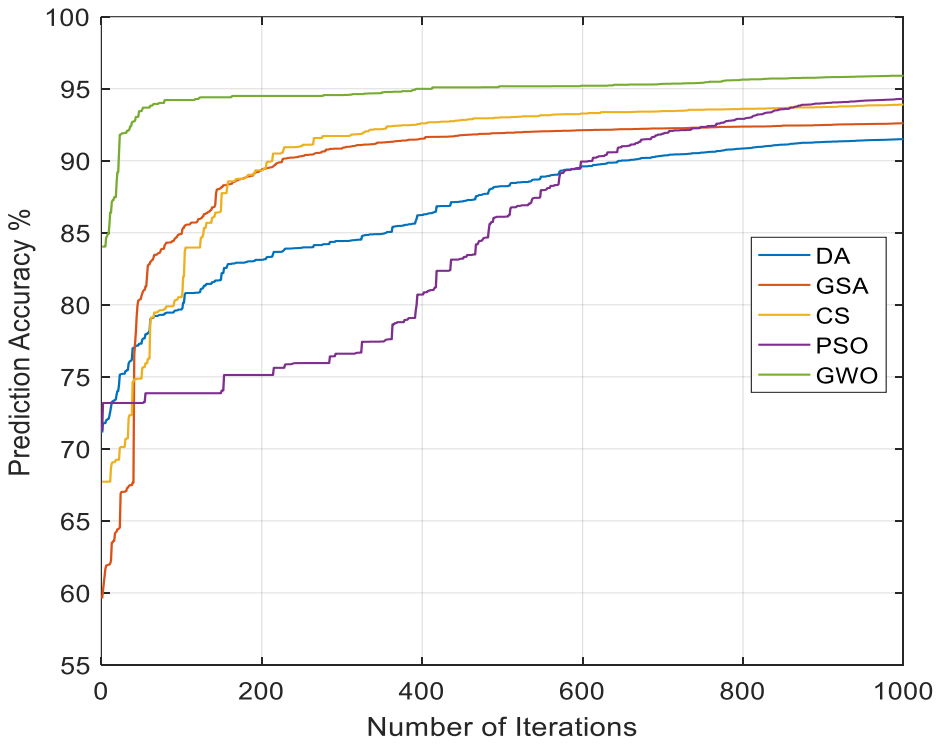

Figure 3-5: Swarm-based prediction accuracy vs Iteration numbers for ROPUF.

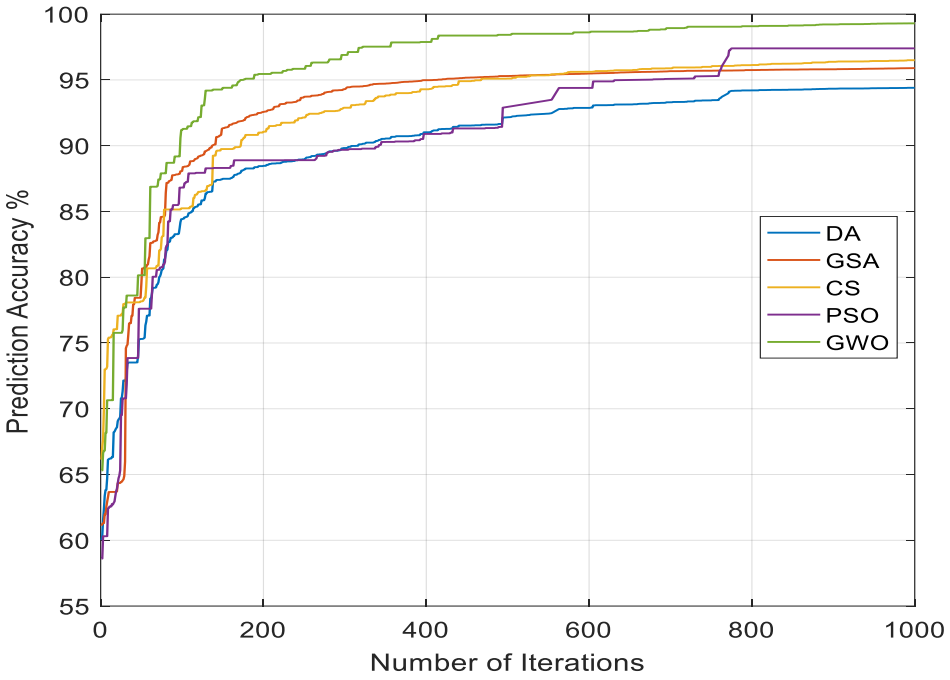

Figure 3-6: Swarm -based prediction accuracy vs Iteration for Configurable ROPUF.

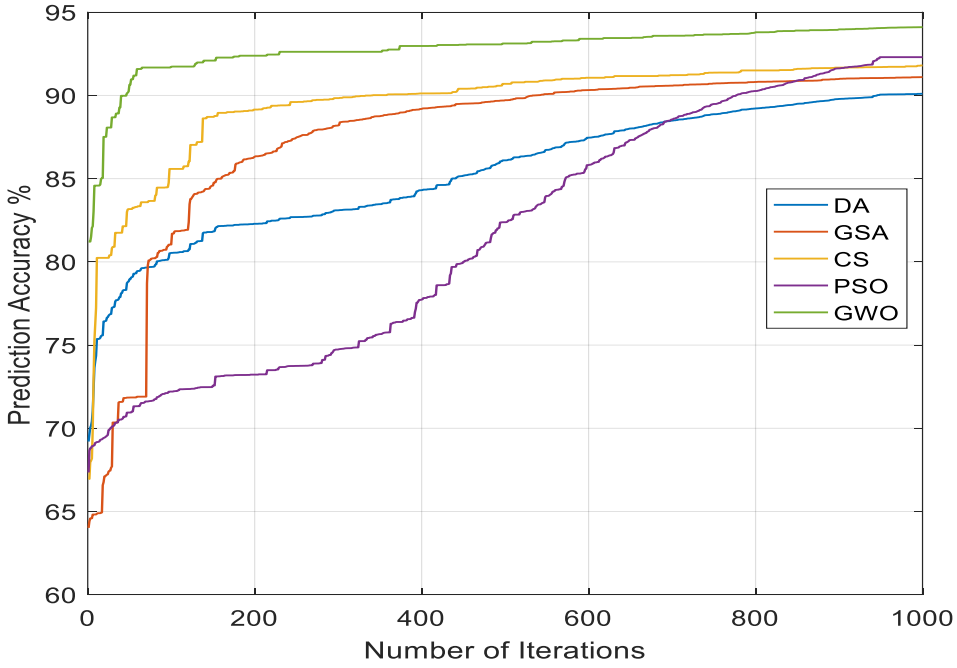

Figure 3-7: Swarm -based prediction accuracy vs Iteration numbers for Arbiter PUF.

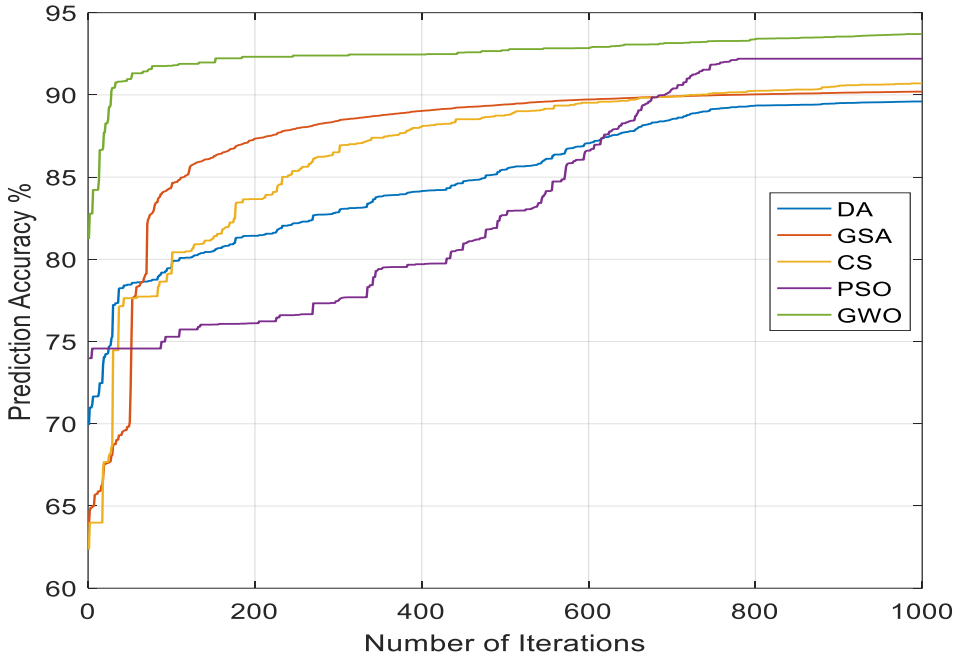

Figure 3-8: Swarm -based prediction accuracy vs Iteration numbers for Inverter ROPUF.

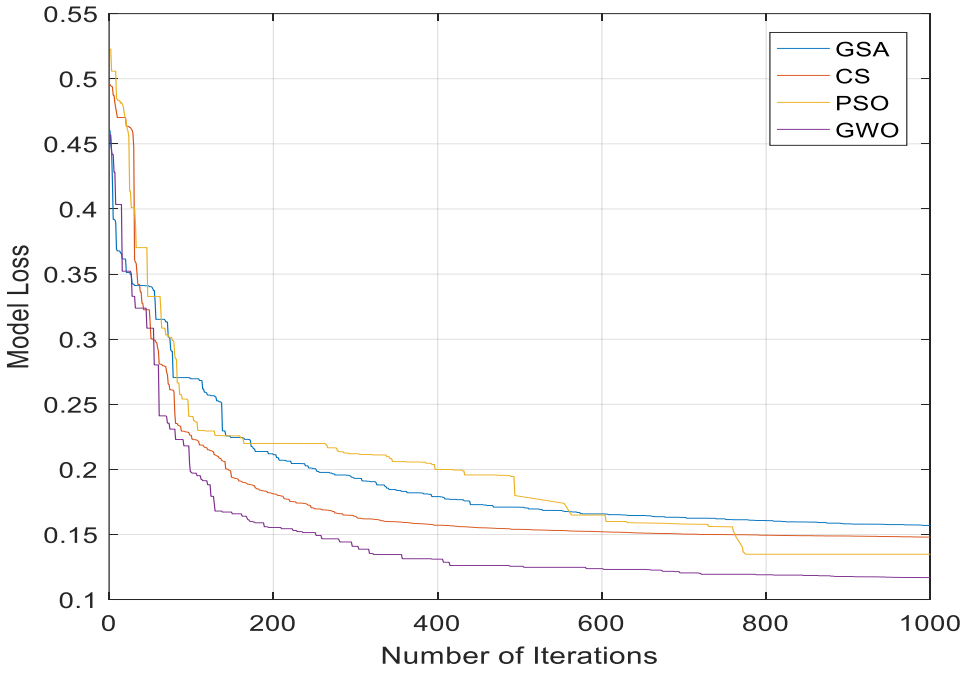

Figure 3-9: Loss function vs number of Iteration for different swarm algorithms

3.10 Chapter Summary

In this chapter, we have introduced a novel implementation of Artificial Neural Network-based modeling attack on various PUFs using different swarm intelligence algorithms. The Swarm Optimization algorithms are used to build ANN models to analyze the vulnerability of the different PUFs for modeling attacks. These training algorithms adjust the weights and biases of the ANN until the highest response prediction accuracy can be obtained by finding the optimum set of weights and biases. Based on the objective function for the SI algorithms, the weights are adjusted in each iteration in order to minimize the loss/error function that is used in neural networks to minimize the training error. From the results, it is observed that the swarm intelligence algorithms produce response prediction accuracy range from (71.1% - 99.3%). Amongst the SI algorithms, the GWO algorithm performs the best in predicting the CRPs. It is observed that the Configurable ROPUF is the most vulnerable and its response can be predicted with an accuracy of 99.3% when the GWO is used. To the best of our knowledge, swarm-based algorithms have not been investigated earlier to test the security of PUFs.

Chapter 4

Machine Learning and Artificial Neural Network Model Attacks on PUFs

In this chapter, different Machine Learning classifiers (ML) and Artificial Neural Networks (ANNs) based modeling attacks are used against different silicon-based PUFs to test and analyze their vulnerability to these attacks. Amongst the Machine Learning classifiers, the Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), K-Nearest Neighbor (KNN), kernel Support Vector Machine (Kernel SVM), and Naive Bayes Classifier (NB) are used. The attacks are extensively performed on different types of PUFs. Also, in this chapter, an ANN-based model is trained with four other optimizers, namely, RMSprop, Adadelta, Adam, and Nadam, to study the vulnerability of the different PUFs to ML attacks.

4.1 Introduction

Several machine learning modeling attacks have been recently used to emulate the PUF's behavior so that a software-based design can imitate the challenge-response pair's relationship of a hardware-based PUF design [24]. Modeling attacks on PUFs have become a massive threat to the PUF based authentication system [74]. It is shown that if an

adversary can get hold of a small set of CRPs from thousands of CRPs generated from a PUF, it can build the PUF's model within a brief period [13]. The Probably Approximately Correct (PAC) learning attack has been performed successfully on many PUFs, including the ROPUF [75]. Researchers have proposed many different solutions in order to prevent modeling attacks. One solution is to alter the PUF's architecture by adding nonlinear elements [76]. The disadvantage of this approach is that it may reduce the reliability of the PUF. In another solution to model-building attacks, researchers have proposed obfuscation of challenge and response bits to hide the relationship between the CRPs [77]. In this approach, different subsets of the PUF response are presented to the verifier, which acts as a key to authenticate the whole response string. Also, researchers have proposed obfuscation of challenges and responses for lightweight authentication for PUF based pervasive devices [78].

4.2 Logistic Regression

Logistic Regression is a popular statistical and supervised machine learning algorithm usually performed on classification problems, its basic uses a logistic function to model a binary dependent variable. For a given challenge $a_1,\ a_2, \ldots a_n$, a single bit response r_i ε (0,1) can be predicted using the associated probability $P(a, r|$ w) where w is the associated weight in each bit of challenge [79]:

$$P(a, r|W) = r * sig(f) + (1 - r)\big(1 - sig(f)\big) \tag{4.1}$$

Where, $sig(f)$ is the output of the network and w is the associated weight in each bit of the challenge.

4.3 Decision Tree

Decision Tree (DT) is a supervised learning algorithm also used for classification. The formation of a decision tree is similar to the structure of a tree. During each split, the input attributes are decided to predict the output, and after many iterative steps of recursion, a decision tree is ready for predicting the output [80]. For a challenge vector $a_1,\ a_2, \ldots a_n$, a single bit response $r_i\ \varepsilon$ (0,1) can be predicted using the decision tree.

4.4 Random Forest Classifier

Random Forest Algorithm (RF) is quite similar to the decision tree algorithm due to the fact it consists of a vast number of different decision trees that are working together as one model. Every individual tree in this algorithm gives out a class prediction, and the most voted tree becomes the final model's prediction [81].

4.5 K Nearest Neighbor

K-Nearest Neighbor (KNN) is considered as the most frequently used and most straightforward classifier algorithm. It can be used for binary classification. Moreover, it is a learning model and a regressive method for pattern recognition. KNN is commonly used in cases where there is no prior knowledge of the distribution of the dataset; it uses the entire dataset during the training phase. The model trains k number of neighbors; more neighbors mean a higher chance to find correct predictions [82]. In the case of ROPUF, the output is either 0 or 1, which makes the value of k=2.

4.6 Support Vector Machine

The main purpose of using a support vector machine is to specify a suitable hyperplane separating both the classes and increasing the margin between classes or the closest point. The hyperplane should have the most expansive achievable distance between the input vectors of both classes with minimal distance to the separating hyperplane [83]. The learning of the support vector machines is dependent mainly on the inner mapping function known as kernels. In a linear kernel, the classification problem is linearly separable, and a linear hyperplane is used for identifying the datasets in two classes using the function:

$$f(v,u) = z^T v + b \tag{4.2}$$

where z is the weight associated with the input features and b is the bias. For an input set of $a_1,\ a_2, \ldots a_n$, a single bit response r_i ε (0,1) can be predicted using support vector machine with linear kernel. The radial basis function is non-linear (Gaussian). It maps the case models to higher-dimensional spaces, managing and estimating a non-linear relationship between the class labels.

$$f(v,u) = \exp\frac{-|v-u|^2}{2\sigma^2} \tag{4.3}$$

where $'\sigma^{2'}$ is the tuning parameter.

4.7 Naive Bayes Classifier

Naive Bayes is a nonlinear learning algorithm. However, the formula in the Naive Bayes theorem is only true when $P(data)$ is different from zero. The prediction boundary separates the non-linearly distributed observations [84].

$$P\ (class|data) = \frac{P(data|class)*P(class)}{P(data)} \quad (4.4)$$

where, $P(class)$ is Prior Probability, $P(data)$ is Prior Probability of Predictor, $P(data|class)$ is Likelihood, $P\ (class|data)$ is Posterior Probability.

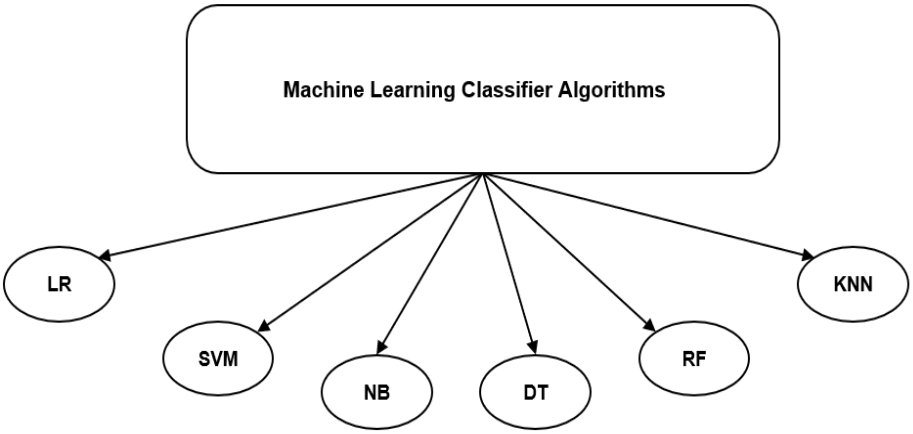

Figure 4-1: Machine Learning algorithms used for predicting PUF CRPs.

4.8 Experimental results of ML classifier modeling attacks

For a challenge vector $C = [C_1, C_2, \dots, C_m]$ of size n, a PUF generates a single bit response vector $R = [r_1, r_2, \dots, r_m]$. For modeling the PUFs, the assumption is that an attacker gets a hold of CRP pairs $(C, r) = [(C_1, r_1), (C_2, r_2) \dots, (C_m, r_m)]$ and tries to model the behavior of those CRPs by using machine learning algorithms to know the remaining set of CRPs. In this research, Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), K-Nearest Neighbor (KNN), kernel Support Vector (Kernel SVM), Naive Bayes Classifier (NB) classifiers are used to study the vulnerability of the different PUFs to machine learning attacks [85]. Figure 4-1 shows the different ML classifiers that are used. Keras framework is used for modeling the PUF's CRPs with Theano and Tensorflow

[86]. For training the CRPs, a 2.3 GHz PC with 16 GB RAM and 2GB Graphics card is used. The Prediction accuracy of the ML classifier algorithm varies for different PUFs. The results obtained from different ML algorithms for different PUFs models are shown in Table 4-1. The prediction accuracy is determined by using cross-validation of ten blocks K-fold method, one of these ten partitions is used as the test set, while the other nine cumulatively serve as the training set. The attacks are extensively performed on different types of PUFs: Configurable Ring Oscillator, Ring Oscillator PUF, Inverter Ring Oscillator, Arbiter PUF. From Table 4.1, it can be concluded that KNN performs better than all other ML classifiers in predicting the CRPs with a prediction accuracy of 92.3% compared to accuracy for the different algorithms. As the model can train more CRPs, the prediction accuracy response increases. The maximum prediction accuracy using 10K CRPs was 82.5 %, which increases up to 92.3 % by increasing the number of CRPs to 30K.

Table 4.1: ML Classifiers Prediction Accuracy for Different PUFs.

Number of CRPs	**10,000**							**30,000**						
Type of PUF	**LR %**	**DT %**	**RF %**	**KNN %**	**SVM %**	**KSVM %**	**NB %**	**LR %**	**DT %**	**RF %**	**KNN %**	**SVM %**	**KSVM %**	**NB %**
ROPUF	69.3	71.2	73.5	76.2	74.1	75.3	67.4	80.9	82.8	85.1	87.2	85.6	86.8	79.0
Configurable ROPUF	78.7	80.1	81.3	82.5	80.9	81.7	75.9	88.9	90.3	91.5	92.3	91.1	91.9	86.1
Inverter ROPUF	64.3	66.2	67.3	69.6	67.8	68.1	60.3	81.7	83.6	84.7	87.2	85.2	85.5	77.7
Arbiter PUF	63.3	64.3	65.6	68.2	66.8	67.1	60.5	83.6	84.6	85.9	88.1	87.1	87.4	80.8

It can be observed that the performance of the algorithms is quite similar in nature, but some perform better than the other algorithms. The best accuracy observed for the algorithms LR, DT, RF, KNN, SVM, KSVM, and NB are 88.9%, 90.3%, 91.5%, 92.3%, 91.1%, 91.9%, and 86.1% respectively. Figure 4-2 represents the performance of different machine learning classifier algorithms predicting different PUFs designs' response behavior: Configurable Ring Oscillator, Ring Oscillator PUF, Inverter Ring Oscillator, Arbiter PUF. The figure shows that the KNN outperforms all the other algorithms with maximum prediction accuracy of 92.3% for Configurable Ring Oscillator PUF.

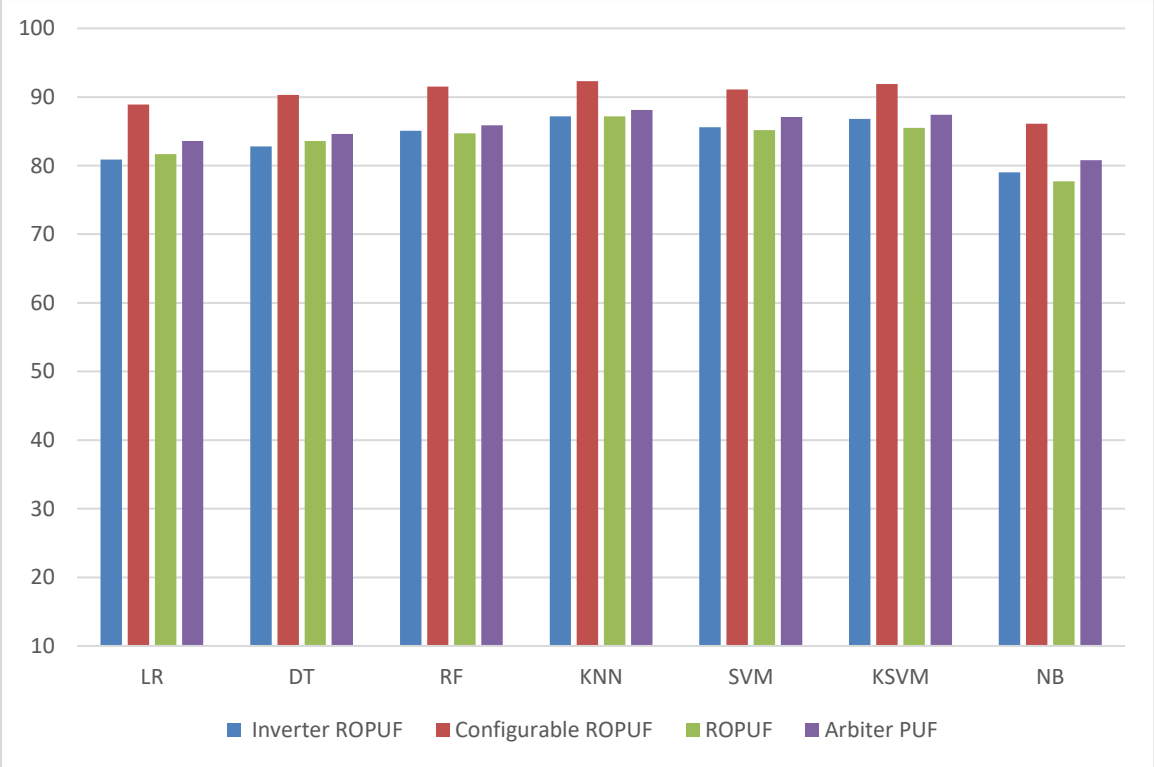

Figure 4-2: Comparison of prediction accuracies for different classifier algorithms.

4.9 Artificial Neural Network Models

An Artificial Neural Network (ANN) is a network structure of connected artificial neurons that can model complex relationships between inputs and outputs using computational and statistical data modeling tools. The neural networks consist of different layers termed as the input layer, output layer, and hidden layer. The first layer from where the network takes the input is known as an input layer, whereas the last layer of the network is termed as an output layer. The layers in between are termed as hidden layers. The number of hidden layers varies depending on the design [87]. The structure of the neural network is shown in Figure 4-3.

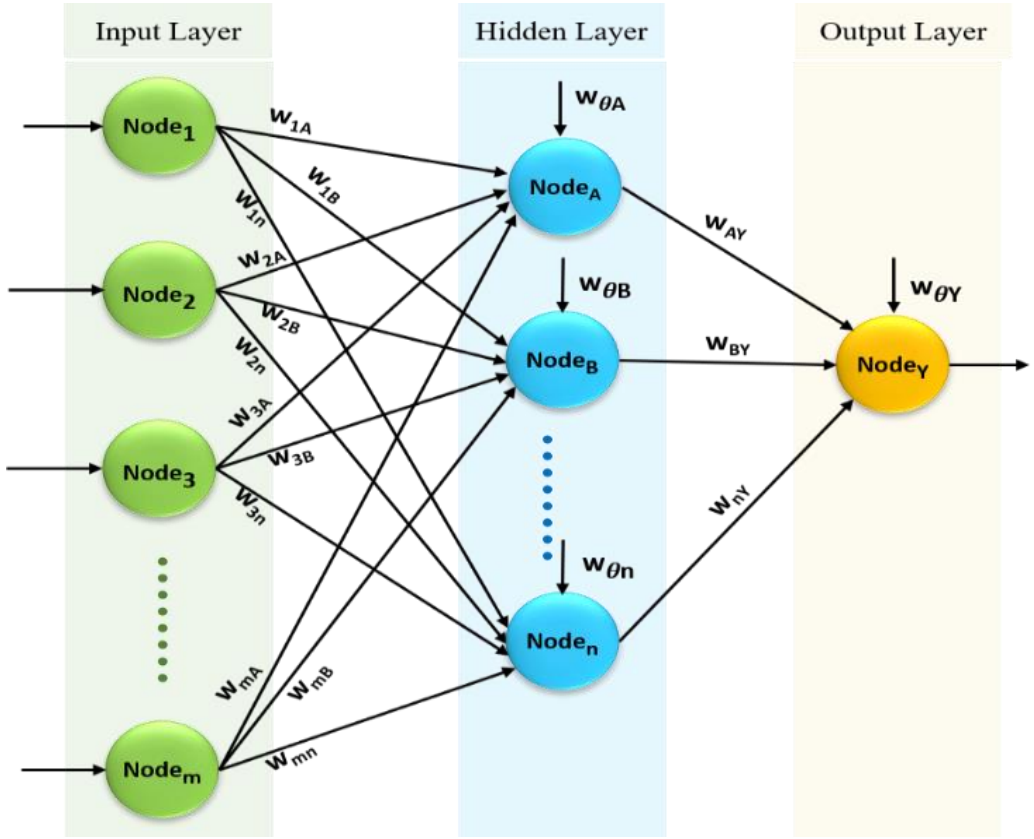

Figure 4-3: Artificial Neural Network Structure.

The input layer is connected and assigned a weight to the hidden layers. Similarly, the hidden layer is connected to the output layers; consequently, the output of any input layer act as an input of the next layer. For each node, weights are assigned and adjust based on the input-output relationship. The output of a 3-layer feed-forward neural network can be given by:

$$Y_j = b_j \sum_{i=1}^{3} W_{i,j}\, x_i \tag{4.5}$$

where Y_j is predicate output, b_j is base weights, $w_{i,j}$ is weights and x_i is input.

The input of a hidden layer is modified by some nonlinear function, sigmoid, which is the activation function:

$$Sigmoid(z) = \frac{1}{1+e^{-z}} \tag{4.6}$$

The weights are updated and calculated according to the difference obtained from model output and the actual output of the training data. This difference is calculated using loss or cost function, which will be minimized until the training loss is least or goes very close to zero. ANN-based models using four well-known optimization algorithms are used to perform attacks on different PUFs. These models are Resilient Backpropagation (RMSprop) optimization, Adadelta, Adam, and Nadam optimizations. The difference between the outputs is calculated using different cost functions. The model is trained until it reaches a point where the error is minimum. Four Multi-Layer Perceptron (MLP) models are designed as the ANN models in this research. All the four designed ANN models use different hyperparameters for their application on the datasets [88,89]. The first model uses resilient backpropagation (RMSprop) optimization as a training algorithm while the other models work on the Adadelta, Adam, and Nadam optimization, respectively. Resilient backpropagation algorithm works on the process of normalizing the magnitude of recent

gradients by dividing them with the average of the root mean squared gradients. Adadelta has an adaptive learning rate and an improvement for Adagrad optimization, as Adadelta reduces the aggressive, monotonically decreasing learning rates of Adagrad, thereby making it more efficient and faster. Adam is another method of adaptive learning rate in which it computes individual adaptive learning rates from the first and second moments of the gradients for updating the parameters [88]. Nadam optimization combines two optimizations together, which are Adam and NAG. Nadam is used for gradients with high curvature [89].

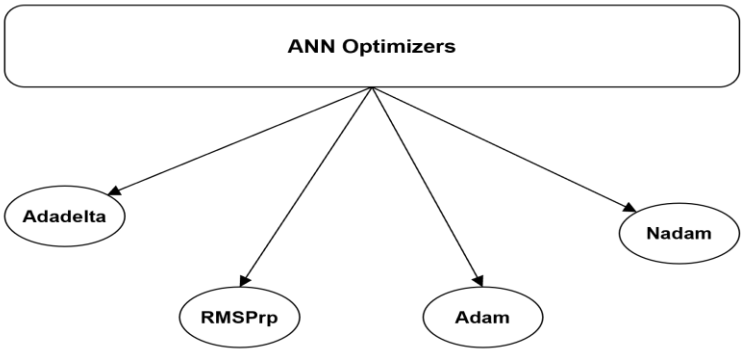

Figure 4-4: ANN-Based Algorithms used for Predicting PUF CRPs

4.10 Experimental results of ANN based modeling attacks

ANN-based models using four well-known optimization algorithms are used to perform attacks on different PUFs. These models are Resilient Backpropagation (RMSprop) optimization, Adadelta, Adam, and Nadam optimizations [71]. Figure 4-4 shows the different algorithms that are used in this work. The ANN learning network structure used in the experiment is a 3- Multi-Layer Perceptron (MLP) with 33 nodes in

the hidden layer. The ANN structure in terms of the number of hidden layers, nodes, and activation functions, the training conditions of the network-based model are given in Table 4.2.

Table 4.2: Initial parameters set in ANN optimizers

Algorithm	Parameter	Default Value
Adam	Alpha (α)	0.001
	Beta1 (β1)	0.9
	Beta2 (β2)	0.999
	Number of iterations	1000
RMSprop	Discounting factor (rho)	0.9
	Momentum	0.0
	Centered	False
	Number of iterations	1000
Nadam	Alpha (α)	0.001
	Epsilon (ε)	1e-08
	amsgrad	False
	Number of iterations	1000
Adadelta	Discounting factor (rho)	0.9
	Epsilon (ε)	1e-08
	**kwargs	clipvalue
	Number of iterations	1000

Table 4.3 represents the results obtained from ANN-based modeling for different optimization attacks on different PUFs. As shown in the Table, the accuracy range for response prediction is between 68.0% to 94.1%. It is also observed that the best accuracy for response prediction is 94.1% for the ANN-based modeling with Nadam on the Configurable ROPUF when we are using 30K number of CRPs.

Table 4.3: ANN-based Prediction Accuracy for Different PUFs.

Number of CRPs	**10,000**				**30,000**			
Type of PUF	**Adadelta %**	**RMSprop %**	**Adam %**	**Nadam %**	**Adadelta %**	**RMSprop %**	**Adam %**	**Nadam %**
ROPUF	75.8	77.1	78.3	79.4	87.7	89.1	90.3	91.4
Configurable ROPUF	83.8	84.4	85.0	84.1	92.7	93.0	93.5	94.1
Inverter ROPUF	68.0	69.2	70.3	70.0	87.0	88.7	89.5	90.3
Arbiter PUF	69.3	70.1	71.9	72.1	88.5	89.9	91.0	91.7

Figures 4-5, 4-6, 4-7, and 4-8 show the prediction accuracy graph. From the modeling of the PUFs, it is observed that, the models can train the data, they are able to predict the responses with high accuracy for different PUFs.

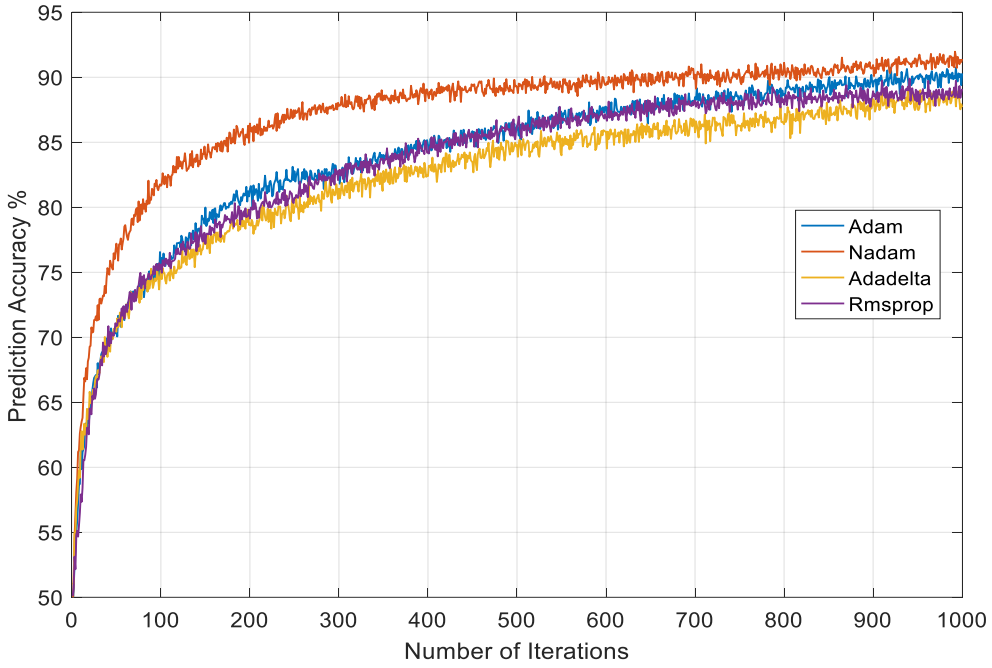

Figure 4-5: ANN-based prediction accuracy vs Iteration numbers for ROPUF.

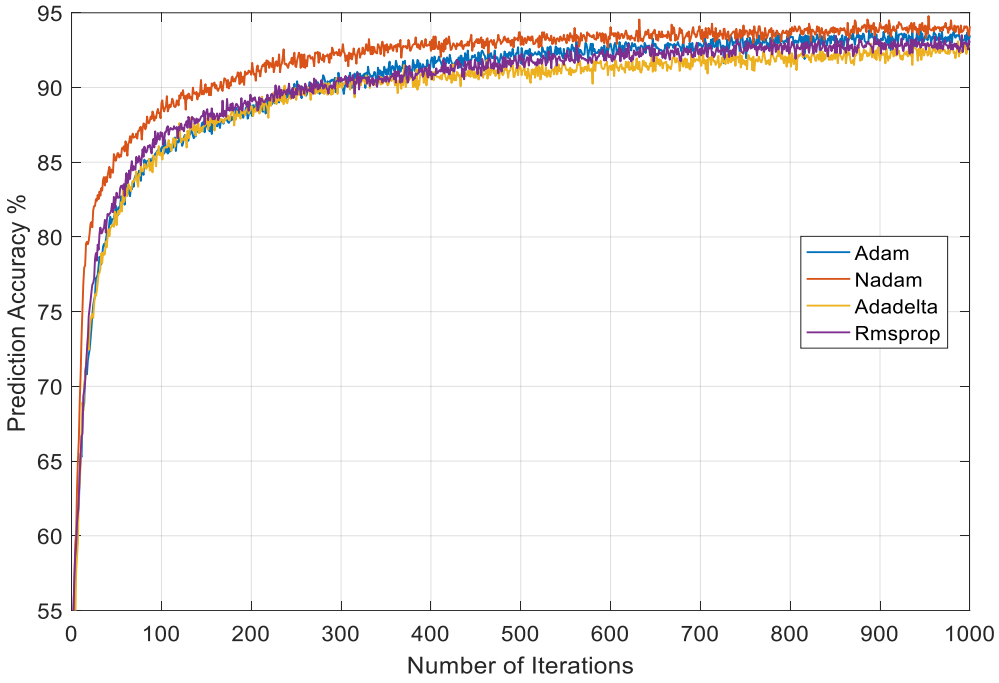

Figure 4-6: ANN-based prediction accuracy vs Iteration for Configurable ROPUF.

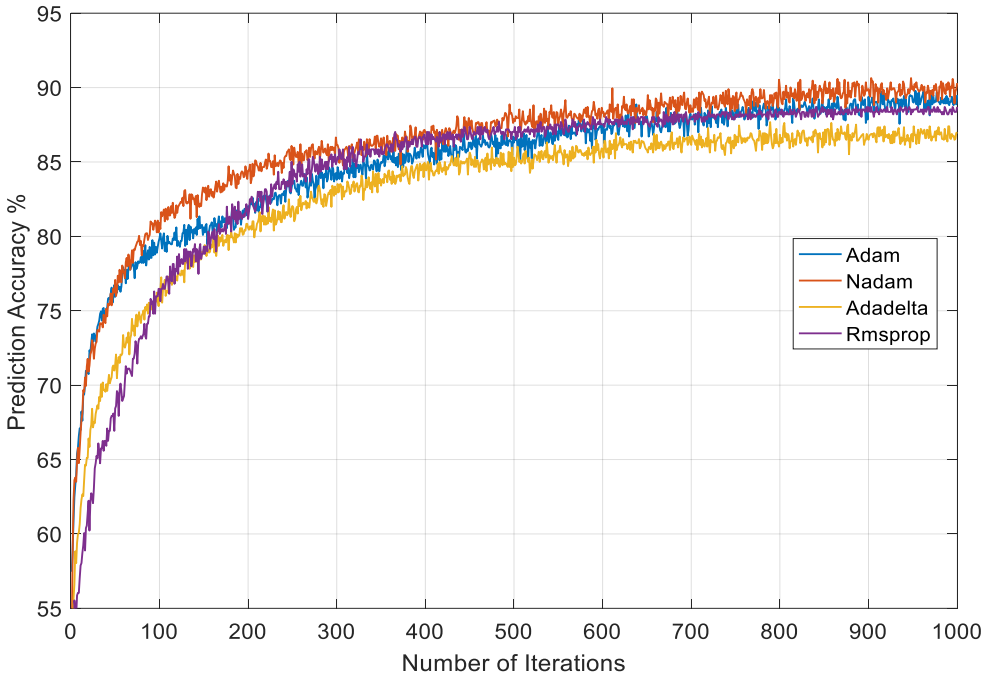

Figure 4-7: ANN-based prediction accuracy vs Iteration numbers for Inverter ROPUF.

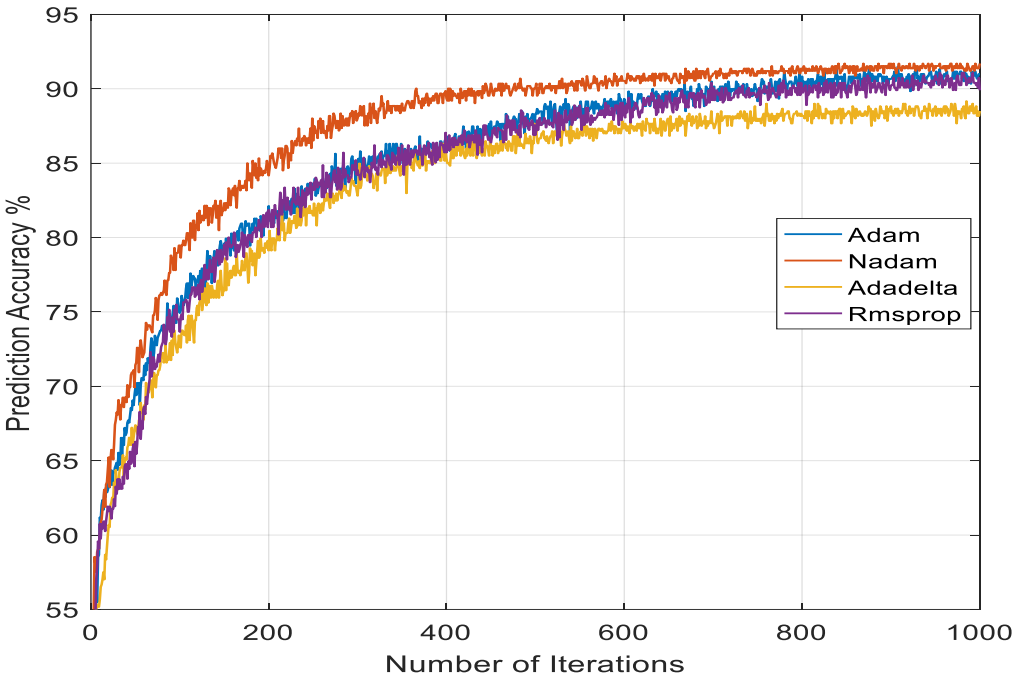

Figure 4-8: ANN-based prediction accuracy vs Iteration numbers for Arbiter PUF.

From the results, it is observed that, for different models of the PUFs, the prediction accuracy increases with the increase in the total iterations number, which is used for training. Figure 4-9 represents the loss graph for different optimizations, and it has been found that Nadam optimization converges faster than the other optimization algorithms.

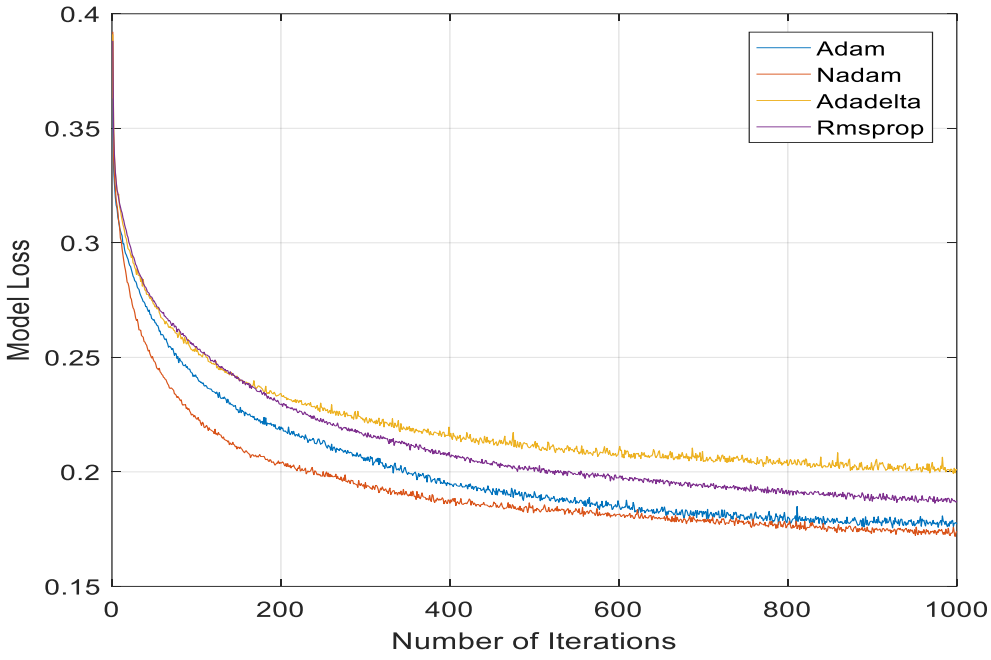

Figure 4-9: Training Loss for Different ANN Optimizers.

4.11 Comparative Analysis Among Different Algorithms

The prediction accuracies are much better for each of the listed PUFs when swarm-based modeling attacks are used. Table 4.4 summarizes the prediction accuracies for the different types of PUFs under study. It is observed from this table that the prediction accuracies, when the Swarm Intelligence models (DA, GSA, CS, PSO & GWO) are used, are much better than the other algorithms for each of the listed PUFs. For easy comparison, the results in Table 4.4 are also shown in the chart of Figure 4-10. It is clear from this figure

that the DA, GSA, CS, PSO and GWO optimizations in ANN give better prediction accuracy results than Adadelta, RMSprop, Adam, and Nadam optimization algorithms. The Swarm Intelligence-based model attacks have prediction accuracies ranging from 71.1% - 99.3%. In contrast, for the machine learning ANN-based models, the prediction accuracies range from 68.0% to 94.1%.

Table 4.4: Prediction accuracy comparison for different optimizers.

Type of PUF	Adadelta %	RMSprop %	Adam %	Nadam %	DA %	GSA %	CS %	PSO %	GWO %
ROPUF	87.7	89.1	90.3	91.4	91.5	92.6	93.9	94.3	95.9
Configurable ROPUF	92.7	93.0	93.5	94.1	94.4	95.9	96.5	97.4	99.3
Inverter ROPUF	87.0	88.7	89.5	90.3	89.6	90.2	90.7	92.2	93.7
Arbiter PUF	88.5	89.9	91.0	91.7	90.1	91.1	91.8	92.3	94.1

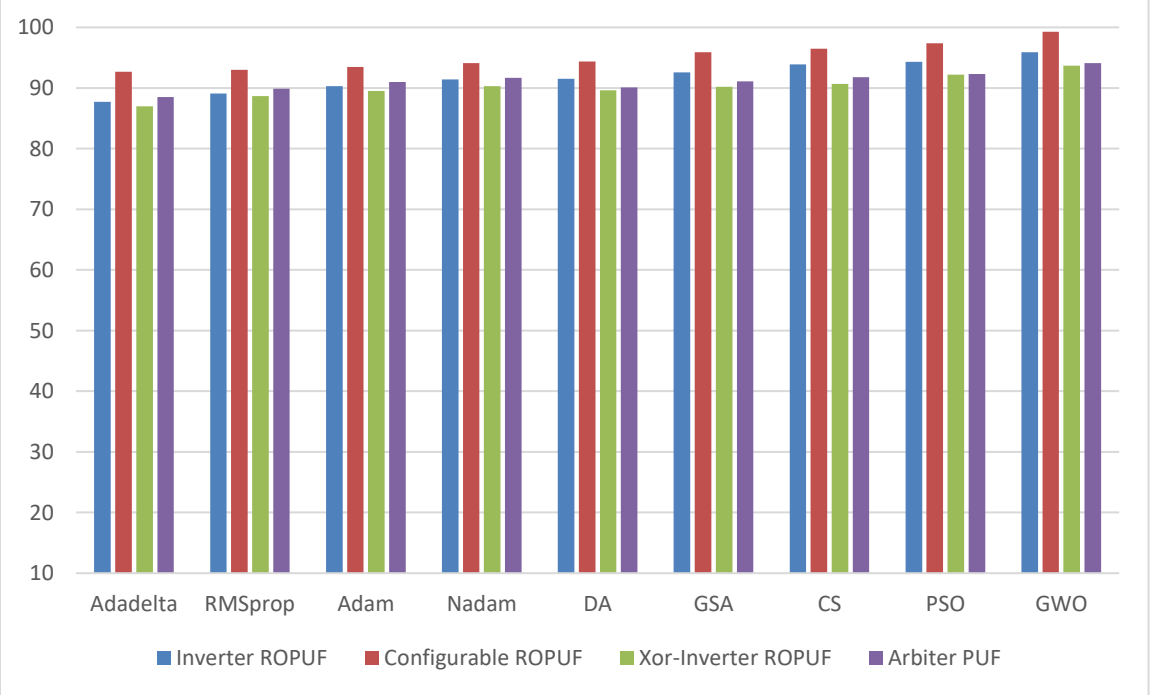

Figure 4-10: Prediction accuracies for different PUFs using various optimization models.

4.12 Statistical analysis of the results

This subsection explains the statistical analysis of the various algorithm results where multiple comparison procedures have been employed. In order to apply statistical analysis, a null hypothesis is defined, which implies that all the algorithms have the same performance without a significant difference; therefore, a denial of this hypothesis suggests the existence of differences between these algorithms [71]. If the hypothesis is rejected, a significance value α is applied to decide the rejection level. The p-values are used to describe the significance of the hypothesis test. If the p-value is more significant than α, then there is not enough evidence to reject the null hypothesis. Otherwise, the hypothesis is rejected, which indicates that the algorithms have different performances. The Nonparametric Friedman test is used to compute p-values to define significant differences between the algorithms' prediction accuracy [90]. Then a significance value α=0.05 is chosen. In computing the Friedman Value $\boldsymbol{F_f}$, the test ranks the algorithms according to the highest prediction accuracy (Rank 1), the second highest (Rank 2), down to the lowest ranking. The Friedman test computes $\boldsymbol{F_f}$ Value as:

$$F_f = \frac{12n}{k(k+1)}\left[\sum R^2 - \frac{k(k+1)^2}{4}\right] \tag{4.7}$$

where R is the ranks, n is the number of PUF datasets, k is the number of algorithms, and the statistic is distributed according to F_f with k − 1 degrees of freedom [91,92]. Table 4.5 shows the obtained average rankings of the algorithms by Friedman test based on prediction accuracy. GWO has the best performance in prediction accuracy among all algorithms; therefore, it has a rank of 1 and will be used as the control algorithm. The result obtained from the Friedman test, including its corresponding associated p-value, is shown

in Table 4.6. From the table, it is observed that the p-value is lower than the level of significance (0.05); therefore, there are significant performance differences between the algorithms, which implies that the null hypothesis is rejected. Considering the differences between the algorithms, we need a post-hoc procedure to identify these differences and then find out the p-value in order to determine the hypothesis rejection degree. Holm's procedure has been used to determine whether the control algorithm presents statistical differences concerning the remaining algorithms [93].

Table 4.5: Average rankings of the algorithms by Friedman test.

Algorithm	**Ranking**
GWO	1
PSO	2
CS	3.25
GSA	4.25
Nadam	4.75
RMSprop	5.75
Adadelta	7

Table 4.6: Results of the Friedman Tests.

Friedman Value	**p-value**
21.99937	0.00121

Holm’s procedure compares the control algorithm, which is GWO, with the other remaining algorithms, which consider a multiple comparison procedure. The test statistic,

z value, is used to find the corresponding probability from the table of the normal distribution:

$$Z = \frac{R_i - R_j}{\sqrt{\frac{k(k+1)}{6N}}} \tag{4.8}$$

where, Ri and Rj are the average rankings by the Friedman test of the algorithms compared [74]. These unadjusted p values are used to compute p-Holm sequentially and test the hypotheses ordered by their significance level of confidence α. Table 4.7 shows that when the highest prediction accuracy algorithm (GWO) is used as a control algorithm, it performs better than Adadelta, RMSprop, Nadam and GSA with α = 0.05, and GWO outperforms all the algorithms with α = 0.10 except PSO.

Table 4.7: Adjusted p-values. GWO is the Control Algorithm

Algorithm	Z	Unadjusted p-value	p-Holm
Adadelta	4.242641	0.000022	0.000132
RMSprop	3.358757	0.000392	0.001960
Nadam	2.651650	0.004006	0.016024
GSA	2.298097	0.010781	0.032343
CS	1.64521	0.049964	0.099928
PSO	0.707107	0.239752	0.239752

4.13 Chapter Summary

This chapter presents a comprehensive study and analysis of the vulnerability of different delay- based PUF designs to various machine learning modeling attacks. From the results, it can be observed that the performance of the algorithms is quite similar in nature, however, KNN performs better than the other algorithms in predicting the CRPs. The best accuracy observed for the algorithms LR, DT, RF, KNN, SVM, KSVM, and NB are 88.9%, 90.3%, 91.5%, 92.3%, 91.1%, 91.9%, and 86.1%, respectively. The results show that the ANN-based algorithms produce better response prediction accuracy results compared to other machine learning classifiers. For the ANN-based modeling, it is observed that the accuracy range for response prediction is between 68.0% to 94.1%, where the best accuracy for response prediction is 94.1% for the ANN-based modeling using the Nadam optimizer on the Configurable ROPUF.

Chapter 5

Design and Implementation of XOR-ROPUF for Thwarting Different Machine Learning Attacks

In this chapter, we introduce the design of a novel XOR-ROPUF capable of thwarting machine learning attacks. This is achieved by feeding back the bit responses from the oscillator output to the challenge generator for generating the next challenge vector. This feedback technique makes the design more protected from attackers who can use invasive and non-invasive methods to steal the CRP data. Different machine learning modeling attacks are used to study the vulnerability of the proposed PUF.

5.1 Introduction

Physical Unclonable Functions (PUFs) are used to provide security and authentication in assured and trusted integrated circuits by producing unclonable cryptographic keys. However, machine learning algorithm attacks have proven to be effective against PUFs [94]. By using machine learning attacks the Challenge-Response Pairs (CRPs) can be predicted using a relatively small subset of training samples. During the past several years, hardware systems have become the targets of different types of specialized attacks [1]. There are documented cases where attackers have embedded secret

circuits, known as Trojans, in integrated circuits (ICs) to steal information and cause malfunction in such devices [2]. Many different approaches, including PUFs, have been studied by researchers to secure the hardware system from such expanding attack threats [3,4]. Physical Unclonable Functions (PUFs) are considered as a promising solution to security threats which help to implement hardware-based security scheme in devices [5]. The PUF's microstructure is unclonable and non-reproducible due to the unpredictable physical factors that get introduced at the time of IC manufacturing. A set of unique Challenge-Response Pairs (CRPs) can be generated based on these manufacturing process variations using PUFs. These unique CRPs can be considered as a signature or an authentication scheme for a specific device; therefore, there is no need for storing security keys in a nonvolatile memory [10].

5.2 Implementation of XOR-ROPUF

The proposed delay-based design generates a unique 'n' bit vector response by developing an 'n' bit response for an 'n' bit challenge. The design consists of NAND, XOR, and INV gates, and an odd number of inverters are used for a ring oscillator. Both XOR and NAND gates can be programmed to be used either as inverters or NAND or XOR gates. The vulnerability analysis for this design for a (n $\times$ n) challenge-response ROPUF shows a significant reduction in the prediction accuracies, thus making the design less vulnerable to ML based attacks. The proposed design is implemented on 10 Xilinx Artix 7 FPGAs that are installed on the Diligent Nexys 4 board. The challenge-response pairs from the FPGA are recorded using the Agilent 16801A logic analyzer. In this design as shown

in Figure 5-1, the responses generated from the PUF are fed back to the challenge generator. An XOR network takes the challenges from the challenge generator and combines the response to generate a new challenge. This process is maintained until the response bits match the same size as the challenges [95]. Changing the challenges for every response bit will secure the design from the attackers to get the information about the challenges provided to the XOR-ROPUF.

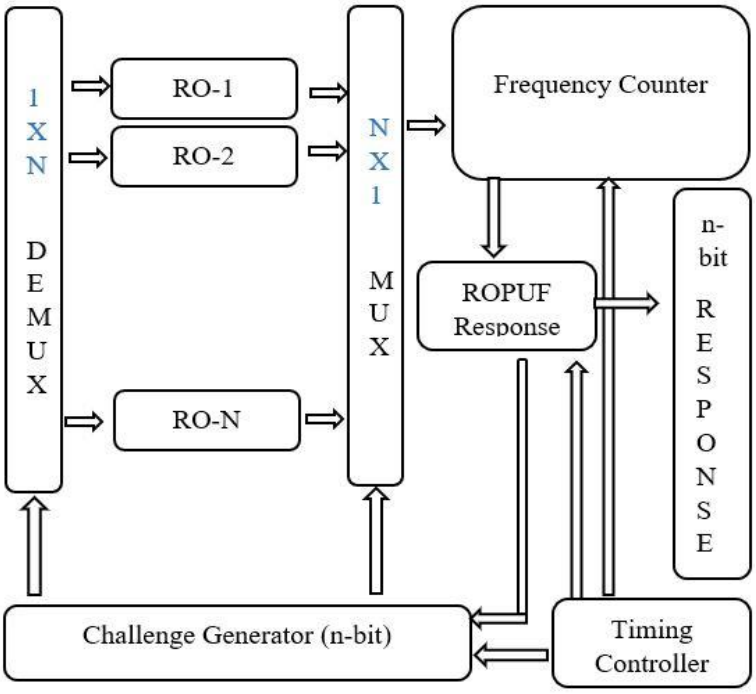

Figure 5-1: Design of XOR-ROPUF to generate n-bit response for an n bit challenge.

A total of 256 oscillators are implemented in each of the FPGA boards with the same fixed routing delay and at the same spatial location. Figure 5-2 represents the schematic of the synthesized XOR-ROPUF.

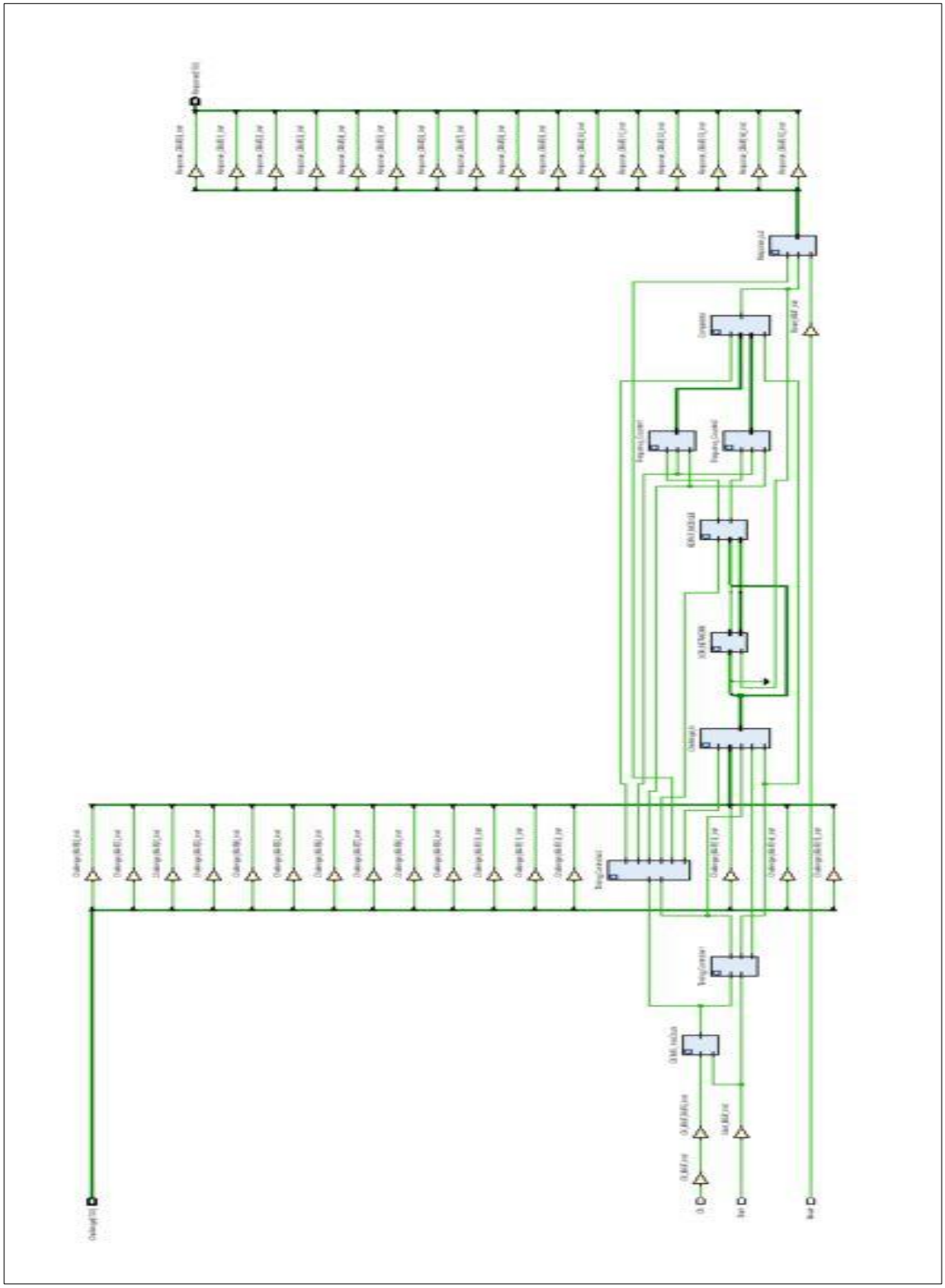

Figure 5-2: XOR-ROPUF Schematic.

The selection of particular ROs is based on a challenge generator. Then the RO is turned on for 0.4 ms to obtain the response from the ring oscillator by using a logic analyzer. The challenge generator network varied the input challenge bits with the output response bits to generate a random challenge for the design. The PUF design consists of demultiplexers, ROPUF network, and multiplexers, as shown in Figure 5-3. The Ring Oscillators are selected using the multiplexer and demultiplexer; the selection is mixed for two different demultiplexers and multiplexers. The challenge mainly consists of two parts, and each part selects the first\second ring oscillator using the first\second demultiplexer and multiplexer. Figure 5-4. shows how the LUTs are mapped according to the XOR inverter PUF design. Ring Oscillators frequencies are collected from the frequency counters and compared to generate the response. The response generation is controlled by the timing controller until the 16-bit response is generated.

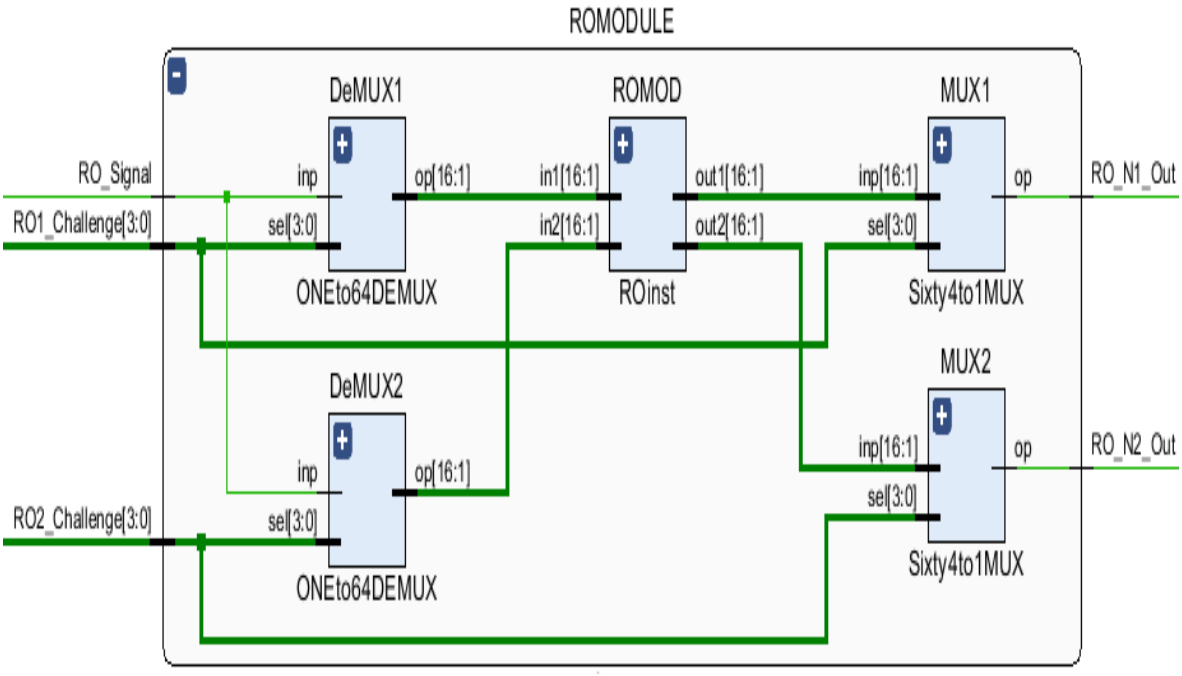

Figure 5-3: XOR-ROPUF design.

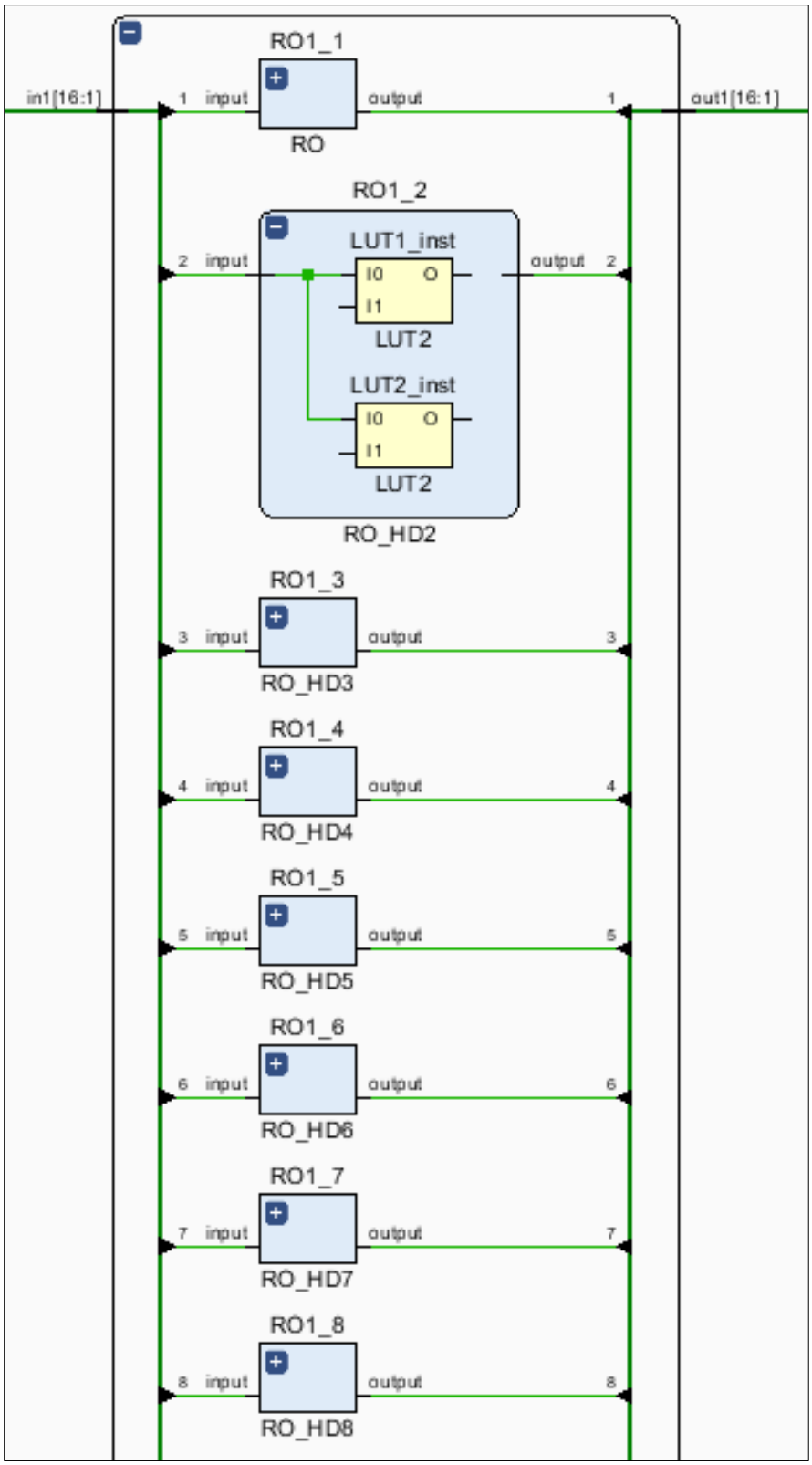

Figure 5-4: Oscillator Network.

The Artix 7 FPGA boards have an internal system clock of 100MHz, which is brought down to 1 kHz to enable and synchronize the different parts of the design, Ring Oscillators, Challenge, Frequency Counter, Frequency Counter Reset, Comparator, and ROPUF Output Enable. Figure 5-5 shows the different timing controllers that are used to operate the design.

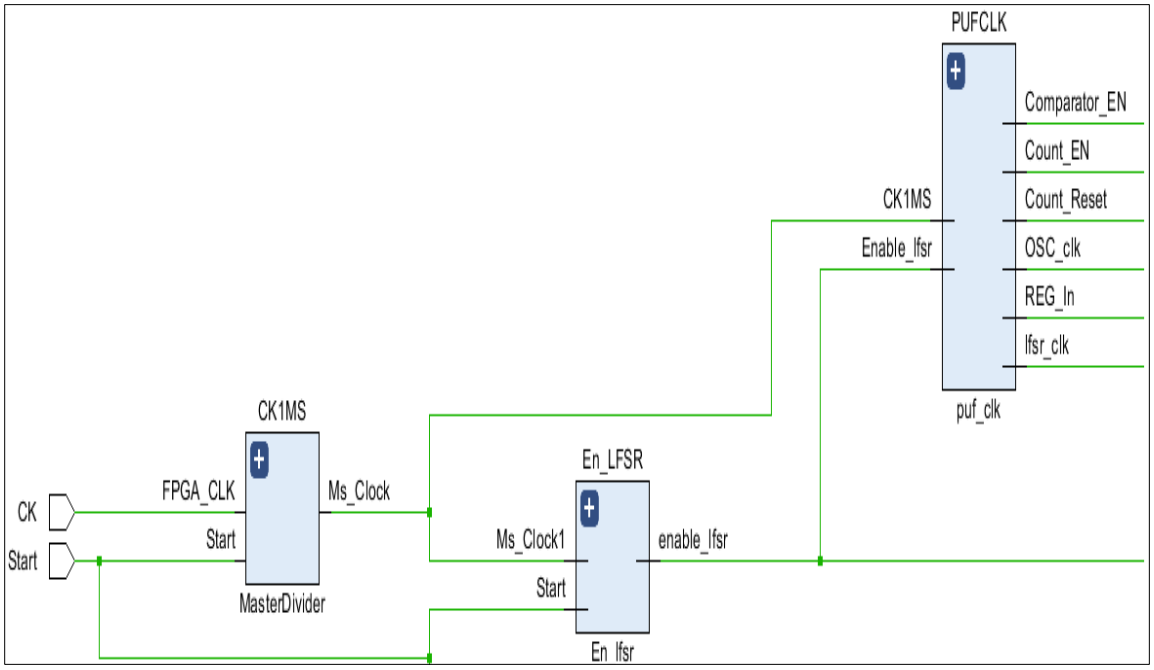

Figure 5-5: Timing Controllers.

A total of 256 oscillators are located in the FPGA using hard XDC macros to provide constant routing for different oscillators. The design has been implemented to select the oscillators in a pair with the same routing, as shown in Figure 5-6. The placement of oscillators is done in such a way so that the difference between the two oscillator frequencies meets a specific threshold frequency. If the oscillators do not meet the criterion, they are moved to another CLB slice.

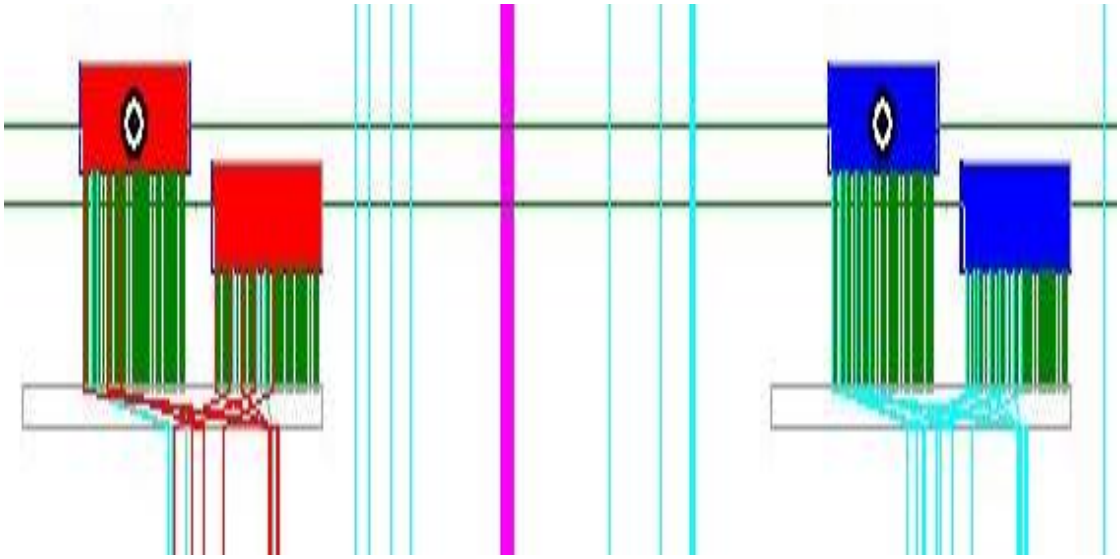

Figure 5-6: Placement of Fixed Routing Ring Oscillators.

5.3 Experimental results of different modeling attacks

In this section, different machine learning classifier algorithms: Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), K-Nearest Neighbor (KNN), kernel Support Vector (Kernel SVM), Naive Bayes Classifier (NB) classifiers, along with four Artificial Neural Network (ANN) based algorithms: RMSprop, Adadelta, Adam, Nadam, and different swarm intelligence algorithms: Dragonfly Algorithm (DA), Gravitational Search Algorithm (GSA), Cuckoo Search Algorithm (CS), Particle Swarm Optimizer (PSO) and the Grey Wolf Optimizer (GWO) are implemented to study the CRPs prediction behavior of the proposed PUF [71,95]. In order to perform the attacks on the results obtained from the proposed PUF design, we assume that an adversary can eavesdrop on a small number of the CRPs and tries to predict the response for different challenges and replicate the PUF's challenge-response behavior. The attacking models are trained

using available PUF CRPs to emulate their behavior using the above different algorithms. For training the CRPs, a 2.3 GHz PC with 16 GB RAM and 2GB Graphics card is used, and the Keras framework is used for modeling the PUF's CRPs with Theano and Tensorflow. The response prediction accuracy is determined by using cross-validation of ten blocks using K-fold method. The challenge vector and the response matrix for the proposed PUF are defined as:

$$C = [C_1, C_2, \dots, C_m]^T \tag{5.1}$$

$$R = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \dots & \vdots \\ r_{m1} & r_{m2} & \dots & r_{mn} \end{bmatrix} \tag{5.2}$$

5.3.1 Machine Learning Classifier Modeling Attacks

Table 5.1 represents the results obtained for attacking XOR-ROPUF. It can be observed that the performance of the algorithms varies from one algorithm to another; however, the prediction accuracy using different ML models is in the range of 5.6% to 20.7%. From the table, it can be concluded that KNN performs better in predicting the CRPs, compared to accuracy for the other algorithms, with a prediction accuracy of 20.7%

Table 5.1: ML classifiers prediction accuracy for XOR-ROPUF design.

Number of CRPs	10,000							30,000						
Type of PUF	LR %	DT %	RF %	KNN %	SVM %	KSVM %	NB %	LR %	DT %	RF %	KNN %	SVM %	KSVM %	NB %
XOR-ROPUF	7.3	8.1	8.6	9.1	5.6	8.1	6.5	18.3	19.1	19.5	20.7	19.2	19.7	17.1

5.3.2 Artificial Neural Network based Modeling Attacks

For ANN attacks, it is noted that the models are unable to predict the responses with higher prediction accuracy, this is shown in Table 5.2 and Figure 5-7. The best prediction accuracy of 23.5% is observed for the Nadam optimization when a 30K CRPs are used.

Table 5.2: ANN based prediction accuracy for XOR-ROPUF design.

Number of CRPs	10,000				30,000			
Type of PUF	Adadelta %	RMSprop %	Adam %	Nadam %	Adadelta %	RMSprop %	Adam %	Nadam %
XOR- ROPUF	9.1	9.7	10.3	10.7	20.9	21.5	22.7	23.5

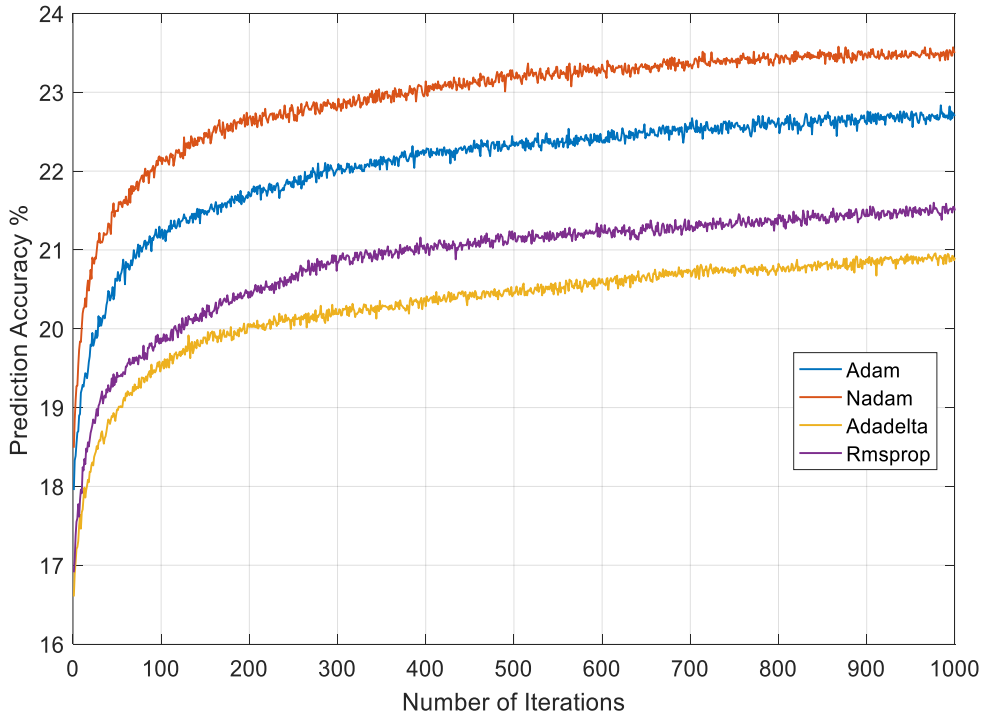

Figure 5-7: Prediction accuracy vs Iteration numbers for different ANN algorithms.

5.3.3 Swarm Intelligence based Modeling Attacks

From Table 5.3, it is found that the prediction accuracies using the swarm models are in the range of 11.1% to 31.7%. Figure 5-8 shows the prediction accuracies versus the number of iterations for the DA, GSA, CS, PSO and GWO models. Here, it is also observed that the GWO model's performance is better than the other algorithms in terms of prediction accuracies 31.7%.

Table 5.3: Swarm Intelligence based prediction accuracy for XOR-ROPUF design.

Number of CRPs	10,000					30,000				
Type of PUF	DA %	GSA %	CS %	PSO %	GWO %	DA %	GSA %	CS %	PSO %	GWO %
XOR- ROPUF	11.1	11.3	11.5	12.4	14.5	23.7	25.1	27.5	28.9	31.7

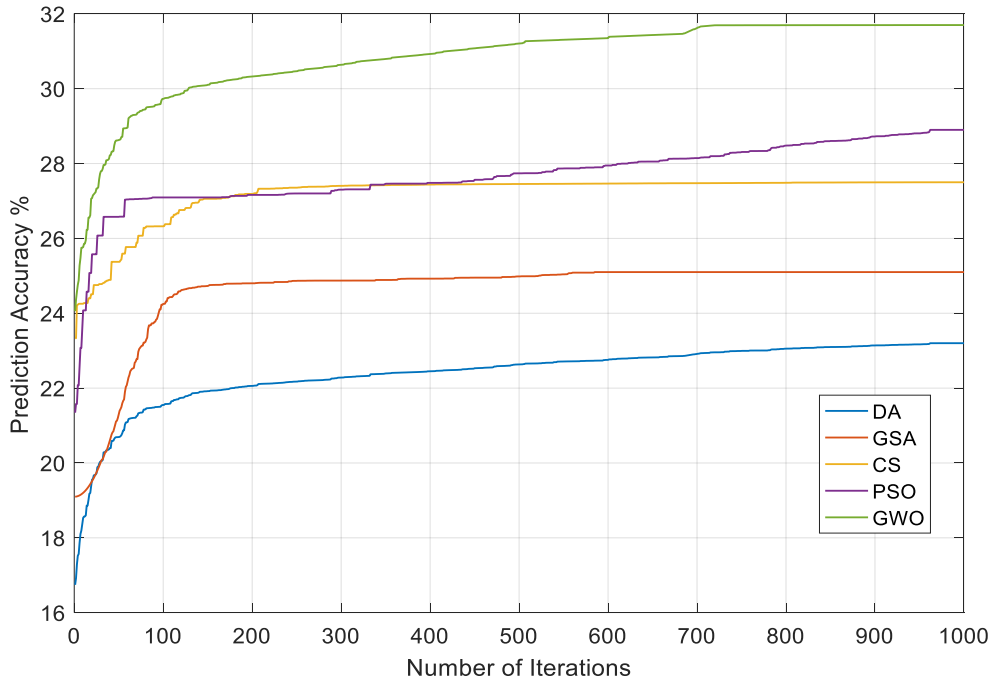

Figure 5-8: Prediction accuracy vs Iteration numbers for different swarms algorithms.

5.4 Comparison Among Different PUF Designs

For a 16-bit challenge, the assumption is that if an attacker can predict at least 13-bits correctly, then the attacker can easily decrypt the response using the combinational approach or brute force approach. The prediction accuracy for 'n' bit response can be calculated as:

$$\textit{Prediction accuracy} = \sum_{r=0}^{n} nCr / \textit{number of challenge bits} \quad (5.3)$$

To generate an n bit response for all PUFs designs, which already discussed in previous phases, requires n number of n bit challenges. Simultaneously, calculating the prediction accuracy for an n bit response has set a tolerance limit of 3-bit error. We assume that if an attacker can get 13 out of 16 possible bits right, then the attacker can break into the system after several tries. The results obtained for modeling attacks on the design and the other PUF designs for n bit response using ML classifiers, ANN-based modeling, and swarm-based modeling are presented in Tables 5.4, 5.5, and 5.6.

Table 5.4: Classifier ML Prediction Accuracy for PUFs.

Type of PUF	LR %	DT %	RF %	KNN %	SVM %	KSVM %	NB %
XOR-ROPUF	18.3	19.1	19.5	20.7	19.2	19.7	17.1
ROPUF	74.7	75.1	75.7	79.1	78.1	78.5	69.9
Configurable ROPUF	80.9	82.3	83.5	84.3	83.1	83.9	78.1
Inverter ROPUF	73.6	74.6	75.9	78.1	77.1	77.4	70.8
Arbiter PUF	70.9	72.8	73.1	77.2	75.9	76.8	68.0

Regarding classifier-based algorithms, the maximum prediction accuracy for the XOR-ROPUF design is 20.7%, whereas the maximum accuracy obtained is 84.3% in the design of Configurable ROPUF.

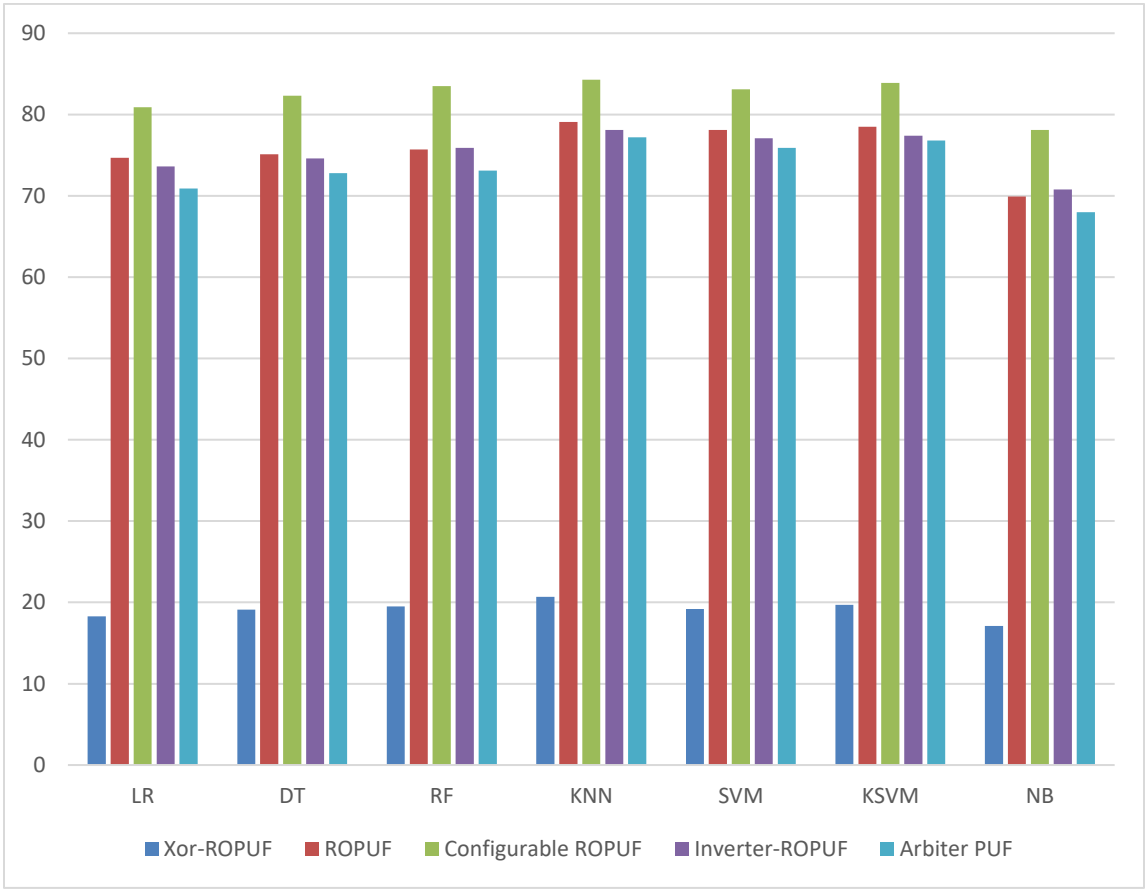

Figure 5-9: Comparison of classifier ML prediction accuracies for PUFs.

From the ANN-based modeling attacks on different PUF designs, it is observed that the XOR- PUF prediction accuracy is low compared to other PUFs, which is presented in Table 5.5 and Figure 5-10.

Table 5.5: ANN-Based Prediction Accuracy for PUFs.

Type of PUF	Adadelta %	RMSprop %	Adam %	Nadam %
XOR-ROPUF	20.9	21.5	22.7	23.5
ROPUF	79.9	80.1	81.3	81.7
Configurable ROPUF	85.3	85.7	86.1	86.7
Inverter ROPUF	79.0	79.5	80.3	81.1
Arbiter PUF	77.5	78.2	78.3	79.1

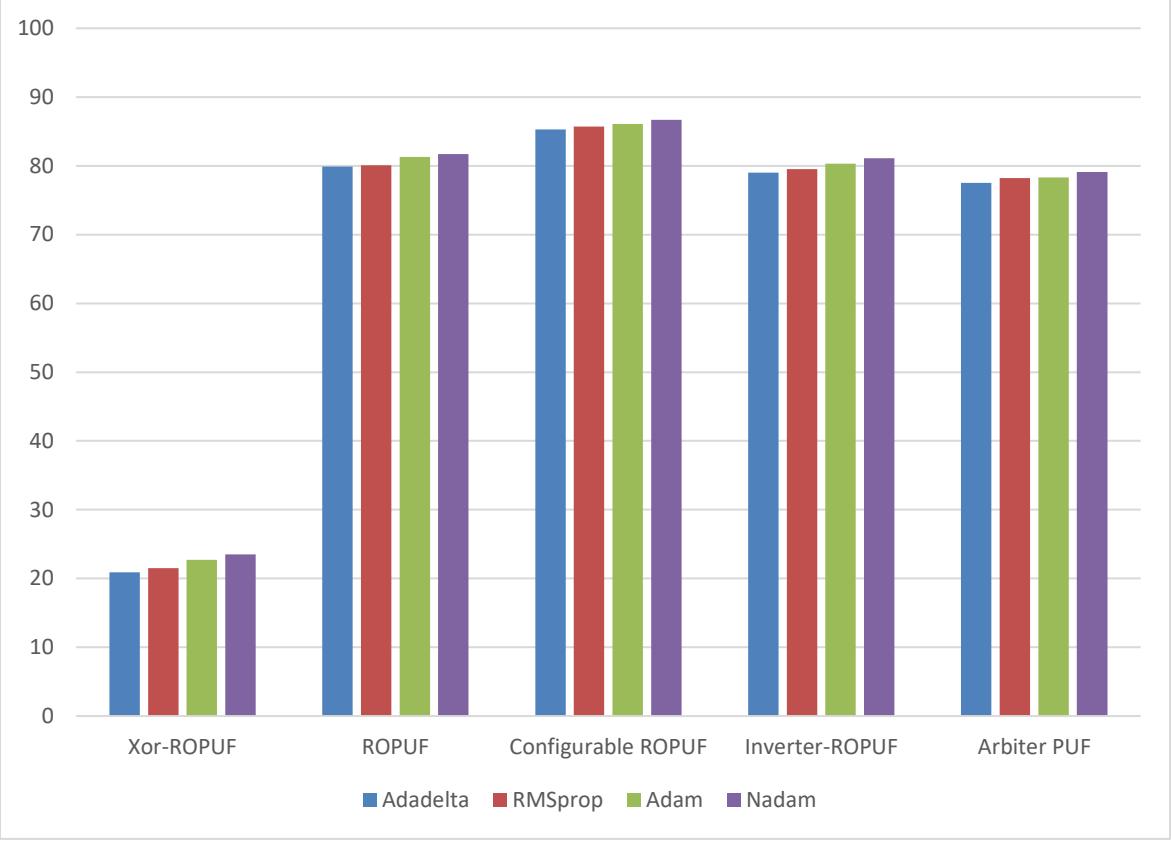

Figure 5-10: Comparison of ANN based prediction accuracies for PUFs.

Table 5.6 lists experimental results for the accuracy for the PUFs using the DA, GSA, CS, PSO and GWO Swarm Intelligence algorithms.

Table 5.6: Swarm based Prediction Accuracy for PUFs

Type of PUF	DA %	GSA %	CS %	PSO %	GWO %
XOR-ROPUF	23.7	25.1	27.5	28.9	31.7
ROPUF	81.5	82.1	82.7	83.4	84.7
Configurable ROPUF	85.9	86.1	86.5	86.9	87.3
Inverter ROPUF	81.5	81.9	82.1	82.5	83.1
Arbiter PUF	81.7	82.0	82.5	83.0	84.2

From the table, it is evident that the PUF structures are vulnerable to Swarm Intelligence-based model attacks with prediction accuracies ranging from 81.5% - 87.3%. while in case of the XOR-ROPUF, it is noted that the models are unable to predict the responses with higher prediction accuracy. The best prediction accuracy of 31.7% is observed for the GWO optimization as shown in Figure 5-11.

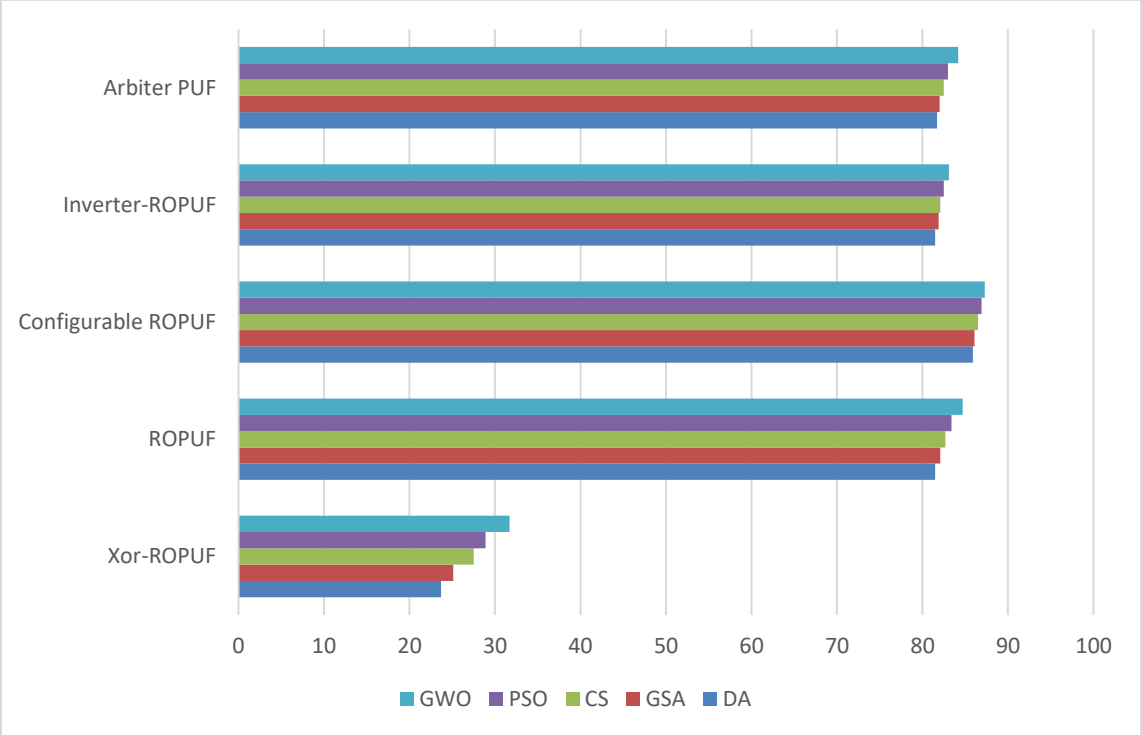

Figure 5-11: Comparison of swarm based prediction accuracies for PUFs.

5.5 Chapter Summary

For the enhancement of PUF security, a novel delay-based XOR-ROPUF design is proposed, which generates a ‘n’ bit response for a ‘n’ bit challenge. When it comes to classifier-based algorithms, the maximum prediction accuracy for the XOR-ROPUF design for an ‘n’ bit response is 20.7%. From the analysis of ANN-based modeling attacks on different PUF designs, it is observed that the XOR-ROPUF prediction accuracy significantly reduces from 86.7% to 23.5%. Also, from the results, it is evident that the other PUF structures are vulnerable to swarm intelligence-based modeling attacks with prediction accuracies ranging from 81.5% - 87.3%. In case of the XOR-ROPUF, it is noted that the models are unable to predict the responses with high prediction accuracy. The best prediction accuracy of 31.7% is observed for the GWO optimization.

Chapter 6

Design and Implementation of a Hybrid Delay Based FPGA PUF Resistant to Machine Learning Attacks

In this chapter, we present a unique delay-based PUF structure that combines challenge-response pairs (CRPs) from an Arbiter PUF with a XOR based Ring Oscillator to generate responses which are found to be less vulnerable to machine learning modeling attacks. Different Machine Learning (ML) classifier, Artificial Neural Network (ANN), and Swarm Intelligence algorithms (SI) are used to study the vulnerability of the proposed hybrid PUF to these attacks. Our results show that the proposed hybrid PUF is much less vulnerable than other designs when subject to machine learning attacks.

6.1 Introduction

Physical unclonable functions (PUFs) are used to extract unique signatures from silicon-based chips which can be used for chip authentication and producing unclonable cryptographic keys. However, researchers have found that PUFs are vulnerable to various machine learning modeling attacks. During the past several years, the use of Field Programmable Gate Arrays (FPGAs) has increased rapidly because of their accelerated reconfigurability, parallelism, and low cost. FPGAs have been used in different

applications such as defense equipment, communication networks, smart grids, and image processing [96]. Due to the increased deployment of FPGA devices in computing systems, hackers continually devise different types of specialized attacks to breach and counterfeit these devices [1]. The increase in different types of attacks has led to expanded research in hardware-oriented security and trust to secure the system from such expanding threats by deploying additional security mechanisms for FPGA-based systems using Physical Unclonable Function (PUF) [7]. PUFs are well-known security primitives for hardware-oriented security applications due to their instance-specific design, which is based on physical properties, making them unpredictable and unclonable to implement hardware-based security schemes in devices [45]. A PUF is a circuit constructed on a semiconductor device that produces a unique signature using the manufacturing process variations to generate unique challenge-response pairs, which can be used as an authentication scheme for a specific device. From the different types of PUFs, delay-based PUFs are widely studied in CMOS-based silicon devices. The most notable amongst the delay-based PUFs are the Arbiter PUFs (APUFs) and the Ring Oscillator PUFs (ROPUFs) [5]. In order to test the resiliency of these PUFs to adversarial attacks, several machine learning-based modeling attacks have been investigated in the past to emulate the behavior of the PUFs by predicting the challenge-response pairs [22,24]. In each scenario, it is assumed that an adversary can somehow (eavesdrop, side-channel, etc.) access a small number of CRPs which can be used as a learning model to replicate the PUF's behavior and predict the remaining CRPs.

6.2 Implementation of proposed design

The proposed design is implemented on Xilinx Artix 7 FPGAs which consists of 2 slices per Configurable Logic Block (CLB), and each CLB has 4 Look-Up-Tables (LUTs). The challenge-response pairs from the FPGA are recorded using the Agilent 16801A logic analyzer. In the proposed PUF design, the output of the Arbiter PUF are fed to the input of the XOR-Inverter based Ring Oscillator [97]. The proposed design takes a 16-bit challenge and generates a 16-bit response that can be used to thwart ML attacks.

6.2.1 Implementation of Arbiter PUF

A16 multiplexer switch is implemented for the Arbiter PUF for generating a 16-bit challenge and a one-bit response. The same challenge bit triggers the multiplexers to create two identical parallel paths on which the input signal propagates with different delays. A D-latch flip-flop is used as an arbiter to determine which response signal reaches the arbiter first. The multiplexer switch is employed as the building block of APUF, as shown in Figure 6-1.

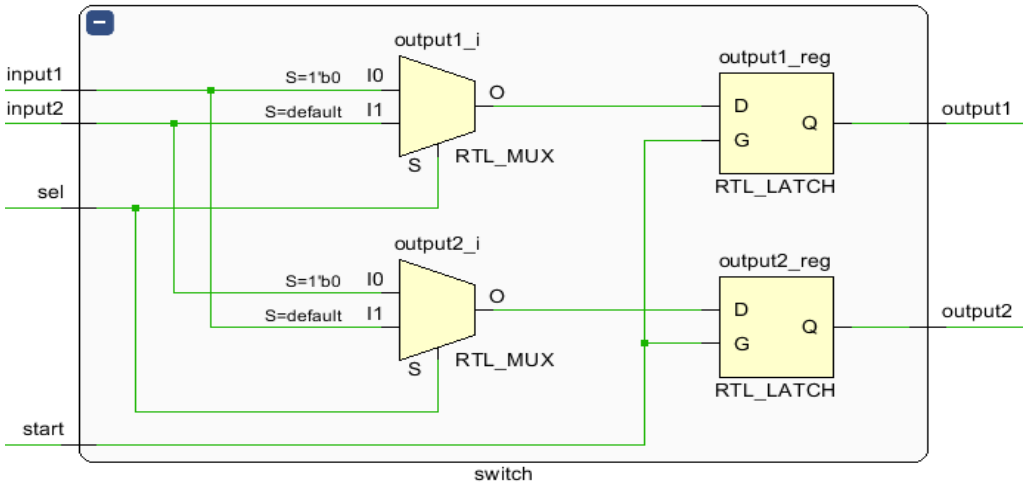

Figure 6-1: RTL level of APUF multiplexer switch

Figure 6-2 shows the connected multiplexers implemented in Arbiter PUF. At the respective clock, a new challenge is generated and fed to the APUF, and the signal is passed, which races across the two paths.

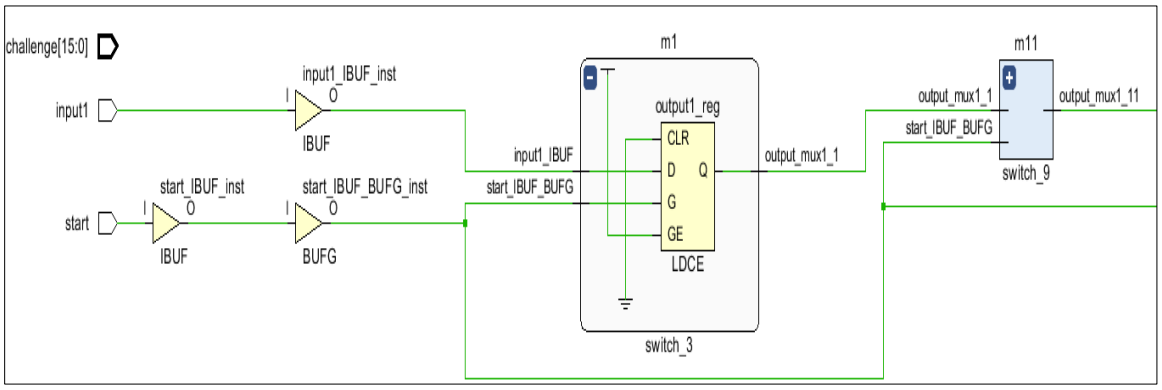

Figure 6-2: Connected Multiplexers in Arbiter PUF

The two paths delay should be the same so that the output is independent of the design delay but entirely rely on manufacturing process variation. An arbiter at the end of the design decides which signal reached first and accordingly determines the output response, either 0 or 1, as shown in Figure 6-3.

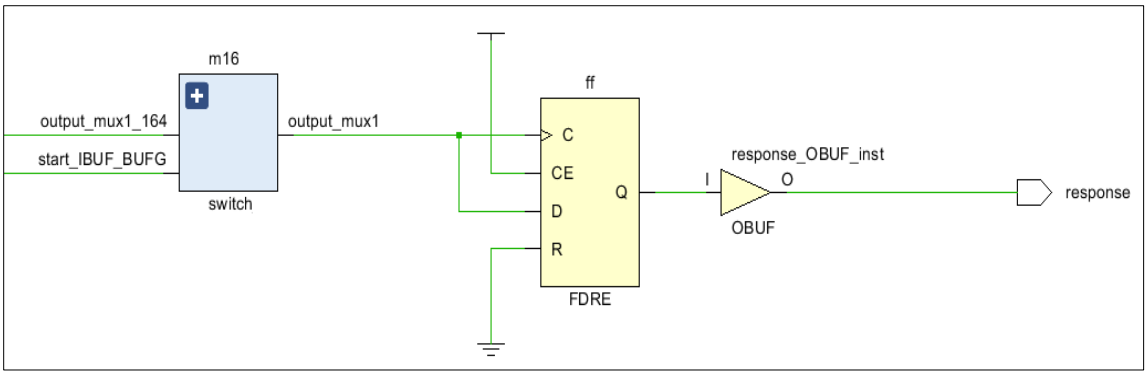

Figure 6-3: APUF Output Response Generator

To generate and feed 16-bit challenges to the APUF, a challenge generator is implemented, which consists of a 4-bit Linear Feedback Shift Register (LFSR) network. These challenge bits are applied at the selection lines of the multiplexers. Also, a clock generator is implemented that triggers leading edge pulses at regular time intervals. The challenge generator and frequency counter are synchronized. Figure 6-4 shows the obfuscator, including of XOR network, where each bit response is obfuscated to generate the challenges for the design. Manual routing using FPGA editor has been created to make the two paths of APUF identical, as shown in Figure 6-5.

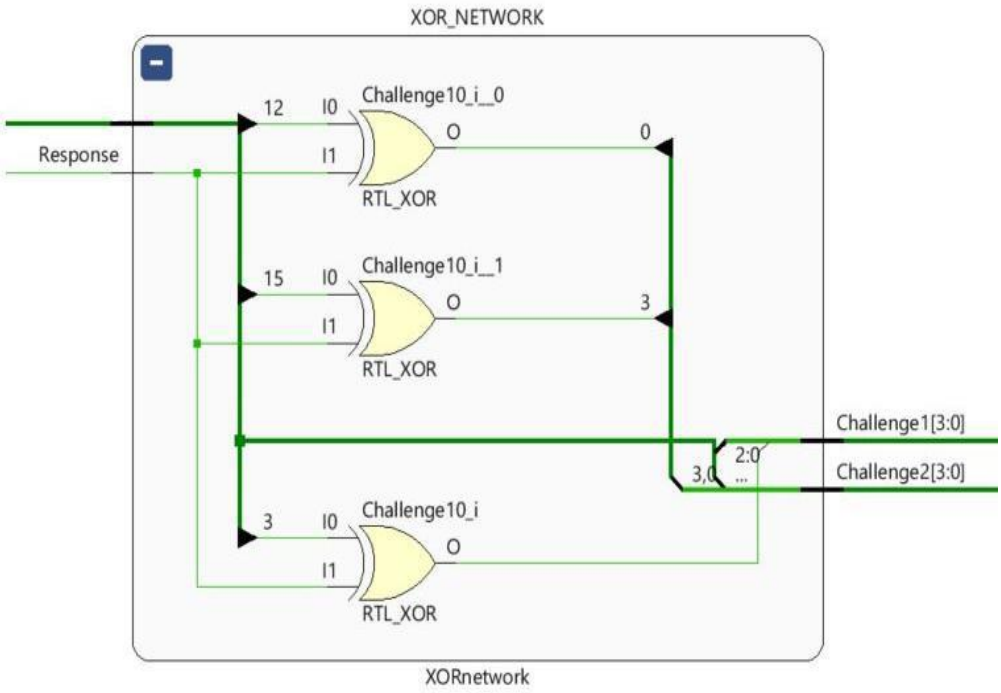

Figure 6-4: Obfuscator Network

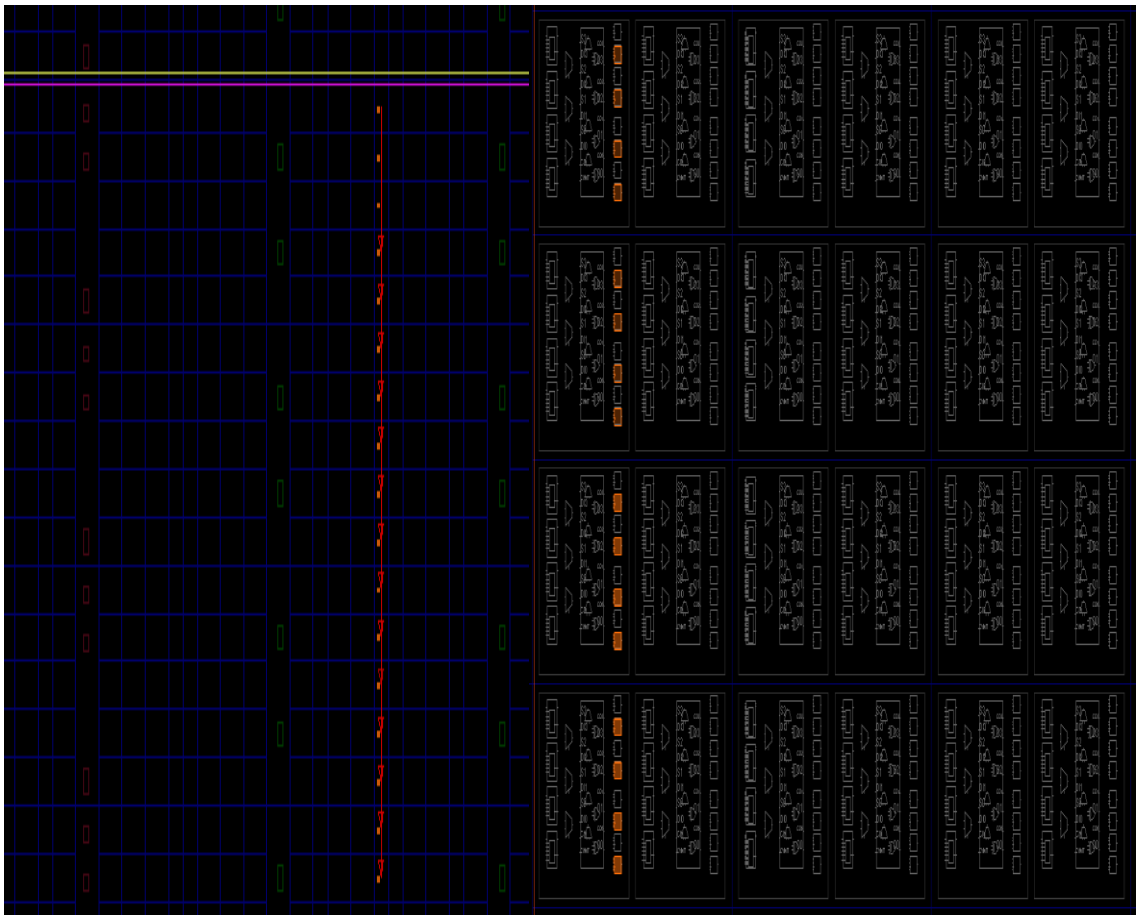

Figure 6-5: Manual Routing of Arbiter PUF

6.2.2 Implementation of XOR- ROPUF

A total of 256 oscillators are implemented in each of the 10 FPGA boards with the same fixed routing delay and at the same spatial location. Figure 6-6 represents the schematic of the XOR-Inverter based Ring Oscillator. Each RO is activated for 0.4ms using timing controller and the corresponding frequency is recorded using logic analyzer.

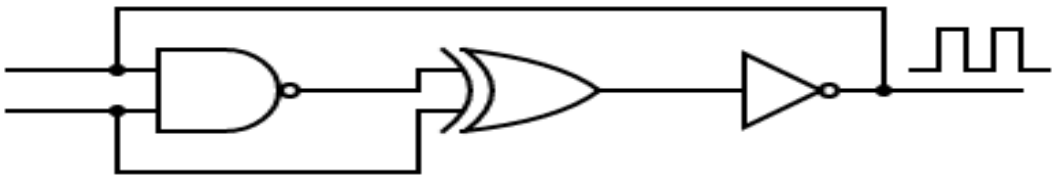

Figure 6-6: XOR-Inverter based Ring Oscillator.

The proper ring oscillators are selected using demultiplexer and multiplexer, which are synchronized to choose the same ring oscillators.

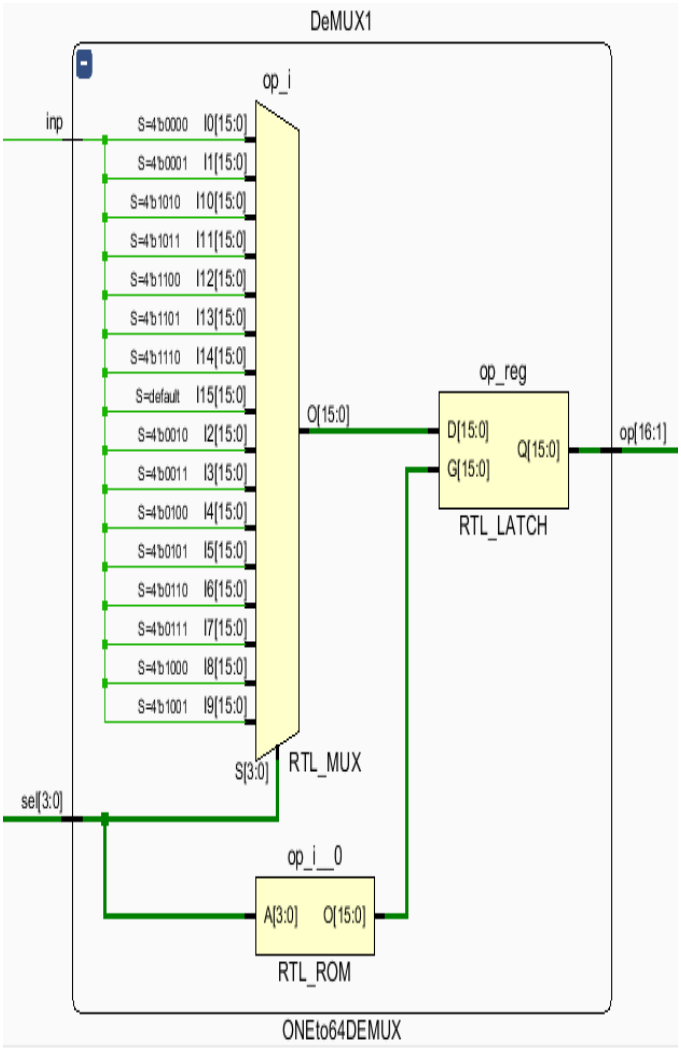

Figure 6-7: Demultiplexer

The ring oscillator enabled signal comes from the demultiplexer to ring oscillator input. The multiplexer selects the output of the ring oscillator and carries the signal to the frequency counter. Figures 6-7 and 6-8 represents designs of the demultiplexer and the multiplexer. According to the selection lines, both demultiplexer and multiplexer are selected to synchronize the input signal and the ring oscillator's output.

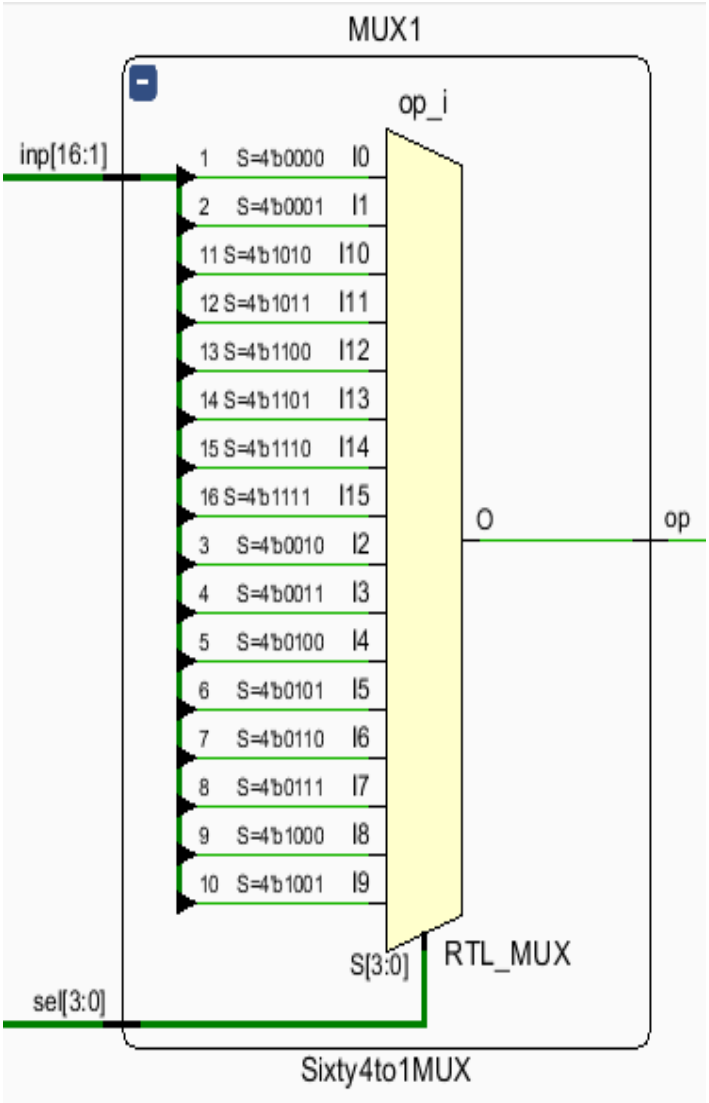

Figure 6-8: Multiplexer

Figure 6-9 shows the output response generation process of the design. Ring Oscillators frequencies are collected from the frequency counters and compared to generate the response. The response generation is controlled by the timing controller until the 16-bit response is generated.

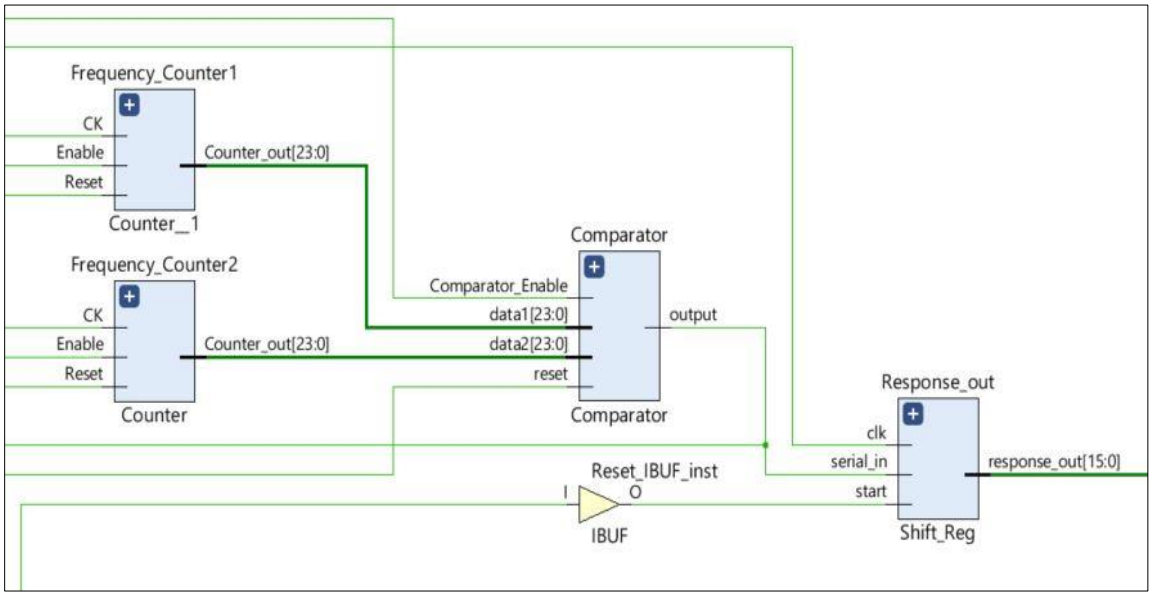

Figure 6-9: Output Response Generator

Figures 6-10 shows the diagram of our proposed architecture. The Arbiter PUF challenges and their respective responses are shuffled and passed through the XOR obfuscation network. The n challenges are passed through the XOR gate to form an n-bit challenge corresponding to the n-bit response. The generated n bit responses from the APUF are fed as challenges to the XOR- Inverter ROPUF in order to select the demultiplexer and multiplexer simultaneously. The responses generated from the XOR-Inverter ROPUF are fed back to the challenge generator to generate the new challenge vector. The final output of the architecture is the 16-bit response corresponding to a 16-bit challenge.

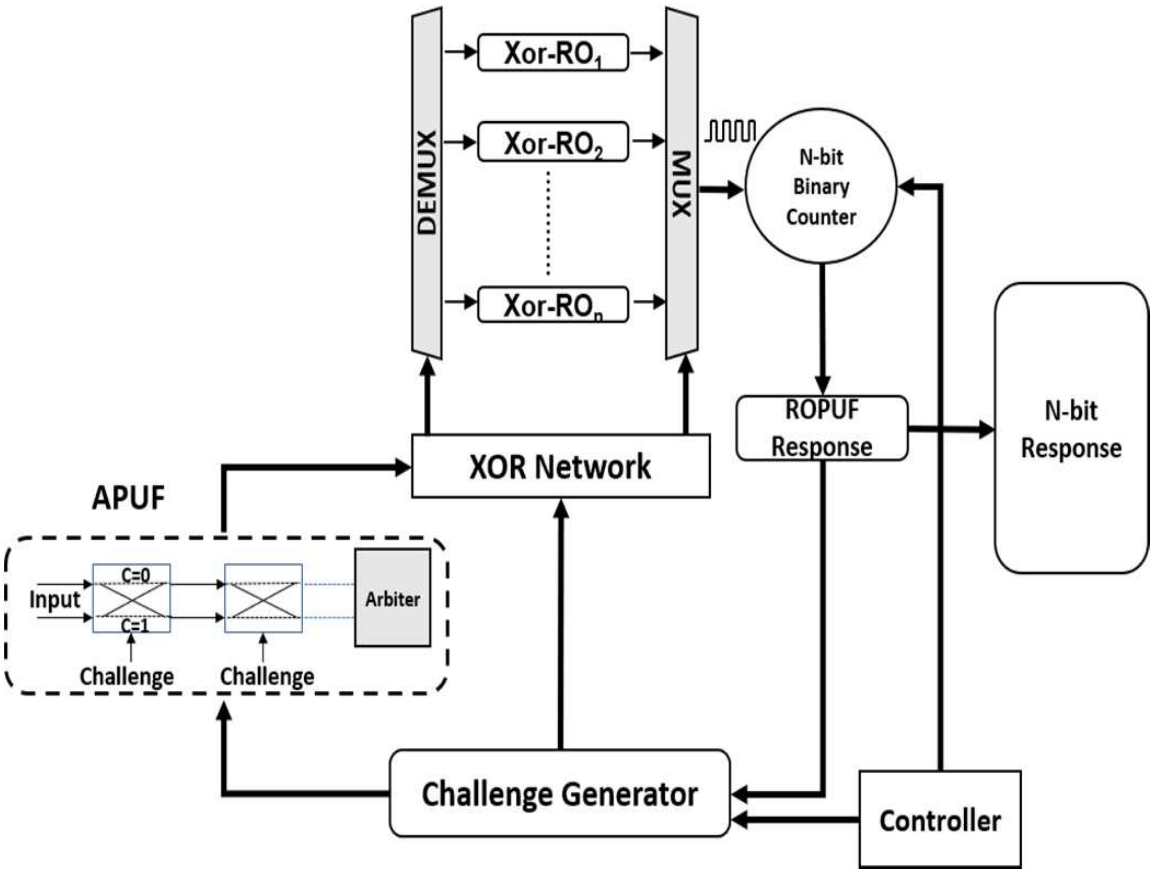

Figure 6-10: Architecture of Hybrid PUF

6.3 Experimental results of different modeling attacks

Different Machine Learning (ML) classifier, Artificial Neural Network (ANN), and Swarm Intelligence algorithms (SI) are used to study the vulnerability of the proposed hybrid PUF to these attacks [97]. For training the CRPs, a 2.3 GHz PC with 16 GB RAM and 2GB Graphics card is used, and the Keras framework is used for modeling the PUF's CRPs with Theano and Tensorflow. The response prediction accuracy is determined by using cross-validation of ten blocks K-fold method. The challenge vector and the response matrix for the proposed PUF are defined as:

$$C = [C_1, C_2, \dots, C_m]^T \tag{6.1}$$

$$R = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \dots & \vdots \\ r_{m1} & r_{m2} & \dots & r_{mn} \end{bmatrix} \tag{6.2}$$

6.3.1 Machine Learning Classifier Modeling Attacks

The results obtained from the classifier ML algorithms for the hybrid PUF are presented in Table 6.1. From the table, it is observed that the KNN algorithm has the best prediction accuracy of 13.7% when 30,000 CRPs are used. Also, it is noted that the prediction accuracy of the ML classifier algorithms increases with the increase in the total number of CRPs.

Table 6.1: ML classifiers prediction accuracy for hybrid PUF

Number of CRPs	10,000							30,000						
Type of PUF	LR %	DT %	RF %	KNN %	SVM %	KSVM %	NB %	LR %	DT %	RF %	KNN %	SVM %	KSVM %	NB %
Hybrid PUF	5.3	6.1	6.7	7.7	4.1	7.1	4.7	11.3	11.1	13.2	13.7	12.1	12.7	10.1

6.3.2 Artificial Neural Network based modeling attacks

Table 6.2 presents the ANN-based modeling results for different number of CRPs. From the data, it is observed that the best prediction accuracy is 15.7% for Nadam optimization.

The prediction accuracies for different models are in the range of 7.6 – 15.7%, as presented in Figure 6-11.

Table 6.2: ANN based prediction accuracy for the hybrid PUF

Number of CRPs	10,000				30,000			
Type of PUF	Adadelta %	RMSprop %	Adam %	Nadam %	Adadelta %	RMSprop %	Adam %	Nadam %
Hybrid PUF	7.6	7.9	8.3	8.7	13.9	14.5	15.2	15.7

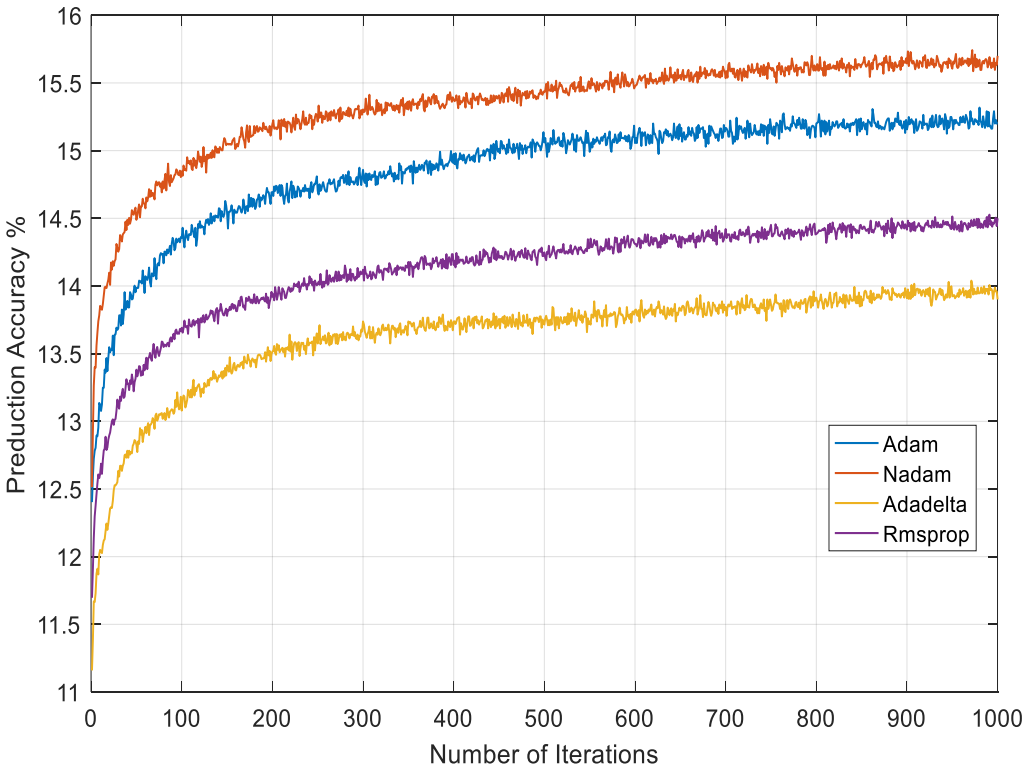

Figure 6-11: Prediction accuracy vs Iteration numbers for different ANN algorithms.

6.3.3 Swarm Intelligence based modeling attacks

The results obtained for swarm based algorithms are presented in Table 6-3. The results show that the algorithms can predict the accuracy of 10.9% for 10,000 CRPs with GWO optimization. However, by increasing the number of CRPs up to 30,000, the GWO algorithms can predict the accuracy of 24.1%. For PSO, the best accuracy is 21.3% whereas, for DA, GSA, and CS, the accuracies obtained are 15.9%, 17.5%, and 19.9%, respectively. With the increase of the number of CRPs, the prediction accuracy for the model increases faster, presented in Table 6.2. The prediction accuracy for all the ANN optimizers is similar, whereas the prediction accuracy can be observed. Figure 6-12 shows the prediction accuracies versus the number of iterations for the Hybrid PUF for the DA, GSA, CS, PSO and GWO models. From the data, it is observed that the best prediction accuracy is 24.1% for GWO optimization.

Table 6.3: Swarm Intelligence based prediction accuracy for hybrid PUF

Number of CRPs	10,000					30,000				
Type of PUF	DA %	GSA %	CS %	PSO %	GWO %	DA %	GSA %	CS %	PSO %	GWO %
Hybrid PUF	7.5	8.8	8.3	9.1	10.9	15.9	17.5	19.9	21.3	24.1

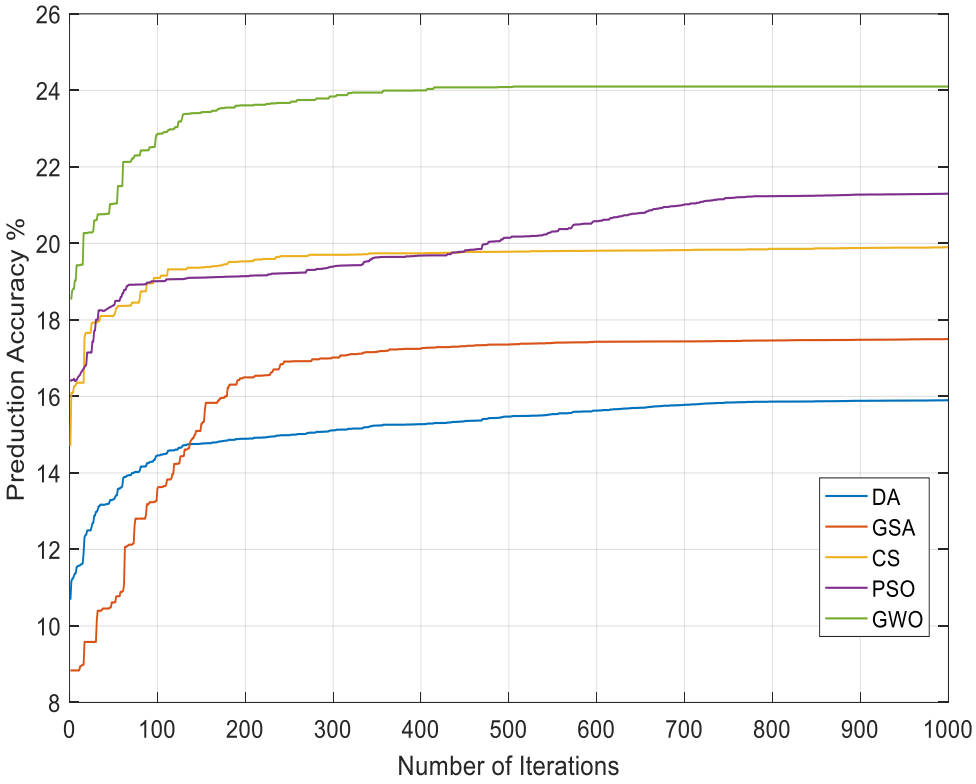

Figure 6-12: Prediction accuracy vs Iteration numbers for different swarms Algorithms.

6.4 Comparison Among Different PUF designs

The results obtained for modeling attacks on proposed design and the other delay based PUF designs using ML classifiers, ANN-based modeling, and swarm-based modeling are presented in Tables 6.4, 6.5, and 6.6. The results are based on 30,000 CRPs. When it comes to classifier-based algorithms, the maximum prediction accuracy for the hybrid design is 13.7%, whereas the maximum accuracy obtained is 84.3% in the design for an n bit response of Configurable ROPUF as presented in Figure 6-13.

Table 6.4: Classifier ML Prediction Accuracy for PUFs.

Type of PUF	LR %	DT %	RF %	KNN %	SVM %	KSVM %	NB %
Hybrid PUF	11.3	11.1	13.2	13.7	12.1	12.7	10.1
XOR-ROPUF	18.3	19.1	19.5	20.7	19.2	19.7	17.1
ROPUF	74.7	75.1	75.7	79.1	78.1	78.5	69.9
Configurable ROPUF	80.9	82.3	83.5	84.3	83.1	83.9	78.1
Inverter ROPUF	73.6	74.6	75.9	78.1	77.1	77.4	70.8
Arbiter PUF	70.9	72.8	73.1	77.2	75.9	76.8	68.0

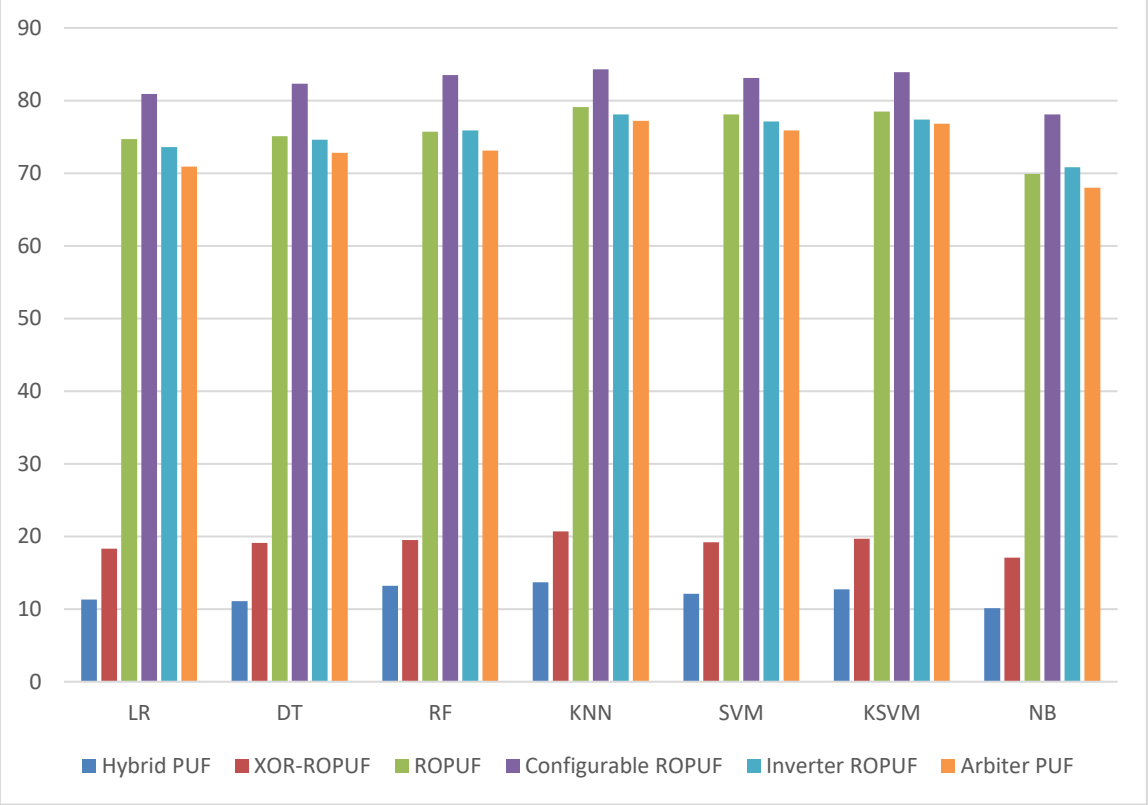

Figure 6-13: Comparison of classifier ML prediction accuracies for PUFs.

From the ANN-based modeling attacks on different PUF designs, it is observed that the XOR- PUF prediction accuracy is low compared to other PUFs, which is presented in Table 6.5 and Figure 6-14.

Table 6.5: ANN-Based Prediction Accuracy for PUFs.

Type of PUF	Adadelta %	RMSprop %	Adam %	Nadam %
Hybrid PUF	13.9	14.5	15.2	15.7
XOR-ROPUF	20.9	21.5	22.7	23.5
ROPUF	79.9	80.1	81.3	81.7
Configurable ROPUF	85.3	85.7	86.1	86.7
Inverter ROPUF	79.0	79.5	80.3	81.1
Arbiter PUF	77.5	78.2	78.3	79.1

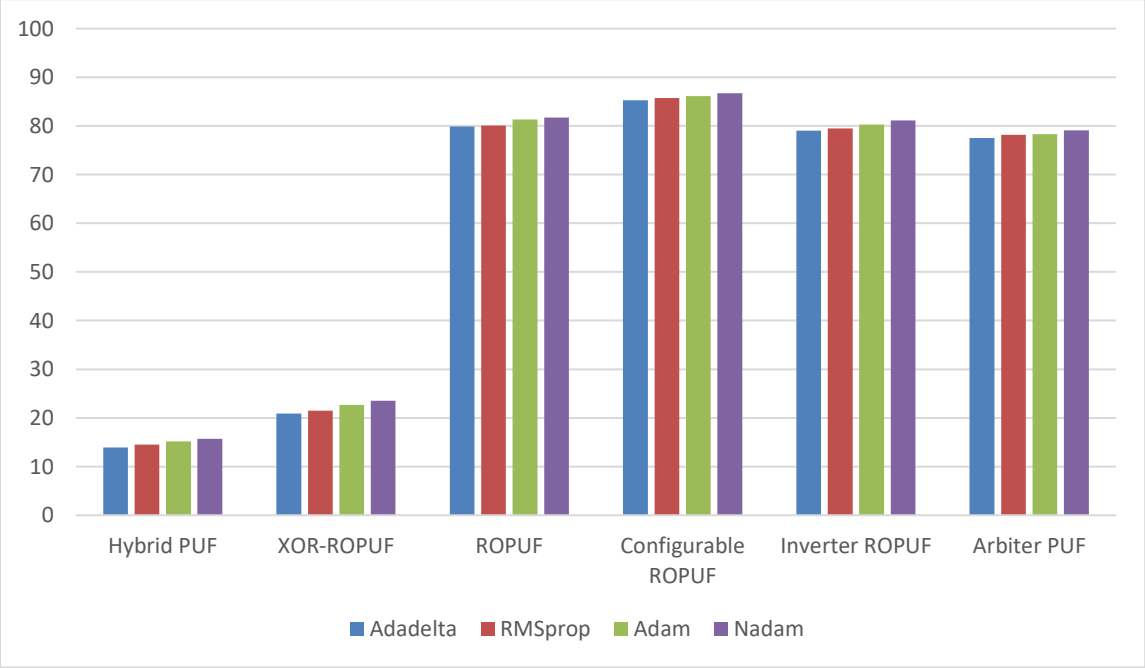

Figure 6-14: Comparison of ANN based prediction accuracies for PUFs.

Table 6.6 lists experimental results for the accuracy for the PUFs using the DA, GSA, CS, PSO and GWO Swarm Intelligence algorithms.

Table 6.6: Swarm based Prediction Accuracy for PUFs

Type of PUF	DA %	GSA %	CS %	PSO %	GWO %
Hybrid PUF	15.9	17.5	19.9	21.3	24.1
XOR-ROPUF	23.7	25.1	27.5	28.9	31.7
ROPUF	81.5	82.1	82.7	83.4	84.7
Configurable ROPUF	85.9	86.1	86.5	86.9	87.3
Inverter-ROPUF	81.5	81.9	82.1	82.5	83.1
Arbiter PUF	81.7	82.0	82.5	83.0	84.2

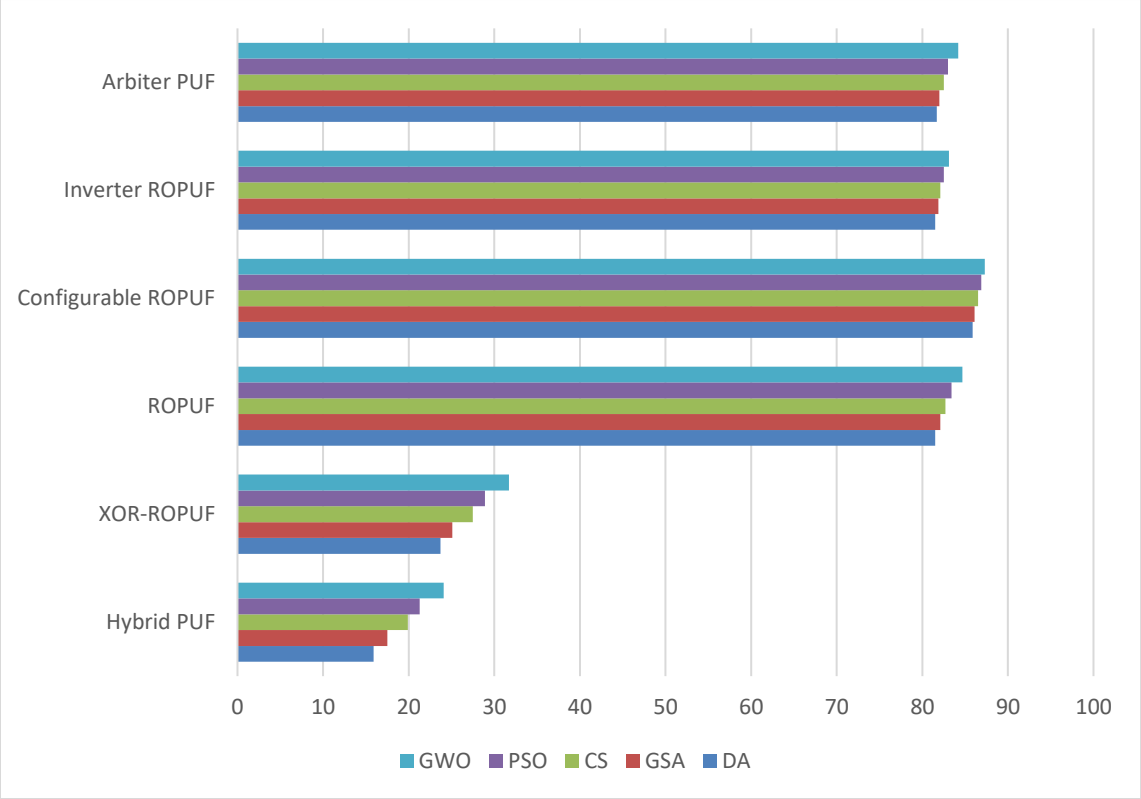

Figure 6-15: Comparison of swarm-based prediction accuracies for PUFs.

From the table, it is evident that the proposed PUF design can resist the different modeling attacks much better than other available designs. When it comes to swarm based algorithms, the proposed design's prediction accuracy ranges from 15.9 % (DA) to 24.1 % (GWO). The best prediction accuracy of 24.1% is observed for the GWO optimization as shown in Figure 6-15.

6.5 Chapter Summary

We present a unique delay-based PUF structure that combines challenge-response pairs (CRPs) from an Arbiter PUF with a XOR based Ring Oscillator to generate responses which are found to be less vulnerable to machine learning modeling attacks. From the results, it is found that the prediction accuracy when different machine learning classifier algorithms are employed to attack the PUF, is drastically reduced and lies in the range of 13.9% to 15.7% for ANN based attacks, whereas the swarm based model accuracy obtained is in the range of 15.9% to 24.1%. Our study indicates that the new design's vulnerability against different machine learning modeling attacks is much less compared to other delay-based PUF designs.

Chapter 7

XOR-ROPUF Application: Hardware-Oriented Security Based Authentication for Internet of Things Devices

The Internet of Things (IoTs) has become in demand nowadays as many embedded devices are connected to the internet to collect a vast amount of data for processing. A substantial amount of this data between IoT devices is confidential information; therefore, attacks on IoT devices are growing, causing challenges to the security of the IoT devices. Physical Unclonable Functions (PUFs) are proposed as a powerful and lightweight solution to secure IoT devices. PUFs authenticate and generate secure cryptographic keys using manufacturing process variations to protect IoT devices from different attacks; this unique identity is based on physical characteristics. Device authentication is an essential task in IoT. In this Chapter, a lightweight XOR-ROPUF based authentication scheme for the security of IoT systems is presented. The proposed management scheme carries out the authentication between the verification authority and the IoT devices to ensure data congeniality and integrity, and thus reduces the risk of cyber attacks.

7.1 Introduction

The need for communication amongst connected devices has led to an increase in demand for the Internet of Things (IoTs). IoT has developed as an expanded network where

millions of devices are connected to the internet to transfer helpful information [98]. IoT refers to the appearance of the network of physical objects that have internet infrastructure, connectivity, and information transfer between these objects and other internet systems to achieve tasks without human interaction [99]. From the most straightforward consumer-based applications such as smartphones and smart homes to complex industry solutions, like the smart power grid and medical devices, the Internet of Things (IoTs) is everywhere and constantly improving how consumers work, as shown in Figure 7-1. IoT devices are becoming widely used in emerging applications such as edge computing, smart and connected cities, medical devices, smart power grids, and intelligent autonomous systems. IoT technology is presumed to be used in various ways and strongly impacts different applications such as personal health monitoring devices, the smart home, and smartphones by providing interconnection and information exchange. They also connect an increasing number of dynamic global information networks consisting of Internet-connected objects, making the IoT a complex growing system [100]. These applications are increasingly integrated into insecure physical environments making the necessary authentication scheme challenging for security in IoT networks and leading to many challenges, like security, privacy, authentications, etc., and need to be protected from the new cyber and physical system attacks [101]. Hardware security is one of the significant challenges in IoT due to the likelihood of these devices communicating in an unsecured system, thus providing an adversary to gain access to the system. Therefore, there is a need to develop a hardware security solution to trust IoT networks [102].

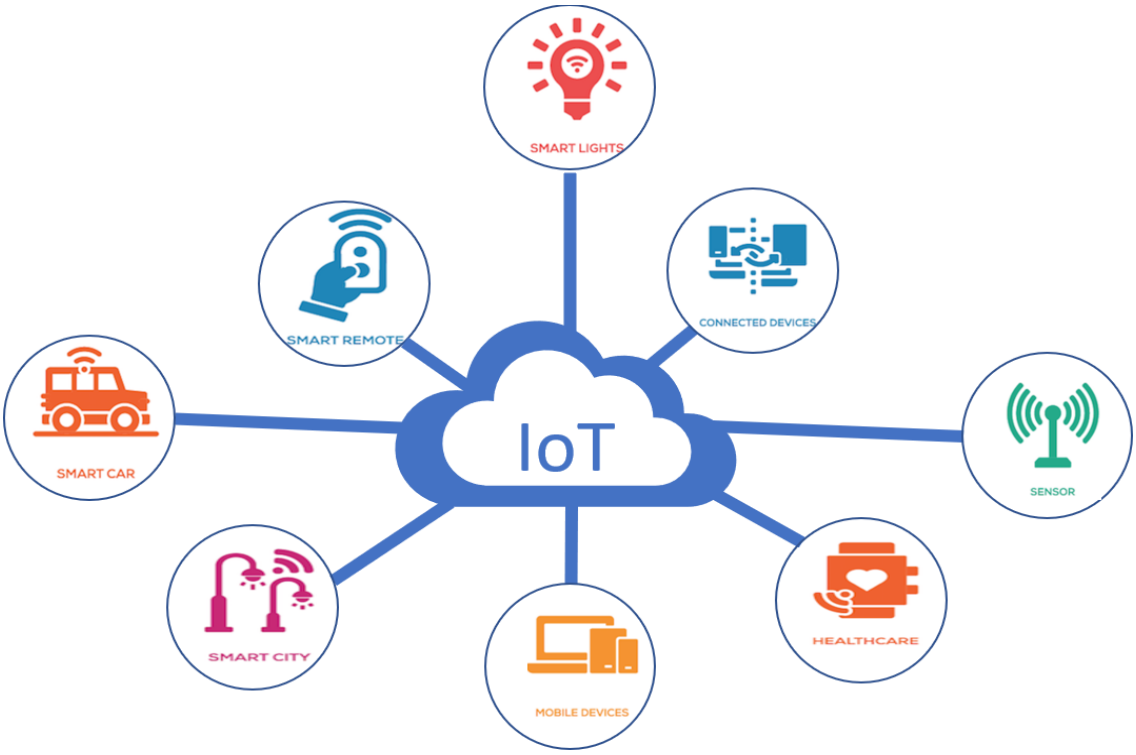

Figure 7-1: The overall picture of Internet of Things devices.

7.2 Physical Unclonable Functions on the Internet of Things

The life process of IoT devices affects different parties and facilities, and thus, various security threats impact these devices. The life-cycle of IoT-based electronic devices is illustrated in Figure 7-2 [103]. The figure shows design specifications, fabrication, test, and deployment of IoT-based consumer electronic devices. As shown in the figure, the life cycle involves multiple parties and facilities, and thus, diverse security threats affect these devices. Further, the major security threats associated with IoT-based and intelligent electronic devices are also shown in Figure 7-2.

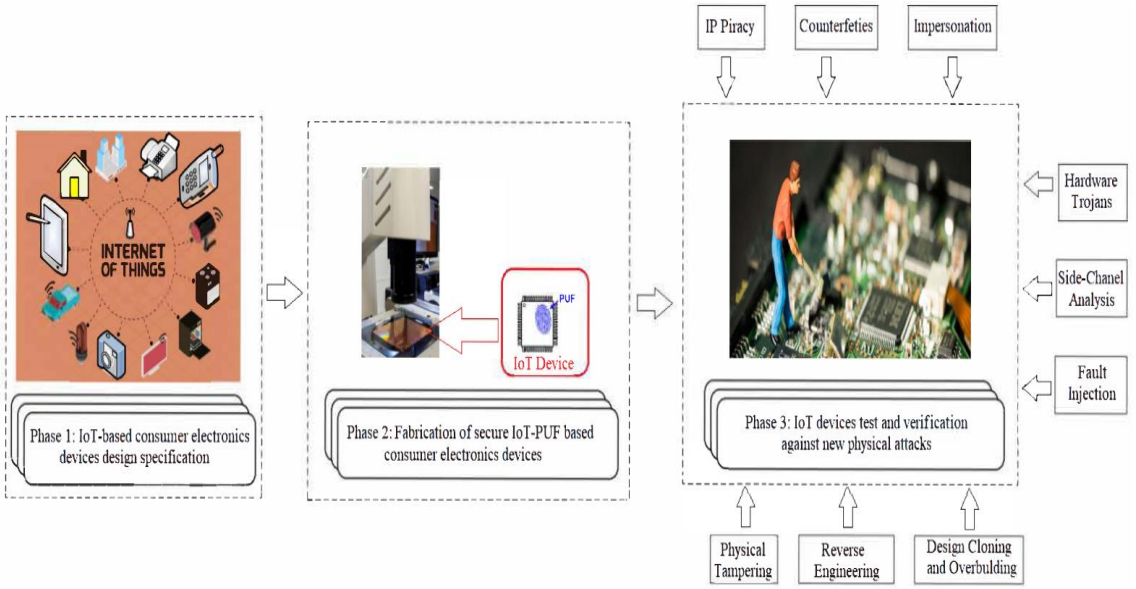

Figure 7-2: The life-cycle of PUF-based IoT and smart electronic device

IoT-based threats have endangered IoT devices' security, operability, reliability, and expansive applications [104]. An adversary can steal design information during manufacturing at an untrusted foundry, and these attacks may lead to leaking secret information from an IoT-based and intelligent electronic device. Security of the Internet of Things (IoTs) is one of the current primary challenges in developing our life and future manufacturing systems. They face complex challenges to secure devices against cyber-attacks. Network authentication is highly required as IoT is growing, and it is essential to have an effective device authentication mechanism to identify the trusted device [105]. Because of the increasing demand for IoT devices, these devices must be inexpensive and challenge the implementation of security functions. Authentication is considered to be the first obstacle to securing IoT systems against various types of attacks. Therefore, using physically unclonable functions (PUF) could solve many IoT security requirements at an acceptable cost [106]. PUFs are low-cost, hardware security primitive for intelligent and

energy-constrained IoT devices. Research for IoT authentications and encryption schemes has proposed the use of PUFs as a low cost solution [107]. Therefore, IoT and smart devices will need to find robust solutions for the IoT devices' security to protect communication over the internet and reduce the new security attacks. Physical Unclonable Functions (PUFs) have been proposed as a lightweight, cost-efficient solution. Implementing a PUF circuit in reconfigurable hardware like a Field Programmable Gate Array (FPGA) ensures secure authentication for IoT devices [108].

7.3 PUF-Based Threats on IoT Devices

Attacks on Internet of Things (IoTs) devices are increasing rapidly; therefore, there are many potential threats for IoT devices that use PUFs for authentication. Possible definitive attacks include reading out private keys from memory, and communication attacks [109]. The main new threat on the PUF-based security system is for an attacker to obtain the ability to provide the correct response for a given challenge. Modeling attacks are an example of the emerging attacks that aim to replicate the behavior of PUF's challenge and response for cloning the secret keys generated by a PUF design [110]. Modeling attacks against PUF-based schemes for IoT devices authentication are classified into different categories:

7.3.1 Machine Learning (ML) attacks

Researchers have found that PUFs are vulnerable to various machine learning modeling attacks; therefore, ML attacks are considered one of the most successful attacks to clone the behavior of PUF design. Several machine learning-based modeling attacks

have been analyzed to mimic the behavior of the PUFs [111]. The efficiency of these algorithms can be determined by the complexity of the different PUF designs.

7.3.2 Man in the Middle Attack

Attackers can eavesdrop by using MITM attacks by recording the network data packets and extracting the information of the CRPs when the system is in operation. In this type of attack, an adversary can extract and record the data packets on the communication network between a server and a device and prevent and store exchanged CRPs [112]. Therefore, after obtaining a set of CRPs, a PUF can be modeled using machine learning.

7.3.3 Side-channel hardware-based attacks

Side-channel attacks use various parameters such as current leakage, voltage variations, and power consumption to establish an attack against semiconductor integrated circuits (IC) and Internet of Things (IoTs) devices [113]. Typical side channel hardware attacks include power analysis side channel, time consumption side channel, electromagnetic side challenge, differential fault analysis side channel and photonic emission side channel analysis. These attacks take advantage of the side challenge parameters to model a robust PUF design.

7.4 Device Authentication in IoT using Delay-based PUF

Device authentication is one of the essential tasks in IoT. The authentication scheme in IoT devices guarantees a more secure and efficiently interoperable alternative to IoT systems. Nevertheless, the remarkable number of smart devices makes it challenging to

secure their authenticated and certified whenever connected to the IoT system. Most existing IoT systems are secured through authentication, ensuring connected IoT devices can access the resource. Here we focus on the authentication of users and devices in IoT using the Physical Unclonable Function (PUF) to enable IoT systems to handle authentic and verified devices.

7.4.1 Implementation of XOR-ROPUF Design

PUF is a robust security mechanism that recognizes a hardware fingerprint, and it provides a unique hardware key depending on the specific device properties. PUFs are commonly used for authentication and secure communication, which fits appropriately into the resource demands of IoT devices. A probable solution can be provided using PUF to generate a unique response for each device due to manufacturing variation. The proposed XOR-ROPUF enhances entropy, allowing the design to generate challenge-response pairs for secure and trusted IoT applications that require robust and lightweight secret and cryptographic key generation. Our lightweight PUF design is a low-cost and efficient hardware security design intended to generate low-cost secret keys for IoT security, including IoT robust authentication. The design of the PUF has been explained earlier in Chapter 5. The design is implemented on Xilinx Artix 7 FPGAs installed on the Diligent Nexys 4 board. Figure 7-3 represents the implemented XOR-ROPUF, the responses generated from the PUF are fed back to the challenge generator. An XOR network takes the challenges from the challenge generator and combines the response to generate a new challenge. The PUF design takes a 16-bit challenge and generates a 16-bit response.

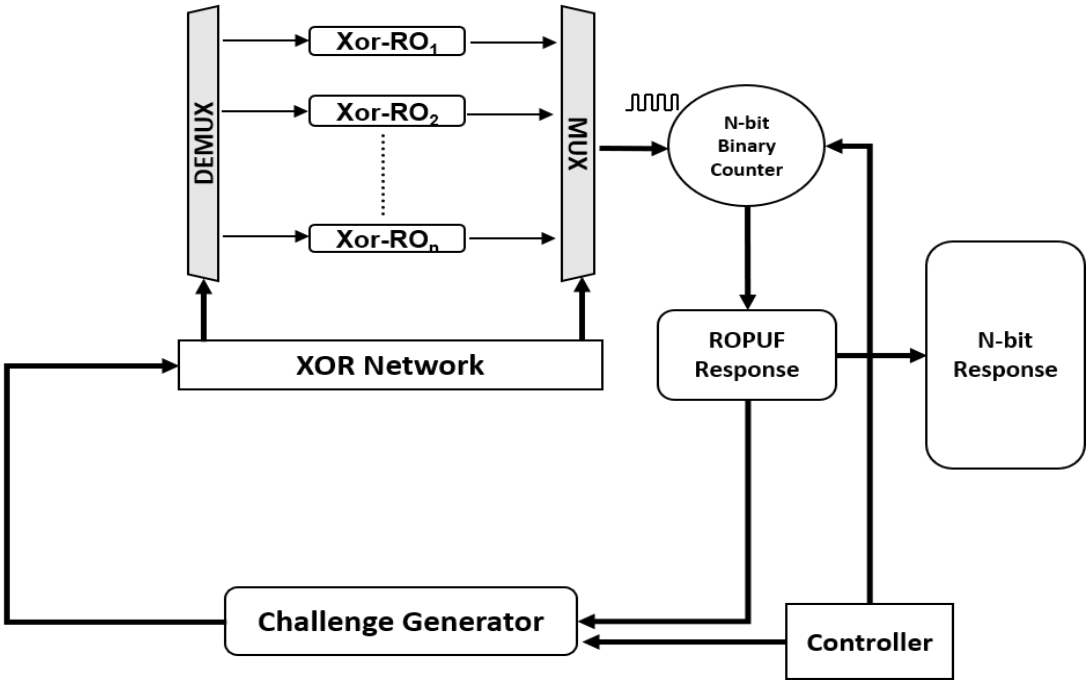

Figure 7-3: XOR-ROPUF Design.

7.4.2 Proposed PUF-Based Authentication Scheme

The device authentication scheme is proposed for IoT applications using a lightweight XOR-ROPUF. The main objective of using the PUF model is to carry out the authentication between the verifier (verification authority) and prover (IoT Device). This scheme presents a suitable technique for authentication using a model of the PUF that provides a compact solution for the authentication of IoT systems. Figure 7-4 shows the proposed scheme between the IoT devices and the authentication server to ensure data congeniality and integrity. The proposed PUF-based IoT device authentication scheme reduces the risk of authentication vulnerability. The proposed technique can be performed by storing and updating challenge and response pairs through communication with the IoT devices to be authenticated.

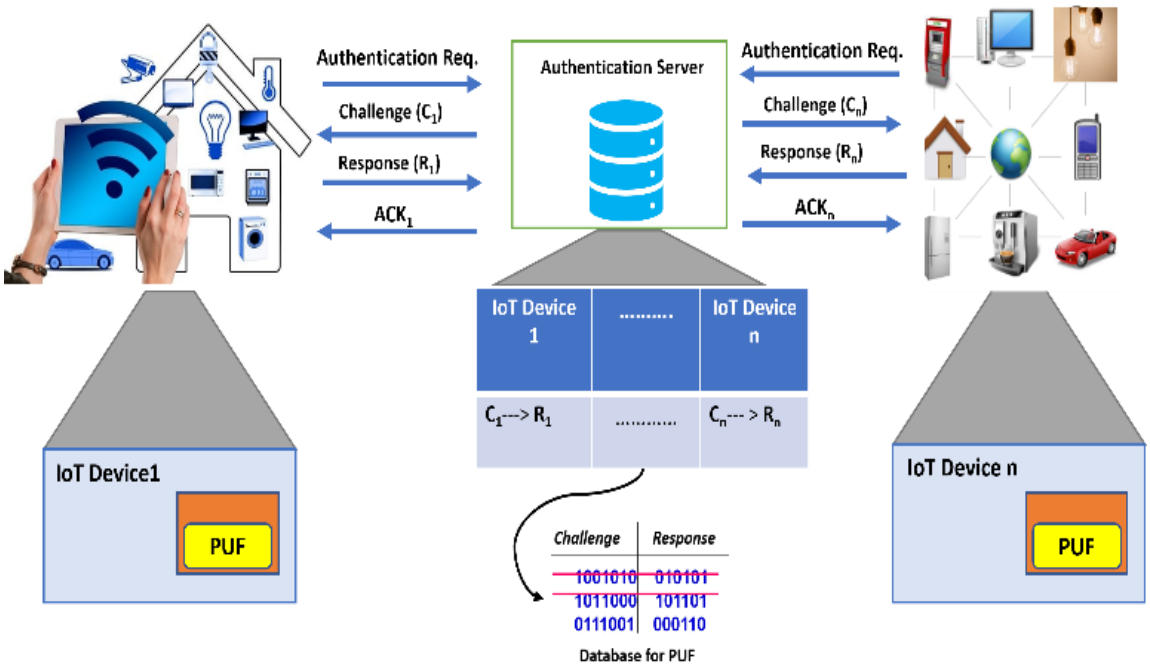

Figure 7-4: The PUF-based IoT Device Authentication scheme.

The PUF-based scheme eliminates some drawbacks because it is a low-cost device authentication solution to identify the trusted hardware. It secures communication among the devices using a lightweight system. IoT device identity can be verified through verification authority to establish trust in an IoT device. The PUF-based authentication scheme consists of a two-step process: enrollment and verification, which ensures the authentication.

- The first phase of the authentication process is 'Enrollment': During the enrollment phase, the IoT device embedding the PUF is directly connected to the server (verification authority). The authentication server sends challenges, and the IoT device embedding the PUF circuit sends back the responses. The authentication server stores all challenge and response pairs in a secure database by

taking and performing each entity. During the enrollment phase, the IDs of all entities are recorded in the database by the verifier. Then the device can be deployed. Figure 7-5 shows how the PUF chip is enrolled individually with the authentication server. The challenge-response pairs (CRPs) are collected and stored during enrollment. The authentication server has a database that stores all challenges for each PUF. The server sends one challenge (Ci) to each PUF and records the response (Ri) generated from each PUF in the database.

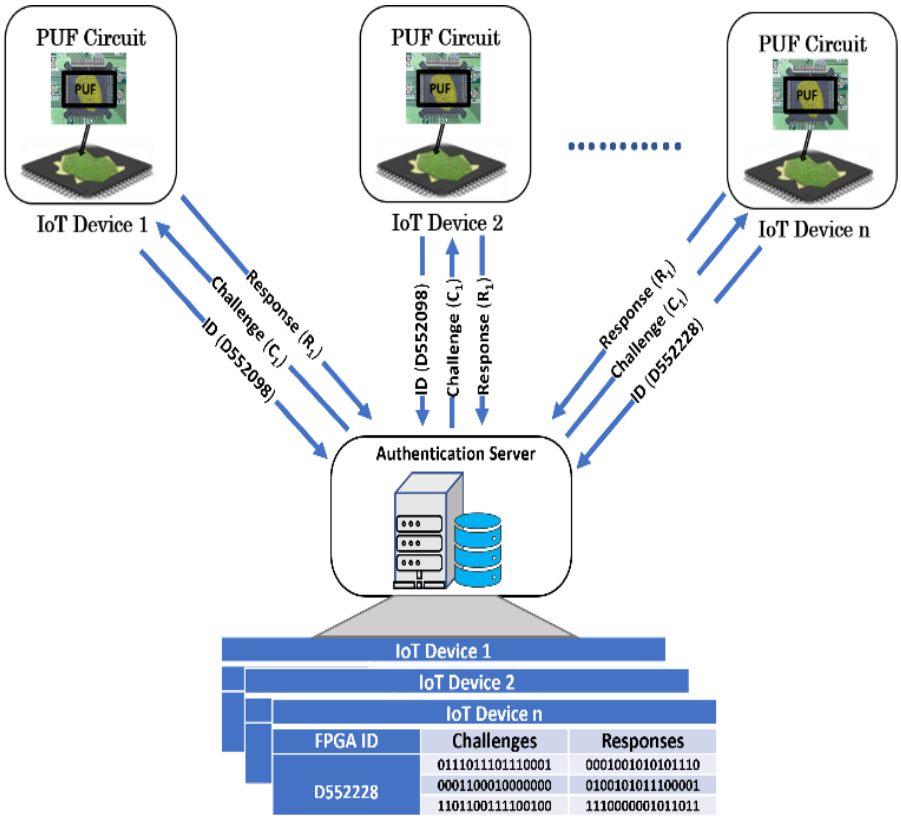

Figure 7-5: Enrollment phase of PUF in authentication scheme.

- The second phase is Verification: During the verification phase, the IoT device needs to be authenticated by the verification authority. Each IoT device is being identified by its unique ID. The server receives the entity's ID, sends an arbitrary

PUF challenge to the IoT device, and checks against the previously registered value in its database. The authentication scheme to authenticate the IoT devices is explained in Figure 7-6.

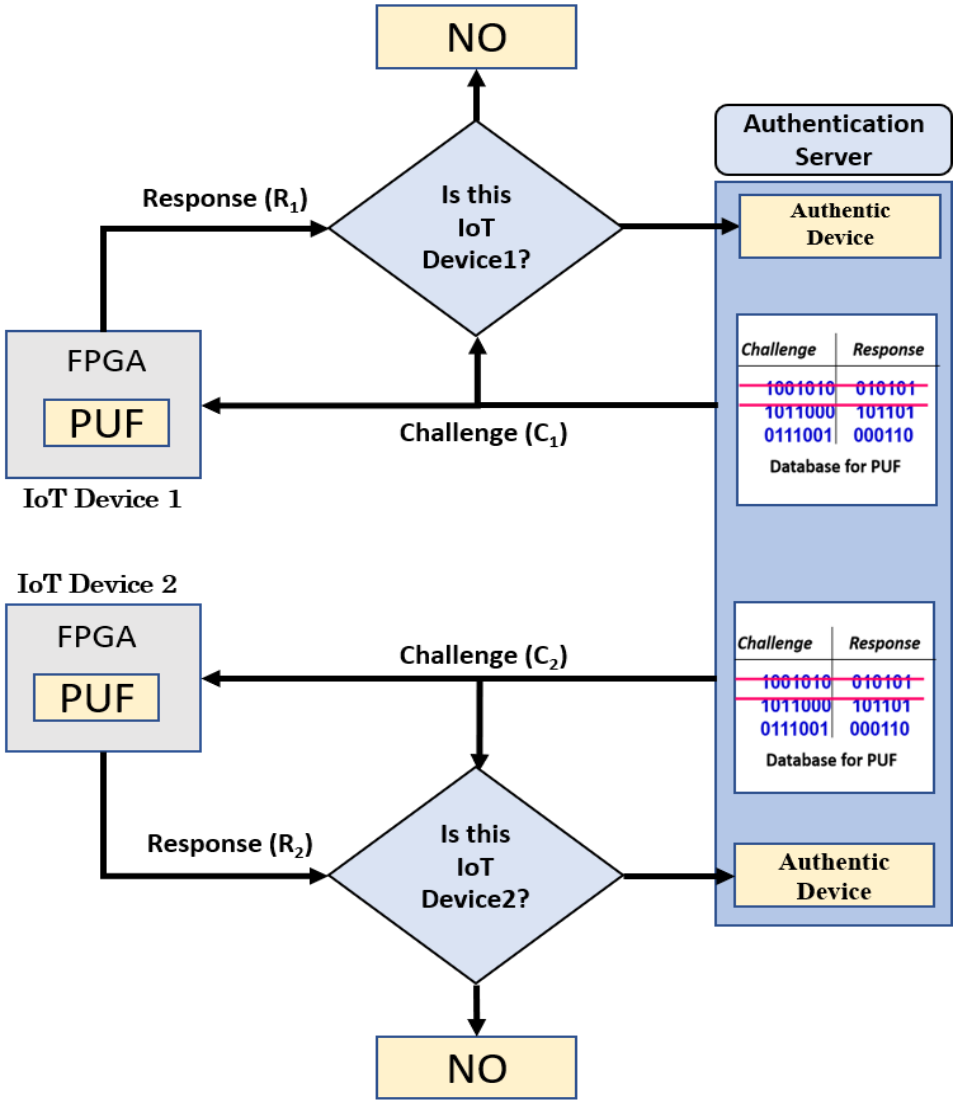

Figure 7-6: PUF based device verification phase.

The CRPs obtained from different Artix 7 FPGAs are listed in Table 7.1. The IoT device is authenticated if the measured response matches the stored response in the verification

authority database. Only if the two values match can the verification authority conclude that the IoT device is authentic.

Table 7.1: Challenge- Response Pairs of Different FPGAs

IoT Device	FPGA ID	Challenges	Responses
1	D552098	0111011101110001 0001100010000000 1101100111100100	1000000000110101 0001110011001101 1100000100110110
2	D552320	0111011101110001 0001100010000000 1101100111100100	1010111000000010 0010010010001010 1100001001110101
3	D552224	0111011101110001 0001100010000000 1101100111100100	1010011001010001 1101011001001001 0011111010001010
…….	…….	…….	……
n	D552228	0111011101110001 0001100010000000 1101100111100100	0001001010101110 0100101011100001 1110000001011011

Our proposed PUF based device authentication scheme is well suited for IoT devices with good reliability. The different parameters (Uniformity, Uniqueness, Bit aliasing, Reliability) for the XOR-ROPUF have been measured and are shown in Table 7.2. However, ten different Artix-7 FPGAs are used for data acquisition for calculating the parameters. For comparing the data with other similar designs, PUF metrics from other designs are included in Table 7.2. The performance of the PUF with quantified parameters and results show that our proposed device authentication scheme is suited for hardware security.

Table 7.2: ROPUF Parameters (Performance Metrics for Different PUFs)

Design	Uniformity	Uniqueness	Bit-Aliasing	Reliability
XOR- ROPUF	50.76%	49.40%	48.35%	99.8%
Inverter ROPUF [114]	47.02%	45.15%	47.20%	98.0%
ROPUF [115]	50.56%	47.24%	50.56%	99.14%
APUF [115]	55.69%	7.20%	19.57%	99.76%

The hardware implementation of the XOR-ROPUF leverages the inherent physical characteristics of the FPGA to secure, reliable, and trusted IoT applications. The proposed secured PUF based scheme eliminates some drawbacks of traditional approaches and explains how PUFs can be used to build low-cost authentication schemes. The proposed PUF-based security enhancement scheme addresses the need for a high-security IoT network.

7.5 Chapter Summary

An authentication scheme is proposed for the security of IoT systems using a lightweight XOR-ROPUF. The proposed management scheme carries out the authentication between the verification authority, the authentication server, and the IoT devices to ensure data congeniality and integrity. The proposed XOR-ROPUF based scheme is a low-cost device authentication solution to identify the trustworthiness of the devices. It secures the communication exchange amongst the devices using a lightweight system and reduces the risk of authentication vulnerability.

Chapter 8

Conclusion

8.1 Summary and Conclusions

Various Machine Learning based attack models have been used recently to breach the security of PUFs. In this research, we have introduced a novel implementation of Artificial Neural Network-based modeling attack on various PUFs using different swarm intelligence algorithms. From the results, it is observed that the swarm intelligence algorithms produce better response prediction accuracies (71.1% - 99.3%) when compared to other well-known algorithms. Amongst the SI algorithms, the GWO algorithm performs the best in predicting the CRPs. It is observed that the Configurable ROPUF is the most vulnerable and its response can be predicted with an accuracy of 88.3% when the GWO is used. To the best of our knowledge, swarm-based algorithms have not been investigated earlier to test the security of PUFs.

This research also presents a comprehensive study and analysis of the vulnerability of different delay- based PUF designs to various machine learning modeling attacks. From the results, it can be observed that the performance of the algorithms is quite similar in nature, however, KNN performs better than the other algorithms in predicting the CRPs.

The best accuracies observed for the algorithms LR, DT, RF, KNN, SVM, KSVM, and NB are 88.9%, 90.3%, 91.5%, 92.3%, 91.1%, 91.9%, and 86.1%, respectively. The results show that the ANN-based algorithms produce better response prediction accuracy results compared to other machine learning classifiers. For the ANN-based modeling, it is observed that the accuracy range for response prediction is between 68.0% to 94.1%, with the ANN-based modeling using the Nadam optimizer the best accuracy.

In order to enhance the security of the PUF, a new delay-based XOR-ROPUF design is proposed, which generates an 'n' bit response for an 'n' bit challenge. When it comes to classifier-based algorithms, the maximum prediction accuracy for the new design for an 'n' bit response is 20.7% whereas the maximum accuracy obtained is 84.3% for an 'n' bit response Configurable ROPUF. From the analysis of ANN-based modeling attacks on different PUF designs, it is observed that the XOR-ROPUF prediction accuracy significantly reduces from 86.7% to 23.5%. Also, from the results, it is evident that the other PUF structures are vulnerable to swarm intelligence-based modeling attacks with prediction accuracies ranging from 81.5% - 87.3%. In case of the XOR-ROPUF, it is noted that the models are unable to predict the responses with high prediction accuracy. The best prediction accuracy of 31.7% is observed for the GWO optimization. In addition, a novel Hybrid PUF is being proposed in this research. This unique delay-based Hybrid PUF for thwarting ML attacks will be obtained by combining an Arbiter PUF with an XOR-ROPUF. From the results, it is found that the prediction accuracy when different machine learning classifier algorithms are employed to attack the PUF, is drastically reduced and lies in the range of 13.9% to 15.7% for ANN based attacks, whereas the swarm based model accuracies obtained is in the range of 15.9% to 24.1%. Our study indicates that the

new design's vulnerability against different machine learning modeling attacks is much less compared to other delay-based PUF designs; furthermore, the designs are scalable and can be adapted to new ML and SI attacks.

Finally, as an industrial application, an authentication scheme is proposed for the security of IoT systems using a lightweight XOR-ROPUF. The proposed management scheme carries out the authentication between the verification authority, the authentication server, and the IoT devices to ensure data congeniality and integrity. The proposed XOR-ROPUF based scheme implements a low-cost device authentication solution for identifying the trusted hardware, securing communication among the devices using a lightweight system, and reducing the risk of authentication vulnerability.

8.2 Contributions

The major contributions of this research are as follows:

- Implementation of a novel Artificial Neural Network-based attack model for studying the vulnerability of PUFs to various swarm intelligence algorithms. To the best of our knowledge, these algorithms have not been used before in studying the vulnerability of PUFs to ANN-based attacks.
- Machine Learning Vulnerability analysis of different delay based PUF using different ML classifiers to study PUFs vulnerability to these ML attacks
- Use of Artificial Neural Network based modeling attacks on various PUFs using different optimizers to test their resiliency to these attacks.

- Design of a novel XOR-ROPUF capable of thwarting machine learning attacks. This is achieved by feeding back the bit responses from the oscillator output to the Challenge Generator for generating the next challenge vector. This feedback technique makes the design more secure from hackers who can use both invasive and non-invasive techniques to steal the CRP data.
- Design of a unique delay-based PUF structure that combines an Arbiter PUF with a XOR- ROPUF for thwarting machine learning attacks.
- Development of an authentication scheme for the security of IoT systems using a lightweight XOR-ROPUF. The proposed management scheme carries out the authentication between the verification authority, the authentication server, and the IoT devices to ensure data congeniality and integrity.

8.3 Future Work

The research work done in this dissertation can be extended to the following topics:

- Analysis of the weak entropy of existing silicon PUFs and improving techniques to thwart modeling attacks
- Development of a new scheme for strong and robust PUFs on an FPGAs that generates a more extensive set of CRPs.
- Application of Swarm intelligence based ANN algorithms method can be used for improving the performance metrics of PUFs and for developing countermeasures against modeling attacks.

References

[1] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain", Proceedings of the IEEE, vol. 102, no. 8, pp. 1207-1228, Aug 2014

[2] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan attacks: Threat analysis and countermeasures", Proc. IEEE, vol. 102, no. 8, pp. 1229-1247, Aug. 2014

[3] Tehranipoor, Mohmmad, Hassan Salmani, and Xuehui Zhang. "Integrated circuit authentication." Switzerland: Springer, Cham. doi 10 (2014): 978-3.

[4] F .Koushanfar, “Hardware Metering: A Survey In: Introduction to Hardware Security and Trust,” pp. 103-122. Springer, 2012.

[5] G. Suh and S. Devadas, “Physical Unclonable Functions for Device Authentication and Secret Key Generation,” in DAC, pp. 9-14, 2007.

[6] C. Herder, M. D. Yu, F. Koushanfar and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," in Proc. Of IEEE, vol. 102, no. 8, pp. 1126-1141, Aug. 2014A.

[7] S. Gören et al., "Partial bitstream protection for low-cost FPGAs with physical unclonable function obfuscation and dynamic partial self reconfiguration", Elsevier Computers & Electrical Engineering, vol. 39, no. 2, pp. 386-397, Feb. 2013.

[8] N. A. Hazari, F. Alsulami and M. Niamat, "FPGA IP Obfuscation Using Ring Oscillator Physical Unclonable Function," NAECON 2018 - IEEE National Aerospace and Electronics Conference, Dayton, OH, 2018, pp. 105-108.

[9] F. Amsaad, T. Hoque, and M. Niamat, “Analyzing the Performance of a Configurable ROPUF controlled by Programmable XOR Gates,”in Midwest Symposium on Circuits and Systems, pp. 1-4, 2015.

[10] M. Choudhury, N. Pundir, M. Niamat and M. Mustapa, "Analysis of a novel stage configurable ROPUF design," 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, 2017, pp. 942-945.

[11] Sölter, Jan. Cryptanalysis of electrical PUFs via machine learning algorithms. Diss. MSc thesis, Technische Universität München, 2009.

[12] U. Rhrmair, F. Sehnke, J. Slter, G. Dror, S. Devadas, and J. Schmidhuber, “Modeling attacks on physical unclonable functions,” in Proceedings of the 17th ACM conference on Computer and communications security, 2010, pp. 237-249.

[13] U. Ruhrmair, J. Solter, 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.

[14] J. Delvaux, "Machine-Learning Attacks on PolyPUFs, OB-PUFs, RPUFs, LHS-PUFs, and PUF–FSMs," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 8, pp. 2043-2058, Aug. 2019.

[15] F. Ganji, S. Tajik and J.-P. Seifert, "PAC Learning of Arbiter PUFs" in Security Proofs for Embedded Systems-PROOFS, Springer, 2014.

[16] J. Delvaux and I. Verbauwhede, "Fault Injection Modeling Attacks on 65 nm Arbiter and RO Sum PUFs via Environmental Changes," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 61, no. 6, pp. 1701-1713, June 2014.

[17] J. Shi, Y. Lu and J. Zhang, "Approximation Attacks on Strong PUFs," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[18] J. Delvaux and I. Verbauwhede, "Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise," 2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), Austin, TX, 2013, pp. 137-142.

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

[20] M. Khalafalla and C. Gebotys, "PUFs Deep Attacks: Enhanced modeling attacks using deep learning techniques to break the security of double arbiter PUFs," 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 2019, pp. 204-209.

[21] Q. Guo, J. Ye, Y. Gong, Y. Hu and X. Li, "Efficient Attack on Non-linear Current Mirror PUF with Genetic Algorithm," 2016 IEEE 25th Asian Test Symposium (ATS), Hiroshima, 2016, pp. 49-54.

[22] I. Saha, R. R. Jeldi and R. S. Chakraborty, "Model building attacks on Physically Unclonable Functions using genetic programming," 2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), Austin, TX, 2013, pp. 41-44.

[23] Xu, Yunhao, et al. "Mathematical Modeling Analysis of Strong Physical Unclonable Functions." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2020).

[24] S. Kumar and M. Niamat, "Machine Learning based Modeling Attacks on a Configurable PUF," NAECON 2018 - IEEE National Aerospace and Electronics Conference, Dayton, OH, 2018, pp. 169-173

[25] Holland, John H. "Genetic algorithms." Scientific american 267.1 (1992): 66-73.

[26] Koza, John R., and John R. Koza. Genetic programming: on the programming of computers by means of natural selection. Vol. 1. MIT press, 1992.

[27] Hansen, Nikolaus, Sibylle D. Müller, and Petros Koumoutsakos. "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)." Evolutionary computation 11.1 (2003): 1-18.

[28] Storn, Rainer, and Kenneth Price. "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces." Journal of global optimization 11.4 (1997): 341-359

[29] Rashedi, Esmat, Hossein Nezamabadi-Pour, and Saeid Saryazdi. "GSA: a gravitational search algorithm." Information sciences 179.13 (2009): 2232-2248.

[30] Erol, Osman K., and Ibrahim Eksin. "A new optimization method: big bang–big crunch." Advances in Engineering Software 37.2 (2006): 106-111.

[31] Formato, R. A. "Central force optimization: a new metaheuristic with applications in applied electromagnetics. Prog Electromagn Res 77: 425–491." (2007).

[32] Shah-Hosseini, Hamed. "Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation." International Journal of Computational Science and Engineering 6.1-2 (2011): 132-140.

[33] Webster, Barry, and Philip J. Bernhard. A local search optimization algorithm based on natural principles of gravitation. 2003.

[34] Kaveh, A., and Siamak Talatahari. "A novel heuristic optimization method: charged system search." Acta mechanica 213.3 (2010): 267-289.

[35] S. Selvaraj and E. Choi. “Survey of Swarm Intelligence Algorithms,” In Proceedings of the 3rd International Conference on Software Engineering and Information Management (ICSIM ’20). Association for Computing Machinery, New York, NY, USA, 69–73, 2020

[36] Kennedy, James, and Russell Eberhart. "Particle swarm optimization." In Proceedings of ICNN'95-International Conference on Neural Networks, vol. 4, pp. 1942-1948. IEEE, 1995.

[37] Dorigo, Marco, Mauro Birattari, and Thomas Stutzle. "Ant colony optimization." IEEE computational intelligence magazine 1.4 (2006): 28-39.

[38] Yang, Xin-She, and Suash Deb. "Cuckoo search via Lévy flights." 2009 World congress on nature & biologically inspired computing (NaBIC). IEEE, 2009.

[39] Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis. "Grey wolf optimizer." Advances in engineering software 69 (2014): 46-61.

[40] A. Kumar and S. Chakarverty, “Design optimization for reliable embedded system using Cuckoo search,” in Proceedings of the 3rd International Conference on Electronics Computer Technology (ICECT ’11), pp. 264–268, April 2011.

[41] Abdulgader, Musbah, Srivathsan Lakshminarayanan, and Devinder Kaur. "Efficient energy management for smart homes with grey wolf optimizer." 2017 IEEE International Conference on Electro Information Technology (EIT). IEEE, 2017.

[42] Liu, Haiqiang, et al. "An intelligent grey wolf optimizer algorithm for distributed compressed sensing." Computational intelligence and neuroscience 2018 (2018).

[43] P. Sedcole and P. Y. K. Cheung, “Within-die delay variability in 90nm FPGAs and beyond," 2006 IEEE International Conference on Field Programmable Technology, Bangkok, 2006, pp. 97-104. doi: 10.1109/FPT.2006.270300.

[44] D. Blaauw, K. Chopra, A. Srivastava and L. Sche_er, “Statistical Timing Analysis: From Basic Principles to State of the Art," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 4, pp.589-607, April 2008.

[45] Maes, Roel. "Physically unclonable functions: Properties." Physically Unclonable Functions. Springer, Berlin, Heidelberg, 2013. 49-80.

[46] R. Maes and I. Verbauwhede, “Physically unclonable functions: A study on the state of the art and future research directions,” in Towards Hardware-Intrinsic Security. Springer, 2010, pp. 3‑37.

[47] M. Roel, “Physically unclonable functions: Constructions, properties and applications,” Katholieke Universiteit Leuven, Belgium, 2012.

[48] R. Maes, “Physically Unclonable Functions: Construction, Properties and Applications,” New York, NY, USA: Springer, 2013

[49] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, “A technique to build a secret key in integrated circuits for identification and authentication applications,” in Symposium on VLSI Circuits, Jun 2004, pp. 176 – 179, digest of Technical Papers.

[50] S.S. Kumar, J. Guajardo, R. Maesyz, G.J. Schrijen, and P. Tuyls, "The butterfly PUF", HOST'08, pp. 67-70, 2008.

[51] R. Pappu: Physical One-Way Functions. PhD Thesis, Massachusetts Institute of Technology, 2001.

[52] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld: “Physical One-Way Functions,” Science, vol. 297, 2002

[53] P. Tuyls, G. Schrijen, B. kori, J. Geloven, N. Verhaegh, and R. Wolters, “Readproof Hardware from Protective Coatings", in Crypto-graphic Hardware and Embedded Systems (Lecture Notes in Computer Science), vol. 4249. Berlin, Germany: Springer-Verlag, 2006, pp. 369-383.

[54] A. Maiti, V. Gunreddy, P. Schaumont, “A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions," in Embedded System Design with FPGAs, Springer international publication, 2012. ISBN:978-1-4614-1361-5

[55] A. Maiti, J. Casarona, L. McHale and P. Schaumont, “A large scale characterization of RO-PUF," 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), Anaheim, CA, 2010, pp. 94-99. doi:10.1109/HST.2010.5513108

[56] Jolliffe, Ian T., and Jorge Cadima. "Principal component analysis: a review and recent developments." Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 374.2065 (2016): 20150202.

[57] Abdi, Hervé, and Lynne J. Williams. "Principal component analysis." Wiley interdisciplinary reviews: computational statistics 2.4 (2010): 433-459.

[58] Fisher, Ronald A. "The use of multiple measurements in taxonomic problems." Annals of eugenics 7.2 (1936): 179-188.

[59] Xanthopoulos, Petros, Panos M. Pardalos, and Theodore B. Trafalis. "Linear discriminant analysis." Robust data mining. Springer, New York, NY, 2013. 27-33

[60] Balakrishnama, Suresh, and Aravind Ganapathiraju. "Linear discriminant analysis-a brief tutorial." Institute for Signal and information Processing 18.1998 (1998): 1-8.

[61] Park, Cheong Hee, and Haesun Park. "A comparison of generalized linear discriminant analysis algorithms." Pattern Recognition 41.3 (2008): 1083-1097.

[62] Schölkopf, Bernhard, Alexander Smola, and Klaus-Robert Müller. "Kernel principal component analysis." International conference on artificial neural networks. Springer, Berlin, Heidelberg, 1997.

[63] Schölkopf, Bernhard, Alexander Smola, and Klaus-Robert Müller. "Nonlinear component analysis as a kernel eigenvalue problem." Neural computation 10.5 (1998): 1299-1319.

[64] Xu, Yong, et al. "A method for speeding up feature extraction based on KPCA." Neurocomputing 70.4-6 (2007): 1056-1061.

[65] Yang, Xin-She, and Mehmet Karamanoglu. "Swarm intelligence and bio-inspired computation: an overview." Swarm intelligence and bio-inspired computation. Elsevier, 2013. 3-23

[66] Brezočnik, Lucija, Iztok Fister, and Vili Podgorelec. "Swarm intelligence algorithms for feature selection: a review." Applied Sciences 8.9 (2018): 1521.

[67] Mirjalili, S. (2015b) 'Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi- objective problems', Neural Computing and Applications, doi:10.1007/s00521-015-1920-1.

[68] Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In Proceedings of the 14th annual conference on computer graphics and interactive techniques. In SIGGRAPH '87 (pp.25–34). New York, NY, USA: ACM. Sambandam, R. K., & Jayaraman, S. (2016).

[69] Yang, X.-S. (2010a). Firefly algorithm, lévy flights and global optimization. In M. Bramer, R. Ellis, & M. Petridis (Eds.), Research and development in intelligent systems XXVI: Incorporating applications and innovations in intelligent systems XVII (pp. 209–218)). London: Springer London.

[70] Shi, Yuhui. "Particle swarm optimization: developments, applications and resources." Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546). Vol. 1. IEEE, 2001.

[71] A. Oun, N. A. Hazari and M. Y. Niamat, "Analysis of Swarm Intelligence Based ANN Algorithms for Attacking PUFs," in IEEE Access, vol. 9, pp. 121743-121758, 2021, doi: 10.1109/ACCESS.2021.3109235.

[72] Kohavi, Ron. "A study of cross-validation and bootstrap for accuracy estimation and model selection." Ijcai. Vol. 14. No. 2. 1995.

[73] A. Oun and M. Niamat, "Defense Mechanism Vulnerability Analysis of Ring Oscillator PUFs Against Neural Network Modeling Attacks using the Dragonfly

Algorithm," 2020 IEEE International Conference on Electro Information Technology (EIT), 2020, pp. 378-382, doi: 10.1109/EIT48999.2020.9208320.

[74] O. Kommerling and M. G. Kuhn, “Design principles for tamper-resistant smartcard processors,” in Proc. USENIX Workshop Smartcard Technology, 1999, pp. 9–20.

[75] F. Ganji, S. Tajik, and J.P. Seifert, " Let me prove it to you: RO PUFs are provably learnable." In Proc of The 18th Annual International Conference on Information Security and Cryptology (2015)

[76] M. Rostami, M. Majzoobi, F. Koushanfar, D. S. Wallach, and S. Devadas, “Robust and reverse-engineering resilient PUF authentication and keyexchange by substring matching,” IEEE Transactions on Emerging Topics in Computing, vol. 2, no. 1, pp. 37–49, 2014

[77] Y. Gao, G. Li, H. Ma, S. F. Al-Sarawi, O. Kavchei, D. Abbott, D. C. Ranasinghe, "Obfuscated challenge-response: A secure lightweight authentication mechanism for PUF-based pervasive devices", IEEE International Conference on Pervasive Computing and Communication Workshops, pp. 1-6, 2016.

[78] H. Gu and M. Potkonjak, "Securing interconnected PUF network with reconfigurability," 2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), Washington, DC, 2018, pp. 231-234.

[79] Dankmar Bohning. 1992. Multinomial logistic regression algorithm. Ann. Inst. Stat. Math. 44, 1 (1992), 197‑200.

[80] Sotiris B. Kotsiantis. 2013. Decision trees: A recent overview. Artific. Intell. Rev. 39, 4 (2013), 261‑283.

[81] QingyaoWu, Yunming Ye,Haijun Zhang, Michael K. Ng, and Shen-Shyang Ho. 2014. ForesTexter: An efficient random forest algorithm for imbalanced text categorization. Knowl.-based Syst. 67 (2014), 105‑116.

[82] Thierry Denoeux. 2008. A k-nearest neighbor classification rule based on Dempster-Shafer theory. In Classic Works of the Dempster-Shafer Theory of Belief Functions. Springer, Berlin, 737‑760.

[83] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. Mach. Learn. 20, 3 (1995), 273‑297.

[84] J. Vaidya, M. Kantarcioglu, C. Clifton Privacy-preserving Naive Bayes classification VLDB J., 17 (4) (2008), pp. 879-898

[85] N. A. Hazari, A. Oun and M. Niamat, "Analysis and Machine Learning Vulnerability Assessment of XOR-Inverter based Ring Oscillator PUF Design," 2019 IEEE 62nd

International Midwest Symposium on Circuits and Systems (MWSCAS), 2019, pp. 590-593, doi: 10.1109/MWSCAS.2019.8885037.

[86] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, and Greg S. Corrado, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems.

[87] M. I. Velazco and C. Lyra, ``Optimization with neural networks trained by evolutionary algorithms," in Proc. Int. Joint Conf. Neural Netw. (IJCNN), vol. 2, 2002, pp. 1516_1521.

[88] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

[89] Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms.

[90] M. Friedman, ``The use of ranks to avoid the assumption of normality implicit in the analysis of variance," J. Amer. Statist. Assoc., vol. 32, no. 200, pp. 675_701, Dec. 1937.

[91] D. J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, 4th ed. London, U.K.: Chapman & Hall, 2006.

[92] J. Derrac, S. García, D. Molina, and F. Herrera, ``A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," Swarm Evol. Comput., vol. 1, no. 1, pp. 3_18, Mar. 2011.

[93] S. Holm, ``A simple sequentially rejective multiple test procedure," Scand. J. Statist., vol. 6, pp. 65_70, Jan. 1979.

[94] Ulrich Ruhrmair and Jan Solter, “PUF modeling Attacks: An introduction and overview,” in Design,Automation and Test in Eurome Conference and Exhibition (DATE) on, pages 1-6, 2014.

[95] Hazari, Noor Ahmad, Ahmed Oun, and Mohammed Niamat. "Machine Learning Vulnerability Analysis of FPGA-based Ring Oscillator PUFs and Counter Measures." ACM Journal on Emerging Technologies in Computing Systems (JETC) 17.3 (2021): 1-20.

[96] Trimberger, Stephen M. Steve. "Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology: This Paper Reflects on How Moore's Law Has Driven the Design of FPGAs Through Three Epochs: the Age of Invention, the Age of Expansion, and the Age of Accumulation." IEEE Solid-State Circuits Magazine 10.2 (2018): 16-29.

[97] A. Oun and M. Niamat, "Design of a Delay-Based FPGA PUF Resistant to Machine Learning Attacks," 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), 2021, pp. 865-868, doi: 10.1109/MWSCAS47672.2021.9531815.

[98] Evans, Dave. "The internet of things: How the next evolution of the internet is changing everything." CISCO white paper 1.2011 (2011): 1-11.

[99] Al-Fuqaha, Ala, et al. "Internet of things: A survey on enabling technologies, protocols, and applications." IEEE communications surveys & tutorials 17.4 (2015): 2347-2376.

[100] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," IEEE Internet of Things Journal, vol. 1, pp. 22-32, 2014.

[101] P. Gope and B. Sikdar, ``Lightweight and privacy-preserving two-factor authentication scheme for IoT devices," IEEE Internet Things J., vol. 6, no. 1, pp. 580_589, Feb. 2019.

[102] A. Sengupta and S. Kundu, ``Guest editorial securing IoT hardware: Threat models and reliable, low-power design solutions," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 12, pp. 3265_3267, Dec. 2017.

[103] Amsaad, Fathi, et al. "Enhancing the Performance of Lightweight Configurable PUF for Robust IoT Hardware-Assisted Security." IEEE Access 9 (2021): 136792-136810.

[104] C. Dong, G. He, X. Liu, Y. Yang, and W. Guo, ``A multi-layer hardware Trojan protection framework for IoT chips," IEEE Access, vol. 7, pp. 23628_23639, 2019.

[105] Babaei, Armin, and Gregor Schiele. "Spatial reconfigurable physical unclonable functions for the Internet of Things." International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage. Springer, Cham, 2017.

[106] X. Xin, J. Kaps, and K. Gaj, ``A configurable ring-oscillator-based PUF for Xilinx FPGAs," in Proc. 14th Euromicro Conf. Digit. Syst. Design, Oulu, Finland, 2011, pp. 651_657.

[107] B. Zhao, P. Zhao, and P. Fan, ``EPUF: A lightweight double identity verification in IoT," Tsinghua Sci. Technol., vol. 25, no. 5, pp. 625_635, Oct. 2020.

[108] Mukhopadhyay, Debdeep. "PUFs as promising tools for security in internet of things." IEEE Design & Test 33.3 (2016): 103-115.

[109] Wallgren, Linus, Shahid Raza, and Thiemo Voigt. "Routing attacks and countermeasures in the RPL-based internet of things." International Journal of Distributed Sensor Networks 9.8 (2013): 794326.

[110] Helfmeier, Clemens, et al. "Cloning physically unclonable functions." 2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST). IEEE, 2013.

[111] Ganji, Fatemeh, Shahin Tajik, and Jean-Pierre Seifert. "A Fourier analysis based attack against physically unclonable functions." International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2018.

[112] Halak, Basel, Mark Zwolinski, and M. Syafiq Mispan. "Overview of PUF-based hardware security solutions for the Internet of Things." 2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS). IEEE, 2016.

[113] S. Tajik, E. Dietz, S. Frohmann, H. Dittrich, D. Nedospasov, C. Helfmeier, J.-P. Seifert, C. Boit, and H.-W. Hubers, ``Photonic side-channel analysis of arbiter PUFs," J. Cryptol., vol. 30, no. 2, pp. 550_571, 2017.

[114] Mustapa Muslim, Mohammed Y. Niamat, Atul Prasad Deb Nath, and Mansoor Alam. "Hardware-oriented authentication for advanced metering infrastructure." IEEE Transactions on Smart Grid 9, no. 2 (2016): 1261-1270.

[115] Maiti Abhranil, Vikash Gunreddy, and Patrick Schaumont. "A systematic method to evaluate and compare the performance of physical unclonable functions." In Embedded systems design with FPGAs, pp. 245-267. Springer, New York, NY, 2013.