Article

Securing IoT Networks Using Machine Learning-Resistant Physical Unclonable Functions (PUFs) on Edge Devices

Abdul Manan Sheikh [1,2,*], Md. Rafiqul Islam [2], Mohamed Hadi Habaebi [2], Suriza Ahmad Zabidi [2], Athaur Rahman bin Najeeb [2] and Mazhar Baloch [1]

[1] Department of Electrical Engineering and Computer Science, A'Sharqiyah University, Ibra 400, Oman; mazhar.baloch@asu.edu.om

[2] Department of Electrical and Computer Engineering, International Islamic University Malaysia, Kuala Lumpur 50728, Malaysia; rafiq@iium.edu.my (M.R.I.); habaebi@iium.edu.my (M.H.H.); suriza@iium.edu.my (S.A.Z.); athaur@iium.edu.my (A.R.b.N.)

* Correspondence: abdul.manan@asu.edu.om

Abstract

The Internet of Things (IoT) has transformed global connectivity by linking people, smart devices, and data. However, as the number of connected devices continues to grow, ensuring secure data transmission and communication has become increasingly challenging. IoT security threats arise at the device level due to limited computing resources, mobility, and the large diversity of devices, as well as at the network level, where the use of varied protocols by different vendors introduces further vulnerabilities. Physical Unclonable Functions (PUFs) provide a lightweight, hardware-based security primitive that exploits inherent device-specific variations to ensure uniqueness, unpredictability, and enhanced protection of data and user privacy. Additionally, modeling attacks against PUF architectures is challenging due to the random and unpredictable physical variations inherent in their design, making it nearly impossible for attackers to accurately replicate their unique responses. This study collected approximately 80,000 Challenge Response Pairs (CRPs) from a Ring Oscillator (RO) PUF design to evaluate its resilience against modeling attacks. The predictive performance of five machine learning algorithms, i.e., Support Vector Machines, Logistic Regression, Artificial Neural Networks with a Multilayer Perceptron, K-Nearest Neighbors, and Gradient Boosting, was analyzed, and the results showed an average accuracy of approximately 60%, demonstrating the strong resistance of the RO PUF to these attacks. The NIST statistical test suite was applied to the CRP data of the RO PUF to evaluate its randomness quality. The *p*-values from the 15 statistical tests confirm that the CRP data exhibit true randomness, with most values exceeding the 0.01 threshold and supporting the null hypothesis of randomness.

Keywords: cryptography; IoT; PUFs; CRPs; machine learning

Academic Editors: Bin Hu, Jiacun Wang and Xiwang Guo

Received: 27 November 2025
Revised: 6 January 2026
Accepted: 8 January 2026
Published: 12 January 2026

1. Introduction

The Internet of Things (IoT) has revolutionized remote monitoring and control of systems by enabling the processing of real-time data from numerous sensing devices. Cisco estimated that approximately 500 billion IoT devices would be deployed by 2030 [1]. Intelligent interfaces allow these devices to interact, collect, communicate, and store data efficiently. A trusted ecosystem built on the principles of authentication, authorization, privacy, confidentiality, availability, and integrity is therefore essential to ensure secure data transactions [2,3]. IoT devices are often constrained by limited storage, processing, sensing, and computing resources [4]. Moreover, the diversity of devices from multiple vendors deployed within IoT networks makes them particularly vulnerable to security threats, with device authentication emerging as a critical factor in maintaining trustworthy communication. Authenticating edge devices is as important as authenticating users to ensure that only legitimate devices gain access to IoT resources. Weakly secured devices can compromise the entire system, leading to severe financial and reputational consequences. The heterogeneous nature of IoT networks, combined with the resource limitations of edge devices, makes traditional authentication schemes impractical. These constraints underscore the need for innovative, lightweight authentication mechanisms tailored specifically to IoT environments [5].

IoT devices operating at the network edge remain highly vulnerable to cyber threats, including data breaches, hardware tampering, and denial-of-service attacks. Traditional cloud-based, centrally managed infrastructures often struggle to counter advanced risks, including Distributed Denial of Service (DDoS) attacks, Man-in-the-Middle (MiTM) intrusions, eavesdropping, and attacks powered by artificial intelligence [6]. To mitigate these risks more effectively, computational capabilities can be shifted closer to the network edge, enabling real-time threat detection and mitigation directly on devices rather than relying solely on remote cloud infrastructure. Cloud-assisted IoT takes advantage of the large storage capacity and powerful computing resources available in cloud platforms to process and analyze data. Large amounts of data are continuously sent from widely distributed IoT nodes and cloud servers experience processing delays, leading to higher latency and slower response times for the services delivered to network users.

Edge Computing (EC) enables local processing and storage of sensitive data, thereby preserving privacy and facilitating faster access to critical information during security investigations [7]. In addition, EC offers advantages such as reduced latency, improved bandwidth efficiency, and enhanced privacy and security [8]. The EC architecture, shown in Figure 1, is typically organized into three layers: the edge devices layer, the edge nodes (or gateway) layer, and the cloud. Edge devices, such as IoT sensors, smartphones, cameras, or industrial machines, are primarily responsible for generating or collecting data. Due to limited processing capabilities, these devices typically perform only basic functions. The edge node layer consists of intermediate devices such as routers, edge servers, or micro data centers, which perform more advanced processing and data aggregation. This processing is further complicated by device heterogeneity and resource limitations. The cloud layer, in turn, handles complex computations, long-term data storage, and advanced analytics that exceed the capabilities of edge nodes. Connectivity among edge devices, edge nodes, and the cloud is supported through WiFi, 5G, or wired infrastructure to ensure efficient data transmission. Edge-driven IoT systems distribute heterogeneous resources across the network to achieve flexibility and scalability, supporting applications such as smart cities and autonomous vehicles. However, edge servers and devices remain constrained by limited power and processing capacity, and offloading large volumes of data and computational tasks can overload the network.

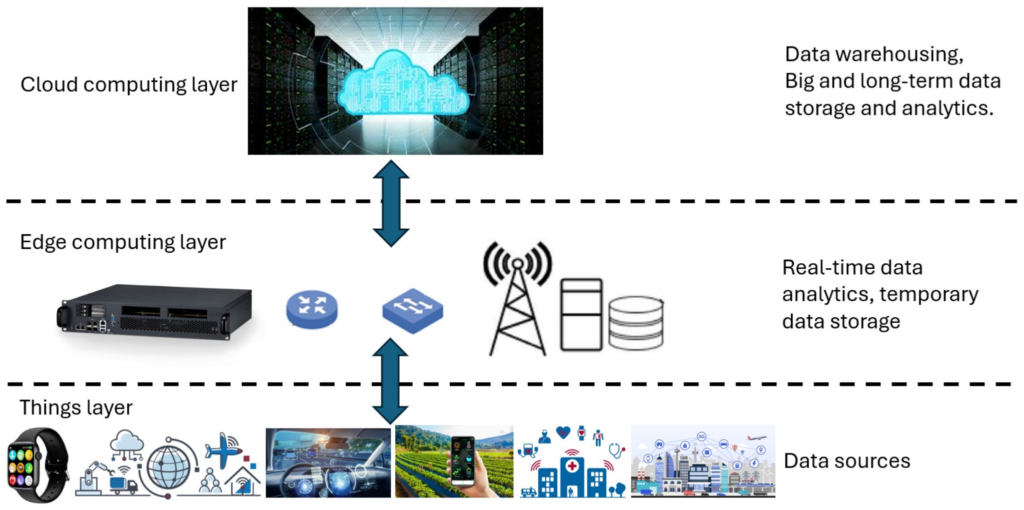


Figure 1. Edge computing architecture.

1.1. Motivations

The integration of EC with IoT introduces challenges due to inherent differences. IoT systems encompass a diverse range of hardware platforms and communication protocols, and their success depends on a unified framework that ensures seamless interoperability between IoT devices and edge nodes. Security and privacy remain primary concerns, complicated further by device heterogeneity and resource limitations such as memory capacity and battery power [9]. As a result, edge devices and servers are highly vulnerable to attacks, and a breach at any point can compromise the entire network. To address these risks, lightweight and robust security mechanisms tailored to the distributed and heterogeneous nature of IoT systems are essential. Several strategies have been proposed to mitigate security challenges in EC. Conventional measures include intrusion detection systems (IDS) to monitor malicious activity, strong access control to regulate permissions, encryption to protect data at rest and in transit, and authentication mechanisms to verify user and device identities. Beyond these methods, advanced technologies such as artificial intelligence (AI) and machine learning (ML) are increasingly used for real-time anomaly detection and adaptive defense against evolving threats [10]. Machine learning–based modeling attacks can learn and replicate the behavior of a PUF with high accuracy, thereby undermining its security. In contrast, blockchain-based approaches aim to ensure transaction integrity [8,11]. In parallel, Physical Unclonable Functions (PUFs) have garnered attention as hardware-based security primitives that provide device authentication, secure key generation, and cryptographic support. PUFs exploit manufacturing variations to generate unique device-specific identifiers that safeguard against cloning and other hardware-based attacks. Field Programmable Gate Arrays (FPGAs), known for their flexibility, reconfigurability, and rapid prototyping, are well-suited for implementing PUFs [12]. Moreover, PUFs can be integrated with intellectual property (IP) cores in FPGA-based systems, making them practical for real-world deployments [13,14].

Meanwhile, ML techniques such as Support Vector Machines, Logistic Regression, and Deep Neural Networks are increasingly used in hardware security to detect hardware Trojans, identify counterfeit integrated circuits (ICs), and evaluate the reliability of PUFs [15]. At the edge, ML models are also applied to analyze data streams and detect anomalies, intrusions, and malicious activities in real time. However, ML models deployed on resource-constrained edge devices are themselves vulnerable to adversarial threats, including poisoning attacks that corrupt training data, evasion attacks that deceive models with crafted inputs, inference attacks that extract sensitive information, and exploratory attacks that exploit model weaknesses or replicate behavior. ML-based modeling attacks use ML methods to learn and reproduce the behavior of a PUF with very high accuracy, which can undermine its security. An adversary can construct a numerical model of the PUF by obtaining a portion of its challenge response pairs through eavesdropping or any other form of unauthorized access [16]. Adversaries typically exploit ML systems by targeting different stages of the learning pipeline. Training data can be poisoned to corrupt the learned model or to implant backdoors, while testing data may be manipulated to expose vulnerabilities and influence model predictions. In addition, privacy-oriented attacks aim to infer sensitive information from the trained model or its gradients [17]. Several studies have reported successful ML-based attacks on various types of physically unclonable functions, including RO-PUFs, using logistic regression and neural network-based models [18]. Among delay-based PUF architectures, RO-PUFs generally offer higher reliability than SRAM PUFs because their responses are less sensitive to environmental variations such as temperature and supply voltage. Although SRAM PUFs demonstrate strong resistance to modeling attacks, they are more susceptible to cloning and invasive attacks. To further enhance resistance against modeling, XOR PUFs introduce nonlinearity by combining the outputs of multiple identical PUF instances through an XOR operation, which makes them harder to model than single arbiter or ring oscillator PUFs. Nevertheless, when an optimal set of challenge–response pairs is available, XOR PUFs can still be effectively compromised using ML-based attacks [19]. These risks are exacerbated by the limited storage and computational capabilities of edge devices. To mitigate such vulnerabilities, integrating FPGA-based PUFs with ML frameworks offers a promising approach to strengthening edge security [20]. PUF-derived keys can be employed to encrypt ML models or authenticate devices contributing training data, protecting against tampering and data leakage. Leveraging FPGAs for PUF design provides flexibility, rapid prototyping, and reconfigurability, while seamless integration with other IP cores enables robust and scalable security solutions. The placement and routing of ring oscillators (ROs) on an FPGA play a critical role in determining the quality of PUF responses, as the physical locations of logic slices directly influence the inherent delay variations. Consequently, the oscillation frequencies of ROs are strongly dependent on their spatial distribution across the FPGA die. To counter machine learning–based modeling attacks, several mitigation strategies have been proposed, including the introduction of non-linearity into PUF architectures, XOR combination of multiple individual PUF responses, random selection of response substrings, and challenge randomization. While these approaches enhance resistance against modeling attacks, they often increase the architectural complexity of the PUF or complicate protocol-level protections, thereby impacting implementation efficiency and scalability [21].

1.2. Research Contributions

This research presents a comprehensive investigation into the design, implementation, and security evaluation of a RO-PUF on an FPGA, with an emphasis on ML-based modeling, resistance analysis, and statistical randomness analysis. The main contributions of this work are summarized as follows:

- FPGA-Based Design and Implementation of RO PUF: The study successfully implemented a configurable RO-PUF architecture on an FPGA platform. A dedicated testbench was developed to acquire large-scale challenge–response datasets under controlled operating conditions, verifying the reproducibility and uniqueness of the PUF behavior across multiple FPGA instances.
- Comprehensive Evaluation of PUF Metrics: The implemented RO PUF was analyzed using standard performance metrics such as uniformity, uniqueness, and reliability

(intra-Hamming distance). The proposed design demonstrated balanced uniformity near the ideal 50%.

- Machine Learning-Based Attack and Accuracy Estimation: To assess the resilience of the proposed RO-PUF against modeling attacks, various ML algorithms, including Logistic Regression (LR), Support Vector Machine (SVM), Multilayer Perceptron (MLP), and k-Nearest Neighbor (k-NN), were employed. Confusion matrix analysis revealed that linear models, such as LR, failed to capture the nonlinear challenge–response relationship, while nonlinear models (SVM, MLP, and k-NN) performed moderately better but exhibited trade-offs between precision and recall. The findings confirmed that none of the models achieved strong predictive accuracy, highlighting the robustness and unpredictability of the proposed RO-PUF against ML-based cloning attempts.
- Randomness Validation Using NIST SP 800-22 test suite: The randomness quality of the generated CRP responses was validated using the NIST statistical test suite, covering tests such as frequency, runs, block frequency, and cumulative sums. The majority of the tests yielded *p*-values greater than 0.01, confirming that the PUF outputs exhibit strong statistical randomness and are suitable for cryptographic and authentication applications.

2. Related Works

Shen et al. proposed an integrated security framework for EC that combines ML and cryptographic techniques to monitor and detect abnormal activities on the network. The study provides valuable insights into the use of Support Vector Machine (SVM), k-Nearest Neighbor (k-NN), and Long Short-Term Memory (LSTM) models for time series prediction, evaluated using performance metrics such as precision, recall, and F1-score [22]. In the Collaborative Edge Computing (CEC) model, the edge layer handles data storage and processing in a distributed manner. Given the limited processing capacity of individual edge devices, these devices cooperate to offload tasks among themselves, a process known as load balancing. The researchers in [23] employed PUFs to authenticate edge devices during load balancing, eliminating the need to store a database of CRPs locally. Cheng et al. integrated blockchain technology with certificateless cryptography, elliptic curve cryptography, and pseudonym-based cryptography to enable mutual authentication between edge servers and IoT devices [24]. The study in [25] proposed a privacy-preserving edge computing approach that utilizes federated learning to train a unified deep learning model across multiple end users collaboratively. Instead of sharing raw data, only model parameters (i.e., gradients) are exchanged between participants. The parameters from local deep learning models on various edge nodes are aggregated at the edge and then distributed to all participants. Through several iterations of local training and parameter aggregation, a deep learning model is developed that preserves user privacy without the need to share raw data. Zhang and colleagues introduced a configurable tristate PUF that can operate as an arbiter PUF, a ring oscillator PUF, or a bistable ring PUF. The design uses a bitwise XOR-based obfuscation mechanism to hide the relationship between challenges and responses. As a result, machine learning models fail to build accurate prediction model, with all attack accuracies remaining between 50% and 60%, which is effectively the same as random guessing [26]. The authors of [27] introduced an authentication method that employs arbiter PUFs together with three protection strategies called challenge splitting, challenge scrambling, and challenge padding. Each strategy disrupts the structure or visibility of the challenge so that machine learning models cannot form an accurate numerical model of the PUF. In ref. [28], a hybrid secure deduplication scheme is introduced that ensures data privacy on the server side while enhancing network performance on the client side. Additionally, an additive homomorphic encryption method is proposed to enable efficient deduplication operations on resource-constrained edge nodes. Intrusion detection systems (IDS) are vital for the security of IoT systems, as they detect traces of known attacks and unusual behavior in IoT devices and their connecting networks [29]. However, the majority of IDS algorithms use conventional encryption and cryptographic techniques. Unfortunately, these approaches are vulnerable to physical attacks, as they primarily rely on storing secret keys in the device's local memory. Blockchain is considered an ideal approach to store the huge amount of data generated by IoT devices with utmost privacy and security using a distributed ledger. However, the resource-constrained IoT devices cannot meet the computational requirements of data mining in blockchain networks. Sarkar et al. have proposed a robust PUF-based authentication system to replace the popular consensus algorithms [30].

A PUF design based on Generalized Galois Ring Oscillators (GenGARO) was implemented on Artix-7 FPGAs, Advanced Micro Devices (AMD), Santa Clara, CA 95054, USA using both 11-LUT and 3-LUT configurations. The proposed design demonstrates strong resilience against various ML models, including Decision Trees (DT), Random Forests (RF), k-Nearest Neighbors (k-NN), Support Vector Machines (SVM), and Multilayer Perceptrons (MLP) [19]. Another approach, the Structure-Obfuscated PUF (SO-PUF), is built on a configurable ring oscillator (CRO) framework and dynamically removes NOT gates at each stage based on the challenge input. As reported in [31], SO-PUF was implemented on an Xilinx Spartan-6 FPGA, AMD, Santa Clara, CA 95054, USA and achieved near-random prediction accuracy. Kareem et al. [32] investigated the vulnerability of three RO-PUF designs to ML attacks and evaluated their prediction accuracy using models such as DT, RF, k-NN, and SVM. Abulibdeh et al. introduced the Algorithmically Optimized Configurable Ring Oscillator PUF (AOCRO), which demonstrated strong resistance to ML modeling attacks. In their evaluation, five different ML algorithms (SVM, MLP, LR, CNN, and CMA ES) were trained on a dataset of one million CRPs, achieving an average prediction accuracy of 61.3%, indicating enhanced robustness compared to conventional CRO PUF designs [33]. Jack Miskelly et al. investigated the impact of ML attacks on Configurable RO PUFs. The study simulated 128-bit CRO PUFs and multi-PUF variants with datasets ranging from 1000 to 10,000 CRPs. The results showed that conventional CRO PUFs could be accurately modeled using an LR, ML model, achieving prediction accuracies exceeding 98–99% [34]. Laguduva et al. proposed a non-invasive, architecture-independent attack on PUFs using CRPs. This method achieved a cloning accuracy of 93.5% without requiring any prior knowledge of the PUF's internal architecture [35]. A summary of the results obtained by the above-mentioned researchers is shown in Table 1.

Table 1. ML model accuracy (%) against RO PUFs.

Article	KNN	SVM	MLP	RF	DT	LR
[19]	56.76–63.24	49.76–69.71	50.10–70.86	62.67–75.81	57.90–72.00	
[31]		49.75	63.13			49.63
[32]	74.6	66.2		72.3	64.6	
[36]		56.64	58.04			51.9
[37]		58.59–61.95				

3. Physical Unclonable Functions (PUFs)

Researchers have identified PUFs as robust security primitives that can guarantee the three pillars of security, namely confidentiality, authenticity, and privacy, of IoT data. PUFs extract unique information from the physical characteristics of the IoT devices [38]. PUF-based authentication protocols enhance the security of resource-constrained edge devices without requiring them to store credentials in their limited non-volatile memory.

They utilize the inherent random variations introduced during manufacturing to generate secret keys dynamically. PUFs provide essential security functions such as authentication and secret key generation, especially in resource-constrained environments like the IoT [39]. The inherent unclonability of PUFs stems from numerous uncontrollable random parameters introduced during fabrication. When a PUF receives an input, known as a challenge (C), it produces a corresponding output response (R). Figure 2 illustrates that physical variations in the fabrication of integrated circuits (ICs) can result in two ICs yielding different responses to the same challenge. This relationship between the input and output is referred to as a CRP [40]. Traditional authentication methods rely on storing secret credentials in a device's memory, which makes them unsuitable for physically unprotected IoT devices. Attackers can exploit physical vulnerabilities to compromise the entire system. PUFs mitigate such risks in two key ways: first, they generate volatile secrets that are not stored in digital memory but are intrinsically embedded in the hardware structure; second, the uniqueness of each PUF enables it to serve as a unique identifier for individual IoT devices [41].

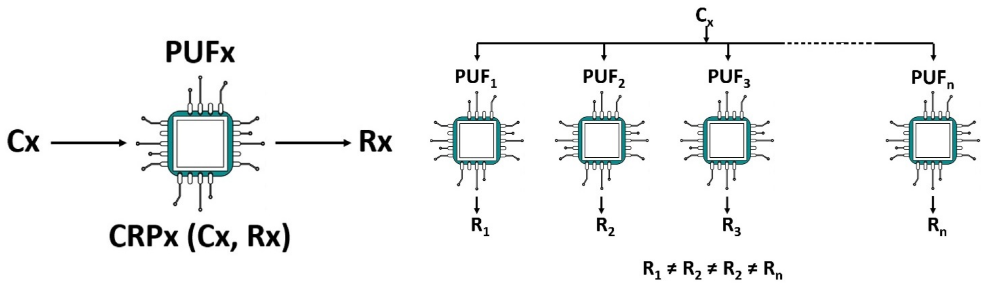


Figure 2. PUFs Challenge & Response Pair (CRP) generation [42].

3.1. PUFs Classification

PUFs can be placed into two groups based on the number of CRPs they can generate: strong and weak. The limited number of CRPs in a weak PUF is typically proportional to the number of components used in its construction. In contrast, strong PUFs provide a vast number of CRPs, making polynomial-time modeling attacks computationally impractical [43]. Strong PUFs are typically employed in authentication and key establishment protocols, whereas weak PUFs are mainly used for identification and secure key storage applications [44]. Examples of strong PUFs include the Arbiter PUF, the XOR Arbiter PUF, and the Lightweight PUF. In contrast, weak PUFs are represented by designs such as the SRAM PUF, Ring Oscillator (RO) PUF, Anderson PUF, Memristor PUF, Thyristor PUF, and One-Time Programmable (OTP) PUF [45]. Additionally, PUFs can be classified as silicon-based and non-silicon PUFs, depending on the manufacturing process adopted. Silicon PUFs rely on fabrication mismatches inherent in integrated circuits and can be further divided into delay-based PUFs and memory-based PUFs. In contrast, non-silicon PUFs are based on physical irregularities in systems composed of non-electronic components. Memory-based PUFs utilize the initial binary states of memory upon power-up, whereas delay-based PUFs exploit variations in signal propagation delays within circuits.

3.2. RO PUFs

Due to inherent random variations in the manufacturing process, two similar ROs do not generate identical oscillation frequencies. The frequency differences between selected RO pairs form the output response of the PUF. An RO PUF consists of an odd number of NOT logic gates arranged in a ring, causing the output to oscillate between logic '1' and '0' at a specific frequency [46]. The basic architecture of an RO PUF, as shown in Figure 3, includes an odd number of inverter gates and an AND gate to enable or disable the feedback loop. The conventional RO-PUF consists of several essential components: n ring oscillators (ROs), two n-to-1 multiplexers (MUXs), two counters, and a comparator circuit. The outputs of each RO are routed to the inputs of both MUXs, whose selected outputs serve as clock signals for the counters. Each counter increments according to the oscillation frequency of the specific RO selected by its respective MUX. Finally, the comparator compares the values stored in the two counters to produce the RO-PUF response corresponding to the applied challenge inputs. The frequency of each RO depends on the delay of the inverters, which is influenced by variations in the manufacturing process.

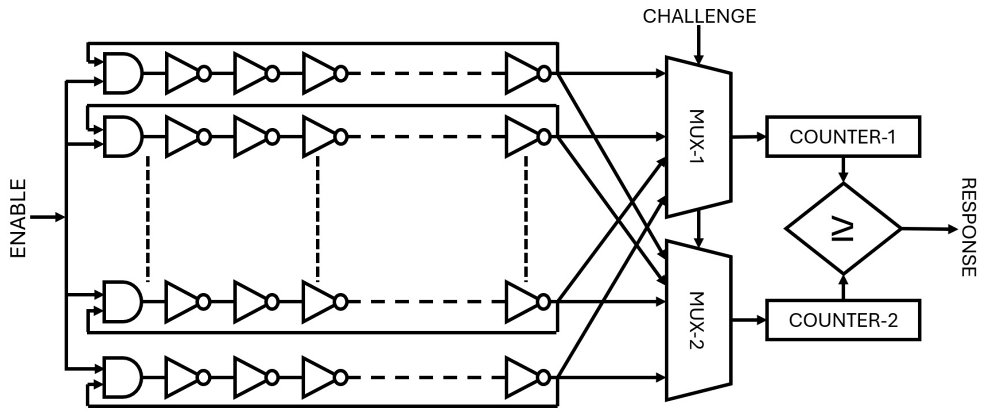


Figure 3. Ring Oscillator (RO) PUF.

The oscillation frequency of the RO PUF, as shown in Equation (1) is inversely proportional to both the odd number of gates (n) and their average propagation delay (t_{pd}). Consequently, f_{osc} is sensitive to inherent variations in gate delay, which causes each instance of the RO structure to exhibit a slightly different oscillation frequency [47].

$$f_{\text{osc}} = \frac{1}{2nt_{\text{pd}}} \tag{1}$$

The delay of each not gate is modelled by Equation (2), where μ_{pd} is the nominal delay and δ_k is a zero-mean random deviation, often approximated by gaussian, $\mathcal{N}(0, \sigma_{pd}^2)$.

$$\text{t}_{\text{pd},k} = \mu_{\text{pd}} + \delta_k \tag{2}$$

The fundamental operation of the proposed RO PUF is outlined in Algorithm 1. Initially, multiple ROs are instantiated, each consisting of an odd number of inverters connected in a closed loop chain, causing continuous oscillation due to intrinsic gate delays. For every response bit, a corresponding challenge input selects a pair of ROs through multiplexers. The oscillation frequencies of two selected ROs are recorded over a fixed time interval using counters. If the first RO exhibits a higher count, indicating a higher oscillating frequency, a response bit of ‘1’ is generated; otherwise, a ‘0’ is assigned.

Algorithm 1 Ring Oscillator PUF (RO-PUF) Algorithm

```
Require: Challenge input $C = \{Sel\_A, Sel\_B\}$, Time window $T$
Ensure: Response bit $R$
 1: Initialize $N$ Ring Oscillators: $RO[0], RO[1], \dots, RO[N-1]$
 2: Initialize two $N$-to-1 MUXs, two Counters: $Counter\_A$, $Counter\_B$, and a Comparator
 3: Select $RO\_A = RO[Sel\_A]$ via MUX A
 4: Select $RO\_B = RO[Sel\_B]$ via MUX B
 5: Connect $RO\_A$ output to $Counter\_A$ clock
 6: Connect $RO\_B$ output to $Counter\_B$ clock
 7: Enable $Counter\_A$ and $Counter\_B$ for fixed time window $T$
 8: During $T$, increment counters on each RO oscillation pulse
 9: After $T$, disable both counters
10: if $Counter\_A > Counter\_B$ then
11:     $R \leftarrow 1$
12: else
13:     $R \leftarrow 0$
14: end if
15: return $R$
```

3.3. PUF Performance Metrices

PUF performance metrics serve as key indicators of functional behavior and security robustness. Hamming distance (HD) serves as a critical metric for measuring the degree of dissimilarity between two responses generated by a PUF and is further distinguished into intra-PUF and inter-PUF comparisons. The intra-PUF HD indicates the dissimilarity between the responses of a single PUF, highlighting the internal consistency or variability of the PUF. In contrast, the inter-PUF Hamming distance compares responses from two different PUFs, offering a gauge of the uniqueness and distinguishability of each PUF's responses. According to the classification presented by Pahlevi et al., the evaluation framework can be grouped into three main categories [48]. The first category, conventional PUF evaluations, includes metrics such as uniqueness, uniformity, and reliability. These metrics assess how distinguishable the responses are between different devices, how evenly distributed the output bits appear, and how consistently the PUF can reproduce the same response under varying environmental or operational conditions. The second category focuses on authentication-oriented metrics, which evaluate the practical suitability of PUFs for real-world security applications. Metrics such as the False Acceptance Rate (FAR) and False Rejection Rate (FRR) measure authentication accuracy, while bit aliasing and the bit error rate (BER) capture bit-level stability and possible device bias. Entropy estimation is also included in this category to quantify the randomness present in the response set. The third category comprises ML-based attack evaluations, which determine how well the PUF can withstand modern modeling attacks aimed at predicting its behavior.

The performance of access control mechanisms is evaluated using four fundamental metrics derived from the confusion matrix, which are defined in Equations (3)–(6): the false acceptance rate (FAR), the false rejection rate (FRR), the true acceptance rate (TAR), and the true rejection rate (TRR). The confusion matrix serves as a security evaluation tool for assessing the effectiveness and dependability of a PUF-based authentication system. FAR indicates instances where unauthorized individuals are mistakenly granted access, while FRR reflects cases where legitimate users are unfairly denied access, highlighting critical error dimensions that must be minimized [49].

$$\mathrm{FRR} = \frac{\text{Number of False Rejections}}{\text{Total Number of Genuine Attempts}} \times 100\% \tag{3}$$

$$\text{FAR} = \frac{\text{Number of False Acceptances}}{\text{Total Number of Impostor Attempts}} \times 100\% \tag{4}$$

$$\text{TRR} = \frac{\text{Number of True Rejections}}{\text{Total Number of Impostor Attempts}} \times 100\% \tag{5}$$

$$\text{TAR} = \frac{\text{Number of True Acceptances}}{\text{Total Number of Genuine Attempts}} \times 100\% \tag{6}$$

The use of a confusion matrix to evaluate an authentication system highlights its importance in assessing how effectively the system differentiates between legitimate access attempts and potential security breaches.

- Uniqueness: It is used to quantify how different devices respond to the same input challenge. In other words, it is defined as the inter-device Hamming Distance (HD) between different devices, and its ideal value is 50%. The HD of Equation (7) estimates the uniqueness of CRPs, where n is the total number of devices, R_i and R_j are the respective responses of the ith and jth devices under the same challenge, $HD(\cdot,\cdot)$ is the Hamming Distance operator, and m is the bit length of each response.

$$\text{Uniqueness} = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{\text{HD}(R_i, R_j)}{m} \tag{7}$$

- Uniformity: It is the probability that the 0 s and 1 s are uniformly distributed in PUF's response. Uniformity measures the balance between zeros and ones in the responses generated by a PUF. It is obtained by computing the average Hamming weight of the responses, as expressed in Equation (8).

$$\text{Uniformity} = \frac{1}{k} \sum_{l=1}^{k} R_l \tag{8}$$

- Reliability: It indicates the ability of a PUF to reproduce the same response bit for a given challenge input even when environmental conditions, such as supply voltage and temperature, vary. An ideal reliability close to 100% means that no bit flips occur across repeated measurements. However, achieving perfect reliability is difficult because PUF outputs are inherently sensitive to these variations.

$$\text{HD}_{\text{intra}}(R, R') = \sum_{b=1}^{m} \left| R_b - R'_b \right| \tag{9}$$

  A standard metric for reliability is the intra-class Hamming Distance, which is illustrated in Equation (9), where R and R' are two responses from the same device under the same challenge.
- Bit aliasing: It complements uniqueness and uniformity by verifying whether a given bit exhibits enough variation. Ideally, each bit appears randomly as 0 or 1 with a typical value of 50%, signifying minimal bias. The aliasing factor for the bth bit is represented in Equation (10), where $R_{i,b}$ is the bth bit of the ith device's response.

$$\text{Aliasing}(b) = \frac{1}{n} \sum_{i=1}^{n} R_{i,b} \tag{10}$$

- Bit Error Rate (BER): The BER defined in Equation (11) gives an estimate of how often a PUF produces incorrect or flipped bits when the same challenge input is applied multiple times under varying environmental conditions, such as temperature and supply voltage.

$$\text{BER} = \frac{\text{Number of flipped bits}}{\text{Total bits measured}} \tag{11}$$

- Entropy: It is used to evaluate the overall randomness of PUF outputs, particularly against advanced modeling attacks and side channel attacks. A higher entropy value reflects a larger and more unpredictable response space. The most widely used metric for this purpose is Shannon entropy, which is defined in Equation (12) as follows:

$$\mathrm{H}(R) = -\sum_{\omega} p(\omega) \log_2 p(\omega) \tag{12}$$

where ω represents every possible output pattern, and $p(\omega)$ denotes the probability associated with each pattern [40].

3.4. ML Modelling Attacks

Resistance and reliability against ML attacks are two major concerns for the overall viability of PUF-based authentication protocols [50]. ML-based attacks on PUFs can be classified as semi-invasive attacks, as adversaries exploit the PUF by intercepting the communication channel between the PUF client and the server. The intercepted data are then preprocessed, and a parametric numerical model is constructed using ML algorithms that can successfully predict PUF responses [51,52]. Adversaries carry out PUF modeling attacks using deep learning, LR, support vector machine, and evolution strategy, assuming access to CRPs [53]. Countermeasures against ML attacks on PUFs typically involve increasing the nonlinearity and complexity of the PUF model or integrating additional modules that enable more complex operating modes and communication protocols with the verifier [54].

Machine Learning

An ML model is developed by analyzing available training data on a computing platform. The presence or absence of labels, which represent target values, plays a critical role in model development. Based on the use of labels, ML methods are broadly classified into supervised and unsupervised learning approaches. Supervised learning relies on labeled training data, whereas unsupervised learning extracts patterns without requiring labels. In contrast, reinforcement learning employs automated feedback mechanisms to iteratively improve model behavior [55]. Each learning paradigm has distinct advantages and limitations, and no single technique can be considered universally optimal without considering application-specific constraints [56]. Like many transformative technologies, ML presents both significant opportunities and challenges. Its widespread adoption is expected to reshape the cyber threat landscape in the coming years. When integrated into malicious workflows, ML techniques can enable attacks that are more adaptive, more resilient, and less detectable by existing defense mechanisms, facilitating the emergence of novel threat vectors. Consequently, the application of ML in security systems introduces new attack surfaces that adversaries can exploit. In particular, several supervised learning algorithms are being investigated to optimize attack strategies and to automate tasks that traditionally require manual statistical analysis [57]. Identifying an appropriate ML model is a challenging task due to the availability of a wide spectrum of classification and regression strategies. Thus, picking a particular model involves evaluating key tradeoffs such as accuracy, computational efficiency, interpretability, and model complexity and often requires iterative experimentation. In this work, four supervised ML models are chosen for modeling DDoS attacks in IoT networks. These algorithms are examined to evaluate their effectiveness in enabling more adaptive and intelligent attack strategies. A brief discussion of these models is presented below.

- Support Vector Machine (SVM): It identifies a hyperplane that best separates data points belonging to two different classes. In a two-dimensional feature space, this hyperplane is represented as a line, while in an n-dimensional space, where n denotes the number of features, it becomes a plane or higher dimensional decision boundary. Among the many possible separating hyperplanes, the SVM algorithm selects the optimal one by maximizing the margin between the two classes. Training an SVM directly on raw data often results in suboptimal performance due to issues such as missing values, outliers, and redundant information. Therefore, data preprocessing is essential and typically includes data cleaning to address missing values and outliers, followed by feature extraction or selection to retain the most informative attributes. When linear separation of classes is not feasible, kernel functions are employed to project the data into a higher dimensional space, where a clear separation becomes achievable [58,59].
$$\mathrm{y} = f(x) = \mathrm{W}^{\mathrm{T}}x + b = \sum_{i=1}^{N} w_i x_i + b \tag{13}$$
SVM separates data across a hyperplane $f(x) = 0$, by solving a constrained quadratic optimization problem. The input data x_i $(i = 1, 2, \ldots, N)$ consists of objects with different labels corresponding to the two classes, namely the positive and negative classes. Equation (13) represents a hyperplane that separates the given data. where N is the number of samples, W is an N-dimensional vector, and b is a scalar. The vector W and scalar b are used to define the position of the separating hyperplane.
- Multi-Layer Perceptron (MLP): is used as an Artificial Neural Network (ANN) classification algorithm. It belongs to a class of nonlinear statistical models comprising of multiple layers of nodes arranged in a directed graph, where each layer is fully connected to the subsequent layer. Typically, MLP consists of three types of layers, namely the input layer, one or more hidden layers, and the output layer. The output y of a neural model against input data x_i $(i = 1, 2, \ldots, N)$ is shown in Equation (14).
$$\mathrm{y} = f(\mathrm{W}^{\mathrm{T}}x) = f\left(\sum_{i=1}^{N} w_i x_i + b\right) \tag{14}$$
where f is the activation function, N is the number of neurons, W are the ANN model weights, and b is the bias vector.
- Logistic Regression (LR): Its a statistical modeling technique, commonly associated with the sigmoid function, that is used to predict a binary outcome based on one or more predictor variables. Binary logistic regression is applied when the dependent variable has two possible outcomes, while multinomial logistic regression is employed when the outcome variable consists of more than two categories [60]. The logistic regression model is expressed as logistic (sigmoid) function, in Equation (15).
$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{15}$$
where z is a linear summation of input features and their coefficients. As illustrated in Equation (16), the model predicts the probability p that the dependent variable y equals 1 given input x for binary classification.
$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{16}$$

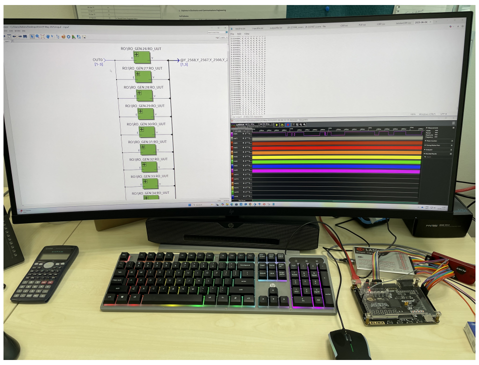
where β_0 is the intercept and β_1 is the coefficient for the input feature. The odds of the event are expressed as $\frac{p}{1-p}$, in Equation (17), representing a linear relationship between the features and the log-odds [61].

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{17}$$

- k-Nearest Neighbors (k-NN): It is a non-parametric ML algorithm applied to both classification and regression tasks. It assigns a class label or predicts a value for a query sample based on the majority class or the average value of its nearest neighbors in the feature space. The algorithm is simple to implement and does not require an explicit training phase. However, its performance deteriorates with high-dimensional data, and it incurs high inference time and memory overhead due to the need to store and compare all training samples. The k-NN algorithm operates in three main stages. First, it computes the distance between the query point and all training samples. Second, it selects the k closest neighbors based on the chosen distance metric. Finally, it predicts the class label or value of the query point using majority voting for classification or mean aggregation for regression.

4. Experimental Setup and Implementation

The experimental setup and synthesized netlist statistics of the proposed RO-PUF are shown in Figure 4. The proposed architecture is implemented and verified using Cyclone IVE FPGA chips (EP4CE10F17C8), Intel, Santa Clara, CA 95054, USA in the AX4010 development board. Intel Quartus Prime 18.1 Lite Edition, Santa Clara, CA 95054, USA is used to develop and verify the RO-PUF design prior to configuring the FPGA device. The Cyclone IV E EP4CE10F17C8 is a low-cost, low-power FPGA developed by Intel. It features 10,320 logic elements, 414 Kbits of embedded memory, 179 user I/O pins, and four PLLs for flexible clock management, all packaged in a 256-pin FBGA. The RO-PUF design, implemented in VHDL, is interfaced with the onboard GPIO pins of the AX4010 development board. All control signals, including challenge inputs and response outputs, are connected to a 16-channel logic analyzer through these GPIO pins. The design incorporates 512 ring oscillators (ROs) comprising five inverters. The outputs of the ROs are routed to two sets of 256-to-1 multiplexers (MUXes). Each MUX selects one RO output based on the challenge input, thereby determining which RO signal is passed to the output. The final response is generated by comparing the frequencies of two counters, Counter_A and Counter_B. An 8-bit linear feedback shift register (LFSR) is used to generate the challenge signals, based on the characteristic polynomial $P(x) = x^8 + x^4 + 1$.



(**a**) Experimental setup

Parameters	#
Total logic elements	1071 / 10,320 (10 %)
Total registers	40
Total pins	12 / 180 (7 %)
Max LUT depth	8.00
f_{max}	94.23 MHz
Worst case slack	9.388 ns
Nominal core voltage	1.2 V
Low junction temperature	0° C
High junction temperature	85° C

(**b**) Post synthesis netlist statistics

Figure 4. FPGA implementation of proposed RO-PUF.

The ML hyperparameters listed in the Table 2 govern both the structural configuration and the learning behavior of a model. Their selection has a direct impact on convergence speed, generalization capability, and computational efficiency. Careful tuning of hyperparameters helps achieve an appropriate balance between underfitting and overfitting, increases robustness to noise, and improves predictive performance on previously unseen data. The ML models, such as SVM, LR, MLP, and k-NN, require careful parameter tuning to achieve optimal performance. For example, as illustrated in Table 3, SVM depends on the choice of kernel and gamma values, LR relies on regularization strength and iteration limits, MLP depends on hidden layer sizes, learning rates, and solver selection, while k-NN requires the selection of an appropriate number of neighbors and distance metrics. The training process involves preparing the PUF CRP dataset by separating the features from the responses and then splitting the data into training and testing sets for supervised learning. Validation and evaluation are carried out using metrics such as accuracy and confusion matrices to assess model performance. While train-test splits provide an initial measure of effectiveness, applying cross-validation techniques can further enhance reliability by reducing variance and ensuring the models generalize well to unseen CRPs.

Table 2. Hyperparameters of Machine Learning Models

Model	Hyperparameters
Support Vector Machine (SVM)	Kernel function, regularization parameter C, kernel coefficient gamma, polynomial degree, class weight, tolerance, shrinking heuristic
Multilayer Perceptron (MLP)	Number of hidden layers, number of neurons per layer, activation function, optimizer, learning rate, learning rate schedule, batch size, maximum epochs, L2 regularization alpha, momentum, early stopping
Logistic Regression (LR)	Regularization type, regularization strength C, solver, maximum iterations, class weight, tolerance, multiclass strategy
K Nearest Neighbors (KNN)	Number of neighbors k, distance metric, distance power p, weight function, search algorithm, leaf size

Table 3. Comparison of Machine Learning Models.

Criterion	SVM	MLP	LR	KNN
Learning type	Supervised	Supervised	Supervised	Supervised
Model nature	Margin based	Neural network	Linear probabilistic	Instance based
Nonlinear capability	kernels	Yes	No	Yes
Interpretability	Medium	Low	High	Medium
Training complexity	Medium to high	High	Low	Very low
Inference complexity	Low	Low	Low	High
Scalability	Medium	High	High	Low
Overfitting risk	Medium	High	Low	Medium
Performance on small datasets	High	Low	High	High
High dimensional data handling	Excellent	Good	Good	Poor

4.1. NIST Randomness Test

Randomness plays a fundamental role in cryptography, as the strength of many security mechanisms relies on the unpredictability of generated sequences. However, generating truly random numbers is inherently challenging, and equally important is the rigorous evaluation of the quality of the generated data. To assess randomness, statistical tests are commonly employed, which yield a p-value. The p-value quantifies the probability that a truly random number generator would produce a sequence exhibiting less apparent randomness than the sequence under evaluation. In other words, it provides a statistical measure of how well the tested sequence aligns with the characteristics of an ideal random source.

Most empirical randomness tests, such as those included in the National Institute of Standards and Technology (NIST) Statistical Test Suite (STS), are built on the principles of statistical hypothesis testing. The NIST STS consists of 15 well-defined tests that evaluate binary sequences for signs of non-randomness. These tests analyze both local and global properties of the data. At the local level, they assess features such as the balance between zeros and ones, or the frequency of specific bit patterns within smaller segments of the sequence. At the global level, they examine broader statistical behavior across the entire bitstream to determine overall randomness. To further refine detection, the bitstream is often partitioned into multiple large segments, where each segment is analyzed independently. The results from these segments are then aggregated into final test statistics, which help to identify localized irregularities or systematic deviations from randomness. This multi-layered evaluation ensures that both subtle and significant weaknesses in the sequence can be detected, providing a comprehensive measure of its suitability for cryptographic applications.

In the context of the NIST STS, interpreting p-values is crucial. For each test, a significance level (commonly set at 0.01) is defined. If the p-value obtained from a sequence is greater than or equal to 0.01, the sequence is considered to have passed that particular randomness test, indicating no strong evidence of non-random behavior. Conversely, a p-value below 0.01 suggests that the sequence may deviate significantly from randomness. Ideally, when a large number of independent sequences are tested, the distribution of p-values across all tests should be uniform within the interval [0,1]. This uniformity demonstrates that the data behaves consistently with what is expected from a true random source. Therefore, both the proportion of sequences passing each test and the uniformity of their p-value distribution are essential criteria in validating the randomness of cryptographic data.

Each NIST STS test is defined by a specific test statistic, which falls into one of three categories:

- Bits: Analyzes characteristics such as proportion of bits, frequency of bit changes, and cumulative sums.
- m-bit blocks: Analyzes distribution of m-bit blocks ($m < 30$) within the sequence or its parts.
- M-bit parts: Analyzes complex properties of M-bit parts ($M > 1000$), such as matrix rank, sequence spectrum, or linear complexity.

Most tests are parameterized by n (sequence length) and may include a second parameter m or M, depending on the test. Table 4 summarizes the number of sub-tests included in the NIST STS suite. Notably, the non-overlapping template matching test has a variable number of sub-tests determined by m [62].

Table 4. Recommended bitstream parameters.

#	Name of the Test	n	M or m	Sub-Test #
1	Frequency	$n \geq 100$	–	1
2	Frequency within a block	$n \geq 100$	$20 \leq M \leq n/100$	1
3	Runs	$n \geq 100$	–	1
4	Longest run of ones	$n \geq 128$	–	1
5	Rank	$n > 38,912$	–	1
6	Spectral	$n \geq 1000$	–	1
7	Non-overlapping Template Matching	$n \geq 8m - 8$	$2 \leq m \leq 21$	148
8	Overlapping Template Matching	$n \geq 10^6$	–	1
9	Maurer's Universal	$n > 387,840$	–	1
10	Linear Complexity	$n \geq 10^6$	$500 \leq M \leq 5000$	1
11	Serial	–	$2 < m < [\log_2 n] - 2$	2
12	Approximate Entropy	–	$m < [\log_2 n] - 5$	1
13	Cumulative Sums	$n \geq 100$	–	2
14	Random Excursions	$n \geq 10^6$	–	8
15	Random Excursions variant	$n \geq 10^6$	–	18

4.1.1. Brief Description of NIST Randomness Tests

- Frequency (Monobit) test: Checks whether the number of ones and zeros is approximately equal.
- Frequency within a block test: Evaluates proportion of zeros and ones in M-bit blocks; expected frequency of ones is $M/2$.
- Runs test: Measures consecutive runs of zeros and ones; checks if transitions occur at expected frequencies.
- Longest run of ones in a block test: Examines if the longest run of ones (and zeros) in M-bit blocks matches the expected distribution.
- Random binary matrix rank test: Evaluates the rank of sub-matrices to detect linear dependencies in the sequence.
- Discrete Fourier Transform (Spectral) test: Detects periodic features using DFT peak heights.
- Non-overlapping template matching test: Detects excessive occurrences of aperiodic m-bit patterns using a sliding window that resets after each match.
- Overlapping template matching test: Counts occurrences of target substrings; window slides by one bit to allow overlaps.
- Maurer's Universal Statistical test: Measures compressibility of the sequence; overly compressible sequences indicate non-randomness.
- Linear complexity test: Estimates the length of the feedback register required to reproduce the sequence; shorter lengths indicate predictability.
- Serial test: Examines frequency of all overlapping m-bit patterns.
- Approximate Entropy test: Compares frequencies of overlapping m-bit and $(m+1)$-bit patterns to detect regularity.
- Cumulative Sum (Cusum) test: Evaluates maximal deviation from zero in the cumulative sum of bits mapped to $\{-1, +1\}$.
- Random Excursions test: Measures the number of cycles with exactly K visits in cumulative sum random walks.
- Random Excursions Variant test: Analyzes frequency of visits to specific states in cumulative sum random walks to detect non-random patterns.

4.1.2. Calibration

Calibration plays a critical role in ensuring the accuracy, reliability, and reproducibility of CRP acquisition when implementing PUFs on FPGAs. Since PUF responses are highly sensitive to environmental conditions such as temperature, voltage fluctuations, and aging, as well as to measurement artifacts including timing misalignment, noise, and signal distortions, calibration methods are employed to minimize errors before the data are fed into ML-based security models.

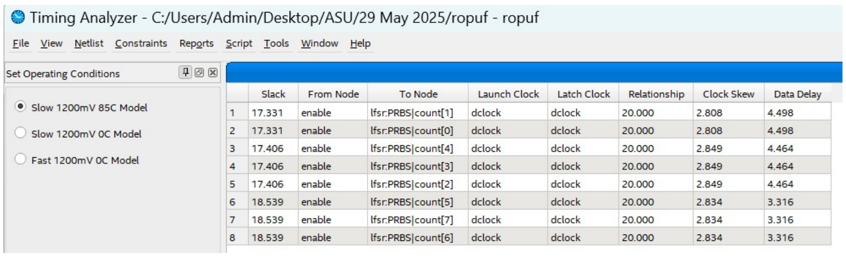
- Timing Calibration: Logic analyzers capture digital signals at high sampling rates ranging from hundreds of MHz to several GHz. However, any misalignment between challenge signals, response outputs, and control clocks can result in corrupted CRPs. The calibration strategies used to address these misalignment issues are discussed below.
    - (i) Use a known reference signal, such as the FPGA internal clock or a test pattern generator, to align acquisition channels.
    - (ii) Apply trigger-based synchronization in the logic analyzer to ensure consistent alignment of challenge vectors with corresponding responses.
- Voltage and Signal Level Calibration: FPGA output signals may degrade due to voltage drop, temperature variations, or I/O mismatches. This can cause the logic analyzer to misinterpret logical '0' and '1' levels. The calibration techniques adopted are,
    - (i) Adjust threshold voltage levels on the logic analyzer to match FPGA I/O standards (e.g., LVTTL, LVCMOS).
    - (ii) Periodically recalibrate using known test vectors to verify that digital transitions are accurately captured.
- Environmental Calibration: PUF responses are known to vary with temperature, supply voltage, and device aging. Environmental calibration ensures that CRPs remain stable and consistent under varying conditions. Calibration strategies include,
    - (i) Use environmental profiling, where CRPs are collected across controlled temperature and voltage ranges, followed by applying corrective models.
    - (ii) Apply ML-based preprocessing such as normalization or majority voting to compensate for environmental drift.
- Noise Filtering and Signal Cleaning: High-frequency noise or transient glitches can distort CRP acquisition and lead to unstable datasets. Techniques used in noise filtering and signal conditioning are,
    - (i) Apply digital filtering techniques (e.g., glitch removal, debouncing) during data preprocessing.
    - (ii) Perform repeated measurements followed by majority voting to ensure transient noise does not bias the dataset.
- Data Alignment and Synchronization: During multi-channel CRP acquisition, timing skew between channels can lead to incorrect challenge–response mapping. The calibration methods used for data alignment and synchronization include,
    - (i) Perform multi-channel skew calibration by applying the same known signal to all acquisition channels and adjusting offsets accordingly.
    - (ii) Use post-processing alignment algorithms to re-synchronize challenge–response mapping before ML training.
- Statistical Calibration for ML Training: Before feeding CRPs into machine learning models, statistical calibration ensures data integrity and uniformity for reliable analysis. The calibration techniques include,

(i) Compute intra-class and inter-class metrics to evaluate the reliability and uniqueness of CRPs.

(ii) Apply whitening techniques (e.g., Linear Feedback Shift Register (LFSR) or hash-based methods) to eliminate bias in raw PUF data.

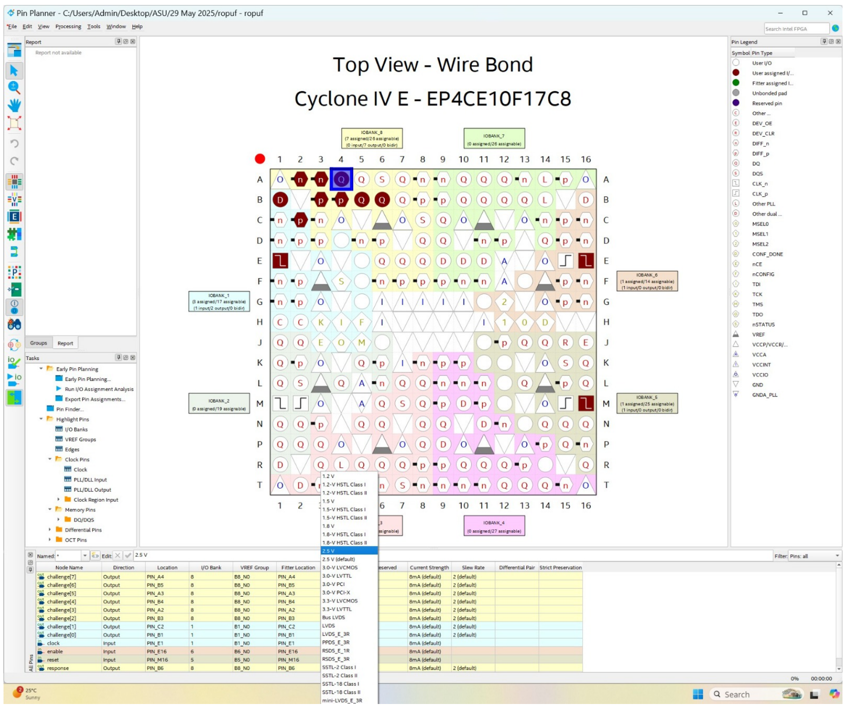
(iii) Normalize datasets to prevent ML models from being influenced by imbalanced or skewed response distributions.

Intel Quartus Prime, illustrated in Figure 5, was employed to achieve timing alignment and voltage level adjustment, with on-chip phase-locked loops providing accurate clock alignment. The Timing Analyzer conducted static timing analysis across the entire design by evaluating specified data times, data arrival times, and clock arrival times in order to verify correct operation and identify potential timing violations. It establishes the timing relationships that must be satisfied for reliable system functionality. Signal synchronization across different clock domains is specified using Synopsys Design Constraints, including clock definitions and input-output delay constraints, to ensure positive setup and hold time margins. During implementation, the LVTTL IO standard was assigned to each pin through the Pin Planner, guaranteeing voltage compatibility with the external logic analyzer.


Timing Analyzer - C:/Users/Admin/Desktop/ASU/29 May 2025/ropuf - ropuf
File View Netlist Constraints Reports Script Tools Window Help
Set Operating Conditions
Slow 1200mV 85C Model
Slow 1200mV 0C Model
Fast 1200mV 0C Model

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	17.331	enable	lfsr:PRBS|count[1]	dclock	dclock	20.000	2.808	4.498
2	17.331	enable	lfsr:PRBS|count[0]	dclock	dclock	20.000	2.808	4.498
3	17.406	enable	lfsr:PRBS|count[4]	dclock	dclock	20.000	2.849	4.464
4	17.406	enable	lfsr:PRBS|count[3]	dclock	dclock	20.000	2.849	4.464
5	17.406	enable	lfsr:PRBS|count[2]	dclock	dclock	20.000	2.849	4.464
6	18.539	enable	lfsr:PRBS|count[5]	dclock	dclock	20.000	2.834	3.316
7	18.539	enable	lfsr:PRBS|count[7]	dclock	dclock	20.000	2.834	3.316
8	18.539	enable	lfsr:PRBS|count[6]	dclock	dclock	20.000	2.834	3.316

(**a**) Timing analyzer



Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	eserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
challenge[7]	Output	PIN_A4	8	B8_N0	PIN_A4		8mA (default)	2 (default)		
challenge[6]	Output	PIN_B5	8	B8_N0	PIN_B5		8mA (default)	2 (default)		
challenge[5]	Output	PIN_A3	8	B8_N0	PIN_A3		8mA (default)	2 (default)		
challenge[4]	Output	PIN_B4	8	B8_N0	PIN_B4		8mA (default)	2 (default)		
challenge[3]	Output	PIN_A2	8	B8_N0	PIN_A2		8mA (default)	2 (default)		
challenge[2]	Output	PIN_B3	8	B8_N0	PIN_B3		8mA (default)	2 (default)		
challenge[1]	Output	PIN_C2	1	B1_N0	PIN_C2		8mA (default)	2 (default)		
challenge[0]	Output	PIN_B1	1	B1_N0	PIN_B1		8mA (default)	2 (default)		
clock	Input	PIN_E1	1	B1_N0	PIN_E1		8mA (default)			
enable	Input	PIN_E16	6	B6_N0	PIN_E16		8mA (default)			
reset	Input	PIN_M16	5	B5_N0	PIN_M16		8mA (default)			
response	Output	PIN_B6	8	B8_N0	PIN_B6		8mA (default)	2 (default)		

(**b**) Pin Planner

Figure 5. Intel Quartus Prime Tools.

5. Results and Discussion

PUFs exploit the device's inherent physical randomness to generate unique and repeatable responses corresponding to challenge inputs. The key metric for measuring randomness, i.e., the balance between 0 s and 1 s in the output, is the relative frequency of bit 1 across all generated responses. This frequency provides a mathematical means to assess how uniformly bit 1 appears, which is critical in determining the PUF's effectiveness for security purposes. In security applications, high unpredictability and an even distribution of binary values are essential. As expressed in Equation (18), relative frequency is calculated by dividing the total number of 1 s by the total number of response bits, thus providing an absolute indicator of the PUF's randomness and, by extension, its reliability and suitability for security applications.

$$p = \frac{1}{KL} \sum_{k=1}^{K} \sum_{l=1}^{L} b_{k,l} \tag{18}$$

where p and K represent the relative frequency of bit 1 in all responses and the total number of responses, while L denotes the response length. In contrast, k and l refer to the response kth and the position lth bit in the response output, respectively.

$$H = -\log_2 \max(p, 1-p) \tag{19}$$

Table 5 lists the uniqueness and uniformity values, which fall within the range of published RO PUF designs, but its reliability is noticeably lower than that of most prior works. However, lower reliability strengthens resilience against ML attacks as it introduces label noise into the challenge response pairs, making the PUF's input–output mapping harder to learn.

Table 5. Comparison of RO PUF metrics reported in the literature.

Reference	Uniqueness	Uniformity	Reliability	Description
[63]	47.64%, 45.15%	49.8%, 48%	98.5%, 96%	RO PUF using three and five stage oscillators on Artix seven FPGA with XOR and inverter logic.
[64]	50.1%	49.45%	98.33%	CLU-based design using XOR and XNOR to create a low hardware CRO PUF.
[65]	49.23%	49.76%	98.05%	A lightweight configurable RO PUF that combines RRAM with CMOS inverters.
[66]	48.64%	46.78%	86%	Strong RO PUF (BST RPUF) designed for improved CRP count and stable responses.
[67]	49.2%	49.8%	97.6%	RO PUF design used as a hardware security primitive for IoT applications.
[68]	44.46%, 47.33%, 47.48%	59.61%, 60.62%, 62.89%	97.96%, 98.09%, 99.16%	Uses one hundred RO blocks with five, eleven, and twenty stages for response generation.
[66]	48.64%	46.78%	BER < 10^{-9}	Highly reliable BST RPUF robust against ML-based modeling attacks.
This work	49.1%	56.86%	60.13%	A configurable RO PUF architecture is implemented on an FPGA platform and its resilience against ML attacks is measured.

To assess the randomness of a bit sequence, Equation (19) defines H as the minimum entropy anticipated from PUF outputs to exhibit a uniform distribution. H peaks at 1 when $p = 0.5$ and reaches its lowest at 0 when $p = 0$ or $p = 1$. The optimal value of p is 0.5 for a binary system, as it produces the maximum entropy of 1 bit and represents the maximum uncertainty or randomness in a system. It also provides the strongest unpredictability and ensures resilience against cloning and prediction. The bitwise probability p and entropy H are plotted against varying CRPs in Figure 6. This study also conducts a vulnerability assessment of the proposed FPGA-based RO-PUF design against machine learning (ML) modeling attacks. The challenge–response pair (CRP) data is split into training and testing sets in an 80% to 20% ratio to build the attack model. This setup enables a clear and meaningful evaluation of the prediction accuracy of various ML models, with CRP lengths scaling up to 80,000. The graph shown in Figure 7 illustrates that the security of the PUF decreases as more CRPs become available. LR and MLP achieve the highest prediction accuracy, meaning they pose the most effective modeling threat, while the SVM also shows moderate attack capability. In contrast, k-NN performs poorly with low and unstable accuracy, indicating strong resistance against such attacks. Accuracy values rise with increasing challenge response pairs and peak around 32,000 to 33,000, after which improvements saturate or slightly decline. Since lower accuracy corresponds to stronger security, the physical unclonable function remains most robust against attacks with limited challenge response pairs and when weaker models like k-NN are used.

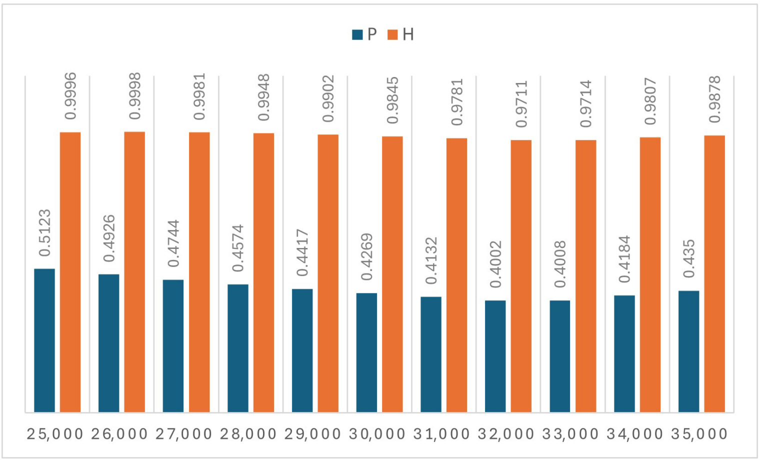


Figure 6. Bitwise probability (P) and entropy (H) against CRP numbers.

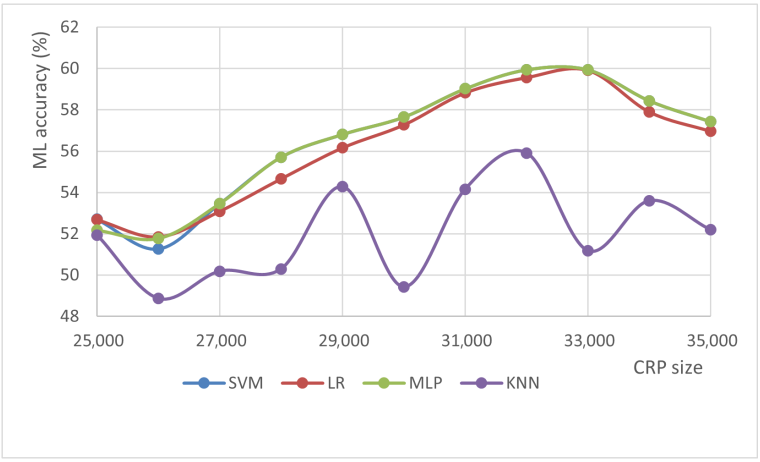


Figure 7. ML accuracy (%) versus varying number of CRP's.

The confusion matrix shown in Figure 8 evaluates the performance of a classification model. The top-left cell represents true positives (TP), indicating the positive instances correctly identified by the model. The bottom-left cell shows false positives (FP), which are negative cases incorrectly classified as positive, also known as type I errors. The top-right cell displays the number of false negatives (FN), referring to positive instances that were mistakenly predicted as negative. Finally, the bottom-right cell indicates true negatives (TN), where the model correctly identified negative instances. The sum of these four values gives the total number of predictions made by the model. Metrics such as accuracy, precision, recall, and F1-score can be derived from the confusion matrix to provide deeper insight into the model's performance. Equation (20) presents an estimation of model accuracy based on the values in the confusion matrix.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{20}$$

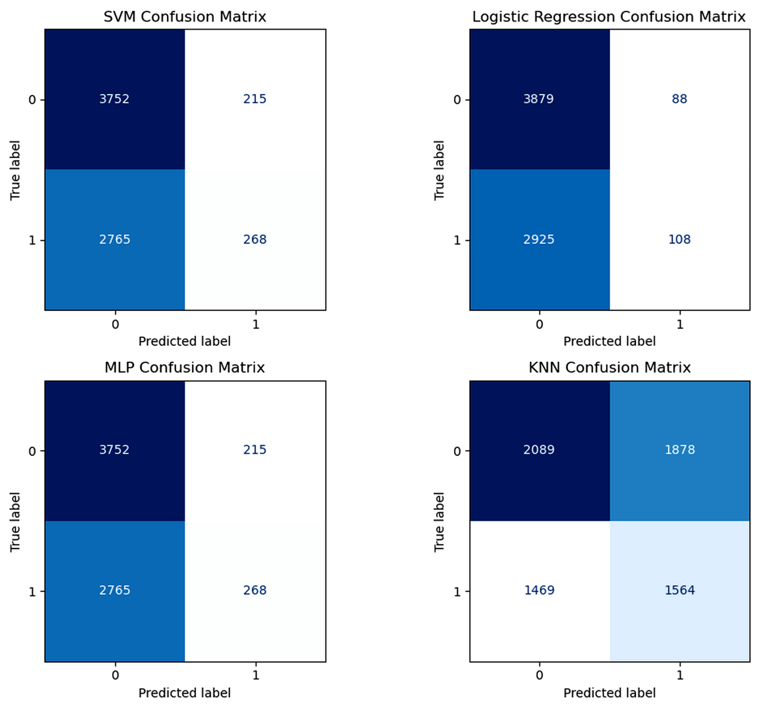


Figure 8. Confusion matrix of ML models (trained using 35K CRPs).

Modeling attacks has become a significant security concern, aiming to imitate the challenge–response behavior of PUFs and replicate their secret authentication keys. In the context of lightweight PUF-based systems, these attacks are generally classified into three main types: (a) machine learning (ML)-based software attacks, (b) side-channel attacks, and (c) hybrid attacks that combine ML techniques with side-channel analysis. A generic model, often polynomial, is used in ML-based modeling attacks for approximating CRP datasets. Adversaries divide the compromised CRP dataset into training and testing subsets. An iterative process is carried out in which challenges are input into the developed PUF model and the corresponding responses are predicted. The error between the predicted and actual response is used to estimate a loss function; subsequently, the model parameters are updated using optimization strategies such as gradient descent in logistic regression (LR) or maximum likelihood estimation for support vector machines (SVMs). Finally, the model's ability to replicate the actual PUF behavior is validated using the test set [69]. However, the effectiveness and precision of ML-based attacks decline as the structural complexity of the PUF increases. In particular, strong PUFs, such as Ring Oscillator (RO) and Arbiter PUFs, which incorporate a high degree of nonlinear logic are more resistant to such modeling efforts.

5.1. Randomness Results

Table 6 summarizes the results of the NIST randomness tests conducted on the RO PUF CRP data. The p-value represents the likelihood that the observed bit sequence could have been produced by an ideal random source under the null hypothesis. In general, a sequence is considered random if its p-value exceeds the predefined significance level, typically 0.01, whereas values below this threshold indicate potential nonrandom behavior. It is important to emphasize that no single p-value alone confirms perfect randomness. Instead, a robust PUF should yield p-values that are uniformly distributed between 0 and 1 across various tests and sequences. Such a uniform distribution, along with the majority of p-values surpassing the 0.01 threshold, indicates that the CRP data exhibit statistical characteristics consistent with true randomness. Thus, it is concluded that the FPGA-based RO-PUF can effectively resist ML modeling attacks, as evidenced by the results in Table 7. The ML models perform no better than random guessing (50% accuracy for binary responses), and the RO-PUF successfully prevents learning patterns, given that the highest accuracy achieved is approximately 60%.

Table 6. RO PUF NIST statistical test results.

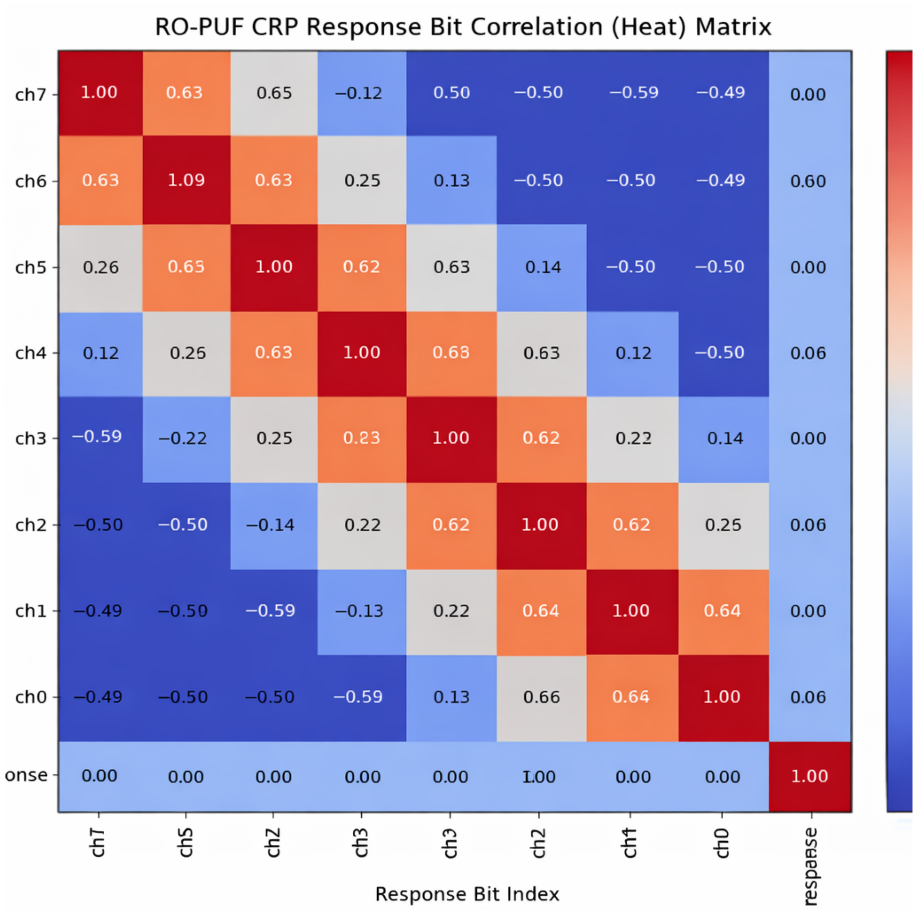
Tests	*p*-Values
Block Frequency	0.444570
Cumulative sums	0.907298
FFT	0.561658
Frequency Test	0.583604
Runs	0.677681
Longest run of ones	0.164698
Rank	0.945607
Non overlapping Template matching	0.51082702 (Average)
Overlapping template matching	0.711526
Universal statistical	0.829717
Approximate entropy	0.120839
Random excursions	0.332701
Random excursions variant	0.447202222
Serial test	0.2013255
Linear complexity	0.420000

Table 7. ML model accuracy (%) against varying CRPs.

Algorithm	25K	26K	27K	28K	29K	30K	31K	32K	33K	34K	35K
SVM	52.7	51.27	53.46	55.71	56.81	57.65	59.02	59.94	59.94	58.44	57.43
LR	52.68	51.85	53.09	54.66	56.16	57.27	58.82	59.56	59.91	57.9	56.96
MLP	52.17	51.77	53.46	55.71	56.81	57.65	59.02	59.94	59.94	58.44	57.43
KNN	51.94	48.87	50.19	50.30	54.28	49.43	54.16	55.91	51.18	53.6	52.19

5.2. Correlation Matrix

The correlation matrix of the RO PUF CRP data in Figure 9 illustrates the degree of dependency between different output-bit values generated by the RO PUF. Each cell in the matrix represents a correlation coefficient between two response bits, ranging from -1 to +1. The diagonal elements show perfect correlation (value of 1), indicating that each bit is fully correlated with itself.



	ch7	ch5	ch2	ch3	ch3	ch2	ch1	ch0	response
ch7	1.00	0.63	0.65	−0.12	0.50	−0.50	−0.59	−0.49	0.00
ch6	0.63	1.09	0.63	0.25	0.13	−0.50	−0.50	−0.49	0.60
ch5	0.26	0.65	1.00	0.62	0.63	0.14	−0.50	−0.50	0.00
ch4	0.12	0.25	0.63	1.00	0.63	0.63	0.12	−0.50	0.06
ch3	−0.59	−0.22	0.25	0.23	1.00	0.62	0.22	0.14	0.00
ch2	−0.50	−0.50	−0.14	0.22	0.62	1.00	0.62	0.25	0.06
ch1	−0.49	−0.50	−0.59	−0.13	0.22	0.64	1.00	0.64	0.00
ch0	−0.49	−0.50	−0.50	−0.59	0.13	0.66	0.64	1.00	0.06
onse	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	1.00

Figure 9. RO PUF Correlation Matrix.

The colors transition from red along the diagonal to blue in the off-diagonal regions, reflecting varying levels of inter-bit correlation. From the heatmap, it can be observed that adjacent bits exhibit a moderate degree of correlation, while distant bits tend to be weakly correlated or nearly independent. This pattern suggests that neighboring ring oscillators may share certain environmental or structural influences, such as local routing, power distribution, or temperature variations, thereby introducing partial dependency among nearby bits. However, as the distance between bits increases, the influence diminishes, and the correlation between them approaches zero. The overall low inter-bit correlation indicates that the RO PUF produces outputs that are largely independent and random, thereby fulfilling one of the key requirements of a strong PUF design. The slight correlation observed between adjacent bits is expected in practical implementations due to physical proximity effects. To further enhance independence, post-processing techniques such as XOR-based bit mixing or improved oscillator placement strategies can be employed. In summary, the correlation matrix demonstrates that the RO PUF achieves good randomness and uniqueness characteristics, making it suitable for secure identification and authentication in hardware security applications.

5.3. Receiver Operating Characteristic (ROC)

The ROC analysis for the RO-PUF in Figure 10 shows that all models—Logistic Regression, Random Forest, Gradient Boosting, SVM, and MLP—produced Area Under the Curve (AUC) values close to 0.49, with ROC curves nearly overlapping the diagonal reference line. This behavior indicates that none of the models could distinguish the RO-PUF responses from random guessing, thereby confirming its strong resistance to ML-based prediction. The intrinsic randomness and frequency-based variations of the oscillators introduce complex, nonlinear behavior that conventional ML algorithms cannot capture effectively using standard training procedures.

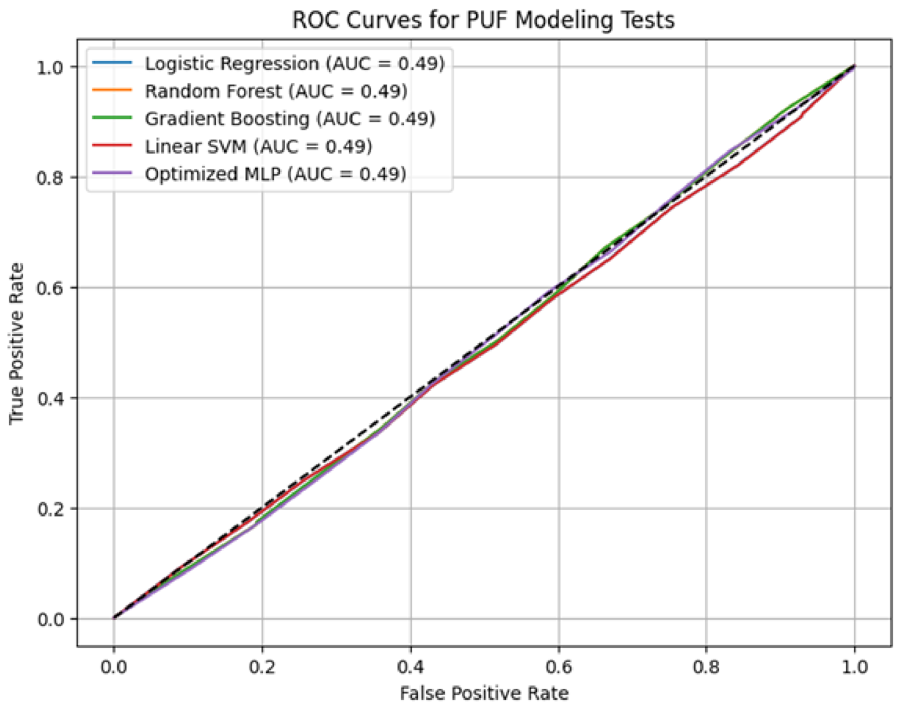
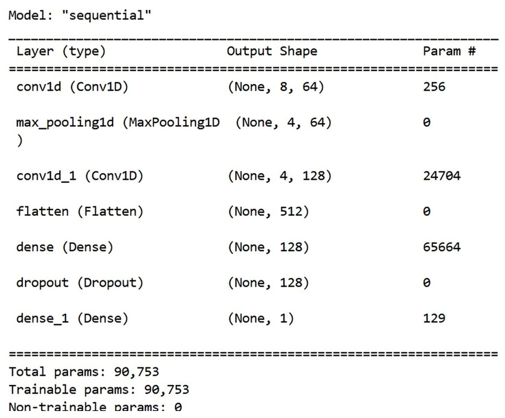


Figure 10. Receiver Operating Characteristic of RO PUF.

5.4. Deep Learning

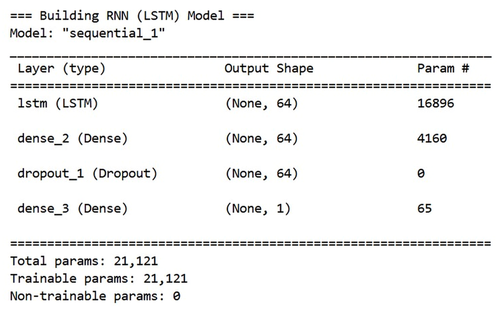
Deep learning is a specialized branch of machine learning and artificial intelligence that employs multi-layered neural network architectures. In these models, data propagate through several successive layers, where early layers extract low-level features and deeper layers progressively combine them to form higher-level, more abstract representations. This hierarchical feature learning makes deep learning particularly effective for large-scale data analytics and statistical learning, such as the identification of cyber threats in intelligent and adaptive security systems. However, implementing large and complex deep learning models in edge computing environments remains challenging due to constraints on computation, memory, and energy resources [70]. Deep neural networks can approximate highly complex functions, with their representational power determined by the number of hidden layers and the number of neurons per layer. Among various deep learning architectures, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have demonstrated strong performance in many real-world applications [71]. CNNs are feedforward networks that exploit convolutional operations to automatically extract discriminative features from input data. A typical CNN architecture comprises convolutional layers, pooling layers for dimensionality reduction, and fully connected layers for classification, making CNNs highly effective for feature extraction and pattern recognition tasks [72]. In contrast, RNNs are designed with internal memory mechanisms that enable them to model temporal and sequential dependencies. By processing inputs in a sequence rather than independently, RNNs are well suited for applications involving time-dependent or ordered data [73].

As shown in Figure 11, a 1D CNN is developed using the Sequential API, comprising two Conv1D layers that successively extract low-level and then high-level temporal features, followed by a MaxPooling1D layer, which performs dimensionality reduction. The resulting feature maps are flattened and fed into a fully connected dense layer to capture global relationships, while a dropout layer is incorporated to mitigate overfitting. The network concludes with a single-neuron output layer that generates a single prediction value, thereby making the model suitable for both regression and binary classification tasks. In addition, an RNN-based model employing a Long Short-Term Memory (LSTM) architecture is implemented to assess the learnability of the RO-PUF responses.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 8, 64)             256

 max_pooling1d (MaxPooling1D  (None, 4, 64)            0
 )

 conv1d_1 (Conv1D)           (None, 4, 128)            24704

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 128)               65664

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 90,753
Trainable params: 90,753
Non-trainable params: 0
```

(**a**) CNN model parameters

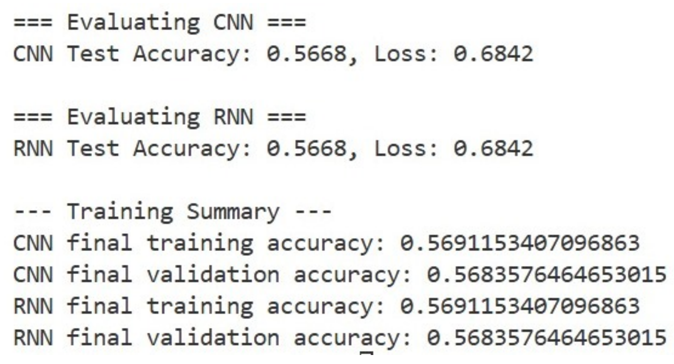


```
=== Building RNN (LSTM) Model ===
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                16896

 dense_2 (Dense)             (None, 64)                4160

 dropout_1 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 21,121
Trainable params: 21,121
Non-trainable params: 0
```

(**b**) RNN model parameters

Figure 11. Deep Neural Networks modeling parameters.

The evaluation results in Figure 12 indicate that both the CNN and RNN (LSTM-based) models exhibit very close performance metrics. On the test dataset, each model achieves an accuracy of 0.5668 with a corresponding loss of 0.6842, suggesting comparable generalization capabilities. During training, the CNN and RNN reach nearly identical final training accuracies of approximately 0.569, while their final validation accuracies are also closely matched at about 0.568. This consistency between training and validation performance implies stable learning behavior with no significant overfitting; however the relatively modest accuracy values indicate that both models face challenges in effectively learning the underlying patterns present in the RO-PUF responses.



```
=== Evaluating CNN ===
CNN Test Accuracy: 0.5668, Loss: 0.6842

=== Evaluating RNN ===
RNN Test Accuracy: 0.5668, Loss: 0.6842

--- Training Summary ---
CNN final training accuracy: 0.5691153407096863
CNN final validation accuracy: 0.5683576464653015
RNN final training accuracy: 0.5691153407096863
RNN final validation accuracy: 0.5683576464653015
```

Figure 12. DNN models accuracy percentage.

6. Conclusions

Advances in machine learning represent a double-edged sword. While they enable powerful data-driven solutions, they have also become increasingly affordable and easy to deploy due to the availability of mature frameworks such as TensorFlow and OpenAI Gym. Coupled with the growing accessibility of high-performance computing platforms, these developments have lowered the barrier to using machine learning algorithms for malicious purposes, including modeling attacks on security primitives. Physically Unclonable Functions offer a cost-effective and reliable method for authenticating IoT devices due to their unique physical characteristics and ease of implementation. Nevertheless, despite being labeled as unclonable, PUFs can be vulnerable to modeling attacks if an adversary gains access to a portion of their challenge–response pairs (CRPs). RO PUFs offer a lightweight, hardware-rooted security primitive that aligns well with the constraints of edge-based IoT systems. Their low power consumption and minimal hardware footprint make them suitable for large-scale deployments while enabling scalable device authentication and secure key generation at the network edge. Interoperability with existing security protocols can be achieved by leveraging RO PUF outputs as root-of-trust anchors, rather than replacing established cryptographic frameworks. Furthermore, the robustness of RO-PUF-based security mechanisms can be enhanced by combining PUF-derived keys with post-quantum cryptographic schemes, thereby strengthening resilience against future computational threats. In this work, we present a machine learning (ML)-resistant strong Ring Oscillator PUF (RO-PUF) architecture implemented on an FPGA. The design utilizes 512 RO chains, each consisting of five inverters, and employs an eight-bit challenge to generate each response bit. Experimental results demonstrate that the proposed RO-PUF significantly improves resilience against ML-based modeling attacks, effectively hindering adversaries from constructing accurate predictive models.

Author Contributions: Conceptualization, A.M.S., M.R.I. and M.H.H.; methodology, A.M.S.; software, A.M.S., M.H.H. and M.B.; validation, A.M.S., M.R.I. and M.H.H.; formal analysis, A.M.S., M.H.H., S.A.Z., A.R.b.N. and M.B.; investigation, A.M.S. and M.R.I.; resources, M.R.I., M.H.H. and M.B.; data curation, A.M.S. and M.R.I.; writing—original draft preparation, A.M.S.; writing—review and editing, M.R.I., M.H.H., S.A.Z., A.R.b.N. and M.B.; visualization, M.H.H., S.A.Z. and A.R.b.N.; supervision, M.R.I. and M.H.H.; project administration, A.M.S.; funding acquisition, A.M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by A'Sharqiyah University, Oman-Internal Research Grant (IRG-16), 2024-26, "Intrusion detection in an IoT network through Machine Learning (ML) of hardware characteristics" and The APC was funded by A'Sharqiyah University, Oman.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
AUC	Area Under the Curve
CEC	Collaborative Edge Computing
CRPs	Challenge Response Pairs
DDoS	Distributed Denial of Service
DT	Decision Trees
EC	Edge Computing
FN	False Negatives
FP	False Positives
FPGAs	Field Programmable Gate Arrays
ICs	Integrated Circuits
IDS	Intrusion Detection Systems
IoT	Internet of Things
IP	Intellectual Property
KNN	K-Nearest Neighbor
LR	Logistic Regression

LSTM	Long Short-Term Memory
MiTM	Man in the Middle
ML	Machine Learning
MLP	Multilayer Perceptron
NIST	National Institute of Standards and Technology
PUF	Physical Unclonable Function
RF	Random Forests
RO	Ring Oscillator
STS	Statistical Test Suite
SVM	Support Vector Machine
TN	True Negatives
TP	True Positives

References

1. Albreem, M.A.; Sheikh, A.M.; Alsharif, M.H.; Jusoh, M.; Yasin, M.N.M. Green Internet of Things (GIoT): Applications, Practices, Awareness, and Challenges. *IEEE Access* **2021**, *9*, 38833–38858. [CrossRef]
2. Kokila, M.; Srinivasa Reddy, S. Authentication, Access Control and Scalability Models in Internet of Things Security—A Review. *Cyber Secur. Appl.* **2024**, 100057. [CrossRef]
3. Albreem, M.A.; Sheikh, A.M.; Bashir, M.J.; El-Saleh, A.A. Towards green Internet of Things (IoT) for a sustainable future in Gulf Cooperation Council countries: Current practices, challenges and future prospective. *Wirel. Netw.* **2023**, *29*, 539–567. [CrossRef]
4. Gupta, D.; Rani, S.; Raza, S.; Qureshi, N.M.F.; Mansour, R.F.; Ragab, M. Security paradigm for remote health monitoring edge devices in internet of things. *J. King Saud Univ.-Comput. Inf. Sci.* **2023**, *35*, 101478. [CrossRef]
5. Badhib, A.; Alshehri, S.; Cherif, A. A robust device-to-device continuous authentication protocol for the internet of things. *IEEE Access* **2021**, *9*, 124768–124792. [CrossRef]
6. Giguère, F. Edge Computing in IoT: Transforming Security Paradigms Through Advanced Forensics. *ResearchGate Prepr.* **2024**. [CrossRef]
7. Thomas, C. IoT and Edge Computing Convergence: Revolutionizing Security and Forensic Applications. *ResearchGate Prepr.* **2024**. [CrossRef]
8. Sheikh, A.M.; Islam, M.R.; Habaebi, M.H.; Kabbani, A.; Zabidi, S.A.; Najeeb, A.R.B. Securing the IoT Edge Devices Using Advanced Digital Technologies. *Asian J. Electr. Electron. Eng.* **2024**, *4*, 52–60. [CrossRef]
9. Kong, L.; Tan, J.; Huang, J.; Chen, G.; Wang, S.; Jin, X.; Zeng, P.; Khan, M.; Das, S.K. Edge-computing-driven internet of things: A survey. *ACM Comput. Surv.* **2022**, *55*, 1–41. [CrossRef]
10. Raza, A.; Yusoff, M.Z.; Khan, M.; Baloch, M.; Shaikh, A.M.; Chauhdary, S.T. Solar Energy Optimal Grid Integration Through Machine Learning Techniques. *Int. J. Energy Convers.* **2024**, *12*, 40–48. [CrossRef]
11. Alnuaimi, R.A.; Almasalmeh, R.K.; Alhammadi, E.A.; Hassan, G.E.B.M.E.; Zia, H. Optimizing Edge Security: Comprehensive Analysis and Mitigation Strategies for Securing Edge Computing. In Proceedings of the 2023 9th International Conference on Optimization and Applications (ICOA), Abu Dhabi, United Arab Emirates, 5–6 October 2023; pp. 1–7.
12. Manan, A. Efficient 16 nm SRAM Design for FPGA's. In Proceedings of the 2018 5th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India, 22–23 February 2018; pp. 457–461.
13. Manan, A. Implementation of image processing algorithm on FPGA. *AKGEC J. Technol.* **2006**, *2*, 25–28.
14. Lata, K.; Cenkeramaddi, L.R. FPGA-based PUF designs: A comprehensive review and comparative analysis. *Cryptography* **2023**, *7*, 55. [CrossRef]
15. Sheikh, A.M.; Islam, M.R.; Habaebi, M.H.; Zabidi, S.A.; Najeeb, A.R.B.; Basahel, A. Machine Learning (ML) assisted Edge security framework on FPGAs. In Proceedings of the 2023 9th International Conference on Computer and Communication Engineering (ICCCE), Kuala Lumpur, Malaysia, 15–16 August 2023; pp. 155–160.
16. Yu, S.; Park, K.; Park, Y. A Machine Learning Attack-Resistant PUF-based Robust and Efficient Mutual Authentication Scheme in Fog-enabled IoT Environments. *IEEE Internet Things J.* **2025**, *12*, 20652–20669. [CrossRef]
17. Sánchez, P.M.; Celdrán, A.H.; Bovet, G.; Pérez, G.M. Adversarial attacks and defenses on ML-and hardware-based IoT device fingerprinting and identification. *Future Gener. Comput. Syst.* **2024**, *152*, 30–42. [CrossRef]
18. Tripathy, S.; Rai, V.K.; Mathew, J. MARPUF: Physical unclonable function with improved machine learning attack resistance. *IET Circuits Devices Syst.* **2021**, *15*, 465–474. [CrossRef]
19. Aparicio-Téllez, R.; Garcia-Bosque, M.; Díez-Señorans, G.; Celma, S. Novel machine learning-resistant RO-based PUF optimized for IoT device authentication. *IEEE Access* **2025**, *13*, 46147–46160. [CrossRef]
20. Sheikh, A.M.; Islam, M.R.; Habaebi, M.H.; Zabidi, S.A.; Najeeb, A.R.B.; Kabbani, A. Integrating Physical Unclonable Functions with Machine Learning for the Authentication of Edge Devices in IoT Networks. *Future Internet* **2025**, *17*, 275. [CrossRef]

21. Anandakumar, N.N.; Hashmi, M.S.; Sanadhya, S.K. Design and analysis of FPGA-based PUFs with enhanced performance for hardware-oriented security. *ACM J. Emerg. Technol. Comput. Syst. (JETC)* **2022**, *18*, 72. [CrossRef]
22. Shen, T.; Ding, L.; Sun, J.; Jing, C.; Guo, F.; Wu, C. Edge Computing for IoT Security: Integrating Machine Learning with Key Agreement. In Proceedings of the 2023 3rd International Conference on Consumer Electronics and Computer Engineering (ICCECE), Guangzhou, China, 6–8 January 2023; pp. 474–483.
23. Aarella, S.G.; Mohanty, S.P.; Kougianos, E.; Puthal, D. Fortified-edge: Secure PUF certificate authentication mechanism for edge data centers in collaborative edge computing. In Proceedings of the Great Lakes Symposium on VLSI 2023, Knoxville, TN, USA, 5–7 June 2023; pp. 249–254.
24. Cheng, G.; Chen, Y.; Deng, S.; Gao, H.; Yin, J. A Blockchain-Based Mutual Authentication Scheme for Collaborative Edge Computing. *IEEE Trans. Comput. Social Syst.* **2022**, *9*, 146–158. [CrossRef]
25. Liu, G.; Wang, C.; Ma, X.; Yang, Y. Keep your data locally: Federated-learning-based data privacy preservation in edge computing. *IEEE Netw.* **2021**, *35*, 60–66. [CrossRef]
26. Zhang, J.; Shen, C.; Guo, Z.; Wu, Q.; Chang, W. CT PUF: Configurable tristate PUF against machine learning attacks for IoT security. *IEEE Internet Things J.* **2021**, *9*, 14452–14462. [CrossRef]
27. Ebrahimabadi, M.; Younis, M.; Karimi, N. A PUF-based modeling-attack resilient authentication protocol for IoT devices. *IEEE Internet Things J.* **2021**, *9*, 3684–3703. [CrossRef]
28. Shin, H.; Koo, D.; Hur, J. Secure and efficient hybrid data deduplication in edge computing. *ACM Trans. Internet Technol.* **2022**, *22*, 80. [CrossRef]
29. Rullo, A.; Bertino, E.; Ren, K. Guest editorial special issue on intrusion detection for the Internet of Things. *IEEE Internet Things J.* **2023**, *10*, 8327–8330. [CrossRef]
30. Sarkar, A.; Ganguly, S.; Sarkar, P.S.; Chatterjee, S.R. PUF-Based Authentication System with Resilience against Multi-Faceted Attacks for Blockchain-based IoT Networks. In Proceedings of the 2024 IEEE 3rd World Conference on Applied Intelligence and Computing (AIC), Gwalior, India, 27–28 July 2024; pp. 1279–1284.
31. Fan, L.; Huang, Z.; Wang, J.; Zhou, L.; Zhu, Y.; Wang, Q. A Novel Configurable RO-Obfuscated PUF Design with Machine Learning Immunity. In Proceedings of the 2023 International Conference on Networking and Network Applications (NaNA), Qingdao, China, 18–21 August 2023; pp. 680–685.
32. Kareem, H.; Dunaev, D. Machine Learning Vulnerability Assessment of Ring Oscillator Physical Unclonable Functions. In Proceedings of the 2023 International Conference on Control, Automation and Diagnosis (ICCAD), Rome, Italy, 10–12 May 2023; pp. 1–5.
33. Abulibdeh, E.; Saleh, H.; Mohammad, B.; Alqutayri, M.; Santikellur, P. Algorithmically Optimized Configurable Ring Oscillator Puf for Iot Devices. Available online: https://ssrn.com/abstract=5327693 (accessed on 24 November 2025).
34. Miskelly, J.; Gu, C.; Ma, Q.; Cui, Y.; Liu, W.; O'Neill, M. Modelling attack analysis of configurable ring oscillator (CRO) PUF designs. In Proceedings of the 2018 IEEE 23rd International Conference on Digital Signal Processing (DSP), Shanghai, China, 19–21 November 2018; pp. 1–5.
35. Laguduva, V.; Islam, S.A.; Aakur, S.; Katkoori, S.; Karam, R. Machine learning based iot edge node security attack and countermeasures. In Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Miami, FL, USA, 15–17 July 2019; pp. 670–675.
36. Abulibdeh, E.E. Reliable and Efficient Hardware Implementation of PUFs for Secure IoT Applications. Ph.D. Dissertation, Khalifa University of Science, Abu Dhabi, United Arab Emirates (UAE), 2024.
37. Teo, J.H.L.; Hashim, N.A.N.; Ghazali, A.; Hamid, F. Ring oscillator physically unclonable function using sequential ring oscillator pairs for more challenge-response-pairs. *Indones. J. Electr. Eng. Comput. Sci.* **2019**, *13*, 892–901. [CrossRef]
38. Zerrouki, F.; Ouchani, S.; Bouarfa, H. PUF-based mutual authentication and session key establishment protocol for IoT devices. *J. Ambient. Intell. Humaniz. Comput.* **2023**, *14*, 12575–12593. [CrossRef]
39. Gao, Y.; Al-Sarawi, S.F.; Abbott, D. Physical unclonable functions. *Nat. Electron.* **2020**, *3*, 81–91. [CrossRef]
40. Al-Meer, A.; Al-Kuwari, S. Physical unclonable functions (PUF) for IoT devices. *ACM Comput. Surv.* **2023**, *55*, 1–31. [CrossRef]
41. Naveed, A.M.; Chua, K.C.; Sikdar, B. Physical unclonable functions for IoT security. In Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security, Xi'an, China, 30 May 2016.
42. Sheikh, A.M.; Islam, M.R.; Habaebi, M.H.; Zabidi, S.A.; Bin Najeeb, A.R.; Kabbani, A. A Survey on Edge Computing (EC) Security Challenges: Classification, Threats, and Mitigation Strategies. *Future Internet* **2025**, *17*, 175. [CrossRef]
43. Chatterjee, B.; Das, D.; Maity, S.; Sen, S. RF-PUF: Enhancing IoT security through authentication of wireless nodes using in-situ machine learning. *IEEE Internet Things J.* **2018**, *6*, 388–398. [CrossRef]
44. Modarres, A.M.A.; Anzabi-Nezhad, N.S.; Zare, M. A new PUF-based protocol for mutual authentication and key agreement between three layers of entities in cloud-based IoMT networks. *IEEE Access* **2024**, *12*, 21807–21824. [CrossRef]

45. Yehoshuva, C.; Adhithan, R.R.; Anandakumar, N.N. A survey of security attacks on silicon based weak PUF architectures. In Proceedings of the International Symposium on Security in Computing and Communication, Chennai, India, 14–17 October 2020; pp. 107–122.
46. Kareem, H.; Dunaev, D. A robust architecture of ring oscillator PUF: Enhancing cryptographic security with configurability. *Microelectron. J.* **2024**, *143*, 106022. [CrossRef]
47. Budnik, M. Design and Evaluation of a Ring Oscillator Based Physically Unclonable Function. Bachelor's Thesis, University of Twente, Enschede, The Netherlands, 2023.
48. Pahlevi, R.R.; Hasegawa, H.; Yamaguchi, Y.; Shimada, H. A Pre-Selection–Enhanced Arbiter PUF for Strengthening PUF-Based Authentication. *IEEE Access* **2025**, *13*, 127526–127544. [CrossRef]
49. Sukarno, P.; Medina, F.R. Enhancing IoT Security: Optimizing PUF Responses through Pre-Processing Techniques. *J. Infotel* **2025**, *17*, 210–228. [CrossRef]
50. Sajadi, A.; Shabani, A.; Alizadeh, B. DC-PUF: Machine learning-resistant PUF-based authentication protocol using dependency chain for resource-constraint IoT devices. *J. Netw. Comput. Appl.* **2023**, *217*, 103693. [CrossRef]
51. Ali, R.; Ma, H.; Hou, Z.; Zhang, D.; Deng, E.; Wang, Y. A reconfigurable arbiter MPUF with high resistance against machine learning attack. *IEEE Trans. Magn.* **2021**, *57*, 1–7. [CrossRef]
52. Anupama, A.; Raja, I.; Roy, D.; Padmakumar, K. A Ring Oscillator-based Strong Physical Unclonable Function with Excellent Resilience to Machine Learning Attacks. *IEEE Internet Things J.* **2025**, *12*, 54816–54829. [CrossRef]
53. Rajput, S.; Dofe, J. Counteracting modeling attacks using hardware-based dynamic physical unclonable function. In Proceedings of the 2023 IEEE International Conference on Cyber Security and Resilience (CSR), Venice, Italy, 31 July–2 August 2023; pp. 586–591.
54. Ferens, M.; Dushku, E.; Kosta, S. When Random is Bad: Selective CRPs for Protecting PUFs against Modeling Attacks. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2024**, *44*, 1648–1661. [CrossRef]
55. Apruzzese, G.; Laskov, P.; de Oca, E.M.; Mallouli, W.; Rapa, L.B.; Grammatopoulos, A.V.; Di Franco, F. The role of machine learning in cybersecurity. *Digit. Threat. Res. Pract.* **2023**, *4*, 1–38. [CrossRef]
56. Shaukat, K.; Luo, S.; Varadharajan, V.; Hameed, I.A.; Chen, S.; Liu, D.; Li, J. Performance comparison and current challenges of using machine learning techniques in cybersecurity. *Energies* **2020**, *13*, 2509. [CrossRef]
57. Bout, E.; Loscri, V.; Gallais, A. How machine learning changes the nature of cyberattacks on IoT networks: A survey. *IEEE Commun. Surv. Tutor.* **2021**, *24*, 248–279. [CrossRef]
58. Huang, S.; Cai, N.; Pacheco, P.P.; Narrandes, S.; Wang, Y.; Xu, W. Applications of support vector machine (SVM) learning in cancer genomics. *Cancer Genom. Proteom.* **2018**, *15*, 41–51.
59. Veisi, H. Introduction to SVM. In *Learning with Fractional Orthogonal Kernel Classifiers in Support Vector Machines: Theory, Algorithms and Applications*; Springer: Singapore, 2023; pp. 3–18.
60. Olowe, K.J.; Edoh, N.L.; Zouo, S.J.C.; Olamijuwon, J. Comprehensive review of logistic regression techniques in predicting health outcomes and trends. *World J. Adv. Pharm. Life Sci.* **2024**, *7*, 16–26. [CrossRef]
61. ScienceDirect Topics. Logistic Regression Model. Available online: https://www.sciencedirect.com/topics/computer-science/logistic-regression-model (accessed on 26 December 2025).
62. Marton, K.; Suciu, A. On the interpretation of results from the NIST statistical test suite. *Sci. Technol.* **2015**, *18*, 18–32.
63. Hazari, N.A.; Alsulami, F.; Oun, A.; Niamat, M. Performance analysis of XOR-inverter based ring oscillator PUF for hardware security. In Proceedings of the 2019 IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 15–19 July 2019; pp. 253–256.
64. Kareem, H.; Dunaev, D. A novel low hardware configurable ring oscillator (CRO) PUF for lightweight security applications. *Microprocess. Microsyst.* **2024**, *104*, 104989. [CrossRef]
65. Cui, Y.; Wang, C.; Liu, W.; Gu, C.; O'Neill, M.; Lombardi, F. Lightweight configurable ring oscillator PUF based on RRAM/CMOS hybrid circuits. *IEEE Open J. Nanotechnol.* **2020**, *1*, 128–134. [CrossRef]
66. He, Z.; Wang, C.; Ke, T.; Zhang, Y.; Cao, W.; Jiang, J. A highly reliable FPGA-based RO PUF with enhanced challenge response pairs resilient to modeling attacks. *IEICE Electron. Express* **2021**, *18*, 20210350. [CrossRef]
67. Zulfikar, Z.; Soin, N.; Hatta, S.F.W.M.; Talip, M.S.A.; Jaafar, A. Routing density analysis of area-efficient ring oscillator physically unclonable functions. *Appl. Sci.* **2021**, *11*, 9730. [CrossRef]
68. Zulfikar, Z.; Soin, N.; Hatta, S.F.W.M.; Talip, M.S.A. Runtime analysis of area-efficient uniform RO-PUF for uniqueness and reliability balancing. *Electronics* **2021**, *10*, 2504. [CrossRef]
69. Bhatia, A.; Bitragunta, S.; Tiwari, K. PUF-AQKD: A Hardware-Assisted Quantum Key Distribution Protocol for Man-in-the-Middle Attack Mitigation. *IEEE Open J. Commun. Soc.* **2025**, *6*, 4923–4942.
70. Ghillani, D. Deep learning and artificial intelligence framework to improve the cyber security. *Authorea Prepr.* **2022**. [CrossRef]
71. Hoffmann, J.; Navarro, O.; Kastner, F.; Janßen, B.; Hubner, M. A survey on CNN and RNN implementations. In Proceedings of the PESARO 2017: The Seventh International Conference on Performance, Safety and Robustness in Complex Systems and Applications, Venice, Italy, 23–27 April 2017; Volume 3.

72. Yu, J.; De Antonio, A.; Villalba-Mora, E. Deep learning (CNN, RNN) applications for smart homes: A systematic review. *Computers* **2022**, *11*, 26.
73. Shiri, F.M.; Perumal, T.; Mustapha, N.; Mohamed, R. A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU. *arXiv* **2023**, arXiv:2305.17473.