MDPI

Article

Fortified-Edge 2.0: Advanced Machine Learning-Driven Framework for Secure PUF-based Authentication in Collaborative Edge Computing

Seema G. Aarella [1,†,‡*], Venkata P. Yanambaka [2,†] and Saraju P. Mohanty [3,†], Elias Kougianos [4,†]

[1] Department of Computer Sci., Austin College; saarella@austincollege.edu
[2] School of Sciences, Texas Woman's University; vyanambaka@twu.edu
[3] Department of Computer Sci. and Eng., University of North Texas; saraju.mohanty@unt.edu
[4] Department of Electrical Engineering, University of North Texas; elias.kougianos@unt.edu

Abstract: This research introduces Fortified-Edge 2.0, a novel authentication framework that addresses critical security and privacy challenges in Physically Unclonable Function (PUF)-based systems for collaborative edge computing (CEC). Unlike conventional methods that transmit full binary Challenge-Response Pairs (CRPs) and risk exposing sensitive data, Fortified-Edge 2.0 employs a machine learning-driven feature abstraction technique to extract and utilize only essential characteristics of CRPs, obfuscating the raw binary sequences. These feature vectors are then processed using lightweight cryptographic primitives, including ECDSA, to enable secure authentication without exposing the original CRP. This eliminates the need to transmit sensitive binary data, reducing the attack surface and bandwidth usage. The proposed method demonstrates strong resilience against modeling attacks, replay attacks, and side-channel threats, while maintaining the inherent efficiency and low power requirements of PUFs. By integrating PUF unpredictability with ML adaptability, this research delivers a scalable, secure, and resource-efficient solution for next-generation authentication in edge environments.

Keywords: Physical Unclonable Function; Security-by-Design; Hardware-Assisted Security; Edge Computing; Secure Authentication; Cybersecurity; Machine Learning; Cryptography; Authentication Protocol; Error Detection; Error Correction

1. Introduction

Edge computing has emerged as a critical paradigm for real-time processing in distributed environments, enabling computation closer to data sources such as Internet-of-Things (IoT) devices, sensors, or local servers [1]. Edge computing approach presents key benefits like reduced latency, bandwidth optimization, enhanced security and privacy, real-time decision making, offline functionality, cost efficiency, and scalable and distributed architecture. These advantages make edge computing a crucial component in supporting IoT systems, and different organizations have proposed various IoT architectures taking various perspectives into account. Edge architecture involves various components like Edge Devices, Edge Nodes, Edge Data Centers, Edge Cloud, and Edge Analytics [2].

The key techniques that enable decentralized edge computing are Virtual Machines (VMs), Containers, Software Defined Networking(SDN), Content Delivery Networks(CDN), Cloudlets, and Micro Data Centers (MDC)/Edge Data Centers (EDC) [3].

Security is crucial in edge computing for protecting data and maintaining system integrity. A comprehensive security architecture for edge computing involves robust technologies like encryption, secure authentication protocols, and other privacy-preserving

Received:
Revised:
Accepted:
Published:

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Future Internet* **2025**, *1*, 0. https://doi.org/

techniques. Advanced cryptographic techniques ensure confidentiality and integrity, while technologies like Artificial Intelligence (AI) and ML enable intelligent decision making and analytics [4]. However, ensuring security at the edge presents significant challenges due to the heterogeneous nature of computing environments and the presence of resource-constrained devices. The availability of high-performance computing and secure network infrastructure varies widely across applications, from well-established *smart cities* to resource-limited *smart villages*.

The increase in devices generating data has led to massive volumes of data that require efficient processing. Consequently, the computing environment has evolved, incorporating various architectures such as cloud, fog, edge, cloudlet, and Mobile Edge Computing (MEC). These architectures facilitate data processing at different levels of the IoT structure. Edge computing, as shown in Figure 1, is one such paradigm that brings computation closer to where data is generated, primarily to reduce latency and alleviate network bandwidth constraints. This shift in computing has paved the way for real-time applications that demand faster response times. Building upon edge computing, collaborative edge computing enables resource sharing across the network to complete computational tasks efficiently. This approach is inherently distributed, with participating computing servers, often referred to as *edge nodes*, located in different geographical regions [5].

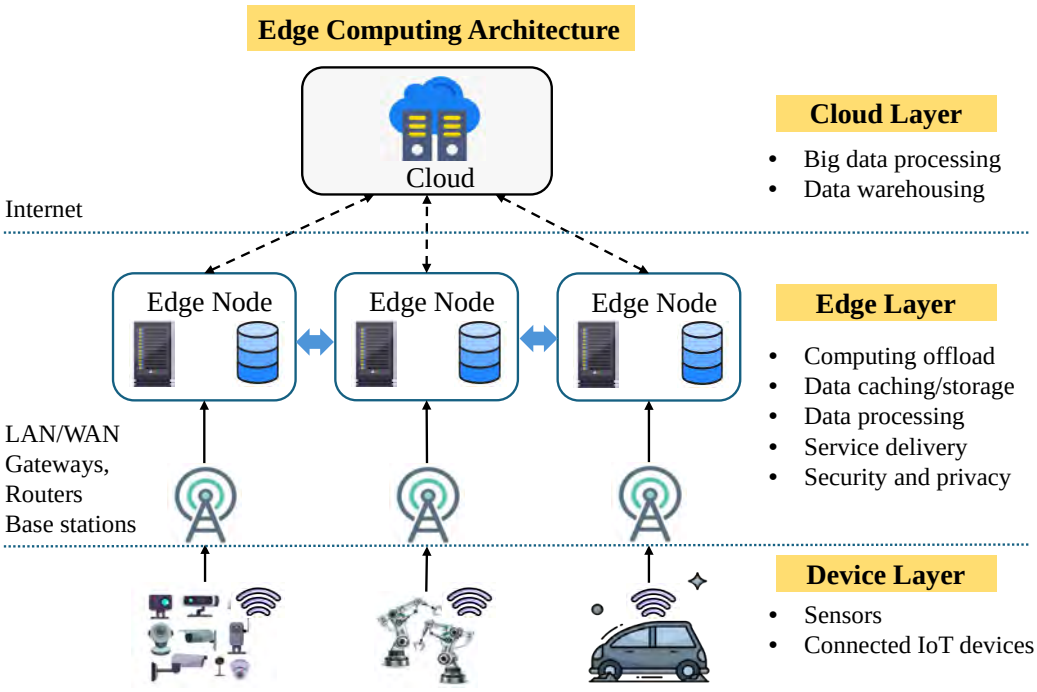

Figure 1. Edge computing architecture.

Edge computing is characterized by processing data closer to its source, enabling faster response times and reducing the amount of data sent to the cloud, which helps conserve network resources. It enhances security and privacy by keeping sensitive data on local devices while also improving system reliability. Additionally, distributed computing minimizes data movement between the cloud and edge devices. While applications requiring extensive computation may still rely on the cloud, real-time applications such as surveillance, real-time monitoring systems, healthcare, and autonomous vehicles benefit from processing at the edge to ensure low latency and quick decision-making.

Edge nodes within this architecture provide the necessary infrastructure, including data processing, storage, networking capabilities, software, and power backup, enabling efficient and resilient operations. However, task computation can be delayed due to node

overload. To address this, edge computing incorporates computation offloading through a controlled process known as load balancing, which transfers queued tasks to the next available edge node (server or data center), ensuring seamless processing without delays. Static and dynamic load balancing methods effectively distribute computing tasks among edge data centers or nodes. Static load balancing methods include Round-robin, minimal combination, and weighted round-robin, and the dynamic methods include predictive load balancing and machine learning-based methods to allocate resources in real-time. Edge computing security systems, therefore, employ a variety of techniques to protect against cyberattacks and ensure secure data transfer between the nodes [6]. Furthermore, the collaborative computing environment is dynamic, requiring nodes to authenticate one another to maintain the security and privacy of both data and computation. Additionally, newly formed nodes must be verified and authenticated, necessitating end-to-end security across a multi-hop network.

Traditional security mechanisms, like cryptography and secure authentication methods originally designed for cloud and high-performance computing environments, are often unsuitable for edge-based deployment. This necessitates the development of lightweight, adaptive, and secure authentication solutions that align with the unique constraints and requirements of edge computing. With the rise of 5G networks and AI-driven automation, edge computing is expected to play a crucial role in the future digital transformation.

Edge computing in resource-constrained collaborative environments demands a lightweight security solution to ensure secure data transmission over untrusted multihop networks. Geographically distributed edge nodes from diverse origins must be verified, and cryptographic authentication protocols play a crucial role in securely authenticating these nodes before task offloading and computation. Physical Unclonable Functions (PUFs) are proposed as a security primitive for a low-power authentication protocol that delivers security against side-channel attacks and data breaches, ensures the integrity of the system [7]. Various PUF architectures are employed that help to develop robust authentication systems for both device and data security for edge computing without the need for explicit key storage. PUF is a hardware security primitive that enables embedded security, incorporating security-by-design(SbD) principles for making security an integral part of the system [8]. PUFs are resistant to cloning, they can generate cryptographic keys on-the-fly, which means sensitive information does not need to be stored in memory [9].While aiming to develop secure and sustainable solutions that are also scalable, it is essential to prioritize energy efficiency and ensure compatibility with resource-constrained environments.

The remainder of this paper is organized as follows: Section 2 presents the novel contributions of the current research. Section 3 provides an in-depth discussion of the state-of-the-art literature related to the research domain. Section 4 explores the security and privacy considerations in edge computing frameworks, with particular emphasis on key challenges and their corresponding mitigation strategies. Section 5 introduces the proposed feature-based authentication process utilizing PUFs. Section 6 describes the experimental setup for implementing the proposed method and presents the results along with a comprehensive analysis. Finally, Section 7 concludes the paper and outlines directions for future research.

2. Novel Contributions of the Current Paper

This research focuses on enhancing the confidentiality, integrity, and scalability of secure authentication systems using PUFs. In a collaborative edge computing environment, where PUFs serve as a security primitive for device authentication and authorization, ensuring data integrity and confidentiality is crucial. In scenarios like load balancing,

where Edge Data Centers (EDCs) must authenticate each other, it is essential to consider how CRP data is stored and managed. Since each EDC stores CRP data locally and communicates it across the network, the data becomes vulnerable to security breaches. To address the challenges of secure data communication and integrity, this research proposes the following novel solutions.

- This research proposes a novel feature-based authentication system that utilizes extracted and vectorized features of PUF data.
- Enhancing data and device integrity by transmitting the vectorized data instead of the raw data using k-mer sequence embeddings.
- Implementing encryption and decryption protocols that use feature vectors as plaintext and convert them to ciphertext using efficient hashing algorithms.
- Utilizing the Elliptic Curve Digital Signature Algorithm (ECDSA) for authentication, eliminating the need for shared secret keys and leveraging asymmetric key cryptography.
- Designing an efficient cryptographic algorithm resistant to man-in-the-middle attacks, ensuring end-to-end data security.
- Communicating both challenges and responses in their vectorized forms to enhance data security and integrity.

2.1. Research Motivation

Cybersecurity at the edge presents significant challenges due to the heterogeneous nature of computing environments. The availability of infrastructure varies across applications, whether in smart cities with well-established networks or smart villages with resource constraints. A resource-constrained environment has unique security and operational challenges that differ significantly from those in a well-developed smart city infrastructure.

In such environments, lightweight yet robust cybersecurity solutions are essential. This research focuses on developing secure authentication protocols for collaborative computing systems that enable seamless processing at the resource-constrained edge. By leveraging PUFs and Machine Learning (ML), this work aims to design a secure, energy-efficient, lightweight, low-power, and low-latency authentication protocol with an emphasis on efficient bandwidth usage.

Security and privacy are critical in cybersecurity, particularly in systems utilizing PUF-based Challenge-Response Pairs (CRPs) for authentication. Secure storage and communication of CRP data are of paramount importance, as any breach, such as fault injection, data manipulation, or data theft, could compromise the entire system. To mitigate these risks, this research introduces a novel feature-based authentication protocol that prevents the direct transmission of CRP data, enhancing data security and resilience against attacks.

Additionally, bandwidth constraints in edge environments pose a significant challenge. Traditional cryptographic protocols are often computationally intensive and require large bandwidth, making them impractical for low-power edge devices. Cryptographic protocols, notably Rivest–Shamir–Adleman (RSA) and Advanced Encryption Standard (AES) are renowned for their robust security features. However, their computational and bandwidth demands could be challenging for resource-constrained environments. To achieve high-level security, RSA requires large key sizes; a 2048-bit RSA key is considered secure for most applications. The use of large keys increases computational overhead and bandwidth requirements. AES is more efficient than RSA in terms of computational requirements, however, the performance of AES depends on underlying hardware and affects the performance throughput, making it unsuitable for real-time applications [10]. Power consumption of the security protocols needs to be consisdered while employing them at

less resourceful environments [11]. This research aims to develop an efficient security mechanism that minimizes bandwidth consumption while maintaining high security, low latency, and adaptability to resource-limited environments.

2.2. Problems Addressed

Edge computing environments are highly heterogeneous, with varying levels of computational power, memory, and connectivity. Smart city applications may have reliable infrastructure, while smart village applications face limited resources and unpredictable connectivity. Conventional cybersecurity solutions are often too computationally expensive for low-power edge devices. The primary goal of this research is to study the security challenges of resource-constrained edge computing environments and present a novel secure authentication protocol that is based on concepts of hardware-assisted security and security-by-design. The key idea is to propose an integrated security model for secure authentication. The research explicitly addresses the challenge of securing authentication in resource-constrained edge environments using PUF, where direct transmission of binary CRP data increases the system's vulnerability to external attacks.

- There is a lack of lightweight yet secure authentication mechanisms for resource-constrained edge. Traditional cryptographic authentication methods like Public Key Infrastructure (PKI) and symmetric encryption consume high computational resources and energy, making them unsuitable.
- Authentication protocols must be resilient against attacks while ensuring low power consumption and minimal latency.
- Edge devices are prone to physical and cyberattacks such as side-channel attacks, replay attacks, cloning, and key extraction. Existing authentication mechanisms lack uniqueness and resistance against cloning, making devices susceptible to unauthorized access.
- Authentication protocols require frequent communication between edge devices and central servers. In collaborative edge computing, there will be frequent communication between the participating clients, like the edge data centers, which increases bandwidth consumption. In remote areas with low network reliability, excessive bandwidth usage can lead to delays and authentication failures.
- Secure storage and transfer of data needs to be ensured to prevent data breaches or authentication failures, especially in systems that use PUF CRP data for authentication purposes.

2.3. Solutions Proposed

This research proposes a novel feature-based authentication mechanism that integrates PUFs and efficient cryptographic algorithms to enhance security at the dynamic collaborative edge. PUFs provide a lightweight yet robust security solution, while cryptographic algorithms ensure secure computation and communication. The proposed feature-based approach focuses on PUF data security, privacy, and reliability, making it a suitable and efficient option for secure authentication in resource-constrained and heterogeneous edge environments.

- PUFs provide hardware-level uniqueness, making each device tamper-resistant and unclonable. PUF-based authentication eliminates the need for storing cryptographic keys, reducing attack vectors such as key theft or tampering.
- ML models enhance authentication by detecting anomalies, improving PUF response stability, and adapting to changing environmental conditions. ML-based feature extraction ensures accurate authentication while maintaining lightweight computation.

- The proposed authentication protocol is designed to be energy-efficient, lightweight, and secure, minimizing computational overhead on constrained devices. It ensures fast authentication while preserving strong cryptographic security.
- The protocol optimizes challenge-response communication to minimize data exchange, reducing bandwidth consumption. By using vectorized challenge-response representations, the system reduces authentication latency without compromising security.

3. Related Prior Research

PUFs have emerged as lightweight hardware security primitives, offering a robust solution for ensuring system integrity and proving authenticity through their inherent uniqueness and unclonability. While PUFs demonstrate resilience against various external threats, the rapid advancement of technology has introduced sophisticated attack vectors aimed at compromising such systems. Consequently, recent research has focused on enhancing PUF-based security by integrating cutting-edge technologies, such as ML and Artificial Intelligence (AI), to reinforce their robustness and adaptability in modern threat landscapes.

Research [12] indicates that PUFs, due to their inherent properties such as irreproducibility, uniqueness, obfuscation, and unpredictability, are highly effective in mitigating a wide range of attacks. These include man-in-the-middle, side-channel, replay, spoofing, reverse engineering, intellectual property hijacking, hardware Trojan insertion, counterfeit hardware, Sybil attacks, denial-of-service (DoS), node capture, and routing-based attacks. Moreover, PUFs eliminate the need to store cryptographic keys in memory, making them particularly suitable for resource-constrained environments due to their low-cost and high-security characteristics.

A study on the use of PUFs for secure authentication in edge data centers demonstrates that PUFs provide an efficient and lightweight solution for implementing security at the edge [13]. By combining effective CRP management with the principles of SbD, PUFs enable security to be integrated as a foundational component of the system. This approach facilitates the creation of secure operating environments with continuous monitoring and enforcement of security throughout the entire life cycle of the device.

ML techniques have been employed to model PUF behavior, which helps in understanding and improving PUF-based security. In the research [14], a deep neural network attack on arbiter-PUF has been studied to identify the vulnerabilities and strengthen the defense. The research shows that deep neural network (DNN) effectively models OAX-PUFs, revealing security vulnerabilities in combinational logic-based PUF designs and calling for new defenses against ML-based attacks.

A study proposes a novel Virtual PUF (VPUF) authentication scheme for IoT networks using a split learning-based encoder–decoder architecture. By offloading computation to the server and transmitting latent representations instead of raw PUF responses, the VPUF reduces power consumption, maintenance, and processing overhead on resource-constrained devices [15]. It addresses limitations of hardware PUFs, such as aging effects and production costs. Experimental results show the scheme achieves 100% authentication accuracy, even under noisy conditions, effectively emulating hardware PUF behavior while ensuring secure and efficient authentication in IoT environments.

A novel research involving 5G secure handover authentication protocol using Spiking Neural Networks and Fuzzy Logic (SNN-FL) to enhance cybersecurity in mobile networks is proposed in [16]. The protocol mitigates attacks such as man-in-the-middle (MITM), replay, desynchronization, and DoS, while ensuring forward and backward key secrecy. It significantly reduces communication overhead, handover latency, and packet loss compared to existing 5G-AKA and 3GPP R16 protocols. Experimental results show the

proposed method achieves high accuracy (98%), precision (0.97), recall (0.97), and F1-score (0.98), making it an efficient and robust solution for secure communication in large-scale, resource-constrained 5G environments.

A machine learning–based authentication framework, PUF-Phenotype, has been proposed in [17], utilizing noisy DRAM-PUF responses for secure device identification without reliance on helper data or traditional error correction. Features have been extracted using a modified VGG16 CNN from visual representations of PUF responses, and classification has been performed using lightweight models such as SVM and RF for both device- and group-level authentication. CRP database storage has been eliminated, and intra-group authentication has been enabled on resource-constrained devices. Achieving over 98% accuracy under varying environmental noise conditions, the system has been demonstrated as a robust and scalable ML-driven alternative for IoT security.

A study involving ML to aid PUF-based authentication is proposed in [18]. This research proposes a lightweight authentication framework for the Internet of Medical Things (IoMT), integrating ML and PUFs to ensure data integrity and privacy. The proposed method eliminates the need for storing CRPs, reducing communication (68 bytes) and computation costs (2.33 ms). A machine learning-controlled PUF generates unpredictable responses, achieving 99.76% accuracy. The framework effectively mitigates impersonation, replay, and man-in-the-middle attacks while preserving anonymity and forward secrecy. Its efficiency and robustness make it suitable for resource-constrained medical devices, outperforming existing approaches in security and performance metrics.

Furthermore, ML and AI have been employed along with PUF for enhancing the security and reliability of the PUF-based security systems. some of the relevant research that addressed various security challenges is listed in Table 1.

Table 1. Comparative table of PUF and ML-based cybersecurity research.

Research Paper	Year	Algorithm	Application	Security Challenges Addressed
Millwood et al. [17]	2023	Deep CNN (VGG16), SVM/RF classifiers	Group-based authentication in IoT	Noise resilience, elimination of helper data, scalability
Sajadi et al. [19]	2023	Delay-based ML-resistant PUF architecture	IoT device authentication	Resistance to ML modeling attacks, lightweight implementation
Zhang et al. [20]	2024	4-layer DNN modeling PUFs with combinational logic	PUF vulnerability analysis	Exposure of security flaws in OR/AND logic-based PUFs
Talukder et al. [21]	2024	Supervised ML (e.g., RF, SVM)	Network intrusion detection in big data environments	Improved detection accuracy, reduced false positives
Mahmood et al. [22]	2024	Supervised and unsupervised ML techniques	Network intrusion detection systems	Optimizing network security
Chen et al. [23]	2023	Transformer-based sequence modeling	Real-time network intrusion detection	Timely detection and handling of sequential data threats
Hernandez et al. [24]	2023	Federated learning with multiple ML models	Distributed intrusion detection across IoT systems	Data privacy, scalability, adaptability
Debicha et al. [25]	2023	Adversarial ML for botnet traffic simulation	IDS robustness evaluation	Exposure of ML-based IDS vulnerabilities, attack mitigation
Kaushik et al. [26]	2025	Statistical feature selection + lightweight ML classifiers	Intrusion detection in constrained IoT systems	Efficient training, improved accuracy, lower computation
Tang et al. [27]	2024	RF profiling integrated with PUF authentication	Remote keyless entry system security	Detection of unauthorized access, mutual authentication
Fortified-Edge 2.0 (Current Work)	2025	ML-based feature extraction and authentication	PUF-based distributed authentication protocol	Privacy protection, secure CRP handling, communication security

4. Security and Privacy in Edge Frameworks

Edge processing has evolved in response to the rapid growth of connected devices that continuously generate large volumes of data. In time-critical applications such as autonomous vehicles, healthcare, and traffic management, where latency is unacceptable, there is a pressing need for faster processing closer to the data source. However, as edge processing moves nearer to the point of data generation, it expands the attack surface, making systems more vulnerable to a range of security threats. The decentralized nature of edge computing complicates the implementation of uniform security measures across all the nodes [28].

The modern era of edge computing adopts a distributed framework, which necessitates effective resource management and secure task offloading to mitigate potential attacks and system failures. Security and privacy becomes critical in edge environments especially when it is a resource-constrained environment. Processing sensitive data at the edge raises concerns about data confidentiality and user privacy. Ensuring secure data handling and storage at the edge is critical to prevent unauthorized access and data breaches [29].

Table 2. Security and privacy challenges and proposed mitigation techniques.

Privacy & Security Challenges	Description	Proposed Mitigation
Expanded Attack Surface [30]	Edge nodes deployed in unsecured environments become easy targets for attackers.	Use of blockchain, encryption, and secure authentication protocols.
Sensor Compromise [31]	Vulnerable medical sensors may be physically tampered or remotely controlled to leak data.	Tamper-resistant hardware, PUF-based key generation, secure boot mechanisms.
Healthcare data security [32]	Secure storage and transmission of medical data	Blockchain integration with cloud, ECDSA, snart contracts
Data Leakage from Edge/Cloud Nodes [33]	Sensitive health data can be exposed from edge caches or cloud storage.	Access control, searchable encryption, and federated data sharing.
Lack of Standardization [34]	Inconsistent protocols and vendors hinder unified security enforcement.	Development of interoperability standards and unified data governance frameworks.
Privacy of User Health Data [35]	Health records must comply with strict data privacy regulations (e.g., HIPAA, GDPR).	Smartcard login, user-controlled access, and privacy-preserving machine learning models.
Malicious Node Injection [36]	Rogue devices may be inserted to manipulate or extract data.	Digital signatures, node authentication, and trust management strategies.

Given the inherent challenges in edge computing, security and privacy must be integrated as core components of the system architecture, as emphasized by several studies summarized in Table 2. In addition to these, persistent issues such as data minimization, infrastructure security, data anonymity, integrity, authorization, authentication, access control, and confidentiality continue to demand effective solutions. The emergence of advanced technologies such as 6G, AI, blockchain, digital twins [37], and edge intelligence has further expanded the research scope, offering new opportunities to design and implement efficient security protocols tailored for edge computing environments.

Security solutions that operate in isolation are not sustainable in the long term, as they tend to increase the attack surface and make the system more vulnerable, ultimately eroding trust in the overall security framework. Moreover, such fragmented approaches can lead to decreased performance efficiency and higher operational costs. A more effective strategy in conjunction with the principles of Security-by-Design (SbD) [38]. This integrated approach ensures end-to-end security throughout the lifecycle of a device or system, enhancing trust, improving performance, and supporting policy compliance from the ground up. Hardware-Assisted Security (HAS) that facilitates SbD by enabling the integration of security mechanisms at the design level through dedicated hardware components has become a foundational principle in modern cybersecurity, especially pertinent to embedded systems, IoT devices, and Cyber-physical systems [39], [40], [41]. This approach also incorporates technologies like trusted execution environments (TEEs), trusted platform modules (TPMs), and hardware security modules (HSMs) to provide security solutions across various applications and computing platforms [42].

Ongoing research in edge computing privacy and security is largely driven by unresolved challenges such as establishing user trust, enabling collaboration among heterogeneous systems, developing low-cost fault-tolerant deployment models, and designing lightweight yet secure authentication and verification mechanisms.

4.1. Hardware Assisted Security - PUF

HAS refers to security mechanisms implemented at the hardware level to protect data, the hardware itself, or the entire system. HAS involves integrating dedicated security

hardware components, modifying hardware designs, or altering system-level architectures to meet specific security requirements. It facilitates the implementation of SbD and Privacy-by-Design (PbD) principles, both of which emphasize proactive security integration during system development [43]. HAS contributes to overall cybersecurity by supporting a range of security functions, as illustrated in Figure 2.

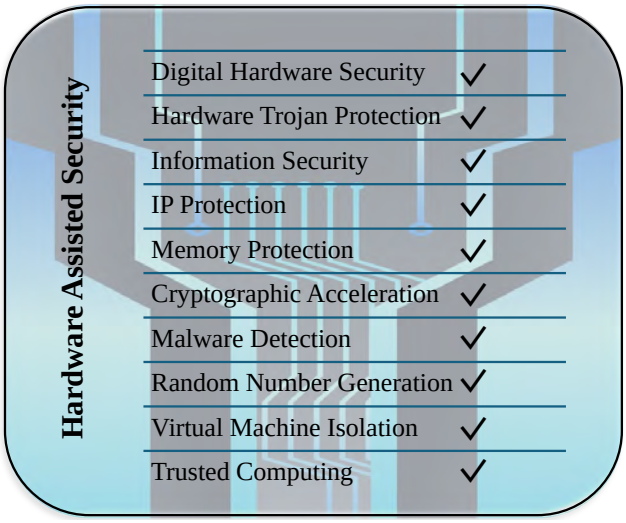

Figure 2. Cybersecurity functions realized through HAS in SbD paradigm.

HAS complements traditional software defenses by embedding protection mechanisms at the hardware level, offering enhanced isolation, observability, resilience, and performance. Despite its advantages, challenges remain in implementation correctness, integration, and long-term adaptability. HAS targets a wide range of threats and vulnerabilities, including software-based attacks such as code flaws, malicious software behavior, and control-flow attacks. It also addresses microarchitectural and hardware-level exploits like side-channel attacks, fault injection, and unpatched hardware flaws. Additionally, HAS mitigates access control violations stemming from improper resource isolation and unauthorized peripheral access. Broader concerns such as hardware immutability, supply chain threats, and policy enforcement issues also fall within the scope of HAS-enabled protections [44].

PUFs in the context of cybersecurity for IoT devices offer a hardware-based solution by generating unique, device-specific responses based on intrinsic manufacturing variations. PUFs eliminate the need to store secret keys, making devices more resistant to physical and invasive attacks such as eavesdropping, side-channel attacks, and unauthorized access resulting from insecure key storage. They play a critical role in secure authentication and lightweight encryption schemes, providing cost-effective security tailored for resource-constrained environments while addressing both privacy and integrity concerns in IoT systems [45].

Utilizing the intrinsic manufacturing variations in a device to generate a unique fingerprint of that hardware offers the advantage of unclonability, this property of the PUFs gives an edge over other hardware-based security methods as the hacker cannot clone the intrinsic properties of the device. Thus, PUFs can provide a low cost hardware-based security to enable device identification and cryptographic key generation.

PUFs are integrated into various systems to bolster security and trust. They find applications in various environments of smart city and smart village infrastructure, providing low-power security solutions in resource-constrained ecosystems. Some of the applications where PUFs are employed are shown in Figure 3.

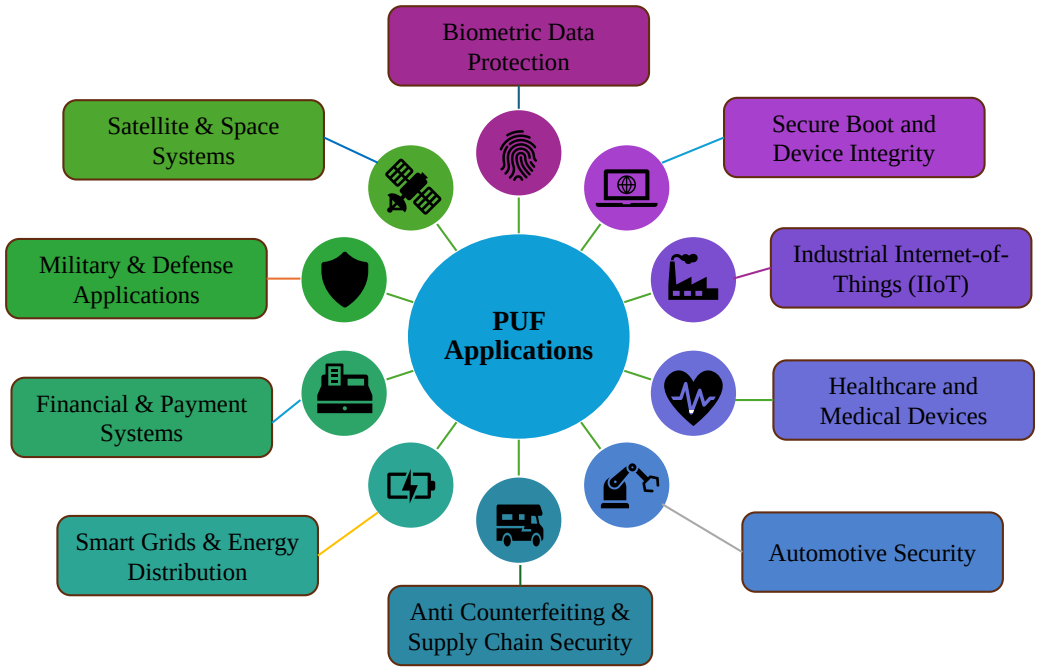

Figure 3. Application areas of PUF-based security solutions.

When employing PUFs for security, several parameters must be carefully considered. Key indicators for assessing the robustness of a well-designed PUF include the amount of helper data required, reliability, security strength, uniqueness, flexibility, and portability.

4.2. PUF Security Challenges

PUF offers promising security solutions for embedded systems and IoT devices, however, their practical deployment faces several challenges. A PUF model needs to satisfy the key metrics that evaluate a PUF as a suitable and strong PUF for security purposes. Metrics like uniqueness, reliability, uniformity, bit aliasing, randomness, and high entropy are the key indicators of a strong PUF [46].

Apart from the PUF features, they face other challenges externally, such as:

- **Environmental variations:** PUFs are vulnerable to external parameters like temperature, voltage, aging, seismic activity, and so on. Ensuring a steady and reliable PUF operation under adverse conditions requires error correction systems without increasing the system complexity.
- **Modelling Attacks:** Advanced ML techniques have been used to predict PUF responses, compromising their security. Certain PUF architectures are vulnerable to ML-based modelling attacks [47],[48], therefore, PUF designs resilient to such attacks need to be considered for security.
- **Limited CRP:** Supposedly, if the PUF responses generated by a device that satisfies all the key metrics making it a strong PUF are finite in numbers, such a device cannot be employed for device authentication that requires frequent authentication.
- **CRP exhaustion:** There is a risk of CRP exhaustion due to repeated use over time, when the PUF device cannot generate unique CRPs, and reusing the existing CRPs is not a viable solution in the context of security.
- **Secure storage:** In collaborative environments where distributed computing in involved, which has PUF as a security primitive, and every local device needs to store a copy of the CRP data, poses a threat to data privacy and security as the distributed environment increases the attack surface, any random vulnerable device can be targeted.

- **Storage space requirements:** Resource-constrained devices will find it challenging to store a large dataset of CRP securely.

Addressing these challenges requires continuous research on robust PUF architectures with improved error detection and correction mechanisms, and standardized evaluation frameworks to ensure their effective and secure deployment in the real-world.

4.3. Machine Learning in Error Detection and Correction

An error occurs when the output information does not match the expected result during execution, or when the output does not accurately reflect the input information during communication. Digital signals often develop errors due to noise during transmission or system malfunctions during execution, leading to bit errors where a binary 0 may flip to a 1, and vice versa. Factors such as noise, cross-talk, temperature variations, voltage fluctuations, and device aging can adversely affect communication networks and hardware performance, introducing errors. Additionally, external attacks that deliberately manipulate devices or data represent another significant cause of errors in information systems.

Traditional error-correcting codes (ECCs) like Hamming, Reed-Solomon, Turbo, and Convolutional codes are employed in correcting errors in communication and storage systems. However, they add redundancy, parity bits, and algebraic techniques to detect and correct errors. These traditional methods face limitations as the data complexity and system noise increase, lowering their adaptability and efficiency [49].

ML has become an indispensable tool in error detection and correction in the current era across various domains, enhancing the system reliability and performance more effectively compared to traditional methods. Traditional methods, which work on predefined set of rules fail to adapt well to complex and evolving data patterns. ML algorithms, however, has the capability to learn from data, identify intricate patters, and adapt to new type of errors, making them highly suitable and effective in dynamic environments. ML has been increasingly employed for detecting and correcting code errors across various programming languages. Algorithms such as K-Means, Graph Neural Networks (GNNs), K-Nearest Neighbors (KNN), Linear Regression, Naive Bayes, Random Forest, and Support Vector Machine (SVM) classifiers have been studied for their effectiveness in identifying and correcting code errors. Some models focus specifically on syntax error detection, while others offer theoretical frameworks for broader error correction strategies [50]. However, a more generalized model for automatically detecting and correcting errors irrespective of the programming language is still an open research question.

ML-based error detection and corrections have been increasingly employed across various domains that involve data processing and communication. KNN-regression has been researched for error detection and correction in aircraft sensors, which are crucial for the mechanical system. The solution is a data-driven approach where the ML model uses autocorrelation online and compares with the offline library to accurately detect failures, if any [51].

ML methods significantly outperform traditional approaches in handling the high variability and noise associated with Power Line Communication (PLC) channels. ML models such as SVM, Deep Learning (DL), Random Forest (RF), Decision Trees (DT), and KNN are trained on PLC datasets to predict errors, optimize network parameters, and reduce bit error rates (BER). In this context, ML-based techniques demonstrate superior performance compared to conventional error correction methods, such as Hamming codes, Reed-Solomon codes, and Turbo codes [52]. Study shows that SVMs are effective in classifying data in noisy environments for optimal code selection, DL improves feature

extraction for better error correction, and RF identifies error patterns and corrects them with low computational overhead.

ML is employed to address the reliability challenges of Digital Computing-in-Memory (DCIM) architectures, focusing on the effects of transient faults and circuit aging that can lead to single-event failures or performance degradation. To mitigate transient faults in NOR gates, the study utilizes model prediction techniques such as the Probabilistic Transfer Matrix (PTM) and Signal Probability Reliability (SPR) methods to analyze fault susceptibility. These ML-based predictive modeling approaches are critical for accurate circuit reliability estimation and for designing robust DCIM systems [53]. Quantum systems represent an emerging research field where ML techniques are increasingly employed in Quantum Error Correction (QEC). Methods such as supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, and deep learning (DL) are used to improve the detection, classification, and correction of errors in quantum systems. The integration of ML significantly reduces computational latency, enhances the robustness of QEC protocols, and offers scalability for future large-scale quantum computing [54].

Furthermore, ML-based error detection and correction methods are applied across various domains. In real-time weather forecasting, ML models correct prediction errors by learning from discrepancies between model forecasts and observational data, thereby enhancing forecast accuracy. Deep learning (DL) techniques have been used to detect errors in medical datasets, such as mislabeled laboratory samples, helping to maintain data integrity, which is vital for accurate diagnoses and treatments. Additionally, chatbots and virtual assistants employ ML to detect and correct errors in user interactions, improving communication accuracy. By learning from past interactions, ML models enhance response relevance and reduce mistakes over time.

5. The Proposed Fortified-Edge 2.0 Framework

To enhance the strength of the authentication system, this research proposes a novel authentication protocol that prioritizes data security. In the current scenario, where critical PUF CRP data is transmitted over a network, ensuring data integrity is essential to prevent compromise by external attacks. Cryptographic methods encrypt plaintext into ciphertext for secure communication. However, this research takes security further by uniquely processing the plaintext before transmission.

Reliability is a crucial aspect of any security system utilizing PUFs. One of the key reliability challenges is the presence of bit errors in PUF responses, which can arise due to environmental variations. Therefore, integrating an effective bit error correction mechanism is essential for a robust PUF-based security system.

In addition to enhancing security at the edge, the proposed framework aligns with broader sustainability objectives by emphasizing computational efficiency and energy awareness in IoT systems [55]. By leveraging Physically Unclonable Functions (PUFs) and lightweight cryptographic primitives such as ECDSA, the authentication process significantly reduces the need for intensive computation and large key storage, which are common drawbacks of traditional security protocols. The use of k-mer-based feature extraction further contributes to a low-overhead implementation by enabling efficient data representation and minimizing processing time. These design choices collectively lead to lower power consumption and reduced thermal load, which are critical for prolonging device lifespans and supporting scalable, energy-efficient deployments in smart city and smart village infrastructures. Therefore, the framework not only addresses the need for robust security in heterogeneous edge environments but also supports the long-term goal of sustainable IoT system design.

An ML-based PUF bit error detection and correction method is proposed in this research [56]. This approach leverages an efficient K-mer sequencing method, which processes binary n-bit PUF responses as binary sequences and extracts relevant features. The ML model is trained on these features, enabling it to detect errors in new data. This method demonstrates high efficiency, achieving up to 99% accuracy. A key advantage of using ML is that it does not introduce additional area or computational overhead, unlike existing methods such as fuzzy extractors or error correction codes.

The ML-based error correction process is illustrated in Figure 4. The error detection and correction process is shown in Algorithm 1 . The n-bit binary PUF responses are transformed into binary sequences, referred to as words. A K-mer of 6 is applied to extract features, which are then vectorized using CountVectorizer(), into unique integer vectors. These vectors serve as references for bit error detection. K-mers algorithms are largely used for analysis of genomic datasets, where K-mers represent a contiguous nucleotide or amino acid sequence of fixed length K. K-mers are used in bioinformatics for sequence alignment, sequence clustering, error correction of sequencing reads, pattern recognition, and so on [57].

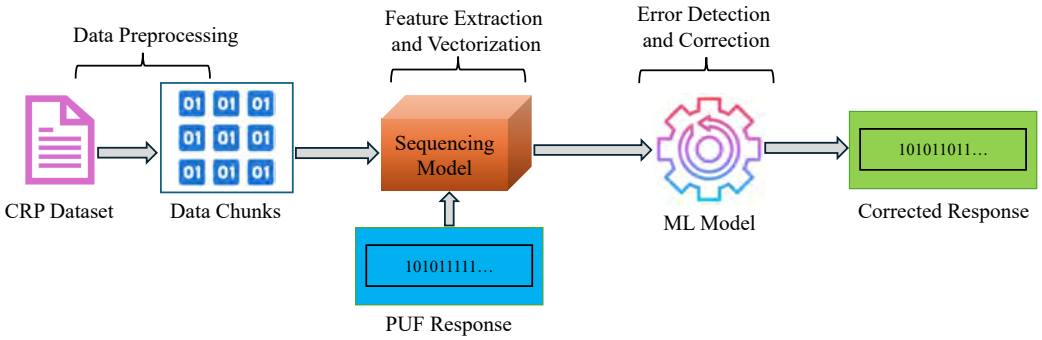

Figure 4. Error detection and correction model.

Algorithm 1 Bit-Flip Error Classification Using K-Mer and Naive Bayes

Require: Dataset $\mathcal{D}$ of N binary sequences, K-mer size k
Ensure: Trained classifier and evaluation metrics

Step 1: Load and Preprocess Data
  Read each sequence from $\mathcal{D}$ into list $S = \{s_1, s_2, \ldots, s_N\}$
  Initialize DataFrame DF with column `sequence` $\leftarrow S$
Step 2: Assign Class Labels
  For each index $i \in [0, N-1]$, compute class label $c_i = \lfloor i/100 \rfloor + 1$
  Add column $DF[\texttt{class}] \leftarrow c_i$
Step 3: Extract K-Mers and Vectorize
  For each sequence s_i in $DF[\texttt{sequence}]$:
    Generate overlapping K-mers $K_i = \{s_i[j : j+k] \mid 0 \le j \le |s_i| - k\}$
    Concatenate K_i into space-separated string w_i
    Set $DF[\texttt{words}_i] \leftarrow w_i$
  Drop original column $DF[\texttt{sequence}]$
Step 4: Feature Transformation
  Apply CountVectorizer with n-gram size 4 on $DF[\texttt{words}]$ to obtain matrix X
Step 5: Split Dataset
  Let $y \leftarrow DF[\texttt{class}]$
  Split X, y into training and testing sets:
    $X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}}$
Step 6: Train and Evaluate Classifier
  Train Multinomial Naive Bayes model on $(X_{\text{train}}, y_{\text{train}})$
  Evaluate performance on X_{test} using accuracy and other metrics

The error correction model uniquely transforms binary responses into integer vectors. By leveraging this transformed data, without directly revealing the original n-bit response—a novel authentication protocol is implemented using feature vectors. The feature-based authentication system, which integrates machine learning and secure authentication protocols, is illustrated in Figure 5.

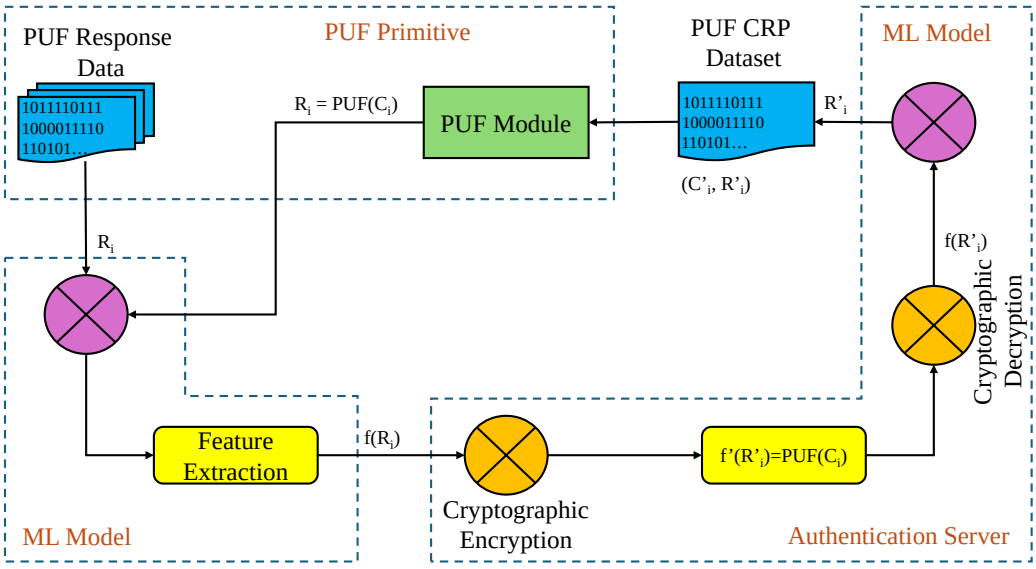

Figure 5. The proposed process flow of feature-based authentication.

The proposed framework illustrates the preprocessing of PUF CRP data and feature extraction using an ML model. Additionally, the model is implemented in a distributed framework, making it suitable for PUF-based security systems deployed in collaborative computing ecosystems at the edge [58]. First, the CRP dataset is used to train the ML model, which is then deployed across multiple edge servers. The distributed framework is implemented using Federated Learning, where each local model is trained on its respective PUF data. Features are extracted, vectorized, and the local model parameters are sent to a global server, which aggregates them using FedAvg. The updated model is then distributed back to the local models at the edge. This distributed error detection and correction system is utilized to authenticate edge devices. In the current scenario, the application considered is Edge Data Centers (EDCs) participating in load balancing at the edge. These EDCs must be authenticated in real time before task offloading.

The proposed authentication protocol operates in multiple stages. In the first stage, Client_1 (EDC-1) initiates authentication with Client_2 (EDC-2) to enable task sharing. The selected challenge is vectorized, encrypted using private keys, and transmitted over the network to Client_2.

Upon receiving the encrypted challenge, Client_2 decrypts it and reconstructs the original n-bit challenge from its vectors using the local Federated Learning (FL) model. The reconstructed challenge is then applied to the PUF module, which generates the corresponding n-bit response. The local ML model at Client_2 vectorizes the response, encrypts the vectors, and transmits them back to Client_1.

Client_1 decrypts the received vectors using its public key and reconstructs the original n-bit response. The extracted response is then verified against the expected value. If the verification is successful, Client_2 is authenticated, and both clients proceed with task sharing. The proposed method eliminates the vulnerability of transmitting crucial PUF data in its original form, improves the data security, and maintains the integrity of the system.

A generalized representation of the combined process flow of the PUF module along

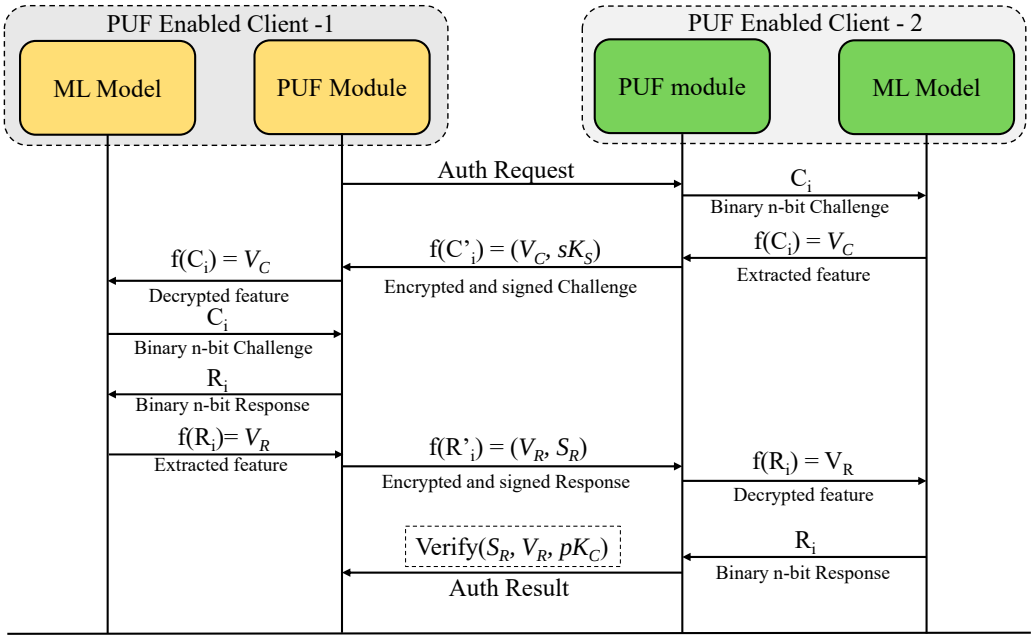

Figure 6. Feature-based authentication between PUF-enabled devices.

Client_1 initiates an authentication request. Client_2's ML model extracts the feature vector of a challenge in response to an authentication request initiated by Client_1. It then encrypts and digitally signs the resulting vectorized challenge before transmitting it to Client_1. Upon receiving the message, Client_1 decrypts it, verifies the signature, and forwards the challenge vector to its local ML model, which reconstructs the original n-bit binary challenge. This reconstructed challenge is applied to the PUF module, producing an n-bit binary response. The ML model in Client_1 then extracts the feature vector from this response. The vectorized response is subsequently encrypted, signed, and sent back to Client_2. Upon receiving the message, Client_2 decrypts and verifies it, and sends the response vector to its ML model, which reconstructs the binary response. This response is then compared with the expected response from the stored CRP dataset. If the response is verified successfully, an authentication result is sent to Client_1.

6. Experimental Results and Analysis

This research utilizes a 64-bit Arbiter PUF architecture implemented on a Xilinx BASYS3 FPGA board. A dataset comprising 100,000 entries is generated using 1,000 unique challenges applied across multiple PUF instances. The machine learning model based on k-mer feature extraction is trained and evaluated on a Raspberry Pi 4 to assess performance in a resource-constrained environment. ECDSA is employed for secure communication due to its efficiency, compact signature size, and strong security guarantees. Despite having a smaller key size, ECDSA offers security comparable to that of RSA with significantly larger keys. This reduction in key length minimizes computational overhead, making ECDSA especially well-suited for edge devices. Furthermore, the compact signatures generated by ECDSA help reduce bandwidth consumption, making it an ideal choice for lightweight, low-latency, and secure authentication frameworks deployed in distributed systems. The algorithm is implemented in a client-server model, where the server can be interchanged with any participating client, which acts as a verifier during mutual authentication in a distributed computing environment.

6.1. Experimental Setup

The ECDSA-based secure authentication protocol uses asymmetric keys (Public/Private), which enhances the integrity of the cryptographic system where both client and server have their unique keys and only public keys are shared. The security level and speed of operation of ECDSA algorithm is very high,the computational cost is moderate. Using public/private key pairs eliminates the need for sharing secret keys, and the algorithm is resistant to man-in-the-middle attacks.

The implementation of the ECDSA authentication protocol using the feature vectors of the CRP data comprises three steps.

1. Feature extraction: N-K-mers are used to extract the features of both challenges and responses. The K-mers are converted to vectors using the CountVectorizer function.
2. Server: The server selects a random challenge at a given index and extracts the vectors, encrypts the challenge vector using its private key, and sends the package to the client.
3. Client: The client will decrypt the package using the server's public key and convert the vectors to their corresponding binary challenge. The client will also convert the generated response to its feature form using k-mers, convert it to vectors and encrypt the vectorized response, sign it with its private key, and send it to the server for verification.

The steps used to extract the characteristics of the challenges and the response are shown in Algorithm 2. This process transforms a binary sequence into a numerical feature vector suitable for secure authentication. It begins by taking a binary sequence B of length m and a k-mer size k, initializing an empty list K to store extracted k-mers. By sliding a window of size k across the sequence, it extracts overlapping substrings K_i from positions $i = 0$ to $i = m - k$, appending each to the list K. After collecting all k-mers, the list is converted into a text string representation. A CountVectorizer is then applied, treating each k-mer as a word and counting the frequency of each unique pattern to generate a fixed-length numerical feature vector V.

Algorithm 2 Feature Extraction Using K-Mers

Require: Binary Sequence B of length m, K-mer size k
Ensure: Feature Vector V
Step 1: Initialize
    Define an empty list $K \leftarrow \{\}$ to store k-mers.
Step 2: Extract K-Mers
for $i \leftarrow 0$ to $m - k$ **do**
    Extract substring $K_i = B[i : i + k]$
    Append K_i to list K
end for
Step 3: Vectorization
    Convert K into a text string representation.
    Apply CountVectorizer to transform K into vector V.
Step 4: Return Feature Vector
    Output V as the final feature representation.

Algorithm 3 shows the steps involved in the evaluation on the server side. The process begins by loading the CRP dataset $\mathcal{CRP}$ from a local file. A challenge C is selected from the dataset, and feature extraction techniques are applied to generate a vectorized representation V_C. To ensure data authenticity and integrity, the server computes a digital signature $S_C = \text{Sign}(V_C, sk_S)$ using its private key sk_S, and transmits the signed challenge tuple (V_C, S_C) to the client. Upon receiving the client's signed response (V_R, S_R), the server performs signature verification by validating S_R against V_R using the client's public key pk_C. If the verification succeeds and the received vectorized response V_R matches the

expected response V_E stored in the dataset, the server ends the authentication process with a success message. Otherwise, an authentication failure message is generated. To facilitate performance evaluation, the server records critical metrics, including execution time, CPU utilization, and memory usage, before terminating the session.

Algorithm 3 Server Authentication Process

Require: Challenge-Response Dataset $\mathcal{CRP}$
Ensure: Authentication Result
Step 1: Initialize
    Load dataset $\mathcal{CRP}$ from CSV.
Step 2: Challenge Selection
    Select a challenge $C \in \mathcal{CRP}$.
    Generate vectorized challenge V_C using feature extraction.
Step 3: Challenge Signing
    Compute digital signature $S_C = \text{Sign}(V_C, sk_S)$.
    Send tuple (V_C, S_C) to client.
Step 4: Receive and Verify Response
    Receive signed response (V_R, S_R) from client.
    Verify signature: $\text{Verify}(S_R, V_R, pk_C)$.
if Signature S_R is valid **then**
    if $V_R = V_E$ (expected response) **then**
        Send authentication success message.
    else
        Send authentication failure message.
    end if
else
    Send authentication failure message (invalid signature).
end if
Step 5: Log and Close
    Log execution time, CPU, and memory usage.
    Close connection.

Algorithm 4 shows the steps involved in the evaluation on the client side. The client authentication process begins with the client establishing a connection to the server and sending an authentication request. Upon receiving a tuple consisting of the encrypted vectorized challenge V_C^{enc} and the corresponding server-generated signature S_C, the client decrypts V_C^{enc} to recover the original vectorized challenge V_C. The authenticity of the challenge is then verified by validating the server's signature S_C against V_C using the server's public key pk_S. If the signature verification succeeds, the client reconstructs the original binary challenge C from V_C, applies it to its embedded PUF to generate the binary response R, and vectorizes the response to produce V_R. Subsequently, the client signs V_R using its private key sk_C to produce a signature S_R. The client then transmits the signed tuple (V_R, S_R) back to the server. If the server's signature verification fails, the client aborts the authentication session to prevent further communication with a potentially compromised server. Finally, the client logs critical performance metrics, including execution time, CPU usage, and memory consumption, before closing the connection.

The secure authentication process using ECDSA is shown in Algorithm 5. The ECDSA Key Management and Signing process begins by attempting to load existing key pairs for both the server (sk_S, pk_S) and the client (sk_C, pk_C) from secure file storage. If the keys are missing or found to be invalid, new ECDSA key pairs are generated for both entities using a specified elliptic curve $\mathcal{C}$, and the resulting keys are securely stored in PEM format for future sessions. Once key management is completed, the signing process involves taking a message M and a corresponding private key sk, computing its digital signature σ using the ECDSA algorithm, and subsequently encoding the signature into a base64 format σ_b for

Algorithm 4 Client Authentication Process

Require: Server Public Key pk_S, Client Private Key sk_C
Ensure: Authentication Result
Step 1: Initiate Authentication Request
    Establish a connection with the server.
    Send an authentication request.
Step 2: Receive and Verify Challenge
    Receive tuple (V_C^{enc}, S_C) from the server.
    Decrypt V_C^{enc} to obtain V_C.
    Verify signature Verify(S_C, V_C, pk_S).
if Signature is valid **then**
    Reconstruct binary challenge C from V_C.
    Apply C to the PUF to generate response R.
    Vectorize R to obtain V_R.
    Sign V_R using client private key to obtain S_R.
    Send tuple (V_R, S_R) to the server.
    Receive authentication result.
else
    Abort connection (invalid server signature).
end if
Step 3: Log and Close
    Log execution time, CPU, and memory usage.
    Close connection.

transmission or storage. For signature verification, the received base64-encoded signature σ_b is first decoded to recover the original signature σ, and the verification is performed against the message M using the public key pk. The verification process outputs whether the signature is valid or invalid, ensuring the integrity and authenticity of transmitted messages within the authentication framework. The mathematical symbols used in the process algorithms are shown in Table 3.

Table 3. Mathematical Symbols Used in ECDSA Key Management

Symbol	Meaning
$\mathcal{C}$	ECDSA Curve (e.g., NIST256p)
sk_S, pk_S	Server's Private/Public Key
sk_C, pk_C	Client's Private/Public Key
M	Message to be signed
σ	ECDSA Digital Signature
σ_b	Base64 Encoded Signature
Sign(M, sk)	Signing Function
Verify(σ, M, pk)	Signature Verification Function
Base64Encode(σ)	Base64 Encoding
Base64Decode(σ_b)	Base64 Decoding

6.2. Analysis of Results

The binary CRP data is converted to its vector form using ML algorithms. The vectorized data is analyzed and visualized first to study the distribution of the feature values across the challenge features and response features. Figure 7 shows the distribution of feature values, showing how challenge features correlate with response features and vectorization.

Algorithm 5 ECDSA Key Management and Signing

Require: ECDSA Curve $\mathcal{C}$, File Paths for Key Storage
Ensure: Persistent Key Pairs for Server and Client
Step 1: Load or Generate Keys
    Server: Attempt to load (sk_S, pk_S) from files.
    Client: Attempt to load (sk_C, pk_C) from files.
if Keys are missing or invalid **then**
    Generate new ECDSA key pair (sk_S, pk_S) for server using curve $\mathcal{C}$.
    Store (sk_S, pk_S) in PEM format.
    Generate new ECDSA key pair (sk_C, pk_C) for client using curve $\mathcal{C}$.
    Store (sk_C, pk_C) in PEM format.
end if
Step 2: Signing a Message
    Given a message M and private key sk, compute:
    $\sigma = \text{Sign}(M, sk)$ using ECDSA.
    Encode signature: $\sigma_b = \text{Base64Encode}(\sigma)$.
    Return signature σ_b.
Step 3: Verifying a Signature
    Given message M, signature σ_b, and public key pk:
    Decode: $\sigma = \text{Base64Decode}(\sigma_b)$.
    Verify using ECDSA: $\text{Verify}(\sigma, M, pk)$.
    Return verification result (valid or invalid).

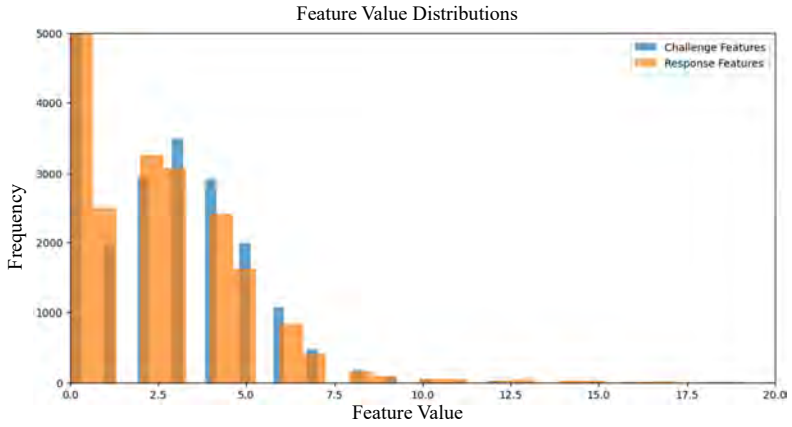

Figure 7. Feature value distribution of challenges and responses.

The correlation is estimated using the Pearson Correlation Coefficient (PCC), which measures the linear correlation between each challenge feature and the corresponding response feature. PCC is the ratio between the covariance of two variables and the product of their standard deviations, the result is always between -1 and 1. Mathematically, it is represented by equation 1.

$$r = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} \tag{1}$$

where:

- $\text{cov}(X, Y)$ is the covariance between variables X and Y,
- σ_X and σ_Y are the standard deviations of X and Y, respectively.

From the evaluations, the correlation values obtained are: Average Correlation: 0.2721, Maximum Correlation: 1.0000, and Minimum Correlation: -0.0964. A correlation average of 0.27 is a low to moderate value, this means the challenge feature and response features are somewhat related but largely independent. This is good for PUF-based security, as a high average correlation would mean that attackers could predict the response from

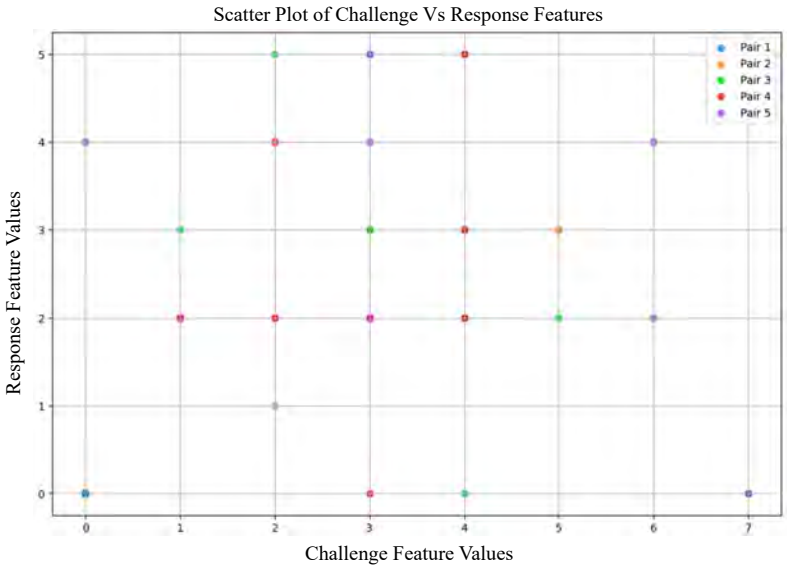

Figure 8. Scatter plot of extracted response features.

the challenge easily. When selecting the challenges for authentication, those with high correlation can be omitted.

A scatter plot of the discrete feature values is shown in Figure 8, which shows that the challenge and response vectors are mostly smaller integer values because of count vectorization frequencies. Many challenge and response pairs seem to occupy a small set of possible values, and a low variation across features aligns with the Pearson correlation average of 0.27 found earlier. For some pairs, as the challenge feature values increase, response feature values also seem to increase, although not strongly linearly.

The evaluation of the entropy distribution of the response features is shown in Figure 9. It is seen that most of the responses generated appear in the high entropy region between 1.4 and 1.6, which implies that the feature values are well spread and less predictable. However, a small number of features have lower entropy in the range of 0.6-1.2, and these responses can be omitted from use in the authentication process. The entropy plot also shows that it varies across features and not every feature is equally random, which is quite expected in real-world PUF response data.

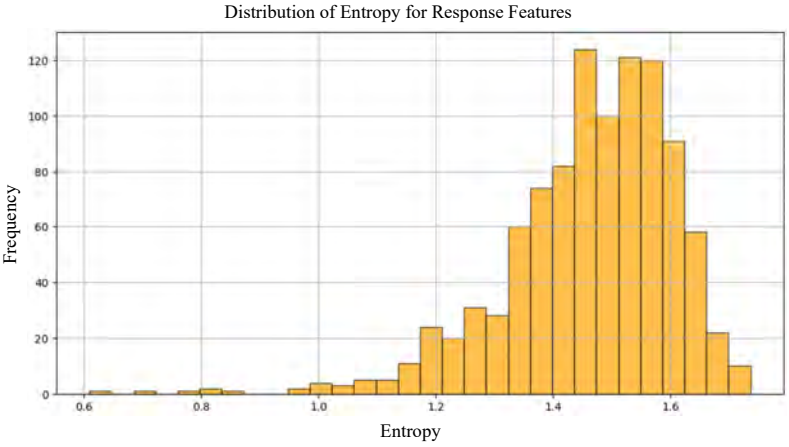

Figure 9. Distribution of entropy for response features.

The challenge vectors generated are visualized to study the distribution of features by analyzing selected challenge vectors as shown in Figure 10, which displays five randomly selected challenge vectors, each line represents a vectorized challenge, showing feature

values across indices. High variability is seen in the vectors, suggesting the diversity of challenge representation.

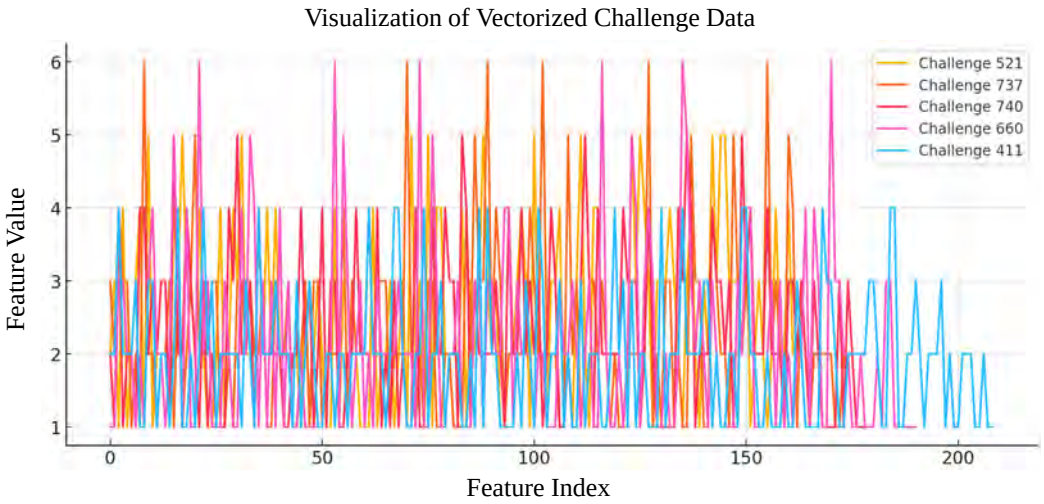

Figure 10. Distribution of challenge vectors.

Figure 11 shows five randomly selected response vectors; the difference in representation from challenge vectors is because of the PUF mechanism. It is seen that some response vectors have higher peak values, which reflect a stronger feature significance. This study will help select a pair of challenge responses with stronger feature representations for authentication purposes.

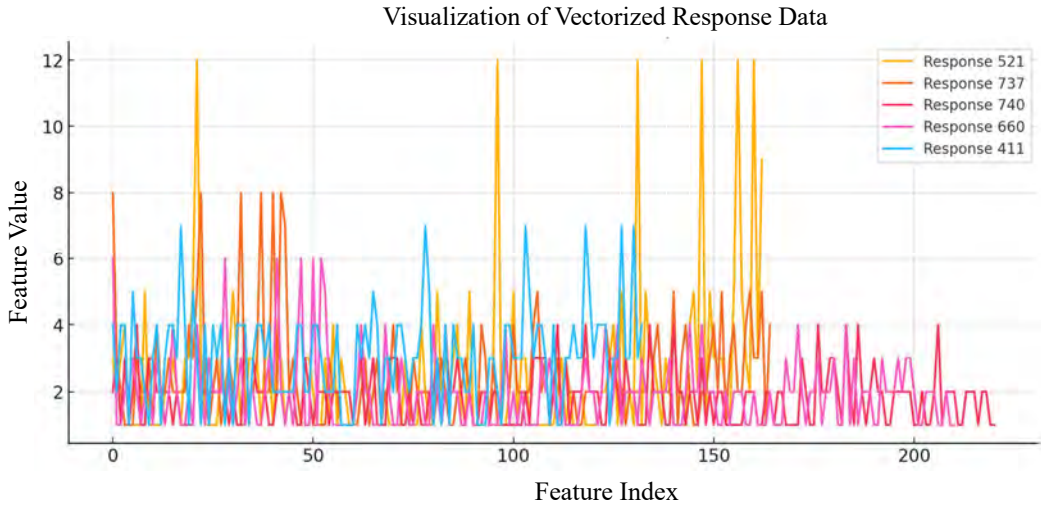

Figure 11. Distribution of response vectors.

The results of the code execution are illustrated in Figure 12 and Figure 13. In the server-side evaluation, the server waits for a client request after opening the connection. the server then selects a binary challenge from the dataset and converts it into its corresponding vectorized representation. This vectorized challenge is securely transmitted to the client. After receiving the client's vectorized response, the server reconstructs the original binary response sequence. It performs authentication by verifying whether the reconstructed response matches the expected response associated with the challenge. If there is a match, the server sends the successful authentication message to the client.

Similarly, on the client side, the client sends an authentication request to the server, and receives a vectorized challenge in the encrypted form, client will decrypt the payload, extract the vectorized challenge and reconstruct the original binary sequence, the binary

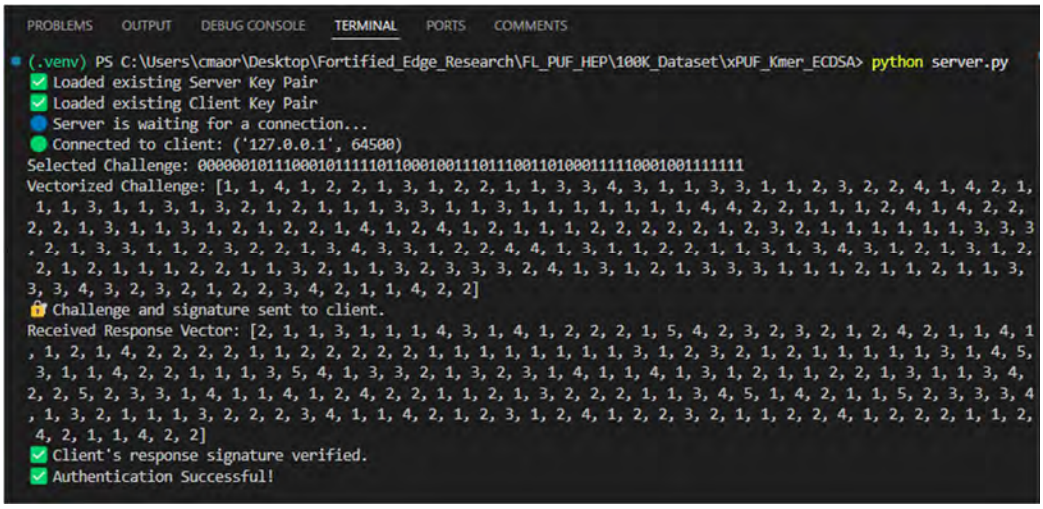

Figure 12. Server-side authentication results.

challenge is then applied to the PUF device and binary response is obtained. The binary response is vectorized, encrypted, and sent to the server for verification. The client will wait for the authentication results. If the authentication fails, the connection between the client and server is closed.

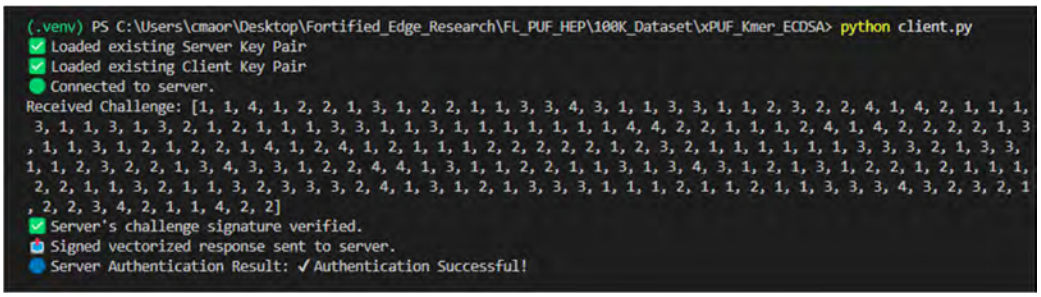

Figure 13. Client-side authentication results.

During each round of server-client authentication, the performance metrics are recorded. For real-time applications in resource-constrained environments, it is important to prove that there is lower computation time and lower bandwidth consumption. The performance metrics of the current protocol are shown in Table 4.

The bandwidth of the authentication process is computed as follows: Total data transferred is the sum of challenge data size in bytes and response data size in bytes, that is 662 bytes and 626 bytes, converted to bits by multiplying by 8. The total execution time is the total time taken by the server to authenticate the client, which is 0.028031 seconds.

$$\text{Bandwidth} = \frac{\text{Total Data Transferred} \times 8}{\text{Total Execution Time}} \tag{2}$$

$$\text{Bandwidth} = \frac{1288 \times 8}{0.028031} \approx 367687.69 \text{ bps} \quad (\approx 368 \text{ Kbps}) \tag{3}$$

The total bandwidth of the authentication process is estimated as close to 368 Kbps.

The total authentication time taken is an average of 0.02 seconds, and the bandwidth consumed for the feature-based authentication protocol is 368 kbps, as shown from the evaluation using equations 2 and 3.

Table 4. Performance metrics of server and client.

Metric	Server Side	Client Side
Total Execution Time (ms)	28.031	28.031
Challenge Signing Time (ms)	1.067	-
Challenge Receiving Time (ms)	-	3.133
Response Receiving Time (ms)	3.952	-
Response Signing Time (ms)	-	1.002
Response Sending Time (ms)	-	0.000
Signature Verification Time (ms)	2.003	1.948
CPU Usage (%)	32.40	36.90
Original Challenge Size (bytes)	105	-
Vectorized Challenge Size (bytes)	662	-
Original Response Size (bytes)	-	105
Vectorized Response Size (bytes)	626	-

6.3. Security Analysis

The proposed feature-based authentication framework addresses critical security challenges inherent to integrated systems employing PUF. By leveraging k-mer-based feature extraction and ECDSA for secure message authentication, the current implementation is designed to be resilient against a range of prominent theoretical threats, as outlined below:

- **Replay attacks:** The proposed framework resists replay attacks by incorporating fresh PUF challenges for each authentication. Since challenges are randomly selected and signed by the server, reusing previous response vectors will fail signature verification or mismatch the expected response, thereby preventing unauthorized reuse.
- **Modeling Attacks:** A machine learning attack aims to build predictive models of a PUF's behavior using observed CRP data. The proposed system prevents this by never transmitting the raw CRPs. Instead, it transmits vectorized abstractions of the challenge and response, which are obfuscated and lack direct correlation to the binary input-output, significantly reducing the feasibility of modeling attacks.
- **Man-in-the-Middle (MitM) Attacks:**All the transmitted data, including the vectorized challenge and response, is digitally signed using ECDSA. Any tampering or substitution by an adversary in transit will result in signature verification failure at the receiver, thereby preventing MitM attacks.
- **Side-Channel Attacks:**: By transmitting only the abstracted feature vectors rather than raw binary data, the protocol minimizes the information exposed over the channel. This design inherently reduces side-channel leakage that could otherwise be exploited to infer the internal PUF behavior or secret keys.
- **Cloning Attacks:**The use of PUFs ensures that device identities are physically bound and unclonable due to inherent hardware variations. Even if the attacker observes multiple authentication sessions, reproducing the exact behavior of the device is not possible without access to the original hardware.

6.4. Comparitive Perspective

The superiority of the current research is demonstrated through significant enhancements over the preliminary Fortified-Edge 2.0 framework [59]. The current work addresses critical security challenges by introducing a secure, low-latency, and low-bandwidth authentication protocol tailored for resource-constrained edge environments. A comparative

analysis of the Fortified-Edge research with state-of-the-art research from Table 1 is presented in Table 5.

The preliminary work presents a machine learning-based authentication monitoring system for the EDC, which continuously monitors the authentication process using parameters such as location, EDC ID, site ID, authentication time, and others. Threats are detected by identifying anomalies in these parameters and flagging them as potentially malicious requests. While the preliminary work ensures security by monitoring incoming authentication requests and processes, it does not address broader security challenges such as data confidentiality and privacy during communication. It does not meet the requirements for a low-latency, low-bandwidth authentication protocol essential for resource-constrained edge environments.

In contrast, the current work effectively addresses these challenges by introducing a secure, lightweight authentication protocol based on machine learning-driven PUF feature extraction. This approach ensures that critical PUF CRP data remains protected during transmission, even across untrusted communication networks. The results from state-of-the-art literature show the efficient use of ML for security solutions in various applications. However, using a feature-based authentication process as proposed by the Fortified-Edge 2.0 research is novel.

7. Conclusions and Future Research

Security solutions leveraging embedded hardware provide strong protection without adding complexity to the system architecture. HAS is an emerging area that enhances privacy and security by incorporating SbD principles, embedding security as an integral and continuous process throughout the life cycle of a device or system. PUFs represent a secure, lightweight, and robust solution within the HAS paradigm, particularly well-suited for optimization in resource-constrained environments such as collaborative edge computing in smart village infrastructures. ML techniques further strengthen the security, integrity, and reliability of PUF-based systems. This research focuses on improving the reliability and security of device identification and data communication by proposing a novel feature-based authentication protocol. By transmitting sensitive CRP data in a vectorized form, the protocol ensures that critical information remains concealed from potential adversaries within the network. The k-mer-based feature extraction model efficiently derives distinctive features from binary challenge and response sequences and supports accurate reconstruction of the original sequences from the extracted vectors. This method enhances the confidentiality and integrity of transmitted data. Statistical analysis of the extracted feature vectors demonstrates strong correlation and high entropy levels, characteristics essential for reliable PUF operation in real-world applications.

The use of PUF eliminates the need for storing and retrieving cryptographic keys significantly reducing the computational complexity and associated power-overhead. Additionally, the ECDSA algorithm is selected over RSA for shorter and secure keys, contributing to reduced computational cost and energy usage. The k-mer based feature extraction further reduces the dimensionality of the data, enabling faster computation.

Performance evaluation shows that the proposed protocol is computationally efficient, consuming minimal CPU resources and achieving execution times of approximately 28ms. Additionally, bandwidth consumption is measured at around 368 Kbps, highlighting the protocol's suitability for deployment in resource-constrained environments. Future work includes evaluating the protocol's resilience against various external attack models, validating its performance across different hardware platforms, and enhancing the design to offer robustness against quantum adversaries.

Table 5. Comparison of results with state-of-the-art literature.

Research	ML Algorithm	Application	Cryptography Algorithm	Auth. Time (s)	Bandwidth (Kbps)	**Accuracy (%)**	Energy efficiency & Sustainability	Scalability
[17]	LR and RF classifiers, CNN for feature extraction	PUF behaviour authentication	NA	NA	NA	98.4	No real-time inference, high energy efficiency	Group-based DRAM-PUF limits scalability
[21]	DT, RF and Extra Tree	Network intrusion detection	NA	NA	NA	99.99	Lacks energy profiling, moderately sustainable	High, tested on various benchmark datasets
[22]	RF, DT, KNN, MLP, ANN, CNN	Network intrusion detection	Multi-factor authentication (MFA)	NA	NA	99.97	Moderate energy efficieny, designed to reduce long-term environmental impact	Supports distributed ML and scalable IDS
[23]	Decision Transformer	Network intrusion detection	NA	NA	NA	99.33	High energy efficieny, high sustainability	DTs scale well with data
[27]	CNN	Remote keyless entry	XOR-based secret key encryption	2.05	NA	99.8	Moderate energy efficiency, Higher hardware dependency reduces sustainability	Limited to vehicle specific setup
[59]	SVM	Authentication Monitoring	PUF based Authentication	NA	NA	99	Design optimized for low resource usage	Designed for collaborative environment
Fortified-Edge 2.0 (Current work)	**K-mer**	**PUF CRP Feature Extraction**	**ECDSA**	**0.028031**	**368**	**99.74**	Lightweight ECDSA and ML model for high energy efficieny and high sustainability	Feature-based encoding supports distributed ecosystem

Compliance with Ethical Standards

The authors declare that they have no conflict of interest, and there was no human or animal testing or participation involved in this research. All data were obtained from public domain sources.

Acknowledgments

This article is an extended version of our preliminary conference paper presented at [59] and [60].

References

1. Satyanarayanan, M. The Emergence of Edge Computing. *Computer* **2017**, *50*, 30–39.
2. Deepak.; Upadhyay, M.K.; Alam, M. Edge Computing: Architecture, Application, Opportunities, and Challenges. In Proceedings of the In Proc. 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS), 2023, pp. 695–702.
3. Cao, J.; Zhang, Q.; Shi, W., Challenges and Opportunities in Edge Computing. In *Edge Computing: A Primer*; Springer International Publishing: Cham, 2018; pp. 59–70.
4. Tirupatamma, N.; Anjali, N.; Keerthi, V.P.M.; Atchi, H.S.; Sammy, F. An Exploration of Edge Computing: Emergence, Evolution, Challenges, Approaches. In Proceedings of the In Proc. 8th International Conference on Inventive Systems and Control (ICISC), 2024, pp. 19–26.
5. Kong, X.; Wu, Y.; Wang, H.; Xia, F. Edge Computing for Internet of Everything: A Survey. *IEEE Internet of Things Journal* **2022**, *9*, 23472–23485.
6. Reddy, K.S.; Balaji, M.; Baba, A.; Yuvaraj, G.; Ramesh, D. An Enhancing Load Balancing and Security in Edge Computing: Comprehensive Review Analysis. In *Proceedings*; 2024; pp. 721–726.
7. Aarella, S.G.; Mohanty, S.P.; Kougianos, E.; Puthal, D. PUF-based Authentication Scheme for Edge Data Centers in Collaborative Edge Computing. In Proceedings of the In Proc. IEEE International Symposium on Smart Electronic Systems (iSES), 2022, pp. 433–438.
8. Danger, J.L.; Guilley, S.; Mukhopadhyay, D.; Rührmair, U. Physically Unclonable Functions. In *Hardware Security: Design, Threats, and Safeguards*; Wiley, 2025; pp. 339–374.
9. Santikellur, P. Physically Unclonable Functions. In *Physically Unclonable Functions*; Springer, 2022; pp. 9–22.
10. Gaur, J.D.; Kumar Singh, A.; Singh, N.P.; Rajan V, G. Comparative Study on Different Encryption and Decryption Algorithm. In Proceedings of the In Proc. International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 2021, pp. 903–908. https://doi.org/10.1109/ICACITE51222.2021.9404734.
11. Suárez-Albela, M.; Fernández-Caramés, T.M.; Fraga-Lamas, P.; Castedo, L. A Practical Performance Comparison of ECC and RSA for Resource-Constrained IoT Devices. In Proceedings of the In Proc. Global Internet of Things Summit (GIoTS), 2018, pp. 1–6. https://doi.org/10.1109/GIOTS.2018.8534575.
12. Shamsoshoara, A.; Korenda, A.; Afghah, F.; Zeadally, S. A survey on physical unclonable function (PUF)-based security solutions for Internet of Things. *Computer Networks* **2020**, *183*, 107593.
13. 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 In Proc. The Great Lakes Symposium on VLSI 2023, New York, NY, USA, 2023; GLSVLSI '23, p. 249–254.
14. Wang, H.; Hao, W.; Tang, Y.; Zhu, B.; Dong, W.; Liu, W. Deep neural network modeling attacks on arbiter-PUF-based designs. *Cybersecurity* **2025**, *8*, 11.
15. Khan, R.; Eldeeb, H.B.; Mefgouda, B.; Alhussein, O.; Saleh, H.; Muhaidat, S. Encoder-decoder-based Virtual Physically Unclonable Function for Internet of Things device authentication using split-learning. *Computers and Security* **2025**, *148*, 104164.
16. Kumar, V.S.; Khalaf, O.I.; Chandan, R.R.; Bsoul, Q.; Gupta, S.K.; Zawaideh, F.; Alsekait, D.M.; AbdElminaam, D.S. Implementation of a Novel Secured Authentication Protocol for Cyber Security Applications. *Scientific Reports* **2024**, *14*, 25708.
17. Millwood, O.; Miskelly, J.; Yang, B.; Gope, P.; Kavun, E.B.; Lin, C. PUF-Phenotype: A Robust and Noise-Resilient Approach to Aid Intra-Group-based Authentication with DRAM-PUFs Using Machine Learning. *IEEE Transactions on Information Forensics and Security* **2023**, *18*, 1234–1245.
18. Sadhu, P.K.; Baul, A.; Yanambaka, V.P.; Abdelgawad, A. Machine Learning and PUF based Authentication Framework for Internet of Medical Things. In Proceedings of the In Proc. International Conference on Microelectronics (ICM), 2022, pp. 160–163.
19. Sajadi, A.; Shabani, A.; Alizadeh, B. DC-PUF: Machine Learning-Resistant PUF-Based Authentication Protocol Using Dependency Chain for Resource-Constrained IoT Devices. *Journal of Network and Computer Applications* **2023**, *217*, 103693.

20. Zhang, W.; Liu, M.; Chen, L. Deep Neural Network Modeling Attacks on Arbiter-PUF-Based Designs. *Cybersecurity* **2024**, *7*, 1–15.
21. Talukder, M.A.; Islam, M.M.; Uddin, M.A.; Hasan, K.F.; Sharmin, S.; Alyami, S.A.; Moni, M.A. Machine Learning-Based Network Intrusion Detection for Big and Imbalanced Data Using Oversampling, Stacking Feature Embedding and Feature Extraction. *Journal of Big Data* **2024**, *11*, 33.
22. Mahmood, R.K.; Mahameed, A.I.; Lateef, N.Q.; Jasim, H.M.; Radhi, A.D.; Ahmed, S.R.; Tupe-Waghmare, P. Optimizing network security with machine learning and multi-factor authentication for enhanced intrusion detection. *Journal of Robotics and Control (JRC)* **2024**, *5*, 1502–1524.
23. Zhou, H.; Chen, J.; Mei, Y.; Adam, G.; Aggarwal, V.; Bastian, N.D.; Lan, T. Real-time Network Intrusion Detection via Importance Sampled Decision Transformers. In Proceedings of the In Proc. IEEE 21st International Conference on Mobile Ad-Hoc and Smart Systems (MASS), 2024, pp. 82–91.
24. Hernandez-Ramos, J.; Karopoulos, G.; Chatzoglou, E.; Kouliaridis, V.; Marmol, E.; Gonzalez-Vidal, A.; Kambourakis, G. Intrusion Detection Based on Federated Learning: A Systematic Review. *ACM Comput. Surv.* **2025**. Just Accepted, https://doi.org/10.1145/3731596.
25. Debicha, I.; Cochez, B.; Kenaza, T.; Debatty, T. Adv-Bot: Realistic Adversarial Botnet Attacks Against Network Intrusion Detection Systems. *Computers & Security* **2023**, *125*, 103176.
26. Kaushik, S.; Bhardwaj, A.; Almogren, A.; Bharany, S.; Altameem, A.; Rehman, A.U.; Hussen, S.; Hamam, H. Robust Machine Learning Based Intrusion Detection System Using Simple Statistical Techniques in Feature Selection. *Scientific Reports* **2025**, *15*, 3970.
27. Tang, Z.T.A.; Yu, K.W.; Yuta, K.; Chen, T.Y.; Karati, A. Enhancing Security of a PUF-based Remote Keyless Entry System using Machine Learning Approach. In Proceedings of the In Proc. of the 2024 6th International Electronics Communication Conference, New York, NY, USA, 2024; IECC '24, p. 66–74.
28. 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.
29. Alzu'bi, A.; Alomar, A.; Alkhaza'leh, S.; Abuarqoub, A.; Hammoudeh, M. A Review of Privacy and Security of Edge Computing in Smart Healthcare Systems: Issues, Challenges, and Research Directions. *Tsinghua Science and Technology* **2024**, *29*, 1152–1180.
30. Myrzashova, R.; Alsamhi, S.H.; Hawbani, A.; Curry, E.; Guizani, M.; Wei, X. Safeguarding Patient Data-Sharing: Blockchain-Enabled Federated Learning in Medical Diagnostics. *IEEE Transactions on Sustainable Computing* **2025**, *10*, 176–189.
31. Alasmari, S.; Anwar, M. Security & Privacy Challenges in IoT-Based Health Cloud. In Proceedings of the In Proc. The International Conference on Computational Science and Computational Intelligence (CSCI). IEEE, 2016, pp. 198–201.
32. Amanat, A.; Rizwan, M.; Maple, C.; Zikria, Y.B.; Almadhor, A.S.; Kim, S.W. Blockchain and cloud computing-based secure electronic healthcare records storage and sharing. *Frontiers in Public Health* **2022**, *Volume 10 - 2022*.
33. An encrypted medical blockchain data search method with access control mechanism. *Information Processing & Management* **2023**, *60*, 103499.
34. Taherdoost, H. Blockchain-Based Internet of Medical Things. *Applied Sciences* **2023**, *13*.
35. Liu, Z.; Chen, Y.; Zhao, Y.; Yu, H.; Liu, Y.; Bao, R.; Jiang, J.; Nie, Z.; Xu, Q.; Yang, Q. Contribution-Aware Federated Learning for Smart Healthcare. *Proceedings of the AAAI Conference on Artificial Intelligence* **2022**, *36*, 12396–12404.
36. Alzoubi, Y.I.; Osmanaj, V.H.; Jaradat, A.; Al-Ahmad, A. Fog computing security and privacy for the Internet of Thing applications: State-of-the-art. *Security and Privacy* **2020**, *3*, e145.
37. Botín-Sanabria, D.M.; Mihaita, A.S.; Peimbert-García, R.E.; Ramírez-Moreno, M.A.; Ramírez-Mendoza, R.A.; Lozoya-Santos, J.d.J. Digital Twin Technology Challenges and Applications: A Comprehensive Review. *Remote Sensing* **2022**, *14*, 1335.
38. Masys, A.J., Ed. *Security by Design: Innovative Perspectives on Complex Problems*; Advanced Sciences and Technologies for Security Applications, Springer International Publishing: Cham, 2018.
39. Wang, X.; Zhang, Z.; Hao, Q.; Xu, D.; Wang, J.; Jia, H.; Zhou, Z. Hardware-Assisted Security Monitoring Unit for Real-Time Ensuring Secure Instruction Execution and Data Processing in Embedded Systems. *Micromachines* **2021**, *12*.
40. Singh, A., Hardware-Assisted Security of Smart IoT Applications. In *Hardware Security: Challenges and Solutions*; Mishra, A.; Goswami, M.; Kumar, M.; Rajput, N.S., Eds.; Springer Nature Switzerland: Cham, 2025; pp. 51–70.
41. Bathalapalli, V.K.V.V.; Mohanty, S.P.; Kougianos, E.; Iyer, V.; Rout, B. PUFchain 3.0: Hardware-Assisted Distributed Ledger for Robust Authentication in Healthcare Cyber-Physical Systems. *Sensors* **2024**, *24*, 938.
42. Jauernig, P.; Sadeghi, A.R.; Stapf, E. Trusted Execution Environments: Properties, Applications, and Challenges. *IEEE Security & Privacy* **2020**, *18*, 56–60.
43. Coppolino, L.; D'Antonio, S.; Mazzeo, G.; Romano, L. A comprehensive survey of hardware-assisted security: From the edge to the cloud. *Internet of Things* **2019**, *6*, 100055.
44. Tan, B. Challenges and Opportunities for Hardware-Assisted Security Improvements in the Field. In Proceedings of the In Proc. 23rd International Symposium on Quality Electronic Design (ISQED), 2022, pp. 90–95.

45. Halak, B.; Zwolinski, M.; Mispan, M.S. Overview of PUF-based hardware security solutions for the internet of things. In Proceedings of the In Proc. IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS), 2016, pp. 1–4.
46. Joshi, S.; Mohanty, S.P.; Kougianos, E. Everything You Wanted to Know About PUFs. *IEEE Potentials* **2017**, *36*, 38–46.
47. Wang, H.; Hao, W.; Tang, Y.; Zhu, B.; Dong, W.; Liu, W. Deep neural network modeling attacks on arbiter-PUF-based designs. *Cybersecurity* **2025**, *8*, 11. https://doi.org/10.1186/s42400-024-00308-7.
48. Ikezaki, Y.; Nozaki, Y.; Yoshikawa, M. Deep learning attack for physical unclonable function. In Proceedings of the In Proc. IEEE 5th Global Conference on Consumer Electronics, 2016, pp. 1–2. https://doi.org/10.1109/GCCE.2016.7800478.
49. Idrissi, I.C.E.; Nouh, S.; Bellfkih, E.M.; Marzak, A.; El Assad Assad, M. Error-Correcting Codes and the Power of Machine Learning: Enhancing Data Reliability. In Proceedings of the In Proc. 14th International Conference on Intelligent Systems: Theories and Applications (SITA), 2023, pp. 1–6.
50. Poniszewska-Maranda, A.; Sarniak, W. Automatic detection and correction of code errors applying machine learning - current research state. In Proceedings of the Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering, New York, NY, USA, 2024; EASE '24, p. 456–457.
51. Swischuk, R.; Allaire, D. A Machine Learning Approach to Aircraft Sensor Error Detection and Correction. *Journal of Computing and Information Science in Engineering* **2019**, *19*, 041009.
52. Akinci, T.C.; Erdemir, G.; Zengin, A.T.; Seker, S.; Idriss, A.I. Machine Learning-Based Error Correction Codes and Communication Protocols for Power Line Communication: An Overview. *IEEE Access* **2023**, *11*, 124760–124781.
53. Chen, Y.G.; Wu, T.Y. Special Session: Overcoming Transient Faults and Aging Effects in Digital Computing-in-Memory Architectures: Detection, Tolerance, and Mitigation Strategies. In Proceedings of the In Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2024, pp. 1–6.
54. Wang, Z.; Tang, H. Artificial Intelligence for Quantum Error Correction: A Comprehensive Review, 2024.
55. Zhu, P.; Zhang, H.; Shi, Y.; Xie, W.; Pang, M.; Shi, Y. A novel discrete conformable fractional grey system model for forecasting carbon dioxide emissions. *Environment, Development and Sustainability* **2024**, pp. 1–29.
56. Aarella, S.G.; Yanambaka, V.P.; Mohanty, S.P.; Kougianos, E. Fortified-Edge 4.0: A ML-Based Error Correction Framework for Secure Authentication in Collaborative Edge Computing, New York, NY, USA, 2024; GLSVLSI '24, p. 639–644.
57. Moeckel, C.; Mareboina, M.; Konnaris, M.A.; Chan, C.S.; Mouratidis, I.; Montgomery, A.; Chantzi, N.; Pavlopoulos, G.A.; Georgakopoulos-Soares, I. A survey of k-mer methods and applications in bioinformatics. *Computational and Structural Biotechnology Journal* **2024**, *23*, 2289–2303.
58. Aarella, S.G.; Yanambaka, V.P.; Mohanty, S.P.; Kougianos, E. Fortified-Edge 5.0: Federated Learning for Secure and Reliable PUF in Authentication Systems. In Proceedings of the IFIP/IEEE 32nd International Conference on Very Large Scale Integration (VLSI-SoC), 2024, pp. 1–6.
59. Aarella, S.G.; Mohanty, S.P.; Kougianos, E.; Puthal, D. Fortified-Edge 2.0: Machine Learning based Monitoring and Authentication of PUF-Integrated Secure Edge Data Center. In Proceedings of the In Proc. IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2023, pp. 1–6.
60. Sadhu, P.K.; Yanambaka, V.P. Easy-Sec: PUF-Based Rapid and Robust Authentication Framework for the Internet of Vehicles. In Proceedings of the In Proc. Internet of Things. Advances in Information and Communication Technology; Puthal, D.; Mohanty, S.; Choi, B.Y., Eds., Cham, 2024; pp. 262–279.