A Comparative Evaluation of Zero Knowledge Proof Techniques

Seyed Mohsen Rostamkolaei Motlagh[1], Claus Pahl[1][a], Hamid R. Barzegar[1] and Nabil El Ioini[2][b]
[1]*Free University of Bozen-Bolzano, 39100 Bolzano, Italy*
[2]*University of Nottingham, 43500 Semenyih, Malaysia*

Keywords: ZKP, Authentication, Distributed Systems, Bulletproof, zk-STARK, Experimental Comparison.

Abstract: Common to many distributed systems such as the Internet-of-Things (IoT) is decentralisation, often with a growing number of devices with diverse computational capabilities and security requirements. If integrated into critical applications, ensuring secure communication and data integrity is critical. In particular, privacy and security concerns are growing with the rise of these architectures. Zero-Knowledge Proof (ZKP) techniques are solutions to improve security without compromising on privacy. While the advantages of ZKP techniques are well-documented, it is important to consider the inherent limitations of these distributed environments, such as restricted processing power, memory capacity, and energy constraints, when aiming to solve security concerns. Here, we select two prominent ZKP techniques, Bulletproof and zk-STARK, and experimentally compare a number of their variants for a set of architecture-specific assessment criteria.

1 INTRODUCTION

Zero-knowledge proofs (ZKPs) are cryptographic tools that can enhance the security in distributed architectures. ZKPs allow one party (the prover) to prove to another party (the verifier) that a given statement is true without revealing any additional information beyond the truth of the statement itself (Goldreich and Oren, 1994), being more strict here than alternatives such as the Schnorr protocol used for authentication. This is valuable in architectures where it is often necessary to verify identities and transactions while preserving the privacy of the underlying data.

The selection of ZKPs as the technology for authentication here is driven by their unique ability to provide strong security guarantees without compromising privacy. Their application can ensure that devices and users in distributed environments such as IoT can be authenticated without revealing sensitive information. Traditional authentication mechanisms often require sharing or exposing some information that could potentially be intercepted or misused. ZKPs mitigate this risk by proving the validity of a claim without disclosing any additional data.

In this architecture, typically centralized authentication servers are used to validate the identity of devices in a network. While this approach is effective in conventional IT infrastructures, it presents several challenges in Internet-of-Things (IoT) environments:

- Scalability Issues: distributed systems can consist of a large number of devices, each requiring authentication. Centralized systems may struggle to handle the high volume of authentication requests, leading to latency bottlenecks (Yang et al., 2017).
- Single Point of Failure: Centralized authentication systems create a single point of failure. If the central server is compromised or becomes unavailable, the entire network's security can be jeopardized (Roman et al., 2013).
- Intermittent Connectivity: Often distributed devices, e.g., those in mobility scenarios, may experience intermittent connectivity. Traditional authentication methods that require constant connectivity to a central server are impractical in such situations (Atzori et al., 2010).
- Resource Constraints: Many devices have limited computational power, memory, and battery life. Traditional authentication methods, which may require extensive cryptographic operations, can be too resource-intensive for these devices (Mukherjee et al., 2017).

To address these challenges, innovative authentication mechanisms are required. Zero-knowledge proofs (ZKPs) offer a promising solution by enabling secure authentication without revealing sensitive information. In the context of IoT, ZKPs can facilitate:

- Scalable Authentication: ZKPs support decentralized authentication mechanisms, reducing reliance on central servers and enhancing scalability

[a] https://orcid.org/0000-0002-9049-212X
[b] https://orcid.org/0000-0002-1288-1082

Rostamkolaei Motlagh, S. M., Pahl, C., Barzegar, H. R. and El Ioini, N.
A Comparative Evaluation of Zero Knowledge Proof Techniques.
DOI: 10.5220/0013269100003944
In *Proceedings of the 10th International Conference on Internet of Things, Big Data and Security (IoTBDS 2025)*, pages 237-244
ISBN: 978-989-758-750-4; ISSN: 2184-4976

- Resilience to Connectivity Issues: ZKPs can enable authentication protocols that do not require constant connectivity to a central server, making them suitable for mobile and intermittently connected devices.
- Efficiency for Resource-Constrained Devices: Certain ZKP protocols are designed to be computationally efficient, making them suitable for devices with limited resources.
- Enhanced Privacy: ZKPs ensure that no sensitive information is disclosed during the authentication process, addressing privacy concerns in mobility scenarios. (Goldwasser et al., 2019)

ZKPs can be broadly classified into two types: interactive and non-interactive. Interactive zero-knowledge proofs involve a series of back-and-forth communications between the prover and the verifier. In contrast, non-interactive zero-knowledge proofs require only a single message from the prover to the verifier, making them more suitable for asynchronous and distributed environments (Goldwasser et al., 2019). We will explore the application of ZKPs in distributed systems where security and privacy are crucial. We will investigate ZKP protocols, including zk-STARKs and Bulletproofs as non-interactive ones in particular, and examine their effectiveness in enhancing security and privacy.

Given the challenges, we investigate ZKPs to enhance security and efficiency within authentication mechanisms for distributed systems. Due to the range of ZKP techniques available, this study focuses exclusively on Bulletproof and zk-STARK, chosen for their complementary strengths and alignment with the unique demands of resource-constrained environments. Specifically, the objectives are to:

- Conduct an in-depth evaluation of Bulletproof and zk-STARK to assess their suitability for distributed applications, focusing on key factors such as efficiency, scalability, and security.
- Identify an ZKP techniques that address the challenges of distributed settings, including dynamic operational conditions, resource constraints, and need for privacy-preserving authentication.

The primary objective here is to select sample protocols based on criteria such as efficiency, scalability, security (Werth et al., 2023a; Werth et al., 2023b), and their applicability to the unique challenges of IoT environments. We experimentally compare ZKP techniques considering the specific limitations of decentralisation and constrained devices. Given the resource constraints —such as limited processing power, memory, and energy— our research work aims to identify ZKP techniques that are most practical and effective for these environments.

While existing literature offers evaluations of ZKP techniques, none specifically address the trade-offs between Bulletproofs and zk-STARKs in resource-constrained IoT environments. Our work fills this gap by providing an experimental comparison, offering new insights into their practical applications.

The concrete Research Questions are: (1) How can Zero-Knowledge Proof (ZKP) techniques be effectively implemented in resource-constrained decentralized environments to enhance security? (2) How do selected ZKPs based on a systematic review compare in terms of proof generation time, verification time, and proof size when deployed in IoT environments with limited computational and memory resources? (3) What are the specific impacts of resource constraints (CPU, memory) on the performance of ZKP techniques in the above applications?

2 SELECTION PROCESS

The selection of Zero-Knowledge Proof (ZKP) techniques is a critical step in ensuring that the chosen methods are suitable for the unique challenges and constraints of some distributed environments. This section outlines the criteria and process used to select the ZKP techniques to be evaluated based on their features, advantages, and relevance.

2.1 Selection Criteria and Process

The selection of ZKP techniques in this study is guided by a set of key criteria, tailored to address the specific requirements and constraints of IoT environments. These criteria ensure the chosen techniques are not only theoretically robust but also practically implementable in resource-constrained, dynamic, and distributed IoT settings. The criteria and corresponding evaluation metrics are outlined in Table 1.

These criteria are pivotal in shaping the performance and security of ZKP techniques within IoT environments. By addressing the unique challenges posed by resource constraints, dynamic connectivity, and scalability demands, these criteria ensure that the selected ZKP techniques not only enhance security but also maintain operational efficiency and scalability, making them practical for real-world IoT applications (Bünz et al., 2018).

The process of selecting the ZKP techniques involved a comprehensive review of existing ZKP protocols, evaluating them against the aforementioned criteria. The selection process included the following

Table 1: Criteria for Selection and Corresponding Metrics for ZKP Techniques in IoT Environments.

Criterion	Description	Metric
Scalability	Ability of the technique to efficiently handle a large number of devices and high volumes of data in IoT environments. Scalability ensures performance is maintained as the system grows.	Evaluated by proof generation and verification times as the number of devices and data size increase. Additionally, the system's ability to maintain performance under varying network loads is assessed.
Efficiency	Reflects the computational and communication efficiency of the ZKP technique, particularly for resource-constrained IoT devices.	Measured by proof generation/verification times, computational resources needed, communication overhead. Lower values indicate higher efficiency.
Security Robustness	Ensures that the ZKP technique can resist a range of attacks, including quantum attacks, to protect data integrity and confidentiality.	Assessed based on theoretical guarantees, such as resistance to known attack vectors and tampering attempts.
No Trusted Setup	Preference for techniques that do not require a trusted setup phase, minimizing reliance on third parties and enhancing decentralization.	Evaluated by the absence of a trusted setup requirement and the complexity, duration, and necessity of initialization steps.
Compact Proof Size	Smaller proof sizes are essential for IoT devices with limited storage and constrained network bandwidth.	Measured in bytes. Smaller proofs are advantageous for minimizing storage requirements and reducing communication overhead.
Privacy-Preserving Capabilities	The ability to maintain privacy by ensuring no additional information beyond the validity of the statement is disclosed during proof verification.	Evaluated through theoretical guarantees and practical assessments of potential information leakage during proof generation and verification.
Applicability to Mobility Scenarios	Indicates how well the ZKP technique performs in dynamic environments involving mobile IoT devices, such as connected vehicles and wearables.	Assessed by performance metrics (proof generation and verification times, communication overhead) under simulated mobility scenarios, including frequent handovers and changing network conditions.

steps: *Literature Review*: An review of the literature on ZKP techniques was conducted to identify potential candidates. This included academic papers, technical reports, and industry publications. *Evaluation of Features*: The ZKP techniques were evaluated based on their features, such as proof generation and verification times, security properties, and whether they required a trusted setup. *Comparison and Analysis*: The techniques were compared against each other based on the selection criteria. Techniques that best met the criteria were shortlisted for further evaluation. *Final Selection and Justification*: Based on the comparison and analysis, Bulletproof and zk-STARK were selected for implementation. These techniques were chosen for their strong alignment with the criteria and their potential to address the specific challenges of constrained devices. *Experimental Validation*: The techniques Bulletproof and zk-STARK were implemented and tested in a controlled environment to validate performance and suitability.

2.2 Comparative ZKP Analysis

The tables below provide a comparative analysis of several ZKP techniques, highlighting their key features and suitability. Table 2 focuses on the technical aspects such as type of ZKP, communication cost, proof size, and setup requirements (Ben-Sasson et al., 2018; Ben-Sasson et al., 2017). These attributes are crucial for understanding the efficiency and practicality of each ZKP technique in IoT applications, where communication bandwidth and storage can be limited. Table 3 details the potential applications, advantages, and disadvantages of each ZKP technique (Bünz et al., 2018). This information provides a comprehensive overview of the suitability of each technique for specific IoT use cases and highlights the trade-offs involved in using each method. By analyzing the attributes, we identified *Bulletproof* and *zk-STARK* as the most suitable techniques for our focus, given their alignment with the selection criteria and potential to address the challenges of IoT devices.

2.3 Bulletproof and zk-STARK

The selection of Bulletproof and zk-STARK is based on their distinct advantages and suitability.

Bulletproofs are a non-interactive zero-knowledge proof type good for compactness and efficiency. Unlike many other zero-knowledge proofs, Bulletproofs do not require a trusted setup phase, making them more practical and secure. They are particularly noted for their effectiveness in range proofs, which are essential in verifying that a secret value lies within a certain range without revealing the value itself. This feature is crucial in the context of cryptocurrencies and financial systems, where Bulletproofs are used to enable confidential transactions. For instance, Bulletproofs allow transaction amounts to remain hidden while still ensuring the transactions are valid and secure (Bünz et al., 2018). Advantages of Bulletproofs: No Trusted Setup Required: Enhances se-

Table 2: Comparative Analysis of ZKP Techniques (Part 1: Technology).

ZKP	*Type*	*Communication Cost*	*Proof Size*	*Setup Requirements*
Groth's zkSNARK	Non-interactive	Low	Moderate	Requires a trusted setup
PLONK	Non-interactive	Low	Very Small	Does not require a trusted setup
FRI	Non-interactive	Very Low	Medium	Does not require a trusted setup
ZKBoo	Non-interactive	Very Low	Very Small	Does not require a trusted setup
Halo	General framework	Varies	Varies	Varies
Bulletproofs	Non-interactive	Low	Very Small	Does not require a trusted setup
zk-STARKs	Non-interactive	Medium	Large	Does not require a trusted setup
zk-SNARK	Non-interactive	Low	Moderate	Requires a trusted setup
Ligero	Non-interactive	Medium	Medium	Does not require a trusted setup

Table 3: Comparative Analysis of ZKP Techniques (Part 2: Applicabilty).

ZKP	*Potential Applications*	*Advantages*	*Disadvantages*
Groth's zk-SNARK	Privacy-preserving cryptocurrencies, anonymous credentials	Efficient, proofs are relatively small	Require a trusted setup, vulnerable to attacks if setup is compromised
PLONK	Privacy-preserving cryptocurrencies, secure voting systems	Very efficient, proofs are very small	Can be less versatile
FRI	Privacy-preserving applications	Very efficient, can handle complex computations	Can be difficult to implement
ZKBoo	Privacy-preserving applications	Very efficient, proofs are very small	Can be less versatile
Halo	Privacy-preserving cryptocurrencies, anonymous credentials	Flexible, versatile	Complex, requires expertise to implement
Bulletproofs	Confidential transactions, privacy-preserving applications	No trusted setup, compact proofs	Higher computational cost for proof generation
zk-STARKs	Large-scale computations, blockchain	Scalable, post-quantum secure	Larger proof sizes
zk-SNARK	Privacy-preserving cryptocurrencies, anonymous authentication, secure voting	Efficient, proofs are relatively small	Requires a trusted setup, can be vulnerable to trust attacks
Ligero	Privacy-preserving applications	Efficient, medium-sized proofs	Can be less efficient in some use cases

curity by eliminating the need for a trusted party to initialize the protocol. Compact Proofs: Bulletproofs produce smaller proof sizes, which are efficient for storage and transmission, making them suitable for resource-constrained IoT devices. Efficient Verification: The verification process in Bulletproofs is computationally efficient, enabling rapid validation of proofs, which is critical for real-time IoT applications.

zk-STARKs (Zero-Knowledge Scalable Transparent Arguments of Knowledge) offer advancements in scalability and transparency. Introduced to address limitations of zk-SNARKs, zk-STARKs eliminate the need for a trusted setup and are designed to be secure against quantum computing attacks. They are particularly suited for applications requiring scalability and high security, such as large-scale data verification and blockchain technologies (Ben-Sasson et al., 2018; Pahl and El Ioini, 2019; Berenjestanaki et al., 2023). Advantages of zk-STARKs include: Scalability: zk-STARKs can handle large-scale computations efficiently, ideal for environments with extensive data processing requirements. Transparency: Reliance on publicly verifiable randomness ensures transparency and eliminates risks associated with a trusted setup. Post-Quantum Security: zk-STARKs are designed to be secure against quantum computing attacks.

3 EXPERIMENT SETUP

This section outlines the implementation of the two ZKP techniques, Bulletproof and zk-STARK, with variations designed to address specific challenges and configurations encountered during the study. The experiments were conducted in a simulated IoT environment, carefully designed to reflect the decentralization, openness, and resource constraints characteristic of the targeted application scenarios.

3.1 Environment Setup

The experimental setup includes the hardware and software environments, test scenarios, and the procedures followed to conduct the experiments.

Hardware Environment. To ensure a realistic evaluation of the techniques, we used a combination of standard IoT devices and more powerful computing resources. Emulated using Docker containers with constrained CPU and memory resources to simulate real-world IoT device limitations. A dedicated server was used to handle proof verification and respond to the clients. Specifications include an *Asus K556U* equipped with an Intel Core i5 processor, 8 GB RAM, and running Ubuntu Server 20.04 LTS.

Software Environment. The software environment was set up to facilitate the implementation and evaluation of the ZKP techniques. Rust was used for implementing the client and server applications for both Bulletproof and zk-STARK. For *Bulletproofs*, the official Rust library for Bulletproofs was used. For *zk-STARK*, a Rust library implementation of zk-STARK was used. Docker was used to containerize the applications, allowing for controlled resource allocation and easy deployment.

3.1.1 Test Scenarios

To evaluate the performance and suitability of Bulletproof and zk-STARK, we designed three test scenarios that reflect real-world use cases. *Scenario 1: Proof Generation and Verification*: Evaluating the performance of ZKP techniques by measuring the time taken to generate and verify proofs, as well as the proof size. *Scenario 2: Resource-Constrained Environment*: Assessing the impact of limited CPU and memory resources on proof generation and verification times. This scenario was tested by running the client and server in Docker containers with constrained resources: *Configuration 1*: CPU: 0.7, Memory: 100 MB, 64-bit Bulletproof *Configuration 2*: CPU: 0.3, Memory: 500 MB, 64-bit Bulletproof *Configuration 3*: CPU: 0.3, Memory: 500 MB, 8-bit Bulletproof *Configuration 4*: CPU: 0.7, Memory: 100 MB, 8-bit Bulletproof Each configuration was run 100 times, and the average results were recorded. *Scenario 3: Concurrent Proof Verification*: Testing the performance of ZKP techniques by sending multiple proofs simultaneously from the client to the server and measuring the impact on verification time.

3.1.2 Procedure

The experimental procedure followed a structured approach to ensure consistency and repeatability: *Implementation* of ZKP techniques, Bulletproof and zk-STARK, on the client and server applications using Rust. *Configuration* of Docker containers to limit CPU and memory resources for the client and server applications. *Data Collection* on proof generation time, verification time, and proof size. Additionally, collect data on the impact of resource constraints and concurrent proof verification. *Repetition*: Perform each experiment 100 times for each configuration to ensure statistical significance. Calculate the average from the trials. *Analysis* of collected data to evaluate the performance of the ZKP techniques based on the defined evaluation metrics. *Validation* of results through repeated trials and cross-comparison with theoretical expectations.

3.1.3 Experimental Challenges

Several challenges were encountered during the experimental setup and execution. *Resource Constraints*: Simulating realistic device constraints using Docker required configuration and monitoring of CPU and memory limits. *Implementation Complexity*: The complexity of implementing advanced cryptographic techniques like Bulletproof and zk-STARK required significant effort and expertise in Rust. *Concurrent Verification*: Ensuring accurate measurement of the impact of concurrent proof verification involves precise timing and synchronization mechanisms. Despite these challenges, the experimental setup provided a robust framework for evaluating the performance and suitability of Bulletproof and zk-STARK.

4 EXPERIMENTAL COMPARISON

The comparison focuses metrics of proof generation time, verification time and proof size under different computational and memory constraints.

We created two different configurations for each protocol, representing different CPU and memory allocations: 0.7/100 (high CPU and low memory) and 0.3/500 (low CPU and high memory). This allows to simulate different real-world scenarios and to provide a complete evaluation of the performance of Bulletproof and zk-STARK protocols under limits of computational and memory resources.

4.1 Proof Generation Time Comparison

The comparison between Bulletproof and zk-STARK focused on three key metrics: proof generation time, verification time, and proof size. The results are sum-

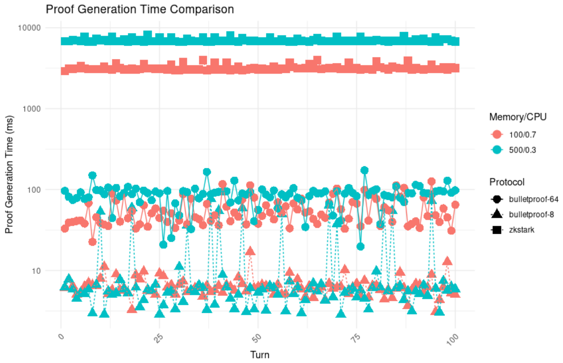

Figure 1: Proof Generation Time Comparison between Bulletproof and zk-STARK Protocols.

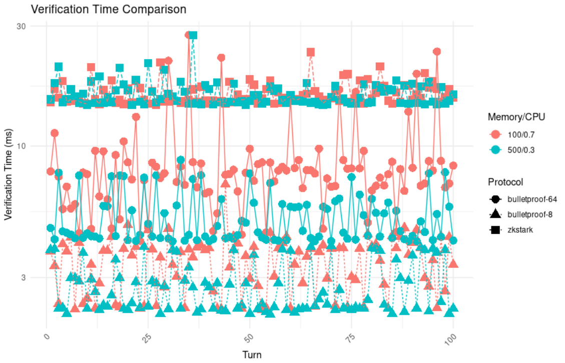

Figure 2: Verification Time Comparison between Bulletproof and zk-STARK Protocols.

marized in Table 4 and Figures 1 and 2.

Figure 1 illustrates the proof generation time for different configurations of Bulletproof and zk-STARK protocols over 100 test runs. The x-axis represents the turn of each test run, while the y-axis shows the proof generation time in milliseconds on a logarithmic scale. The shapes of the points differentiate between the protocols: triangles for Bulletproof-8, circles for Bulletproof-64, and squares for zk-STARK. The colors represent different CPU/Memory allocations: red for 0.3/500 and blue for 0.7/100.

1) Bulletproof (8-bit and 64-bit): The proof generation time for Bulletproof with 8-bit configuration is consistently lower compared to the 64-bit configuration across all tests. This is expected, as generating proofs with lower bit sizes requires less computational effort. The CPU/Memory allocation impacts the proof generation time. Higher CPU allocation (0.7 CPU) results in faster proof generation times compared to lower CPU allocation (0.3 CPU). There is a consistently lower position of the blue points compared to the red points for both 8-bit and 64-bit configurations.

2) zk-STARK: zk-STARK exhibits a considerably higher proof generation time compared to Bulletproof across all tests, especially for the 64-bit configuration, caused by the computational intensity of zk-STARK's cryptographic operations. The impact of CPU/Memory allocation is also significant for zk-STARK. Higher CPU allocation leads to reduced proof generation times – see consistently lower blue points compared to red points.

The average proof generation times for each protocol and configuration are presented in Table 4. These averages highlight the performance differences between the protocols and the impact of different CPU/Memory allocations on proof generation time.

The requirement for proof generation time is significant to many restricted devices as they do not have a lot of computation power to use. The time it takes to generate a proof is important when considering any potential real-world cases as it will determine the practicality and security of the system. Empirical results in this document help to affirm that Bulletproofs, due to their lower proof generation times in contrast to zk-STARKs, make it more applicable for real-time use-cases which necessitate a quick generation of a proof. In conclusion, Bulletproofs are more suited for deployment in resource-limited environments and applications requiring quick, secure operations.

4.2 Verification Time Comparison

Figure 2 illustrates the verification time for different configurations of Bulletproof and zk-STARK protocols over 100 test runs. The x-axis represents the turn of each test run, while the y-axis shows the verification time in milliseconds on a logarithmic scale. The shapes of the points differentiate between the protocols: triangles for Bulletproof-8, circles for Bulletproof-64, and squares for zk-STARK. The colors represent different CPU/Memory allocations: red for 0.3/500 and blue for 0.7/100.

1. Bulletproof (8-bit and 64-bit): The verification time for Bulletproof follows a similar trend to the proof generation time. The 8-bit configuration results in lower verification times compared to the 64-bit configuration across all test runs, due to the reduced computational effort required for lower bit sizes. Higher CPU allocation (0.7/100) results in faster verification times compared to the lower CPU allocation (0.3/500), demonstrating the importance of computational resources in the verification process. This is visible from consistently lower position of blue points compared to red points for both 8-bit and 64-bit.
2. zk-STARK: zk-STARK shows higher verification times compared to Bulletproof across all test runs, reflecting its higher computational complexity.

Table 4: Performance Metrics for Bulletproof and zk-STARK.

Configuration	*Proof Generation Time (ms)*	*Verification Time (ms)*	*Proof Size (bytes)*	*CPU/Memory*
Bulletproof-8_1	6.933	3.692	480	0.7/100
Bulletproof-8_2	8.545	2.887	480	0.3/500
Bulletproof-64_1	55.200	8.634	672	0.7/100
Bulletproof-64_2	72.915	4.693	672	0.3/500
zk-STARK_1	3217.507	16.155	55400	0.7/100
zk-STARK_2	7082.275	16.918	55400	0.3/500

The increased complexity of zk-STARK's cryptographic operations results in longer verification times. Similar to proof generation, higher CPU allocation reduces the verification time for zk-STARK, highlighting the significant impact of resource constraints on zk-STARK performance. This is seen by the consistently lower blue points compared to the red points.

The average verification times for each protocol and configuration are presented in Table 4. These averages emphasize the performance differences between the protocols and the effect of different CPU/Memory allocations on verification time.

4.3 Comparative Analysis

Our findings on verification times align with and expand upon the existing body of research. Prior studies have indicated that Bulletproof generally exhibits faster verification times compared to zk-STARK due to its more efficient cryptographic operations. For instance, Ben-Sasson et al. (2018) (Ben-Sasson et al., 2018) demonstrated that zk-STARKs, while providing strong security guarantees and scalability, involve more computationally intensive operations compared to Bulletproof, resulting in longer verification times.

In comparison to our results, the study by Bünz et al. (2018) (Bünz et al., 2018), which introduced Bulletproofs, highlighted their efficiency and smaller proof sizes, leading to quicker verification processes. Our findings corroborate this, showing that Bulletproof maintains lower verification times across different CPU and memory configurations. Specifically, our results indicate that Bulletproof's 8-bit and 64-bit configurations consistently outperform zk-STARK in verification time, reinforcing the suitability of Bulletproof for real-time applications in resource-constrained IoT environments.

Similar studies focusing on ZKP in IoT contexts (Narula et al., 2018; Androulaki et al., 2018) also underscore the criticality of low verification times for practical deployment. They argue that IoT devices, with their limited computational power, benefit significantly from ZKP techniques that minimize computational overhead. Bulletproof's lower verification times make it more appropriate for deployment in IoT systems where timely verification is crucial.

4.4 Proof Size Comparison

The proof sizes for different configurations vary. For Bulletproof it increases with the bit size used for proof generation. The 8-bit configuration results in smaller proof sizes (480 bytes) compared to the 64-bit configuration (672 bytes). Smaller proof sizes are advantageous in IoT environments as they require less storage space and bandwidth, which are typically limited in such settings. zk-STARK produces significantly larger proofs (55400 bytes) compared to Bulletproof. This is due to the inherent complexity and design of zk-STARK, which involves more extensive cryptographic computations. Larger proof sizes can be a disadvantage in IoT environments.

4.5 Verification Time - Concurrency

To evaluate the performance under concurrent requests, the verification time was measured while sending 4 requests simultaneously. The average verification time over 100 test runs (totaling 400 verification times per protocol) varies between 9.236 ms for Bulletproof (64-bit) and 27.352 ms for zk-STARK.

The results reveal that Bulletproof provides a significantly lower verification time than zk-STARK for multiple concurrent requests. This is important for IoT applications, as devices may need to manage multiple tasks at the same time. The Bulletproof's lower verification time demonstrates its efficiency and consequent suitability for real-time applications in resource-constrained scenarios. Likewise, the higher verification time of zk-STARK with concurrent requests leads to thinking of its practicality in such scenarios. Hence, selecting the appropriate ZKP protocols based on the required use-case is important.

4.6 Discussion

The evaluation results demonstrate the comparative strengths and weaknesses of Bulletproof and zk-STARK for distributed environments, particularly IoT applications. Bulletproof outperforms zk-STARK in proof generation time, with the former generating proofs far more quickly. This efficiency is critical for devices with limited computational power, where prolonged proof generation can lead to delays and increased energy consumption, negatively impacting battery life and operational sustainability. zk-STARK, by contrast, exhibits higher proof generation times due to its computational complexity, making it less suited for resource-constrained scenarios. In terms of verification time, Bulletproof excels in both single-request and concurrent-request scenarios, making it suitable for real-time applications requiring low-latency performance. zk-STARK, while providing stronger security guarantees, incurs longer verification times, which limits applicability in time-sensitive environments. Bulletproof's ability to handle concurrent verification requests further highlights its robustness and practicality for high-load applications. zk-STARK's longer verification times under such conditions indicate potential bottlenecks in scenarios requiring rapid and simultaneous proof verifications. While zk-STARK's scalability and post-quantum security provide valuable long-term benefits, these come at the cost of higher computational and storage requirements, reducing its practicality for IoT applications. Overall, Bulletproof is a more feasible choice for resource-constrained IoT systems, balancing efficiency and security effectively. Its lower proof generation and verification times, coupled with compact proof sizes, make it better for real-time and low-power applications. zk-STARK is suited for scenarios prioritizing robust security over performance.

5 CONCLUSIONS

This study evaluated the performance of Bulletproof and zk-STARK zero-knowledge proof systems in distributed, decentralized, and resource-constrained environments, focusing on proof generation time, verification time, and proof size. The results show that Bulletproof consistently outperforms zk-STARK in terms of efficiency, producing faster proofs, smaller proof sizes, and shorter verification times, making it a practical choice for real-time, energy-efficient, and bandwidth-limited IoT applications. While zk-STARK provides robust security and scalability, its higher resource demands limit its suitability for constrained settings. Future work could explore hybrid approaches that combine both protocols and investigate optimizations (El Ioini and Pahl, 2018) to enhance zk-STARK's performance in resource-constrained environments.

REFERENCES

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al. (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. In *EuroSys*, pages 1–15.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Comp Netw*, 54(15):2787–2805.

Ben-Sasson, E., Bentov, I., Horesh, Y., and Riabzev, M. (2018). Scalable, transparent, and post-quantum secure computational integrity. *Crypt ePrint Arch.*

Ben-Sasson, E., Chiesa, A., Tromer, E., and Virza, M. (2017). Scalable zero knowledge via cycles of elliptic curves. *Algorithmica*, 79:1102–1160.

Berenjestanaki, M. H., Barzegar, H. R., El Ioini, N., and Pahl, C. (2023). Blockchain-based e-voting systems: a technology review. *Electronics*, 13(1):17.

Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., and Maxwell, G. (2018). Bulletproofs: Short proofs for confidential transactions and more. In *Sym Sec&Priv*.

El Ioini, N. and Pahl, C. (2018). Trustworthy orchestration of container based edge computing using permissioned blockchain. In *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, pages 147–154.

Goldreich, O. and Oren, Y. (1994). Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32.

Goldwasser, S., Micali, S., and Rackoff, C. (2019). The knowledge complexity of interactive proof-systems. In *Providing sound foundations for cryptography*.

Mukherjee, M., Matam, R., Shu, L., Maglaras, L., Ferrag, M. A., Choudhury, N., and Kumar, V. (2017). Security and privacy in fog computing: Challenges. *IEEE Access*, 5:19293–19304.

Narula, N., Vasquez, W., and Virza, M. (2018). {zkLedger}:{Privacy-Preserving} auditing for distributed ledgers. In *Symposium on networked systems design and implementation*.

Pahl, C. and El Ioini, N. (2019). Blockchain based service continuity in mobile edge computing. In *IOTSMS*, pages 136–141.

Roman, R., Zhou, J., and Lopez, J. (2013). On the features and challenges of security and privacy in distributed internet of things. *Comp Netw*, 57(10):2266–2279.

Werth, J., Berenjestanaki, M. H., Barzegar, H. R., El Ioini, N., and Pahl, C. (2023a). A review of blockchain platforms based on the scalability, security and decentralization trilemma.

Werth, J., El Ioini, N., Berenjestanaki, M. H., Barzegar, H. R., and Pahl, C. (2023b). A platform selection framework for blockchain-based software systems based on the blockchain trilemma.

Yang, Y., Wu, L., Yin, G., Li, L., and Zhao, H. (2017). A survey on security and privacy issues in internet-of-things. *IEEE Internet of things Jrnl*, 4(5):1250–1258.