engineering proceedings

MDPI

Proceeding Paper

Zero-Knowledge Proof Protocols for Smart Contract Verification in Smart City Environments †

Benjamin Oholeguy [1], Jeff Nijsse [2,*] and Shu Su [3]

[1] School of Engineering, Computer, and Mathematical Sciences, Auckland University of Technology, Auckland 1010, New Zealand; benjaoholeguy@gmail.com
[2] Department of Software Engineering, RMIT University, Hanoi 10000, Vietnam
[3] Department of Mathematical Sciences, Auckland University of Technology, Auckland 1010, New Zealand; shu.su@aut.ac.nz
* Correspondence: jeff.nijsse@rmit.edu.vn
† Presented at the 2026 International Conference on Energy, Infrastructure and Environmental Research (EIER 2026), Ho Chi Minh City, Vietnam, 15–17 July 2026.

Abstract

While zero-knowledge proofs are increasingly advocated for privacy-preserving smart city applications, practitioners lack empirical guidance on which zk-SNARK protocol to select for permissioned blockchain deployments. Existing benchmarks overwhelmingly target permissionless chains such as Ethereum, leaving a gap for environments like Hyperledger Fabric where smart city systems are more commonly deployed. This paper addresses that gap by presenting a controlled latency comparison of three zk-SNARK protocols—Groth16, PlonK, and Fflonk—within a Hyperledger Fabric prototype that implements a reusable inter-contract private verification pattern: a residential apartment smart contract proves its energy consumption to a utility contract without revealing granular underlying data. Over 10,000 sequential verification queries, Fflonk achieved the lowest mean latency (20.54 ms, $\sigma = 1.47$ ms) compared with Groth16 (22.49 ms, $\sigma = 1.90$ ms) and PlonK (24.35 ms, $\sigma = 1.12$ ms). However, Fflonk requires approximately 90% more resources at deployment than Groth16, presenting a concrete setup-cost versus runtime-latency trade-off. These results provide actionable selection criteria for practitioners designing latency-sensitive smart city systems on permissioned blockchains.

Keywords: zero-knowledge proof; blockchain; smart contract; smart city; zk-SNARK; Groth16; PlonK; Fflonk; Hyperledger Fabric; latency

1. Introduction

Smart cities depend on the secure and privacy-preserving exchange of data between interconnected systems—energy grids, transport networks, and utility providers—that generate large volumes of sensitive information [1]. Blockchain technology has emerged as a foundational tool for decentralised trust in such environments [2,3], yet blockchains are not inherently suited to storing large or sensitive datasets due to block-size constraints and the public visibility of on-chain data [4]. Despite these limitations, the decentralisation of transaction authentication [5], the elimination of single points of failure [6], and the tamper-resistant properties of distributed ledgers make blockchain a compelling substrate for smart city applications—provided that a complementary privacy layer is introduced.

Zero-knowledge proofs (ZKPs) supply this layer by enabling one party (the prover) to demonstrate the validity of a statement to another party (the verifier) without disclosing

Check for updates

Academic Editor: Filbert H. Juwono

Published: 12 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

the underlying data [7]. Among ZKP constructions, Succinct Non-interactive Arguments of Knowledge (zk-SNARKs) are especially suited to blockchain environments due to their compact proof size and efficient verification [8]. Recent work has begun to explore ZKP integration with Hyperledger Fabric: van der Velden [9] measured the overhead of a single ZKP scheme on Fabric, and Kang et al. [10] proposed architectural extensions for ZKP support, while Guo et al. [11] benchmarked multiple SNARK protocols on Ethereum. However, these studies either evaluate a single protocol, target permissionless chains, or focus on architectural feasibility rather than cross-protocol latency. To our knowledge, no controlled, cross-protocol latency comparison of zk-SNARK proving systems exists for permissioned blockchain deployments, the very environment that smart city consortia prefer [4,12].

This paper addresses that gap with three contributions. First, we provide, to our knowledge, the first cross-protocol latency comparison of Groth16 [8], PlonK [13], and Fflonk [14] executed on Hyperledger Fabric, measuring 10,000 sequential verification queries per protocol. Second, we introduce an inter-contract private verification pattern—demonstrated through an energy-consumer (apartment) and energy-supplier interaction—that generalises to any smart city scenario in which two contracts must verify a claim without sharing raw data. Third, we quantify the engineering trade-off: Fflonk achieves the lowest and most consistent latency ($\bar{x} = 20.54$ ms, $\sigma = 1.47$ ms) while consuming approximately 90% more deployment resources than Groth16, giving practitioners actionable selection criteria.

The remainder of this paper is structured as follows. Section 2 provides background on ZKPs and reviews related work. Section 3 describes the prototype design, including the Circom circuit and the inter-contract verification pattern. Section 4 presents and analyses the latency results. Finally, Section 5 summarises the findings and outlines directions for future research.

2. Background and Related Work

2.1. Zero-Knowledge Proofs

A zero-knowledge proof is a cryptographic technique that enables a prover to demonstrate the truth of a statement to a verifier without revealing the underlying information [7]. ZKPs have found applications in blockchains related to identity verification (e.g., Galaxy, Filecoin) [15,16], privacy-preserving transactions (e.g., Zcash) [16,17], and data verification [16,18]. The concept of computational soundness ensures that the system remains resilient against attempts by a computationally powerful and malicious prover to deceive the verifier [18].

2.2. ZKP Verification in Smart Cities

Verification is essential in smart cities, especially for managing connected systems, decentralised architectures, and data-sharing tasks [12]. The well-known Fiat–Shamir heuristic [19,20] presents a non-interactive ZKP verification scheme involving two actors: Alice (the prover) and Bob (the verifier). Alice knows her secret s and computes $PK_k = g^s \pmod{p}$. She selects a random number r_n and creates a commitment $PK_{r_n} = g^{r_n}$. Alice then calculates her own challenge $c = \text{Hash}(g \| s \| PK_k \| PK_{r_n})$, computes $r = r_n - c \cdot s$, and sends PK_{r_n}, c, and r to Bob. Bob verifies by computing $V_{\text{verify}} = g^r \cdot (g^s)^c$ and checking whether $V_{\text{verify}} = PK_{r_n}$.

2.3. zk-SNARK Protocols

Groth16. Groth [8] introduced Succinct Non-interactive Arguments of Knowledge (SNARKs), which are more compact and efficient to verify than earlier pairing-based constructions. Groth16 requires the evaluation of a single pairing product equation in-

volving three pairings, making it well-suited for blockchain applications with limited computing resources.

PlonK. Gabizon et al. [13] introduced PlonK (Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge), a universal and updateable proof system, with subsequent work refining its arithmetization [16]. PlonK is efficient in verification time and has been adopted in several environments, including Filecoin for decentralised storage and Zcash for private transactions [16]. Unlike Groth16, PlonK employs a universal structured reference string that can be reused across circuits, eliminating the need for a new trusted setup ceremony each time a circuit changes.

Fflonk. Introduced by Gabizon and Williamson [14], Fflonk is a Fast-Fourier inspired, verifier-efficient variant of the PlonK proving system, from which its name derives. It reduces the number of elliptic-curve group operations required during verification—from 16–18 in the original PlonK to just five—at the cost of increased prover computation [11]. In terms of the SnarkJS implementation, Fflonk avoids the separate phase 1 and phase 2 setup steps and consumes approximately 5% and 30% fewer resources than Groth16 and PlonK, respectively, during verification. However, smart contract deployment using Fflonk requires approximately 90% more resources than Groth16 [11].

Protocol selection rationale. These three protocols were selected because they occupy distinct points in the design space of practical zk-SNARK proving systems, enabling a comparison between established and more recent approaches under an identical SnarkJS toolchain. Groth16 serves as the established baseline: it produces very small proofs that enable fast verification, but it requires a fresh circuit-specific trusted setup. PlonK was chosen for its reusable, universal setup, which makes it more flexible when the underlying circuit changes. Fflonk was selected because it is a newer, PlonK-based construction that aims to further improve verification efficiency. Together, these protocols provide a representative comparison between established and emerging zk-SNARK approaches while remaining directly comparable through a shared implementation and circuit.

2.4. Empirical Benchmarking of zk-SNARKs

A growing body of work has sought to characterise the practical performance of zk-SNARK protocols, yet most evaluations remain confined to permissionless blockchains. Guo et al. [11] benchmarked ZK-friendly hash functions alongside Groth16, PlonK, and Fflonk on EVM-compatible chains, reporting gas costs and proof-generation times. Their results showed that Fflonk can achieve lower on-chain verification costs than Groth16 when the number of public signals is large, but the study did not assess end-to-end verification latency in a permissioned environment.

On the permissioned side, van der Velden [9] measured the performance impact of integrating a single ZKP scheme into Hyperledger Fabric for privacy-preserving authentication, finding that ZKP-enabled transactions incurred 30–87.5% higher latency compared with a non-ZKP baseline. While this work confirmed that ZKP overhead on Fabric is non-trivial, it evaluated only a single protocol and did not compare alternative proving systems. Kang et al. [10] proposed FabZK, an architectural extension to Hyperledger Fabric that adds native support for Pedersen commitments and zero-knowledge range proofs. Their contribution focuses on system design and privacy guarantees rather than a comparative latency analysis of different SNARK back-ends.

Taken together, these studies establish the feasibility and architectural foundations for ZKPs on permissioned blockchains, but none provides a controlled, cross-protocol latency comparison of the kind that practitioners need to make informed protocol-selection decisions. The present work addresses this gap.

https://doi.org/10.3390/engproc2026164006

2.5. Trusted Setup

Phase one of generating a trusted setup for a zk-SNARK is called the powers of tau. A trusted setup ceremony is a one-time process that produces public parameters while also creating a trapdoor that must be securely destroyed. A randomised procedure Setup() derives a Structured Reference String (SRS) and a private key τ during the ceremony. In some cryptocurrency applications, Multiparty Computation (MPC) is employed so that no single participant knows the private key [21]. These ceremonies vary in the number of participants, rounds, and trust assumptions, and are typically coordinated by a centralised entity.

3. Prototype Design

The verification circuit was implemented using Circom, a domain-specific language for defining arithmetic circuits for zk-SNARK proofs. The tool has three components: a circuit compiler that defines constraints to be verified without revealing underlying data; a proof generator that creates a zk-SNARK proof verifying compliance with circuit conditions; and a proof verifier that confirms the hidden input data satisfy the specified constraints.

The circuit mathematically guarantees to the utility that the client correctly computed the severe cubic penalty for their load impact, explicitly eliminating the requirement to transmit granular behavioural data (`factorA` and `factorB`) to centralised servers. Instead of continuously streaming high-frequency smart meter readings to the utility to determine demand response compliance, the client's local energy management system evaluates the specific load parameters internally. The utility receives only the final computed tariff and the cryptographic proof, verifying billing accuracy without ever accessing the client's sensitive appliance usage signatures. Consequently, the utility securely enforces grid stability via economic disincentives while the client maintains absolute data sovereignty over their localised consumption patterns.

The circuit definition is given in Listing 1.

Listing 1. Circuit definition in Circom syntax showing the two private inputs (`factorA` and `factorB`), the public input (`expectedCube`), and the constraints that verify the cubed product of the private factors equals the public value without revealing the factors themselves.

```
template CubedProductVerification() {
    signal input factorA;
    signal input factorB;
    signal input expectedCube;

    signal baseProduct;
    signal squaredProduct;
    signal output computedCube;

    baseProduct <== factorA * factorB;
    squaredProduct <== baseProduct * baseProduct;
    computedCube <== baseProduct * squaredProduct;

    computedCube === expectedCube;
}
component main {public [expectedCube]} = CubedProductVerification();
```

The signals `factorA` and `factorB` represent private inputs to the circuit. The signal `expectedCube` is a public input visible during proof generation and verification. The constraints define the following mathematical relationships: baseProduct = factorA × factorB, squaredProduct = baseProduct^2, and computedCube = baseProduct × squaredProduct. The assertion `computedCube === expectedCube` enforces circuit validity.

https://doi.org/10.3390/engproc2026164006

This arithmetic circuit was deliberately kept simple. A fixed, low-complexity circuit allows the three protocols to be compared in a controlled environment in which the measured latency differences can be attributed to the proving systems themselves rather than to circuit-specific factors. Isolating protocol-level latency in this way was the explicit goal, rather than reproducing the full computational complexity of a production smart city deployment. The circuit nonetheless captures the essential interaction studied here—a private computation whose correctness is proven to a counterparty without revealing the inputs—and the effect of larger, more complex circuits is identified as a direction for future work (Section 5).

3.1. Smart Contract Interaction Model

The interaction between a residential apartment, for example, (acting as the energy consumer) and an energy provider/utility (acting as the supplier) smart contracts is mediated by a ZKP verification function shown in Figure 1. The apartment smart contract generates a proof of its energy usage, which is submitted to the energy smart contract for verification. The ZKP verify function receives the signal from the apartment contract and determines whether the request is authorised, without revealing the underlying consumption data.

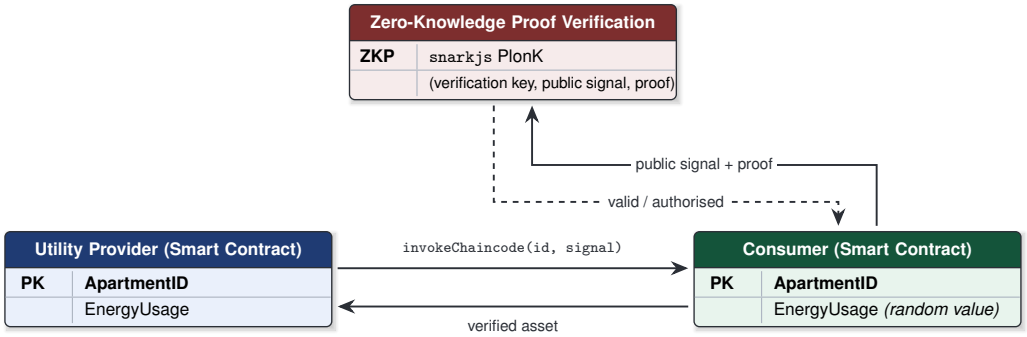


Figure 1. Smart contract interaction model realising the inter-contract private verification pattern. The utility provider invokes the consumer (apartment) contract via `invokeChaincode(id, signal)`; the consumer's signal is checked by the SnarkJS (PlonK) zero-knowledge verifier, and only the verified asset is returned to the provider, so the underlying energy-consumption data is never disclosed.

3.2. Verification Sequence

The residential apartment smart contract uses the SnarkJS library with each of the three protocols (Groth16, PlonK, and Fflonk) to perform verification. The process begins with the energy smart contract recording the start time when requesting energy usage from the apartment. After receiving the response, the energy smart contract records the end time, calculates the latency, and stores it for each asset.

4. Results and Analysis

The experiment was configured to run 10,000 sequential verification queries for each protocol. The goal was to measure the end-to-end latency of retrieving energy usage data from the apartment smart contract via the energy smart contract. The three protocols show relatively similar performance; however, Fflonk performed most consistently, followed by Groth16 and then PlonK. Table 1 summarises the key performance metrics across the three protocols.

To confirm that these differences are statistically significant rather than artefacts of sampling variation, we analysed the full set of 10,000 queries per protocol. A one-way ANOVA shows a highly significant effect of protocol on latency ($p < 0.001$), and pairwise

Welch t-tests confirm that all three protocols differ from one another ($p < 0.001$ in every case) with large effect sizes (Cohen's d between 1.2 and 2.9). With 10,000 observations each, the 95% confidence intervals are narrow and non-overlapping, placing the differences well beyond measurement uncertainty. In practical terms, however, the absolute gaps remain on the order of 2 ms, so whether they are material will depend on the latency budget of the target application.

Table 1. Comparative latency performance of the three zk-SNARK protocols over 10,000 verification queries, reporting the mean per-query latency ($\bar{x}$) and its standard deviation (σ), together with the modal (peak) latency and the percentage of queries occurring at that peak.

Metric	PlonK	Groth16	Fflonk
Mean latency $\bar{x}$ (ms)	24.35	22.49	20.54
Std. deviation σ (ms)	1.12	1.90	1.47
Peak latency (ms)	25.00	22.00	21.00
Peak frequency (%)	45.50	28.82	34.73

Fflonk offers several advantages over Groth16 and PlonK. In terms of the SnarkJS protocol implementation, Fflonk avoids the separate phase 1 and phase 2 setup steps and consumes approximately 5% and 30% fewer resources than Groth16 and PlonK, respectively, during verification. However, smart contract deployment using Fflonk requires approximately 90% more resources than Groth16 [11].

The latency distribution in Figure 2 for PlonK shows the majority of queries completing between 23 ms and 25 ms. The highest frequency occurs at 25 ms, representing 45.5% of queries, followed by 24 ms at approximately 33.8%, and 23 ms at around 9.8%. Very few queries exceeded 26 ms. One outlier at 59 ms was excluded.

The Groth16 protocol exhibits a latency distribution centred between 21 ms and 23 ms exluding one outlier at 394 ms. The peak occurs at 22 ms with 28.82% of queries, followed by 21 ms at 21.94%. This represents a mean latency improvement of approximately 2 ms over PlonK.

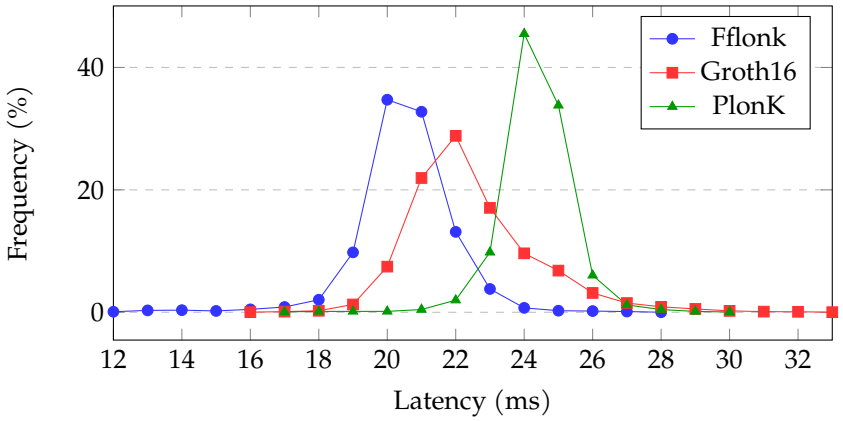


Figure 2. Latency distribution over 10,000 sequential verification queries, showing the frequency (%) of queries at each per-query latency (ms) for the three protocols. Fflonk concentrates at the lowest latencies, Groth16 at intermediate values, and PlonK at the highest, consistent with the mean latencies reported in Table 1.

Fflonk demonstrates the least variation in latency and the fewest outliers among the three protocols, reinforcing its reliability. The distribution is centred between 20 ms and 22 ms, with the peak at 21 ms representing 34.73% of queries, followed by 22 ms at 32.76%.

https://doi.org/10.3390/engproc2026164006

After 22 ms, longer verification times are rare, confirming Fflonk as the most consistent protocol in this evaluation.

5. Conclusions and Future Work

The use of ZKP protocols adds a layer of privacy and security to enable the verification of data between smart contracts without revealing the underlying data. This research found that Fflonk consistently offered the best performance, demonstrating the lowest latency and the most reliable results. This suggests that Fflonk is a suitable choice for use cases that require fast, efficient communication between smart contracts where speed is a critical factor.

Some limitations should be noted. First, experiments were conducted on a single local machine under controlled network conditions, so results may not generalise to distributed deployments with variable network latency. Second, the evaluation used a fixed-complexity Circom circuit (a simple multiplier); performance relationships may shift for larger or more complex circuits with a greater number of constraints. These limitations suggest caution when extrapolating the results directly to production smart city environments.

Next steps in the research are as follows. First, extending the study to connect two smart contracts across different blockchains (e.g., connecting Hyperledger Fabric with Ethereum) would improve the practicality of the approach and bring it closer to the goal of blockchain interoperability in a smart city context [4]. Second, creating an agnostic hub where multiple blockchain types can interconnect would be beneficial in a smart city context [12]. Lastly, measuring latency under circuit scaling conditions and evaluating the performance of ZK rollups represent important next steps for this line of investigation.

Author Contributions: Conceptualization, B.O. and J.N.; methodology, B.O.; software, B.O.; validation, B.O., J.N. and S.S.; formal analysis, B.O. and S.S.; investigation, B.O.; writing—original draft preparation, B.O.; writing—review and editing, J.N. and S.S.; supervision, J.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All the code developed to support this study are publicly available in our GitHub repository at https://github.com/benjaoholeguydeveloper/dissertation (accessed on 28 October 2026).

Acknowledgments: During the preparation of this manuscript, the authors used Google Gemini (3.1) for assistance with writing, formatting, and figure generation. The authors reviewed and edited the output and take full responsibility for the content of the published article.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

EVM	Ethereum Virtual Machine
MPC	Multiparty Computation
SNARK	Succinct Non-interactive Argument of Knowledge
SRS	Structured Reference String
ZKP	Zero-Knowledge Proof
zk-SNARK	Zero-Knowledge Succinct Non-interactive Argument of Knowledge

References

1. Giffinger, R. Smart City Concepts: Chances and Risks of Energy Efficient Urban Development. In *Smart Cities, Green Technologies, and Intelligent Transport Systems*; Communications in Computer and Information Science; Springer International Publishing: Cham, Switzerland, 2015; Volume 579, pp. 3–16. [CrossRef]
2. Zhang, K.; Ni, J.; Yang, K.; Liang, X.; Ren, J.; Shen, X.S. Security and privacy in smart city applications: Challenges and solutions. *IEEE Commun. Mag.* **2017**, *55*, 122–129. [CrossRef]
3. Rao, P.M.; Deebak, B.D. Security and privacy issues in smart cities/industries: Technologies, applications, and challenges. *J. Ambient Intell. Humaniz. Comput.* **2023**, *14*, 10517–10553. [CrossRef]
4. Martinez, S.; Ameigenda, A.; de Barros, B.; Llambías, G.; González, L.; Ruggia, R. Leveraging Zero-Knowledge Proofs for Blockchain Interoperability: Experiences with Ethereum and Hyperledger Fabric. In *Proceedings of the 2024 L Latin American Computer Conference (CLEI)*; IEEE: New York, NY, USA, 2024; pp. 1–10. [CrossRef]
5. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 28 October 2026).
6. Amet, M.; M, D.; Srivastava, G.; Crichigno, J. A Cross-Chain Interoperability Architecture for Smart City Environments. In *Proceedings of the 2022 IEEE Globecom Workshops (GC Wkshps)*; IEEE: New York, NY, USA, 2022; pp. 263–268. [CrossRef]
7. Fiege, U.; Fiat, A.; Shamir, A. Zero knowledge proofs of identity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, New York, NY, USA, 1987*; STOC '87; Association for Computing Machinery: New York, NY, USA, 1987; pp. 210–217. [CrossRef]
8. Groth, J. On the size of pairing-based non-interactive arguments. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 305–326. [CrossRef] [PubMed]
9. van der Velden, T. Zero-Knowledge Proofs on Hyperledger Fabric: Performance Impact of Privacy-Preserving Authentication. Master's Thesis, University of Amsterdam, Amsterdam, The Netherlands, 2024.
10. Kang, H.; Dai, T.; Jean-Louis, N.; Tao, S.; Gu, X. FabZK: Supporting Privacy-Preserving, Auditable Smart Contracts in Hyperledger Fabric. In *Proceedings of the 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*; IEEE: New York, NY, USA, 2019; pp. 543–555. [CrossRef]
11. Guo, H.; Feng, Y.; Wu, C.; Li, Z.; Xu, J. Benchmarking ZK-Friendly Hash Functions and SNARK Proving Systems for EVM-compatible Blockchains. *arXiv* **2024**, arXiv:2409.01976. [CrossRef]
12. Biswas, S.; Yao, Z.; Yan, L.; Alqhatani, A.; Bairagi, A.K.; Asiri, F.; Masud, M. Interoperability benefits and challenges in smart city services: Blockchain as a solution. *Electronics* **2023**, *12*, 1036. [CrossRef]
13. Gabizon, A.; Williamson, Z.J.; Ciobotaru, O. PlonK: Permutations over Lagrange-Bases for Oecumenical Noninteractive Arguments of Knowledge. Cryptology ePrint Archive, Paper 2019/953. 2019. Available online: https://eprint.iacr.org/2019/953 (accessed on 28 October 2026).
14. Gabizon, A.; Williamson, Z.J. Fflonk: A Fast-Fourier Inspired Verifier Efficient Version of PlonK. Cryptology ePrint Archive, Paper 2021/1167. 2021. Available online: https://eprint.iacr.org/2021/1167 (accessed on 28 October 2026).
15. Sun, N.; Zhang, Y.; Liu, Y. A privacy-preserving KYC-compliant identity scheme for accounts on all public blockchains. *Sustainability* **2022**, *14*, 14584. [CrossRef]
16. Ambrona, M.; Schmitt, A.L.; Toledo, R.R.; Willems, D. New Optimization Techniques for PlonK's Arithmetization. Cryptology ePrint Archive, Paper 2022/462. 2022. Available online: https://eprint.iacr.org/2022/462 (accessed on 28 October 2026).
17. Zhou, L.; Diro, A.; Saini, A.; Kaisar, S.; Hiep, P.C. Leveraging zero knowledge proofs for blockchain-based identity sharing: A survey of advancements, challenges and opportunities. *J. Inf. Secur. Appl.* **2024**, *80*, 103678. [CrossRef]
18. Ernstberger, J.; Chaliasos, S.; Zhou, L.; Jovanovic, P.; Gervais, A. Do You Need a Zero Knowledge Proof? Cryptology ePrint Archive, Report 2024/050. 2024. Available online: https://ia.cr/2024/050 (accessed on 28 October 2026).
19. Fiat, A.; Shamir, A. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Proceedings of the Advances in Cryptology—CRYPTO '86*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1987; Volume 263, pp. 186–194. [CrossRef] [PubMed]
20. Hasan, J. Overview and applications of zero knowledge proof (ZKP). *Int. J. Comput. Sci. Netw.* **2019**, *8*, 436–440.
21. Nikolaenko, V.; Ragsdale, S.; Bonneau, J.; Boneh, D. Powers-of-tau to the people: Decentralizing setup ceremonies. In *Proceedings of the International Conference on Practice and Theory in Public Key Cryptography*; Pöpper, C., Batina, L., Eds.; Springer: Berlin/Heidelberg, Germany, 2024; pp. 105–134. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.