Proceedings of CONF-CDS 2026 Symposium: Data-Centric AI Security: Securing Models, Learning Agents, and Autonomous Systems
DOI: 10.54254/2755-2721/2026.CH34915

# *A Survey on Zero-Knowledge Proofs: Trade-Offs and Application-Oriented Adaptation*

**Minhui Le**

*School of Mathematics and Information Science, Guangzhou University, Guangzhou, China*
*32315300035@e.gzhu.edu.cn*

***Abstract.*** The increasing demand for verifiable computation in privacy-sensitive distributed systems has driven the widespread adoption of Zero-Knowledge Proofs (ZKPs). However, the various kinds of current ZKP frameworks—which include zk-SNARKs, zk-STARKs, Bulletproofs, and folding-based systems—introduce complex trade-offs across proof size, prover cost, and trust assumptions, making system selection challenging in actual practice. This paper presents a systematic, application-oriented survey that connects ZKP design choices with real-world deployment constraints. It provides a comparative analysis of major constructions to evaluate their performance and security properties. Furthermore, these trade-offs are mapped to representative application scenarios, including Layer 1/Layer 2 blockchain scaling, Decentralized Identity (DID), and Verifiable Machine Learning (zkML), explaining how different systems are selected based on application-specific requirements. In addition, the paper discusses emerging paradigms such as hardware acceleration, binary field optimizations, and lookup-based zkVMs, which aim to address the prover bottleneck. Overall, this survey provides a structured understanding of the strengths and limitations of existing ZKP systems and offers insights for the design of scalable and privacy-preserving infrastructures.

***Keywords:*** Zero-Knowledge Proofs, Verifiable Computation, Privacy-Preserving Systems, Blockchain

## 1. Introduction

In the rapid development of blockchain, artificial intelligence and distributed systems, there is a contradiction between data openness and confidentiality of information. Current computer architectures should find a balance between computational accuracy and confidential data protection. However, the existing approaches are not adequate to handle the problems of efficiency, security and privacy. Zero-Knowledge Proofs (ZKPs), introduced by Goldwasser, Micali and Rackoff in 1985, offer an important method that enables a prover to prove the truth of a statement to a verifierwithout revealing any private information [1]. Recently, practical systems such as zk-SNARKs, zk-STARKs and Bulletproofs have made it easier to use ZKP techniques to solve practical problems [2, 3].

Although some advances have been made, current Zero-Knowledge Proof (ZKP) systems show great differences in terms of performance, trust assumptions and security guarantees. These

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.CH34915

differences lead to various trade-offs such as proof size, computational cost and setup requirements. However, the existing comparative studies do not provide a common system to compare these trade-offs systematically, thus offering little help in choosing suitable ZKP constructions for practical applications [4, 5].

To solve this problem, this paper carries out a comprehensive survey on ZKP systems from both the viewpoint of trade-off and application. Particularly, this paper puts forward a unified classification system for ZKP systems and compares the main constructions, which associates their trade-offs with common application fields. By this comparison, the objective is to explain the advantages and disadvantages of ZKP systems and provide suggestions for the development of more efficient and secure privacy-preserving applications.

## 2. Background

### 2.1. Formal foundation of Zero-Knowledge Proofs

A ZKP is a protocol where a prover P can convince a verifier V that a specific statement holds true, without revealing information about the witness. In formal terms, given an NP language L and a public statement $x \in L$, the prover shows possession of a private witness w making the relation $R(x, w) = 1$ verifiable in polynomial time [1]. Rather than just checking answers, ZKPs act as the mathematical anchor for trustless environments. Researchers have demonstrated that standard cryptographic assumptions any NP language admits a ZKP, making them incredibly versatile for secure verification [2, 6].

### 2.2. Essential security properties

Any valid ZKP system must fulfill three primary conditions [4]. First, Completeness means that an honest prover will always successfully convince an honest verifier. Second, Soundness dictates that a malicious actor cannot fake a proof for a false statement, except with negligible probability. Third, *Zero-Knowledge* guarantees that the verifier learns nothing except the statement's validity. Cryptographers usually prove this last property by constructing a polynomial-time simulator that generates transcripts indistinguishable from real executions without accessing the witness [6]. Together, these three rules form the security boundary for all practical ZKP deployments.

### 2.3. Motivation for system diversity

Theoretical properties such as completeness, soundness and zero-knowledge give a clear mathematical foundation. However, applying these protocols in practice reveals several engineering difficulties. The different environments have various objectives and thus do not agree on some aspects. For instance, an on-chain smart contract is limited by its computational resources, so the final proof must be very small to reduce the gas cost. On the other hand, generating client-side proofs on a mobile phone has another restriction: the prover cannot use too much memory or else the device may crash. Moreover, some large-scale enterprise networks may consider complete transparency important to avoid the risks caused by a trusted setup. It is mathematically impossible for one method to produce the shortest proofs, minimize the prover's consumption of resources and require no trusted setup at the same time. This leads to the present division in the Zero-Knowledge Proofs (ZKP) community. The wide range of implementation restrictions affects the development of these cryptographic systemshistorically.

DOI: 10.54254/2755-2721/2026.CH34915

## 3. Evolution of ZKP systems

The history of ZKP development represents a constant push-and-pull between lowering communication costs and removing trust assumptions. The shift from multi-round interactive proofs to non-interactive succinct arguments (SNARKs) occurred thanks to the Fiat-Shamir heuristic applied in the random oracle model [1, 7]. Highly optimized SNARKs, for example, the Groth16 construction, reduced proof sizes to only a few hundred bytes, but they required an unacceptable "trusted setup" stage [8].

To eliminate this trusted setup assumption, cryptographers have developed trustless methods. STARKs applied hash-based commitments through the FRI procedure, while Bulletproofs used inner-product arguments to reduce the necessity of initial trust [9, 10]. The latest structural transformation involves combining schemes such as Nova [11]. Unlike the conventional recursive verification, combining schemes add up the computational steps gradually. This approach significantly reduces the burden on the prover during the long execution process. From this change, it is clear that protocol designs are gradually advancing from centralized trust toward setups that are scalable, trustless, and computationally feasible.

## 4. Comparative analysis: trade-offs in ZKP frameworks

Designing a practical ZKP system forces engineers to choose among fast verification, cheap proof generation, and zero trust requirements. No single protocol can maximize all variables at once [7]. This section dissects zk-SNARKs, zk-STARKs, Bulletproofs, and folding methodologies across several metrics: proof size, prover time, verifier time, setup type, quantum resistance, and recursion capability.

### 4.1. The role of underlying mathematical primitives

The performance characteristics of a ZKP system depend directly on its mathematical foundation. *Pairing-based SNARKs* like PLONK and Groth16 use elliptic curve pairings combined with KZG polynomial commitments [8, 12]. This math yields constant-time verification and incredibly small proofs. The downside involves a mandatory trusted setup and a vulnerability to future quantum computers.

In contrast, *Hash-based STARKs* discard pairings entirely. By combining collision-resistant hashing with Reed-Solomon proximity testing, STARKs secure post-quantum resistance and total transparency [9]. The cost paid for this security is a polylogarithmic proof size—which grows as calculations get more complex—and steep memory requirements during proof generation.

### 4.2. Deep dive into asymptotic performance and trust

Table 1 outlines the mathematical bounds of these major ZKP frameworks. *zk-SNARKs* rule when verifier speed matters most, as Groth16 proofs stay at O(1) size regardless of how massive the circuit becomes. *zk-STARKs* handle massive computational scale much better; even though their proofs are larger ($O(\log^2 N)$), their prover times scale far better for deep circuits than pairing-based equivalents. *Bulletproofs* strike a compromise by securing transparency and O(log N) sizes, but their O(N) verification time limits their practical use to small-scale circuits like range proofs [10]. Finally, *Folding Schemes* (like Nova) shift the paradigm by achieving an O(1) marginal cost for the prover per computational step [11]. Accumulating traces rather than generating full proofs at every step

DOI: 10.54254/2755-2721/2026.CH34915

saves massive amounts of time during iterative workloads, although a final SNARK wrapper is still usually needed to compress the result.

These distinct architectural variations and their performance boundaries are systematically organized in Table 1. The data highlights a sharp polarization between the microscopic proof sizes of pairing-based schemes and the zero-trust transparency of hash-based protocols. Ultimately, the compiled metrics indicate that ZKP framework selection is never an absolute technical optimization but rather a calculated compromise tailored to specific environmental constraints.

Table 1. Asymptotic complexities and properties of mainstream ZKP systems

| System Family | Proof Size | Prover Time | Verifier Time | Trusted Setup | Post-Quantum | Recursion / Accumulation |
|---|---|---|---|---|---|---|
| zk-SNARKs (Groth16) [8] | O(1) | O(N log N) | O(1) | App-Specific | No | High Cost (Full Recursion) |
| zk-SNARKs (PLONK) [12] | O(1) | O(N log N) | O(1) | Universal | No | Moderate (Custom Gates) |
| zk-STARKs [9] | $O(\log^2 N)$ | O(N log N) | $O(\log^2 N)$ | Transparent | Yes | Moderate (FRI-based) |
| Bulletproofs [10] | O(log N) | O(N log N) | O(N) | Transparent | No | Limited (Inner Product) |
| Folding Schemes [11] | O(1)* | O(N)* | O(1)* | Trans./Univ. | Depends on commitment | Native (O(1) per step) |

*Note: Complexities for folding schemes represent the marginal cost per step; final verification requires a succinct SNARK wrapper to achieve O(1) size and time.

## 5. Application-driven system selection: solving the adaptation problem

Choosing a ZKP protocol requires strict alignment with the deployment environment's limits [3]. This section evaluates how four major use cases dictate protocol choice.

### 5.1. Blockchain scaling: L1 and L2 design requirements

For the Layer 1 (L1) execution engines such as Ethereum, the block gas limits affect the system design. In this case, Succinct Non-Interactive Arguments of Knowledge (SNARKs) are appropriate. A protocol like Groth16 ensures that the verification of a proof requires few computing resources, which reduces the economic burden on the whole network [8, 13]. However, for the Layer 2 (L2) rollups, the prover has to collect a great amount of user transactions quickly. Usually, STARK-based structures are used in the rollup field because they can handle heavy computations efficiently during the proving process. Besides, avoiding trusted setups can help to reduce the operational difficulties faced by the L2 operators when updating the network [9, 13]. In short, the verification on the chain should be concise, while the off-chain aggregation needs efficient provers.

### 5.2. Privacy-preserving identity and client-side computation

Decentralized Identity (DID) protocols make mobile devices produce proofs locally. The two restrictions in this case are the insufficient hardware memory and the complete refusal to depend on any central authority [3]. Bulletproofs meet the requirements of DID without any difficulty. The absence of a trusted setup ensures the real user's sovereignty, and the small proof size of O(log N) is

DOI: 10.54254/2755-2721/2026.CH34915

convenient to transmit over bad networks. Although the verification time increases linearly (O(N)), typical identity checks usually involve simple circuits, making the slowdown negligible. In mobile environments, it is always more important to eliminate trust and reduce the memory usage than to have fast verification [10].

### 5.3. Verifiable machine learning (zkML) and the scale problem

When examining the operation of a neural network (zkML), the number of arithmetic constraints increases greatly. Ordinary SNARK provers exhaust all available RAM when dealing with such circuits. The folding methods (for example, Nova) overcome this memory problem [11]. By processing each layer of the machine learning model at a time, the proof state is updated step by step. The memory usage of the prover remains constant irrespective of the depth of the neural network. For massive computations, the disadvantage of a larger final proof can be ignored given the benefit of lower memory consumption.

## 6. Challenges and open problems

Cryptographers are currently racing to solve the "prover gap"—the massive time difference between running a native program and generating a proof for it. The industry is testing several new paradigms to shrink this delay, mapped out in Table 2.

### 6.1. The prover bottleneck and hardware-software co-design

Generating proofs requires heavy Multi-Scalar Multiplication (MSM) and Number Theoretic Transform (NTT) mathematics [7]. To speed this up, companies are actively moving *Hardware Acceleration* from academic papers to physical FPGA and ASIC clusters [14].

On the software side, developers are restructuring the math itself. *Binary Field ZKPs* (like the Binius protocol) abandon large prime fields [15]. Running proofs over towers of small binary fields aggressively cuts down both memory limits and compute time, creating algorithms that naturally fit physical hardware constraints.

### 6.2. The zkVM revolution: developer experience and lookup arguments

Writing custom arithmetic circuits needs special cryptographic knowledge which prevents widespread use. Lookup-based zkVMs (such as Jolt and Lasso) are trying to eliminate this obstacle [16]. Instead of converting virtual machine instructions into complicated mathematical conditions, these zkVMs transform instructions into pre-made lookup tables directly. This new structure greatly improves the speed of the prover and enables ordinary software engineers to develop zero-knowledge applications by using languages like Rust or C++.

In order to understand how these new architectural changes will bridge the prover gap, the main technical points and the corresponding design compromises are arranged in Table 2. These approaches show a clear difference between the expansion of physical hardware and the re-design of underlying algebraic systems. It is still important to handle the performance issues to transform the ZKP frameworks into practical digital infrastructures.

DOI: 10.54254/2755-2721/2026.CH34915

Table 2. Summary of emerging ZKP paradigms and associated trade-offs

| Frontier Direction | Key Representative (s) | Primary Breakthrough | Core Trade-off | Primary Bottleneck | Future Maturity |
|---|---|---|---|---|---|
| Hardware Acceleration | FPGA/ASIC Provers [14] | Parallelizing MSM & NTT | High R&D cost vs. Peak performance | Hardware bandwidth / Memory wall | Widely adopted (industry-driven) |
| Binary Field ZKPs | Binius [15] | Proving over towers of small fields | Speed vs. Arithmetic complexity | Proof size / Network bandwidth | Actively researched |
| Advanced Folding | SuperNova [17] | Support for non-uniform computation | Scalability vs. Accumulator complexity | Constraint system complexity | Actively researched |
| Lookup-based zkVMs | Jolt / Lasso [16] | Eliminating constraints via lookups | Prover speed vs. Pre-computed table size | Compiler abstraction gap | Early-stage adoption |

## 7. Conclusion

This research examines in detail the architectural features and basic trade-offs in current zero-knowledge proof (ZKP) systems. By comparing some advanced models like zk-SNARKs, zk-STARKs, Bulletproofs and folding structures, the investigation links theoretical cryptography with practical implementation problems. The findings indicate that there is no one method which can satisfy all the needs for different systems. For Layer 1 verification, pairing-based protocols are still applied for small size proofs, whereas transparent hash-based architectures are effective for handling large transactions on Layer 2. However, in client-side identity applications, low memory consumption is considered more important than other metrics, whereas in complex machine learning processes, incremental folding techniques are required to overcome hardware limitations.

Although these conclusions give useful design ideas, further research is necessary due to the constraints of this study. As it only provides a brief overview of a dynamic field, future work should concentrate on more efficient primitives. Additionally, this analysis mainly focuses on qualitative asymptotic bounds rather than empirical execution data. In the future, it would be beneficial to enhance this method by incorporating quantitative performance measurements under the same hardware conditions.

In the upcoming years, the significant progress in zero-knowledge infrastructure will concentrate on three main areas. First, standard high-level compilers and lookup-based ZKVM interfaces should be developed to facilitate the understanding of circuit design by general software engineers. Second, common software packages should be designed for various FPGA and ASIC architectures in order to ensure portability of hardware accelerators without frequent program rewriting. Finally, as zero-knowledge proofs are extended to support large-scale decentralized systems, it is crucial to verify the accuracy and security of circuits at the compiler level. Emphasizing these aspects will accelerate the transition of zero-knowledge proofs into a widely accepted foundation technology for secure digital networks.

## References

[1] Goldwasser, S., Micali, S., & Rackoff, C. (2019) The knowledge complexity of interactive proof-systems. In Providing sound foundations for cryptography: On the work of shafi goldwasser and silvio micali, 203-225.

DOI: 10.54254/2755-2721/2026.CH34915

[2] Goldreich, O., Micali, S., & Wigderson, A. (1991) Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. Journal of the ACM (JACM), 38(3): 690-728.
[3] Lavin R, Liu X, Mohanty H, et al. (2024) A survey on the applications of zero-knowledge proofs. arXiv preprint arXiv: 2408.00243.
[4] Blum, M., Feldman, P., & Micali, S. (1988) Non-interactive zero-knowledge and its applications. In Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC), 103-112.
[5] Sheybani, N., Ahmed, A., Kinsy, M., & Koushanfar, F. (2025) Zero-knowledge proof frameworks: A systematic survey. arXiv preprint arXiv: 2502.07063.
[6] Goldreich, O. (2001) Foundations of cryptography: volume 1, basic tools. Cambridge University Press.
[7] Thaler, J. (2022) Proofs, arguments, and zero-knowledge, foundations and trends in privacy and security, 4(2-4): 117-660.
[8] Groth, J. (2016) On the size of pairing-based non-interactive arguments. In Annual international conference on the theory and applications of cryptographic techniques, 305-326.
[9] Ben-Sasson, E., Bentov, I., Horesh, Y., & Riabzev, M. (2018) Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive.
[10] Bünz, B., Bootle, J., Boneh, D., et al. (2018) Bulletproofs: short proofs for confidential transactions and more. In 2018 IEEE symposium on security and privacy (SP), 315-334.
[11] Kothapalli, A., Setty, S., & Tzialla, I. (2022) Nova: recursive zero-knowledge arguments from folding schemes. In Annual International Cryptology Conference, 359-388.
[12] Gennaro, R., Gentry, C., Parno, B., & Raykova, M. (2013) Quadratic span programs and succinct NIZKs without PCPs. In Annual international conference on the theory and applications of cryptographic techniques, 626-645.
[13] Sun, X., Yu, F. R., Zhang, P., et al. (2021) A survey on zero-knowledge proof in blockchain. IEEE network, 35(4): 198-205.
[14] Qiu, P., Wu, G., Chu, T., et al. (2024) MSMAC: Accelerating multi-scalar multiplication for zero-knowledge proof. In Proceedings of the 61st ACM/IEEE Design Automation Conference, 1-6.
[15] Diamond, B. E., & Posen, J. (2025) Succinct arguments over towers of binary fields. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, 93-122.
[16] Arun, A., Setty, S., & Thaler, J. (2024) Jolt: Snarks for virtual machines via lookups. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, 3-33.
[17] Kothapalli, A., & Setty, S. (2022) SuperNova: Proving universal machine executions without universal circuits. Cryptology ePrint Archive.