

Zero-Knowledge Proofs in Distributed Systems: zk-Snarks and zk-Starks For Privacy-Preserving and Scalable Blockchain Applications

Tintu George

Assistant Professor, Department of BCA AI, Sri Ramakrishna College of Arts & Science, Coimbatore, India

Article information

Received: 8th June 2026

Received in revised form: 10th July 2026

Accepted: 11th August 2026

Available online: 12th September 2026

Volume: 1

Issue: 3

DOI: <https://doi.org/10.5281/zenodo.22702428>

Abstract

A zero-knowledge proof lets one party convince another that a statement is true while revealing nothing beyond the fact of its truth. The idea, introduced in cryptographic theory in the 1980s, sounded paradoxical and remained largely academic for thirty years. It has since become one of the most consequential tools in distributed systems, because a blockchain is exactly the setting where the paradox pays off: every participant must be convinced that a transaction is valid, yet privacy demands that the details stay hidden, and scalability demands that verifying a computation cost far less than performing it. Modern succinct proof systems deliver both. This paper explains zero-knowledge proofs for a systems audience and contrasts the two families now driving deployment. zk-SNARKs produce extremely small proofs that verify in milliseconds but have historically required a trusted setup whose compromise would be catastrophic. zk-STARKs eliminate that trusted setup and resist quantum attack, at the cost of substantially larger proofs. We cover the properties that define a zero-knowledge argument, the path from interactive proofs to non-interactive succinct ones, the concrete trade-offs between the two families, and the two applications reshaping blockchains: confidential transactions that hide amounts and parties, and validity rollups that move computation off-chain while proving its correctness on-chain.

Keywords:- Zero-Knowledge Proofs, ZK-SNARK, ZK-STARK, Blockchain, Privacy, Verifiable Computation, Succinct Arguments, Trusted Setup, Rollups, Post-Quantum Cryptography.

I. INTRODUCTION

Among the stranger results in theoretical computer science is the proof that you can convince someone a statement is true without telling them anything about why. Goldwasser, Micali, and Rackoff formalised this in 1985 as the zero-knowledge proof, a protocol satisfying three properties at once: an honest prover can always convince the verifier of a true statement, no prover can convince the verifier of a false one except with negligible probability, and the verifier learns nothing from the interaction beyond the statement's truth [1], [2]. For decades this was a beautiful idea with few practical consumers.

Blockchains changed that. A public ledger is a distributed system with two requirements in tension. Every node must independently verify that each transaction obeys the rules, which seems to demand that every node see every detail; yet users want privacy, and the network wants to avoid the cost of having every node re-execute every computation [8], [13]. Zero-knowledge proofs resolve both tensions with one mechanism. A proof can establish that a transaction is valid without revealing its amounts or participants, and a proof can establish that a large off-chain computation was performed correctly while costing the verifier almost nothing to check [4], [14].

The same primitive thus serves privacy and scalability, which is why it has moved from theory to the centre of blockchain engineering.

This paper introduces zero-knowledge proofs for systems engineers and compares the two practical families in deployment. Section II states the defining properties and Figure 1 depicts the prover-verifier relationship. Section III traces the route from interactive proofs to the succinct, non-interactive arguments that real systems use. Sections IV and V contrast zk-SNARKs and zk-STARKs and the trade-offs that distinguish them, and Sections VI and VII examine the privacy and scaling applications and the open challenges that remain.

II. WHAT A ZERO-KNOWLEDGE PROOF GUARANTEES

A zero-knowledge argument is defined by three properties, and grasping them is enough to understand why the technology is useful even without its mathematics [1]. Completeness says that if the statement is true and both parties follow the protocol, the verifier will be convinced. Soundness says that if the statement is false, no cheating prover can convince the verifier except with vanishingly small probability, so a proof that passes can be trusted. Zero-knowledge, the third and most surprising property, says that the verifier learns nothing from the proof except that the statement is true; in particular it learns nothing about the prover's secret, the witness, that makes the statement true [1], [2]. Figure 1 shows the setting: a prover holding a secret witness for a public statement produces a proof, and a verifier checks it against the public statement alone. The classic illustration is proving you know a password without sending it, but the general form, proving you correctly performed an arbitrary computation without revealing its private inputs, is what makes the primitive so powerful in distributed systems [15], [17].

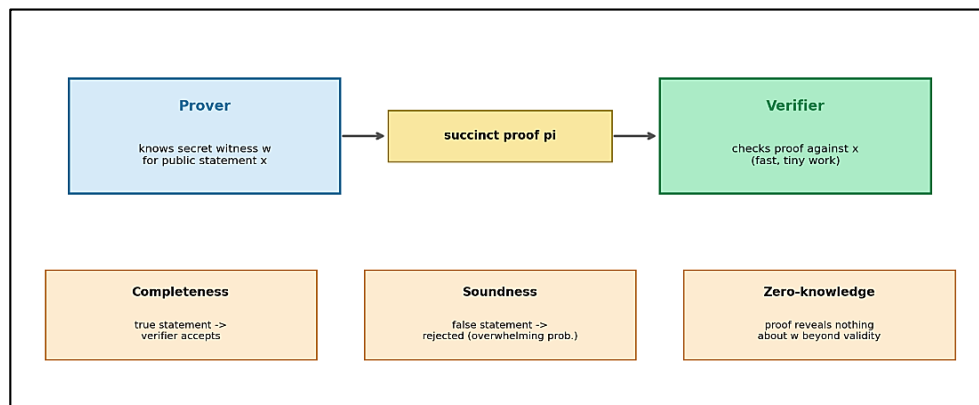


Fig 1: A zero-knowledge argument. The prover convinces the verifier that it holds a valid witness for a public statement; completeness, soundness, and the zero-knowledge property together let the verifier trust the claim while learning nothing about the secret.

III. FROM INTERACTIVE PROOFS TO SUCCINCT ARGUMENTS

A. Removing the Interaction

The original zero-knowledge proofs were interactive: prover and verifier exchanged several rounds of challenges and responses [1], [2]. That is unworkable on a blockchain, where a proof must be posted once and checked later by many parties who cannot interact with the prover. The Fiat-Shamir heuristic and the subsequent theory of non-interactive proofs solved this by letting the prover generate the verifier's challenges itself from a cryptographic hash, collapsing the dialogue into a single message that anyone can verify [5], [15]. Non-interactivity is the first property a deployable system needs.

B. Making the Proof Succinct

The second property is succinctness: the proof must be small and fast to verify, ideally far smaller than the computation it attests to, so that verifying is much cheaper than re-executing. This is the property captured by the term SNARK, a succinct non-interactive argument of knowledge [3]. A line of work running from Pinocchio [9] through quadratic span programs [5] to the highly efficient Groth16 construction [6] made such arguments practical, reducing a proof of an arbitrary computation to a handful of group elements that a verifier checks with a constant, tiny amount of work regardless of how large the original computation was [17]. Verifiable computation in this sense, proving that a program ran correctly on hidden inputs, is the engine beneath every application in this paper [4], [16].

IV. zk-SNARKs

zk-SNARKs are the family that brought zero-knowledge proofs into production, and their defining strength is extreme succinctness. A Groth16-class proof is only a few hundred bytes and verifies in milliseconds, independent of the size of the computation proved, which is precisely what a blockchain needs when thousands of nodes must each check the proof cheaply [4], [6]. This efficiency is what made the first privacy-preserving cryptocurrency, which hides transaction amounts and participants behind zk-SNARK proofs of validity, technically feasible [8].

The strength comes with a serious caveat: most efficient zk-SNARKs require a trusted setup. Before the system can be used, a set of public parameters must be generated from secret randomness, and if anyone learns that randomness, often called the toxic waste, they can forge proofs of false statements undetectably [4], [11]. The security of the entire system thus rests on that secret having been destroyed. The practical mitigation is a multi-party computation ceremony in which many participants jointly generate the parameters such that the setup is secure as long as even one participant was honest and discarded their share [11]. Newer constructions reduce the burden further by making the setup universal and updatable, so a single ceremony serves many programs rather than requiring a fresh one per circuit [12]. Still, the trusted setup remains the principal structural weakness of the SNARK approach.

V. zk-STARKs AND THE TRADE-OFF

zk-STARKs were designed to remove exactly that weakness. They are transparent, requiring no trusted setup at all, because they rely only on cryptographic hash functions and public randomness rather than secret parameters, which eliminates the toxic-waste risk entirely [7]. A second consequence of building on hash functions is post-quantum security: unlike pairing-based zk-SNARKs, whose hardness assumptions a large quantum computer would break, STARKs rest on assumptions believed to survive quantum attack [7]. They also scale well for very large computations, with prover overhead growing only quasi-linearly in the size of the statement. The cost is proof size. A STARK proof is typically tens to hundreds of kilobytes, orders of magnitude larger than a SNARK proof, and verification, while still far cheaper than re-execution, is correspondingly heavier [7]. On a blockchain, where every byte stored and every verification performed has a direct cost, this is a real disadvantage. The comparison therefore has no universal winner, and the choice is an engineering trade-off, illustrated in Figure 2 and summarised in Table 1: SNARKs minimise on-chain footprint at the price of a trusted setup and quantum vulnerability, while STARKs accept larger proofs in exchange for transparency and post-quantum security [6], [7], [12].

Table 1. zk-SNARKs Versus zk-STARKs

Property	zk-SNARK (Groth16-class)	zk-STARK
Proof size	Very small (hundreds of bytes)	Large (tens-hundreds of KB)
Verification cost	Very low (ms, near-constant)	Higher but still sublinear
Trusted setup	Required (per-circuit or universal)	None (transparent)
Post-quantum secure	No (pairing-based assumptions)	Yes (hash-based)
Cryptographic basis	Elliptic-curve pairings	Collision-resistant hashes

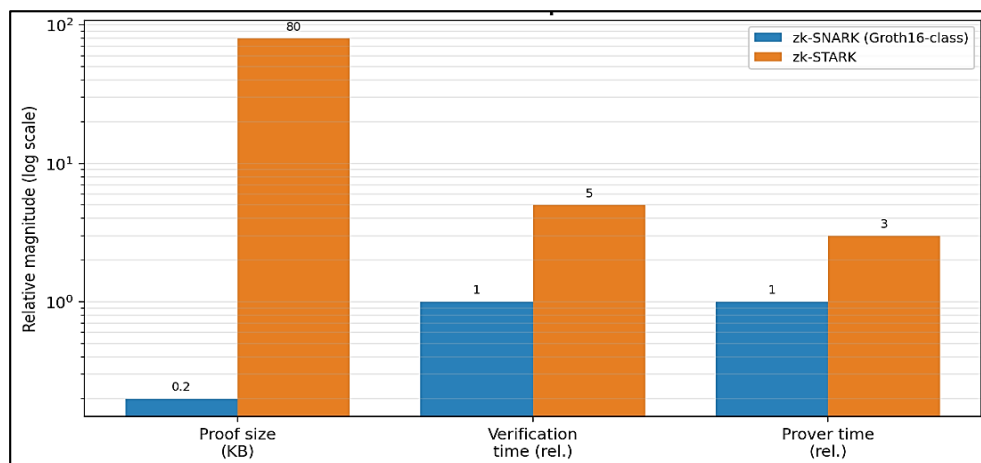


Fig 2: Representative trade-offs between the two families on a log scale. zk-SNARKs win decisively on proof size and verification cost; zk-STARKs pay for that with much larger proofs but need no trusted setup and resist quantum attack.

VI. APPLICATIONS IN DISTRIBUTED LEDGERS

A. Confidential Transactions

The first application is privacy. On a transparent ledger every transaction is public, which is unacceptable for many real uses. Zero-knowledge proofs let a user prove that a transaction is valid, that the inputs exist, that they are authorised to spend, and that inputs balance outputs, without revealing the amounts or the parties involved [8]. The pioneering deployment used zk-SNARKs to realise fully shielded payments on a public blockchain, and related techniques such as Bulletproofs provide compact range proofs that hide transaction amounts without any trusted setup [8], [10]. The general pattern is to replace public data with a proof that the hidden data satisfies the rules.

B. Validity Rollups for Scaling

The second application, and the one currently driving the most engineering investment, is scaling. A blockchain's throughput is limited because every node re-executes every transaction. A validity rollup, often called a zk-rollup, breaks this limit by executing a large batch of transactions off-chain and posting to the main chain only a succinct proof that the entire batch was processed correctly, together with the resulting state [14]. Every node then verifies one small proof instead of re-executing thousands of transactions, so the chain inherits the security of full verification at a fraction of the cost [4], [14]. This is verifiable computation applied at scale, and it is the leading approach to scaling smart contract platforms without weakening their security guarantees [13], [14]. Tooling that compiles high-level programs into provable circuits has made building such systems accessible to ordinary developers rather than only cryptographers [16].

VII. INSIDE A SNARK: FROM PROGRAMS TO POLYNOMIALS

It is worth opening the box, because the internal pipeline explains both the power and the fragility of these systems. A proof system cannot reason about a program directly; the computation must first be expressed as an arithmetic circuit, a network of addition and multiplication gates over a finite field that computes the same result [9], [17]. The circuit is then written as a constraint system, most commonly a rank-one constraint system, in which each gate becomes an equation that a correct execution must satisfy, so that proving the computation was done correctly reduces to proving that a set of assignments satisfies all the constraints [5], [9]. The next step is the conceptual heart of succinctness: the satisfaction of all those constraints is encoded into polynomials, and the prover commits to those polynomials so that the verifier can check a few random points instead of every constraint, relying on the fact that distinct low-degree polynomials almost never agree at a random point [5], [20]. Figure 3 traces this pipeline from program to proof. The polynomial-commitment scheme is the component that most distinguishes modern systems from one another, and advances in commitments have driven much of the recent progress [20]. The same pipeline also explains the dominant security risk noted earlier: a mistake in translating a computation into a circuit produces a proof that is perfectly valid for the wrong statement, which is why circuit correctness and auditing matter as much as the cryptography [16].

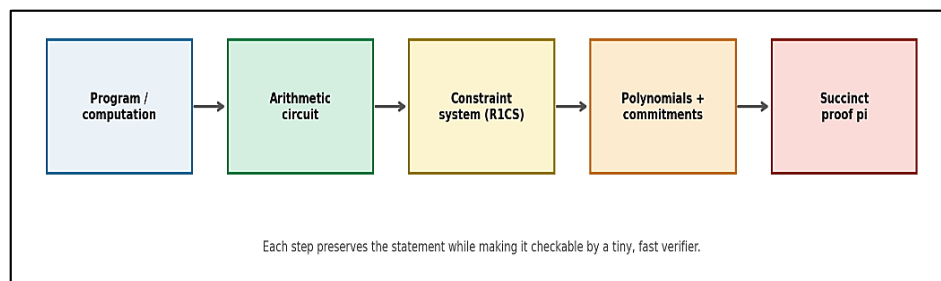


Fig 3: The internal pipeline of a succinct proof. A computation becomes an arithmetic circuit, then a constraint system, then polynomials the prover commits to, letting the verifier check a few random points rather than the whole computation.

VIII. THE MODERN PROOF-SYSTEM LANDSCAPE

The clean two-family split of Sections IV and V is a useful simplification of a landscape that has grown rich. The pairing-based construction that minimises proof size also fixes its parameters to a single circuit, so a change to the program demands a fresh trusted setup [6]. A major line of work removed that rigidity with universal and updatable setups, in which one ceremony produces parameters reusable across many programs, balancing succinctness against operational practicality [12], [21]. A separate family abandons the trusted setup entirely while keeping proofs far smaller than STARKs: inner-product arguments yield short proofs for confidential transactions and range statements with no setup at all, at the cost of verification that grows with the statement [10], [22]. Transparent constructions have also been built from polynomial commitments that need no secret parameters,

narrowing the gap between the SNARK and STARK worlds from the SNARK side [19]. Meanwhile the hash-based, FRI-driven STARK approach continues to anchor the transparent, post-quantum end of the spectrum [7]. The honest summary is a design space governed by a trade-off among three goods, succinctness, transparency, and prover efficiency, in which no single system maximises all three; Table 2 places representative systems within it.

Table 2. Representative Modern Proof Systems

System / family	Setup	Proof size	Post-quantum
Groth16 (pairing-based)	Trusted, per-circuit	Smallest	No
PLONK / Marlin (universal)	Trusted, universal + updatable	Small	No
Bulletproofs (IPA)	Transparent (none)	Small-to-medium	No
Halo-style (recursive)	Transparent (none)	Small	No
STARK (FRI, hash-based)	Transparent (none)	Large	Yes

IX. RECURSION, AGGREGATION, AND THE zkEVM

The application that is currently absorbing the most engineering effort, validity rollups, has pushed two further capabilities to the foreground. The first is recursion: a proof can attest to the validity of another proof, so that a single succinct proof can vouch for an entire chain of computations, and incrementally verifiable computation lets a long-running process maintain one constant-size proof of its whole history [18]. Recursion enables proof aggregation, combining many independent proofs into one that a verifier checks at the cost of a single proof, which is exactly what lets a rollup compress thousands of transactions into one on-chain verification [14], [18]. The second capability is generality. Early systems proved fixed, hand-built circuits; the ambition now is the zkEVM, a proof system for the full instruction set of a general smart-contract virtual machine, so that arbitrary existing contracts can be executed off-chain and proven correct without being rewritten as bespoke circuits [13], [14]. Realising this is formidable, because expressing a complete virtual machine as efficient constraints strains every part of the pipeline, and it is a major reason proving remains computationally expensive and why hardware acceleration of the prover has become an active field [7], [15]. Together, recursion and general-purpose proving are turning zero-knowledge proofs from a tool for specific statements into a general substrate for verifiable computation at scale.

X. CHALLENGES AND CONCLUSION

Significant challenges remain. Proving is computationally expensive: generating a proof can take orders of magnitude longer than performing the computation it attests to, which shifts cost from verifier to prover and motivates ongoing work on faster proving and hardware acceleration [7], [15]. Expressing a computation as the arithmetic circuit these systems require is intricate and error-prone, and a bug in the circuit can silently void the security guarantee, which has made better compilers and formal auditing a priority [16], [17]. For zk-SNARKs the trusted setup remains a structural risk that universal and updatable constructions reduce but do not fully remove, while for zk-STARKs the large proof size is the standing cost [11], [12].

Even so, the trajectory is unmistakable. A primitive that was a theoretical curiosity for three decades is now foundational infrastructure for privacy and scalability in distributed ledgers [1], [8], [14]. The two families embody a clean trade-off, zk-SNARKs minimising on-chain footprint at the price of a trusted setup and quantum exposure, zk-STARKs offering transparency and post-quantum security at the price of proof size, and active research is narrowing the gap from both directions [6], [7], [12]. As proving costs fall and tooling matures, the ability to prove a computation correct without revealing its inputs is becoming a general building block of distributed systems, well beyond the blockchains where it first found a compelling use.

REFERENCES

- [1] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186–208, Feb. 1989.
- [2] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems," *Journal of the ACM*, vol. 38, no. 3, pp. 690–728, Jul. 1991.
- [3] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proc. 3rd Innovations in Theoretical Computer Science Conf. (ITCS)*, Cambridge, MA, USA, 2012, pp. 326–349.
- [4] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von Neumann architecture," in *Proc. 23rd USENIX Security Symposium (USENIX Security)*, San Diego, CA, USA, 2014, pp. 781–796.

- [5] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct NIZKs without PCPs," in *Proc. 32nd Annual Int. Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, Athens, Greece, 2013, pp. 626–645.
- [6] J. Groth, "On the size of pairing-based non-interactive arguments," in *Proc. 35th Annual Int. Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, Vienna, Austria, 2016, pp. 305–326.
- [7] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," *IACR Cryptology ePrint Archive*, Rep. 2018/046, 2018.
- [8] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from Bitcoin," in *Proc. IEEE Symp. Security and Privacy (S&P)*, San Jose, CA, USA, 2014, pp. 459–474.
- [9] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proc. IEEE Symp. Security and Privacy (S&P)*, Berkeley, CA, USA, 2013, pp. 238–252.
- [10] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *Proc. IEEE Symp. Security and Privacy (S&P)*, San Francisco, CA, USA, 2018, pp. 315–334.
- [11] S. Bowe, A. Gabizon, and I. Miers, "Scalable multi-party computation for zk-SNARK parameters in the random beacon model," *IACR Cryptology ePrint Archive*, Rep. 2017/1050, 2017.
- [12] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge," *IACR Cryptology ePrint Archive*, Rep. 2019/953, 2019.
- [13] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," Ethereum White Paper, 2014.
- [14] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, vol. 10, pp. 93039–93054, 2022.
- [15] J. Thaler, "Proofs, arguments, and zero-knowledge," *Foundations and Trends® in Privacy and Security*, vol. 4, nos. 2–4, pp. 117–660, 2022.
- [16] J. Eberhardt and S. Tai, "ZoKrates—Scalable privacy-preserving off-chain computations," in *Proc. IEEE Int. Conf. Blockchain*, Halifax, NS, Canada, 2018, pp. 1084–1091.
- [17] C. Reitwiessner, "zkSNARKs in a nutshell," Ethereum Foundation Technical Report, 2016.
- [18] S. Bowe, J. Grigg, and D. Hopwood, "Recursive proof composition without a trusted setup," *IACR Cryptology ePrint Archive*, Rep. 2019/1021, 2019.
- [19] B. Bünz, B. Fisch, and A. Szepieniec, "Transparent SNARKs from DARK compilers," in *Proc. 39th Annual Int. Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, Zagreb, Croatia, 2020, pp. 677–706.
- [20] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *Proc. 16th Int. Conf. Theory and Application of Cryptology and Information Security (ASIACRYPT)*, Singapore, 2010, pp. 177–194.
- [21] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward, "Marlin: Preprocessing zkSNARKs with universal and updatable SRS," in *Proc. 39th Annual Int. Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, Zagreb, Croatia, 2020, pp. 738–768.
- [22] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, "Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting," in *Proc. 35th Annual Int. Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, Vienna, Austria, 2016, pp. 327–357.