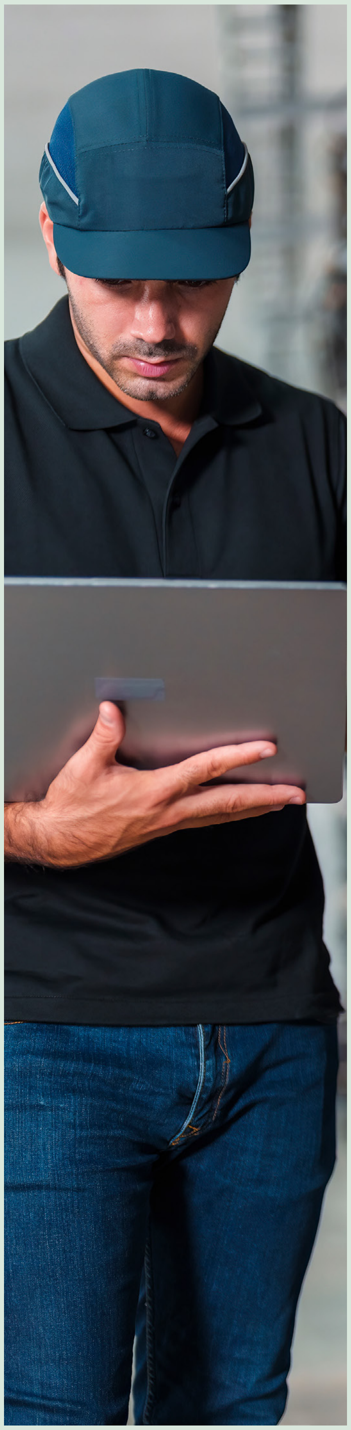
WHITE PAPER



ZERO-KNOWLEDGE PROOFS: ACHIEVING PRIVACY AND SCALABILITY IN BLOCKCHAIN APPLICATIONS

Abstract

Zero-knowledge proofs (ZKPs) are cryptographic protocols that allow a prover to establish the truthfulness of their statement to the verifier without revealing any additional information about the statement.

The security of zero-knowledge proof lies in its properties of soundness and completeness. It must not be possible for a prover to prove an incorrect statement to a verifier and submitted proof must convince verifier about correctness of the statement.

This paper explores zero-knowledge proof (ZKP) protocols as tools to improve privacy and scalability in blockchain applications while maintaining and enhancing trust as well as data integrity. This paper compares the most popular ZKP protocols such as zk-SNARK, zk-STARK, and Bulletproofs. It also provides references to blockchain projects where zk-proofs are being used to improve privacy and scalability.

Infosys®
Navigate your next

Table of the content

Introduction

"Zero-knowledge (ZK) proof systems are an ingenious cryptographic solution to the tension between the ideals of personal privacy and institutional integrity, enforcing the latter in a way that does not compromise the former."

(ref: Scalable, transparent, and post-quantum secure computational integrity, Eli Ben-Sasson et al., 2018)

In 1985, Authors introduced zero-knowledge proof through their paper titled "The Knowledge Complexity of Interactive Proof Systems" (Goldwasser et al., 1985). Authors of the paper demonstrated knowledge of the solution without revealing the solution itself. This ability of demonstrating knowledge of the information, without revealing it, helps Blockchain solutions achieve privacy and scalability.

Blockchain is a shared immutable record of transactions (ledger) distributed across participants of the network. Transactions are validated and grouped into blocks through a consensus mechanism and committed to the ledger. Transactions on the ledger follow a protocol and are facilitated by smart contracts, which are a set of rules embedded in the code that reside on the ledger itself. Any interested party can verify the current state of a pubic blockchain ledger and check the transaction as all details of the transaction (sender, receiver, amount) are public. To some extent, privacy is maintained in public blockchains as it is difficult to map transacting wallet addresses with real-world identities. The transaction amount, however, is disclosed. The throughput on a blockchain network is low as transactions must be validated and consensus must be achieved to write the block on to the ledger. At present, "block time" (time separating 2 blocks) on the Ethereum (ETH) blockchain is around 12 seconds and it takes around 15 minutes to finalize a block. A block is considered finalized when it is not possible to modify the block without burning at least 1/3rd of the total staked Ethereum.

In the past few years, blockchain technology has proved its relevance in almost all industries and verticals. Through participation in a blockchain network, organizations can utilize trustworthy data in a trust-less setup without the need for expensive reconciliation processes. This has created much needed transparency in the system, brought in huge operational efficiencies, and helped create new business models.

The privacy challenge

While more and more businesses are adopting blockchain, they have realized some obvious issues with the transparent nature of blockchain applications. Confidentiality and privacy, in some scenarios, are the key requirements from both business and regulatory perspectives. For example, a manufacturer will want to execute transactions with a dealer in a manner that deal rates are not disclosed to other dealers. Such scenarios underline the need for private transactions. In a traditional set-up, privacy comes at the cost of trade-off between transparency and trust. Enterprises clearly need better ways of establishing the truth without compromising either transparency, trust, or privacy of the information.

Patient medical records are confidential and private as per regulations implemented in many countries. Therefore, hospitals and health organizations may find it difficult to share such information even for research and analysis by machine learning models. Similarly, financial data needs extra protection and care to respect the financial privacy of the individual. Projects like Zcash (privacy protecting digital currency), and Monero (decentralized currency to keep finances confidential) are utilizing zk-proofs to carry out private transactions. RockyBot (an AI trading bot) can present proof of correctness of data analysis without revealing the data.

Zero-knowledge proofs provide authoritative evidence of the correctness of facts without disclosing the facts themselves. Integration of zero-knowledge proofs with blockchain takes care of privacy through cryptographic blinding and ensures data integrity without diluting the transparency and trust provided by a blockchain network. This could help in improving adoption of blockchain solutions and has the potential to open a set of completely new use cases.

The scalability challenge: Throughput and performance

Beside privacy, the industry expects blockchain applications to deliver throughput and scalability that match traditional systems and databases. Payment systems like Visa require 24000 tps while blockchain networks are offering throughput between the range of 1-100 tps. Some blockchain networks such as Hyperledger Fabric claim to offer better throughput of approximately 1500 - 3000 tps in a lab environment. However, such throughput is way behind the industry requirements. With the gaming industry entering Web 3.0 and Metaverse, the calls for better throughput are getting louder. Smart contracts on blockchain are constrained with limited computation and storage power, which in turn restrict compute-intensive use cases from leveraging blockchain technology.

Platforms such as ImmutableX, Starknet, and Polygon zkEVM are effectively utilizing the power of zero-knowledge proofs to improve scalability. Compute-heavy operations are performed off-chain and proof of correctness of the computations is made public on blockchain. The submitted proof is short and verifiable at relatively miniscule computational cost by all interested parties.

Understanding Zero-Knowledge Proof

Zero knowledge proof (ZKP) helps prover in proving the knowledge of a secret. While doing so ZKP does not reveal the secret itself. There are two approaches to construct zero knowledge proofs:

- Interactive approach: In this approach, prover and verifier interact with each other. During these interactions, prover convinces verifier about the knowledge of the secret without revealing the secret itself.
- Non-interactive approach: In this approach, there is no need for prover and verifier to interact with each other. Prover creates the proof off-line and submit it to the verifier. Verifier can verify the proof.

Terminology:

Claim
The statement made by a prover (for example, my age is greater than 18 years)

Prover
Entity (user, device, or application) that will prove the claim

Proof
The cryptographic evidence of correctness of the statement/claim generated by the prover

Witness
The information or fact supporting the claim (for example, date of birth certificate issued by competent authority)

Verifier
Entity (user, device, or application) that is verifying the proof

The Characteristics of the Zero-knowledge proof:

Completeness
When the claim is true, an honest verifier should be convinced by an honest prover

Soundness
If the claim is false, a dishonest prover cannot convince an honest verifier about correctness of the claim (except with negligible probability)

Zero knowledge
The verifier learns nothing from the proof except that the claim is true

Interactive zero-knowledge proofs

In an interactive zero-knowledge proofs system the prover and verifier interact with each other. The verifier presents a challenge, and the prover could respond to the challenge. The response to the challenge can be correct only if the prover has knowledge of the information. Messages are exchanged till the verifier has the answer to the problem and is convinced about the correctness of the solution, implying that the prover has knowledge of the information.

Two balls of different colors and a blind-folded man is a famous example to demonstrate interactive zero-knowledge proof.

Take two identical balls, but of different colors, say black and white. The objective is to prove to a blind-folded person that the two balls are distinguishable. The blind-folded person does not know that the balls are of different colors. He shuffles the balls randomly behind his back and shows you one of the balls. He asks you if he has switched the ball. He repeats this multiple times. If you can answer him correctly each time, he is convinced that balls are indeed distinguishable without knowing the color of the balls.

Using interactive zero-knowledge proof technique, oracles (Middleware for communication between off-chain data providers and blockchains) can help in decentralizing proof-systems off-chain. Multiple oracles can independently carry out proof tests and collaboratively make decisions through consensus.

Interactively proving the knowledge could be less efficient in many use-cases especially when one needs to prove the same thing to multiple verifiers.

Non-interactive zero-knowledge proofs

Non-interactive zero-knowledge (NIZK) proof brings in transferability, meaning that a prover can generate proof independent of a verifier and submit it to multiple verifiers. Each verifier can verify the proof independently and take further action accordingly.

A non-interactive approach is suited for systems where multiple verifiers may want to verify proof. For example, to grant access to websites for adults (age > 18). A user can submit same zk-proof to claim adult status without revealing their age or other personal details to multiple websites.

Today three major non-interactive zero-knowledge proof technologies are being used:

- Zk-SNARK (Zero Knowledge Succinct Non-interactive Argument of Knowledge) – Zk-SNARK adopts non-interactive approach to produce very short proofs (also called as argument of knowledge). It is used by Zcash and Polygon zkEVM
- STARK – Scalable transparent argument of knowledge used by Polygon zkEVM and Starknet
- Bulletproofs - Bulletproofs are short, non-interactive zero knowledge proofs used by Monero, a private, decentralized cryptocurrency

Each of the above-mentioned protocols makes certain assumptions such as the existence of non-collision hash functions. The security of the protocol depends on the assumptions, Violation of those assumptions will break the security of the protocol. For example, to generate random challenges Fiat-Shamir transformation depends on cryptographic hash functions. It assumes that secure hash function will be used in random challenge generators. Schnorr NIZK proof uses Fiat-Shamir transformation. A poor random generator in Fiat-Shamir transformation may compromise the security of the protocol. It is prudent to include some contextual information in proofs (or hashes that are being calculated in the proof) to ensure security and take care of any sort of attacks such as replay attacks. In Schnorr NIZK proof, the hash function uses the user ID of the prover and verifier to avoid replay attacks.

Zk-Succinct Non-interactive Argument of Knowledge (zk-SNARK)

Authors presented zk-SNARK in their paper "Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data" (Nir Bitansky et al., 2019). The zk-SNARK protocol has been added to blockchains like ZCash for shielded or private transactions.

Zk-SNARK protocol helps in defining a function f() which takes two inputs x (a publicly known information) and w (witness or a secret known only to the prover). The solution of the function f() is z (we shall refer to it as proof); z = f(x, w). The prover knows w and generates the proof(z).

The prover provides proof (z) to the verifier. There is no way for the verifier to calculate the witness (w) from the proof (z). At the same time, the verifier is assured that z is calculated from a correct w. The probability of calculating z satisfying f() with incorrect w is negligibly low.

A claim or a statement can be classified as an NP statement. An NP statement is a set of statements which can be easily verified in polynomial time using a deterministic Turing machine. However, it is extremely difficult to generate proof of the statement without knowing the witness (secret). The prover who knows the witness for the NP statement can produce a short proof attesting to the truth of the NP statement. The proof has the properties mentioned below and can be verified by anyone.

Zero-knowledge

It is extremely hard to extract any knowledge from the proof because the proof is indistinguishable from a random set of characters

Succinct

The proof is short and easy to verify. Zk-SNARK is succinct from the verification process perspective. For a given security parameter λ, the proof has the size $O_\lambda(1)$, where $O_\lambda(\cdot)$ is some polynomial in a security parameter λ and verification time, $t = O_\lambda(|f| + |x| + |z|)$

Non-interactive

The proof is sufficient to prove to any number of verifiers independent of the prover (that is, without direct interactions with the prover)

Argument of knowledge

The proof z is referred to as argument of knowledge because z is not a direct proof of knowing the witness. Correct z can be computed when prover knows the correct witness. Probability of computing correct z without knowing the witness is extremely low which makes it extremely difficult and time-consuming to construct fake proofs

Multiple implementations for zk-SNARK such as Pinocchio and PLONK are in use. Snarkjs is a javascript implementation of zk-SNARK using Groth16, PLONK and FFLONK. It is used in IDEN3 which is a blockchain-based identity management solution supporting privacy by design with zk-SNARK.

Diagram 1 depicts the implementation approach for zk-SNARK.

G is a key generator function which takes two inputs - secret (λ) and a program (C, qualification criteria) and generates a pair of keys - prover key (Pk) and verifier key (Vk). The secret to generate the key may be misused later and therefore should be destroyed with the cognizance of all parties involved. Both prover and verifier keys can be made public. Going forward, any prover can use the prover key to generate proof and any verifier can use the verifier key to verify a submitted proof.

Proof generator (P) takes 3 arguments as input - a prover key, a publicly known information, and a witness (secret), to generate a proof. The generated proof can now be submitted to any verifier.

A verification function uses a verification key, and publicly known information (the one used by prover) to verify the submitted proof. The outcome of the verification is either true or false. In the process of verification, no information about the witness is divulged.

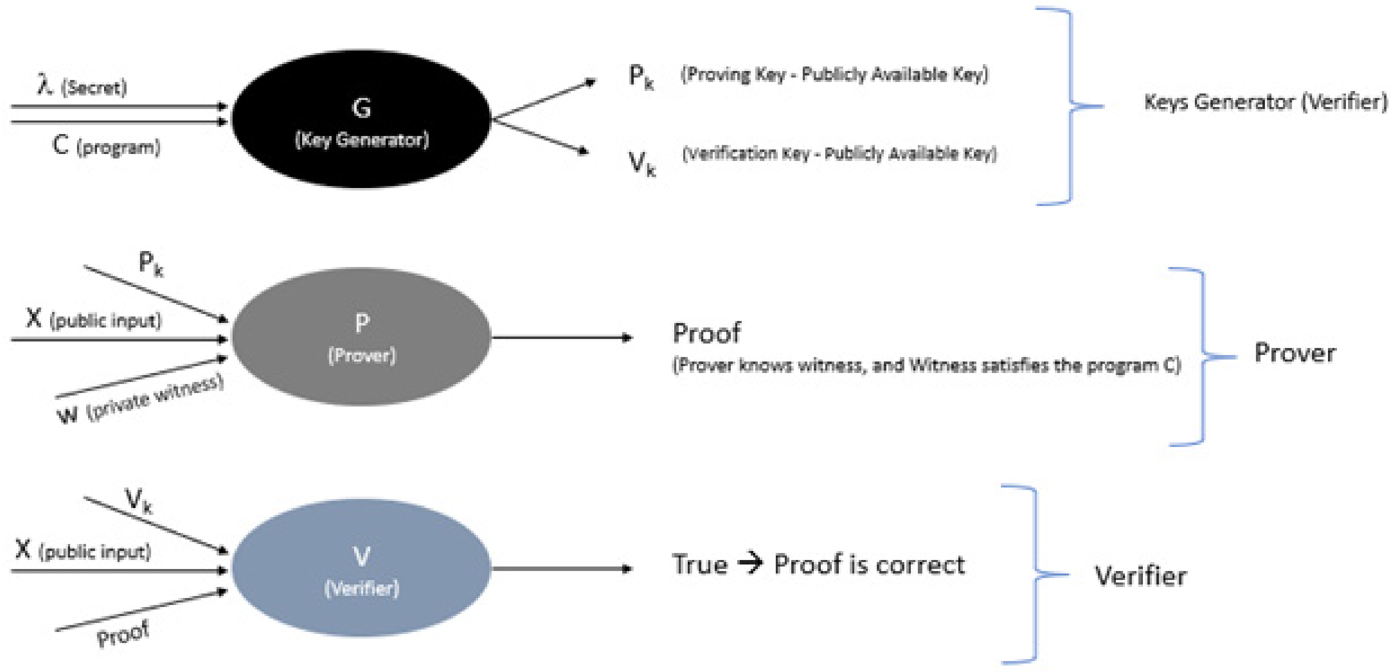
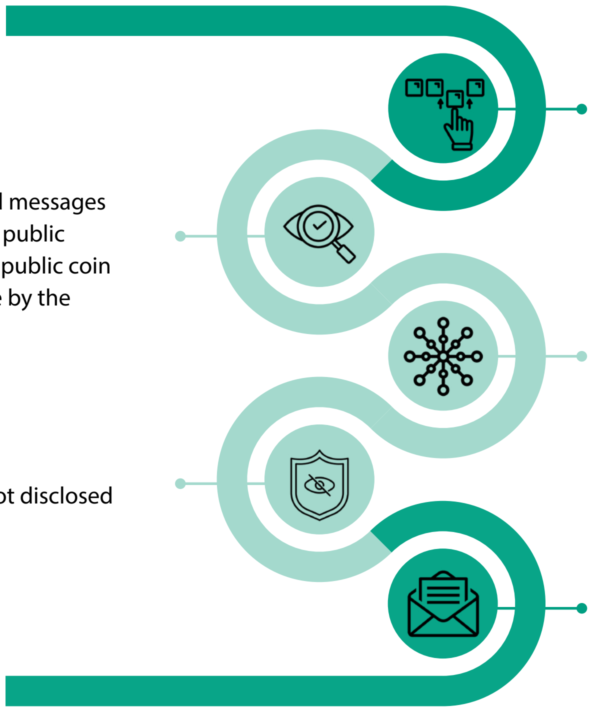

λ (Secret)
C (program)
G
(Key Generator)
P_k (Proving Key - Publicly Available Key)
V_k (Verification Key - Publicly Available Key)
Keys Generator (Verifier)
P_k
X (public input)
W (private witness)
P
(Prover)
Proof
(Prover knows witness, and Witness satisfies the program C)
Prover
V_k
X (public input)
Proof
V
(Verifier)
True → Proof is correct
Verifier

Diagram 1: Implementation approach for zk-SNARK

Scalable Transparent Argument of Knowledge (STARK)

Zk-SNARK has a key drawback that it cannot work without a trusted setup. STARK improved this drawback by utilizing reliable hash functions, quasi-linear PCP (probabilistically checkable proofs), interactive oracle proofs, and fast algebraic local coding protocols (like FRI). STARK achieved faster proof generation and verification at scale as compared to SNARK. It produces short proofs which a verifier can verify without gaining any knowledge of the witness.

Key properties of STARK systems are:



Universality (Turing completeness)
Apply to any computation C (CI Statement)
• CI statement – Computation C, with public input x, and auxiliary private input w, reached output y in T cycles

Transparent

No trusted setup is needed, and messages sent by verifier are random and public coins (random public coins – in public coin protocol, random choices made by the verifier are made public)

Scalable

Proofs of computational integrity of computations lasting T cycles are generated within T cycles (quasi-linear in T) and verification is exponentially faster than T (log T cycles)

Privacy

A prover's private input (w) is not disclosed

Post quantum security
Collision-resistant hash functions used in STARK are currently known to be quantum-resistant

In 2018, StarkWare pioneered the use of STARK validity proofs to address Ethereum's scalability issues. Prover processes transactions in batches off-chain to generate STARK proof. Verifiers can validate the proof on-chain with significantly less computation in quick time. StarkWare received a $12 million grant from Ethereum Foundation for developing scaling solutions using STARK. StarkWare has developed STARK-based solutions such as StarkEx, Starknet, and Cairo.

StarkEx
This is the largest layer-2 scaling network on Ethereum since 2020. Key features include massive scale, support for ERC-20, ERC-721, ERC-1155, and ERC-1155 off-chain minting, and multi-asset trade. StarkEx validity proofs make certain that data committed on-chain is sourced from computations carried out with integrity. STARK ensure privacy from other users as well as from the operator.

Starknet

This is a decentralized validity rollup (zk-rollup) which operates at layer 2 network over Ethereum and enables applications to achieve massive scale without compromising Ethereum's composability and security. It produces STARK proofs off-chain and then the integrity of the proof's computation may be validated on-chain using STARK verifier.

Cairo

Cairo (open source) is Turing-complete and the most efficient programing language for leveraging STARK proofs. It powers StarkEx, Starknet and some of the other scaling applications such as:

dYdX

A platform for trading perpetual contracts with low fees, deep liquidity, and 20x more buying power

ImmutableX

It is a platform for Web3 games and free NFT minting. To Address security, ImmutableX uses STARK proofs on Ethereum.

Polygon zkEVM is a zero-knowledge scaling solution where validity proof of correct state change is submitted to L1 network (Ethereum mainnet) for verification at a later point in time. ZkProver deployed on aggregator validates batches and provide validity proofs (zk-SNARK proofs). ZkProver has main state machine executor, a collection of secondary state machines, a STARK builder, and a zk-SNARK builder. The main state machine executor uses zkASM (zero-knowledge assembly language) to interpret EVM byte code and defines polynomial constraints using Polynomial Identity Language. Every valid batch of transaction must satisfy above defined polynomial constraints. Secondary state machines together provide every computation required for proving correctness of transactions. STARK proof builder produces proofs that are required to demonstrate that all defined polynomial constraints for the batch are satisfied. These STARK proofs are large and consume more storage on ledger as compared to zk-SNARK proofs. ZkProver uses zk-SNARK proofs to prove correctness of STARK proofs. Zk-SNARK-builder generates zk-SNARK proofs which are then committed to the L1 network as validity proofs. Verification of zk-SNARK proofs is relatively less compute-intensive as compared to STARK proofs. Cheaper verification of proofs at L1 is an added advantage of using zk-SNARK proofs in addition to storage.

Bulletproofs

Bulletproofs is a short, non-interactive zero-knowledge proof to prove the validity of a statement in privacy preserving manner. Bulletproofs is a zero-knowledge range proof protocol. Using Bulletproofs, one can prove that an integer a ϵ Zp lies in finite range of the form [0, 2n] where Zp is a finite field with prime order p and n is the number of bits required to express maximum integer in the range. The main idea is to prove that a given integer can be represented in maximum n bits and each of those n bits can either be 0 or 1.

Greg Maxwell used Bulletproofs to introduce the notion of confidential transactions where input and output transaction amounts are hidden in Pedersen commitments. A Pedersen commitment is a point C on an elliptical curve that is cryptographically binding to a message data (m) that is, finding C for alternative m and binding factor (r) is not feasible (due to computational complexity). The curve point (C) is random and contains no information about m and message m cannot be decrypted from C. The point C is generated using a random 256-bit integer (blinding factor) and message m. Given m and r, it is easy to verify that Pederson commitment is correctly generated output. To make confidential transaction publicly verifiable, zero-knowledge proof is attached with a confidential transaction. The proof is sufficient to confirm that the sum of the committed outputs is less than the sum of the committed inputs, and that none of the outputs is negative, and fall in the interval [0, 2n], where 2n is much smaller than the group size. In the context of cryptocurrency, group size implies total supply of the cryptocurrency.

Most of the implementations of confidential transactions use range proofs over committed values, where the proof size is linear in n. The size of confidential transactions increases significantly due to Range proofs. In current implementations, the size of the confidential transaction (with 2 outputs and 32 bits of precision) is 5.4 kb of which 5 kb is allocated to range proof. Due to the large size of range proof, the storage requirement of confidential transactions is high. Bulletproofs optimizes the size of range proofs.

Bulletproofs achieves zero-knowledge through a discrete logarithm problem and the Fiat-Shamir heuristic (aka Fiat-Shamir transformation). In an interactive proof system, first prover sends a commitment to the verifier. The verifier responds with a challenge value (generated uniformly at random). The prover then computes the proof based on the challenge value and the commitment. Using Fiat-Shamir transformation, an interactive knowledge proof system can be converted to a non-interactive knowledge proof system. The thought behind Fiat-Shamir transformation is that instead of the verifier sharing the challenge value, the prover needs to calculate the challenge value by itself. The prover sends commitments to a hash function and generates a challenge message. Using challenge and commit messages, the prover can now, non-interactively, create a proof. The verifier can verify this proof in a non-interactive manner. The security of the Fiat-Shamir transformation is dependent on the randomness of the used hash functions or random oracles and input provided to random oracles. A random oracle is a black-box that provides random responses from its output domain to every unique query. The response to a query is the same every time it is repeated.

Unlike zk-SNARK, trusted setup is not a requirement of Bulletproofs. A prover can combine multiple range proofs for transactions with multiple outputs into a single proof. Bulletproofs greatly improves the size of confidential transactions thereby addressing the storage problem which reduces the overall transaction cost. In comparison to zk-SNARK, the verification time of Bulletproofs is longer.

Confidential transactions are supported in some of the side-chains and private blockchains. Monero is a crypto currency, and one can transfer it in private and untraceable manner. Monero adopted Bulletproofs to achieve confidential transactions (privacy) and it achieved around 80% reduction in transaction size and fees.

Differences between STARK, zk-SNARK, and Bulletproofs

All the zero-knowledge protocols discussed above utilize complex mathematics, cryptography, and makes assumptions – for example, existence of collision-resistant hash functions, setting up of trust environment, etc. Before usage, these protocols should be evaluated for suitability to the use case under consideration. These protocols should be evaluated for their computational efficiency in the generation or verification of proofs. Protocols with simpler proof constructions take less time and could be capable of generating proof on most of the available commercial hardware. Zk-proof protocols generate proofs of different sizes which may depend on witness size. Proofs must be submitted to the blockchain ledger where storage is expensive. Choosing a protocol with shorter proofs will help in optimizing storage. Quick verification is a requirement of most applications. In general, verification complexity of the proofs is less as compared to generation of proofs.

Zk-SNARK relies on trusted setup, that is, a set of keys are to be trusted to create and verify proofs. The secrets used to set up trusted keys must be destroyed once keys are generated to prevent any misuse. It should be noted that zk-SNARK is not quantum safe and may not be used where post-quantum security is required. Proofs generated using zk-SNARK are quite small even though the prover needs more computing power and time to generate proofs as compared to STARK.

STARK generates larger proofs, but they are considered quantum safe. STARK proofs are considered more secure as they are based on time-tested collision-resistant hash functions. Proof generation time of STARK is comparatively low.

Like STARK, Bulletproofs do not need any trusted setup. However, the time taken in generating proof and its verification is more as compared to zk-SNARK or STARK. Bulletproofs generates proofs of smaller size compared to STARK but larger than that of zk-SNARK.

Table 1 lists the key differences between various non-interactive zk-proof protocols.

	Zk-SNARK	STARK	Bulletproofs	Remarks
Trusted setup required	✓	✗	✗	No dependency on trusted setup is good
Post-quantum security	✗	✓	✗	Good to have post-quantum security
Prover: algorithmic complexity	f(n*log(n))	f(n*poly-log(n))	f(n*log(n))	Computational complexity of proof creation. n is the size of the witness.
Verifier: algorithmic complexity	f(1)	f(poly-log(n))	f(n)	n is the size of the witness. Computational complexity of verification is dependent on size of the witness or proof which is not desirable.
Prover time (computational power)	🟡	🟢	🔴	Time and computation power consumed for building the proof.
Verifier time	🟢	🟡	🔴	Verifiers take considerably less time (in ms) in verifying the proof as compared to generation of the proof.
Proof size	🟢	🔴	🟡	Proofs are stored on the blockchain ledger, short proofs are desirable.

Table 1 – Key differences between zk-SNARK, STARK, and Bulletproofs protocols

Use Cases and Adoption in Blockchain Applications

Authentication and access control:

I am who I claim to be, and I am allowed.

In an identity system, the issuer issues verifiable credentials (verification of the credential can be done without interacting with the issuer) to a holder (usually identity owner). At the time of authentication, the holder creates and shares a verifiable presentation (by clubbing data from one or more verifiable credentials without compromising verifiability of the data) with the verifier for verification. Based on the result of the verification, the verifier (usually a service provider) can grant access to the resources. However, the personal information embedded in the presentation is visible to the verifier and this leaves a gap in privacy.

Zk-proofs provide the ability to prove knowledge of personal information (and that it satisfies required criteria/relationship) without revealing it. Anonymous credentials of a user's identity such as credentials without personal information are very useful in reducing the risk associated with identity disclosure and still provide the ability to achieve regulatory compliance.

Hyperledger AnonCreds (Anonymous Credentials) specification uses ZKP in verifiable presentations. These credentials allow users to prove eligibility or authorization without revealing identity thereby enabling privacy-preserving access control and preventing correlation based on signatures.

Semaphore is a protocol designed to cast a signal. It allows a user to prove their membership of a group and send signals such as votes or endorsements without disclosing identity.

In attribute-based access control (ABAC), access is granted when an entity possesses the required attributes. Through zk-proofs, an entity can prove possession of required attributes ensuring privacy during access policy enforcements.

Verifiable computation:

Compute off-chain and make validity proofs available on-chain for verification.

Verifiable computation helps address scalability issues with existing public blockchains (L1) by off-loading transaction processing to the fast Layer 2 (L2) chain which submits validity proofs for a batch/bundle of executed transactions. This also helps in reducing the overall cost of the transaction by optimizing on expensive Layer 1 (L1, like Ethereum, Bitcoin) computation and storage.

L2 solutions create transaction bundles at a pre-specified frequency and submit bundled data (on-chain or off-chain) along-with validity proof. Validium and zero-knowledge rolls-ups (Zk-rollups) are available L2 scaling solutions for Ethereum blockchain. Both solutions move computation and state-storage off-chain using zk-proofs. Zk-rollups bundle transaction data into batches and store batch data on-chain, while Validium does not store the transaction data on-chain.

Machine learning (ML) models are being trained on distributed data (federated learning). However, due to privacy concerns, data owners may resist sharing the data. In such scenarios, validity proofs (SNARK/STARK) are used to prove the computational integrity of ML models. Zero-knowledge proofs of inference of machine learning models (ZKML) are submitted to the blockchain. Zk-proofs validate the accuracy of the local models without the need to share the parameters/data. This helps in addressing privacy concerns as well as in achieving scalability.

RockyBot is the world's first "fully on-chain" AI trading bot. Rocky is both trustless and autonomous, making decisions by itself without a central authority, much like a DeFi protocol. Zk-proofs of the Rocky trading model are made available and validated on Ethereum blockchain. Rocky uses an L1 contract to hold funds and exchanges WEth/USDC on Uniswap. L2 contract implements a 3-layer neural network for predicting future WEth prices.

Privacy-preserving collaboration and data analytics

By implementing ZKP, data providers can share specific properties about their data (for example, statistical properties) without sharing the actual data. This allows organizations to perform data analytics on sensitive information without accessing the raw data. In the process, individual privacy is protected while allowing for valuable insights into the data.

Collaborative analysis or machine learning requires data from multiple parties. Zero-knowledge proofs can be employed for collaborative analysis to protect privacy.

Verifiable supply chain: Proof of existence, proof of delivery, proof of authenticity, proof of quality, and compliance proof.

In a blockchain-based supply chain system, ZKP finds its usage in proving authenticity and integrity of the product while maintaining confidentiality of the data. Network participants can generate zk-proofs of adherence to compliance requirements and make them available on blockchain for verification. To increase the integrity of the supply chain, participants can post proofs of accuracy of the data being contributed to the supply chain for verification by other parties.

MediLedger by Chronicled showcases the chain of custody. Each time a good is moved, zk-proof is submitted to the public ledger. Using verification keys, anyone can verify the custody of the good.

Banking and financial services

The financial services industry is looking forward to employ ZKP to achieve privacy, scalability, and security in financial processes/ transactions thereby enhancing the trust and integrity of the system. ZKPs can be used in digital currencies to prove ownership and validity of funds without revealing specific transaction details or personal identities. Furthermore, transactions can be done privately at Layer 2, proof of which may be submitted to main-net and can be made available publicly for verification.

dYdX is the next generation trading platform for trading perpetual contracts. It utilizes zk-STARK to achieve scalability and privacy while reaping the benefits of decentralization.

Zcash is using zk-SNARK for transaction privacy (transaction shielding). When user initiates a private transaction, proving and verifying keys are generated. Prover and verifier can use these keys to prove or verify the transaction. Transaction inputs and outputs are represented as Pedersen commitments (cryptographic constructs which hide the value but allow its verification, which means that the information about the value is available in the commitment even though the value itself is hidden/private). To establish validity of the private transaction, the spender uses these Pedersen commitments as the input and creates zk-SNARK proof. The zk-SNARK proof thus created is sufficient to prove, spender is authorized to spend the input, the sum of inputs is equal to the sum of outputs and there is no double spend. When a transaction is submitted, the Zcash network verifies the proof to confirm validity of the transaction.

Auditing and compliance

With ZKPs, organizations can prove compliance to regulatory or contractual obligations without disclosing underlying confidential data. This may eliminate the need for sharing sensitive information with the competent authorities and yet authoritatively claim compliance with regulations and policies.

Conclusion

Adoption of zero-knowledge proofs helps enhance privacy, confidentiality, and scalability in blockchain applications while ensuring data integrity and authenticity. Layer 2 solutions are harnessing zk-proof to improve scalability and efficiency in blockchain networks. In the emerging field of collaborative analytics and federated machine learning, zero-knowledge proofs find relevance in facilitating efficient data processing and ensuring verifiability while maintaining the privacy of sensitive information.

The suitability of a protocol depends on the validity of security assumptions made in the protocol for a given scenario and use case requirements or constraints. Key protocol parameters that help in decision making are the feasibility of the trusted setup, computational complexity at the prover's end, prover time, proof size, and verification time.

Zk-SNARK produces the shortest proofs but requires a trusted setup. Zk-SNARK and Bulletproofs are found to be better on small-depth circuits (low complexity) and for computations that require repeated execution of the same fixed circuits. Zk-STARK has the fastest prover time for all kinds of circuits and can be a better choice for cases requiring large-scale computations.

Verifiable supply chain, verifiable computation, shielded transactions, anonymous credentials, identity management, autonomous trading systems, and auditing and compliance are some of the key use cases of zk-proofs in blockchain applications, among several others.

References

1. Review of Zero knowledge proofs (Chen et al., December 2021)
2. The Knowledge Complexity of Interactive Proof Systems (Goldwasser et al., December 1985)
3. Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data (Bitansky et al., December 2012)
4. The hunting of the SNARK (Bitansky et al., October 2016)
5. Why and how Zk-SNARK Works (Maksym Petkus, June 2019)
6. Proofs that Yield Nothing But Their Validity All Languages in NP Have Zero-Knowledge Proof Systems (Goldreich et al., July 1991)
7. Schnorr Non-Interactive Zero-Knowledge Proof (F. Hao, September 2017)
8. Scalable, transparent, and post-quantum secure computational integrity (Eli Ben-Sasson et al., March 2018)
9. Publicly Verifiable Non-Interactive Zero-Knowledge Proofs (Lapidot et al., 1991)
10. From PCP to zk-STARK
11. Zero knowledge proofs: SNARK vs STARK (Blog by Consensys, Mattison Asher, May 2021)
12. Verification of Quantum Computation: An Overview of Existing Approaches (Gheorghiu et al., July 2018)
13. Polygon zkEVM documentation
14. Bulletproofs: Short proofs for Confidential Transactions and More (Benedikt Bunz et al, 2018)
15. Bulletproofs in Crypto – An introduction to non-interactive ZKP (Panther Team, September 2022)
16. AnonCreds specifications
17. Validium
18. ZCash Specifications (Hopwood et al., December 2023)
19. RockyBot
20. StarkWare
21. Cairo
22. Semaphore
23. zk-SNARKs vs zk-STARKs: Comparing Zero-knowledge Proofs

About the Author

Vivek Rastogi

Senior Technology Architect

He possesses extensive experience and expertise in harnessing state-of-the-art technologies such as blockchain to address the specific needs of various industry sectors, particularly with in banking and financial realms. He has worked with prominent clients across multiple regions and played a crucial role in guiding them through their digital transformation journey by contributing towards development of high-performing innovative solutions to modernize business and deliver value to the customers.

For more information, contact askus@infosys.com

Infosys®
Navigate your next

Infosys.com | NYSE: INFY

Stay Connected