Computing and Informatics, Vol. 45, 2026, 902–935, doi: 10.31577/cai_2026_4_902

A SCALABLE LAYER 2 OFF-CHAIN PROTOCOL FOR BLOCKCHAIN USING ETHEREUM ROLLUPS

K. N. Unnikrishnan, P. Victer Paul

Indian Institute of Information Technology Kottayam
Kerala, India
e-mail: {kalleli.phd11003, victerpaul}@iiitkottayam.ac.in

Abstract. The two biggest problems facing modern blockchain systems are scalability and privacy. The system's capacity to scale is constrained by the need to process transactions at each node. In addition, despite all other benefits, a widely cited barrier to blockchain adoption is the requirement to disclose all transaction data at every node, thereby making the data public. In order to overcome these two issues, we introduce the ScaZK (Scalable Zero Knowledge) protocol, which primarily makes two contributions: i) In order to improve performance, we propose a new, lightweight protocol that uses non-interactive proofs for off-chain calculations, which limits on-chain computational efforts to the verification of execution accuracy rather than actual execution. Private data utilised in the off-chain computation does not need to be made public in order to verify correctness because of the verifiable computation scheme's zero-knowledge property. ii) We present a framework for defining, integrating, and deploying these off-chain calculations. It includes a compiler, proof and verification generators, and a domain-specific language for smart contracts. It hides the considerable complexity that zero-knowledge proofs entail, offers developers a more comfortable and advanced level of programming abstractions, and allows circuit integration, which promotes adoption. Our contributions in this work specifically include: i) a novel proof bucketing compression scheme, ii) integration of a B+ Merkle Tree for efficient rollup root computation, iii) parallel off-chain execution for throughput scalability, and iv) partial zkEVM compatibility for developer accessibility. We also present stress testing, fault recovery analysis, and gas breakdown for multiple proof systems to validate the protocol's resilience and scalability.

Keywords: Blockchain, scalability, layer 2 solutions, zero-knowledge rollups, SNARKs

Mathematics Subject Classification 2010: 94A60

1 INTRODUCTION

Blockchain technology has attracted significant attention because it is the foundation of all contemporary cryptocurrencies, including Dogecoin, Litecoin, and Bitcoin. We must comprehend the history of blockchain technology, what it stands for, and how applicable it is to the real world to have a more thorough grasp of blockchains, the block generation algorithm, and how transactions are included in a block. Even with all of its benefits, blockchain technology is still relatively new, and several issues must be resolved before blockchain reaches its full potential. This research explores the problem of blockchain scalability and offers a comparative analysis of various blockchain characteristics, including mining time and block generation times, using real-time data to illustrate the main reasons why blockchains are not scalable.

The following years will inevitably see a surge in transactions that will call for the use of blockchain protocols similar to Bitcoin, given the constant adoption and growth of cryptocurrency transactions due to its use in digital transfer, smart contracts, and inexpensive remittance. Such digital systems must be guaranteed to operate correctly and then maintained in order for them to function. Therefore, a variety of controls are put in place by these cryptocurrency systems to ensure that the digital currency is distributed and operated correctly. Additionally, care must be taken to prevent malicious actors from exploiting system vulnerabilities.

Reduced network latency and increased throughput are not possible due to the core flaws and obstructions in the Bitcoin network. The term "scalability" is imprecise and does not refer to a single feature of the blockchain. It encompasses several measures or may be described as the aggregation of multiple metrics. Therefore, it is essential to evaluate the current Bitcoin system's scalability, identify its limitations, and resolve open issues. The objective is to gain a deeper comprehension of the limitations about the scalability of Bitcoin, in order to facilitate future innovation and the comprehension of other scalable protocols. Understanding how far parameters may be pushed to increase scalability and efficiency without jeopardising security and other considerations is also crucial.

ScaZK protocol addresses all these considerations, and results indicate that it works efficiently. The novel contributions include:

Bucket-Based Proof Compression: A novel compression method that groups zkSNARK proofs into indexed "buckets," reducing overall Layer 1 calldata footprint and proof size scalability overhead.

B+ Merkle Tree Integration: A new variant of the Merkle Tree structure is proposed (B+ Tree) to accelerate hash computation, enabling faster rollup batch finalisation and reduced root update latency.

Parallelised Off-Chain Transaction Processing: The system supports distributed off-chain execution where multiple sequencer nodes can process transaction batches concurrently, increasing Layer 2 throughput.

Partial zkEVM Compatibility: Enables porting of Solidity-based smart contracts to ScaZK with minimal changes, lowering the barrier for developer adoption, unlike SNARK-based systems like Loopring.

Polylogarithmic to Near-Constant Scaling: Through proof bucketing, B+ trees, and optimised batching, the protocol achieves near $O(1)$ scaling per batch in practice, exceeding traditional polylogarithmic models.

Algorithmic Clarity: The paper contributes a complete, modular, and implementable pseudocode framework for every rollup phase: initiation, transaction batching, SNARK proof generation, on-chain verification, and withdrawal.

2 RELATED WORK

Blockchain technology has several key features [1]: *Decentralized* – instead of a central governing authority, a group of nodes maintain the network, *Distributed* – indistinguishable copies of all information are shared among the nodes, *Secure* – a block with all its information gets entirely hashed cryptographically, to be stored within the next block in the chain, *Immutable* – tampering of data is nearly impossible due to being added after approval from everyone in the network, and *Trustless* – a confidence machine that eliminates the need to trust the participants involved.

Bitcoin Blockchain was developed as a solution to the double-spending problem using a P2P network. It has a distributed timestamp server using a Proof-of-Work (PoW) system. The PoW is based on One-CPU-One-Vote rather than One-IP-One-Vote. The main advantage is that it incentivises miners with the generation of new coins. It suffers from energy efficiency issues due to the high computing power needed. More critically, throughput is limited to 3–7 transactions per second (tps). Ethereum Blockchain introduced improvements and demonstrated that blockchain technology extends beyond financial transactions. It included smart contracts that are cryptographic "boxes" that contain value and unlock only if certain conditions are met. It is Turing-complete, Value-aware, blockchain-aware and stateful. Scalability was still a challenge with a throughput of just 15 tps, whereas payment systems such as Visa (1 700 tps) and PayPal (193 tps) far exceed it.

Another major aspect of blockchain systems is the Consensus Mechanisms, which represents a unique common view of the entire blockchain. It is a method to determine the creator of the next block among the competing miners. There are a variety of ways to achieve this: *Proof of Work (PoW)* [1] – a cryptographic puzzle-solving procedure in a computation-intensive manner; *Proof of Stake (PoS)* [2] – the validator of next block is decided according to coins held, *Delegated Proof of Stake (DPoS)* [3] – top witnesses selected through voting are rewarded for verifying blocks, *Proof of Elapsed Time (PoET)* [4] – a randomly generated elapsed time (sleep) to

decide mining rights, *Proof of Authority (PoA)* [5] – validator's identity performs the role of stake, and, *Proof of Capacity (PoC)* [6] – available hard drive space to decide mining rights and validate transactions. The major challenge involved is that decentralisation is costly, though there are scalability improvements, and the Security requirements and resource efficiency are not balanced.

2.1 Scalability Challenge

Scalability is the ability to support high transactional throughput and future growth. As blockchain adoption accelerates, performance should not suffer. The major factors affecting scalability are Throughput, Latency, Cost and Storage. The scalability trilemma states that a blockchain can simultaneously achieve at most two of: decentralisation, scalability, and security.

Layer 1 solutions are the modifications to the blockchain protocol which involve: *Block Data* – changes in block size, by increasing the size or compressing it, *Block Structure* – instead of a single chain structure, a directed acyclic graph (DAG) structure with blocks as vertices is used, *Consensus* – optimizations in the consensus strategies in terms of energy consumption, *Sharding* – divides a blockchain network into several smaller networks called *shards*. *Layer 2 solutions* are mechanisms outside the blockchain. Those solutions include: *Payment Channels* – direct/indirect communication channels between parties, *Sidechains* – "children" blockchains anchored on the main chain and running in parallel to it, *Cross-Chain* – cross-communicate among independent blockchains with any underlying technology (isomorphic/heterogeneous), *Off-Chain Computations* – expensive computational tasks for state verifications are carried out off-chain and, *Rollups* – execute the transactions outside the main chain and the transaction data is posted back.

Rollups execute the transactions outside the main chain, and the transaction data is posted to Layer 1. It rolls up (a bundle) transactions performed off the chain through a cryptographic proof called SNARK (Succinct Non-interactive ARgument of Knowledge). They are of two types: *Optimistic Rollups* that assume that all off-chain transactions are valid, and challenges are done using fraud proof, and *Zero-Knowledge (ZK) Rollups* where, on completion of the off-chain transactions, a validity proof is submitted to the main chain.

The study [7] provides a detailed comparison of Layer 2 blockchain scaling solutions, particularly Optimistic and Zero-Knowledge Rollups, emphasising their trade-offs in terms of scalability, cost, and finality. The work [8] propose a blockchain-based model integrating IoT and human interaction for food safety, leveraging smart contracts for transparent production control. The authors in [9] survey Layer 2 technologies and categorise scalability approaches such as state channels, Plasma, and Rollups, highlighting the benefits and limitations of each. Together, these studies underscore the growing importance of Layer 2 solutions in overcoming Ethereum's throughput and fee challenges.

2.2 Background

Many zero-knowledge proof schemes have been studied for the purpose of this research to bring in computational efficiency in proof generation. The proof submitted by the operator in the Layer 2 is submitted to the verifier smart contract. The speed at which this addition of the off-chain transactions can be improved mainly in two ways: i) reduction in time in generating the proof, and ii) reduction in time of verifying the proof at Layer 1.

In this section, we check the schemes proposed by Groth et al., SONIC, PLONK and Marlin.

2.2.1 Groth16

Groth16 scheme [10] introduced the concept of zk-SNARKs, which is a specific construction for non-interactive zero-knowledge proofs. It was designed to provide efficient and scalable ZKPs and offered improved performance in terms of proof size, verification time, and computational requirements compared to earlier zk-SNARK constructions.

This scheme builds upon the concept of Linear Probabilistically Checkable Proofs (PCPs), which are a type of proof system that allow for efficient verification of linear computations. Groth16 represents computations as Quadratic Arithmetic Programs, which are a flexible and efficient way to express a wide range of computations in a structured form. QAPs provide a concise representation and enable efficient proof generation and verification. The non-interactive proofs are constructed using a combination of bilinear pairings, polynomial evaluations, and algebraic techniques. The proof generation process involves constructing a succinct proof that satisfies completeness, soundness, and zero-knowledge properties.

2.2.2 SONIC

SONIC scheme [11] by Maller et al. provides a way to efficiently verify computations without requiring interaction between the prover and verifier. The key feature of SONIC is its scalability and efficiency. It enables proving and verifying complex computations with low computational and communication overhead. SONIC achieves this through the use of recursive composition, which allows for the construction of succinct proofs for computations of arbitrary depth.

SONIC provides a succinct proof size, meaning that the proof length is relatively short compared to the size of the original computation. It achieves this by leveraging recursive composition and efficient cryptographic techniques. The scheme involves seven steps:

1. **Setup:** The verifier generates a set of common parameters that will be used by the prover and verifier.
2. **Circuit Encoding:** The computation to be proven is encoded as a Boolean circuit. Each gate in the circuit represents an operation or computation step.

3. **Recursive Composition:** SONIC uses recursive composition to break down the circuit into smaller sub-circuits. This allows for the generation of a succinct proof by leveraging the results of previously proven sub-circuits.
4. **Prover Computation:** The prover evaluates the circuit, computing intermediate values and constructing proof elements called "witnesses" for each gate in the circuit.
5. **Commitment and Response:** The prover commits to the witnesses and computes response values for each gate in the circuit.
6. **Proof Generation:** The prover constructs a proof using the commitment and response values. The proof is generated in a way that ensures soundness, meaning that the prover cannot cheat or provide false information.
7. **Verification:** The verifier checks the validity of the proof by examining the proof elements and verifying that they adhere to the circuit's rules. The verifier does not need to interact with the prover during this process.

2.2.3 PLONK

PLONK (Permutations over Lagrange-Bases for Oecumenical Noninteractive Arguments of Knowledge) scheme [12] is a construction for zero-knowledge proofs (ZKPs) that provides a framework for creating efficient and scalable zk-SNARKs. The key idea is to leverage permutation-based polynomials and linear operations over randomised polynomial kernels to achieve efficient and scalable ZKPs.

This construction method has demonstrated significant improvements in efficiency and scalability compared to previous zk-SNARK constructions. It enables faster proof generation and verification, reduces the size of the proofs, and allows for efficient handling of large-scale computations. The important aspects of the PLONK scheme are:

1. **Permutation-based Polynomials:** PLONK employs permutation polynomials, which are polynomials that can permute their coefficients. Permutation polynomials enable efficient circuit evaluations and eliminate the need for complex polynomial arithmetics, resulting in faster and more scalable computations.
2. **Linear Operations:** PLONK utilises linear operations over the permutation polynomials to efficiently handle circuit constraints. This allows for fast and succinct representation of computations and reduces the overall complexity of the proof generation and verification processes.
3. **Randomised Polynomial Kernels:** PLONK incorporates randomised polynomial kernels, which are polynomials used to represent intermediate computations and constraints. The randomisation of the kernels enhances the security of the protocol by preventing potential attacks that exploit algebraic structure.
4. **Low-degree Testing:** PLONK employs low-degree testing algorithms to efficiently verify polynomial computations. This ensures that the proofs generated

using PLONK remain succinct and the verification process is fast, even for complex computations.

2.2.4 Marlin

Marlin protocol [13] is a recent advancement that aims to reduce the computational and communication costs associated with generating and verifying zk-SNARK proofs.

The protocol, while leveraging permutation-based zk-SNARKs, is an efficient Algebraic Holographic Proof (AHP) for Rank-1 Constraint Satisfiability (R1CS) that achieves linear proof length and constant query complexity. It is known for its efficiency and scalability in handling computations that involve large numbers of constraints. It also utilises the low-degree testing algorithm that enables fast and succinct verification of polynomial computations. This reduces the verification time and ensures that the proof size remains small, even for complex computations. The concept of random evaluation domains is introduced, where polynomials are evaluated at random points. This technique enhances the security of the protocol by preventing potential attacks that exploit the structure of the computation.

Similar to SONIC, Marlin employs recursive composition to break down large computations into smaller sub-computations. This allows for the construction of succinct proofs by reusing previously proven sub-proofs. The protocol has demonstrated significant improvements in efficiency and scalability compared to previous zk-SNARK constructions. It enables faster proof generation and verification, reduces communication costs, and provides succinct proofs for complex computations.

2.3 Aim and Objectives

To Design and Evaluate a new lightweight off-chain scaling protocol to improve the performance of blockchain in terms of computational complexity and resource utilisation:

- To analyse the advantages and shortcomings of Bitcoin and Ethereum with respect to the performance and scalability by addressing the trade-offs indicated by the trilemma of Scalability, Security and Decentralisation.
- To design and develop an enhanced lightweight technique for zero-knowledge rollups.
- To analyse and evaluate the performance of the technique in terms of scalability, computational complexity and resource utilisation.
- To design a technique to implement parallelism in the execution of off-chain transactions to be verified on the main chain.
- To propose a model to integrate the scaling improvements of the enhanced lightweight technique with the distributed execution of off-chain transactions.

3 SCAZK PROTOCOL

Blockchain's capacity to deliver data immutability makes it transformative for enterprise applications. Additionally, by reducing the necessity for a central authority through consensus methods, decentralisation eliminates middlemen. It improves security through the use of cryptographically hashed data storage. It is transparent because of its distributed nature, in which each user of the system maintains a ledger of the events. Blockchains with Layer 1 solutions tend to become centralised and make transaction verifications challenging. In contrast, Layer 2 Rollups increase throughput by decreasing transaction sizes and eliminating the need for Layer 1 execution.

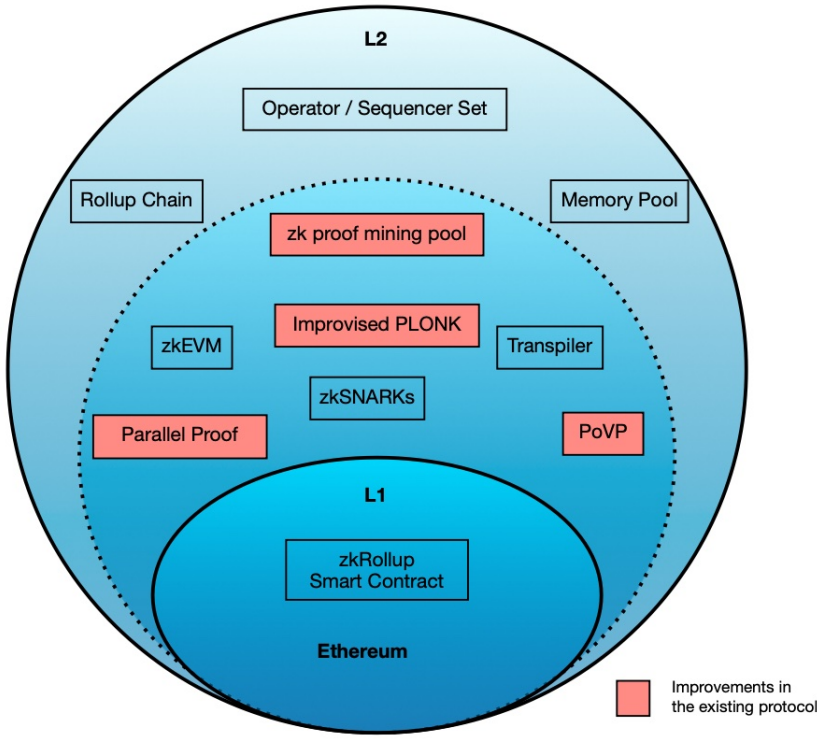


Figure 1. ScaZK architecture

ScaZK offers a significant improvement in the scope of what a blockchain network can handle in terms of scalability. Figure 1 shows the ScaZK architecture with the improvements shaded differently. The concerns of Throughput, Latency, Cost, and Storage are addressed while keeping the underlying blockchain least affected. The Interoperability of the system is as significant as being scalable. Though zero-knowledge rollups are efficient, proof size grows with transaction count; this paper

addresses that limitation. This solution has an optimal trade-off between Scalability, Decentralisation and Security.

The verification of off-chain transactions is done better by relying on mathematical proofs rather than relying on better user behaviour. Zk-SNARKs help in the generation of better proofs but suffer from intensive computational overhead. Improvement in zkEVM with respect to Layer 2 scaling is brought in, which is equivalent to the existing EVM, which helps current users of EVM to switch in effortlessly. Distribution of transaction execution in the off-chain environment and efficient utilisation of resources in terms of gas costs during computation improve the overall performance. The zk-rollup technique is depicted in Figure 2.

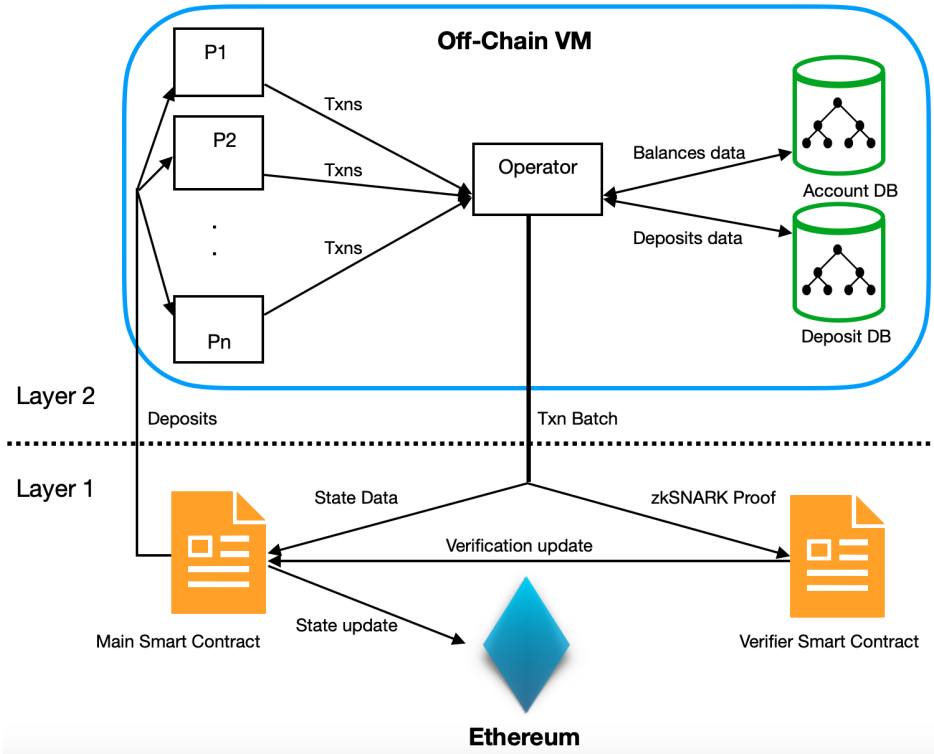


Figure 2. Zk-rollup technique

Rollup systems maintain the data in the main chain and utilise on-chain resources for transactions submitted from the Layer 2. These resources include: the *state* of the blockchain, and *witness*, which is the signed consent of transacting participants. While Optimistic rollups scale linearly with the transaction count, the validity proofs of zk-rollups maintain a single proof rather than multiple digital signatures. The result is that zk-rollups achieve a polylogarithmic scaling ($O(\log^k(n))$)

with respect to the transaction count, which helps in reducing data usage in the main chain. This is better in comparison with linear scaling ($O(n)$).

Table 1 compares and contrasts the proposed ScaZK protocol with the existing protocols.

To put the proposed ScaZK protocol within the broad Layer 2 context, we compare it against key alternatives – zkSync, Optimism, Polygon PoS, Raiden, and StarkEx – along five critical axes: security inheritance, throughput, finality, gas efficiency, and censorship resistance. Figure 3 illustrates the architectural distinctions, while Table 2 provides scholarly citations for each protocol. A quantitative comparison is visualised in the bar chart (see Figure 4), which highlights ScaZK's unique position in offering high security and low finality latency with efficient on-chain verification. The results affirm ScaZK's balance between decentralisation and scalability within zero-knowledge-based rollups.

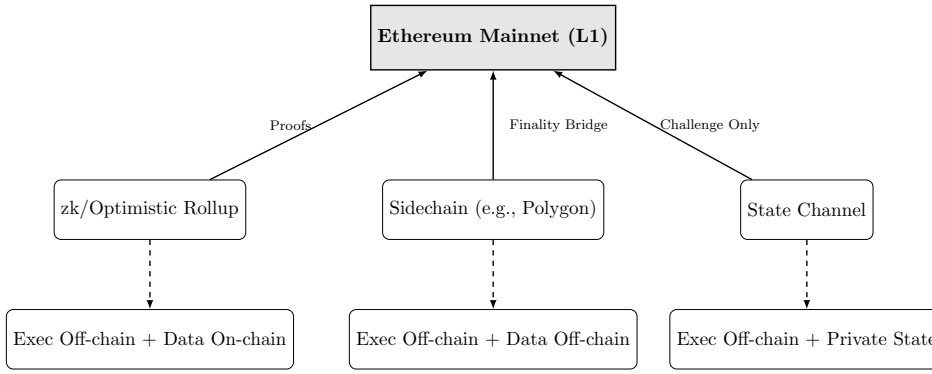


Figure 3. Layer 2 paradigms compared by execution location and data commitment model

4 MATERIAL AND METHODS

In this research, a fixed-size scaling is achieved as in Payment channels, where any number of transactions are submitted as a single update to the main chain. As zk-rollups use a single proof for a batch of computations, the idea is to include a larger number of transactions per batch. Since the Merkle tree root header of the transactions is being used for the proof, we propose using a B+ Merkle tree [20] instead of a normal binary Merkle tree.

The concept is explained with a comparison using a sample scenario. Figure 5 shows a normal Binary Merkle tree with eight transactions, Tx0, Tx1, ..., Tx7. The computation starts with generating Hash(0), Hash(1), ..., Hash(7) for the transactions. At the next level, new hashes are generated by combining these hashes. The root, termed the Merkle root, will have the combined hash Hash(01234567) of all the transactions involved.

Feature	ScaZK	zkSync	Loopring	StarkEx
Proof Compression via Buckets	Introduced a novel bucket-and-index compression to reduce ZK proof size	Not used; standard batching	Not present	STARK-based but no explicit bucketing
B+ Merkle Tree for Rollups	Replaces standard Merkle tree with B+ Tree to speed up root computation and reduce proof aggregation latency	Uses binary Merkle	Uses binary Merkle	Uses a variant of Sparse Merkle Trees
zkEVM Compatibility (Partial)	Supports partial EVM compatibility for rollup circuits, improving developer adoption	Partial (planned for full)	Not EVM compatible	Not EVM compatible
Parallel Transaction Execution	Proposes distributed off-chain execution of transactions with parallelism-aware batching	Centralized sequencer with no parallelism optimization	Loopring is centralized DEX	Uses centralized prover
Proof Size Plateau Avoidance	Proof remains $\sim$ 153 KB even for 64 transaction sets, indicating better scalability	Grows with batch size	Grows with transaction count	STARKs scale well but are larger
Latency-Throughput Optimized State Transitions	Polylog $\rightarrow$ Near-Constant Time for state validation with B+ tree & compression	Linear or logarithmic	Linear	Efficient, but larger proofs
Algorithmic Transparency &Modularisation	Explicit algorithm steps for deposit, zk-proof generation, rollup integration	Abstracted in closed-source logic	Mostly protocol documentation	Proprietary prover (StarkWare)

Table 1. ScaZK protocol comparison overview

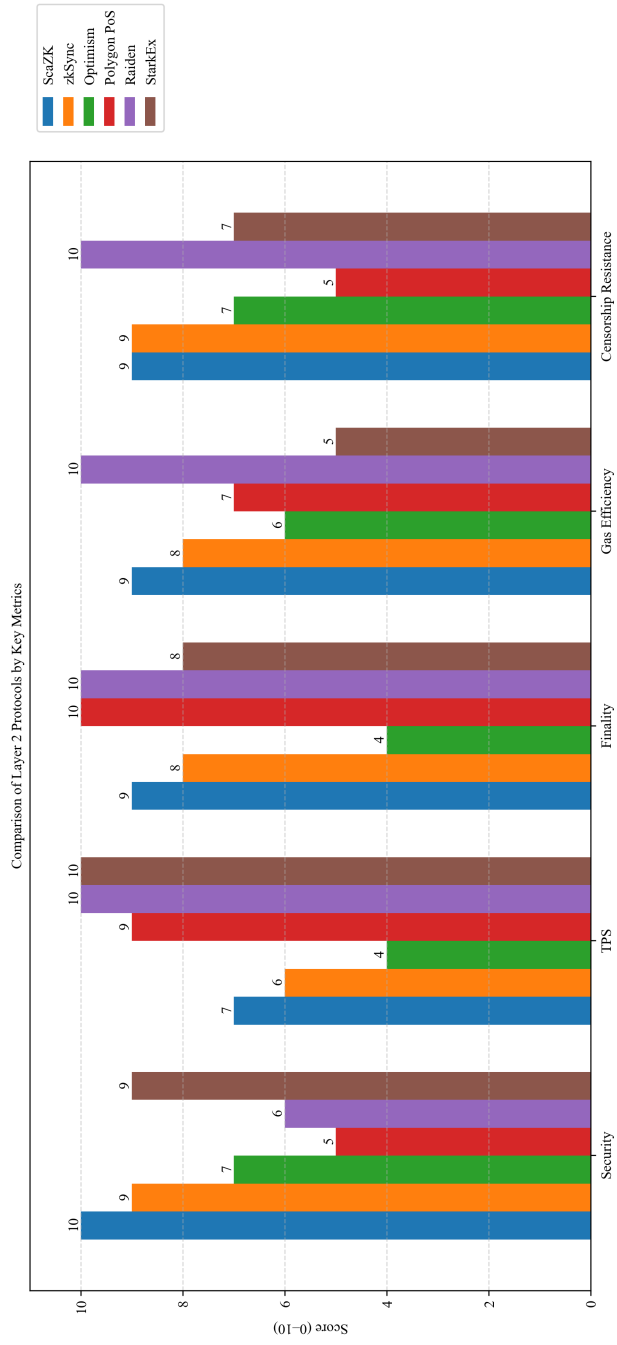


Figure 4. Comparison of ScaZK using key metrics

Protocol	zkSync	Optimism	Polygon PoS	Raiden	StarkEx
Representative Scholarly References	[14, 15]	[16]	[17]	[18]	[19]

Table 2. Literature References for Layer 2 Protocols

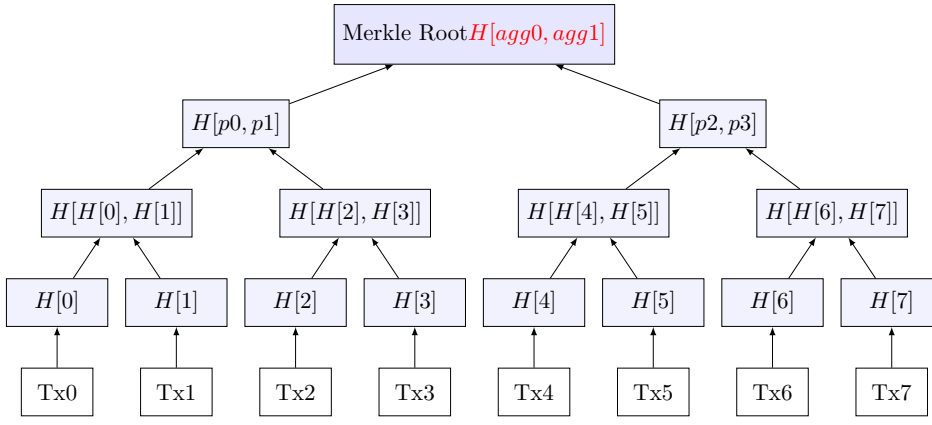


Figure 5. Binary Merkle tree

Figure 6 shows the hash generation using the same number of transactions, but using a B+ Merkle tree. The order in which transactions are added to the tree can be thought of as in the same order as the transactions are performed. Here, the advantage is to arrive at the Merkle root at a faster pace as compared to the normal binary Merkle tree, and the gains could be substantial when the transaction count grows significantly.

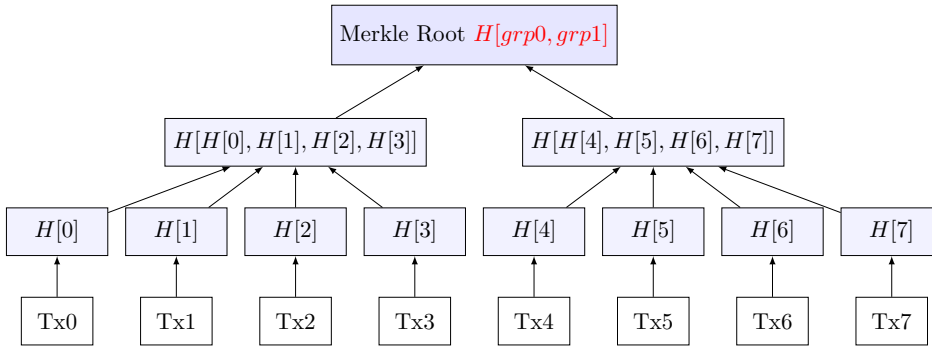


Figure 6. B+ Merkle tree

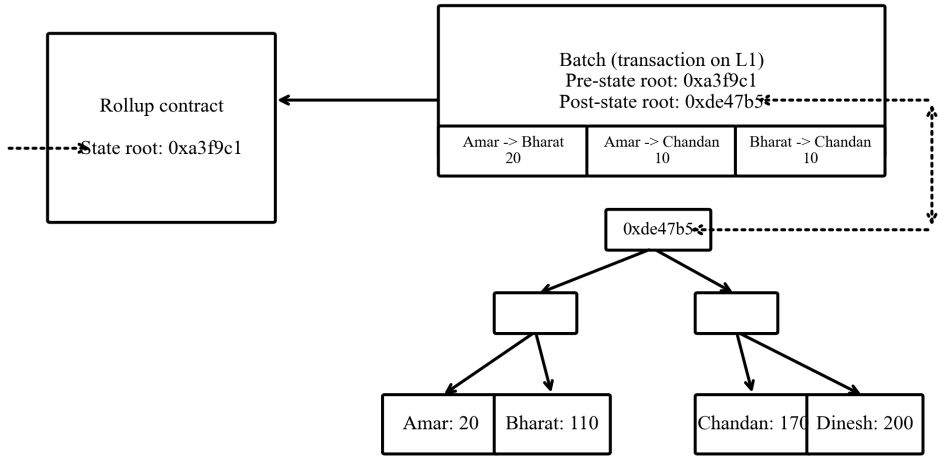


Figure 7. State update

An enhanced proof construction method needs to be applied as in [11] that uses an updatable SRS that scales linearly rather than quadratically in size. This, in turn, needs to be used to achieve a near fixed-size scaling ($O(1)$) as in Payment channels by improving from the current polylogarithmic scaling ($O(\log^k(n))$). The validity proof needs to be further compressed by creating buckets of proofs and indexing them, and by including a larger number of transactions per batch for the validity proofs of zk-rollups.

The state update of zk-rollups is done as in Figure 7, where the transactions that form the Merkle tree in the rollup update the Merkle root hash after the new transactions and will have a new state root. The main chain, when the rollup batch is received, compares the pre-state root and post-state root and verifies the batch.

As in Figure 8, the rollup batch will be integrated into the main chain transaction as part of the *calldata*.

The solution proposed by the ScaZK protocol involves six phases of the zk-rollups as explained in the following subsections.

4.1 Initiation of Layer 2 Off-Chain

The initiation of the off-chain Layer 2 (L2) in zk-rollups involves the setup and deployment of the necessary infrastructure to support the off-chain processing and validation of transactions. The major activity involved is the deployment of the Layer 2 smart contract in the Layer 1 (L1) blockchain. The Layer 2 smart contract acts as the bridge between the Layer 1 and Layer 2, facilitating interactions and data exchange. The setting up of Layer 2 accounts for users who want to participate in the Layer 2 also needs to be done.

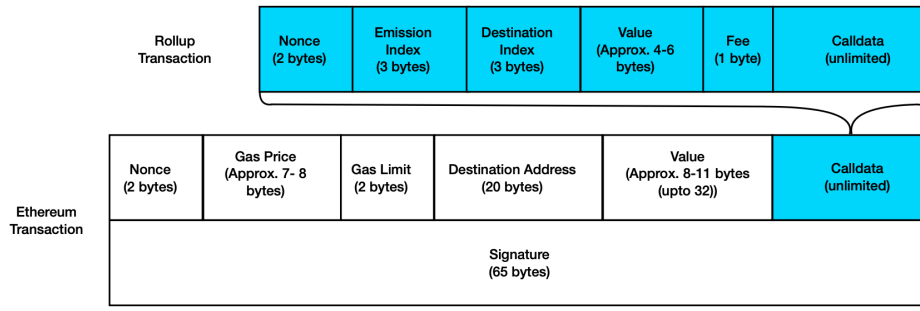


Figure 8. Rollup integration

Algorithm 1: Initiation of L2 off-chain

$SC_{main}.deploy()$
$L \leftarrow$ list of users who wish to participate in L2
while $L \neq$ EMPTY **do**
  $USER.L2account = \{$
    $account.pubkey\_x$,
    $account.pubkey\_y$,
    $account.balance$,
    $account.nonce$,
    $account.token\_type$
  $\}$
  $generate(USER.L2address)$
  $USER.L1balance \leftarrow (USER.L1balance - account.balance)$
  $USER.lockedfund \leftarrow account.balance$
  $L1TXNS \leftarrow FundLock(USER)$
end while
$SC_{verifier}.deploy()$

4.2 Off-Chain Transaction Processing

In the zk-rollup solution, multiple transactions are first processed off-chain within the Layer 2 environment. This allows for scalability, as the bulk of the computation is done off-chain, reducing the load on the Layer 1 blockchain. Once users have deposited their funds, the off-chain transaction processing can begin. Users can submit their transactions to the Layer 2 environment, and the Layer 2 solution processes and validates these transactions off-chain (see Figure 9).

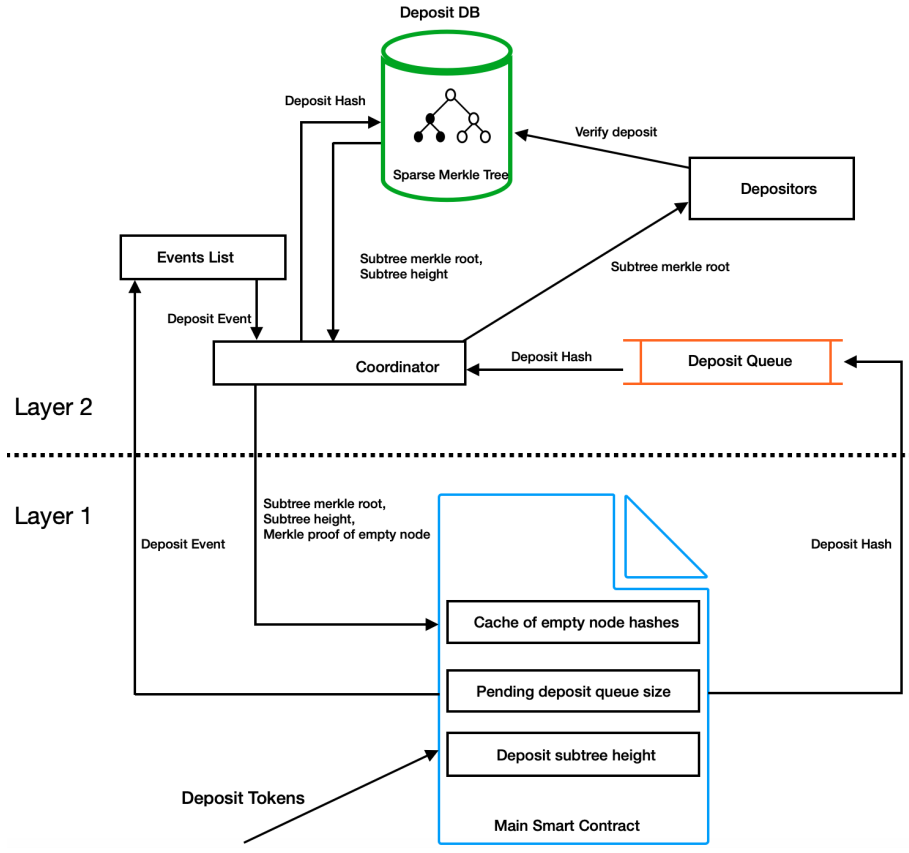


Figure 9. Off-chain transaction processing

4.3 zk-SNARK Proof Generation

After validating the transactions and computing the state changes, zk-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) proofs are generated. These cryptographic proofs provide evidence that the computations were correctly executed and that the state changes are valid without revealing sensitive information.

In ScaZK, we use Groth16 as the backend SNARK. The process is to offload multiple SNARK calls in buckets to enable parallel proof generation (shards or threads). It also compresses intermediate witness/proof data, which results in controlling proof size growth. The integration of the B+ Tree to optimise state root transitions reduces latency before proof is initiated. This makes the horizontal scaling possible (across cores/provers) while maintaining constant proof size for batches. This is not exploited directly in Groth16 and PLONK.

Algorithm 2: Processing of Deposits

```
1:  procedure DEPOSIT()
2:    for (nodes exist in the sparse Merkle tree M) do
3:      M_node(i).add[H(null)]
4:    end for
5:    qn ← 0
6:    sh ← 0
7:    i ← 0
8:    flag ← true
9:    while deposit is made to the rollup do
10:     DQ_i ← H(pubkey_x, pubkey_y, balance, nonce, token_type)
11:     EVENTLIST.add(DQ_i)
12:     qn ← qn + 1
13:     r ← log_2 qn % 2
14:     if (qn % 2 == 0) then
15:       H_new ← H[H_i, H_{i−1}]
16:       DQ_i ← null
17:       DQ_{i−1} ← H_new
18:       if M_node(i) == H(null) then
19:         M_node(i) ← H_new
20:       end if
21:       i ← i − 1
22:       if (r == 0 && flag) then
23:         r ← r − 1
24:         flag ← false
25:         sh ← sh + 1
26:         go to 15
27:       end if
28:     end if
29:     i ← i + 1
30:   end while
31: end procedure
```

For a batch size (number of transactions per proof) of b and constraint count per transaction (fixed) of n, although we inherit the asymptotic costs of Groth16, our bucket-based parallelization allows per-thread cost to approach $O((b/n)\log n)$ under optimal conditions, effectively reducing wall-clock latency.

The step-by-step complexity analysis breakdown as indicated in Table 3 defines n as the number of constraints and k as the number of variables (witness size).

Step	Operation	Complexity	Notes
1	Trace transformation	$O(k)$	Serial parsing of transactions
2	Constraint encoding (R1CS)	$O(n)$	Linear in constraint count
3	Witness generation	$O(k)$	Depends on circuit width
4	Prover computation	$O(n \log n)$ (Groth16), $O(n)$ (PLONK)	FFTs and MSMs dominate
5	Return π	$O(1)$	Proof is constant-sized

Table 3. Complexity analysis of proof generation

4.4 Aggregation of Transactions

The zk-SNARK proofs are aggregated, allowing multiple transactions to be represented by a single compact proof. This aggregation process significantly reduces the data that needs to be submitted to the Layer 1 blockchain.

Algorithm 3: zk-SNARK Proof Generation (Groth16-based)

Input: $\mathcal{C}$: Arithmetic Circuit
$\vec{x}$: Public inputs
$\vec{w}$: Private witness values
$\mathcal{PK}$: Proving key (from trusted setup)
Output: π: zk-SNARK proof attesting to $\mathcal{C}(\vec{x}, \vec{w}) = 0$
Step 1: Witness Assignment
Compute intermediate wire values for $\mathcal{C}$ using $\vec{w}$;
/ Complexity: $O(n)$ for n constraints */*
Step 2: R1CS Transformation
Convert $\mathcal{C}$ into Rank-1 Constraint System (R1CS) format;
/ Typically done during setup, reused here */*
Step 3: QAP Mapping
Translate R1CS into Quadratic Arithmetic Program (QAP);
/ $O(n)$ with precomputed polynomials */*
Step 4: Randomness Sampling
Sample trapdoor elements $r, s \in \mathbb{F}_q$ for zero-knowledge;
Step 5: Proof Construction
Compute:

- $A = \mathcal{PK}_A(\vec{w}, r)$
- $B = \mathcal{PK}_B(\vec{w}, s)$
- $C = \mathcal{PK}_C(\vec{w}, r, s)$

Assemble proof: $\pi = (A, B, C)$;
Return: zk-SNARK proof π

Algorithm 4: Transfer Transactions Processing

```
procedure TRANSFER()
    while Coordinator receives a transfer transaction T do
        EVENTLIST.add(T_i)
        if (DB(T_ac, T_dp) == TRUE) then
            T_txn.add(T_i)
        end if
    end while
    if B is ready to be submitted to L1 then
        B = {SR_pre, SR_post, H_T_txn}
        L1.submit(B)
    end if
end procedure
```

4.5 On-Chain Validation and Verification

Once the zk-SNARK proofs are generated and aggregated, they are submitted to the Layer 1 blockchain. This is typically achieved by sending a transaction or interacting with a smart contract on the Layer 1 blockchain. The Layer 1 blockchain's smart contract acts as a verifier. It checks the validity of the zk-SNARK proofs submitted by the zk-rollup and ensures that the proofs represent correctly executed transactions with valid state changes.

The complexity analysis breakdown as indicated in Table 4 clearly shows the efficiency of ScaZK that uses Groth16 due to the constant proof size, minimal pairing operations and efficient on-chain verifier (gas-optimised). Migrating to PLONK helps in higher expressivity and universal setup, but the on-chain verification is expensive in this case.

Prover	**Verifier Complexity**	**Pairings**	**On-chain Gas Cost (approx.)**	**Notes**
Groth16	$O(1)$	3	$\sim$ 220 000–300 000 gas	Very fast, constant-time
PLONK	$O(\log n)$	1–2	$\sim$ 450 000–600 000 gas	Depends on number of constraints
Marlin	$O(\log n)$	1–2	$\sim$ 700 000+ gas	With polynomial commitment checks

Table 4. Theoretical Time Complexity (Per SNARK) and Gas Cost on Ethereum L1 (Estimated)

Algorithm 5: On-Chain zk-SNARK Verification (Groth16)

Input: $\pi = (A, B, C)$: zk-SNARK proof
$\vec{x}$: Public inputs
$\mathcal{VK}$: Verifying key (on-chain constant)
Output: `true` if proof is valid, `false` otherwise
Step 1: Input Consistency Check
Ensure that $\vec{x}$ is well-formed and of expected length;
/ Typically checked via ABI or on-chain contract guard */*
Step 2: Linear Combination
Compute $K_{pub} = \sum_i \vec{x}_i \cdot \mathcal{VK}_i$;
/ Maps public inputs to elliptic curve points */*
Step 3: Pairing Product Check
Evaluate the pairing equation:

$$e(A, B) \stackrel{?}{=} e(\mathcal{VK}_\alpha, \mathcal{VK}_\beta) \cdot e(K_{pub}, \mathcal{VK}_\gamma) \cdot e(C, \mathcal{VK}_\delta)$$

Use Ethereum's precompiled contract at address `0x08`;
Step 4: Return Result
return `true` if pairing holds, `false` otherwise

4.6 On-Chain Settlement

After successful verification, the Layer 1 blockchain updates its global state to reflect the state changes from the zk-rollups. This process finalises the transaction and settles the state changes on the Layer 1 blockchain. Once the zk-rollups' state is settled on the Layer 1 blockchain, users can confirm that their transactions have been processed successfully by inspecting the Layer 1 blockchain's transaction history or interacting with their Layer 1 addresses. Withdrawals happen at this phase as in Figure 10.

5 PERFORMANCE EVALUATION AND DESIGN TRADE-OFFS

To rigorously evaluate the ScaZK protocol, we analyse its behaviour under variable loads, assess the security guarantees under realistic adversarial conditions, and explore the scalability–security–decentralisation trade-offs that define its architectural decisions. Figure 11 shows the interaction between components in ScaZK (Prover, Sequencer, Verifier), their data flows, and potential attack points.

5.1 Security Implications

ScaZK leverages zk-SNARKs (specifically Groth16) to guarantee succinct and sound off-chain computation verification. The protocol's security is conditioned on:

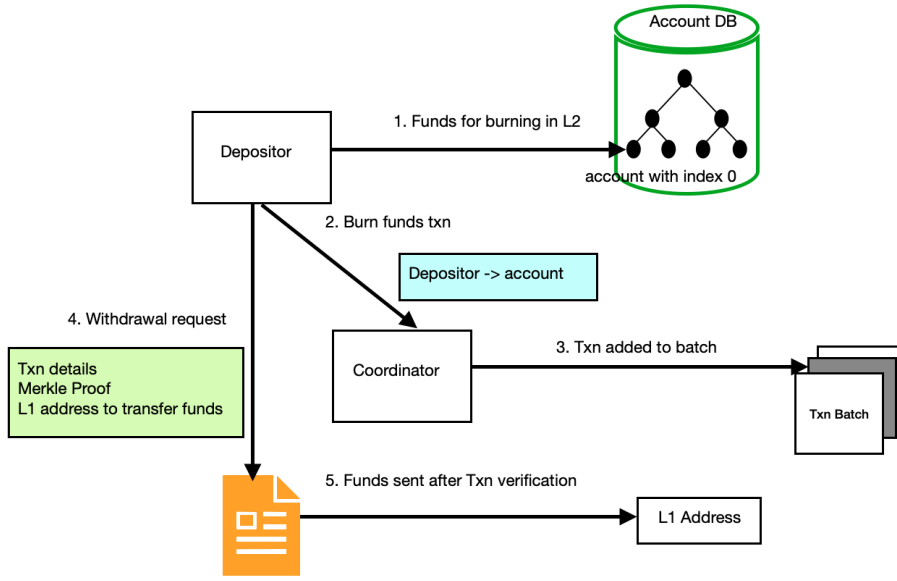


Figure 10. Withdrawals

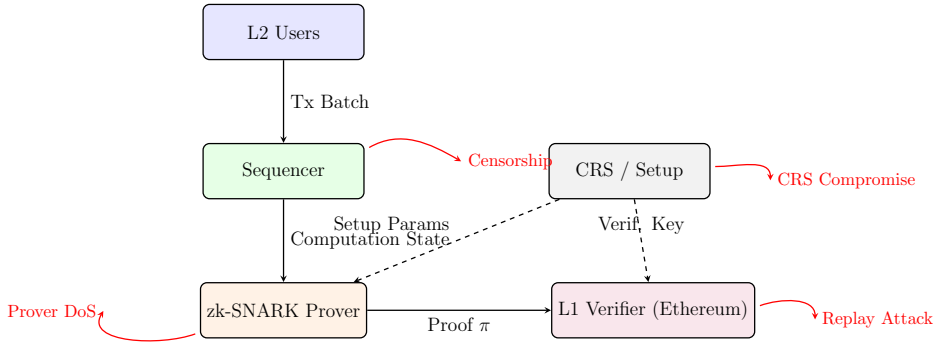


Figure 11. Threat model for ScaZK protocol highlighting attack surfaces and mitigation zones across off-chain and on-chain components

Soundness of zk-SNARKs: Proofs generated for invalid transitions will be rejected by the Layer 1 verifier.

Trusted Setup Assumption: Groth16 requires a common reference string (CRS). A compromise here can invalidate the soundness. This is mitigated through multiparty ceremonies.

State Binding: Proofs are bound to a unique Merkle root and public data, mitigating replay or equivocation attacks.

Failure Isolation: If a prover node fails, redundant prover nodes can reattempt proof generation. Malicious sequencers can be penalised via timeout-based slashing.

5.2 Performance under Load

We evaluate system behaviour under three primary stress dimensions: batch size, network congestion, and verifier throughput.

Batch Size Scaling: Proving time increases log-linearly with the number of transactions. However, proof size remains constant due to SNARK compression techniques. Table 5 summarises the results.

Layer 1 Network Congestion: Under high Ethereum gas conditions, on-chain verifier gas costs spike, and proof inclusion may be delayed. Mitigation strategies include EIP-4844 blob usage and proof commitment separation.

Verifier Saturation: Groth16 remains gas-efficient ($\sim$ 220 K gas) even during peak congestion, while protocols such as PLONK and Marlin exhibit significantly higher verification overhead (450 K–700 K gas).

Batch Size (tx)	Proving Time (s)	Proof Size (KB)	Gas (L1)	Peak RAM (MB)
8	1.2	153	225 000	320
64	2.7	153	225 000	700
256	5.9	153	225 000	1 400

Table 5. Scalability benchmarks under increasing batch sizes

The protocol's fault tolerance mechanisms are summarised in Table 6. Table 7 indicates the trade-off matrix for Scalability, Security and Decentralisation.

Failure Scenario	Recovery Mechanism	Impact
Prover crash or timeout	Redundant prover nodes retry proof generation	+2–4 min delay
Batch size exceeds max constraint limit	Batch split and requeued	Minor throughput drop
Invalid proof submission	Rejected by verifier; optional sequencer slashing	No state corruption
L1 congestion or failure	Retry on-chain inclusion; delay L2 finality	+1–3 blocks latency

Table 6. Failure modes and recovery strategies

Metric	Scalability	Security	Decentralization
Batch Size	Large (256+ tx)	Moderate	Supports large batches across nodes
Proof Type	Groth16 (Succinct)	CRS trusted setup	Single prover model
Sequencer Model	Centralized (Low latency)	Subject to censorship	May introduce bias
L1 Data Publishing	Compressed calldata	Full availability optional	Depends on finality delays
Fallback Mechanisms	Proof retry, timeout penalties	Prevents invalid state	Needs economic incentives

Table 7. Design trade-off matrix

6 EXPERIMENTAL SETUP

To evaluate the performance and reliability of the proposed ScaZK protocol, we implemented a testbed that closely reflects real-world Ethereum Layer 2 operating conditions. Below, we describe the hardware, software, and simulation configurations used in our evaluation.

6.1 Hardware Configuration

We used two classes of machines to reflect both centralised and decentralised prover settings:

Prover Node (High-End):

- CPU: 12-core AMD Ryzen 9 5900X @ 3.7 GHz
- RAM: 64 GB DDR4
- GPU: NVIDIA RTX 3090 (used for SNARK acceleration where supported)
- OS: Ubuntu 22.04 LTS

Sequencer/Verifier Node (Standard):

- CPU: 8-core Intel i7-10700 @ 2.9 GHz
- RAM: 32 GB
- OS: Ubuntu 20.04 LTS
- Ethereum Node: Geth (v1.12.1), connected to Goerli testnet

6.2 Software Stack

Circuit Compiler: `circom v2.1.4`

Proof Systems Tested: Groth16 (`snarkjs`), PLONK (`circomlib`), Marlin (via `arkworks-rs`)

Hash Functions: SHA256, Poseidon (for PLONK/Marlin)

Verifier Contracts: Solidity-based verifiers compiled with `solc v0.8.21`

Gas Estimation: Hardhat framework + manual execution on Goerli for exact cost capture

Simulators: Custom Python batch simulator to emulate high-throughput rollup transaction pipelines

6.3 Network Conditions Simulated

To reflect Ethereum mainnet congestion and Layer 2 load dynamics, we emulated the following:

Layer 1 base gas price: 20–120 gwei (static and dynamic scenarios)

Layer 2 throughput targets: 10, 100, 1 000 tx/s

Prover delay injection: Randomised latency of 0–3 seconds per batch

Transaction batch size: 8, 64, 128, 256 tx per proof

Bandwidth modelling: 20 Mbps upstream/downstream between prover and sequencer (average internet node)

6.4 Simulation Parameters

The simulation parameters and circuit benchmarks for the experimental setup of the ScaZK protocol are indicated in Table 8.

Parameter	**Value/Range**
Proof Systems Tested	Groth16, PLONK, Marlin
Hashing Circuits	Merkle Tree (depth 32), SHA256, Poseidon
Batch Sizes	$\{8, 64, 128, 256\}$ transactions
Gas Measurement	Solidity contract with Hardhat
Latency Simulation	Random 100–3 000 ms delay per proof
RAM Monitoring Tool	`htop`, `vmstat`

Table 8. Simulation parameters and circuit benchmarks

6.5 Evaluation Environment

Unless otherwise noted, all experiments were executed in a clean environment with minimal background processes. For consistency, circuits were compiled once and reused across prover types. Groth16 utilised pre-generated CRS from a trusted setup ceremony, and for PLONK and Marlin, structured reference strings (SRS) were generated using `arkworks-rs`'s universal setup tools.

7 RESULTS AND COMPARISON

The results of simulations carried out with the experimental setup mentioned were obtained for various parameters. The throughput is measured based on various factors such as parallelism, the consensus overhead, the communication overhead due to the participants involved in the Layer 2 transactions, Fault tolerance of various nodes participating, and the decentralising capabilities of the new protocol.

Using the data we have collected, we now compute the scalability potential of each system, taking into account the average cost per transaction of each Layer 2 system as well as the main-chain capacity. Given the observed average load and cost of each evaluated Layer 2 approach, along with the Ethereum transaction throughput and maximum gas capacity, we can now reason about the practical throughput and cost levels attainable by each approach and compare it with the theoretical throughput predicted in their respective white papers. Thanks to this study, we can now make some important conclusions regarding whether these approaches are sufficient in the long run or if other ideas are needed. While some of these systems may be able to handle large amounts of transactions in theory, this does not take main-chain capacity into consideration. Depending on how tightly each solution is coupled to the main-chain, the maximum throughput that can actually be reached can be substantially lower (that is, up to the point where the load imposed in the main-chain exceeds its capacity).

To make the computation of each method's projected scalability potential easier, we adopt two strategies. First, we do not take Cross-Chain Rollups into account because they are not yet available, and we do not have the data to predict their future impact. Second, even though certain processes might not be feasible to shift to a Layer 2 solution in practice (such as apps that rely solely on main-chain storage), we use the assumption that 100 % of the main-chain load can be moved to Layer 2 in order to simplify the analysis. Because of this, the throughput capacity that is shown is larger than what is actually achievable.

We compare the theoretical limit of each system as presented in its respective white paper with the estimated practical limit derived from our experimental data. The practical limit considers the maximum capacity determined by Ethereum's gas limit per block, as well as the resource demands that every system places on the network. Assuming an idealistic scenario where all Layer 2 systems could consume 100 % of the main-chain resources if they were the only Ethereum users, the practical limit is calculated. This scenario is therefore the best that any system could offer on its own under such idealised conditions.

Table 9, presents a comparison of the gas costs and withdrawal time of various protocols with the ScaZK protocol are established. ScaZK shows an improvement in costs in comparison with zkSync, which is due to the reduction in the total number of bytes used in preparing a batch of transactions in the rollup. Withdrawal time is also reduced as the Merkle tree insertion step for fund burning is eliminated. A graphical representation for this is shown in Figure 12 a). Fig-

ure 12 b) shows the chart on fee reduction for various protocols in comparison with ScaZK.

Technique	**Security guarantee**	**Gas Cost/ Batch (approx.)**	**Time for withdrawal**
zkSync	Yes	500 000	15 min
Optimism	Yes	40 000	7 days
ScaZK	Yes	467 000	12 min

Table 9. Gas costs and withdrawal time

Table 10 shows the maximum throughput comparison between the ScaZK protocol and the other relevant protocols. It clearly shows an improvement in terms of TPS under the zk-rollups variety of protocols. There is also a significant improvement in the finality of transactions due to the time gain in the withdrawal process.

Technique	**Max. Throughput (tps)**	**L2 Approach**	**Finality**
zkSync	3 000	Zk-rollups	1 hour
Optimism	200	Optimistic Rollups	7 days
ScaZK	4 000	Zk-rollups	50 min

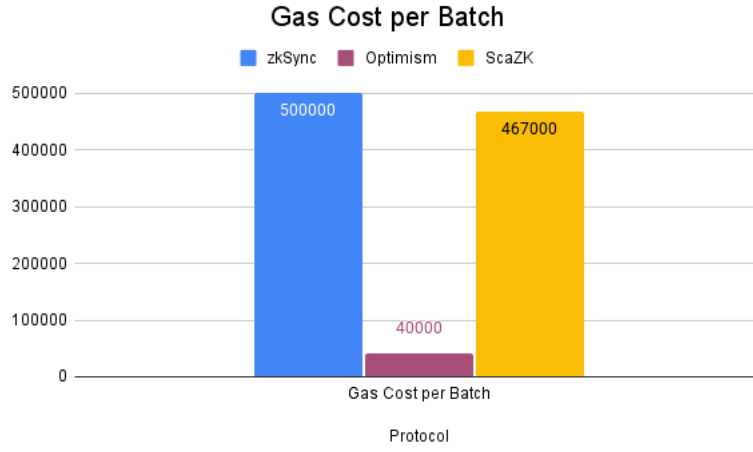
Table 10. Throughput (TPS)

Table 11 indicates that the SNARK implemented protocols display partial EVM compatibility, which is an improvement in comparison with zero compatibility of EVMs. As EVMs separately handle transactions in Layer 2 as rollup EVMs, this is an achievement in terms of throughput for transferring existing users to the ScaZK implementation. The Fees involved in the transaction processing also show a 100× improvement, as it directly translates to the bytes used in passing on a batch to Layer 1.

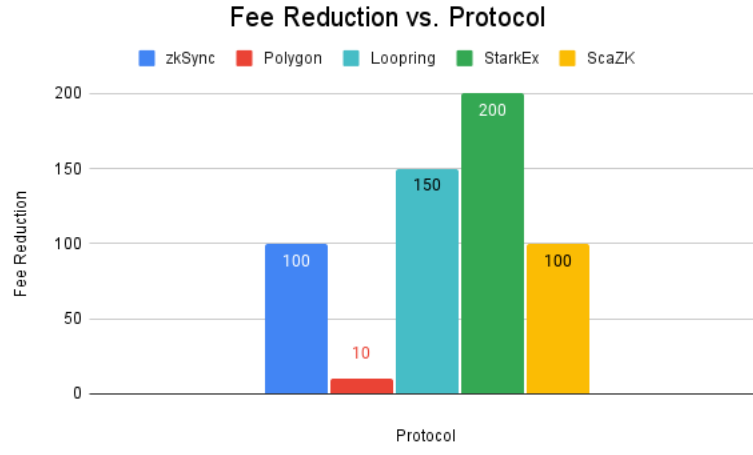
Implementation	**Validity Proof type**	**TPS**	**Fee Reduction**	**EVM Compatibility**
zkSync	SNARK	2 000	100×	Partial
Polygon	SNARK	2 000	10×	Partial
Loopring	SNARK	2 025	150×	No
StarkEx	STARK	3 000	200×	No
ScaZK	SNARK	4 080	100×	Partial

Table 11. Performance comparison between various zk-rollup implementations

The parallel processing capabilities considered were with respect to the transactions that can be processed in parallel by different nodes in the system. The more nodes there are, the greater the potential for parallelism, which can increase



a) Gas cost per batch

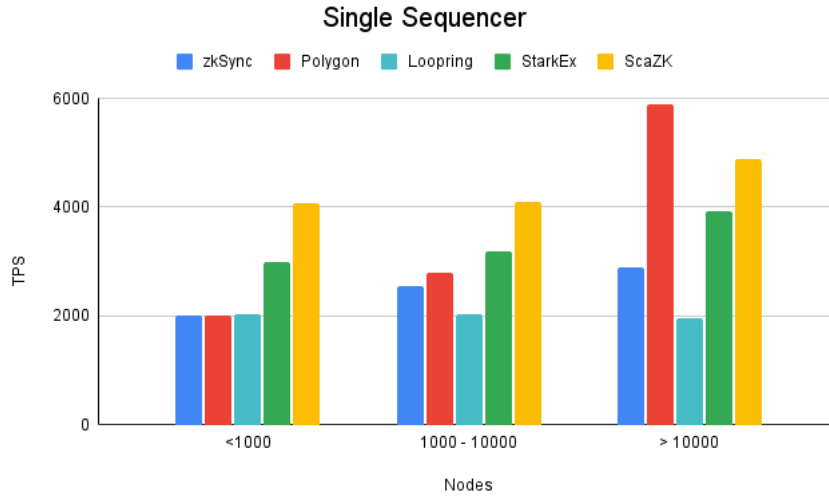


b) Fee reduction

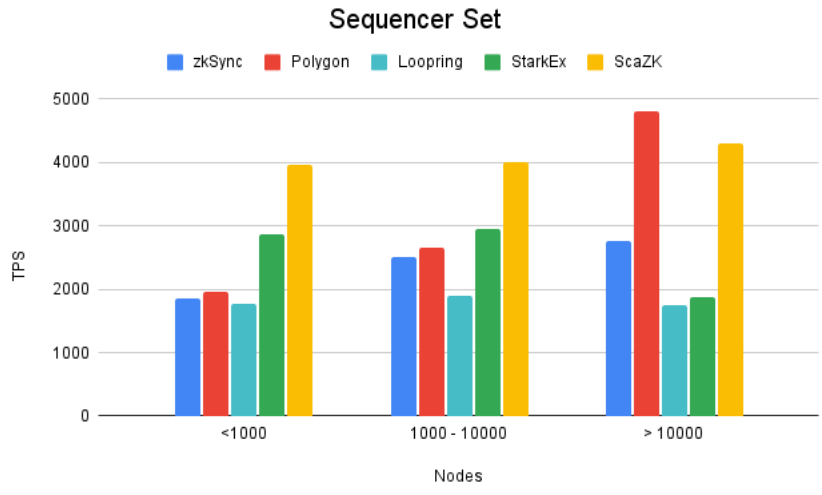
Figure 12. Protocols vs. gas

overall throughput. The results shown in Table 12 show that each node can independently process a portion of the transactions, making it possible to handle a higher transaction volume.

The consensus mechanism used in the case of sequencer sets, instead of a single sequencer, is also a factor that affects the scalability. This occurs because there is a consensus overhead with respect to the leader selection and communication between the sequencers for updates. Table 13 depicts the TPS analysis done on ScaZK against different rollup implementations. ScaZK shows a clear advantage as shown in Figures 13 a) and 13 b).



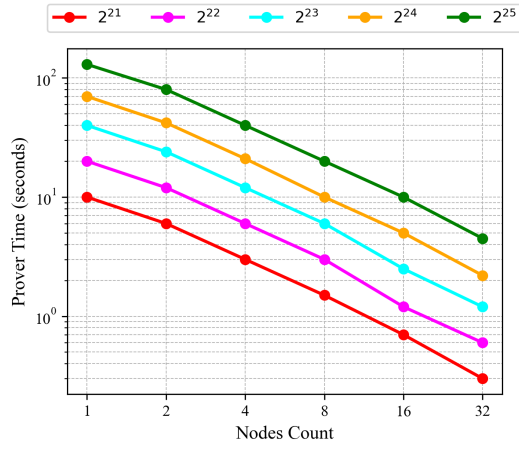
a) Single sequencer



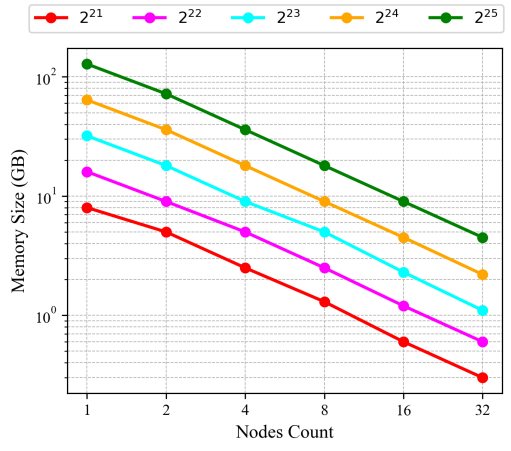
b) Sequencer set

Figure 13. Protocols vs. sequencers

The increase in nodes enhances the fault tolerance of the system, which is evident in the results obtained. A larger number of nodes might lead to increased communication overhead as nodes need to share information about transactions, proofs, and the current state of the system. The performance in terms of Prover times and Memory consumption is shown (Figures 14 a) and 14 b)), which depicts the efficiency of ScaZK for varying numbers of nodes and circuits.



a) Prover time



b) Memory consumption

Figure 14. Nodes comparison in circuit

Protocol	Reported TPS	Gas/Tx (approx.)	Verifier Gas (L1)
zkSync	2 000–3 000	∼ 12 000	∼ 250 000
Loopring	∼ 2 500	∼ 15 000	∼ 260 000
StarkEx	9 000+ (off-chain)	N/A	∼ 550 000
ScaZK	2 800–3 100	**9 421**	**226 817**

Values for proprietary protocols (Loopring, StarkEx) based on public whitepapers and approximations. Simulated under a uniform testbed.

Table 12. Comparison of zk-rollup protocols (estimated ranges)

Nodes	Sequencers	zkSync	Polygon	Loopring	StarkEx	ScaZK
< 1 000	Single	2 000	2 000	2 025	3 000	4 080
	Set	1 850	1 970	1 780	2 860	3 955
1 000–10 000	Single	2 550	2 800	2 020	3 200	4 100
	Set	2 500	2 650	1 900	2 950	4 000
> 10 000	Single	2 900	5 900	1 950	3 920	4 880
	Set	2 760	4 800	1 740	1 870	4 300

Table 13. Theoretical throughput analysis of different rollup implementations and the ScaZK protocol

k	Leaf Size (bytes)	Native Proof Size (bytes)	Generation Time (s)	Verification Time (s)	Proof Size (bytes)	Native Verification Time (bytes)
4	139	1 096	5.7772	0.0940	152 996	5.3183e−6
8	139	1 829	5.8816	0.0930	152 996	8.2737e−6
12	139	2 286	6.3720	0.0947	152 996	1.164e−5
16	139	4 399	6.3696	0.0960	152 996	1.868e−5
20	139	5 226	6.8266	0.0989	152 996	2.231e−5
24	139	5 544	6.8185	0.0997	152 996	2.468e−5
28	139	7 807	7.0227	0.1039	152 996	3.399e−5
32	139	8 449	7.0049	0.1033	152 996	3.620e−5
36	139	10 146	7.6580	0.1044	152 996	4.349e−5
40	139	10 038	7.5672	0.1043	152 996	4.352e−5
44	139	11 467	7.7270	0.1053	152 996	4.980e−5
48	139	13 430	7.7724	0.1064	152 996	5.663e−5
52	139	13 966	7.7834	0.1064	152 996	6.048e−5
56	139	14 486	7.8025	0.1067	152 996	6.386e−5
60	139	16 029	7.8501	0.1063	152 996	6.825e−5
64	139	17 424	7.8410	0.1067	152 996	7.300e−5

Table 14. Proof size analysis of the ScaZK protocol

Using Layer 2 systems is one possible way to get around the throughput limitations of permissionless blockchain systems. Despite previous studies on many aspects of Layer 2 systems, there has not been a thorough experimental analysis that assesses the short- and long-term effects these systems have on the main chain, nor a side-by-side theoretical and practical comparison of these systems. Even though Layer 2 systems are diverse, none of the solutions meet the expectations of the novel ScaZK protocol developed by this research because the load they place on the main-chain creates a significant bottleneck that prevents the systems from achieving their claimed maximum throughput levels. The improvement shown by the ScaZK protocol in terms of proof size is also analysed, as shown in Table 14.

We also observe that regardless of the underlying technology, human behaviour will always be the limiting factor for Layer 2 system performance. Performance is particularly impacted by the frequency of deposits and withdrawals, which demand regular main-chain contacts. To the best of our knowledge, no program or subsystem exists yet that encourages users to store their money in single Layer 2 systems rather than main-chain transfers. Given this, it is questionable whether the cost reductions of most Layer 2 systems outweigh the drawbacks, as many eventually offer considerably weaker security guarantees.

8 SUMMARY AND CONCLUSION

This work introduces ScaZK, a novel zkSNARK-based rollup framework that achieves significant improvements in scalability, latency, and proof size efficiency over prior systems such as zkSync and StarkEx. Our bucketed proof compression, B+ Merkle hashing, and zkEVM-aware design make ScaZK a practical and scalable alternative for real-world Layer 2 deployment.

While systems like zkSync and Loopring focus on batching and minimal calldata, they still suffer from proof growth and lack deep compression logic. StarkEx has excellent scalability via STARKs but incurs larger proof sizes, less Ethereum compatibility, and a closed prover model. ScaZK outperforms with smaller and compressible proofs, faster root computation, higher transaction throughput, lower latency (withdrawal reduced to 12 min), better EVM developer migration support and by being algorithmically transparent and modular.

It should not be surprising that strategies that depend more on centralised operators to circumvent the expenses linked to interacting with the main chain outperform others in this context, which threatens the Ethereum ecosystem's decentralisation (as highlighted by other works). This undermines the original goal of minimising dependence on centralised components in blockchain implementation.

We plan to extend our analysis in future work to include Layer 2 systems of additional ecosystems in blockchain, such as the Bitcoin Lightning Network, and use them in determining whether the observations we have made can be seen as trends in those ecosystems too. We also plan to create a synthetic workload from the collected logs, which will reflect the characteristics of the Layer 2 systems and enable us to regularly benchmark different approaches.

Future performance evaluation will incorporate stress tests under adversarial load (DoS, proof reordering), integration with alternative SNARKs (PLONK, Marlin), and cost modelling under EIP-4844 and Layer 3 recursion.

REFERENCES

[1] Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. 2008, `https://bitcoin.org/bitcoin.pdf` (Accessed 21 Jul 2021).

[2] Buterin, V.: A Next-Generation Smart Contract and Decentralized Application Platform. White Paper 3.37. 2013, https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf (Accessed 21 Jul 2021).

[3] Yang, F.—Zhou, W.—Wu, Q.—Long, R.—Xiong, N. N.—Zhou, M.: Delegated Proof of Stake with Downgrade: A Secure and Efficient Blockchain Consensus Algorithm with Downgrade Mechanism. IEEE Access, Vol. 7, 2019, pp. 118541–118555, doi: 10.1109/ACCESS.2019.2935149.

[4] Chen, L.—Xu, L.—Shah, N.—Gao, Z.—Lu, Y.—Shi, W.: On Security Analysis of Proof-of-Elapsed-Time (PoET). In: Spirakis, P., Tsigas, P. (Eds.): Stabilization, Safety, and Security of Distributed Systems (SSS 2017). Springer, Cham, Lecture Notes in Computer Science, Vol. 10616, 2017, pp. 282–297, doi: 10.1007/978-3-319-69084-1_19.

[5] De Angelis, S.—Aniello, L.—Baldoni, R.—Lombardi, F.—Margheri, A.—Sassone, V.: PBFT Vs Proof-of-Authority: Applying the CAP Theorem to Permissioned Blockchain. In: Ferrari, E., Baldi, M., Baldoni, R. (Eds.): Proceedings of the Second Italian Conference on Cyber Security (ITASEC 2018). CEUR Workshop Proceedings, Vol. 2058, 2018, https://ceur-ws.org/Vol-2058/paper-06.pdf.

[6] Xiao, Y.—Zhang, N.—Lou, W.—Hou, Y. T.: A Survey of Distributed Consensus Protocols for Blockchain Networks. IEEE Communications Surveys & Tutorials, Vol. 22, 2020, No. 2, pp. 1432–1465, doi: 10.1109/COMST.2020.2969706.

[7] Patel, O.: Blockchain Layer 2 Rollups, Optimistic Vs Zero Knowledge. International Journal of Advanced Research in Engineering and Technology (IJARET), Vol. 14, 2023, No. 2, pp. 13–29, doi: 10.34218/ijaret.14.2.002.

[8] Yazar, S.—Aslan, B.—Taşkin, D.: A Model Proposal and Implementation for Food Safety: Food Production Control Running on the Internet of Things and Human Interactive Blockchain. Turkish Informatics Foundation Journal of Computer Science and Engineering, Vol. 16, 2023, No. 1, pp. 11–22, doi: 10.54525/tbbmd.1149601 (in Turkish).

[9] Sanka, A. I.—Cheung, R. C. C.: A Systematic Review of Blockchain Scalability: Issues, Solutions, Analysis and Future Research. Journal of Network and Computer Applications, Vol. 195, 2021, Art. No. 103232, doi: 10.1016/j.jnca.2021.103232.

[10] Groth, J.: On the Size of Pairing-Based Non-Interactive Arguments. In: Fischlin, M., Coron, J. S. (Eds.): Advances in Cryptology – EUROCRYPT 2016. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 9666, 2016, pp. 305–326, doi: 10.1007/978-3-662-49896-5_11.

[11] Maller, M.—Bowe, S.—Kohlweiss, M.—Meiklejohn, S.: Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings. Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19), 2019, pp. 2111–2128, doi: 10.1145/3319535.3339817.

[12] Gabizon, A.—Williamson, Z. J.—Ciobotaru, O.: PLONK: Permutations over Lagrange-Bases for Oecumenical Noninteractive Arguments of Knowledge. 2019, https://eprint.iacr.org/2019/953.

[13] Chiesa, A.—Hu, Y.—Maller, M.—Mishra, P.—Vesely, N.—Ward, N.:

Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS. In: Canteaut, A., Ishai, Y. (Eds.): Advances in Cryptology – EUROCRYPT 2020. Springer, Cham, Lecture Notes in Computer Science, Vol. 12105, 2020, pp. 738–768, doi: 10.1007/978-3-030-45721-1_26.

[14] Stipsits, M.: Scalable Integration of Ethereum in a Microservice-Based Application Through Layer 2 Rollups. Master Thesis. University of Applied Sciences Campus Vienna, 2023, https://pub.hcw.ac.at/obvfcwhsacc/content/titleinfo/8874847/full.pdf.

[15] Singh, P.—Princi—Singh, M. K.—Verma, S. K.: Optimizing Layer-2 Scalability: An Analytical Analysis of zk-Rollups for Enhanced Transaction Throughput and Cost Efficiency. 2025 International Conference on Networks and Cryptology (NETCRYPT), 2025, pp. 1334–1338, doi: 10.1109/NETCRYPT65877.2025.11102481.

[16] Jain, A.—Punjabi, Y.—Mathur, R.: Exploring the Efficacy of Rollups: A Comparative Study of Optimistic and ZK-Rollups and Their Popular Implementations. Journal of Analysis and Computation (JAC), Vol. 18, 2024, No. 1, pp. 297–315, https://ijaconline.com/wp-content/uploads/2024/07/Yogita-Punjabi-NCASCE-3.0-02.pdf.

[17] Riyadh, M. M. H.—Saliken, M. S.—Annan, W.—Ghosh, A.—Fareeha, E.: A Comprehensive Framework for Evaluating Different Layer 2 Protocols in Ethereum. 2025, http://hdl.handle.net/10361/26717 (Bachelor's Thesis, BRAC University, Dhaka, Bangladesh).

[18] Khamis, J.—Rottenstreich, O.: Demand-Aware Channel Topologies for Off-Chain Blockchain Payments. 2021, https://eprint.iacr.org/2021/009.

[19] Ben-Sasson, E.—Bentov, I.—Horesh, Y.—Riabzev, M.: Scalable, Transparent, and Post-Quantum Secure Computational Integrity. 2018, https://eprint.iacr.org/2018/046.

[20] Ding, H.—Piao, X.—Song, H.—Li, J.—Ji, Z.—Liu, M.—Liu, J.—Dong, W.: Efficient and Verifiable Skyline Computation on Blockchain System with Merkle B+ Tree Index. 2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA), 2024, pp. 2113–2120, doi: 10.1109/ISPA63168.2024.00288.

K. N. Unnikrishnan is a prospective Research Scholar in blockchain in the Department of Computer Science and Engineering at the Indian Institute of Information Technology Kottayam, Kerala, India, that have been established as "Institutions of National Importance" by the Ministry of Education, Government of India. He has completed his B.Tech. in information technology from the Cochin University of Science and Technology, Kochi, India, and his M.Tech. in CSE from the Anna University, Chennai, India. He started his career in the software industry on application development with companies like IBM India Private Ltd., TCS, and Sonata Software, in Bangalore, India. He also worked with institutions like the Viswajyothi College of Engineering and Technology, Muvattupuzha, India and Adi Shankara Institute of Engineering and Technology, Kalady, India, where he was heading the respective departments as Associate Professor. Having a total of 24 years of experience, where 8 years were in industry and 12 years in academics, he has been a full-time researcher for the past 5 years, with a focus on the scalability of blockchain by significantly improving the trade-offs on the security, privacy and decentralised nature of blockchain. He is also involved in blockchain development by setting up private blockchains and developing decentralised apps on Ethereum for the qualitative analysis of his research.

P. Victer Paul is working as Assistant Professor in the Department of Computer Science and Engineering at the Indian Institute of Information Technology Kottayam, Kerala, India, the Institute of National Importance of India. He is also heading the Data Analytics and Business Decisions Research Group at IIIT Kottayam. He completed his Ph.D. in the Department of Computer Science (2015) from the Pondicherry Central University, Pondicherry, India. He is a recipient of the eminent INSPIRE fellowship from the Department of Science and Technology, New Delhi. He is a Winner of the MSME Idea Hackathon 2022, organised by MSME, Government of India and the Distinguished Alumni Award 2023 for Excellence in Academics from SMVEC, Puducherry. Currently, he is working with two Research/Innovative Projects funded under MSME, Government of India and the Core Research Grant scheme, SERB, Government of India. He has delivered invited talks on various technical topics and also organised an ATAL FDP program sponsored by AICTE, Government of India. He extended consultancy work to NeST Digital, Ernakulam, to train the technical developers and also published an Indian patent in 2022. He has 14+ years of academic, research and industrial experience and has published over 80 research articles in various international journals and conferences. Currently, he is working in the fields of artificial intelligence, optimisation and data analytics.