Analyzing and Benchmarking ZK-Rollups

Stefanos Chaliasos[1], Itamar Reif[2], Adrià Torralba-Agell[3], Jens Ernstberger, Assimakis Kattis, and Benjamin Livshits[1,4]

[1] Imperial College London, UK
[2] Astria, United States
[3] Universitat Oberta de Catalunya, Spain
[4] Matter Labs

Abstract. As blockchain technology continues to transform the realm of digital transactions, scalability has emerged as a critical issue. This challenge has spurred the creation of innovative solutions, particularly Layer 2 scalability techniques like rollups. Among these, ZK-Rollups are notable for employing Zero-Knowledge Proofs to facilitate prompt on-chain transaction verification, thereby improving scalability and efficiency without sacrificing security. Nevertheless, the intrinsic complexity of ZK-Rollups has hindered an exhaustive evaluation of their efficiency, economic impact, and performance.

This paper offers a theoretical and empirical examination aimed at comprehending and evaluating ZK-Rollups, with particular attention to ZK-EVMs. We conduct a qualitative analysis to break down the costs linked to ZK-Rollups and scrutinize the design choices of well-known implementations. Confronting the inherent difficulties in benchmarking such intricate systems, we introduce a systematic methodology for their assessment, applying our method to two prominent ZK-Rollups: Polygon zkEVM and zkSync Era. Our research provides initial findings that illuminate trade-offs and areas for enhancement in ZK-Rollup implementations, delivering valuable insights for future research, development, and deployment of these systems.

Keywords: Zero-Knowledge Proofs, ZK-Rollups, Benchmarking, Blockchain Scalability

1 Introduction

Blockchain technology, with leading chains such as Bitcoin [22] and Ethereum [37], has introduced novel solutions for finance and various applications, reshaping the landscape of digital transactions by removing the need for centralized entities. However, the surge in their adoption has brought to light a critical challenge: scalability. The inherent limitation in the number of transactions these networks can process per second has prompted an effort within the blockchain community to seek and develop a plethora of innovative solutions [40].

Two dominant strategies have enjoyed practical adaptation in recent years to address scalability. The first involves the creation of new, modern blockchains

designed from the ground up to process transactions more efficiently than their predecessors [38,5], albeit at the cost of missing the established security and network effects of blockchains like Ethereum. The second strategy revolves around Layer 2 (L2) solutions, or off-chain scalability solutions, with rollups being the most promising and widely adopted in practice [35]. Rollups work by executing transactions on a faster, secondary blockchain (L2) and then posting the resulting state root, along with transaction data, back to the main blockchain — Layer 1 (L1). This ensures the integrity of the rollup's state is verifiable and secure, leveraging the underlying blockchain's security.

Among the various rollup approaches, two stand out: optimistic [16] and ZK-Rollups [3]. Optimistic rollups rely on a system of trust and fraud proofs to validate state transitions, which introduces a delay in withdrawals due to the required challenge period. In contrast, ZK-Rollups utilize Zero-Knowledge Proofs (ZKPs) for immediate on-chain verification of state transitions, enhancing both scalability and efficiency without compromising the security of the L1 chain. Despite their advantages, ZK-Rollups introduce additional complexity and, to date, there has been limited research focused on a thorough evaluation of their overall efficiency, limitations, and economics.

Benchmarking ZK-Rollups presents a multifaceted challenge. The deployment of these systems is inherently complex, and their diverse design choices complicate direct comparisons. Further, identifying common payloads for benchmarking and establishing appropriate metrics are non-trivial tasks. In response to these challenges, this work embarks on a comprehensive theoretical and empirical analysis of ZK-Rollups. We dissect the operational and per-transaction costs of ZK-Rollups, examine the design decisions of prominent implementations, and propose a methodology for their benchmarking. This includes addressing the challenges inherent in benchmarking these systems, defining key research questions, and developing a reproducible methodology to ensure that our findings are publicly accessible and verifiable.

The results of this study aim to illuminate the trade-offs inherent to different ZK-Rollup implementations, offering insight into their advantages and areas in need of improvement. By providing a deeper understanding of the economics underpinning these systems, we hope to inform efforts to decentralize currently centralized systems. Furthermore, as Rollups as a Service continues to grow, our analysis seeks to arm users and buyers with the knowledge necessary to make their decisions, enabling them to compare different rollups using our benchmarking infrastructure tailored to their specific needs.

Research Questions. Next, we outline a series of research questions that will shape our analysis of ZK-Rollups. These questions are designed to uncover critical insights into the performance, cost structure, and overall efficiency and profitability margins of ZK-Rollups. Our investigation aims to provide a comprehensive understanding of these systems.

RQ1 Fixed Costs: What are the fixed costs associated with ZK-Rollups? Specifically, what are the expenses related to settling on L1, such as committing batches and verifying proofs? Additionally, what is (if any)

the constant proving cost per batch (e.g., aggregation or compressing a Scalable Transparent Argument of Knowledge (STARK) into a Succinct Non-interactive Argument of Knowledge (SNARK))?

RQ2 **Marginal Costs**: How long does it take to prove a batch, and what is the cost associated with proving a batch? What are the data availability (DA) costs in terms of bytes posted? How do state diffs compare to posting transaction data?

RQ3 **Trade-off Between Fast Finality and Cost Minimization**: In ZK-Rollups, achieving transaction finality requires verifying the proof of the batch that includes the transaction on L1. This necessitates producing the proof first. There is a trade-off between having large, compact batches that are slower to prove and smaller batches that are faster to prove but potentially lead to less amortization of costs.

RQ4 **Cost Breakdown**: How are costs distributed across different components of a ZK-Rollup transaction, including DA, proof generation, L1 posting, and verification? Understanding this distribution is crucial for identifying areas for optimization.

RQ5 **Impact of EIP-4844**: How has the introduction of EIP-4844 influenced the cost dynamics of ZK-Rollups? This question explores the effects of EIP-4844 on the cost efficiency and practicality of ZK-Rollups.

1.1 Contributions

- **Qualitative Analysis of ZK-Rollups' design choices:** We conduct a comprehensive theoretical analysis of ZK-Rollups, detailing the costs associated with processing transactions and examining the diverse design choices across different implementations.
- **Towards Benchmarking ZK-Rollups:** Addressing the significant challenges inherent in benchmarking ZK-Rollups, we develop and present a structured methodology for their evaluation. This includes the implementation of our benchmarking approach on prominent implementations such as Polygon ZK-EVM and zkSync Era, providing a blueprint for systematic assessment of ZK-Rollups' efficiency and costs.
- **Results and Insights:** Offering findings from our benchmarking efforts, we aim to contribute to the ongoing discourse on ZK-Rollups by identifying key factors that influence their development and pinpointing areas in need of improvement. These preliminary results are intended to guide future research and development efforts in the field.

2 Background

2.1 Scaling Blockchain and Rollups

Blockchain scalability has been a persistent challenge, particularly for established networks like Ethereum [37], which processes only tens of transactions per

second.[5] Efforts to enhance scalability have focused on two primary strategies: base layer scaling and L2 scaling solutions. Base layer scaling, which includes techniques such as sharding and novel consensus protocols, involves either the modification of existing blockchains — a complex and daunting task — or the development of new blockchain architectures. Although modern blockchains such as Solana [38] and Sui [5] have shown success, they often lack the established security, liquidity, and comprehensive ecosystem found in legacy blockchains like Ethereum.[6]

L2 scaling solutions, on the other hand, offer a promising avenue for scalability without altering the base layer, i.e., L1. Among these solutions, payment channels [1,2,19,34], Plasma [29], and rollups [35] have been the most prevalent solutions. Payment channels enable instant, bi-directional payments between two parties by establishing a network of interconnected channels, exemplified by Bitcoin's Lightning Network. However, they require capital lockup and constant base layer monitoring, making them suitable for specific, long-term use cases. Plasma attempted to solve various issues in different ways. Sguanci et al. [33] provides an overview of the main types of Plasma constructions. However, every attempt at Plasma had some trade-off that resulted in a poor user experience.

Rollups have emerged as hybrid L2 solutions, distinguishing themselves by offloading computation off-chain while retaining data on-chain, thus addressing the data availability issue while inheriting L1's security. Rollups batch and execute transactions on an auxiliary L2 blockchain that uses the same VM as L1 or a different one. This separation of transaction execution from consensus allows rollups to process significantly more transactions per second than their L1 counterparts. By submitting a summary of the rollup's state — typically, the root of a Merkle tree — to a smart contract in the underlying blockchain, rollups not only ensure data availability, but also inherit the security properties of the L1 network. Altering the L2 state recorded on L1 would require breaking the security of L1, making it both difficult and costly. This architecture enables rollups to offer an efficient, secure scaling solution for legacy blockchains. Notably, this model, i.e., rollup-centric scaling,[7] has gained traction as the *principal* method for scaling Ethereum, with two predominant variants: optimistic rollups [16] and ZK-Rollups [3].[8]

Optimistic rollups operate on a principle of trust, where state transitions are accepted without immediate verification, relying instead on fraud proofs to challenge incorrect state updates. This approach, while efficient, requires a challenge period, introducing a delay in withdrawals. In contrast, ZK-Rollups leverage ZKPs to verify state transitions on-chain, offering a more immediate

[5] https://l2beat.com/scaling/activity

[6] According to https://defillama.com/chains (accessed: 10/5/2024), Ethereum has 57.85% of the total TVL for all chains, while Solana has only 4.46%.

[7] https://ethereum-magicians.org/t/a-rollup-centric-ethereum-roadmap/4698

[8] As of 18/3/2024, rollups have more than 34B USD TVL according to https://l2beat.com.

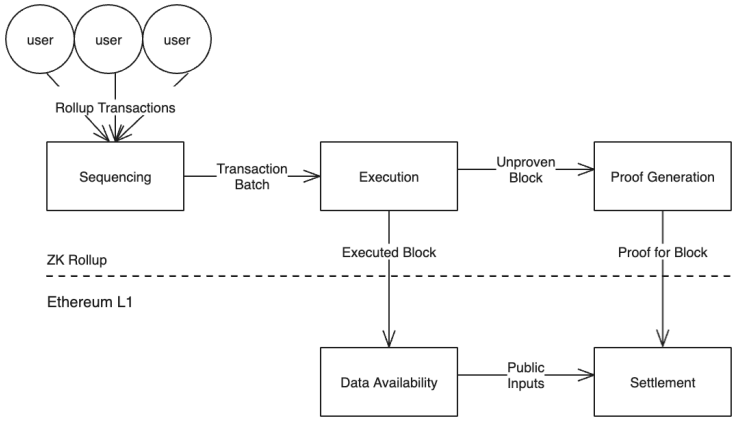


Fig. 1: High-level simplified overview of transaction processing in ZK-Rollups.

and efficient validation process without the need for a challenge period. This method not only improves the scalability and efficiency, but also maintains the integrity and security of the L1 chain.

2.2 ZK-Rollup Components and Transaction Lifecycle

In this section, we outline the main components of a ZK-Rollup. For simplicity, we abstract out certain details, including bridging and forced transactions [14]. In addition, we do not discuss various sequencing methodologies, such as decentralized or shared sequencers [21]. Figure 1 illustrates the key components involved in processing transactions within a ZK-Rollup. Users initiate the process by signing and submitting transactions to the L2 network. A sequencer then undertakes the tasks of processing, ordering, executing, and batching these transactions. In some architectures, these functions may be distributed across different components. Subsequently, the sequencer forwards these batches to a relayer, which posts their resultant state to the L1 rollup contract. Concurrently, the sequencer sends the batch to a coordinator (or aggregator), which, in turn, sends the batch to the prover. In most ZK-Rollups, the coordinator consolidates multiple proofs into a single aggregated proof (i.e., a proof of proofs) and submits this final proof to the rollup contract. The contract then verifies the proof, finalizing the state of the L2 as immutable and verified in the L1.

Components.

– **Sequencer.** The sequencer provides users with the first confirmation of transaction inclusion and ordering. It is the entity that aggregates user transactions into blocks and batches, either by providing them with a transaction submission endpoint or by pulling transactions from the mempool. It provides the canonical sequence of transactions that will be fed into the state

transition function. Currently, for the systems discussed in this work, sequencer implementations rely on a trusted operator that provides this service and to which users submit their transactions. Many of the projects discussed are working on decentralizing their sequencer design, but the design space remains nascent.

- **Execution.** For most existing rollups, execution is typically done by the same entity as the sequencer. The transaction batch created by the sequencer is taken as input to the rollup's state transition function (STF), which is executed to create both the resulting block(s) and related state root(s), as well as the transaction batches, and potentially the intermediate state snapshots (state diffs) that are required for proof generation.
- **Data Availability.** Data availability (DA) refers to the ability of clients of the blockchain protocol to retrieve the data required to verify the validity of a given batch. Traditionally provided as part of the consensus algorithm that underlies a blockchain system, the modular architecture used by rollup-based blockchain systems separates the guarantees provided by the L1 blockchain from those provided by the rollup operators. Relying on L1 for the rollup's liveness, the data required to assert safety must be posted on the L1. DA data involve any execution artifacts required by the settlement logic for verifying the validity proof, such as transaction data or the state diff of the transactions, as well as all data required to reproduce the state of the L2.
- **Prover.** The prover is responsible for generating validity proofs for the executed batches. Given the execution artifacts, the prover creates the required witness data and executes a SNARK or STARK proof to prove the validity of the execution. Note that when a STARK is used, it is typically wrapped into a SNARK to enable efficient verification.
- **Settlement Logic.** The generated proofs are then posted to the L1 smart contract responsible for settling a given batch. An executed batch must be agreed upon as "valid" in order to coordinate between decentralized actors (the blockchain's users). This is done by providing a block's resulting state, proof of that state's validity, and the inputs required for verifying the proof.

Transaction Lifecycle Status.

- **Pending:** A user has signed and submitted the transaction to the L2 network.
- **Preconfirmed:** The sequencer has processed the transaction and included it in a block. If users trust the sequencer, they can regard the transaction as processed. Currently, reliance on centralized sequencers enables near-instant preconfirmation, yet this raises the challenge of maintaining such efficiency without centralization. Preconfirmation significantly enhances user experience, allowing users to treat most transactions as effectively complete. However, it is important to note that for withdrawals from L2, users must await the finalization of transactions.

- **Committed:** The transaction is part of a batch committed to the L1 contract, allowing others to reconstruct the L2 state, including this transaction from the L1 data.
- **Verified/Finalized:** The batch containing the transaction has been proven, and the proof has been verified in the L1 contract, marking the transaction and its batch's state on L2 as immutable.

2.3 Costs of ZK-Rollups

Next, we analyze the costs associated with processing a transaction within a ZK-Rollup. Specifically, we distinguish between costs that are transaction-specific and those that are constant per batch, meaning they apply to each processed batch regardless of the number of transactions that are included in it.

We also separate out all costs associated with the transaction's fee mechanism, as these can be temporal in nature, and add an orthogonal dimension to the physical costs that are incurred per transaction (and which are a direct result of the system's design choices). To this end, we quantify costs in terms of the underlying empirical variable measured: for example, data availability costs are presented in terms of bytes rather than their actual L1 gas cost.

Fixed Batch Costs. Each batch carries inherent fixed costs that must be paid regardless of the number of transactions it includes.

1. **Settlement:** This involves (a) calling the Ethereum L1 contract to commit to a specific batch, and (b) submitting proofs and executing the verifier logic (e.g., SNARK verifier) for the committed batches.
2. **Proof Compression:** Some constructions involve compressing (or converting) the block's proof from one proof system to another. This typically involves proving the verification of the aggregated proof in a cheaper (with regards to the verification cost) proof system (e.g., Groth16) so that the cost of settlement is lower.

Marginal Transaction Costs. In addition to the batch-specific costs, each transaction included in a rollup's block incurs the following additional costs:

1. **Data Availability:** This is measured in bytes of the transaction's calldata. The calldata needs to be posted to the data availability provider, e.g., Ethereum L1, so that the rollup's state can be reconstructed.
2. **Proving Costs:** These are divided into the following: (a) Additional witness generation work required. (b) Proof generation for the transaction's execution. (c) Some constructions incorporate a final step, aggregating batches of proofs into a single proof. Additional transactions may require more aggregation work in this context.
3. **L2 Execution Costs:** These include computing the state transition resulting from the transaction along with any related costs due to associated long-term storage requirements. Can be thought of as the costs of operating the rollup's infrastructure: sequencing, execution, and relaying.

	ZK-Rollups					
	Polygon ZK-EVM	**Scroll**	**zkSync Era**	**Starknet**	**RISC0 Zeth**	**Aztec**
VM	zkEVM	zkEVM	zkEVM	General Purpose zkVM	General Purpose zkVM	Privacy-Focused zkVM
Proof System	STARK + FFLONK[1]	Halo2-KZG	Boojum + PLONK-KZG[1]	STARK + FRI	STARK + FRI	HONK + Protogalaxy + Goblin PLONK + UltraPlonk
Published Data	TX Data	TX Data	State Diffs	State Diffs	N/A	N/A
Compatibility	EVM-Compatible	EVM-Compatible	Solidity-Compatible	N/A	N/A	N/A
Hardware	CPU-based / >128-cores / >1TB RAM	GPU-based / 4 GPUs / >48-cores / >192GB RAM	Many GPU-based / 1 GPU / 16-cores / 64GB RAM	N/A	N/A	N/A

Fig. 2: High-level comparison of different ZK-Rollups. [1]: In those ZK-EVMs, the last step moves from a STARK-based proof system to a SNARK proof.

3 Qualitative Analysis

In this section, we examine the fundamental components and design choices influencing the performance, efficiency, and complexity of various ZK-Rollups. Figure 2 summarizes a qualitative overview of ZK-Rollups.

3.1 Proof System

Recent advancements in proof systems have led to a significant acceleration of ZK-VMs and ZK-EVMs, with research focusing on developing proof systems to optimize for better performance and efficiency of the proving algorithm. Notable developments include zkASM [28] and PIL [27] ZK languages, developed by Polygon for their ZK-EVM; Boojum [20], developed for zkSync Era; and the halo2 KZG fork [32] implemented by the Scroll team, among others. A common strategy in ZK-VM design is to also apply recursion, a technique in which one ZK proof is verified inside of another, allowing the usage of cheaper verification circuits at the settlement phase while using more complex proof systems in the proving process. For example, Polygon ZK-EVM leverages a STARK proof to initially prove batch correctness, which is then compressed via recursion before being encapsulated in a SNARK proof for submission to L1. This method benefits from SNARKs' efficient verification and constant proof size. Similar methodologies are used in various ZK-EVM platforms.

However, the choice of a proof system and its specific implementation can lead to some important trade-offs, particularly between the speed of proof generation and the computational resources required. This balance is crucial, as it can influence the overall performance and user experience of ZK-Rollups.

3.2 Transaction Data vs. State Diffs

In L1 blockchains, all transactions in a block are stored along with the Merkle root of the final state. This information is disseminated across the network

through a "gossiping" protocol [17], and the root of trust is established through re-execution and validation of the state root by the participants. For example, in proof-of-stake networks, validators stake their tokens and vote on the resulting state to ensure consensus [8].

ZK-Rollups, in contrast, establish their root of trust through the verification of a ZKP [3]. The ZKP attests to the correctness of the final state, and its validation is sufficient for participants to accept a batch of blocks as canonical. Verifying the final state requires publicly available inputs, typically either the transaction data included in the batch or intermediate state transition snapshots, known as "state diffs." Each method has its trade-offs. State diffs are more cost-effective because they omit signatures and publish only the final state changes after multiple transactions, thus allowing for better cost amortization. However, this approach does not preserve a complete transaction history and can complicate the mechanisms of enforcing transactions and reproducing the state through data posted in the L1. Currently, ZK-Rollups such as zkSync Era and Starknet utilize state diffs due to their efficiency benefits, while solutions like Polygon ZK-EVM and Scroll opt for publishing transaction data to maintain data completeness. Both approaches are exploring innovative compression techniques to further optimize cost efficiency.

3.3 EVM Compatibility

Buterin identifies four main categories of ZK-EVMs [7], which are implementations of ZKP circuits that validate the correctness of Ethereum Virtual Machine (EVM) execution, ranging from fully Ethereum-equivalent to language-compatible. Fully Ethereum-equivalent ZK-EVMs replicate the EVM's behavior and data structures precisely, ensuring seamless operation for existing Ethereum applications. EVM-equivalent ZK-EVMs maintain core functionalities but introduce slight variations in data structures while ensuring identical behavior when executing EVM-bytecode. EVM-compatible approaches might exclude certain precompiles or slightly modify the gas metering mechanism, which could affect the execution of specific transactions in edge cases.

The most flexible, language-compatible ZK-EVMs, utilize compilers to translate Solidity into different targets, optimizing efficiency and potentially enhancing functionality beyond strict EVM equivalence. This spectrum of compatibility reflects a trade-off between maintaining strict adherence to the EVM and pursuing efficiency gains or advanced features through innovation. For instance, the Polygon ZK-EVM aims for a close EVM equivalence to balance compatibility with performance improvements, while zkSync Era opts for a language-compatible approach with its `zksolc` compiler, prioritizing efficiency and adaptability.

Another approach, not described in Buterin's classification, is employed by RISC0's Zeth [31]. Based on a prover for the RISC-V Instruction Set Architecture (ISA), Zeth leverages Rust and its LLVM-based compiler toolchain to utilize

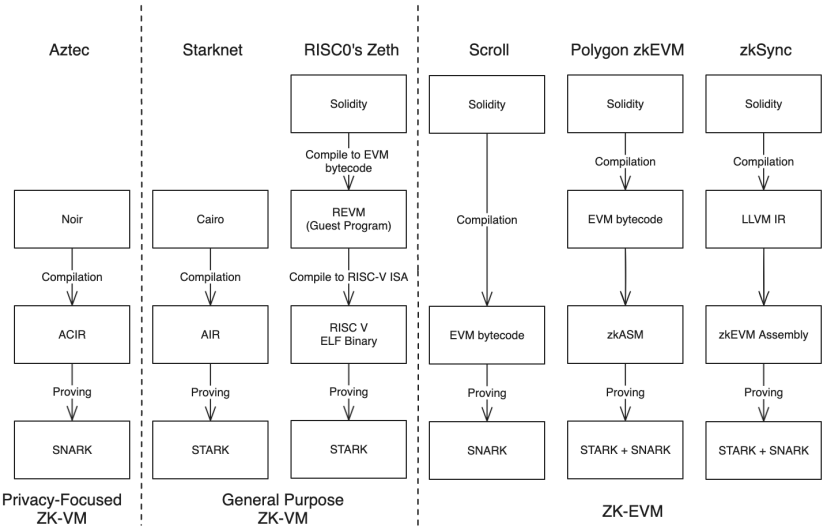


Fig. 3: High-level overview of ZK-Rollup compilation pipelines.

a suite of robust crates, such as revm[9], ethers[10], and alloy[11], enabling the proof of execution for EVM-based transactions and blocks without the need for additional domain-specific implementation circuits by leveraging RISC0's prover. This approach diverges from traditional ZK-EVM designs by proving the correctness of computations at the ISA level rather than focusing exclusively on EVM bytecode. The use of RISC-V as the underlying architecture allows for a high degree of flexibility and the potential to leverage a broader range of programming languages supported by the LLVM ecosystem. This approach results in a fully EVM-equivalent ZK-EVM.

The landscape of ZK-EVMs has seen a rapid evolution of innovative approaches in recent years, though not all are "EVM compatible" to the same degree. Several implementations discussed in this section were not included in our direct performance comparison due to differing definitions of compatibility. We further discuss "EVM compatibility in Appendix A.

3.4 ZK-Rollups Prover Implementations

At a high level, there are two primary approaches for ZK-Rollup prover design: assembly-based and EVM opcode-based implementations, each offering a distinct approach to handling computations and proofs. Assembly-based VMs implement a specific Instruction Set Architecture (ISA), focusing on proving the correctness of lower-level execution steps that express abstractions around the

[9] https://github.com/bluealloy/revm
[10] https://github.com/gakonst/ethers-rs
[11] https://github.com/alloy-rs/alloy

underlying proof system. This method closely aligns with traditional hardware architectures, where each instruction within the set is designed to perform well-defined atomic operations [13,31]. In contrast, opcode-based ZK-EVMs rely on specific circuits that each prove the execution of an EVM opcode or an EVM state transition, with the specific goal of proving the validity of the execution of an EVM transaction. Figure 3 provides an overview of the compilation and proving process followed by different ZK-Rollups.

One example of an assembly-based ZK-VM is Starkware's Cairo. Built as an abstraction on top of Algebraic Intermediate Representation (AIR), Cairo assembly generates polynomial constraints over a table of field elements that represent the state throughout the program's execution. This table serves as the trace (or witness) for the STARK-based prover, which then proves whether the trace satisfies Cairo's semantics [13]. While the Cairo framework allows one to generate application-specific circuits, in practice, it is used in Starknet to generate circuits for a single set of constraints for the von Neumann architecture-based Cairo CPU. The Cairo CPU is an AIR-generated STARK for a Cairo program that implements a register-based general-purpose VM.

Another example of an assembly-based implementation is RISC0's RISC-V-based ZK-VM. This approach involves compiling a Rust program to RISC-V ISA, referred to as the Guest Program. The Guest Program is executed to produce an execution trace, corresponding to the intermediate states of the RISC-V VM throughout the execution. The trace is then used as a witness by the RISC-V prover, which provides proof that the execution follows RISC-V semantics. Unlike Cairo, which uses a novel ISA tailored for STARK, RISC0 builds a prover for the existing RISC-V ISA. Using the LLVM compiler tool-chain, RISC0 can execute and prove any language that can be compiled to RISC-V ISA [30]. Using that approach, you can pass an EVM implemented in Rust as the guest program and prove EVM transactions, as demonstrated by Zeth [31] (c.f. Section 3.3).

In contrast to assembly-based ZK-VMs, the most common approach is to directly target EVM bytecode. While the former approaches rely on an intermediate ZK-VM for circuit implementation, many mature ZK-Rollups have chosen to implement specialized circuits that directly prove the execution of EVM bytecode, or close to EVM bytecode. Currently, all major ZK-Rollups beyond Starknet utilize specialized circuit-based ZK-EVM implementations.

For instance, zkSync Era, Polygon's zkEVM, and Scroll implement a ZK-EVM to prove the execution of transactions, but employ slightly different methods. Polygon and Scroll use Solidity's compiler to allow existing programs written in Solidity to be deployed, executed, and proved on a ZK-Rollup by implementing specialized circuits that validate EVM traces. In contrast, zkSync Era takes a higher-level approach by compiling Solidity directly to ZK-EVM bytecode (using zksolc), which is then executed and proved using circuits designed to verify the ZK-EVM bytecode's correctness instead of the EVM bytecode.

Another approach is Aztec's Abstract Circuit Intermediate Representation (ACIR) and Private Execution Environment (PXE)-based system. Aztec's smart contracts are written in Noir [24], a domain-specific language developed by Aztec

Labs for SNARK-based proving systems. Noir compiles to ACIR, an intermediate representation used to generate circuits for proving. Due to additional features provided by Aztec, specifically the support for nullifier-based private transactions, Aztec's execution model separates the handling of private data from the processing of public transactions. Transactions are first executed and proved locally by users inside the PXE, only propagating the public components of a transaction and a ZKP of the validity of its privately executed components to rollup nodes. This ZKP ensures the validity of private transactions, while the execution of their public components ensures the validity of the block.

3.5 Target Hardware and Prover Architecture

The hardware configurations required for ZK-VM providers vary significantly between projects, reflecting the diverse computational demands of their proof systems. Our analysis divides prover designs by target hardware and parallelization strategies into the following categories:

1. Single CPU-optimized implementations such as Polygon's ZK-EVM, which demands a high-capacity setup with a 96-core CPU and at least 768 GB of RAM.
2. Single GPU and CPU proving, such as Scroll's system, which parallelizes the execution of multiple blocks using a single GPU but then aggregates the proofs into a single proof that is posted on the chain using a CPU.
3. Cluster-based approaches: both zkSync and Risc0's systems rely on two stages. At first, the state transitions are divided into segments and proved in parallel, after which the proofs are aggregated and also in parallel. The key difference with Scroll's approach is that aggregation is also parallelized across a large cluster of GPUs or CPUs.

This highlights an essential consideration: Each ZK-VM implementation is meticulously optimized for specific hardware configurations, rendering direct performance comparisons on identical machines less meaningful. Furthermore, ongoing research on hardware acceleration for ZK-EVMs aims to further enhance proving times and system performance.

4 Experimental Evaluation

In this section, we delve into our methodology for benchmarking and analyzing the costs associated with ZK-Rollups. Benchmarking these systems is essential because it sheds light on the areas most in need of optimization. With many ZK-Rollups projects that aim to decentralize their core components, this analysis offers a timely opportunity to assess the costs involved and explore how these systems can achieve profitability and sustainability. Additionally, we consider projects interested in deploying specialized, application-specific rollups using existing infrastructure, aiming to discern the cost-related trade-offs of each stack.

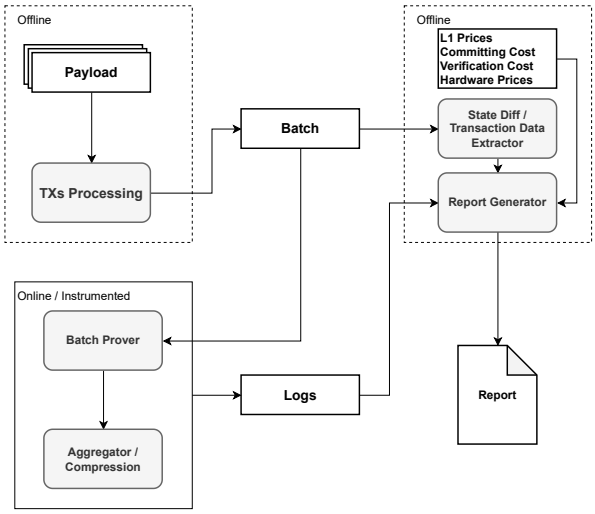


Fig. 4: Overview of the benchmarking procedure.

4.1 Benchmarking and Analyzing ZK-Rollups

Our analysis is designed to simplify the understanding of how external factors influence the costs and, consequently, the fees associated with ZK-Rollups. We begin by identifying the primary challenges in benchmarking ZK-Rollups, particularly focusing on their core component: the ZK-EVM. Then, we present our methodology and the decisions made to navigate these challenges, and finally, we answer the targeted research questions (c.f. Section 1).

Challenges and methodology. In this subsection, we outline the inherent challenges in benchmarking ZK-Rollups and detail our methodology for addressing these challenges. The benchmarking process, depicted in Figure 4, provides a high-level overview of how we evaluate ZK-Rollups.

Initially, transactions from the selected payload are processed using the sequencer and any auxiliary tools provided by the ZK-Rollup to create a batch. This step can be performed on standard hardware without the need for specialized, expensive machines. Subsequently, this batch is fed into the prover pipeline within an instrumented environment designed to capture the necessary metrics. The proving phase generally consists of at least two steps, though this can vary. The final phase involves calculating the total costs, incorporating L1 data, batch posting and batch verification costs on L1, and hardware expenses. By integrating both the offline and the online analyses, we generate a comprehensive report that breaks down the costs.

Metrics. Determining appropriate metrics is a fundamental challenge. We focus on straightforward metrics that facilitate a basic comparison between different

systems, making it easier for teams to integrate their ZK-EVMs into our benchmarking framework. In future work, we plan to expand our metrics to include power consumption, RAM usage, and GPU/CPU time. The primary metrics in this preliminary study are as follows:

- **Seconds per Proof**: The time required to generate a proof.
- **USD per Proof**: The cost of generating a proof, calculated by multiplying the clock time by the rate for using a cloud service like AWS or GCP.
- **USD per Proving a Transaction**: The cost associated with proving a single transaction.

Configuration. Choosing the right hardware configuration for benchmarking is another significant challenge. Ideally, different systems should be tested on identical or at least similar hardware specifications. However, this is nearly unfeasible for ZK-EVMs, as they are designed with distinct objectives in mind and optimized for vastly different hardware setups. Thus, we chose to benchmark the systems according to the hardware specifications recommended by each team for production use. This decision aims to capture the optimal cost-time efficiency based on each system's specific optimizations. Additionally, different proving systems have different requirements for resources (e.g., RAM) and different options for optimizations (e.g., through parallelization), meaning that it is even more difficult to compare to standardized machines.

Payloads. Deciding on the appropriate payloads for benchmarking is also challenging. Given the varying degrees of compatibility EVM between systems, selecting a common payload for comparative analysis is difficult. Moreover, the complexity of these systems further complicates setup and benchmarking efforts with specific payloads. Ideally, benchmarking would utilize historical Ethereum blockchain data, but the lack of necessary tooling in most ZK-Rollups complicates this approach. Instead, we focus on benchmarking common smart contract functionalities, including *native transfers*, *ERC-20 transfers*, *contract deployment*, *native Solidity/YUL hashing*, and the *Keccak precompile*.

These payloads represent typical blockchain operations, though future work may expand this list for more detailed analysis (e.g., token swaps, DAO voting, and NFT mints). Finally, note that we parameterized all payloads to different sizes, e.g., 1 ETH transfer vs. 10 ETH transfers vs. 100 ETH transfer, meaning that we have benchmarked 3 different batches of size 1, 10, and 100. Nevertheless, we recognize the necessity for additional research to develop more comprehensive workloads for benchmarking those systems.

Reproducibility. Ensuring reproducibility is crucial. Regardless of the metrics, configurations, and payloads, it is essential that the benchmarking process be designed so that third parties can validate the results. To accomplish this goal, we publish all configurations, scripts, and instructions used in our benchmarking process. This includes detailed specifications of the machines used, scripts for setting up the environment, and step-by-step instructions for running tests with the specified payloads. Our goal is for others to be able to replicate our findings

by following our documentation, thus reinforcing the validity and reliability of our results while being able to easily extend our benchmarks.

L1 Data. To obtain data from L1 for our analyses, we selected a period from April 10, 2024 (block `19621224`), to May 10, 2024 (block `19835630`). During this period, we crawled the main contracts of the selected ZK-Rollups and retrieved the required information: gas consumption for committing batches to L1 (excluding the gas cost related to Data Availability) and batch verification costs, i.e., the cost for verifying the proof and updating the smart contracts. This information is essential for comprehending fixed costs and is not connected to variable costs. Instead of our local payloads, we utilized on-chain data to determine the typical configurations employed by the rollups, including batch sizes.

Threats to Validity. Our study faces some validity threats that should be considered when interpreting the results. Firstly, our chosen payloads may not fully represent the range of real-world scenarios. We focused on providing easily executable payloads that are highly relevant, such as ETH and ERC-20 transfers, contract deployments, and SHA-256 hashes, commonly seen in other ZKP benchmarks [12]. However, this selection may overlook other important transaction types and interactions. Secondly, the results for certain components might be biased due to the specific payloads used. For instance, our payloads tend to favor state diffs as they primarily involve interactions with a single contract, which may not accurately reflect more complex and diverse batches. We have made efforts to clearly state any potential biases in the relevant sections of the paper. Finally, there are cases where parts of the systems are not fully open-sourced or could not be run in a controlled sandboxed environment. In those instances, we decided not to include our results for those systems in the analysis, as these results could not be independently verified by us. To ensure the quality and reliability of our findings, we narrowed our results to systems for which we could provide high-quality data.

4.2 Results

In this section, we present the results of our benchmarking and analysis of ZK-Rollups. We begin by detailing the systems we analyzed and the hardware configurations used for our tests. Following this, we dive into each research question, providing comprehensive answers based on our findings. Our goal is to offer information on the performance, costs, and trade-offs of different ZK-Rollup implementations.

Selected ZK-Rollups. For our analysis, we selected zkSync Era and Polygon ZK-EVM. The primary reason for choosing these two is that they are the only ZK-Rollups that are both EVM-compatible and fully open-sourced among the popular deployed ZK-Rollups. While Scroll is also EVM-compatible and widely used, we excluded it from our study because its GPU-prover is closed-source, which limits our ability to conduct a thorough and transparent analysis. By focusing on zkSync Era and Polygon ZK-EVM, we ensure that our benchmarking is based on systems with relatively stable code bases and deployed on a large

ZK-Rollup	Commit	Hardware Configuration	Hourly Cost
Polygon zkEVM	`d37e826`	r6i.metal (128 vCPUs, 1024GB RAM)	$8.06
zkSync Era	`4794286`	g2-standard-32 (32 vCPUs, 1 NVIDIA L4 GPU, 128 GB RAM)	$1.87

Fig. 5: Selected ZK-Rollups and Hardware Configurations. The Polygon ZK-EVM machine is hosted on AWS, while the zkSync Era is on GCP. Both machines have 1 TB SSD disk space. Note that spot prices could reduce the hourly cost significantly.

scale. Figure 5 summarizes the specific versions (commits) of the ZK-Rollups we analyzed and the hardware configurations used for our experiments. Note that we selected those machines after advising the teams developing the analyzed systems.

Benchmarking zkSync Era's Prover. We could only manage to run zkSync Era as a black-box system where multiple processes exchange messages, perform computations, and write logs. This is because Era is designed to run as a mini-cluster. Additionally, the instructions and configurations on how to set up this cluster are not publicly available. Since we were unable to run the prover components independently in a sandboxed environment and measure the performance ourselves, we decided not to include the prover time and cost in our analysis. Unfortunately, given the current state of the prover and its limited documentation, it was quite complicated to instrument the system for benchmarking as we intended. However, we were still able to reliably obtain proof compression time, and DA costs for Era. We leave as future work a fine-grained benchmarking of the Era prover.

4.3 Fixed Costs

The results presented in Figure 6 highlight the fixed costs for zkSync Era and Polygon ZK-EVM based on historical data from 10 April 2024 to 10 May 2024. The fixed costs encompass gas costs for committing and verifying batches, and proof compression costs. Note that zkSync Era has an additional transaction for finalizing transactions and enabling withdrawals, which we also consider a fixed cost. The two solutions employ different approaches: zkSync Era supports significantly larger batches, while Polygon ZK-EVM utilizes smaller batches and employs aggregation to derive a final proof. Both systems convert a STARK proof into a SNARK, incurring a fixed proving cost that remains constant regardless of the input size.

For zkSync Era, the data show a median batch size of 3,895 transactions, resulting in a median batch cost of $18.93 and a cost per transaction of approximately $0.0047. The large batch sizes in zkSync Era allow for the costs to be

	Metric	Era	Polygon
Batch Size	Median Batch Size (Min/Max)	3,895 (485/5,000)	27 (1/158)
	Theoretical Max Batch Size	5,000	498
Commit Batches	Median Gas Cost	230,686	324,088
	Median Batches Per Tx (Min/Max)	1 (1/1)	8 (1/15)
Verify Batches	Median Gas Cost	458,527	378,461
	Median Batches Per Tx (Min/Max)	1 (1/1)	20 (1/160)
Execute Batches	Median Gas Cost	3,303,634	-
	Median Batches Per Tx (Min/Max)	26 (4/45)	-
Proof Compression	Median Time in Seconds (USD Cost)	1,075 (0.56)	311 (0.70)
	Median Batches per Proof	1	20
Normalized Costs	Median Gas Per Batch (USD)	816,275 (18.37)	59,434 (1.34)
	Median Batch Cost USD	18.93	1.38
	Median Gas Per Tx (USD)	209 (0.00471)	2,201 (0.04962)
	Median Tx Cost USD	0.00486	0.05111
	Median Gas Per Tx – Full Batches (USD)	163 (0.00367)	119 (0.00268)
	Median Cost Per Tx – Full Batches USD	0.00378	0.00275

Fig. 6: Fixed costs breakdown for ZK-Rollups using historical data. The USD cost of gas is based on 7.5 Gwei, and an Ether price of $3,000. Compression costs are derived from Figure 5. For Polygon ZK-EVM we also include the batch proof aggregation costs in proof compression. In zkSync Era, the execution of batches signifies L1 finality, thus fully finalizing the batch and allowing fund withdrawals from the system. The numbers are crawled from on-chain data from 10 April 2024 to 10 May 2024.

distributed across a greater number of transactions, thereby reducing the cost per transaction. In contrast, Polygon ZK-EVM processes smaller batches, with a median batch size of 27 transactions. This leads to a median batch cost of $1.38 and a cost per transaction of $0.0511. Notably, in their ideal scenario where the batches are completely full, both systems can achieve negligible fixed costs per transaction (i.e., less than $0.004).

One critical insight derived from the analysis is that filling batches to their maximum capacity is essential to minimize fixed costs per transaction. This strategy allows the cost to be amortized over many transactions. However, it may negatively impact the finality if L2 does not have sufficient usage. Furthermore, even with enough usage, the marginal costs or proving time might increase, leading to slower finality. These trade-offs and their implications will be further examined in subsequent subsections.

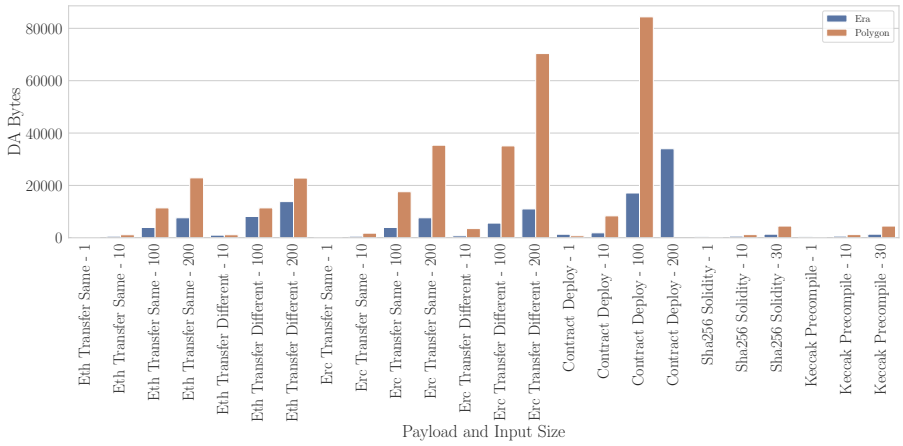


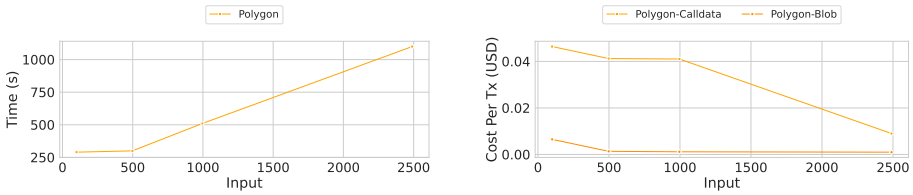
Fig. 7: Comparison of DA (Data Availability) requirements for various payloads between zkSync Era, which uses state diffs, and Polygon ZK-EVM, which posts transaction data.

4.4 Marginal Costs

The incremental costs of ZK-Rollups are essential to grasp their scalability and effectiveness. This section explores the duration required to produce a proof for a batch, the related expenses, and the data availability (DA) expenses in bytes needed for submission to the L1. Furthermore, we evaluate the efficiency of state differences as opposed to uploading full transaction data.

The design of the provers for Polygon and zkSync Era shows marked differences. The prover for zkSync Era is designed to support the processing of large batches, supporting up to 5,000 transactions, and operates on more affordable hardware. On the other hand, the architecture of Polygon's prover is geared towards quick-proof generation, albeit at the cost of requiring pricier equipment. We noticed that the time it takes for zkSync Era to generate proofs extends with larger input sizes. In Figure 11 in Appendix B, we show some preliminary measurements that demonstrate this pattern. In contrast, Polygon's prover maintains a consistent output time of either 190 or 200 seconds, independent of input size. While Polygon's prover offers speed advantages for smaller batches, this comes at the trade-off of requiring expensive hardware. Nevertheless, with increasing batch sizes, Polygon's prover becomes more cost-effective, as it takes the same time to prover different number of transactions. This demonstrates another difference in the design: Era's prover is more elastic as smaller batches will be faster, and larger ones will be slower, whereas in Polygon batches proving will be either 190 or 200 seconds.

Figure 7 illustrates the data availability (DA) needs in bytes for both Polygon and zkSync Era. A primary distinction is zkSync Era's use of state diffs, in contrast to Polygon's approach of posting entire transaction data. The state



(a) Proving and Compression Time (b) Cost Comparison

Fig. 8: Trade-off between proving/compression times (affecting finality) and cost per transaction for Polygon ZK-EVM.

Size	Fixed Costs			Marginal Costs			Total Cost Per Tx (USD)	
	Commit	Verify Proof	Comp.	Proving	DA-Blob	DA-Calldata	w/ Blob	w/ Calldata
100	44%/36%	52%/42%	1%/1%	3%/2%	0%	20%	0.16	0.20
498	44%/20%	52%/23%	1%/1%	3%/1%	0%	55%	0.03	0.07
996	43%/13%	50%/15%	1%/0%	5%/2%	0%	70%	0.02	0.06
2,490	40%/6%	46%/7%	2%/0%	12%/2%	0%	84%	0.01	0.05

Fig. 9: Cost breakdown for ETH transfers in Polygon ZK-EVM. The blob price per byte is 1 wei, the normal gas price is 7.5 Gwei, the ETH price is $3,000, and we use the costs of the machines from Figure 3 for the provers. We also capture the cost per byte, i.e., for blobs, we do not charge for the whole blob if it is less than the blob size. The first percentage is when we consider blobs as DA, and the second is for calldata.

diffs used by zkSync Era result in notably superior compression, making the DA requirements disparity more pronounced as the input size grows. The cost-effectiveness of zkSync Era's state diffs is due to this enhanced compression. It is crucial to acknowledge that our payloads are naturally advantaged by state diffs since our transactions usually engage with a specific contract, enhancing compression potential. Although the actual difference may be smaller in practical scenarios, it remains considerable. Future investigations into this subject are recommended.

4.5 Trade-off Between Fast Finality and Cost Minimization

Figure 8 demonstrates the trade-off between fast finality and cost minimization in Polygon ZK-EVM, using benchmarks of 100, 498, 996, and 2490 ETH transfers. As depicted in Figure 8, Polygon's prover time increases linearly. Note that, as mentioned before, Polygon's prover always takes 190-200 seconds per batch. So, to prove 996 transactions, we are required to prove two batches and aggregate them. Further, the cost per transaction amortized better when using larger inputs, demonstrating a trade-off between fast proving (i.e., faster finality) or processing larger inputs and achieving better cost-effectiveness.

	Size in Bytes		Calldata		Blobs	
Input	**Era**	**Polygon**	**Era**	**Polygon**	**Era**	**Polygon**
ERC-20 Transfers (200)	10,999	70,357	$23.05	$136.96	3.3×10^{-11}	2.11×10^{-10}
Contract Deployment (100)	17,087	84,369	$35.97	$182.43	5.1×10^{-11}	2.53×10^{-10}
ETH Transfer (2,490)	88,693	283,905	$194.53	$661.95	2.66×10^{-10}	8.52×10^{-10}

Fig. 10: Impact of EIP-4844. The gas cost used is 50 Gwei (representing pre-Dencun upgrade prices), the ETH price is $3,000, and the blob gas cost is 1 wei. Note that for blobs, typically, you have to pay for a full blob, but here, we compute the cost only for the required bytes.

4.6 Cost Breakdown

Figure 9 provides a detailed cost breakdown for ETH transfers across different batch sizes, showing the distribution of fixed and marginal costs. For small batch sizes, fixed costs such as committing, verifying, and proof compression dominate the total cost per transaction. As batch sizes increase, DA costs become more significant, particularly when blobs are used instead of calldata. In addition, for larger batch sizes, proving costs become increasingly relevant.

One interesting observation is that Polygon has optimized for fast proving time, leading to very fast and relatively cheap proving prices. In our preliminary investigation of Era, we noticed that orthogonal to Polygon, they have optimized more for data compression. In both systems, proof compression takes a fraction of the cost 1% and can be ignored, demonstrating that the use of such compression techniques is essentially free of non-trivial additional computational burdens.

In the context of blob pricing, proving time takes up merely 5% of the total cost incurred on Polygon. Furthermore, we observe that where blobs are used for DA, marginal costs of proving over the underlying batch size grow at a higher rate. When using calldata as DA, the situation changes: in this case, the marginal costs for calldata usage are non-trivial for Polygon. Nevertheless, the cost of proving remains at almost negligible levels (less than 2% of the cost of a full batch) at all batch sizes when using calldata DA. However, the vast majority of costs in Polygon (around 97% of the total cost for a batch of 2,490 transactions) stem from its DA requirements to store full transaction records.

4.7 Impact of EIP-4844

The introduction of EIP-4844 has significantly influenced the cost dynamics of ZK-Rollups, particularly in terms of DA costs. As illustrated in Figure 10, EIP-4844 has generally minimized DA costs by allowing for more efficient data posting. However, it is important to note that for blob transactions, rollups are required to pay for an entire blob of approximately 125 KB.

This implies that for rollups required to frequently submit small batches to L1, utilizing calldata could be more economically viable than using blobs. This underscores the importance of additional research into the existing constraints of

blobs and how the market might evolve as demand grows. Moreover, the present low pricing of blobs has not fully realized market price discovery. It is important to consider that sustained low prices might establish impractical expectations, which could result in substantial price hikes as the market realigns.

More specifically, given a shift to blob-based pricing that significantly reduces DA costs for ZK-Rollups across the board, the recorded results allow for projecting on the relative costs of transaction processing at large batch sizes for each of the two assessed systems.

Given that Polygon has not yet implemented blob-aware logic, we expect that any marginal changes to its transaction cost model will occur after the new DA market matures. To this end, the ratio of the relative marginal costs of proving on Era and Polygon respectively goes from $20\times$ ($= 39\%/2\%$) to $5\times$ ($= 52\%/12\%$), suffering a $4\times$ drop. This is indicative that, upon subsidizing DA costs through blobs, the large relative DA cost for transactions on Polygon is much less pronounced (relative to Era) than before. This can be expected as the Era prover is "already" more constrained by proving (given its more efficient use of state-diffs) and thus sees a lower marginal cost benefit from the DA repricing. This is also reflected in how marginal proving costs go from 39% to 52% (increase by a third) on Era due to blobs, while they almost triple (from 2% to 5%) on Polygon.

The above implies that Polygon could marginally benefit more by blob repricing, lowering its DA cost bottleneck, and moving in the direction of proving its main (marginal) cost. The latter regime seems to already be the case with Era where, although the new blob pricing system will provide benefits, they will not be affecting the substantial proving costs and thus will have lower relative impact. As DA costs continue to fall, we also expect that both (all) ZK-Rollups systems converge to respective "maximal" marginal proving costs (relative to DA), at which point DA will be a comparatively much smaller part of the overall cost structure and will not need to be further subsidized.

4.8 Lessons Learned

Below we revisit the original research questions from Section 1.

- **Trade-off between Fast Finality and Cost Efficiency (RQ1–RQ3)**: Our experiments highlight a primary trade-off between achieving fast finality (i.e., rapid proving time) and maximizing cost efficiency through cost amortization. Filling batches fully allows for better cost amortization per transaction, impacting both fixed and marginal costs. While fixed costs become less significant for large batches, they dominate the costs for smaller batches, making efficient batch filling crucial for overall cost efficiency. This creates a dynamic where rollups perform better at higher utilization levels, somewhat paradoxically. This insight raises questions about designing an optimal metering mechanism for ZK-Rollups.
- **State Diffs Efficiency (RQ2)**: Our preliminary benchmarking indicates that state diffs are highly cost-efficient in reducing data availability costs

for ZK-Rollups. By publishing only the state changes instead of complete transaction data, state diffs offer significant compression benefits, especially as the input size increases.
- **Importance of Sequencing (RQ4)**: The proving process benefits greatly from efficiently filling batches with transactions. It is essential to match the transactions to the capabilities of the underlying prover. For example, Polygon's prover can handle a limited number of Keccak operations per proof. Proving a single Keccak costs the same as proving the maximum number of Keccak operations the prover can handle in a batch. Therefore, sequencing transactions to fit the prover's optimal capabilities can significantly reduce the cost per transaction. This highlights the importance of developing sophisticated sequencing strategies and suggests a need for further research into the topic to guide the development of efficient sequencing for ZK-Rollups.
- **Impact of EIP-4844 (RQ5)**: The Dencun upgrade has substantially reduced DA costs, enabling near-zero cost transactions for ZK-Rollups. However, as blob prices may increase in the future, optimizing ZK-Rollups for cost efficiency remains critical. Additionally, strategies such as blob-space sharing or selectively using calldata instead of blobs during periods of high blob prices or for small batches could be viable options under certain circumstances and specific requirements.

5 Open RQs and Future Work

In this section, we present the open questions that remain unanswered in our current work and sketch a roadmap for future research. These questions not only underscore the complexities inherent in ZK-Rollups but also highlight areas that require further exploration.

- **Decentralization's Impact on Costs**: A pivotal question revolves around how decentralizing L2 core components will influence transaction processing costs. Specifically, we seek to understand whether the decentralization will lead to negligible cost implications or if it will significantly alter the economic landscape of ZK-Rollups. Another related open question but beyond the scope of this work is L2 MEV and cross-chain MEV.
- **Batch Size and Sequencing Optimization**: The size of transaction batches is a critical factor affecting proving costs. Future iterations of our work will delve into how variations in batch size and executed opcodes in a batch influence the cost per transaction and the overall proving time. While current systems may operate with an optimal batch size tailored to their specific needs, emerging forks (e.g., app-chains) may require adjustments to accommodate different priorities. This analysis aims to provide valuable insights for optimizing batch size and contents in response to evolving requirements.
- **Metering Mechanism Evaluation**: Another area of interest is the examination of the metering mechanisms employed by ZK-EVMs, which traditionally mirror those of the EVM [37]. Given that proving certain EVM

opcodes might be relatively more costly than their execution, we plan to investigate potential discrepancies in pricing. Through micro-benchmarks, we will explore whether such mismatches pose significant challenges, such as the under-pricing of specific opcodes that could lead to DoS attacks.
- **Proving Market Mechanisms**: We also intend to explore various proving market mechanisms and assess how they might influence the cost dynamics of proof generation [36]. This exploration could shed light on potential economic models conducive to more efficient and cost-effective proving processes.
- **Throughput Limitations**: Identifying the maximum transactions per second (TPS) each rollup can achieve, based solely on proving and DA on Ethereum and excluding other market dynamics, is another critical inquiry. This analysis will help quantify the scalability limits of current ZK-Rollup implementations.

In addition to addressing these open questions, our future work will expand the scope of our benchmarks. We aim to conduct micro-benchmarks at the opcode level to gain a finer-grained understanding of proving costs. Moreover, we plan to introduce macro benchmarks with diverse payloads beyond those presented in this study. Furthermore, by replaying blocks of Ethereum on different ZK-Rollups, we aspire to provide deeper insight into their performance and cost-efficiency. We also plan to analyze ZK-VM-based ZK-Rollups and compare them with native ZK-Rollups to shed more light on the debate between specialized circuit implementations versus using generic VMs to produce ZKPs.

6 Related Work

Benchmarking Blockchains and EVM Implementations. Benchmarking blockchains has garnered significant attention from both academic and practitioner communities. Gramoli et al. [15] developed a comprehensive benchmark suite for six popular blockchains with smart contract capabilities, conducting an extensive evaluation using five realistic decentralized application payloads across various configurations for each blockchain. Their primary objective was to assess latency and throughput. In a subsequent study, Nasrulin et al. [23] introduced Gromit, a tool focusing on blockchains with various consensus algorithms, with both studies focusing on the overall performance of the blockchains under examination rather than specific aspects such as execution node implementations.

In this work, our focus shifts to benchmarking the core components of ZK-Rollups, paralleling efforts such as those of Cortes-Goicoechea et al. [10], who benchmarked the five most prominent Ethereum consensus clients to evaluate their resource consumption. Similarly, Zhang et al. [39] employed simple microbenchmarks to compare the efficiency of WASM EVM nodes against Geth and Openethereum. Our future endeavors include conducting micro-benchmarks to gain deeper insights into ZK-EVM implementations. Furthermore, both academic and industry efforts have explored benchmarking EVM nodes using either straightforward macro benchmarks [11], like ERC-20 transfers, or fuzzy techniques [25] to assess the performance characteristics of EVM implementations

comprehensively. Mirroring this approach, we utilized macro benchmarks to evaluate the performance of ZK-EVM implementations, with plans to adapt tools such as flood [25] for future ZK-EVM benchmarking. Busse et al. [6] evaluate EVMs on various machines to pinpoint any noticeable differences.

While our current analysis does not extend to testing various machines for each ZK-Rollup, our objective is to perform such analyses to determine the most cost-effective hardware configurations for running each ZK-EVM and to identify the most optimized setup for each prover. Lastly, Perez, and Livshits [26] evaluated the EVM's metering mechanism, identifying potential DoS attack vulnerabilities. Inspired by their findings, we plan to conduct stress tests on ZK-EVM implementations to uncover any mispricing in the proving costs of specific EVM opcodes.

Benchmarking ZKPs. Benchmarking efforts for ZKPs play a crucial role in improving the understanding and performance of cryptographic libraries and primitives. Benarroch et al.[4] highlighted the inherent challenges and outlined best practices for implementing benchmarks for ZKP Domain Specific Languages (DSLs). Building upon this foundation, our work outlines the specific challenges of benchmarking ZK-Rollups, particularly focusing on ZK-EVMs, and proposes a comprehensive methodology to tackle these challenges. Ernstberger et al. [12] introduced zk-Bench, a detailed benchmarking framework and estimator tool designed for evaluating the performance of low-level public-key cryptography libraries and SNARK DSLs.

Our research complements these efforts by focusing on ZK-EVMs, which represent some of the most complex systems employing SNARKs. In addition to these efforts from academia, the Celer Network published a blog post[12] that benchmarks the time and memory costs of proving SHA 256 circuits across various ZKP tools. Parallel to our efforts, Delendum has developed a framework[13] dedicated to benchmarking ZK-VMs. An interesting future work could be to compare the performance of ZK-VM-based ZK-EVMs versus native solutions.

Analyzing Rollups and ZK-Rollups. Chaliasos et al. [9] provided a taxonomy of security issues that might occur in systems utilizing ZKPs including ZK-Rollups. Thibault et al. [35] have conducted an extensive survey on the use of rollups as a scalability solution for the Ethereum blockchain, discussing the various types, highlighting key implementations, and offering a qualitative comparison between optimistic rollups and ZK-Rollups. Koegl et al. [18] have compiled a comprehensive list of known attacks on rollup systems, shedding light on their potential impacts. In contrast, our work delves into ZK-Rollups, providing a detailed overview and explanation of their characteristics through a qualitative analysis. In addition, we focus on the empirical benchmarking of ZK-Rollups and the meticulous analysis of their associated costs. This dual approach not only enriches the understanding of ZK-Rollups as a scalability solution but

[12] https://blog.celer.network/2023/07/14/the-pantheon-of-zero-knowledge-proof-development-frameworks/

[13] https://github.com/delendum-xyz/zk-benchmarking

also underscores the economic viability and technical challenges of implementing ZK-Rollups.

7 Conclusions

In this paper, we have undertaken a comprehensive analysis of ZK-Rollups, focusing on their scalability, efficiency, and economic implications. Our theoretical and empirical evaluations of Polygon ZK-EVM and zkSync Era reveal the inherent trade-offs in their design and implementation, providing critical insights into their operational costs and performance.

Addressing the challenges of benchmarking ZK-Rollups, we have developed a structured methodology that allows for a thorough evaluation of these systems. Our results highlight possible improvements and suggest directions for future research. This research not only improves the understanding of ZK-Rollups, but also aims to guide and influence the development of more efficient rollups.

Acknowledgements

We thank Ramon Canales, Emil Luta, Roman Brodetski, Robert Remen, Fractasy Romero, Ignasi Ramos, Héctor Masip Ardevol, and Carlos Matallana for their insightful feedback and help with technical issues. We thank the Ethereum Foundation and TL;DR Research for supporting this work. We thank Cristina Pérez-Solà for proofreading this manuscript and supervising Adrià Torralba-Agell while working on this project. This work is also partially supported by the project PID2021-125962OB-C31 SECURING/CYBER, funded by the Ministerio de Ciencia e Innovación, la Agencia Estatal de Investigación and the European Regional Development Fund (ERDF).

References

1. Elli Androulaki, Christian Cachin, Angelo De Caro, and Eleftherios Kokoris-Kogias. Channels: Horizontal scaling and confidentiality on permissioned blockchains. In *European Symposium on Research in Computer Security*. Springer, 2018.
2. Lukas Aumayr, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. Blitz: Secure multi-hop payments without two-phase commits. In *USENIX Security Symposium*, 2021.
3. Barry Whitehat. Roll up token. https://github.com/barryWhiteHat/roll_up_token, 2018. Accessed: 2024-03-19.
4. Daniel Benarroch, Aurélien Nicolas, Justin Thaler, and Eran Tromer. 'community proposal: A benchmarking framework for (zero-knowledge) proof systems. *QEDIT, Tel Aviv-Yafo, Israel, Tech. Rep*, 2020.
5. Same Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, et al. Sui lutris: A blockchain combining broadcast and consensus. *arXiv preprint arXiv:2310.18042*, 2023.

6. Anselm Busse, Jacob Eberhardt, and Stefan Tai. Evm-perf: high-precision evm performance analysis. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–8. IEEE, 2021.
7. Vitalik Buterin. The different types of zk-evms. https://vitalik.eth.limo/general/2022/08/04/zkevm.html, 2022. Accessed: 2024-03-19.
8. Vitalik Buterin. *Proof of stake: The making of Ethereum and the philosophy of blockchains*. Seven Stories Press, 2022.
9. Stefanos Chaliasos, Jens Ernstberger, David Theodore, David Wong, Mohammad Jahanara, and Benjamin Livshits. Sok: What don't we know? understanding security vulnerabilities in snarks. *arXiv preprint arXiv:2402.15293*, 2024.
10. Mikel Cortes-Goicoechea, Luca Franceschini, and Leonardo Bautista-Gomez. Resource analysis of ethereum 2.0 clients. In *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pages 1–8. IEEE, 2021.
11. Ziyad Edher. evm-bench. https://github.com/ziyadedher/evm-bench, 2024. Accessed: 2024-03-19.
12. Jens Ernstberger, Stefanos Chaliasos, George Kadianakis, Sebastian Steinhorst, Philipp Jovanovic, Arthur Gervais, Benjamin Livshits, and Michele Orrù. zk-bench: A toolset for comparative evaluation and performance benchmarking of snarks. *Cryptology ePrint Archive*, 2023.
13. Lior Goldberg, Shahar Papini, and Michael Riabzev. Cairo–a turing-complete stark-friendly cpu architecture. *Cryptology ePrint Archive*, 2021.
14. Jan Gorzny, Lin Po-An, and Martin Derka. Ideal properties of rollup escape hatches. In *Proceedings of the 3rd International Workshop on Distributed Infrastructure for the Common Good*, pages 7–12, 2022.
15. Vincent Gramoli, Rachid Guerraoui, Andrei Lebedev, Chris Natoli, and Gauthier Voron. Diablo: A benchmark suite for blockchains. In *Proceedings of the Eighteenth European Conference on Computer Systems*, pages 540–556, 2023.
16. Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S Matthew Weinberg, and Edward W Felten. Arbitrum: Scalable, private smart contracts. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1353–1370, 2018.
17. Lucianna Kiffer, Asad Salman, Dave Levin, Alan Mislove, and Cristina Nita-Rotaru. Under the hood of the ethereum gossip protocol. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II 25*, pages 437–456. Springer, 2021.
18. Adrian Koegl, Zeeshan Meghji, Donato Pellegrino, Jan Gorzny, and Martin Derka. Attacks on rollups. In *Proceedings of the 4th International Workshop on Distributed Infrastructure for the Common Good*, pages 25–30, 2023.
19. Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Srivatsan Ravi. Concurrency and privacy with payment-channel networks. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 455–471, 2017.
20. MatterLabs. Boojum. https://github.com/matter-labs/era-boojum, 2022. Accessed: 2024-03-19.
21. Shashank Motepalli, Luciano Freitas, and Benjamin Livshits. Sok: Decentralized sequencers for rollups. *arXiv preprint arXiv:2310.03616*, 2023.
22. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
23. Bulat Nasrulin, Martijn De Vos, Georgy Ishmaev, and Johan Pouwelse. Gromit: Benchmarking the performance and scalability of blockchain systems. In *2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pages 56–63. IEEE, 2022.

24. noir contributors. `noir` zksnark language, 2022. URL: https://aztec.network/aztec-nr/.
25. Paradigm. Flood. https://www.paradigm.xyz/2023/06/flood, 2023. Accessed: 2024-03-19.
26. Daniel Perez and Benjamin Livshits. Broken metre: Attacking resource metering in evm. *arXiv preprint arXiv:1909.07220*, 2019.
27. Polygon. Pil. https://docs.polygon.technology/zkEVM/spec/pil/, 2022. Accessed: 2024-03-19.
28. Polygon. zkasm. https://docs.polygon.technology/zkEVM/spec/zkasm/, 2022. Accessed: 2024-03-19.
29. Joseph Poon and Vitalik Buterin. Plasma: Scalable autonomous smart contracts. *White paper*, pages 1–47, 2017.
30. RISC-Zero. Risc zero vm. https://github.com/risc0/risc0, 2022. Accessed: 2024-03-19.
31. RISC-Zero. Zeth. https://www.risczero.com/blog/zeth-release, 2022. Accessed: 2024-03-19.
32. Scroll. halo2 kzg. https://github.com/scroll-tech/halo2, 2022. Accessed: 2024-03-19.
33. Cosimo Sguanci, Roberto Spatafora, and Andrea Mario Vergani. Layer 2 blockchain scaling: A survey. *arXiv preprint arXiv:2107.10881*, 2021.
34. Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. A 2 l: Anonymous atomic locks for scalability in payment channel hubs. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1834–1851. IEEE, 2021.
35. Louis Tremblay Thibault, Tom Sarry, and Abdelhakim Senhaji Hafid. Blockchain scaling using rollups: A comprehensive survey. *IEEE Access*, 10:93039–93054, 2022.
36. Wenhao Wang, Lulu Zhou, Aviv Yaish, Fan Zhang, Ben Fisch, and Benjamin Livshits. Mechanism design for zk-rollup prover markets. *arXiv preprint arXiv:2404.06495*, 2024.
37. Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.
38. Anatoly Yakovenko. Solana: A new architecture for a high performance blockchain v0. 8.13. *Whitepaper*, 2018.
39. Yixuan Zhang, Shuyu Zheng, Haoyu Wang, Lei Wu, Gang Huang, and Xuanzhe Liu. Vm matters: A comparison of wasm vms and evms in the performance of blockchain smart contracts. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 2024.
40. Qiheng Zhou, Huawei Huang, Zibin Zheng, and Jing Bian. Solutions to scalability of blockchain: A survey. *Ieee Access*, 8:16440–16455, 2020.

A EVM Compatibility Properties

The term "EVM compatibility" encompasses various properties, including:

- Support for the standard Solidity compilation toolchain, allowing existing Solidity contracts to be ported over without additional work.
- Compliance with Ethereum's exact state transition logic.
- Adherence to Ethereum's gas cost metering mechanism.
- Adherence to Ethereum's JSON-RPC client API.

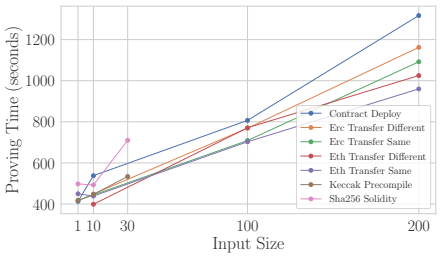


Fig. 11: Proving time increases depending on input size for the Era prover. Note that this result is not complete and does not count for potential parallelization.

– Support for Ethereum's smart contract standards (e.g., ERC-20) and precompiles (e.g., `keccak`).
– Support for Ethereum's existing wallet infrastructure.
– Support for Ethereum's existing development infrastructure.

B Proving Time for Era

Figure 11 provides an overview of the increase of proving time in zkSync Era given different batch sizes of various inputs. This highlights a design orthogonal to Polygon ZK-EVM where, for each batch, the time needed to be proved is either 190 or 200 seconds. It is important to note that both systems optimize for various aspects. For example, zkSync optimizes for data compression through its state diffs mechanism, whereas Polygon optimizes for fast proving times. It is important to note that Figure 11 is not complete. Due to the complexity of the system, its modular architecture, and the lack of documentation, we did not manage to measure the complete proving time precisely. Finally, note that the setup for Era can be improved by: (1) using more machines to parallelize computation and reduce finality time, (2) using cheaper machines for non-GPU computations, e.g., witness generation, to reduce proving costs.