HOCHSCHULE
MITTWEIDA
University of Applied Sciences

MASTERTHESIS

Mr.
Tai Nguyen Nhan, B.Sc.

Ethereum Scaling Solutions: Exploring Zero-Knowledge Ethereum Virtual Machines and Their Applications

Mittweida, December 2024

Faculty of **Applied Computer Sciences & Biosciences**

MASTERTHESIS

Ethereum Scaling Solutions: Exploring Zero-Knowledge Ethereum Virtual Machines and Their Applications

Author:
Tai Nguyen Nhan

Course of Study:
Blockchain & Distributed Ledger Technologie

Seminar Group:
BC22w1-M

First Examiner:
Prof. Dr.-Ing. Andreas Ittner

Second Examiner:
Dipl.-Volkswirt Mario Oettler

Submission:
Mittweida, 29.12.2024

Defense/Evaluation:
Mittweida, 2025

Bibliographic Description

Nguyen Nhan, Tai:
Ethereum Scaling Solutions: Exploring Zero-Knowledge Ethereum Virtual Machines and Their Applications. – 2024. – 65 S.
Mittweida, Hochschule Mittweida – University of Applied Sciences, Faculty of Applied Computer Sciences & Biosciences , Masterthesis, 2025.

Abstract

This master thesis provides a comprehensive analysis of five Zero-Knowledge Ethereum Virtual Machines (zkEVMs) developed atop Ethereum, with the objective of identifying the most suitable solution for specific use cases. The analysis covers key dimensions, including technical architecture, security, ecosystem, governance, and cost considerations. The findings highlight that Linea and Scroll are particularly well-suited for startups prioritizing robust ecosystem support and resources. Taiko and Polygon zkEVM demonstrate strong compatibility with existing Ethereum applications, making them ideal for projects focusing on seamless migration. Taiko further excels in security and decentralization, making it a preferred choice for applications such as NFT marketplaces. ZkSync proves to be the most effective solution for low-cost, high-volume trading platforms. Lastly, ZkSync, Polygon zkEVM, and Scroll are identified as excellent options for projects aiming to actively participate in governance and contribute to shaping the ecosystem. This thesis aims to serve as a practical guide for stakeholders navigating the zkEVM landscape.

Contents

Contents I

Additional Lists IV

1 Introduction 1

2 Research Question 2

3 Ethereum Virtual Machine 3
3.1 Architecture of EVM . . . 3
3.1.1 World State . . . 4
3.1.2 Virtual ROM . . . 5
3.1.3 Program Counter . . . 5
3.1.4 Gas available (Gas) . . . 5
3.1.5 Stack . . . 6
3.1.6 Memory . . . 6
3.2 Transaction flow through the EVM . . . 6

4 Zero-Knowledge Proofs 7
4.1 ZkRollup . . . 7
4.2 Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs) . 8
4.3 Zero-Knowledge Scalable Transparent Arguments of Knowledge (zk-STARKs) . . . 8
4.4 Circuits . . . 9
4.5 Limitations . . . 9

5 Zero-Knowledge Ethereum Virtual Machine 11
5.1 Challenges in building a zkEVM . . . 11
5.2 Advancements in zkEVM Development: From Polynomial Commitments to Hardware Acceleration . . . 11
5.3 zkEVMs Types . . . 12
5.3.1 Type 1 (Fully Ethereum-equivalent) . . . 12
5.3.2 Type 2 (fully EVM-equivalent) . . . 13
5.3.3 Type 2.5 (EVM-equivalent, except for gas costs) . . . 13
5.3.4 Type 3 (almost EVM-equivalent) . . . 13
5.3.5 Type 4 (High-Level-Language Equivalent) . . . 13

6 Different zkEVMs 14

7 Architecture 16
7.1 Scroll . . . 17
7.1.1 Roller Network . . . 18
7.1.2 Rollup and Bridge Contract . . . 19
7.1.3 Workflow . . . 19
7.2 Taiko . . . 20
7.2.1 Based Rollups . . . 20

7.2.2 Multi-Proof System . . . 21
7.2.3 ZkVM . . . 23
7.2.4 Contestable Rollup . . . 24
7.2.5 Booster Rollups . . . 26
7.2.6 Inception Layer . . . 26
7.3 Linea . . . 27
7.3.1 Prover . . . 28
7.4 Polygon ZkEVM . . . 30
7.4.1 zkProver . . . 31
7.4.2 Aggregration Layer . . . 33
7.5 ZkSync . . . 34
7.5.1 Account Abstraction . . . 34
7.5.2 ZkPorter . . . 35
7.5.3 Compiler Toolchain . . . 36
7.5.4 Zk HyperChains . . . 37

8 Security 39
8.1 Audits . . . 39
8.2 Bug Bounty Program . . . 39
8.3 Emergency Response Mechanisms . . . 41
8.4 Sequencer and Proposer . . . 42

9 Ecosystem 44
9.1 Metrics . . . 44
9.2 Decentralized Application . . . 46
9.3 Ecosystem Incentives . . . 47
9.3.1 Grants and Investing . . . 47
9.3.2 Technical Support & Networking . . . 48
9.3.3 Community Programs . . . 48

10 Governance 50
10.1 Scroll . . . 50
10.2 Taiko . . . 50
10.3 Linea . . . 51
10.4 ZkSync Era . . . 51
10.5 Polygon zkEVM . . . 51
10.5.1 System Smart Contract Governance . . . 52
10.6 Summary . . . 52

11 Smart Contracts 53
11.1 Bridging . . . 54
11.2 Development . . . 54
11.2.1 Taiko . . . 54
11.2.2 Linea . . . 55
11.2.3 ZkSync . . . 55
11.2.4 Scroll . . . 56
11.2.5 Polygon zkEVM . . . 56
11.3 Cost . . . 56

12 Szenarios **58**
12.1 Scenario 1: Startup Building a Decentralized Web3 Application 58
12.2 Szenario 2: A company wants to build a Trading plattform 59
12.3 Szenario 3: A company wants to migrate its Ethereum native application to a zkRollup 60
12.4 Szenario 4: A company wants to build a Decentralized NFT Marketplace 61
12.5 Scenario 5: A Project Wants to Participate in the Governance to Make Decisions that Benefit Their Project 61

13 Summary **63**

14 Fazit **65**

Bibliography **66**

Additional Lists

List of Figures

3.1 From Source Code to CPU. . . . 3
3.2 Ethereum Virtual Machine Architecture. . . . 4
4.1 Working of a zkRollup. . . . 8
4.2 A simple example of an arithmetic circuit. . . . 9
5.1 ZkEVMs Types. . . . 12
6.1 Distribution of zkRollups into zkEVM Types. . . . 14
7.1 Scroll's Architecture. . . . 17
7.2 Roller Workflow. . . . 18
7.3 Scroll Workflow. . . . 20
7.4 Workflow of Based Rollup. . . . 21
7.5 Multi-Proof System. . . . 22
7.6 Working of a zkVM. . . . 23
7.7 Taiko's Hekkla Tier Configuration. . . . 25
7.8 Taiko's Inception Layers. . . . 27
7.9 Linea Architecture. . . . 27
7.10 Linea Prover's Pipeline. . . . 29
7.11 Polygon Prover Architecture. . . . 30
7.12 Polygon Prover's State Architecture. . . . 31
7.13 Components of Polygon's Prover. . . . 32
7.14 Unified Liquidity with the Aggregation Layer. . . . 33
7.15 Combination of zkRollup and zkPorter. . . . 36
7.16 Compiler Toolchain. . . . 37
7.17 Hyperchain's Architecture and their Workings. . . . 37
8.1 Reward Comparison of Big Bounty Programs. . . . 40

List of Tables

7.1 ZkEVM Architecture Comparison. . . . 16
8.1 zkEVM MultiSig Roles and Permissions. . . . 41
8.2 Handling Sequencer and Proposer Failures Across zkEVMs. . . . 42
9.1 Comparison of zkRollups Ecosystem Metrics (Dec. 2024). . . . 45
9.2 Categorization of zk Rollups by Key Domains. . . . 47
11.1 Fee Comparison for Contract Creation and Minting across zkEVMs (Testnets). . . . 56
11.2 Gas Usage and Prices for Contract Creation and Minting across zkEVM Testnets. . . . 57
11.3 Fee Comparison for Contract Creation and Minting across zkEVMs (Mainnet Estimation). . . . 57
13.1 Advantages and Disadvantages of zkEVM's Platforms. . . . 64

Listings

11.1 A simple ERC-1155 Contract. 53
11.2 Version Downgrading to Solidity Version 0.8.19. 55

1 Introduction

Bitcoin, launched in 2009, was the first blockchain to enable decentralized peer-to-peer transactions through the use of blockchain technology. Its genesis block, mined on January 3, 2009, marked the beginning of a new era in digital finance [1]. While Bitcoin primarily served as a digital currency, its design was limited in functionality.

In 2014, Vitalik Buterin founded Ethereum by uploading the Ethereum Whitepaper, titled "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform" [2]. Vitalik two primary objectives were to relieve the Bitcoin blockchain and create a network of decentralized computers capable of managing applications and programs. This vision was realized with the launch of Ethereum in 2015, which introduced the Ethereum Virtual Machine (EVM). The EVM enabled the execution of decentralized applications (dApps) and introduced smart contracts, allowing for programmable and automated interactions on the blockchain.

As of December 2024, Ethereum, after nine years, has reached a market cap of around $400 billion, with its price fluctuating between $3,000 and $3,500 per token [3]. Prominent projects like Chainlink, Aave, and Uniswap have been built on Ethereum [4–6]. However, despite its success, Ethereum faces significant scalability challenges. As the demand for transactions increases, the network struggles with high gas fees, slow transaction processing times, and limited throughput, with just 15 transactions per second [7].

To address Ethereum's scalability challenges without compromising its core principles of decentralization and security, Layer 2 rollups have emerged as a key solution. According to Vitalik Buterin, rollups represent the most viable approach for scaling Ethereum in the short to medium term [8]. One notable type of rollup is the Zero-Knowledge (ZK) Rollup, which improves efficiency by bundling multiple transactions into batches that are executed off-chain. An example of a zkRollup is Loopring [9], which is a decentralized exchange and payment service. However, Loopring does not support custom smart contracts, limiting its flexibility compared to Ethereum's EVM.

This limitation has led to the development of Zero-Knowledge Ethereum Virtual Machines (zkEVMs), which aim to provide a fully compatible environment for running Ethereum smart contracts while maintaining the scalability benefits of zkRollups. Unlike Loopring, zkEVMs support custom smart contracts, enabling developers to build dApps in a way that mirrors the Ethereum ecosystem.

2 Research Question

In light of the growing importance of zkEVMs and their potential to scale Ethereum, this master thesis aims to address the following central question:

Which Layer-2 zkEVM solution is best suited for a project?

To provide a well-rounded answer, each protocol will be examined in depth, addressing the following sub-questions:

1. How is the architecture structured? What does the prover implementation look like, and what features does the platform offer?
2. How is the security of the protocol ensured? Does it have mechanisms to protect user assets?
3. How strong is the ecosystem? What decentralized applications (dApps) are available, and what incentives and support does the protocol provide for builders and the community?
4. How does the protocol enable user participation in governance and protocol upgrades?
5. What does the development process for smart contracts look like, and what are the associated costs?

Chapter 3 begins with an in-depth exploration of the EVM, focusing on its underlying architecture and operational mechanics. Chapter 4 follows with an examination of Ethereum scaling solutions, introducing Zero-Knowledge Proofs and zkRollups. Building on this foundation, Chapter 5 discusses zkEVMs, emphasizing their challenges and various implementation types. Chapter 6 introduces the specific zkEVMs under consideration, while Chapter 7 provides a detailed analysis of each, including their architectures, proof systems, and unique features.

Chapter 8 examines their security mechanisms, including audits, bug bounty programs, emergency response mechanisms, and sequencer/proposer failure handling. Chapter 9 explores their ecosystems, addressing incentives, technical support, and the types of dApps deployed on each platform. Chapter 10 discusses their governance structures, focusing on how users can participate in the governance process.

In Chapter 11, smart contracts are deployed on each platform, and the associated costs are estimated. Finally, Chapter 12 employs a scenario-based approach to identify the most suitable zkEVM for specific use cases.

3 Ethereum Virtual Machine

Over the last few decades, programming languages have been invented to make it easier for developers to understand and write applications. For example, Python is a high-level language that uses natural language elements and abstraction, enhancing the readability and maintainability of the code [10]. In addition, these languages often provide features like dynamic typing or automatic memory management [11].

While all of these high-level instructions are readable by humans, CPUs do not directly understand them. A CPU processes and executes instructions written in machine code. Therefore, a compiler translates high-level programming languages into machine code, enabling the CPU to understand and execute these instructions.

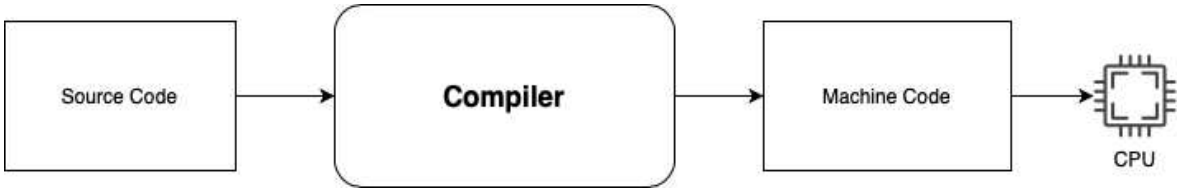


Figure 3.1: From Source Code to CPU.

A virtual machine (VM) is a software that behaves exactly like a machine with its own operating system, disk space, memory and CPU, that are borrowed from a physical host computer. It can run in a window as a separate computing environment and is partitioned from the rest of the system, meaning that the software inside a VM can't interfere with the host computer's primary operating system. One of the most famous Virtual Machines are for example VMWare or Virtualbox [12].

The EVM is a decentralized computation engine that executes smart contracts across the Ethereum network. Unlike traditional virtual machines (VMs), which emulate hardware to run entire operating systems and applications independently of the underlying physical infrastructure, the EVM's sole job is to update the Ethereum state by computing valid state transitions as a result of smart contract execution, as defined by the Ethereum protocol. Smart contracts are written in high-level languages like Solidity or Vyper and are compiled into bytecode. This bytecode is interpreted by the EVM, which then executes the corresponding instructions.

In summary, the EVM operates similarly to a normal computer. A developer uses a high-level programming language to write smart contracts, compiles it into machine code, and the EVM executes it using a CPU borrowed from the main host.

3.1 Architecture of EVM

The EVM is composed of three main components. The first is the **machine state**, a volatile part that includes elements such as the program counter, memory, and stack. The second component is the immutable **virtual ROM**, which stores the EVM bytecode that is executed.

Lastly, there is the **world state**, which holds all Ethereum accounts, along with their balances, storage, code, and other relevant data. In this section, each of these components will be explored in detail [13].

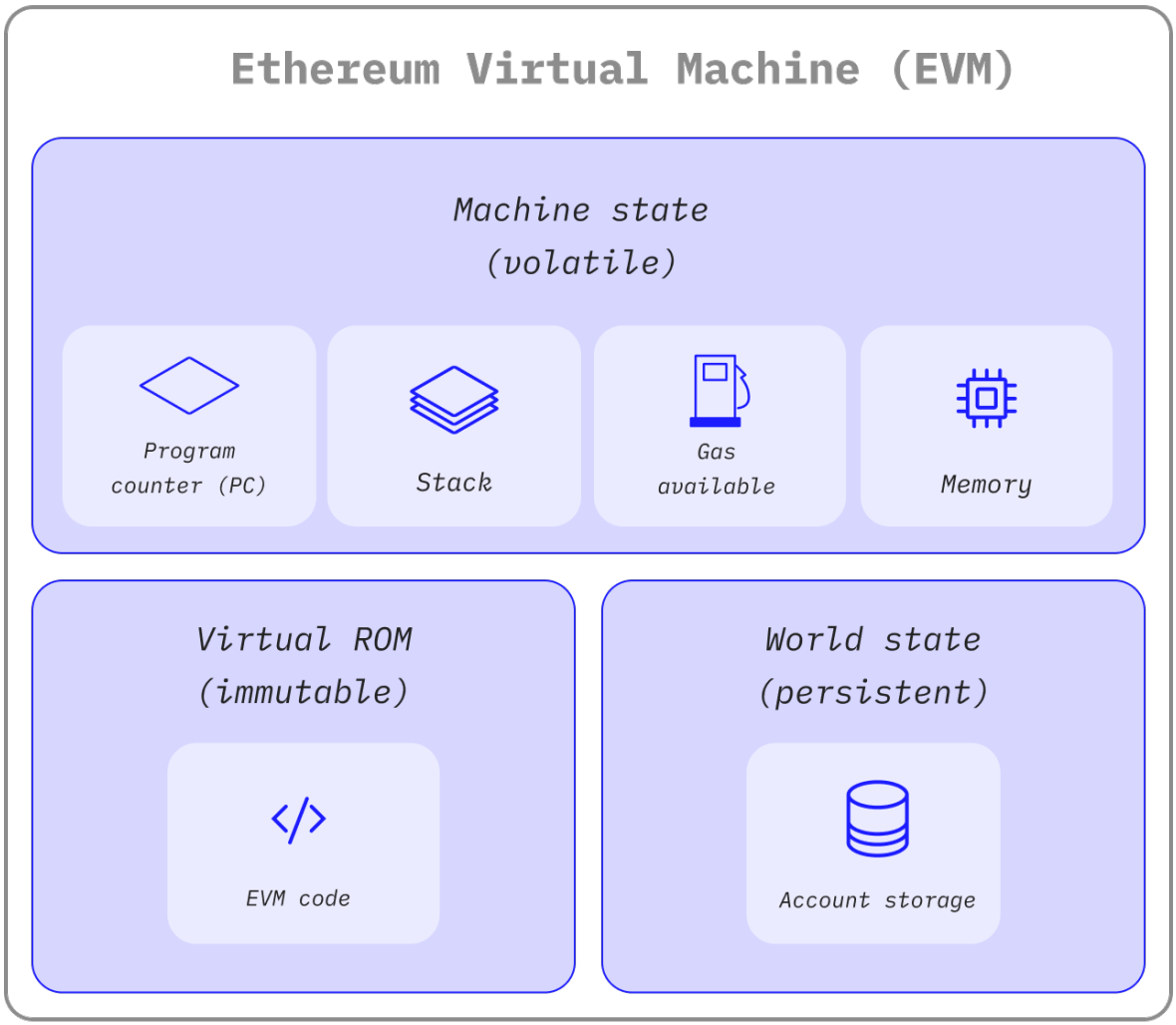


Figure 3.2: Ethereum Virtual Machine Architecture. [14]

3.1.1 World State

The World State in Ethereum is a mapping of 160-bit addresses to their corresponding accounts. Each account can either be an externally-owned account (EOA), controlled by a user with private keys, or a contract account, which represents a smart contract deployed on the network and controlled by its code. Each account consists of four key components:

1. An Ether balance, measured in wei.
2. A nonce, which indicates the number of successful transactions initiated from the account.
3. A storage root, used as a persistent data store exclusively by smart contracts.
4. A code hash, present only in smart contract accounts. EOAs have no associated code and maintain an empty storage.

The World State is organized using a modified Merkle-Patricia Trie, a cryptographic data structure that connects all accounts through hashes and condenses the entire state into a single root hash stored on the blockchain. This trie is both deterministic and cryptographically verifiable. The root hash is uniquely generated from all the individual state pieces, so if two states are identical, their root hashes and the hashes used to derive them will also match, which can be confirmed using a Merkle proof.

3.1.2 Virtual ROM

The virtual ROM is a read-only memory that contains the immutable "EVM bytecode," which only the EVM can interpret and execute. Once smart contracts are deployed, their code cannot be changed.

3.1.3 Program Counter

When a transaction results in smart contract code execution, the program counter keeps track of the position in the code that the EVM is currently executing.

3.1.4 Gas available (Gas)

Gas is Ethereum's unit for measuring the computational and storage resources required to perform actions on the Ethereum Blockchain. Since each Ethereum transaction requires computational resources to execute, those resources have to be paid for to ensure Ethereum is not vulnerable to spam and cannot get stuck in infinite computational loops [15]. Payment for computation is made in the form of a gas fee. For example [16]:

- Adding two numbers costs 3 gas
- Keccak-256 hash costs 30 gas + 6 gas for each 256 bits of data being hashed
- Sending a transaction costs 21,000 gas

Gas fees vary depending on the complexity of the operation and the amount of data involved. However, the total gas consumption for any transaction is constrained by the **gas limit**. The gas limit is the maximum amount of gas that a user is willing to spend on a transaction. It serves as a safeguard, ensuring that no transaction can run indefinitely or consume excessive computational resources.

When a user initiates a transaction, they set both a gas price and a gas limit:

- The **gas price** is the amount of Ether (ETH) the user is willing to pay per unit of gas. The higher the gas price, the more likely miners are to prioritize the transaction.
- The **gas limit** is the maximum amount of gas the user is willing to spend on the transaction. If the transaction requires more gas than the specified gas limit, it will fail with an "out of gas" error, and any changes made during the transaction are reverted.

It's essential for users to set a gas limit that is high enough to cover the computational cost of the transaction. If the gas limit is set too low, the transaction will not have enough resources to complete and will fail. Conversely, if the gas limit is set too high, the transaction may succeed, but any unused gas will be refunded to the user.

In summary, the **gas available field** tracks how much gas is left for the transaction to continue operating.

3.1.5 Stack

The EVM operates as a traditional stack-based machine, where it uses a last-in, first-out (LIFO) stack to manage computation. It processes operations by pushing and popping 256-bit integers, which are the standard size for all elements in the stack.

3.1.6 Memory

The Memory is a temporary place to store data during execution. It is volatile because it is reset between transactions or intra-transaction calls.

3.2 Transaction flow through the EVM

Ethereum supports two types of transaction. One is the *Contract creation*, which deploys a new smart contract and adds a new entry to the World State. The other one is a *message call*, initiated by an externally-owned account (EOA), which interacts with another EOA or a smart contract, updating the World State. [17]

For simplicity, the focus is placed on a typical Ethereum transaction, such as an Ether transfer between accounts. The transaction proceeds as follows:

- **Transaction Initialization:** A user (EOA) signs and broadcasts a transaction containing the recipient's address, amount, gas limit, and gas price, along with a signature for authentication.
- **World State Update:** The transaction, once mined, updates the World State. The sender's Ether balance decreases by the transfer amount and gas fee, while the recipient's balance increases accordingly.
- **Code Execution:** For contract interactions, the program counter tracks execution in the virtual ROM, with gas available ensuring enough resources. The EVM uses a stack and memory for processing.
- **Finalization:** After execution, the updated balances are finalized in the World State, and unused gas is refunded. The transaction is then added to the blockchain, updating the Merkle-Patricia Trie to reflect the new state.

4 Zero-Knowledge Proofs

The theoretical foundation of Zero-Knowledge Proofs (ZKP) were first described in a 1985 MIT paper by Goldwasser, Micali and Rackoff called "The Knowledge Complexity of Interactive Proof-Systems" [18]. The authors defined ZKPs as "proofs that convey no additional knowledge other than the correctness of the proposition in question." This concept can be likened to proving knowledge of a secret password without disclosing the password itself. From a technical perspective, ZKPs can be categorized into two types: interactive and non-interactive proofs[19].

In an interactive proof, there's a dynamic exchange between the prover and the verifier. Here, the verifier sends out specific random questions or challenges to the prover, who then responds accordingly. This dialogue continues through a series of rounds until the verifier is assured of the proof's accuracy.

In a non-interactive proof, the prover creates a proof that anyone can verify using the same proof without any further interaction. An example will be later provided.

For a ZKP to be secure, it must satisfy three properties [20]:

- **Completeness:** If the statement is true, an honest verifier will be convinced by an honest prover.
- **Soundness:** If the statement is false, no cheating prover can convince the honest verifier that it's true, except with some small probability.
- **Zero-Knowledge:** : If the statement is true, no verifier learns anything other than the fact that the statement is true.

4.1 ZkRollup

The main idea behind a rollup is that a network participant, known as a sequencer, aggregates multiple transactions off-chain (on layer 2). The sequencer then generates a compact proof, which confirms the legitimacy of these transactions on the main blockchain (layer 1). Although these proofs utilize ZKPs, their current focus isn't on privacy enhancement. Instead, they aim to validate the transactions efficiently and thus they are termed *validity proofs* [19]

The rollup's state is managed by a smart contract on the layer 1 network, which essentially serves as the on-chain verifier for these proofs. This means that the foundational layer (smart contracts) employs a validity proof, which, through ZKP, mathematically confirms that the state changes suggested by Layer 2 are accurate and result from executing the specified batch of transactions. A illustration can be seen in Figure 4.1.

In essence, the sequencer acts as the prover, creating proofs off-chain on Layer 2 and then transmitting a transaction to the on-chain smart contract. This smart contract is responsible for verifying the proof, ensuring that the Layer 2 transactions are valid and correctly executed without compromising the network's principles of security and decentralization.

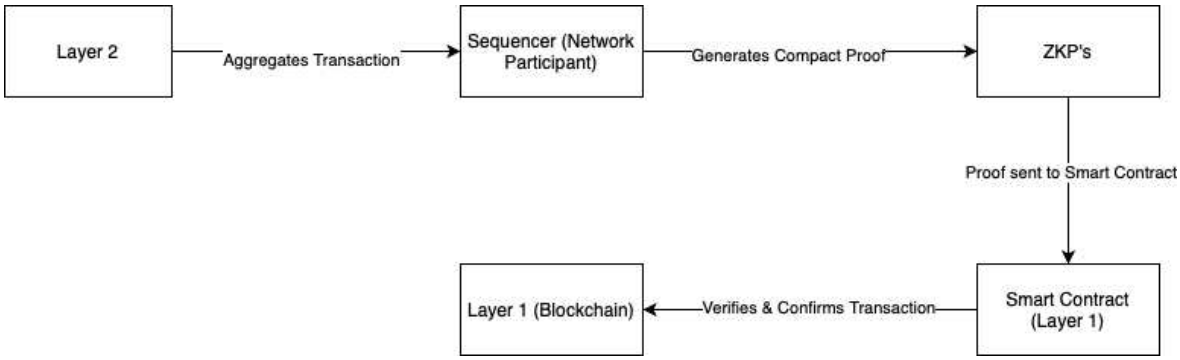


Figure 4.1: Working of a zkRollup.

4.2 Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs)

A widely-used form of ZKP within this context is the zk-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) that were introduced in a 2012 paper co-authored by Nir Bitansky, Ran Canetti, Alessan- dro Chiesa, and Eran Tromer [21]. Zcash was the first widespread application of ZK-SNARKs. The term “Succinct” means that the proofs can be verified within a few milliseconds, with a proof length of only a few hundred bytes.

A key aspect of some zk-SNARKs is the need for a trusted setup ceremony to generate keys for creating and verifying proofs. This phase involves a ceremony where special keys are generated, which are essential for the creation and validation of proofs within the system. The integrity of these proofs is foundational to the security and reliability of the zk-SNARKs system [22]. However, if the secret parameters used during the ceremony are not completely destroyed afterward, they pose a severe security risk. Malicious actors could potentially exploit these secrets to create false proofs. Such false proofs could compromise the entire system, allowing for fraudulent activities without detection. To mitigate the risks associated with the trusted setup, these ceremonies often involve a large number of participants, to minimize the chance of any single point of failure. By distributing the responsibility among many, the process aims to ensure that no single participant can compromise the system [22].

4.3 Zero-Knowledge Scalable Transparent Arguments of Knowledge (zk-STARKs)

Zk-STARKs is an alternative to zk-SNARKs that has been introduced in a 2018 paper by Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev [23]. A key difference to zk-SNARKs is that zk-STARKs can offer enhanced security though the non- requirement of a trusted-setup but the proofs can take longer to verify and can therefore be considered less efficient as a result. Zk-STARKs have larger proof sizes than zk-SNARKs, which means that verifying zk-STARKs may take more time and be more gas-intensive than zk-SNARKs. However, Unlike most zk-SNARKs, zk-STARKs rely on hash functions which are considered to be quantum-resistant [22].

4.4 Circuits

Zk-SNARKs are the preferred protocols in blockchains due to their non-interactive nature, succinct proof size, and sublinear verification time [24]. These protocols enable the verification of a wide range of calculations or claims without revealing the details of the computations. They can be modeled using arithmetic circuits over a finite field, breaking down the computations into a series of mathematical steps. To get an intuitive feeling how an arithmetic circuit might look like let's take a look on the Figure 4.2:

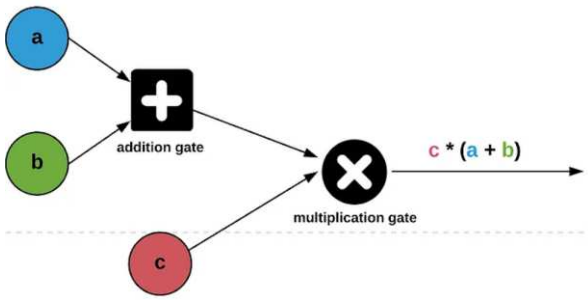


Figure 4.2: A simple example of an arithmetic circuit [25].

In this example *a,b,c* are *inputs gates* that has no incoming *input wires*. An input gate contains the data to be processed by the circuit, which can be either a variable or a number. A wire carries *integers*. Wires going into a gate contain the data to be processed by the gate. So for instance the two input gates containing a and b are fed into the addition gate. The addition gate adds *a* and *b* together and then sends *(a + b)* out via the output wire. The input gate containing the variable, *c*, is fed into the multiplication gate along with the term *(a + b)*. The multiplication gate computes *c * (a + b)* and sends the result out on its output wire, completing the evaluation of the circuit [25].

To verify a zk-SNARKs proof it is necessary to use an elliptic curve. In Ethereum, the curve is `alt_bn128` (also referred as BN254), which has prime order *r*. With this curve, it is possible to generate and validate proofs of any F_r arithmetic circuit. So in summary, zk-SNARKs permit proving any computational statement that can be modelled with an F_r-arithmetic circuit are called **zk-SNARKs circuits** [25].

4.5 Limitations

While zkRollups are secure and efficient, their current applications are mostly limited to payments and token swaps. For example, the Loopring Protocol, a popular zkRollup, makes it very difficult to create custom smart contracts or develop dApps. Developers have to use specific languages like Rank-1 Constraint Systems (R1CS), which are not only complex but also require a deep understanding of ZKPs.

Additionally, zkRollups do not support composability, meaning applications built on different zkRollups cannot easily interact within Layer 2. This limitation greatly reduces the flexibility and interconnectedness needed for Decentralized Finance (DeFi) applications.

However, over the last few years there have been many efforts to build a *universal EVM circuit* for executing smart contracts. With this approach many developers can develop a zkEVM. This concept will be discussed in the next section.

5 Zero-Knowledge Ethereum Virtual Machine

A zkEVM is a virtual machine that executes smart contract transactions in a way that's compatible with both ZKP computations and existing Ethereum infrastructure. This enables them to be part of zero-knowledge rollups, layer-2 scaling solutions that increase transaction throughput while lowering costs.

5.1 Challenges in building a zkEVM

Building such an zkEVMs comes with major challenges due to its high computational overhead and there are several reasons why [26]:

- **Limited support of elliptic curves:** Ethereum natively supports secp256k1 elliptic curve, which is used for public key cryptography in transactions and signatures. However, zk-SNARKs and other zero-knowledge techniques typically rely on different curves like BN254 or BLS12-381, which are optimized for the efficient generation of zk-proofs.
- **The EVM word size is 256 bit**: The EVM uses 256-bit integers, while zk proofs work best with prime fields. Mixing these requires range proofs, adding about 100 steps per EVM operation and making the circuit roughly 100 times larger.
- **EVM has many special Oppcodes**: EVM is different from traditional VM, it has many special opcodes like CALL and it also has error types related to the execution context and gas.
- **Ethereum storage layout carries a huge overhead**: Ethereum's storage layout relies on Keccak and a large Merkle Patricia Trie (MPT), both of which are inefficient for zk-proofs and cause high proving costs. For instance, Keccak is 1000 times larger than the zk-friendly Poseidon hash in a proof circuit. However, replacing Keccak would create compatibility issues with Ethereum's existing infrastructure.

Even proving a simple addition operation requires handling the full complexity of an entire EVM circuit. If there are many operations in the execution process, the prover must deal with a proportional increase in complexity. The EVM circuit has to include all possible logic, making it much larger than just the addition operation. As a result, even verifying basic operations involves significant overhead due to the need to process the entire EVM circuit.

5.2 Advancements in zkEVM Development: From Polynomial Commitments to Hardware Acceleration

Despite the challenges, developing zkEVMs has become increasingly feasible in recent years. The introduction of polynomial commitment schemes has been a major advancement, enabling efficient and compact proofs and reducing the overhead of zk-proof generation and verification [27].

Another key development is in recursive proofs, which involve proving one proof within another [28]. Previously, this was difficult and costly due to the need for special elliptic curves. However, new techniques like Halo have made recursion more efficient without requiring these curves. Aztec has also improved efficiency by using lookup tables for proof aggregation, which makes verification circuits smaller and more scalable [29].

Additionally, hardware acceleration has enhanced proving efficiency. For example, Scroll has proposed using GPU and ASIC/FPGA accelerators that are 5 to 10 times faster than Filecoin's implementation [30].

5.3 zkEVMs Types

With these advancements, the development of zkEVMs has progressed significantly, leading to various types of zkEVMs implementations [31]. The following illustration emphasizes the different Types of zkEVMs.

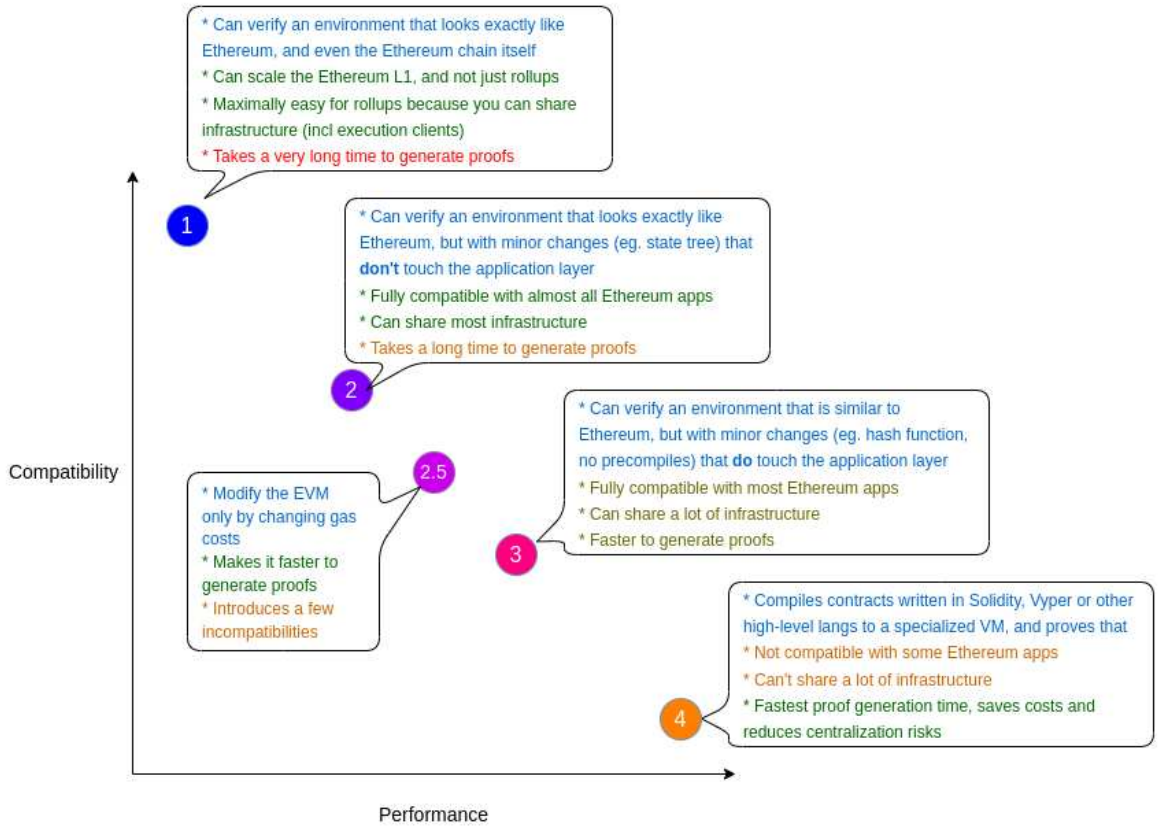


Figure 5.1: ZkEVMs Types [31].

5.3.1 Type 1 (Fully Ethereum-equivalent)

Type 1 zkEVM strive to be fully and uncompromisingly Ethereum-equivalent. They do not change any part of the Ethereum system to make it easier to generate proofs. They do not replace hashes, state trees, transaction trees, precompiles or any other in-consensus logic. This makes them fully compatible with all Ethereum-native applications and enables

tools like block explorers and execution clients to be reused. Howevever, one of the biggest disadvantages is that Ethereum was not designed to be zk-friendly, so there are many parts of the Ethereum protocol that take a large amount of computation to ZK-prove. Which means to proof a block it can take many hours.

5.3.2 Type 2 (fully EVM-equivalent)

Type 2 zkEVM aim to be fully EVM-equivalent but not entirely Ethereum-equivalent. This means they function just like Ethereum from the perspective of applications but have some external differences, such as in data structures like the block structure and state tree.

Most Ethereum applications and tools can still be used, though with minor adjustments. For example, while the Ethereum execution client may require some modifications to work on certain zkEVMs, it remains compatible with most of the existing EVM debugging tools and developer infrastructure.

While these modifications significantly improve prover times, they do not solve every problem. The slowness from having to prove the EVM as-is, with all of the inefficiencies and ZK-unfriendliness inherent to the EVM, still remains.

5.3.3 Type 2.5 (EVM-equivalent, except for gas costs)

Increasing the gas costs for specific operations can improve the proving times for some of the most difficult proof generation scenarios. However, this can break some applications and requires developer modifications, but it’s generally considered less risky than "deeper" EVM changes.

5.3.4 Type 3 (almost EVM-equivalent)

Type-3 zkEVMs sacrifice some EVM features to enable easier application development and proof generation, such as changes to precompiles, VM memory, the stack, and how smart contract code is treated. While most Ethereum applications will work in this environment, some may need to be rewritten.

5.3.5 Type 4 (High-Level-Language Equivalent)

Type 4 systems compile code from high-level languages like Solidity or Vyper into a format optimized for ZKPs, making them more efficient than directly using the EVM. By avoiding ZK proofs for every step of EVM execution, they significantly reduce overhead.

However, while contracts written in these high-level languages can be compiled, additional effort is required for dApps that rely on handwritten EVM bytecode. Type 4 zkEVMs also need specialized developer tools, such as debuggers and tracers, that work at the opcode level. On the plus side, they support custom opcodes not available on Ethereum, allowing developers to implement new features beyond what Ethereum offers by default.

6 Different zkEVMs

This chapter explores various zkEVMs that are either currently available on the market or undergoing rapid development. These zkEVMs operate as Layer 2 solutions, inheriting Ethereum's robust security properties while bundling thousands of transactions into a single batch for submission to the main chain. The primary objective of developing a zkEVMs is to achieve full compatibility with the EVM, including all opcodes and architectural elements.

The following illustration will categorize different zkEVMs based on their respective EVM types:

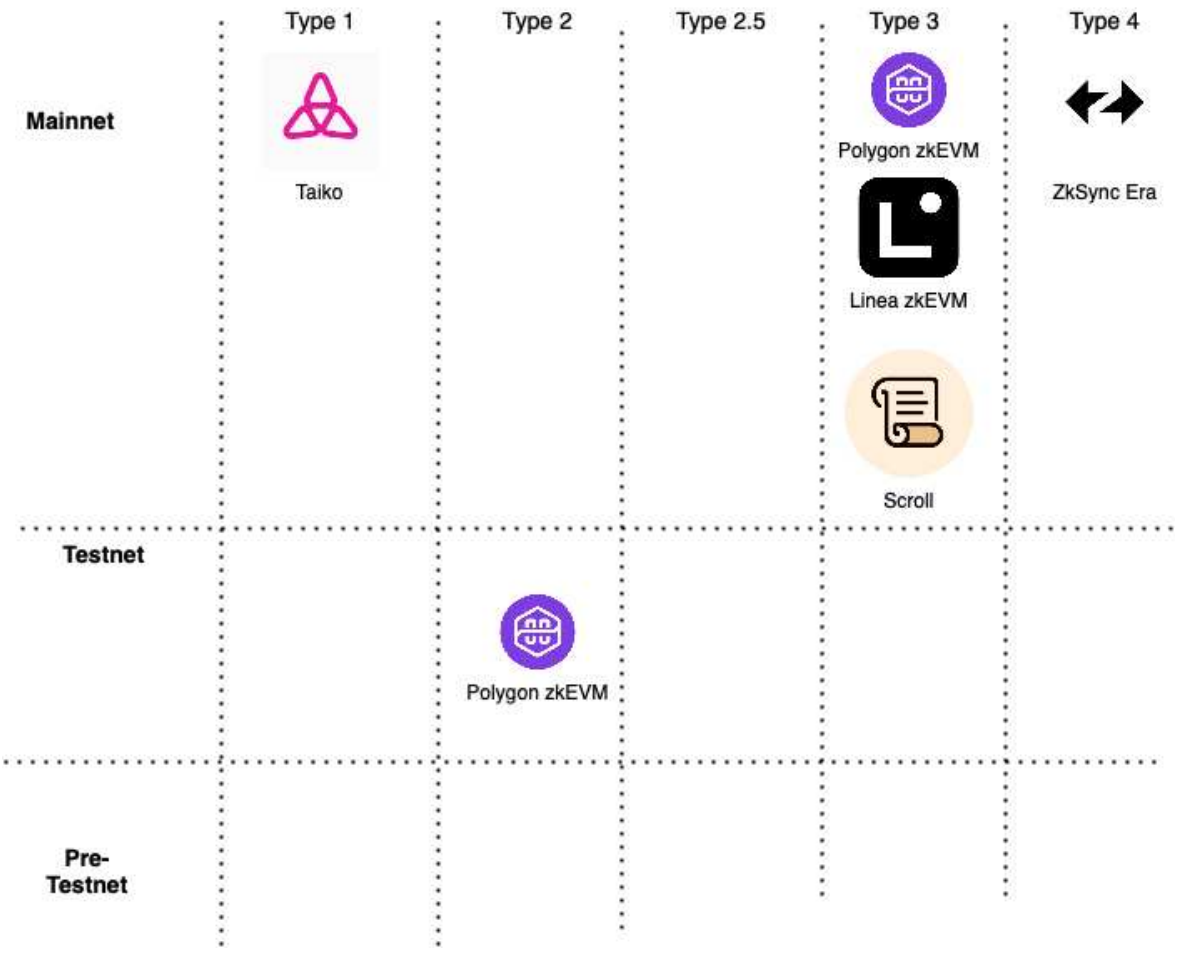


Figure 6.1: Distribution of zkRollups into zkEVM Types.

As illustrated, there are five distinct zkEVMs to be examined.

Taiko was founded in March 2022 by Daniel Wang, who previously created Loopring in 2017. After two years of development and seven major testnets, Taiko officially launched its mainnet on May 27, 2024 [32]. It remains the only zkEVM currently classified as Type 1, prioritizing full Ethereum equivalence.

Scroll, established in 2021, progressed rapidly through its development phases. The pre-alpha testnet opened in August 2022, followed by the alpha testnet on Goerli in February 2023, and ultimately, its mainnet launch in October 2023 [33]. Scroll is currently classified as a Type 3 zkEVM.

Linea, developed by ConsenSys, entered the zkEVM space backed by a company known for its extensive contributions to the Ethereum ecosystem. Founded by Ethereum co-founder Joseph Lubin, ConsenSys built products like MetaMask, Infura, and Truffle. Linea's mainnet launched in August 2023 [34], with the project categorized as a Type-3 zkEVM.

Polygon zkEVM, developed by Polygon Labs, represents another significant milestone in scaling Ethereum. While Polygon is often associated with its Proof-of-Stake (PoS) sidechain, Polygon zkEVM should not be confused with this earlier network. The Polygon PoS sidechain operates as a separate blockchain with its own consensus mechanism, running parallel to Ethereum. In contrast, Polygon zkEVM is a Layer 2 solution that leverages zero-knowledge rollups to enhance Ethereum's scalability. The Polygon zkEVM testnet launched in October 2022 with a fully functional ZK proving system. Following audits, the mainnet beta launched in March 2023 [35]. Initially classified as Type-3, Polygon transitioned to Type-2 after its *Etrog* update.

Finally, **ZkSync**, created in 2019 by Matter Labs, focuses on addressing Ethereum's scalability challenges. Its first version, ZkSync 1.0, launched in June 2020 as a zkRollup. In October 2022, Matter Labs released ZkSync 2.0, adding smart contract functionality, later rebranded as ZkSync Era [36]. On March 24, 2023, Matter Labs officially launched its zkEVM, making ZkSync Era the only Type-4 zkEVM currently in operation.

7 Architecture

This section examines the architecture of the zkEVMs. Although their architectures differ in several significant ways, they all include two core components: a sequencer and a prover. The sequencer handles tasks such as receiving L2 transactions from users, ordering them, generating transaction blocks, batching them, and submitting these batches to the bridge contract. The prover is responsible for generating the ZK-SNARK proofs.

The Table 7.1 below outlines the key similarities and differences in their designs.

	Scroll	Taiko	Linea	Polygon zkEVM	ZkSync Era
Sequencing	Layer 2	Layer 1 (Based Sequencing)	Layer 2	Layer 2	Layer 2
Proof	SNARK	SNARK	SNARK	SNARK +STARK	SNARK +STARK
Native Account Abstraction	No	No	No	No	Yes
Scalability Solutions	No	Inception Layers	No	Aggregation Layer	Hyperchains, ZkPorter
Multi-Prover Design	No	Yes	No	No	No
Compiler	Standard Solidity Compiler	Standard Solidity Compiler	Standard Solidity Compiler	Standard Solidity Compiler	Custom Compiler Toolchain

Table 7.1: ZkEVM Architecture Comparison.

Sequencing in Layer 2 is generally more centralized, which offers significant practical benefits during the early stages of adoption. This approach simplifies the architecture, making it easier to implement, test, and deploy, though it does come with trade-offs in decentralization [37]. While most platforms use Layer 2 for sequencing, Taiko stands out by adopting a more advanced Layer 1-based sequencing model.

All solutions employ SNARKs, but Polygon zkEVM and ZkSync Era go a step further by incorporating both SNARKs and STARKs. ZkSync Era also distinguishes itself by supporting Native Account Abstraction, a feature not integrated by the other platforms.

In terms of scalability, Taiko uses "Inception Layers," Polygon zkEVM features an "Aggregation Layer," and ZkSync Era introduces more sophisticated solutions like "Hyperchains" and "ZkPorter." Taiko also offers a unique multi-prover design, which enhances decentralization and robustness, a feature not supported by the other platforms.

Finally, while most platforms rely on the standard Solidity compiler, ZkSync Era sets itself apart by using a custom compiler toolchain.

The following sections provide an in-depth exploration of various zkEVM architectures. The discussion begins with Scroll, outlining its components, structure, and the interactions that culminate in generating the final proof. This is followed by an analysis of the Taiko architecture, introducing key concepts such as Based Rollups, Multi-Prover Systems, Contestable Rollups, Booster Rollups, zkVMs, and the Inception Layer.

Subsequently, the focus shifts to the Linea architecture, emphasizing its distinctive prover implementation. The Polygon zkEVM is then explored, with attention to its state machine prover and the Aggregation Layer (AggLayer). Lastly, ZkSync is reviewed, highlighting its unique features, including Account Abstraction, zkPorter, the Compiler Toolchain, and zkHyperchains.

7.1 Scroll

The architecture of Scroll consist of mainly three infrastructure components. First is the Scroll Node that is the central interface for applications interacting with Scroll. Second are Rollers that act as provers within the network, responsible for generating validty proofs for the zkRollup. Third is the Rollup and Bridge contracts that connects to the base layer of Ethereum. It ensure data availability for L2 transaction and allow users to pass assets and messages between L1 and L2 [38].

Let's take a look at the Figure 7.1.

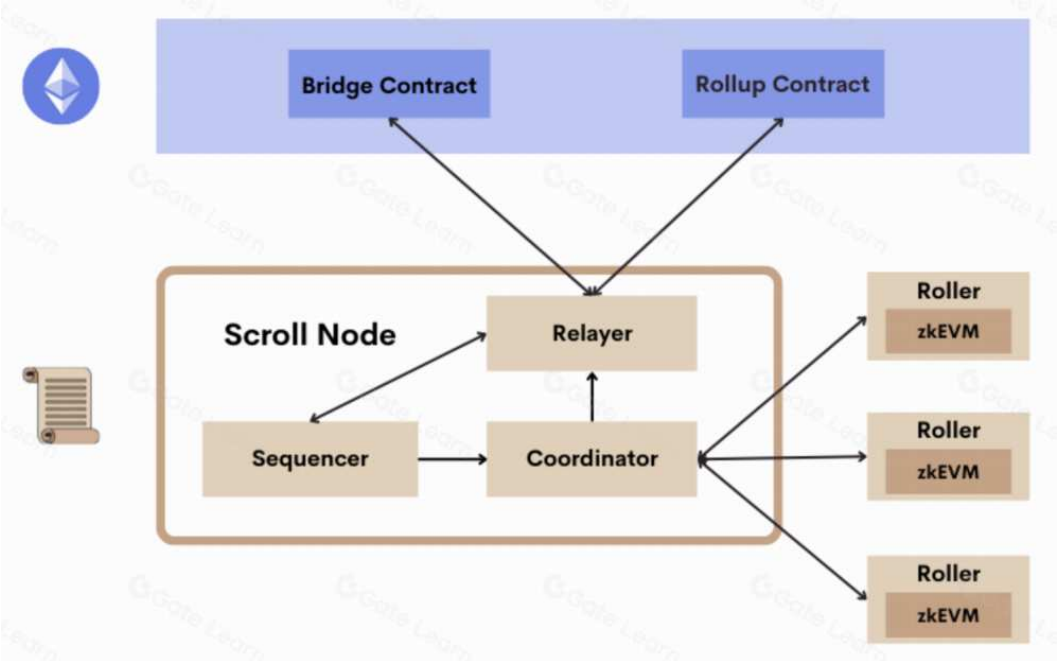


Figure 7.1: Scroll's Architecture [38].

The scroll node is composed of three key modules: the *Sequencer*, the *Coordinator*, and the *Relayer*.

The Sequencer is responsible for handling Layer 2 (L2) transactions through a JSON-RPC interface. It collects transactions from the L2 mempool, processes them, and generates new L2 blocks along with updated state roots. After a block is created, the **Coordinator** takes over, receiving the *execution trace*. This trace is essentially a detailed log of every computation and operation that occurred during the block's execution, such as opcode executions, state changes, and intermediate results. It is then sent to a randomly selected Roller within the Roller Pool for proof generation.

Simultaneously, the Relayer ensures seamless communication between Ethereum and Scroll by monitoring bridge and rollup contracts on both networks. It tracks the status of L2 blocks, including their data availability and validity proofs, while also managing deposit and withdrawal events on bridge contracts. These messages are relayed between Ethereum and Scroll to maintain synchronization.

7.1.1 Roller Network

As mentioned, the execution trace is sent to a randomly selected Roller within the Roller Pool for proof generation. To better understand the Roller's role, let us examine the Figure 7.2.

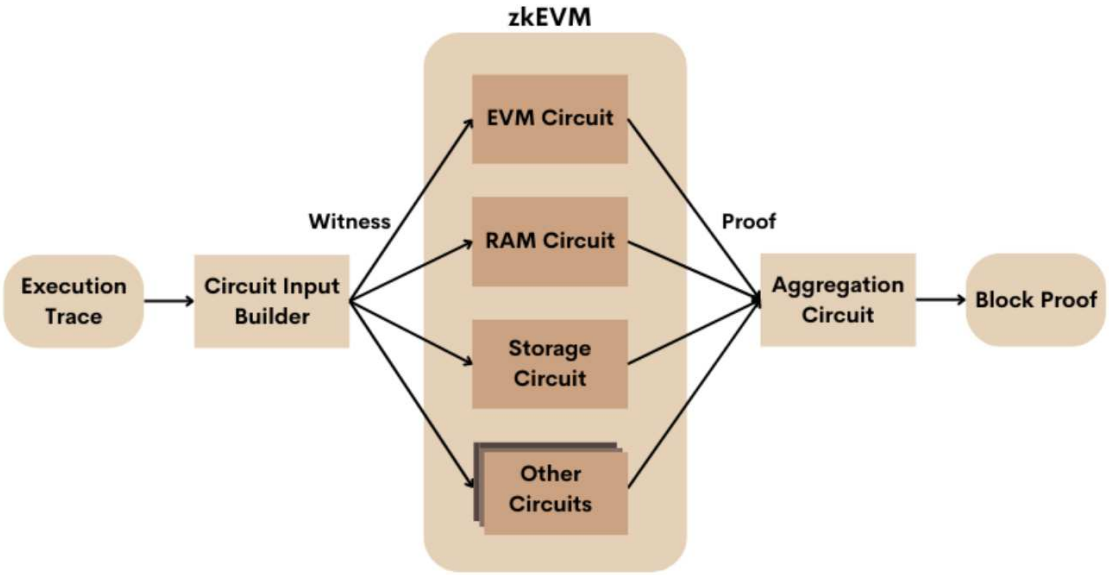


Figure 7.2: Roller Workflow [38].

As a first step, the Roller transforms the execution trace, provided by the Coordinator, into *circuit witnesses* using the **Circuit Input Builder**. These witnesses serve as inputs to the zkEVM circuits, which encode the logic of execution trace. Simply put, circuit witnesses translate the computational process into a format that the zkEVM circuits can process [38].

zkEVM circuits, in turn, are mathematical representations of Ethereum's execution rules. They define how transactions are validated, storage is updated, and state transitions occur. The circuit witnesses act as inputs for these circuits, enabling the Roller to verify that the

execution trace adheres to Ethereum's expected computational rules [39, 40]. Each zkEVM circuit focuses on a specific part of the computation, such as arithmetic operations, memory management, or storage updates.

Using the circuit witnesses and zkEVM circuits, the Roller generates individual proofs for each circuit. These proofs confirm that the computations in the execution trace were performed correctly according to the zkEVM's rules.

Since the execution trace involves multiple zkEVM circuits, the Roller generates multiple individual proofs. To optimize efficiency and minimize the verification burden, these proofs are aggregated into a single proof that represents the entire block.

To further reduce proving time and costs, hardware accelerators such as GPUs, FPGAs, and ASICs are utilized. Scroll specifically employs a pipelined accelerator with two subsystems designed to handle large-scale polynomial computations and multi-scalar multiplications on elliptic curves, which are critical for proof generation [30]. This approach makes proof generation 5 to 10 times faster than Filecoin's implementation.

7.1.2 Rollup and Bridge Contract

Scroll's proof generation process seamlessly integrates with its *Rollup* and *Bridge* contracts. Once the Roller generates and aggregates the final proof confirming the correctness of an L2 block, the Rollup contract receives the state root updates and L2 block data from the Sequencer and stores the state roots on Ethereum's mainnet. This ensures data availability and allows the Scroll Relayer or any participant to reconstruct L2 blocks using Ethereum's security guarantees. Upon receiving a valid proof, the Rollup contract marks the corresponding L2 block as finalized, solidifying the proof's role in maintaining consistency between L1 and L2.

In parallel with the proof verification process, the ***Bridge contracts*** enable the secure transfer of assets and messages between Ethereum (L1) and Scroll (L2). Built atop a robust message-passing protocol, these contracts leverage the proven execution trace verified by the Rollup contract. Users can bridge ERC-20 assets trustlessly, with the assurance that the underlying computations have been validated through zkEVM proofs. This integration ensures that all cross-layer asset transfers adhere to Ethereum's security model, reinforcing the trustless nature of Scroll's architecture.

7.1.3 Workflow

The following Figure 7.3 will illustrate how L2 blocks in Scroll are generated, committed to base layer Ethereum, and finalized [38].

- First, **The Sequencer** creates a series of blocks. For the i-th block, it produces an execution trace T and forwards it to the Coordinator. At the same time, it submits the transaction data D as calldata to the Rollup contract on Ethereum to ensure data availability, while sending the resulting state roots and commitments tied to the transaction data to the Rollup contract as part of the state.

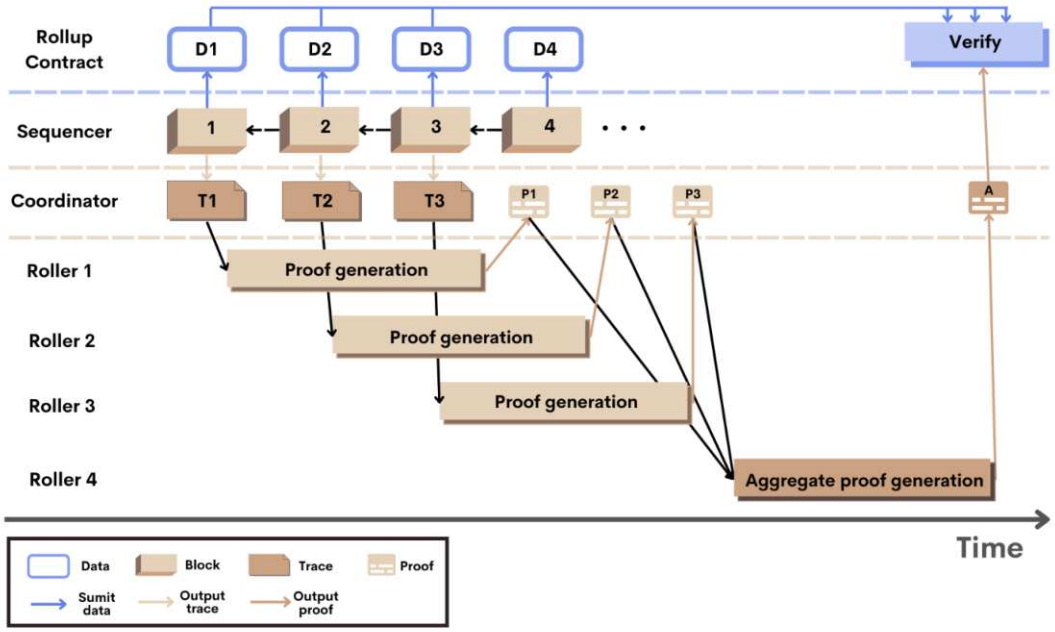


Figure 7.3: Scroll Workflow.

- Second, The **Coordinator** randomly selects a Roller to generate a validity proof for each block trace. To speed up the proof generation process, proofs for different blocks can be generated in parallel on different Rollers.
- After creating the proof P for the i-th block, the Roller sends it to the Coordinator. After every k blocks, the Coordinator assigns another Roller to combine the k block proofs into one aggregated proof, called A.
- Finally, the Coordinator submits the aggregate proof A to the Rollup contract to finalize L2 blocks i+1 to i+k by verifying the aggregate proof against the state roots and transaction data commitments previously submitted to the rollup contract.

7.2 Taiko

This section discusses Taiko's concepts of Based-Rollup, Multi-Prover System, Contestable Rollups, Booster Rollups, zkVMs, and the Inception Layer.

7.2.1 Based Rollups

Taiko makes use of a concept called based rollups. Ethereum researcher Justin Drake proposed *based rollups* in March 2023 as an approach to overcoming issues facing existing rollup solutions. He describes a based rollup as follows: "A rollup is said to be based, or L1-sequenced, when its sequencing is driven by the base L1. More concretely, a based rollup is one where the next L1 proposer may, in collaboration with L1 searchers and builders, permissionlessly include the next rollup block as part of the next L1 block [41]."

In based rollups, the order of transactions within an L2 block is typically determined by the L2 builder or sequencer, while the L1 proposer include the pre-validated rollup block in the L1 blockchain. Therefore it relies on Ethereum validators to sequence transactions and blocks instead of a centralized sequencer. An illustration is provided to make this concept clearer.

With based sequencing, Taiko benefits from the same liveness guarantees as Layer 1 (L1). They ensure reliable transaction processing through their integration with the L1 blockchain, without the risk of reduced performance that is often seen in other rollup designs.

Additionally, based rollups promote decentralization by leveraging the existing L1 infrastructure. This approach encourages collaboration among L1 searchers and block builders, facilitating the inclusion of rollup blocks within L1 blocks and strengthening the overall network.

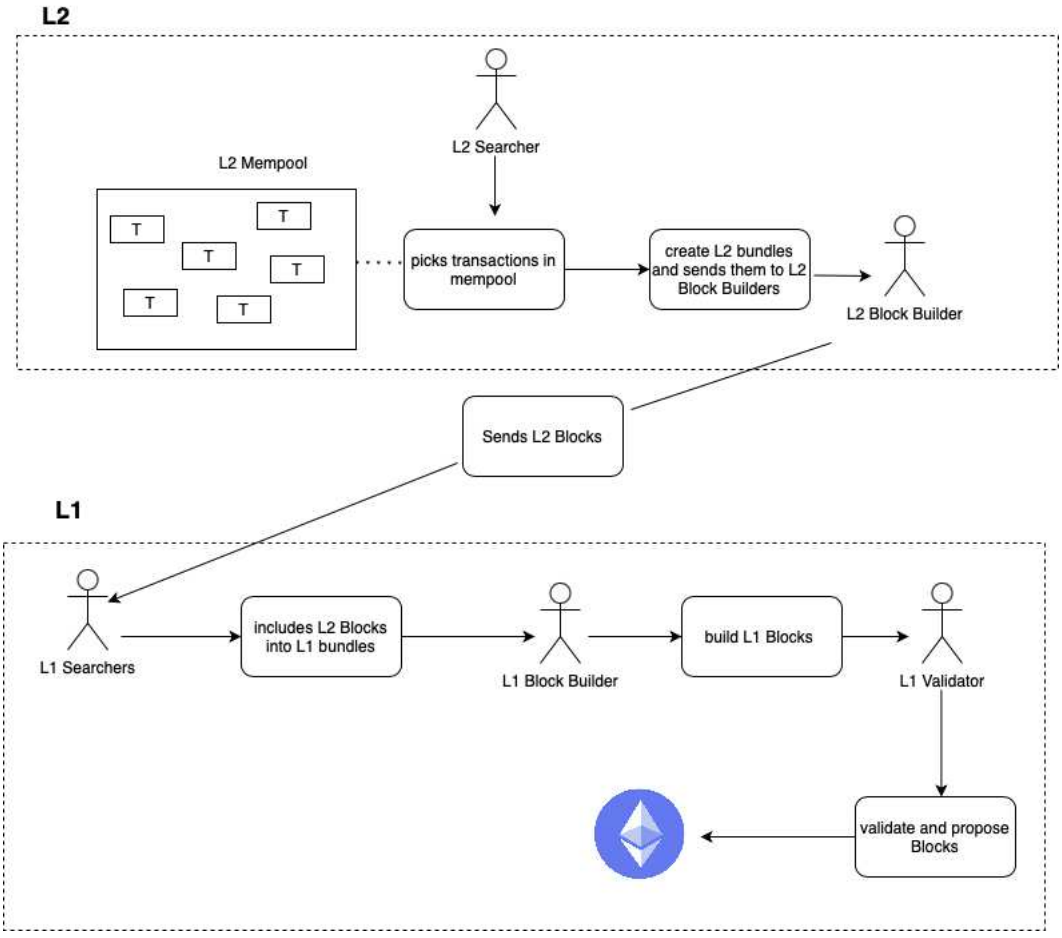


Figure 7.4: Workflow of Based Rollup.

7.2.2 Multi-Proof System

In general, ZK rollups leverage ZKPs to validate the correct execution of L2 transactions. However, the complexity of ZK circuits—particularly zkEVM circuits—can be substantial, often consisting of tens of thousands of lines of code. This inherent complexity increases the likelihood of bugs occurring [42].

By generating multiple proof types for a single block, it mitigates the risk associated with potential vulnerabilities. In the event that one proof is compromised, it is unlikely that the others will share the same vulnerability, thereby providing an additional layer of security. It can be compared to Ethereum client diversity for example Geth, Besu or Nethermind.

Taiko suggests that SGX could serve as an additional type of validity proof alongside traditional ZKPs. This means that in addition to using ZK proofs to verify transactions, the protocol may incorporate SGX (Software Guard Extension) as a complementary method to enhance validation processes [43].

SGX is a type of Trusted Execution Environment (TEE) made by Intel. TEE is an area on the main processor of a device that is separated from the system's main operating system (OS). It ensures data is stored, processed and protected in a secure environment. SGX enables applications to execute code and protect secrets inside their own trusted execution environment, providing protection from malicious software [43]. Taiko has recently just released Raiko, an implementation of Taiko's multi-prover for Taiko & Ethereum blocks. It currently supports SGX, Risc0 and SP1 [44].

In summary, there are two types of proofs: the ZK proof and the SGX proof. Both are sent to the Rollup Smart Contract on Layer 2, which performs several tasks. It verifies all the proofs, ensures that they correspond to the same blockhash, and checks whether the expected number of proofs have been provided. An simple illustration can be found below:

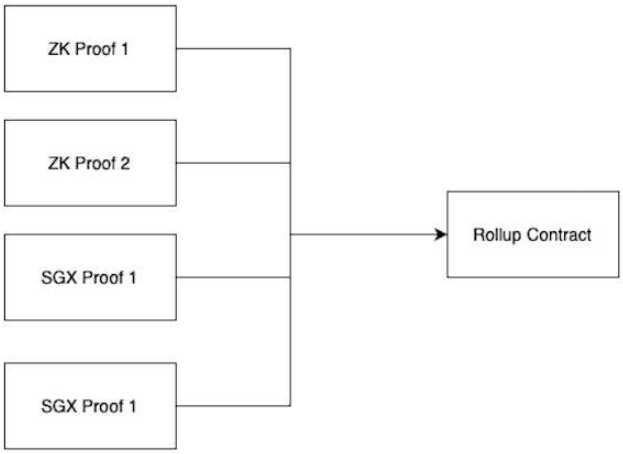


Figure 7.5: Multi-Proof System.

SGX provides strong performance benefits, running computations with almost no delay. However, because it depends on trusting Intel, it is not sufficient as a standalone security solution. Instead, SGX should be used within a multi-prover system as an additional layer of security.

Vitalik Buterin recently highlighted this point [45], stating that SGX is useful when it adds security but problematic when it introduces reliance. For example, a rollup that only relies on SGX would be highly insecure due to the centralization risk.

7.2.3 ZkVM

As disscussed in chapter 5, a zkEVM is a virtual machine that executes Ethereum smart contracts in a way that is compatible with ZKP computation. The EVM was not originally designed to operate within a zk-circuit, as zero-knowledge technology was relatively unknown at the time. As a result, implementing a zkEVM requires various trade-offs, such as opting for partial compatibility with Ethereum or having gas costs that do not accurately represent prover costs.

A zkVM, or zero-knowledge virtual machine, is a specialized virtual machine that operates as a circuit within a ZKP system. It ensures secure and verifiable trustworthiness by using ZKPs to validate its computations. Rather than demonstrating that a specific program has been executed correctly, the zkVM provides proof of the execution of the entire virtual machine itself. This approach allows it to support multiple programs while maintaining privacy and security, as it does not disclose any sensitive information about the individual programs being executed [46].

Let's assume the following diagram to explain a zkVM further in the context of Ethereum:

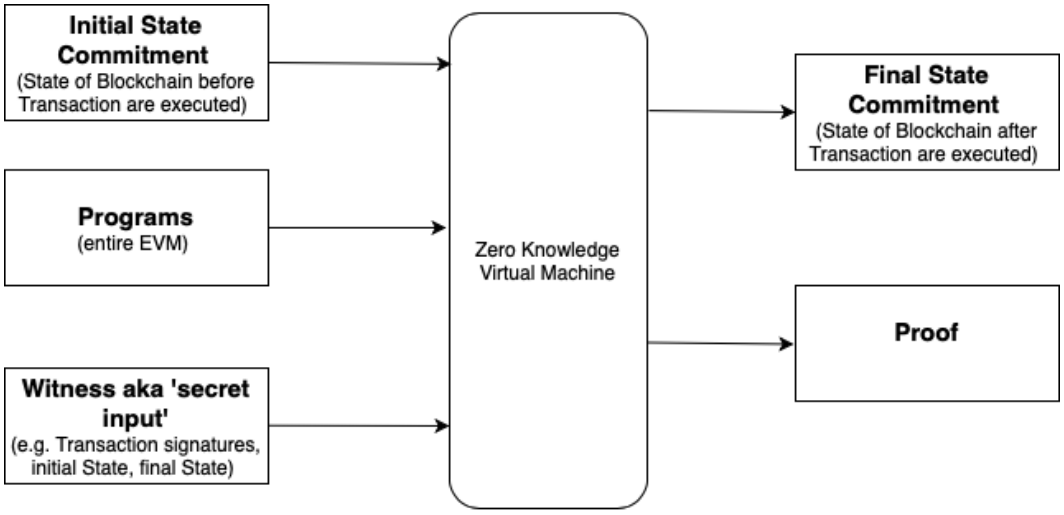


Figure 7.6: Working of a zkVM.

The zkVM has three primary inputs [47]:

- **Initial State**: This represents the state of the Ethereum blockchain before the transactions are executed.
- **Program**: In this context, the program can refer to any arbitrary program; however, in this case, the term "program" is used to denote the entire EVM.
- **Witness**: This consists of secret values that are only known to the zkVM and not to the verifier. The witness may include transaction signatures, the initial state, and the final state.

All of these inputs are processed by the zkVM, which ultimately generates the **Final State** and the corresponding proof.

Taiko is transitioning to a zkVM model. Initially, it used a specialized, circuit-based approach similar to ASICs, where the EVM opcodes were implemented manually as cryptographic circuits.Now, they have moved towards a more general-purpose model that is analogous to using a CPU, where a client, already written and tested, generates zk-proofs as it runs [46]. As mentionend, Taiko's multi prover implementation Raiko supports **Risc0** and **SP1**, which are both ZkVM's. This means each prover can configure their systems to leverage different zkVM architectures, allowing for flexibility in proof generation and verification.

7.2.4 Contestable Rollup

Since Zk-Proofs System are prone to errors, Taiko has introduced the concept of **Contestable Rollups** that features contestation and employs *based sequencing* along with a *multi-prover* design (as discussed in section 7.2.1). To better understand how contestation operates within Taiko, consider the following example [48]:

Alice proposes a new block. Bob then submits a proof for the state transition from $H_1 \rightarrow H_2$, where H_1 is the parent hash and H_2 represents the new block's hash. Bob posts a 10,000 TKO validity bond, after which his proof enters a cooldown period. Bob's proposed state transition and proof are publicly visible.

Cindy notices an error in Bob's transition and argues that the correct transition should be $H_1 \rightarrow H_3$ instead of $H_1 \rightarrow H_2$. She challenges Bob's proof during the cooldown period by posting her 10,000 TKO contestation bond. However, Cindy does not provide an alternative proof or explicitly declare the correct transition. The contested transition now awaits a new, higher-tier proof, which can be submitted by Bob or any other prover.

General Contestation Rules in Taiko

In Taiko, each proof, except for the highest-tier proof, requires the original prover to pay a validity bond in Taiko tokens. The proof then enters a cooldown window, during which it can be contested. Contesters are not required to provide a fraud or validity proof but must post a contestation bond in Taiko tokens. If a contestation occurs, a higher-tier proof is required to resolve the dispute before the block can be verified.

Contestation Outcomes

- **If the contester wins:** The contester retrieves their contestation bond and receives one-quarter of the original prover's validity bond. The new prover earns one-quarter of the original prover's validity bond as a proving fee, while the remaining half is forfeited.
- **If the original prover wins:** The original prover reclaims the validity bond and receives one-quarter of the contestation bond as a reward. The new prover (who may be the original prover) earns one-quarter of the contestation bond, while the remaining half is forfeited.
- The new prover is also required to pay a validity bond in accordance with the new tier's rules, unless they are providing the highest-tier proof, in which case the state transition is considered final, and no further contestation is allowed.

Example of Contestation Process

There are two possible scenarios when contestation occurs in Taiko:

Scenario 1: Bob's Transition is Correct

David submits a tier-3 proof for the state transition $H_1 \rightarrow H_2$, validating Bob's original claim. As a result:

- David earns a 2,500 TKO reward and becomes the current prover by posting a 20,000 TKO validity bond.
- Cindy forfeits her 10,000 TKO contestation bond.
- Bob is refunded his 10,000 TKO validity bond and receives a 2,500 TKO reward.
- A new cooldown period begins for David's proof.

Scenario 2: Bob's Transition is Incorrect

David submits a tier-3 proof for a transition from $H_1 \rightarrow H_4$, proving that Bob's transition was incorrect. In this case:

- David receives a 2,500 TKO reward and becomes the current prover by posting a 20,000 TKO validity bond.
- Cindy retrieves her 10,000 TKO contestation bond and earns an additional 2,500 TKO reward.
- Bob's 10,000 TKO validity bond is confiscated.
- A new cooldown period begins for David's proof.

To maintain a pool of available high-tier provers, Taiko introduces a mechanism that randomly assigns a minimum required tier for each new block. The following tier configuration is from the Hekkla Testnet [49]:

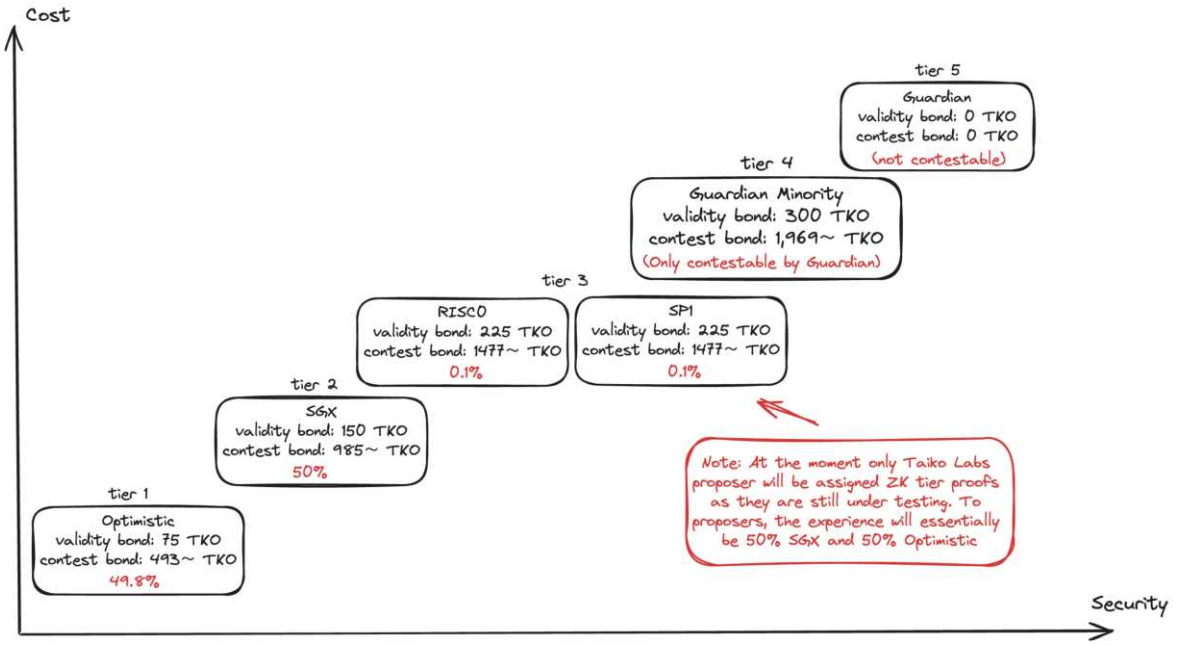


Figure 7.7: Taiko's Hekkla Tier Configuration.

For example Optimistic tier are used 49.8% of the time and SGX 50% of the time to reduce cost while only 0.2% are needed for a ZK-proof. Guardian Provers on the other hand are multisig signers who collectively serve as the higher tier in the proof hierarchy during the first couple of years after launch.

7.2.5 Booster Rollups

Rollups are often viewed as separate chains. While some designs aim to easily bring L1 data to rollups, they still don't create a unified environment for scaling Ethereum efficiently. If scaling Ethereum requires hundreds or thousands of rollups, treating each as an isolated unit with its own smart contracts and rules isn't practical—developers would need to copy-paste their code on each rollup [50].

Brecht Devos, the co-founder of Taiko Labs and former Chief Architect at Loopring has proposed the concept of *Based Booster Rollup* [50]. In this approach, a smart contract only needs to be deployed once on L1, and it automatically scales across all L2 rollups. Similar to program parallelization, where multiple CPU/GPU threads run the same code with their own local memory, Booster Rollups execute transactions as if on L1, with access to all L1 state but maintaining their own storage. This way, both execution and storage are scaled on L2, while L1 acts as the shared base environment [51].

For users, this approach eliminates the dealing with fragmentation and switching between L2s. Their favorite dApps will be available on all L2s seamlessly. This design also reduces transaction costs and increases throughput, allowing users to enjoy a more scalable and secure Ethereum.For developers, it enables them to scale their dApps without redeploying on every L2. By deploying once on L1, their dApp automatically scales across all current and future boosted L2s [52].

Taiko has already implemented the Based Contestable Rollup in its protocol on both the Mainnet and Testnet, but it will soon be upgraded to the Booster Rollup design.

7.2.6 Inception Layer

Taiko has implemented Inception Layers as the core in it's protocol, where it was first introduced in the Alpha-4 testnet Eldfell. Since Taiko is a Type-1 (Ethereum-equivalent) zkEVM, it allows the same rollup protocol to be deployed on Ethereum or Taiko without requiring any changes in the code. That is, one can deploy Taiko on Ethereum as an L2, or Taiko on Taiko as an L3. Or one can deploy Taiko protocol as multiple L2s, multiple L3s, multiple L4s, etc. limitlessly scaling Ethereum.

While Taiko is not limited to being used strictly as an L3, L3 can refer to any arbitrary rollup designed for customized scaling, whereas L2 serves general-purpose scaling. For instance, L2 could be used for trustless scaling, while L3 might be applied to weakly-trusted scaling solutions, such as validium.

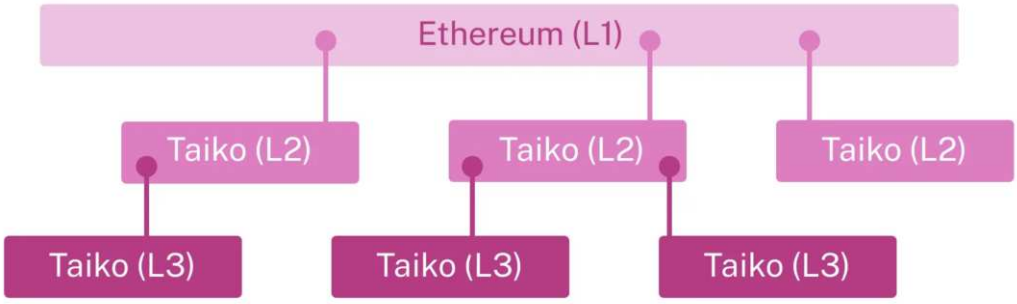


Figure 7.8: Taiko's Inception Layers [53].

7.3 Linea

Currently, Linea architecture consists of a three components which are the **Coordinator**, the **Sequencer** and the **Prover** [54]. The following diagramm is a good representation of the main components of Linea, and how they interact:

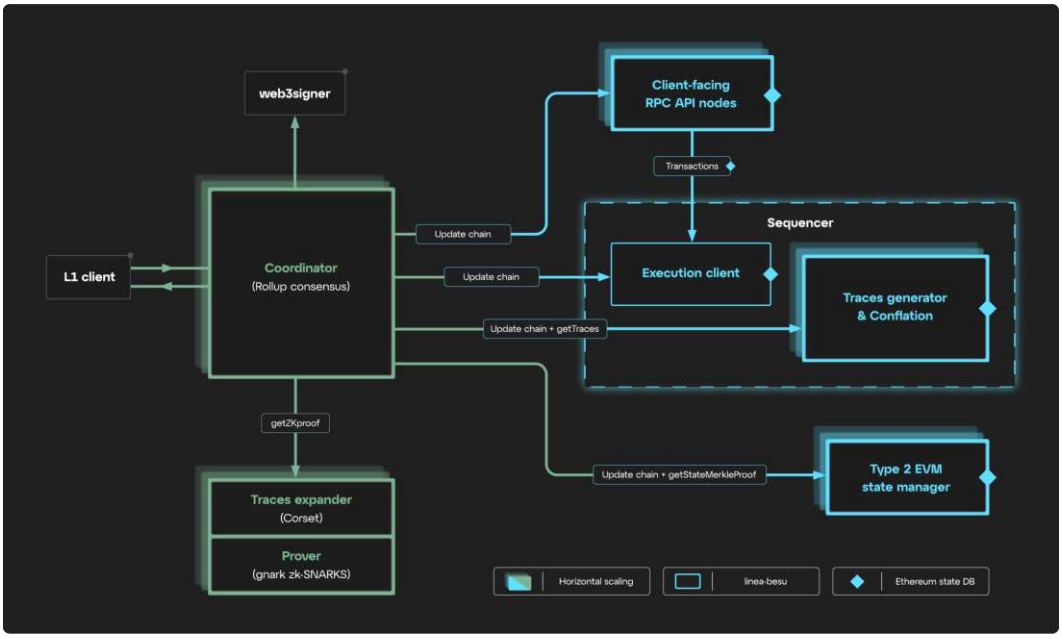


Figure 7.9: Linea Architecture.

Coordinator The coordinator is Linea's consensus client responsible for managing information both internally, between different components of Linea's execution client, and externally, with other blockchains, Linea's data availability layer, and the nodes syncing its network state [55].

Execution client The execution client is zkGeth, a version of geth that has been modified to work with zk-proving technology. However, Linea is currently building *linea-besu* that leverage the full the Consensys stack by using the same Besu client software that is used to execute blocks on Ethereum coupled with a plugin system [56, 57].

Sequencer The sequencer is part of Linea execution client and is responsible for ordering, building, and executing blocks [57]. Linea's sequencer takes transactions from the memory pool and builds them into blocks, similar to Besu (Hyperledger Execution Client) on Ethereum. However, it also works with the Coordinator to ensure the blocks are provable by the zero-knowledge prover and as compact as possible for efficient storage on Ethereum. This is handled by the *Traces Generator* and *Conflator* subsystems within the Sequencer.

Traces Generator Once the sequencer has built its blocks, they are executed; and in the process, the EVM produces data known as traces. These traces specify the state of the network, and the state of the accounts involved in the transaction, at each granular step of each transaction's execution [57]. these traces will use the prover to produce a proof. Additionally, Linea's sequencer puts these traces through an additional process: trace conflation.

Conflator Conflation is the process of combining two or more blocks' worth of transactions into one data set [57]. In a zkEVM environment, Ethereum's 'source of truth' consists of the data submitted to it, which includes the ZK proof, the list of transactions verified by the proof, and the Merkle tree. This shifts the focus from "how many transactions fit in a block" to "how many transactions fit in a proof." By conflating multiple blocks into one, Linea's proving system becomes much more efficient.

EVM State Manager The main task of the state manager is to receive blocks that have been executed by the sequencer and use the trace data from their execution to update the state of the network. Linea uses a Merkle-Patricia Trie to record the world state, maintain consensus, and process blocks. At the same time it utilize sparse Merkle tree (SMT), which is used to more efficiently track, manage, and update storage slots representing accounts [54]. It will then pass the network state to the prover in the form of Merkle proofs.

Prover The prover is responsible for generating the zk-SNARK proof by retrieving information from the Coordinator and the execution client. The next section provides an in-depth exploration of the Linea Prover implementation.

7.3.1 Prover

In 2022, ConsenSys launched a whitepaper describing their Prover implementation. This implementation incorporates complex components, including arithmetization, Arcane (a cryptographic tool), polynomial interactive oracle proofs, lattices, hash functions, error-correcting codes, and Vortex, a list polynomial commitment (LPC) [58].

The following diagram illustrates the pipeline of the Linea Prover System, which begins with arithmetization, followed by the Arcane and Vortex stages, before the proof is finalized. Each step will now be described.

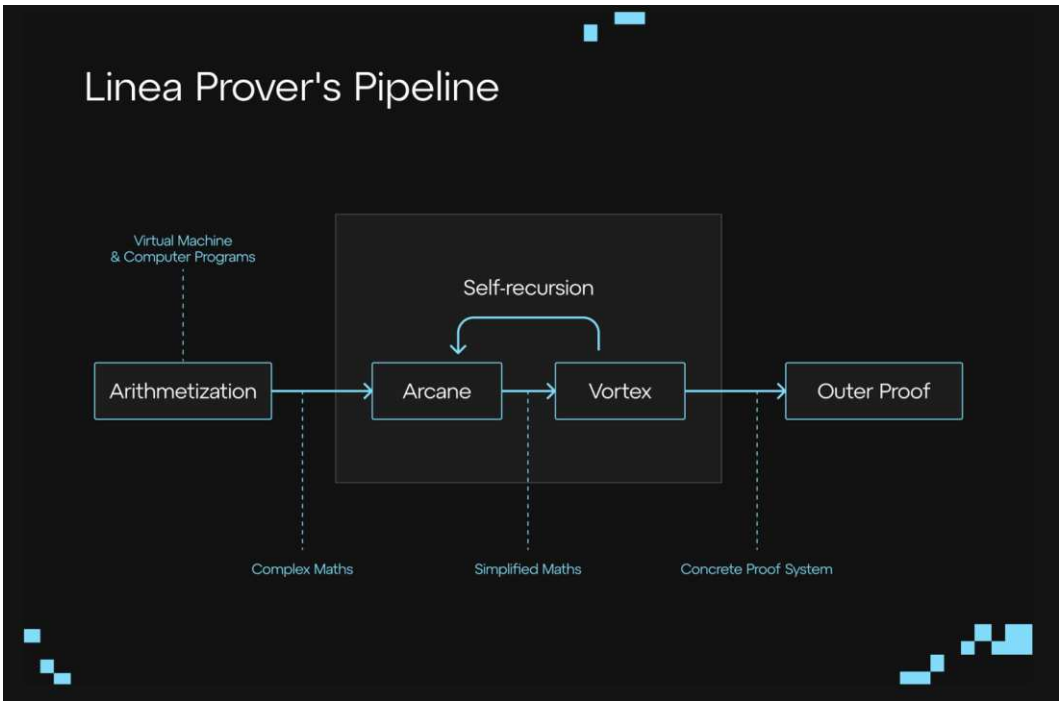


Figure 7.10: Linea Prover's Pipeline [59].

Arithmetization Arithmetization is a technique used to transform complex computational tasks into simple mathematical equations [59]. It breaks down the steps of a computer program into smaller operations that can be expressed as polynomials. For example x^{256}. With arithmetization x^{256} can be represented as series of equation like: $y_1 = x^2$; $y_2 = y_1^2 = x^4$; $y_3 = y_2^2 = x^8$; .. $y_8 = y_7^2 = x^{256}$.

The collection of these equations is called a **trace**. A trace is valid if it satisfies all the equations; if the final equation $y_8 = x^{256}$ is correct, it means the computation was done properly. Proving the execution of the EVM is the same as proving that "the prover knows a valid trace" for a specific block and state transition.

Arcane After arithmetization the traces will go through an inner-proof system that recursively shrinks down the proof. To prove a transaction on Linea, it needs to prove that some traces satisfy some constraints. To work with the prover more easily, these constraints needs to be turned into something more homogenous(i.e. polynomial evaluations). *Arcane compiles* the arithmetization into an Interactive Oracle Proof (IOP) model [60].

An IOP is a type of interactive proof where the verifier does not need to read the entire message from the prover [60]. Instead, the verifier can query an oracle—essentially a third party that possesses the knowledge the prover has—probabilistically to obtain the necessary information. Because Linea doesn't want to rely on a third party it transform the oracle into a polynomial commitment scheme.

Vortex The polynomial commitment scheme in Vortex leverages lattice-based cryptography instead of elliptic curve cryptography. This choice is driven by several factors: lattice-based hashes are optimized for recursion, efficient for hardware acceleration, and well-suited for

SIMD parallelism (a class of high-performance computing architecture) [60]. Additionally, lattice-based cryptography is quantum-resistant, offering enhanced protection against evolving technologies [60].

Despite these advantages, Vortex proofs remain resource-intensive. To address this, the system applies self-recursion to compress proofs, making them smaller and faster to verify [59]. This compression process continues until no further reduction is possible.

Outer proof system Finally, the prover wraps the compressed Vortex proof in a Plonk proof for an additional layer of optimization. PlonK is a construction based on advanced cryptographic techniques ensuring minimal computational overhead for the verifier [60]. This final compression step is key to ensuring that the proof is both small and quick to verify, making it highly efficient for use on-chain.

7.4 Polygon ZkEVM

The Polygon zkEVM architecture consists of three primary components: the "Trusted Sequencer," the "Trusted Aggregator," and the consensus contract [61]. The following diagram illustrates this architecture.

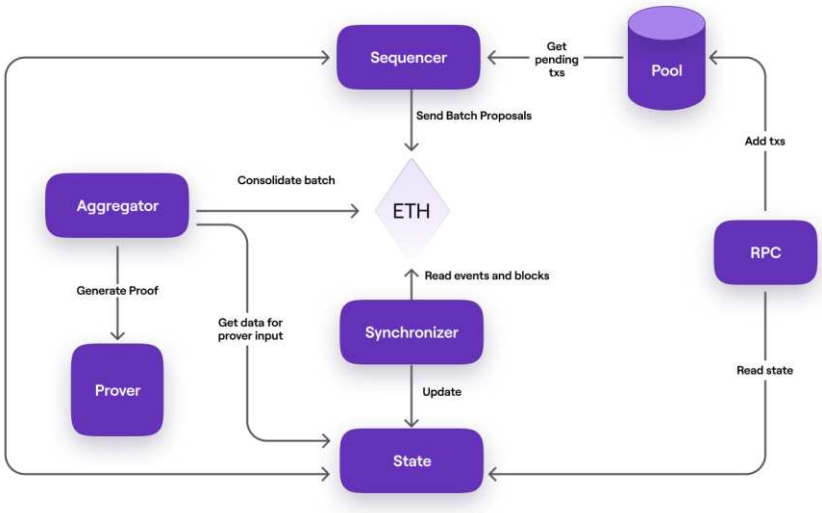


Figure 7.11: Polygon Prover Architecture.

RPC Node The RPC-Node (e.g. Metamask) is the interface for Users, who connect to the zkEVM network and submit their transactions to a database called Pool DB.

Pool DB The Pool DB is the storage for transactions submitted by Users. These are kept in the pool waiting to be put in a batch by the Sequencer.

Sequencer The trusted sequencer receives L2 transactions from users, orders them, generates blocks of transactions, fills batches, and submits them to the consensus contract's.

Trusted Aggregator The trusted aggregator computes the L2 state based on batches of L2 transactions executed by the trusted sequencer. On the other hand, the primary function of the trusted aggregator is to receive the L2 batches validated by the trusted sequencer and produce ZKPs verifying the computational integrity of these batches. It employs the Prover for this purpose.

Consensus contract The consensus contract used by both the trusted sequencer and the trusted aggregator in their interactions with L1 is the PolygonZkEVM.sol contract (in this diagram as ETH).

Synchronizer The Synchronizer is the component that updates the State DB by fetching data from Ethereum.

State The State DB is a database for permanently storing state data.

Prover The Prover is a complex cryptographic tool capable of producing ZK-proofs. A detailed explanation is provided in the next chapter.

7.4.1 zkProver

The zkProver is primarily composed of a cluster of 13 state machines (SM), including one main state machine, six secondary state machines, and six auxiliary state machines [62]. The main state machine has the capability to delegate various tasks to these specialized state machines, by sending appropriate instructions called **Actions**, depicted in the below diagram. Every Action, whether from the generic Main SM or the specific SM, must be supported with a proof that it was correctly executed [62].

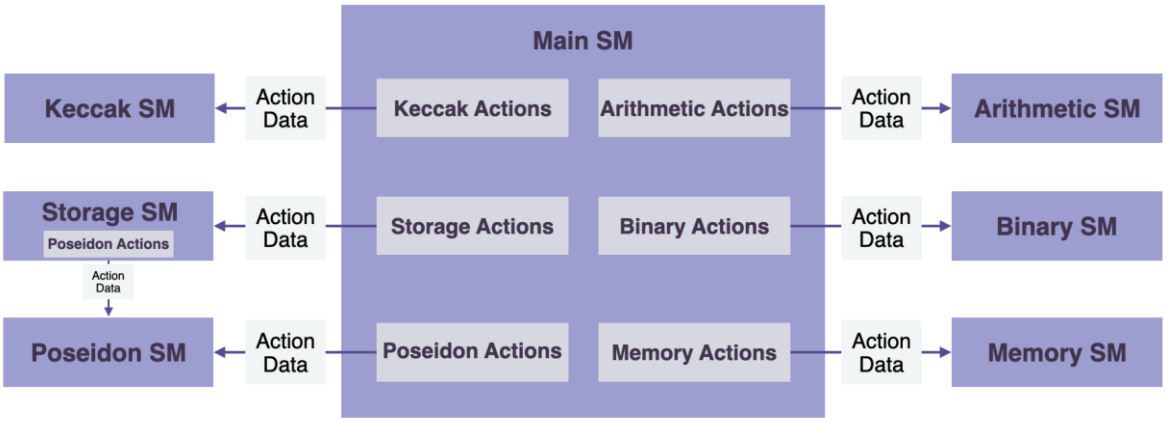


Figure 7.12: Polygon Prover's State Architecture [62].

Zero-Knowledge Assembly(zkASM) To map these instructions from the zkProver's Main state machine to other state machines, Polygon has developed the **Zero-Knowledge Assembly (or zkASM) language** [62]. zkASM codes create assembly instructions that tell the SM Executor how to perform calculations. Because the Executor follows the logic and rules of the zkASM codes closely, verifying computations is easy. In the case of state machines with firmware, zkASM serves as the interpreter for the firmware.

The Polynomial identity language (or PIL) Another language that Polygon developed specifically for the zkProver is the Polynomial identity language (or PIL) [62]. Almost all state machines express computations in terms of polynomials. Therefore, state transitions in state machines must satisfy computation-specific polynomial identities. The most reliable and effective commitment schemes till-date are the Polynomial Commitment Schemes, therefore PIL transform calculations into some polynomial language, where verification essentially comes down to verifying whether execution fulfils specific polynomial identities.

The zkProver consist of four components which are depicted in the following Figure 7.13.

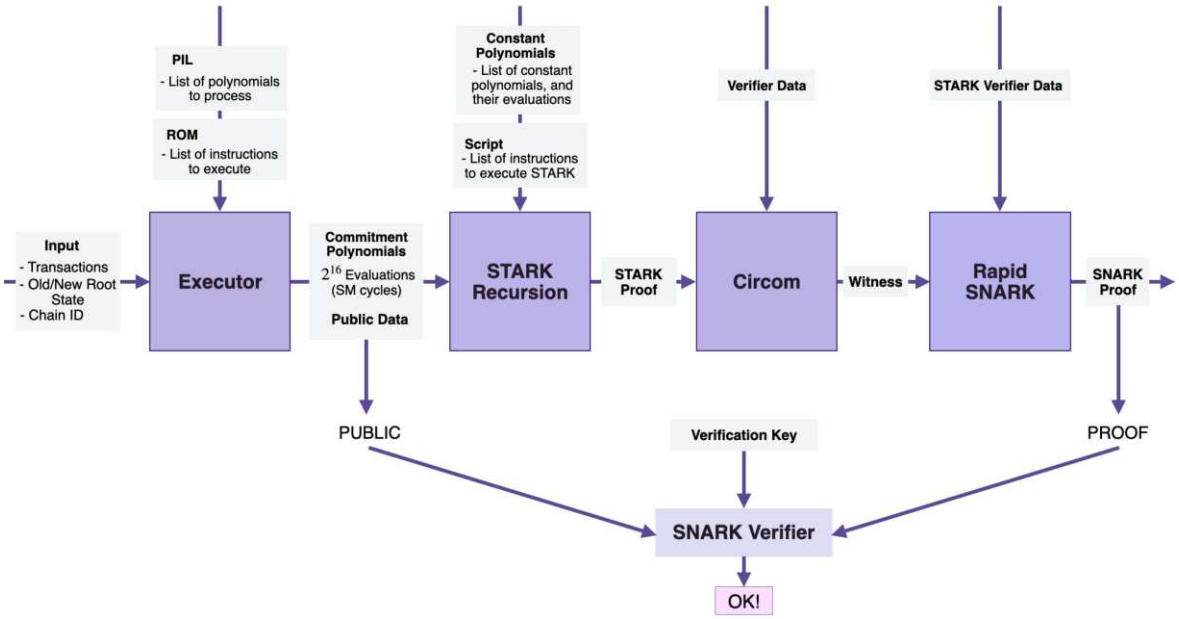


Figure 7.13: Components of Polygon's Prover [62].

The Executor The Executor or Main state machine Executor handles the execution of the zkEVM [62]. This is where EVM Bytecodes are interpreted using a the zkASM language. It takes as inputs the transactions, old and new states, along with the PIL (polynomials and registers) and the ROM (execution instructions). The Executor creates rules that each valid transaction batch must follow. Using PIL, the Executor runs instructions on the hardware and produces the committed polynomials, which represent the state machine cycles (a list of all states). It also generates public data needed by the zk-SNARK verifier.

STARK recursion component Once the Main state machine Executor has converted transactions and related data to committed polynomials, the STARK Recursion component takes some inputs and generate a zk-STARK proof [62]. It called STARK recursion because it generates multiple zk-STARK proofs, bundles them, then creates a zk-STARK proof for each bundle, eventually combining everything into a single zk-STARK proof. This way, hundreds of zk-STARK proofs are represented and proved with only one zk-STARK proof [62].

Circom The single zk-STARK proof produced by the STARK Recursion Component is the input to a CIRCOM component. CIRCOM is a language for building complex zero-knowledge circuits. It is designed as a low-level circuit language, mimicking the design of electronic circuits, for naturally defining Arithmetic circuits [63]. CIRCOM takes as input a zk-STARK proof in order to perform two tasks:

- Create an arithmetic circuit that matches the zk-STARK proof, written as an R1CS (a specific circuit format).
- Generate a "witness," which is a set of values that show the circuit works correctly with the given proof.

zk-Snark Prover Zk-STARK proofs are used because of their speed, however they are much larger compared to zk-SNARK proofs. Therefore, the last component of the zkProver is the zk-SNARK Prover [62]. The zk-Snark Prover takes as inputs the witness from the CIRCOM component and the STARK verifier data to generate a zk-SNARK proof. zk-SNARKs are therefore published as the validity proofs to state changes.

7.4.2 Aggregation Layer

On 23 February, 2024, Polygon Labs has introduced the Aggregation Layer (AggLayer), that unifies different L1 and L2 Chains to solve the scaling limitations and bad UX due to fragmented liquidity and state. From the perspective of Ethereum, all rollups are simply smart contracts holding a state root with a bunch of assets, plus a verifier that says what can go in and out [64]. In a normal multichain L2 ecosystem, there are many bridges to Ethereum—a bridge for every chain. That means that even in “unified” multichain L2 ecosystems, to transfer assets between chains, without using a third-party bridging service, requires a withdrawal to Ethereum [64].

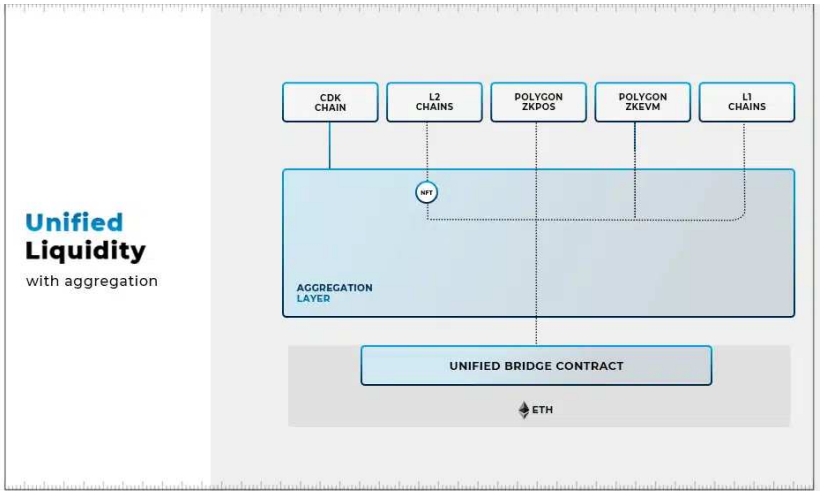


Figure 7.14: Unified Liquidity with the Aggregation Layer.

The AggLayer aggregates ZKPs from connected Layer 1 and Layer 2 chains and introduces a single unified bridge, where each chain will have a local copy of the unified bridge root, enabling cross-chain transactions that don’t require withdrawing to Ethereum or the security risks of third-party bridges [64].

By utilizing a unique three-stage process of pre-confirmation, confirmation, and finalization, the AggLayer ensures atomic transactions where users can submit transaction bundles across multiple chains with the guarantee that they will either all execute successfully or none will be included [65, 66].

Currently, the only protocol connected is Polygon zkEVM, while other chains are working to integrate into the AggLayer soon [64].

7.5 ZkSync

ZkSync is oficially catogorized as a Type-4 zkEVM. Despite not having bytecode-level equivalence as with Type-2 and Type-3 zkEVMs, ZkSync prioritizes performance and scalability. This approach allows for very fast proof generation times and also for Flexibility in Design that introduces features that are not available to other zkEVM such as having *Account Abstraction* and *LLVM Compiler*.

The main components of ZkSync architecture consist of a Sequencer and a Prover to generate validity proofs:

Sequencer ZkSync operates a centralized Sequencer that monitors Ethereum Layer 1 (L1), maintains Layer 2 (L2) state, and manages transaction ordering for the ZKsync protocol. It includes RPC services and the ETH Operator module, which features EthWatcher for tracking events, EthTxAggregator for batching L2 transactions, and EthTxManager for signing and dispatching transactions. The Sequencer organizes transactions into blocks, utilizing Tree and TreeBackup for L2 storage and StateKeeper for executing and storing transactions [67].

Prover Previously, ZkSync Era relied on a SNARK-based proof system incorporating elements of PLONK (Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge) [68]. However, ZkSync has now transitioned to a new proof system called Boojum, which is based on STARK technology. Boojum is a Rust-based cryptography library specifically designed for ZkSync Era, implementing an upgraded version of arithmetic circuits tailored to the protocol's needs [69].

The Boojum proof system offers several advantages over its predecessor. It operates on consumer-grade hardware, requiring only 16 GB of GPU RAM instead of the previous demand for 100 GPUs with 80 GB of RAM each. Boojum also wraps STARK proofs with a non-transparent pairing-based SNARK, which reduces storage needs and verification costs [69].

Currently, the Boojum upgrade is live on the ZkSync Era mainnet in an experimental phase, generating and verifying "shadow proofs" using real production data [69]. This testing process allows ZkSync Era to fine-tune the system, identify potential issues, and mitigate risks before full migration.

7.5.1 Account Abstraction

In Ethereum there exist two types of Accounts:

1. **Externally Owned Accounts (EOAs)**: Controlled by private keys, typically used by individual users.
2. **Contract Accounts:** Controlled by smart contract code.

Account Abstraction (AA) unifies these account types, enabling any account to execute arbitrary code and facilitate features like multi-signature wallets, social recovery, and custom authentication mechanisms improving the account's security. For example, In an EOA, losing private keys means losing access to all funds. With account abstraction, however, users can set custom security rules that limit unauthorized access and control transaction initiation. This feature allows defining specific validation criteria, such as requiring multiple signatures or setting transaction windows, giving users more control over how their account and transactions are managed [70].

ZkSync has a Native Account Abstraction that is built into its protocol and introduces the concept of *Smart Accounts* and *Paymasters* [70].

Smart Accounts Smart Accounts are fully programmable, allowing users for various customisations such as signature schemes, native multi-sig capabilities, spending limits, and application-specific restrictions. Simply saying users can determine how transactions should be processed in the future.

Paymasters Paymasters can sponsor transactions for users, enabling users to pay transaction fees in ERC20 tokens.

7.5.2 ZkPorter

ZkSync is taking a hybrid approach by introducing ZkPorter that makes rollups more scalable. The L2 state will be divided into 2 sides: zkRollup with on-chain data availability and zkPorter with off-chain data availability. Data availability refers to the confidence a user can have that the data required to verify a block is really available to all network participants [71].

The data availability of zkPorter accounts will be secured by ZkSync token holders, termed Guardians. They will keep track of state on the zkPorter side by signing blocks to confirm data availability of zkPorter account. In addition, Guardians participate in proof of stake (PoS) with the ZkSync token, so any failure of data availability will cause them to get slashed. It is important to note that the Guardians in ZkSync are powerless, they cannot steal funds. They can only freeze the zkPorter state [72].

Both components will be composable and interoperable: contracts and accounts on the zkRollup side can interact seamlessly with accounts on the zkPorter side, and vice versa. Each user must consider their risk tolerance and intended use case on ZkSync . For instance, a user might opt for the zkRollup to transfer 10 ETH to a DeFi application, while choosing zkPorter for a transaction to change a sword's color in a metaverse game.

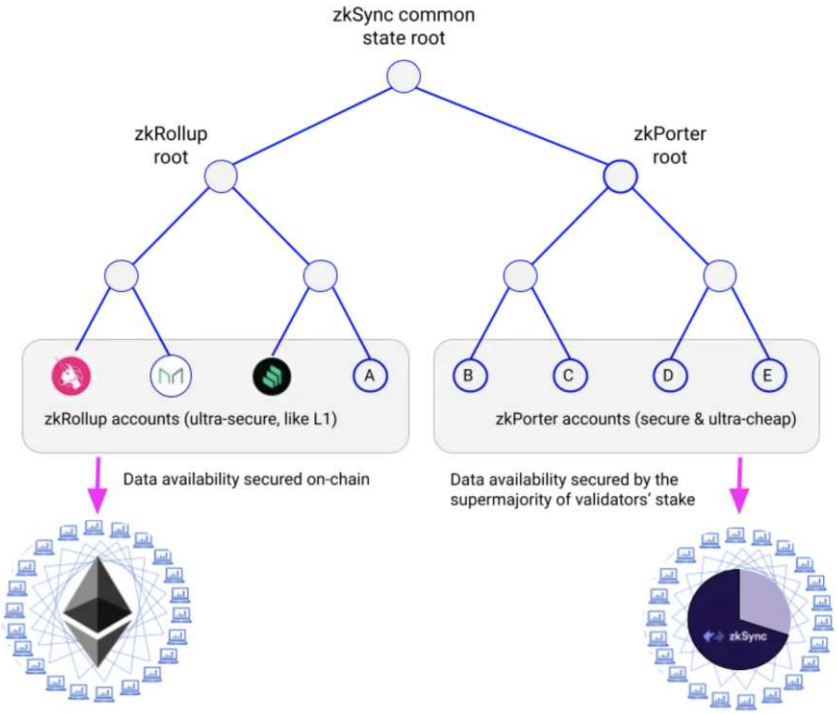


Figure 7.15: Combination of zkRollup and zkPorter [72].

ZkPorter is designed to significantly reduce fees and enhance throughput beyond ZkSync's zkRollup, offering fees over 100 times lower than zkRollup and over 10,000 times lower than Ethereum layer 1. With a theoretical maximum throughput of 20,000 transactions per second, ZkPorter achieves nearly 10 times the capacity of zkRollup and approximately 100 times that of Ethereum mainnet [73]. These cheap fees however come with a cost, because the Merkle root data is kept off-chain it will not be guaranteed to the user that they will be able to bridge over their funds to their layer 1 account. Without a complete record of the Merkle root state changes there is no way for the user to prove to the layer 1 chain who the true owner is [73].

7.5.3 Compiler Toolchain

ZkSync requires a compiler because it is a Type-4 zkEVM, meaning it is not fully compatible with Ethereum's bytecode. Instead of directly executing EVM bytecode, ZkSync introduces the zkEVM LLVM-based compiler toolchain for smart contract languages. LLVM-based compilers are compilers built on the LLVM (Low-Level Virtual Machine) framework, which is an open-source compiler infrastructure designed for flexibility, modularity, and the generation of optimized machine code [74]. ZkSync Compiler Toolchain can be seen the diagram below.

The compilation process works as follows [74]:

1. The high-level source code, written in Solidity or Vyper, is first compiled using standard compilers (such as `solc` for Solidity and `vyper` for Vyper).
2. The output from these compilers, including intermediate representations (IRs), abstract syntax trees (ASTs), and metadata, is then passed to ZkSync's custom IR compilers (`zksolc` or `zkvyper`).

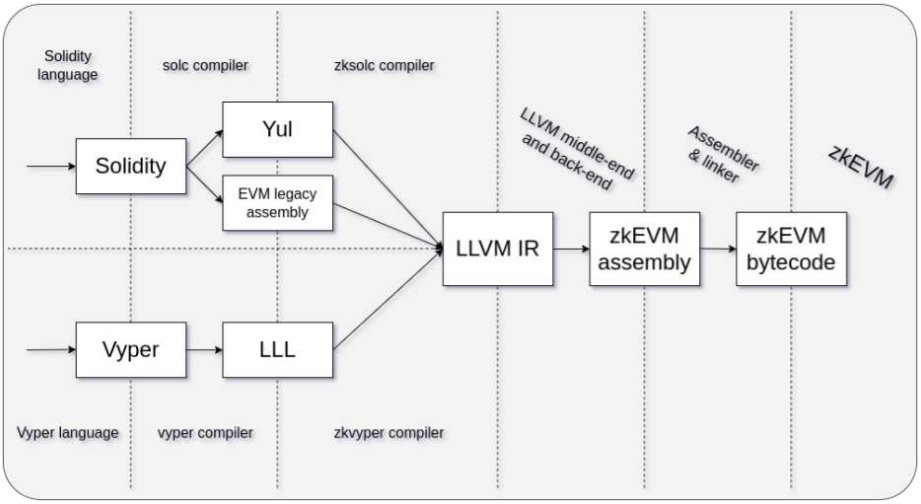


Figure 7.16: Compiler Toolchain [74].

3. These custom IR compilers translate the IR into LLVM IR, optimize it using the LLVM framework, and produce zkEVM text assembly.
4. Finally, an assembler compiles the zkEVM assembly into zkEVM bytecode.

In the future the compiler toolchain will be further developed to accept code in Rust and C++ as well.

7.5.4 Zk HyperChains

ZkSync introduces the concept of Hyperchains, that are sovereign zk-chains that run on top of Layer-1 or Layer-2 networks while remaining connected and able to derive security and finality from the main chain (L1). ZkSync's hyperchains, at its core, solve the challenge of sharing liquidity and asset movements between L2 rollups and Layer-1 blockchain [75].

Hyperchains function like an email interface, where messages can be sent from any address on any domain to a completely different address on another domain. These interactions occur almost instantaneously and with full trust, as each chain has access to the state of the others, facilitated by a shared bridge smart contract deployed on the main chain [75]. An illustration is provided below.

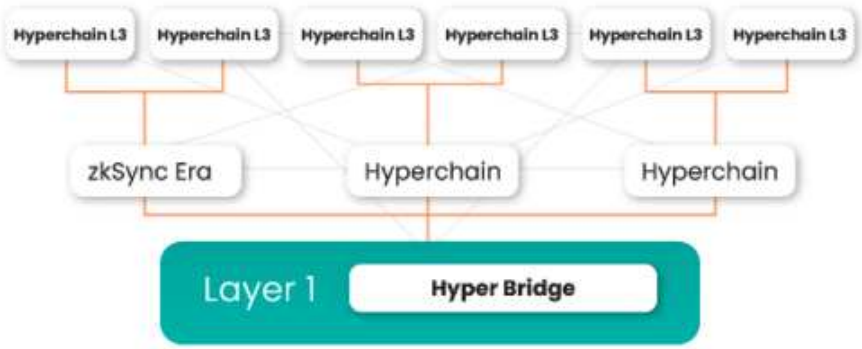


Figure 7.17: Hyperchain's Architecture and their Workings [75].

Imagine a project or developer wants to build its own rollup. They can use ZK Stack, a highly optimized, open-source, and modular framework for creating custom zk-chains on Layer-2 and Layer-3. For instance, a gaming company may want to customize its ecosystem to make it more player-centric by offering gasless transactions, setting the highest possible transactions per second (TPS) to achieve massive throughput, or configuring it as a permissionless rollup [75].

This flexibility lets developers customize their rollups for specific needs, ensuring better performance and scalability. At the same time, it maintains connectivity with other hyperchains through built-in bridges and security from Ethereum.

8 Security

While zkRollups are inherently secure due to the cryptographic strength of zk-SNARKs, which prove that all off-chain transactions are executed correctly without exposing sensitive information, they are not entirely risk-free. Potential vulnerabilities remain, such as bugs in the bridge contract, which could result in the loss of funds. Additionally, users must trust that the zkRollup operator will include their transactions promptly and without censorship, highlighting the importance of robust governance, data availability, and censorship resistance mechanisms beyond the cryptographic guarantees provided by zk-SNARKs.

This section examines Security Audits, an integral part of zkRollups, as well as Bug Bounty Programs that allow the submission of vulnerabilities in exchange for rewards. Additionally, the section covers Emergency Security Measures.

8.1 Audits

Security audits are essential for identifying vulnerabilities and errors in smart contracts across dApps, protocols, and blockchains. These audits ensure that smart contracts function as intended, free from hidden flaws or security risks. This is particularly critical for zkRollups, where significant amounts of money are often involved, and a single exploit could undermine trust in the entire ecosystem. Components such as bridges, rollup contracts, and zkEVM circuits are key areas subject to these audits [76].

One of the strengths of zkEVM projects is their focus on security audits. Each of these projects has undergone thorough testing and multiple external audits to ensure the safety of their mainnets. Trusted firms such as OpenZeppelin, ConsenSys Diligence, Trail of Bits, and Quill Audits have conducted these reviews, providing assurance of their security [77–81].

However, Polygon zkEVM is still in its mainnet beta phase, with ongoing security audits and assessments. The project has acknowledged that data and crypto-assets could be at risk due to potential bugs that are still being addressed [78].

8.2 Bug Bounty Program

DeFi has created new opportunities for users and developers. But as DeFi grows, it also becomes a bigger target for hackers looking to steal funds and assets.

Bug bounty programs help reduce these risks. They reward people for finding and reporting bugs or security flaws. Instead of exploiting issues, researchers are encouraged to share them, helping to keep the zkRollups safe.

While security audits are an essential first step, no amount of pre-launch code review can completely eliminate the possibility of bugs in a live smart contract. This is why platforms like Scroll, Linea, ZkSync, and Polygon zkEVM have launched bug bounty programs on Immunefi,

a leading platform for smart contract and DeFi security [82]. On Immunefi, researchers review code, disclose vulnerabilities, earn rewards, and help make the crypto ecosystem safer. The following table highlights the rewards for finding bugs in three categorizes: *Smart Contract*, *Blockchain & DLT* and Website & Application.

	Scroll	Taiko	Linea	Polygon zkEVM	ZkSync
Smart Contract					
Low	-	-	1000$	-	1000$
Medium	5000$	-	5000$	5000$	5000$
High	10.000$ - 50.000$	-	-	20.000$	50.000$
Critical	50.000$ - 1.000.000$	-	up to 100.000$	up to 1.000.000$	2.300.000$
Blockchain & DLT					
Low	-	-	-	-	-
Medium	-	-	-	5000$	-
High	10.000$ - 50.000$	-	-	20.000$	10.000$
Critical	50.000$ - 1.000.000$	-	-	up to 1.000.000$	500.000$
Website & Applications					
Critical	-	-	-	-	25.000$

Figure 8.1: Reward Comparison of Big Bounty Programs.

Among the platforms analyzed, ZkSync has the most extensive bug bounty program, offering rewards up to $2,300,000 for critical vulnerabilities in smart contracts. Polygon zkEVM and Scroll also provide significant rewards, up to $1,000,000 for critical issues related to blockchain and smart contracts, though their coverage is more limited. Linea's program is focused only on smart contracts, with a maximum payout of $100,000 for critical findings, and does not address blockchain or application vulnerabilities. Taiko, on the other hand, does not have a publicly available bug bounty program.

8.3 Emergency Response Mechanisms

Currently, all of these platforms are still in the early stages of development. Most have launched their mainnets between 2023 and 2024, making them less than a year old. Due to their relatively young age, security issues may arise from the inherent complexities of developing zk- zkEVMs.

To mitigate potential risks, Emergency Response Mechanisms are designed to quickly address vulnerabilities or exploits that may occur post-launch. They can also be called training wheels, that are safety features that are put during the initial phases of the rollup's deployment [83]. This includes such as automatic contract pausing, withdrawal rate limits, or multi-signature governance to halt operations in the event of a critical bugs. The website L2Beat [84] provides a helpful summary of which contracts and addresses have privileged permissions. Multi-signature contracts always require a specified number of trusted addresses to execute certain actions. The following table summarizes the contracts, the participants, and the actions they can perform.

zkEVM	Contract	Role/Threshold	Permissions
Taiko	MultiSig	Admin (3/4)	- Pause block proposals - Upgrade proxies
	Guardian Provers	Prover (6/8)	- Prove highest-tier blocks
	Minority Prover	Prover (1/8)	- Prove second-tier blocks
	Chain Watchdog	Watchdog (1)	- Pause proving blocks
Linea	MultiSig	Admin (4/8)	- Upgrade core contracts - Override proofs
	Operators	Operator (-)	- Prove blocks, post data
	Pauser	Pauser (-)	- Pause ERC20/USDC Bridge
Scroll	MultiSig	Participant (4/5)	- Upgrade proxies/verifier - Revert batches
	Executor MultiSig	Participant (1/5)	- Execute timelock transactions
	Emergency MultiSig	Participant (2/4)	- Revert, pause contracts
Polygon	Security Council	Admin (6/8)	- Pause bridge - Remove upgradeability delay
	Rollup Manager Admin MultiSig	Rollup Manager (2/3)	- Set timeouts and aggregator - Deactivate emergency state
	Escrow Admin	Admin (5/10)	- Upgrade bridges
ZkSync Era	Security Council	Admin (9/12)	- Freeze ZK stack chains - Approve governance proposals
	Guardians	Admin (5/8)	- Extend veto period - Approve/cancel L2 proposals
	Matter Labs Multisig	Admin (4/7)	- Manage validators - Revert batches

Table 8.1: zkEVM MultiSig Roles and Permissions.

The table shows how different zkEVM platforms handle governance and security through multi-signature contracts. Taiko and Scroll use layered systems where Guardian Provers focus on key tasks, and Watchdogs or Minority Provers handle smaller but important roles, like pausing blocks. Linea focuses more on upgrading core contracts and proving blocks. Polygon and ZkSync Era have more complex setups, requiring many participants to approve major actions, like pausing bridges or approving proposals.

8.4 Sequencer and Proposer

The Proposer and Sequencer are essential components of every rollup, ensuring that transactions are ordered and submitted to Ethereum. The Proposer submits blocks and transaction data to Ethereum, while the Sequencer collects transactions, processes them, and generates Layer 2 blocks. However, in most rollups, the Sequencer and Proposer are typically centralized and operated by the rollup itself [84].

This table shows how each zkEVM handles Sequencer or Proposer failures and the capabilities users have to propose blocks or transactions to be included on L1.

zkEVM	Sequencer Failure	Proposer Failure
Scroll	- No mechanism for transactions to be included	- Only whitelisted proposers can publish state roots on L1 - Withdrawals are frozen
Taiko	- Uses Based Sequencing - Can propose L2 blocks directly on the L1 contract - TaikoAdmin Multisig can pause block proposals	- Provers can examine blocks proposed on L1 - Anyone can register an SGX instance and create proofs for blocks
Linea	- No mechanism for transactions to be included	- Only whitelisted proposers can publish state roots on L1 - Withdrawals are frozen
Polygon zkEVM	- No mechanism for transactions to be included - Functionality exists but is currently disabled	- Users can submit proofs to the L1 bridge - 5-day delay for proving and finalizing state transitions - Delays can only be lowered during an emergency state
ZkSync Era	- Transactions can be submitted to an L1 queue but cannot be forced - Sequencer cannot selectively skip transactions but can halt the queue entirely	- Only whitelisted proposers can publish state roots on L1 - Withdrawals are frozen - Governance can attempt to replace proposers

Table 8.2: Handling Sequencer and Proposer Failures Across zkEVMs.

In the case of Sequencer failure, Scroll and Linea lack mechanisms to include transactions, making them highly dependent on the Sequencer's continuous operation. Taiko provides more resilience with its "Based Sequencing," allowing block proposals directly on the L1

contract and offering control through the TaikoAdmin Multisig. Polygon zkEVM also has functionality to handle Sequencer failure, though it is currently disabled, while ZkSync Era allows transactions to be queued on L1 but cannot force their inclusion. Regarding Proposer failure, Scroll, Linea, and ZkSync Era rely on whitelisted proposers and governance to handle failures, with withdrawals frozen. In contrast, Taiko and Polygon zkEVM enable greater user participation, allowing users to examine blocks or submit proofs directly to L1, although Polygon introduces delays.

9 Ecosystem

A ZK Rollup thrives on adoption—its success grows with the number of users and dApps it attracts. In this context, the term "ecosystem" refers to the network of dApps and supporting infrastructure built on these Layer 2 solutions. This ecosystem spans diverse areas such as DeFi, gaming, NFTs, and more, creating an interconnected ecosystem.

Another factor is how new projects can be supported to contribute to the ecosystem, as they play a crucial role in driving innovation and expanding the network's value. New projects, whether they are dApps, tools, or services, can help the ecosystem grow by introducing functionalities, attracting additional users, and contributing to the overall health of the network.

This section provides a detailed examination of the ecosystem, focusing on key metrics used to measure its performance. It also explores major dApp categories that contribute to its growth. Additionally, the section discusses the incentives offered by the rollup to promote adoption and development, including grants, technical support, and community-driven initiatives.

9.1 Metrics

One of the biggest challenges is measuring the strength of an ecosystem. This is difficult because of the many interconnected factors that contribute to its success, such as the number of active developers, the diversity of dApps, user engagement, and transaction volumes. Each of these elements plays a role but can be hard to quantify on its own. Additionally, external factors like market trends, regulatory shifts, and technological innovations can have a significant impact. To address this challenge, a set of metrics is defined to assess the strength of the ecosystem, with some recommendations sourced from the Chainlink Blog [85]:

- **Age:** Age is one of the fundamental indicators of a protocol's maturity and stability. Older protocols generally have more time to refine their technology, grow their user base, and attract developers.
- **Total Value Locked**: Total Value Locked (TVL) shows the overall value in US dollars of digital assets stored within a protocol. Unlike common metrics like the number of users, TVL reflects the actual value a protocol manages. This includes assets from larger investors, or "whales," who often hold a significant share of the liquidity in the protocol.
- **Number of GitHub Stars**: GitHub users are able to "star" repositories, enabling them to bookmark a repo for later use or simply to show support for a project. In addition to stars, the number of forks and contributors on GitHub repos provides further context regarding the impact of a project.
- **Unique Addresses:** Unique addresses refers to the total number of individual addresses. It's an important metric as it shows how many users have interacted with the protocol so far.
- **Daily Active Users:** Daily active users (DAUs) refers to the number of users who are active on an protocol each day.

- **Transaction Volume:** Transaction volume refers to the number of transactions made during a specific period. It will will become a key metric as those ecosystems continue to grow.
- **Number of dApps**: A higher number of dApps indicates a more vibrant and diverse ecosystem, attracting a broader user base and increasing overall adoption. Each dApp represents a unique use case, whether in finance, gaming, or social interaction, contributing to the overall utility and appeal of the rollup. The number of dApps are mainly taken from thedapplist

This data can be retrieved from various sources that provide such information, including websites like L2Beat, growthepie.xyz, theblock.so, rollup.wtf, and others [86–88]. GitHub stars are assessed based on the main repository, typically named Linea-mono, taiko-mono, or similar, which contains the core functions necessary for the rollup's operation.

	Scroll	Taiko	Linea	Polygon zkEVM	ZkSync Era
Age Mainnet	14 Month	7 Month	16 Month	21 Month	21 Month
Total Value Locked	$839.70M	$339.01M	$920.71M	$94.44M	$1.27B
Number Github Stars	719	4550	56	549	3119
Unique Addresses	6.4M	1.86M	9M	706K	2.4M
Weekly Active Users	101.77K	347K	287.24K	12.01K	208K
Transaction Volumes	100K	4M	220K	15K	180K
Number of dApps	100+	100+	300+	36	180+

Table 9.1: Comparison of zkRollups Ecosystem Metrics (Dec. 2024).

Analysis of Ecosystem Metrics

- **Mainnet Age**: Among the five protocols, Polygon zkEVM and ZkSync Era have the longest mainnet age at 21 months, followed by Linea at 16 months. Scroll, with 14 months of mainnet age, is relatively younger, while Taiko, at only 7 months, is the newest, which may indicate it is still in the process of refining its technology and attracting users.
- **Total Value Locked (TVL)**: ZkSync leads with $1.27B, followed by Linea with $920.71M. This suggests that ZkSync Era may have a larger financial footprint, while Scroll and Taiko lag behind in TVL.
- **Number of GitHub Stars**: Taiko stands out with 4550 stars, indicating a higher level of developer interest and activity compared to the others. Scroll and ZkSync Era show moderate developer engagement, while Linea's low 56 stars may suggest a smaller developer community or less public attention.

- **Unique Addresses**: Linea boasts the highest number of unique addresses at 9 million, indicating strong user adoption. In contrast, Scroll has 6.4 million, which is also significant, while Taiko and ZkSync Era have smaller user bases of 1.86M and 2.4M, respectively.
- **Weekly Active Users**: Taiko leads with 347K weekly active users, indicating a high level of engagement, far surpassing other rollups. Scroll, with 101.77K users, shows decent activity, while Linea's smaller 287.24K weekly active users suggest a more niche audience.
- **Transaction Volumes**: Taiko again leads with 4 million transactions, which could indicate higher usage or more frequent transactions. Scroll's 100K transactions are comparatively modest, showing less transaction activity. The low 15K transactions for Polygon zkEVM further suggest less adoption in this regard.
- **Number of dApps**: Linea dominates with 300+ dApps, indicating a rich ecosystem of decentralized applications. ZkSync Era has 180+ dApps, while others like Scroll and Taiko are still developing their dApp ecosystems, with 100+ dApps.

In summary, **Scroll** and **ZkSync Era** show relatively strong TVL and unique address metrics, but **Taiko** leads in terms of GitHub activity and weekly users. **Linea** shines with a large number of dApps and unique addresses, but may have room for improvement in terms of developer engagement and transaction volumes.

9.2 Decentralized Application

As of today, Scroll, Taiko, Linea, Polygon zkEVM, and ZkSync Era have developed extensive ecosystems and offer websites that allow users to browse through various categories. By analyzing these categories, they can be classified into several key domains, as follows:

1. **DeFi & Financial Services:** DeFi, Wallets, DEXs, Payments, Bridges, etc.
2. **Gaming & Entertainment:** Games, NFTs, Gambling, AI-based dApps.
3. **Infrastructure & Tools:** Dev tools, Analytics, Launchpads, Security.
4. **Community & Governance:** DAOs, Social apps, Marketplaces, Digital Identity.
5. **Privacy & Security:** Privacy-enhancing tools, Digital ID, Security services.

As observed in the table above, these rollups support a wide range of dApps.

1. **DeFi & Financial Services:** This is the most prominent category, with almost all rollups supporting dApps in DeFi, wallets, decentralized exchanges (DEXs), and bridges. Notably, ZkSync Era stands out with a broad offering that includes payments, on/off ramps, and marketplaces, while Linea focuses on centralized exchanges (CEX) and funding services.
2. **Gaming & Entertainment:** This category sees significant contributions from Scroll, Taiko, and ZkSync Era, with support for games, NFTs, and gambling dApps. Linea also includes entertainment and AI-based applications, broadening the scope of entertainment use cases.
3. **Infrastructure & Tools:** Most rollups excel in offering development tools, analytics, launchpads, and security services. Notably, Linea's ecosystem stands out with its extensive offerings in dev tools, data services, and security.

	DeFi	Gaming & Entertainment	Infra. & Tools	Community & Governance	Privacy & Security
Scroll	DeFi, Wallet, Bridges	Gaming, NFT	Infrastructure, Tooling	Community, Social	Privacy
Taiko	DeFi, Wallet	Gaming, NFT	Infrastructure, Tooling, Explorers	Community	Privacy
Linea	DeFi, CEX, Funding	Entertainment, AI, NFT	Infrastructure, Dev Tools, Data Service, Launchpad, Security	Social, Identity	Security
Polygon zkEVM	DeFi, CeX	Gaming, NFT	B2B, Utility, Tools	DAO, Social	-
ZkSync Era	DeFi, DEX, Wallets, Marketplaces, Payments, On/Off Ramps, High Risk	Games, NFT, Gambling	Developer Tools, dApp Tools, Analytics, Infrastructure	DAO, Governance, Social	Privacy, Security, Digital ID

Table 9.2: Categorization of zk Rollups by Key Domains.

4. **Community & Governance:** This category is well-represented across all rollups, with platforms like Polygon zkEVM and ZkSync Era supporting decentralized autonomous organizations (DAOs) and social applications. Linea also offers tools for digital identity management.
5. **Privacy & Security:** Privacy and security are emphasized by all rollups, although with varying focus. Scroll, Taiko, and ZkSync Era focus on privacy-enhancing tools and digital identities. ZkSync Era's strong emphasis on privacy and security services further adds to its value in this domain.

9.3 Ecosystem Incentives

New projects and builders consistently seek ecosystems that provide robust support for their development needs. This includes access to technical guidance, grants, networking opportunities, comprehensive documentation, and developer tools. Rollups encourage developers to build on their platforms by providing these resources because it benefits them. More projects mean higher usage, more transactions, and greater adoption, which strengthen the rollup's network. A strong ecosystem also attracts more users and shows how the rollup can handle real-world applications, making it more appealing and competitive.

9.3.1 Grants and Investing

Currently Taiko, Linea and Polygon Labs are offering grants and investment opportunities through different initiatives.

Scroll Scroll has implemented various incentive programs to encourage developer participation, including financial rewards and token distributions for contributors to open-source projects. In addition, Scroll regularly organizes hackathons that provide developers with opportunities to showcase their skills while offering substantial prize pools. For example, recent hackathons have featured prize pools of up to $85,000 [89].

Taiko Taiko is fostering ecosystem growth through three funding tracks. The *Community Track* supports early-stage projects on Taiko, welcoming applications from areas like gaming, media, ZKP applications, and AI. The *Partner Track* targets established projects or service providers with active users and communities, offering opportunities for integration with Taiko. The *RFP Track* invites experienced builders to work on high-impact projects requested by Taiko Labs, open to both new teams and previous grantees [90].

Linea Linea introduced the Linea Ecosystem Investment Alliance (LEIA). LEIA is an investment syndicate of more than 45 leading venture capital firms, positioned to support the builders and startups building on Linea. Beyond financial backing, LEIA also offers invaluable mentorship and technical guidance. All builders and startups are eligible to apply for funding and will receive a response within two weeks of their application [91].

Polygon zkEVM The Community Grants Program is a community grant program that supports builders, teams, and creators committed to the growth of the Polygon community. It distributes more than 1 billion POLs over a span of 10 years to projects that are built on Polygon or willing to migrate to Polygon [92].

9.3.2 Technical Support & Networking

Linea and Polygon Labs offer programs for technical support and networking. In contrast, Taiko, Scroll, and Linea do not provide direct programs. However, projects and builders can still connect with them through Discord, GitHub, or other social media platforms.

Linea When projects receive funding through LEIA, they also receive technical guidance and mentoring.

Polygon zkEVM Projects and builders can book a call with the *Developer Relations Team* for support and guidance across various domains, including App Builders, Chain Builders, Infrastructure Builders, and Researchers [93]. Additionally, builders can leverage the *Polygon Solutions Provider Network*, which offers comprehensive tools and infrastructure tailored to project needs, such as oracles, bridges, RPC providers, wallets, and more. This platform connects projects with more than 100+ solution providers in real time, enabling efficient collaboration and integration [94].

9.3.3 Community Programs

Scroll Scroll encourages community participation through two initiatives: Scroll Sessions and Scroll Canvas [95, 96]. Scroll Sessions rewards users with marks based on the value of assets they bridge and hold within the ecosystem, with extra marks for using DeFi services like

lending and liquidity provision. Scroll Canvas lets users display their achievements with non-transferable badges that represent milestones verified through the Ethereum Attestation Service. These badges serve as permanent records of accomplishments within the Scroll ecosystem.

Taiko Taiko fosters user adoption and ecosystem growth through its Trailblazers Program, which rewards users and developers for exploring areas like DeFi, bridging, and gaming, ensuring long-term engagement [97]. Additionally, Hackathons and Developer Events play a key role, with Taiko Labs regularly hosting and participating in events to strengthen its developer community.

Linea Linea has two key programs: Linea Surge and Linea Voyage [98]. Linea Surge incentivizes DeFi participation by rewarding users with LXP-L points for providing liquidity, bridging assets, staking, and participating in DEX liquidity pools. The program operates in phases called "Volts," offering higher rewards for early participants and includes a referral system. The Linea Voyage program rewards community engagement with non-transferable, soul-bound tokens (SBTs) called Voyage XPs. These tokens recognize individual contributions and unlock perks such as official community roles, exclusive Linea merchandise, and other rewards.

ZkSync Era The ZkSync Ignite program aims to boost DeFi activity on the ZkSync Era network by distributing 325 million ZK tokens over nine months to incentivize liquidity providers [99].

10 Governance

Ethereum is a well-known example where protocol decisions involve various community members, such as node operators, users, and validators. In contrast, zkRollups are still under development, with some only beginning to implement governance, making them somewhat centralized for now. Their ultimate goal is to achieve fully decentralized governance, owned and managed by the community, similar to Ethereum.

This section will explore the current governance structures of these zkRollups as well as their future plans. Over time, it is anticipated that most zkRollups will transition to decentralized governance, aligning with Ethereum's overarching vision.

10.1 Scroll

On October 22, 2024, Scroll introduced its native token, SCR, which gives users the ability to engage in the platform's governance. SCR holders can delegate their tokens to chosen representatives, who vote on their behalf, ensuring their interests are reflected in the decision-making process [100].

The governance process within Scroll DAO begins with community members proposing ideas on the ScrollDAO forum. These proposals are open for feedback and require the support of at least three participants before advancing. Once a proposal is finalized, the Governance Manager reviews it and determines whether it will be included in the next voting cycle, typically held every four weeks. This schedule helps maintain fairness and prevents voter fatigue. After voting concludes, proposals are either automatically executed on-chain through a timelock or handed over for off-chain implementation, depending on their nature and approval status.

10.2 Taiko

On July 2024, like Scroll, Taiko has introduced it's native token for users to participate in the platform governance. Currently the DAO is still in progress and under development. Taiko uses the the Aragon DAO that is built using the Aragon framework, a platform that provides tools for creating and managing DAOs on blockchain networks [101]. Mainly the Dao consist of 3 key elements:

1. **Optimistic Voting Plugin**: Only selected addresses can create proposals. These adresses belong to the two multisig's governed by the Security Council. During a 14-day delay, token holders can create a veto on the proposal. If it passes a threshold, it will be rejected; otherwise, it will be executed.
2. **MultiSig**: It allows members of the Security Council to create and approve proposals. After 3 approvals are registered, they are relayed to the Optimistic Token Voting plugin.
3. **Emergency Multisig**: Smilarly, this allows Security Council members to create and approve proposals. If 6 out of 8 signers approve them, proposals can be relayed to the Optimistic Token Voting plugin with a delay period of 0 seconds, which allows for immediate execution.

10.3 Linea

Currently Linea doesn't have a Governance Framework or DAO in place. However, it has formed a Swiss non-profit organization to manage its development and governance [102]. This Swiss Association model provides community members, including token holders, with legal rights. Furthermore, Linea plans to launch its token in Q1 2025 and has proposed transitioning to a proof-of-stake model for block validation, along with implementing an auction system for block proposers.

10.4 ZkSync Era

On June 2024, ZkSync introduces *ZKNation*, a new governance community, that allows participants to actively engage in the development and governance of the ZKsync protocol [103]. Through this initiative, token holders can delegate authority, propose upgrades, negotiate on key issues, and vote using ZKNation's on-chain governance framework.

The governance system, a *Three-Body Governance*, includes three on-chain groups [103]:

1. **Token Assembly**: Made up of token holders and their delegates, who propose and vote on protocol updates.
2. **Guardians**: A group dedicated to upholding ZKsync's values, with powers to veto or take emergency actions. It will start with at least five members.
3. **Security Council**: A team of 12 experts who ensure protocol security, review updates, and can temporarily pause the protocol in case of threats.

When emergency happens, they security council can freeze layer-1 assets through a multisig (with a threshold of 12hours) or making emergency upgrades necessary, based on security assessments. Also, Guardians have the power to veto governance proposals that they deem harmful, malicious, or misaligned. Working alongside the Token Assembly and Security Council, the Guardians act as a check against potential governance attacks or decisions that could compromise ZKsync's integrity. Their role is to ensure that the protocol's development and governance remain true to its founding principles, effectively balancing decentralized decision-making with protection against adversarial actions.

10.5 Polygon zkEVM

Polygon Labs has established a comprehensive Governance Framework designed to facilitate updates and improvements to the Polygon protocols. Through the Governance Hub, users can explore the three core pillars of governance [104], which are:

1. **Protocol Governance** that describes the The Polygon Improvement Proposal (PIP) framework. It introduces Improvement Proposals (PIP) that are documents that describe standards for the Polygon ecosystem and the processes through which the Polygon community introduces.
2. **System Smart Contract Governance** that facilitates the upgrades of protocol components that are implemented as smart contracts.

3. **Community Treasure Governance** facilitates ecosystem and public goods funding for the longevity of Polygon protocols. It proposes a Community Treasury Board that will govern the Community Treasury to fund projects and public goods.

Especially for Polygon zkEVM the second Pillar is important, because Polygon's zkEVM are essentially smart contracts deployed on Ehteruem layer 1.

10.5.1 System Smart Contract Governance

Within the System Smart Contract Governance, the Polygon Protocol Council plays a critical role. It comprises 13 members who act in the ecosystem's best interest and manage a multi-signature wallet. The Council has three primary governance responsibilities [104]:

1. Executing regular upgrades
2. Executing emergency upgrades
3. Ensuring transparency through Council Transparency Reports for both regular and emergency upgrades

Authority is delegated to the Protocol Council, which creates on-chain proposals based on PIPs (Polygon Improvement Proposals) submitted by the community. When an upgrade is queued for the timelock, a community voting window opens. If there is no majority opposition, the upgrade is automatically executed after the timelock expires. However, if the quorum for majority opposition is met, the upgrade is canceled.

The standard timelock duration is set to 10 days, with a majority consensus requiring 7 out of 13 members. For emergency upgrades, a supermajority consensus of 10 out of 13 members is required.

10.6 Summary

While some protocols, like Scroll and ZkSync Era, have already introduced governance tokens and structured voting processes, others, such as Taiko and Linea, are still building their frameworks.

Key governance features include token-based voting, multi-signature councils, and proposal evaluation systems, balancing decentralization with security. Protocols like Polygon zkEVM demonstrate how governance can integrate technical upgrades with community decision-making through transparent processes.

Overall, while zkRollups are still maturing, their governance models reflect a clear ambition toward decentralized control, inspired by Ethereum's well-established community-driven framework.

11 Smart Contracts

This section will test a smart contract written in Solidity and deploy it on each platform. The smart contract is a sample implementation of an ERC-1155 token, featuring minimal functions.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC1155/ERC1155.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Strings.sol";

contract TokenContract is ERC1155, Ownable {
    using Strings for uint256;

    event TokenMinted(uint256 indexed tokenId, address indexed receiver, uint256
        amount, string tokenURI);

    string public name;
    uint256 private nextTokenId;

    struct TokenDetails {
        uint256 totalSupply;
        string uri;
    }

    mapping(uint256 => TokenDetails) private tokenDetails;

    constructor(string memory _name, string memory _baseURI, address _owner)
    ERC1155(_baseURI)
    Ownable(_owner) {
        name = _name;
    }

    function mint(
        address _receiver,
        uint256 _amount,
        string memory _tokenURI
    ) external onlyOwner returns (uint256) {
        require(_amount > 0, "Amount must be greater than 0");
        uint256 tokenId = nextTokenId++;
        tokenDetails[tokenId] = TokenDetails({
            totalSupply: _amount,
            uri: _tokenURI
        });
        _mint(_receiver, tokenId, _amount, new bytes(0));
        emit TokenMinted(tokenId, _receiver, _amount, _tokenURI);
        return tokenId;
    }
}
```

Listing 11.1: A simple ERC-1155 Contract.

When comparing zkEVMs, the primary focus is on smart contract deployment, associated costs, and the process of bridging funds, as well as the ability to run on these networks. Testing a full dApp on each zkEVM would be overly complex for this comparison. Developing and testing dApps on each zkEVM would involve managing different tools, infrastructure, and setups for each network, making the process time-consuming and difficult to handle. Instead, by focusing on the costs of deploying smart contracts and evaluating their functionality, a clear understanding of each zkEVM's performance can be achieved without the complexity of building and testing full dApps. This approach simplifies the comparison and allows for a focus on key factors that impact the execution of smart contracts. It is important to note that these tests were conducted in early November 2024, and conditions may have changed since then.

11.1 Bridging

When developing smart contracts on zkEVMs, the initial step is to bridge assets from Ethereum to the target platforms. Bridging from Layer 1 to Layer 2 typically incurs a cost ranging from $5 to $15. To manage costs and increase flexibility, smart contracts will be deployed on the Sepolia or Holesky testnets. Free testnet tokens will be obtained from a faucet and bridged to the corresponding Layer 2 testnet using each zkEVM's official testnet bridge. It is important to note that deploying on testnets does not fully reflect real-world fees and costs; however, it provides a useful estimate of potential fee variations across different zkEVM implementations. Furthermore, fees are influenced by network utilization and traffic, which means they may fluctuate under real conditions.

11.2 Development

After successfully bridging assets to these networks, the next step is to deploy the smart contracts using Remix. Remix is an open-source, web-based integrated development environment (IDE) primarily used for writing, testing, debugging, and deploying smart contracts on the Ethereum blockchain [105]. For testing purpose, this is enough, whereby for production ready smart contract development hardhat [106] can be used instead.

11.2.1 Taiko

Deploying on Taiko is straightforward. As a Type-1 zkEVM, Taiko allows smart contracts to be deployed without modification, enabling direct function calls as on Ethereum. Taiko's testnet currently operates on Holesky, Ethereum's newest testnet, launched in September 2023 to replace Goerli. Holesky serves as a network for testing validation and staking and is open for users who wish to test protocol upgrades before deploying them on the mainnet.

Holesky added support for EIP-4844 (proto-danksharding) in February 2024, an important upgrade to help Ethereum scale. By using Holesky, Taiko can test its Layer 2 solution in an environment closer to the main Ethereum network, especially in areas like validator participation and future upgrades.

11.2.2 Linea

When attempting to deploy the same code to the Linea Sepolia test network, an error stating "invalid OPCODE PUSH0" was encountered. According to Linea, a downgrade to Solidity version 0.8.19 was necessary because the PUSH0 opcode, introduced with Ethereum's Shanghai upgrade, became available starting with Solidity version 0.8.20. Linea currently supports code compiled only with version 0.8.19 or lower.

As a result of this version downgrade, adjustments to the import statements and constructor function were required. The current version of OpenZeppelin mandates Solidity 0.8.20 or higher, and in this version, the Ownable constructor no longer requires arguments. Consequently, the necessary changes include:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v4.9/
    contracts/token/ERC1155/ERC1155.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v4.9/
    contracts/access/Ownable.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v4.9/
    contracts/utils/Strings.sol";

....
   constructor(string memory _name, string memory _baseURI)
    ERC1155(_baseURI)
    Ownable() {
        name = _name;
    }
....
}
```

Listing 11.2: Version Downgrading to Solidity Version 0.8.19.

11.2.3 ZkSync

ZkSync is a Type-4 zkEVM, meaning it is EVM-compatible but not EVM-equivalent. As a result, the default Solidity compiler in Remix cannot be used directly. Instead, the built-in ZkSync plugin in Remix must be utilized, as it includes the required compiler to compile the smart contract. After compiling the contract with the appropriate ZkSync compiler, the smart contract was successfully deployed on the ZkSync Era Sepolia testnet.

11.2.4 Scroll

Just like the Taiko, there is no change in the development process. Scroll is deployed on the Sepolia Testnetwork.

11.2.5 Polygon zkEVM

Polygon zkEVM is deployed on a Sepolia-anchored testnet called Cardona. This means that Polygon uses Sepolia as the root (L1) chain for its zkEVM network. Developers deploying on Cardona can continue to rely on the availability of essential resources such as validators, infrastructure, faucets, and tooling in a sustainable and future-proof environment.

Additionally, Polygon zkEVM does not support Solidity versions higher than 0.8.19. While smart contract deployment is successful, the mint function fails to execute with Solidity 0.8.20 or higher. As a result, downgrading to version 0.8.19 was necessary, similar to the approach taken with Linea.

11.3 Cost

Cost is a critical factor when selecting a zkEVM. Even the most secure or decentralized zkEVM may encounter adoption challenges if transaction fees are prohibitively high. This section compares fees across various zkEVMs, each employing a distinct methodology for fee calculation. It is important to note that the following table reflects fees observed on testnets, which may not accurately represent real-world costs. For reference, the data was gathered on **14th November 2024**, with Ethereum priced at **3000 USD**. Table 11.1 shows the costs of contract creation and token minting across selected zkEVM testnets.

Action	Scroll	Taiko	Linea	Polygon zkEVM	ZkSync
Testnet	Sepolia	HoleSky	Sepolia	Cardano	Sepolia
Create (USD)	266.83	0.77	0.47	0.15	0.32
Mint (USD)	13.88	0.04	0.02	0.01	0.01

Table 11.1: Fee Comparison for Contract Creation and Minting across zkEVMs (Testnets).

The fee comparison across zkEVM testnets shows significant differences in the costs of contract creation and token minting. Scroll has the highest fees, with contract creation costing $266.83 and minting $13.88, while Polygon zkEVM has much lower fees. However, these values raise questions about how accurately testnet costs reflect the conditions on the mainnet. To gain a more accurate estimate of real gas fees, an independent estimation will be conducted.

To ensure a more consistent fee comparison, the gas usage and average gas prices have been analyzed, as shown in Table 11.2. This analysis aims to provide a clearer understanding of the fees under more realistic conditions.

Action	Scroll	Taiko	Linea	Polygon zkEVM	ZkSync
Create (Gas Used)	2,367,942	2,367,942	2,873,006	2,873,006	4,242,588
Mint (Gas Used)	123,161	123,161	123,741	122,671	130,884
Gas Price (Gwei)	0.07	0.20	0.16	0.50	0.045

Table 11.2: Gas Usage and Prices for Contract Creation and Minting across zkEVM Testnets.

Gas prices depend on multiple factors, including network utilization, rollup design, execution costs, and proof generation requirements. For example, Polygon zkEVM's high testnet gas price suggests heavy usage or increased complexity due to its zero-knowledge prover. In contrast, Scroll and ZkSync exhibit lower gas prices, making them more cost-efficient at the observed time.

To better understand fee estimation, consider Scroll's testnet data. With an average gas price of 0.07 Gwei, the cost of deploying a contract consuming 2,367,942 gas is calculated as follows:

$$\text{totalFee} = (0.07 \times 10^{-9} \times 2,367,942) + 0.00002 = 0.000189\,\text{ETH}.$$

At an Ethereum price of 3000 USD, this translates to:

$$0.000189 \quad \text{ETH} \times 3000\,\text{USD/ETH} = 0.57\,\text{USD}.$$

For simplicity, the same formula can be used to calculate fees on other chains. However, it is important to note that each rollup may use a different formula. For example, Taiko incorporates both proposer and prover fees in its calculations, which could result in variations in the overall fee structure compared to other zkEVMs. These differences should be considered when making fee comparisons across different platforms.

Action	Scroll	Taiko	Linea	Polygon zkEVM (	ZkSync
Create	0.57	1.48	1.44	4.37	0.64
Mint	0.09	0.13	0.06	0.24	0.08

Table 11.3: Fee Comparison for Contract Creation and Minting across zkEVMs (Mainnet Estimation).

Looking at the table, Polygon zkEVM stands out as the most expensive chain for deploying smart contracts because of its high gas price. Taiko and Linea have similar costs, while Scroll and ZkSync are the cheapest options. Interestingly, the smart contract on ZkSync used 50% more gas on ZkSync compared to other chains. This could be because of ZkSync's design, which may need more gas for storage or state changes. On average, deploying a simple smart contract costs around $1.70.

12 Szenarios

Every company or project has different requirements when selecting a zkRollup that meets its needs. Some require extensive technical support, others prioritize full compatibility with Ethereum, while certain projects focus on minimizing fees or achieving faster transaction finality. Evaluating these criteria helps identify which zkRollup solution best aligns with a project's goals.

To clarify recommendations, common project scenarios and their objectives will be defined. This approach helps match specific zkRollups to project needs. For instance, building a trading plattform, migrating projects, or launching an NFT marketplace may require different features such as scalability, low fees, or interoperability. Mapping these objectives to suitable zkRollups ensures more informed decision-making. Considering trade-offs like decentralization, governance structures, and ecosystem maturity further supports selecting the optimal solution based on business goals.

12.1 Scenario 1: Startup Building a Decentralized Web3 Application

A startup is building a decentralized Web3 application and must select a zkRollup solution to launch and scale its platform.

Objectives

- **Rapid Development:** The startup aims to launch quickly with minimal setup and development overhead.
- **Access to Funding:** The startup seeks a zkRollup ecosystem with grants, accelerator programs, or ecosystem funds to secure financial resources.
- **Ecosystem Connectivity:** The zkRollup should have a vibrant ecosystem with active projects, partnerships, and collaborations, offering opportunities for networking and co-development.

When analyzing the objectives, several important factors come into play. First, startups need funding to quickly bootstrap their businesses. To achieve this, they require a strong ecosystem that offers partnerships and technical support. Technical superiority, protocol speed, or compatibility with Ethereum is often a secondary priority. Additionally, the maturity of an ecosystem typically correlates with the age of the zkRollup. The longer a zkRollup has been in the market, the more likely it is to have an extensive ecosystem.

When examining the Table 9.1, several important metrics stand out. Total Value Locked (TVL) and the number of dApps are key indicators of a strong ecosystem. A high TVL reflects user trust in the protocol for managing their assets and funds, while a large number of dApps indicates that other projects see potential in the zkRollup. In this context, notable examples include **Linea**, **Scroll**, and **ZkSync**.

Diving deeper into Grants and Investing Opportunities (as mentionend in section 9.3.1), ZkSync does not offer official incentives for developers to build on its protocol. Currently, only Linea has established the Linea Ecosystem Investment Alliance, enabling startups to raise capital, receive mentorship, and participate in online and onsite networking events.

Although Scroll does not have a dedicated funding program like Linea, its ecosystem provides access to developer grants through partnerships and community-driven initiatives. Developers can participate in hackathons organized by Scroll, where they can showcase their projects, gain industry recognition, and compete for rewards such as financial support, mentorship, and development resources.

In summary, **Linea** and **Scroll** are ideal choices for startups needing funding and ecosystem support. Linea offers the Linea Ecosystem Investment Alliance for capital, mentorship, and networking opportunities, while Scroll provides access to developer grants and hackathons, fostering community-driven support and recognition.

12.2 Szenario 2: A company wants to build a Trading plattform

Objectives

- **Low Fee:** The platform should minimize transaction fees to attract a larger user base and encourage frequent trading.
- **High Transaction Output:** Ensuring high throughput is critical to handling large trading volumes, especially during peak market activity.
- **High Liquidity:** The platform must maintain deep liquidity to enable efficient trading and minimize slippage.

In this scenario, the company has already been established. For example, a bank might want to offer a trading platform to its customers, enabling them to trade various assets with one another. These assets can include cryptocurrencies, stocks, bonds, and other financial instruments.

To provide the best user experience, the bank aims to offer low fees, high transaction throughput, and deep liquidity. Low fees ensure that customers can trade frequently without incurring significant costs, making the platform attractive for both retail and institutional investors. High transaction throughput guarantees that trades are processed quickly. High liquidity ensures that customers can execute trades at favorable prices with minimal slippage, even during peak trading periods.

Among all the zkRollups, **ZkSync** is the most favorable choice. One reason is that ZkSync has the highest TVL in its protocol, which minimizes slippage while trading. While the fees are quite similar to those of other protocols, as discussed in section 11.3, ZkSync offers a feature that others do not have: zkPorter (section 7.5.2). Traders who use zkPorter can benefit from a 100x reduction in fees and 10 times higher throughput compared to a normal zkRollup. While using zkPorter is less secure, traders who don't require absolute security might find it valuable.

In summary, **ZkSync** is the best choice for a trading platform due to its high TVL, which minimizes slippage, and its zkPorter feature, which significantly reduces fees and increases throughput. While security may be a concern in zkPorter, ZkSync offers a strong balance of speed, low fees, and high throughput, making it ideal for traders.

12.3 Szenario 3: A company wants to migrate its Ethereum native application to a zkRollup

Objectives

- **Compatibility:** The migration should target zkEVMs that maintain compatibility with existing Ethereum standards.

In this scenario, the company already operates a successful Ethereum-based application and seeks to migrate to a zkRollup for enhanced scalability, cost efficiency, and performance. The chosen zkRollup should be compatible with Solidity smart contracts and EVM bytecode, allowing the company to leverage their existing codebase without requiring significant changes. Additionally, the use of familiar development tools such as Hardhat,Truffle, Explorers would facilitate a faster migration process.

To effectively fulfill their objectives, section 5.3 provides an overview of the different types of zkEVMs. Only Type-1 zkEVM are fully Ethereum-equivalent, which makes them fully compatible with all Ethereum-native applications. **Taiko** is currently the only Type-1 zkEVM that is fully Ethereum equivalent. While it's technically an optimistic rollup with a growing ecosystem, projects migrating to Taiko will still benefit from fast and easy migration.

After Taiko, **Polygon zkEVM** also offers Type-2 zkEVM compatibility. While it doesn't provide full equivalence with Ethereum, it achieves a high level of compatibility with the Ethereum ecosystem. Projects migrating to Polygon zkEVM may need to make some adjustments to their applications, but they can take advantage of a strong developer ecosystem thanks to Polygon's brand recognition and extensive network of partnerships. Additionally, they can benefit from technical guidance through Polygon's Developer Relations teams and gain networking opportunities through the Polygon Solutions Network Provider.

In summary, when migrating an existing Ethereum-based application to a zkRollup, **Taiko** and **Polygon zkEVM** are the top choices. Taiko offers full Ethereum compatibility, making it ideal for a seamless migration with minimal changes to the existing codebase. On the other hand, Polygon zkEVM, while not fully Ethereum-equivalent, provides strong compatibility and a well-established developer ecosystem, which can assist with the migration process.

12.4 Szenario 4: A company wants to build a Decentralized NFT Marketplace

Objectives

- **Security and Decentralization:** The marketplace must offer secure transactions and user control over digital assets, even in the event of a sequencer or proposer failure. The project should prioritize a zkRollup with decentralized validators to ensure the platform remains operational and assets remain accessible.

When developing an NFT marketplace, the platform must provide proof of ownership for each NFT to ensure that the rightful owner is recognized at all times. This includes digital artwork, intellectual property, or other tokenized assets.

In most zkRollups, the proposer or sequencer is operated by the rollup itself. And In case of failure, processing transactions becomes nearly impossible, leading to halted trading, auctions, and new listings. This can cause market disruptions and financial losses for creators and traders, while also risking delays or even data loss in extreme cases. Therefore, ensuring security and continuous functionality is extremly important.

As discussed in section 8.4, **Taiko** allows users to self-propose blocks and transactions in case of the rollup's failure. This makes it an ideal candidate for developing an NFT marketplace. In addition, contestable proofs (section 7.2.4) ensure that incorrect or malicious state updates can be challenged. In an NFT marketplace, when a fraudulent transaction is submitted, contestable proofs allow users or third parties to challenge the transaction, protecting rightful ownership.

In summary, the ability to self-propose blocks and transactions in case of failure, along with contestable proofs, ensures that an NFT marketplace stays secure and functional even during disruptions. These features allow users to challenge fraudulent transactions, protecting ownership records and preventing financial losses. Platforms using Taiko's zkRollup technology are therefore more resilient and reliable for NFT trading.

12.5 Scenario 5: A Project Wants to Participate in the Governance to Make Decisions that Benefit Their Project

Objectives

- **Active Participation in Governance:** The project intends to take part in governance processes such as proposing and voting on protocol changes, ensuring that the decision-making process reflects its needs and objectives.
- **Influencing Ecosystem Growth:** By participating in governance, the project aims to shape the ecosystem in ways that benefit its development, such as promoting better scalability, reducing fees, or enhancing security features.

For a project looking to engage in decentralized governance, it is crucial to select a zkEVM that has a robust governance or DAO framework in place. A prime use case for governance participation is when a project needs to influence the scalability of the zkEVM platform it relies on. Let's imagine a DeFi application that has grown rapidly and now faces challenges due to transaction congestion or high gas fees. The project's team may need to propose an upgrade to the zkEVM protocol to optimize its scalability, either through technical improvements, changes to the consensus mechanism, or other enhancements.

To implement such an upgrade, the project would need a governance structure that allows it to submit and vote on proposals, ensuring that the upgrade benefits its specific use case and aligns with the broader community's interests. This governance process must be transparent, secure, and offer adequate mechanisms for community feedback and decision-making, so that both the project and the ecosystem can scale effectively and efficiently.

In this case, **ZkSync**, **Polygon zkEVM**, and **Scroll** are ideal choices. ZkSync's *ZKNation* framework provides a three-layer governance, while Polygon zkEVM offers a comprehensive governance model through its *System Smart Contract Governance* process, allowing projects to submit proposals and execute protocol upgrades. Scroll, with its community-driven governance, also supports efficient proposal submission and voting, making it a suitable option for projects seeking to drive protocol upgrades. Together, these three zkEVMs provide the governance features necessary for a project to influence and implement scalability enhancements.

So in summary, For a project looking to influence scalability upgrades, **ZkSync**, **Polygon** zkEVM, and **Scroll** are the best options. Each provides a transparent and secure governance model that allows projects to propose, vote on, and implement necessary changes to improve scalability.

13 Summary

In this paper, an extensive analysis of various zkEVMs, including Scroll, Taiko, Linea, Polygon zkEVM, and ZkSync Era, was conducted. The underlying EVMs that power each rollup were examined in detail. Following this, an overview of Zero-Knowledge Proofs (ZKPs), zkRollups, zk-SNARKs, zk-STARKs, and zk-circuits was provided, emphasizing the foundational proof mechanisms for each zkEVM.

IIn Chapter 5, the challenges in developing zkEVMs were examined, such as the complexity of supporting elliptic curves, the 256-bit word size of the EVM, and the inefficiencies in Ethereum's storage layout. Key advancements in zkEVM development were also highlighted, including the use of polynomial commitment schemes, recursive proofs, and hardware acceleration, which have made zkEVMs more feasible and efficient. Additionally, the different types of zkEVMs were introduced, ranging from Type-1 (fully Ethereum-equivalent) to Type-4 (high-level language equivalent).

In Chapter 6, the zkRollups were categorized into different types in a clear and accessible table, with an introduction to the rollups discussed throughout the paper. Chapter 7 provided an in-depth analysis of each architecture, its components, prover implementation, and features. In Chapter 8, the importance of security was discussed, emphasizing the necessity of audits, the role of bug bounty programs, emergency response mechanisms, and how each rollup addresses failures in the sequencer or proposer.

In Chapter 9, the ecosystem of zkRollups was examined, focusing on key metrics to monitor, the types of dApps deployed, and how zkRollups provide ecosystem incentives such as grants, investment opportunities, technical support, networking, and community programs.

Chapter 10 reviewed the governance structures of each zkRollup. It explored Scroll's community-driven governance via SCR tokens, Taiko's evolving DAO built on Aragon, Linea's Swiss-based governance framework, ZkSync Era's three-body governance system, and Polygon zkEVM's structured upgrade and proposal system.

In Chapter 11, a simple ERC-1155 smart contract was deployed on various zkEVM platforms to compare compatibility, minting processes, and deployment costs using Solidity versions 0.8.19 and 0.8.20. A custom compiler was required for the ZkSync deployment. Polygon zkEVM was found to have the highest deployment costs, while the other platforms exhibited only slight cost differences.

Lastly, In Chapter 12, a scenario-based approach was used to match zkRollups to project requirements. By evaluating factors like scalability, fees, and ecosystem maturity, the best options were identified for various use cases. Linea and Scroll were recommended for Web3 startups, ZkSync for trading platforms, Taiko and Polygon zkEVM for Ethereum app migrations, Taiko for NFT marketplaces, and ZkSync, Polygon zkEVM, and Scroll for governance participation. The following Table 13.1 highlight again the advantages and disadvantages of each zkRollup.

Platform	Advantages	Disadvantages
Scroll	- Strong developer ecosystem via incentives, hackathons, and grants. - High liquidity/TVL through community programs like Scroll Session. - Released Governance Framework. - Integrated with major Ethereum protocols, including DeFi, NFTs, and gaming.	- zkEVM Type-3: Limited compatibility with some dApps/tools.
Taiko	- Based sequencing ensures L1 handles proposing and sequencing. - Supports self-proposing and self-sequencing during sequencer failure. - Fully compatible with Ethereum-native applications, tools, and infrastructure. - Multi-proof system enhances proof security. - Proofs are contestable by anyone.	- Technically an optimistic rollup. - No governance in place (actively under development). - Ecosystem still expanding. - High computational costs for generating zk-proof.
Linea	- Developed by Consensys with seamless integration into MetaMask and Infura. - Strong developer incentives through grants (LEIA) and hackathons. - High liquidity/TVL via initiatives like Linea Surge and Linea Voyage. - highest number of dApps deployed	- zkEVM Type-3: Limited dApp and tool compatibility. - No governance or DAO structure yet.
Polygon zkEVM	- Strong developer ecosystem due to its brand and network of partnerships. - zkEVM Type-2: Enhanced compatibility with Ethereum-native tools. - Comprehensive developer support through Developer Relations Team and Polygon Solutions Network Provider - Established governance framework.	- Lower liquidity and TVL compared to competitors. - Mainnet still in beta, carrying potential risks. - High operational costs and gas fees.
ZkSync Era	- Highest TVL among zkRollups. - Native account abstraction for smart wallet development. - zkPorter support for enhanced scalability and cost savings. - Allows building custom rollups through Hyperchains. - Established on-chain governance through ZkNation.	- zkEVM Type-4: Requires a custom compiler for smart contracts. - Limited compatibility with Ethereum tools. - Data availability concerns for zkPorter.

Table 13.1: Advantages and Disadvantages of zkEVM's Platforms.

14 Fazit

zkRollups are still in the early stages of rapid development, and as such, this master thesis will likely become outdated within a year and will need regular updates. Since the time of writing (August to December 2024), several significant changes have occurred. For instance, in November 2024, Scroll announced its Governance Framework, something that was previously missing. Additionally, Scroll is expected to implement a Multi-Prover Design, similar to Taiko's approach. Speaking of Taiko, its Total Value Locked (TVL) has tripled in just two months, from September to December.

Looking ahead, it is expected that all zkRollups will eventually achieve full decentralization and become Type-1 zkEVMs. This will be driven by the development of more efficient algorithms and cryptographic techniques that make zero-knowledge proofs faster and more practical. As these advancements occur, the differences between zkEVMs will gradually diminish. However, in the cryptocurrency space, the focus is on collaboration rather than a "winner-takes-all" mentality. In the future, it is expected that different chains will increasingly communicate and interoperate, resulting in a more interconnected ecosystem, where everyone benefits from the strengths of each chain.

Bibliography

[1] *Genesis block - Bitcoin Wiki*, Mar. 2024. [Online]. Available: https://en.bitcoin.it/wiki/Genesis_block (visited on 12/21/2024).

[2] V. Buterin, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.", en, 2014.

[3] *Ethereum price today, ETH to USD live price, marketcap and chart*, en. [Online]. Available: https://coinmarketcap.com/currencies/ethereum/ (visited on 12/21/2024).

[4] *Chainlink: The Standard for Onchain Finance*, en. [Online]. Available: https://chain.link/ (visited on 12/21/2024).

[5] *Aave*, en. [Online]. Available: https://aave.com/ (visited on 12/21/2024).

[6] *Uniswap Interface*, en-US. [Online]. Available: https://app.uniswap.org/ (visited on 12/21/2024).

[7] Shin, Mike, *zkEVM Explained: The Ultimate Ethereum Scaling Solution*, en, May 2023. [Online]. Available: https://blog.thirdweb.com/zkevm/ (visited on 09/11/2024).

[8] Buterin, Vitalik, *An Incomplete Guide to Rollups*, Jan. 2021. [Online]. Available: https://vitalik.eth.limo/general/2021/01/05/rollup.html (visited on 09/12/2024).

[9] Loopring, *Loopring - zkRollup Layer 2 for Trading and Payment*, Jan. 2024. [Online]. Available: https://loopring.org/#/ (visited on 01/17/2024).

[10] Vaishnavi Jonna, *High-Level Programming Languages: Bridging the Gap Between Human and Machine - ellow.io*, en-US, Section: Blog, May 2024. [Online]. Available: https://ellow.io/high-level-programming-languages/ (visited on 12/16/2024).

[11] admin, *Why is Python Considered a High-Level Programming Language?*, en-US, Feb. 2024. [Online]. Available: https://dscsdelhi.com/why-is-python-considered-a-high-level-programming-language/ (visited on 12/16/2024).

[12] Microsoft, *What Is a Virtual Machine and How Does It Work | Microsoft Azure*, en-US. [Online]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-virtual-machine (visited on 09/11/2024).

[13] Kakarot Labs, *The concept of ZK-EVM | Kakarot ZK-EVM*, en. [Online]. Available: https://docs.kakarot.org/starknet/architecture/understanding-zkevm/ (visited on 12/16/2024).

[14] Github: Otto-AA, *Ethereum Virtual Machine (EVM)*, en, Jun. 2024. [Online]. Available: https://ethereum.org/en/developers/docs/evm/ (visited on 12/16/2024).

[15] W. Wslyvh, *Gas and fees*, en, Aug. 2024. [Online]. Available: https://ethereum.org/en/developers/docs/gas/ (visited on 09/10/2024).

[16] Dr Gavin Wood, "ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER", en, Sep. 2024.

[17] Alchemy, *How do Ethereum transactions work?*, en. [Online]. Available: https://docs.alchemy.com/docs/how-ethereum-transactions-work (visited on 09/11/2024).

[18] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems", en, in *Proceedings of the seventeenth annual ACM symposium on Theory of computing - STOC '85*, Providence, Rhode Island, United States: ACM Press, 1985, pp. 291–304, ISBN: 978-0-89791-151-1. DOI: 10.1145/22145.22178. [Online]. Available: http://portal.acm.org/citation.cfm?doid=22145.22178 (visited on 01/17/2024).

[19] A. Berentsen, J. Lenzi, and R. Nyffenegger, "An Introduction to Zero-Knowledge Proofs in Blockchains and Economics", en, *Review*, vol. 105, no. 4, 2023. DOI: 10.20955/r.105.280-94. [Online]. Available: https://research.stlouisfed.org/publications/review/2023/05/12/an-introduction-to-zero-knowledge-proofs-in-blockchains-and-economics (visited on 01/17/2024).

[20] Chainlink, *Zero-Knowledge Proof (ZKP) — Explained | Chainlink*, en, Sep. 2023. [Online]. Available: https://chain.link/education/zero-knowledge-proof-zkp (visited on 01/17/2024).

[21] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again", in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12, New York, NY, USA: Association for Computing Machinery, Jan. 2012, pp. 326–349, ISBN: 978-1-4503-1115-1. DOI: 10.1145/2090236.2090263. [Online]. Available: https://doi.org/10.1145/2090236.2090263 (visited on 01/17/2024).

[22] Chainlink, *Zk-SNARK vs zkSTARK - Explained Simple | Chainlink*, en, Nov. 2023. [Online]. Available: https://chain.link/education-hub/zk-snarks-vs-zk-starks (visited on 01/17/2024).

[23] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity", en, Aug. 2018.

[24] B. WhiteHat, J. Baylina, and M. Belles, "Baby Jubjub Elliptic Curve", en,

[25] H. Zeilberger, *Simple Explanations of Arithmetic Circuits and Zero-Knowledge Proofs*, en, Dec. 2019. [Online]. Available: https://medium.com/web3studio/simple-explanations-of-arithmetic-circuits-and-zero-knowledge-proofs-806e59a79785 (visited on 01/17/2024).

[26] Scroll, *zkEVM - Scroll*, en, Sep. 2021. [Online]. Available: https://scroll.io/blog/zkevm (visited on 09/16/2024).

[27] D. Boneh, J. Drake, B. Fisch, and A. Gabizon, *Efficient polynomial commitment schemes for multiple points and polynomials*, Publication info: Preprint. MINOR revision., 2020. [Online]. Available: https://eprint.iacr.org/2020/081 (visited on 12/16/2024).

[28] Brendan Farmer, *Recursive proofs on Mir | Mir | Scaling Ethereum*, Jan. 2021. [Online]. Available: https://mirprotocol.org/blog/Recursive-proofs-on-Mir (visited on 12/16/2024).

[29] A. Gabizon and Z. J. Williamson, *Plookup: A simplified polynomial protocol for lookup tables*, Publication info: Preprint. MINOR revision., 2020. [Online]. Available: https://eprint.iacr.org/2020/315 (visited on 12/16/2024).

[30] Y. Zhang *et al.*, "PipeZK: Accelerating Zero-Knowledge Proof with a Pipelined Architecture", en, in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, Valencia, Spain: IEEE, Jun. 2021, pp. 416–428, ISBN: 978-1-66543-333-4. DOI: 10.1109/ISCA52012.2021.00040. [Online]. Available: https://ieeexplore.ieee.org/document/9499783/ (visited on 12/16/2024).

[31] Buterin, Vitalik, *The different types of ZK-EVMs*, Aug. 2022. [Online]. Available: https://vitalik.eth.limo/general/2022/08/04/zkevm.html (visited on 09/17/2024).

[32] Vishal Chawla, *Ethereum Layer 2 Taiko goes live on mainnet*, en, Section: Layer 2s and Scaling. [Online]. Available: https://www.theblock.co/post/296831/ethereum-l2-taiko-mainnet (visited on 12/16/2024).

[33] Scroll, *A Letter from Scroll: Mainnet is Here! - Scroll*, en, Oct. 2023. [Online]. Available: https://scroll.io/blog/founder-letter (visited on 12/16/2024).

[34] Consensys, *Consensys Launches Linea Mainnet, Unlocking a New Level of User Experience and Scalability for Ethereum*, en, Aug. 2023. [Online]. Available: https://consensys.io/blog/consensys-launches-linea-mainnet-unlocking-a-new-level-of-user-experience (visited on 12/16/2024).

[35] S. Sriram, *Polygon zkEVM Mainnet Beta Gets Launch Date*, en, Feb. 2023. [Online]. Available: https://unchainedcrypto.com/polygon-zkevm-mainnet-beta-gets-launch-date/ (visited on 12/16/2024).

[36] 0xKira, *Everything You Need to Know About ZkSync Era | CoinMarketCap*, en, 2023. [Online]. Available: https://coinmarketcap.com/academy/article/everything-you-need-to-know-about-zksync-era (visited on 10/31/2024).

[37] Unchained Crypto, *What Are Sequencers in Layer 2 Protocols Such as Optimism, Arbitrum, and Base?*, en, May 2024. [Online]. Available: https://unchainedcrypto.com/what-are-sequencers-in-layer-2-protocols-such-as-optimism-arbitrum-and-base/ (visited on 12/18/2024).

[38] Scroll, *An overview of Scroll's architecture*, Aug. 2022. [Online]. Available: https://scroll.mirror.xyz/nDAbJbSIJdQIWqp9kn8J0MVS4s6pYBwHmK7keidQs-k (visited on 12/19/2024).

[39] Scroll, *zkEVM Overview*, en, 2024. [Online]. Available: https://docs.scroll.io/en/technology/zkevm/zkevm-overview/ (visited on 12/19/2024).

[40] Polygon Labs, *zkEVM and execution traces - Polygon Knowledge Layer*, 2024. [Online]. Available: https://docs.polygon.technology/zkEVM/architecture/proving-system/exec-trace-and-zkevm/ (visited on 12/19/2024).

[41] Justin Drake, *Based rollups—superpowers from L1 sequencing - Layer 2*, en, Section: Layer 2, Mar. 2023. [Online]. Available: https://ethresear.ch/t/based-rollups-superpowers-from-l1-sequencing/15016 (visited on 12/19/2024).

[42] Shen, Haichen, *Scaling Security: Multi-Prover Implementation on Scroll - Scroll*, en, Jan. 2024. [Online]. Available: https://scroll.io/blog/scaling-security (visited on 10/05/2024).

[43] L. A. Brecht, *Why multi-prover matters. SGX as a possible solution.* Nov. 2023. [Online]. Available: https://taiko.mirror.xyz/Kx1Mp4WJjd83K1KDEwp1pM7xi9QmpSahxJg3S_N7NE4 (visited on 10/05/2024).

[44] taikoxyz, *Taikoxyz/raiko*, original-date: 2023-09-07T15:35:46Z, Oct. 2024. [Online]. Available: https://github.com/taikoxyz/raiko (visited on 10/05/2024).

[45] Taiko [@taikoxyz], *Gm Taikonauts! Join us this Wednesday for a discussion about multi-proofs with @VitalikButerin, @protolambda, and @Brechtpd. We'll chat about what multi-proofs are, what designs are the most elegant ones, and other similar topics.* en, Tweet, Dec. 2023. [Online]. Available: https://x.com/taikoxyz/status/1734226720969146649 (visited on 10/05/2024).

[46] Saniya, *Zeroing into zkVMs*, Apr. 2024. [Online]. Available: https://taiko.mirror.xyz/e_5GeGGFJIrOxqvXOfzY6HmWcRjCjRyG0NQF1zbNpNQ (visited on 10/05/2024).

[47] M. Ikola, *New Paradigm in Ethereum L2 Scaling: Multi-proving and ZK-VMs*, en-US, Dec. 2023. [Online]. Available: https://www.mikkoikola.com/blog/2023/12/11/new-paradigm-in-ethereum-l2-scaling-multi-proving-and-zk-vms (visited on 10/06/2024).

[48] umede, *Based Contestable Rollup (BCR): A configurable, multi-proof roll...* Dec. 2023. [Online]. Available: https://taiko.mirror.xyz/Z4I5ZhreGkyfdaL5I9P0Rj0DNX4zaWFmcws-0CVMJ2A (visited on 10/08/2024).

[49] Taiko Labs, *Network configuration*, en. [Online]. Available: https://docs.taiko.xyz/network-reference/network-configuration/ (visited on 10/08/2024).

[50] D. Brecht, *Booster rollups - scaling L1 directly - Layer 2*, en, Section: Layer 2, Oct. 2023. [Online]. Available: https://ethresear.ch/t/booster-rollups-scaling-l1-directly/17125 (visited on 10/09/2024).

[51] Brecht Devos [@Brechtpd], *L1CALL is great. It allows the execution of L1 smart contracts directly from L2 (without persisting any of the state changes of course). L1DELEGATECALL could also be helpful to help connect L1 and L2s and make it look more like a single parallelized execution environment.* en, Tweet, Aug. 2023. [Online]. Available: https://x.com/Brechtpd/status/1688533026156744704 (visited on 10/09/2024).

[52] umede, *Based Booster Rollup (BBR): A new major milestone in Taiko's roa...* Nov. 2023. [Online]. Available: https://taiko.mirror.xyz/anPjF35Mrc_xzYgOTbUmfjr_MlhE3L8ZBZIxqmz9GZ8 (visited on 10/09/2024).

[53] Taiko Labs, *Inception layers*, en, 2024. [Online]. Available: https://docs.taiko.xyz/core-concepts/inception-layers/ (visited on 12/19/2024).

[54] Linea, *Architecture | Linea*, en, Dec. 2024. [Online]. Available: https://docs.linea.build/get-started/concepts/architecture (visited on 12/19/2024).

[55] Decentralized Dog, *Everything You Need To Know About Linea Network and its DeFi Voyage | CoinMarketCap*, en, 2023. [Online]. Available: https://coinmarketcap.com/academy/article/linea-network-defi-voyage (visited on 12/19/2024).

[56] flagship, *Exploring Linea: The Layer 2 Solution from the Makers of MetaMask*, en, Aug. 2023. [Online]. Available: https://flagship.fyi/outposts/blockchains/linea-crypto-layer-2-solution-metamask-consensys-ethereum-zk-rollup/ (visited on 12/19/2024).

[57] Linea, *Sequencer | Linea*, en, Dec. 2024. [Online]. Available: https://docs.linea.build/get-started/concepts/sequencer (visited on 12/19/2024).

[58] L. Prover, *Linea Prover Documentation*, Publication info: Preprint., 2022. [Online]. Available: https://eprint.iacr.org/2022/1633 (visited on 12/19/2024).

[59] Enrique Menendez, *The Linea Prover Explained*, Aug. 2024. [Online]. Available: https://linea.mirror.xyz/h_Y_XazAtqDHODCqFMDs3jY2jn4B-Un8fepRP1xStBg# (visited on 12/19/2024).

[60] Emily Lin, *The Linea Prover for a Very Smart High Schooler — Linea*, Jun. 2023. [Online]. Available: https://linea.mirror.xyz/B3b1lUK8--UKZ_Qehk7SfOyvdcGbcuoyvNsSukHgOY8 (visited on 12/19/2024).

[61] Polygon Labs, *zkEVM - Polygon Knowledge Layer*. [Online]. Available: https://docs.polygon.technology/zkEVM/ (visited on 10/29/2024).

[62] Polygon Labs, *zkProver - Polygon Knowledge Layer*, 2024. [Online]. Available: https://docs.polygon.technology/zkEVM/architecture/zkprover/#circom-library (visited on 12/19/2024).

[63] Polygon Labs, *CIRCOM - Polygon Knowledge Layer*, 2024. [Online]. Available: https://docs.polygon.technology/zkEVM/concepts/circom-intro-brief/ (visited on 12/19/2024).

[64] Polygon Labs, *Aggregated Blockchains: A New Thesis*, en, Jan. 2024. [Online]. Available: https://polygon.technology/blog/aggregated-blockchains-a-new-thesis (visited on 11/28/2024).

[65] James Bachini, *Aggregated Blockchains and Polygon's Agglayer*, en-US, Section: Analysis, Feb. 2024. [Online]. Available: https://jamesbachini.com/aggregated-blockchains-and-polygons-agglayer/ (visited on 11/28/2024).

[66] Antier Solutions, *Polygon's AggLayer: Why This New Rollup Thesis is a Game Changer in 2024?*, Feb. 2024. [Online]. Available: https://www.antiersolutions.com/agglayer-why-polygons-scalability-solution-is-a-game-changer-in-2024-beyond/ (visited on 11/28/2024).

[67] ZkSync, *Sequencer / Server*, en, Jun. 2024. [Online]. Available: https://docs.zksync.io (visited on 11/05/2024).

[68] pseudotheos, *Exploring Zero Knowledge: zkSync and the zkEVM*, Feb. 2022. [Online]. Available: https://pseudotheos.mirror.xyz/JF8_qjziArLgVPhC0ADl_Adz9PRRSEURb7vGLR9_9AE (visited on 11/05/2024).

[69] zkSync, *Boojum Upgrade: zkSync Era's New High-performance Proof System f…* Jul. 2023. [Online]. Available: https://zksync.mirror.xyz/HJ2Pj45EJkRdt5Pau-ZXwkV2ctPx8qFL19STM5jdYhc (visited on 11/06/2024).

[70] K. B. Academy, *Native Account Abstraction in ZKSync: Transforming the Blockchain Experience for Users*, en, Jul. 2024. [Online]. Available: https://kbaiiitmk.medium.com/native-account-abstraction-in-zksync-transforming-the-blockchain-experience-for-users-4a0c88181199 (visited on 10/31/2024).

[71] @wackerow, *Data availability*, en, May 2024. [Online]. Available: https://ethereum.org/en/developers/docs/data-availability/ (visited on 11/04/2024).

[72] M. Labs, *zkPorter: A breakthrough in L2 scaling*, en, Jan. 2022. [Online]. Available: https://blog.matter-labs.io/zkporter-a-breakthrough-in-l2-scaling-ed5e48842fbf (visited on 11/04/2024).

[73] Dabillstevens, *zkSync - a Simple Overview of a Complex Technology*, en, Feb. 2023. [Online]. Available: https://medium.com/@dabillstevens/a-simple-overview-of-zksyncs-complex-technology-aa31e66500cd (visited on 11/04/2024).

[74] ZkSync, *Compiler Toolchain Overview*, en, Aug. 2024. [Online]. Available: https://docs.zksync.io (visited on 11/05/2024).

[75] S. Sharma, *What are zkSync Hyperchains & how they bring hyper scalability to L3 dApps?*, en-US, Nov. 2023. [Online]. Available: https://www.zeeve.io/blog/what-are-zksync-hyperchains-a-guide/ (visited on 11/06/2024).

[76] Kanishk Tagade, *Smart Contract Audits: Importance & How Does It Work*, en-US, Section: Security Audit, Jun. 2024. [Online]. Available: https://www.getastra.com/blog/security-audit/smart-contract-security/ (visited on 12/19/2024).

[77] Scroll, *Audits & Bug Bounty Program*, en, 2024. [Online]. Available: https://docs.scroll.io/en/technology/security/audits-and-bug-bounty/ (visited on 12/19/2024).

[78] Polygon Labs, *Security First: Polygon zkEVM Mainnet Beta*, en, 2024. [Online]. Available: https://polygon.technology/polygon-zkevm/security-first (visited on 12/19/2024).

[79] ZkSync, *Audits*, en, 2024. [Online]. Available: https://docs.zksync.io (visited on 12/19/2024).

[80] QuilAudits, *Taiko Audit Report | QuillAudits*, en, 2024. [Online]. Available: https://www.quillaudits.com/leaderboard/taiko (visited on 12/19/2024).

[81] Openzeppelin, *Linea V2 Audit*, en, 2024. [Online]. Available: https://blog.openzeppelin.com/linea-v2-audit (visited on 12/19/2024).

[82] *Immunefi*. [Online]. Available: https://immunefi.com/ (visited on 12/19/2024).

[83] Vitalik Buterin, *Proposed milestones for rollups taking off training wheels - Magicians / Primordial Soup*, en, Section: Magicians, Nov. 2022. [Online]. Available: https://ethereum-magicians.org/t/proposed-milestones-for-rollups-taking-off-training-wheels/11571 (visited on 12/19/2024).

[84] *L2BEAT - The state of the layer two ecosystem*, en-us. [Online]. Available: https://l2beat.com/scaling/summary (visited on 12/19/2024).

[85] Chainlink, *12+ Key Metrics for the Rapidly Expanding Web3 Ecosystem | Chainlink*, en-US, Oct. 2022. [Online]. Available: https://blog.chain.link/web3-metrics/ (visited on 12/19/2024).

[86] *Rollup.wtf*. [Online]. Available: https://rollup.wtf/ (visited on 12/19/2024).

[87] *Growthepie*, en. [Online]. Available: https://www.growthepie.xyz (visited on 12/19/2024).

[88] *Layer 2: ZK Rollups Archives*, en. [Online]. Available: https://www.theblock.co/data/scaling-solutions/zk-rollups (visited on 12/19/2024).

[89] *V0RTEx 01 | A Scroll Hackathon | Hackathon | DoraHacks*. [Online]. Available: https://dorahacks.io/hackathon/v0rtex-01 (visited on 12/19/2024).

[90] Taiko, *Grant Program – Taiko*, en. [Online]. Available: https://taiko.xyz/grant-program (visited on 12/19/2024).

[91] Linea, *Linea Ecosystem Investment Alliance Announces Seven Inaugural Investments*, Jun. 2024. [Online]. Available: https://linea.mirror.xyz/N3rGX2njnWIrNhjRxsGDYnayDQvF1juzbMKbCLE12Tw (visited on 12/19/2024).

[92] Polygon Labs, *1 Billion Unlocked Over Decade in Community Grants Program for Polygon Builders*, en, Jun. 2024. [Online]. Available: https://polygon.technology/blog/1-billion-unlocked-over-decade-in-community-grants-program-for-polygon-builders (visited on 10/29/2024).

[93] Polygon Labs, *Meet the Dev Rel*, en. [Online]. Available: https://polygon.technology/community/meet-the-devrel#team (visited on 10/29/2024).

[94] Polygon Labs, *With Solution Provider Network (SPN), Developers Can Now Find and Connect with Any Solution Provider on Polygon Protocols*, en, Sep. 2024. [Online]. Available: https://polygon.technology/blog/with-solution-provider-network-spn-developers-can-now-find-and-connect-with-any-solution-provider-on-the-polygon-protocols (visited on 10/29/2024).

[95] *Scroll - Canvas And Badges*, en. [Online]. Available: https://scroll.io/canvas-and-badges (visited on 12/19/2024).

[96] *Scroll - Scroll Sessions*, en. [Online]. Available: https://scroll.io/sessions?utm_source=BitgetWallet (visited on 12/19/2024).

[97] Taiko Labs, *Trailblazers: Season 2 Announcement*, Sep. 2024. [Online]. Available: https://taiko.mirror.xyz/IcZCVH0501eFNKmRlro3SPd4onneBH2RutoxwBneyj0 (visited on 12/19/2024).

[98] *Linea Voyage: The Surge | Linea Help Center*, en. [Online]. Available: https://support.linea.build/linea-voyage/linea-surge (visited on 12/19/2024).

[99] *ZKsync Ignite*, en. [Online]. Available: https://zksyncignite.xyz/ (visited on 12/19/2024).

[100] *Welcome*, en-us. [Online]. Available: https://scroll-governance-documentation.vercel.app/gov-docs/content/overview/ (visited on 12/19/2024).

[101] *Aragon/taiko-contracts*. [Online]. Available: https://github.com/aragon/taiko-contracts (visited on 12/19/2024).

[102] Linea, *Introducing the Linea Association, a Major Step Toward Network D... — Linea*, Nov. 2024. [Online]. Available: https://linea.mirror.xyz/0sExLyJh33ktk6x2kg3l2z0xDVUDt3ilJzsA_Kx0sp0 (visited on 12/19/2024).

[103] ZkSync, *Welcome to ZKsync Onchain Governance*, en, Sep. 2024. [Online]. Available: https://blog.zknation.io/zksync-governance-system/ (visited on 12/19/2024).

[104] *Governance Pillars*, en. [Online]. Available: https://polygon.technology/governance-pillars (visited on 12/19/2024).

[105] *Remix - Ethereum IDE*. [Online]. Available: https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa7a.js (visited on 12/19/2024).

[106] *Hardhat | Ethereum development environment for professionals by Nomic Foundation*, en. [Online]. Available: https://hardhat.org (visited on 12/21/2024).