Master's Thesis

Design and Implementation of a Prototype for an Optimistic Rollup System

Dakila Madushanka Serasinghe

Examiners: Prof. Dr. Peter Thiemann
Prof. Dr. Andreas Podelski

Adviser: Dr. Thi Thu Ha Doan

Albert-Ludwigs-University Freiburg
Faculty of Engineering
Department of Computer Science
Chair for Programming Languages

June 22nd, 2023

Writing period

22. 12. 2022 – 22. 06. 2023

Examiners

Prof. Dr. Peter Thiemann

Prof. Dr. Andreas Podelski

Adviser

Dr. Thi Thu Ha Doan

Declaration

I hereby declare, that I am the sole author and composer of my Thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Freiburg, 22.06.2023

Place, Date

Signature

Abstract

As the popularity of blockchain technology grows, the demand for efficient and scalable solutions becomes more apparent. Layer-1 blockchains frequently encounter difficulties in managing a large number of transactions, leading to higher fees and longer processing times. These challenges have hindered the ability of blockchains to deliver efficient user experiences and achieve widespread adoption.

Various solutions have emerged to address scalability issues in Layer-1 blockchains, with Rollups being a prominent contender. Rollups provide a Layer-2 scaling solution that complements blockchain infrastructure by introducing additional layers or protocols. They aim to enhance transaction throughput and lower costs by leveraging off-chain computations. This approach lightens the load on the parent chain, improving scalability while maintaining decentralization and security.

Rollups come in various flavors, including Optimistic Rollups and Zero-Knowledge Rollups. Optimistic Rollups assume transaction validity until proven otherwise and employ fraud proofs for protocol integrity. Zero-Knowledge Rollups utilize advanced cryptographic techniques for enhanced security and faster finality.

The objective of this thesis is to design and implement an Optimistic Rollup prototype for a simple payment system. We extensively discuss various aspects of Optimistic Rollups, including current architectural trends and approaches in the state-of-the-art. Additionally, we analyze Layer-2 economics, identifying critical cost factors and associated challenges in the Rollup protocol.

Having presented our Rollup architecture, we delve deeper into the exploration of data compression techniques utilized in Optimistic Rollups. We perform comprehensive experiments to evaluate the efficacy of our compression mechanism.

Subsequently, we conduct an extensive analysis, focusing on the scalability and usability aspects of our implementation. This involves comparing key system performance metrics with existing works. Further, we identify the limitations inherent in our Rollup and propose potential avenues for future research and development.

Finally, the thesis concludes by summarizing the research findings, emphasizing the contributions and insights gained.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

The advent of blockchain technology has revolutionized various industries by offering decentralized, transparent, and secure trustless financial systems. However despite the continued popularity and adoption of blockchains, one critical challenge, "scalability" has been blocking the way reducing its usability on many applications. The Layer-1 blockchains, with its current architectures, often encounter limitations when it comes to handling a high volume of transactions efficiently. These constraints pose significant obstacles in achieving widespread mass adoption of blockchain technology.

One of the key issues plaguing Layer-1 blockchains is transaction cost. As the number of transactions increases, the associated fees and processing time also increase correspondingly. In 2021, Ethereum transaction cost rose astronomically to 200 United States Dollars worth of Ether[1]. This inflating increase in transaction fees can make micro-transactions economically infeasible and discourages users from engaging with blockchain applications.

Furthermore, the time required to confirm transactions on the blockchain can be significant, which hampers real-time transactional capabilities. Currently Bitcoin process up to 7 transactions per second while Ethereum runs approximately 12 transactions per second. But compared to centralized payment systems, for example Visa with 65,000 transactions per second [2], current blockchains fall behind in competing with traditional financial institutes.

To support widespread adoption of blockchains maintaining a high throughput and a competitive transaction fee, blockchain architectures should be easily scalable in various dimensions. They should be cheap enough to facilitate commercial transactions, efficient with minimum latencies and robust without network congestion.

However, in context of achieving these ideal aspects, the notion of scalability trilemma [3], suggests that trade-offs are inevitable between decentralization, security and scalability in a blockchain. Reaching these properties simultaneously within a cohesive architecture has emerged as the primary challenge for the blockchain community.

To address these scalability challenges, there have been various approaches explored

by both academia and industry. These solutions are typically classified into two domains: Layer-1 and Layer-2. Layer-1 solutions primarily concentrate on architectural modifications of the underlying blockchain itself. But they have various limitations such as hard forks and network congestion, which render them as impractical as a feasible scalability solutions.

In contrast, Layer-2 scaling solutions have emerged as a promising avenue for enhancing the performance and efficiency of Layer-1 blockchains. Layer-2 solutions aim to complement existing blockchain infrastructure by introducing additional layers or protocols that alleviate the burden on the base layer and enable higher transaction throughput. By leveraging off-chain transactions, Layer-2 scaling solutions seek to mitigate the transaction cost and scalability limitations faced by Layer-1 blockchains. These solutions have the potential to unlock the scalability needed for blockchain networks to support mass adoption and revolutionize industries on a global scale.

One of the Layer-2 scaling solutions, known as "Rollups," has emerged as a potential ultimate solution for the blockchain scalability issue. With many technological and architectural breakthroughs, Rollups attempt to mitigate the scalability trilemma by offering a middle ground without compromising on decentralization, security, and scalability.

There are two primary realizations of Rollups that have been proposed in blockchain, Optimistic Rollups and Zero-Knowledge Rollups. In this thesis, we design and implement a prototype for an Optimistic Rollups system in Ethereum, while exploring latest architectures, advantages, and limitations associated with these solutions.

This work focuses on a simple payment mechanism considering the difficulty of a generalized construction with arbitrary smart contract execution. By implementing a prototype, we aim to gain a deeper understanding of the architectural and protocol considerations, as well as the trade-offs involved in the design of Rollups.

1.1 Thesis Organization

The primary objective of this work is to design and implement a prototype for an Optimistic Rollups system. While achieving this, we structure this thesis into several chapters.

- In the **Chapter 1** we give a brief introduction to the problem and our objective of this work.
- The **Chapter 2** presents a comprehensive analysis on the scalability issue

of blockchains while introducing Layer-2 solutions in subsequent subsections. Towards the end of the chapter, it demonstrates the viability of Rollups as a solution for addressing the scalability issue in blockchains.

- The State-of-the-Art in Rollups is discussed in **Chapter 3**, showcasing the latest architectural trends in both Optimistic and Zero-Knowledge Rollups. A further analysis is conducted on Layer-2 economics, presenting the latest Rollup cost structure and challenges in Optimistic Rollup systems.

- The **Chapter 4** presents the implementation of our Optimistic Rollup prototype, with an in-depth explanation on the design process and architecture.

- **Chapter 5** is dedicated to conducting an in-depth analysis of our Rollup system. It covers experiments on data compression of this work and analyses the scalability and usability of the system by comparing with existing works. It further identifies the limitation of our work and discusses future directions.

- Finally we conclude the Thesis in **Chapter 6**, encapsulating the key findings and insight derived from this work.

2 Background

2.1 Scalability Trilemma

The fundamental properties of an ideal blockchain are decentralization, security and scalability. Today every Layer-1 blockchain is trying to reach all these properties without sacrificing one aspect for the sake of the others. But the "scalability Trillema"[3] states, it is impossible to reach all these aspects simultaneously, without making compromises on at least one of them.

- **Decentralization** Decentralization refers to the distribution of authority and control across multiple participants in a network. It guarantees that no central party can govern the execution of transactions and the chain is progressed trustlessly without any censorship.
- **Security** Security resembles the chain's capacity to withstand an attack by malicious network participants safeguarding the integrity and immutability of the data.
- **Scalability** Scalability is the capacity of handling a high throughput in the network increasing the usability.

Currently we shall observe three categories of implementation approaches, which are capitalizing on two properties out of three.

Traditional blockchain consensus mechanisms such as Proof of Work (PoW) are based on active participation of every node in the network. This makes blockchains, decentralized and secure making them immune to majority attacks. But they move away from scalability due to the requirement of re-execution. With the adoption of Proof of Stake (PoS) mechanism with bonds in Ethereum, are more scalable compared to PoW at the cost of decentralization. In the latest Ethereum version (ETH2), a validator has to stake 32ETH to participate in the protocol. This high barrier on bonding leads to concerns on centralization in the network.

In contrast to this, a more scalable and secure blockchain can be achieved using other consensus mechanisms such as Delegated Proof of Stake (DPoS) and Proof of

Authority (PoA). As the participants are less in these consensus systems, it shall reach high throughput due to less network latencies for consensus. But they are not decentralized having incurred trust on a limited set of operators.

In the third category, a decentralized and scalable blockchain shall be realized using a multi-chain structure having multiple blockchains connected through bridges. but the security of the whole ecosystem is at stake, if one of the bridges get compromised.

In order to address the challenges associated with scalability in blockchain networks, different solutions have been proposed through researches from both academia and industry. We shall divide these solutions mainly into two categories, Layer-1 solutions and Layer-2 solutions. Layer-1 solutions involve in changing the fundamental properties of traditional blockchains such as modifying block data[4] and size[5], introducing alternative consensus mechanism[6, 7], sharding the blockchain[8] and Directed Acylic Graphs (DAG)[9, 10, 11]. There are several studies which have gone through the limitations and challenges on these techniques [12, 13].

It is seen, change of consensus mechanisms does not scale the network without introducing trust assumptions. Again alteration of blockchain properties, leads to a hard fork making the blockchain backward incompatible. And a bigger block size leads to higher propagation delays and operational costs.

Due to these limitations of Layer-1 solutions for scaling, a separate avenue of research emerged known as Layer-2 blockchain protocols. These protocols aim to address scalability challenges by introducing alternative approaches that operate alongside the base Layer-1 blockchain.

These solutions are called Layer-2 as they built upon the traditional blockchains, which we have already referred to as Layer-1. The Layer-2 solutions identify the execution of transactions as the bottleneck to scaling and focus on bring it off-chain. Different Layer-2 solutions have been introduced in the literature which comes with their own assumptions and limitations.

In the subsequent sections, we delve into various Layer-2 solutions, examining their architectures, benefits, and constraints.

2.2 State Channels

State Channels[14] are one of the earliest Layer-2 concepts proposed in the Blockchain community[15] to tackle scalability and higher transaction costs in blockchains. They are peer-to-peer protocols which allow two or more parties to interact off-chain while assuming the same security level as of a Layer-1 blockchain. Channels facilitate

users to initiate arbitrary number of off-chain transfers at zero transaction cost as it removes the requirement of going through the mainnet for each transaction between them. Further it increases the privacy of user and improves the throughput of the underlying blockchain as it minimizes the Layer-1 foot-print.

2.2.1 Architecture

Channels reduce the interaction with Layer-1 to the minimum possible. In an optimistic scenario where all parties are honest, connection to the mainnet only happens at two occasions, when opening the channel and closing the channel. But in practice, when we assume no trust between users, there is another lead to the mainnet to resolve the latest state of the channel in case of a dispute.

Accordingly we can deduce the architecture of a channel into several phases.

- **Channel Opening** Channel is opened when all participants fund the channel by committing assets in the underlying blockchain. These funds will function as a stake for the entire lifetime of the channel, ensuring the honesty of user. Participants will further agree on the initial state of the channel by signing the genesis state. This will act as the starting point of the channel.

- **Channel Update** In the second phase, users can make peer-to-peer off-chain transfers between them. This is where the core advantages of state channels are realized. Here participants make off-chain transfers with cryptographically signed messages with zero costs compared to on-chain. Each of these messages suggest a new state on the state channel. All participants will go through received messages by peers and sign should they agree on the new state.

  As long as all participants are acting along the protocol, channel update phase will last until channel closing phase.

- **Channel Dispute** All paticipants in a channel needs to be in full consensus on the latest state of the channel. If one party is not agreeing on the latest state or is remaining silent without responding for state approvals, a dispute arises in the protocol. A dispute resolution mechanism based on a challenge-response protocol, shall identify the latest valid state accepted by all parties. Once the latest state is identified, channel can either return to the update phase where users continue to use the channel or progress to channel closing phase.

- **Channel Closing** The final phase of a channel is moving assets back into the Layer-1. There could be two cases which lead to the closing phase. In the

optimistic case all parties act honestly and synchronously with state updates and agree on the final state of the channel. With the on-chain verification on the final state, channel get closed distributing locked funds according to the final outcome. In a pessimistic case channel will lead to the aforementioned dispute resolution protocol and be resolved after a certain challenge period.

2.2.2 Types of State Channels

Payment Channels

Payment Channels [16] were among the initial ideas of off-chain blockchain transactions while focusing only on simple payments. It enables arbitrarily many off-chain transfers between peers with only two on-chain transactions when opening and closing the channel.

A payment channel is a two party agreement where they initialize the contract on-chain mutually agreeing on an expiration time. Then they continue to communicate on the latest state off-chain until the expiration with a valid final state. Payment channels have been applied in both Bitcoin Lightning network [17] and in Ethereum Raiden [18].

Generalized Channels

Generalized channels [20, 21] or generalized state channels extend payment channels improving off-chain execution from simple payments to arbitrary state transitions. It enables Off-chain execution of any script that is supported by the underlying blockchain.

While generalized state channels improve scalability of Layer-1 enabling same level state executions, it incurs implications on the security. Since channels run on a set of rules imposed by users, it does not guarantee the same consensus based security as of a Layer-1 chain.

Channel Networks

Channel Networks [22] create a network of "ledger channels" further eliminating the need of on-chain transactions. Ledger channels are the foundation channels in the network which get directly funded through the Layer-1 chains. There are two main flavours of channel networks [23].

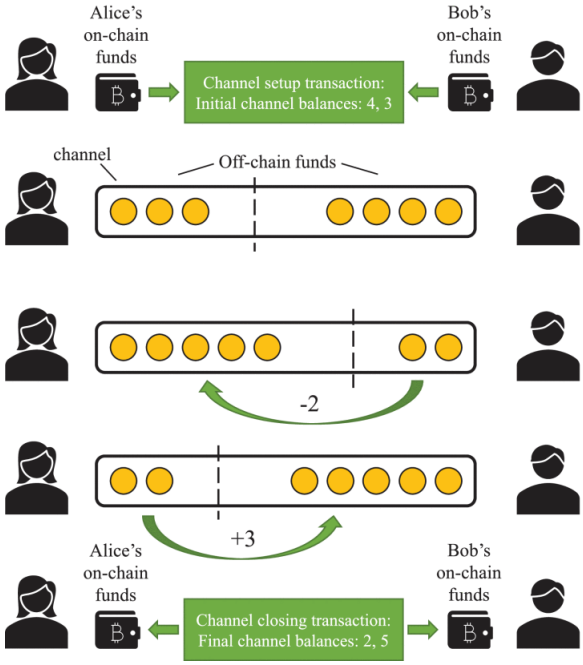


Figure 1: The lifecycle of a payment channel[19]. Initially, Alice and Bob deposit some of their on-chain funds to a channel-establishing on-chain transaction. Once the transaction is confirmed, the channel is created and the nodes can transact off-chain by updating the balances on both sides. Once they do not need the channel anymore, they close it via another on-chain transaction and withdraw their funds according to the final channel balances.

– **Payment Channel Networks (PCNs)** Payment channel networks [24] facilitate multiple parties to connect through ledger channels. Here a single party can initiate a transaction to another, despite the fact that they do not share a channel between them. The connection is made via an intermediary party which share a channel with each of the parties involved in the transaction. PCNs introduce several drawbacks as well. It relies on the liveness assumption of the intermediary, reduces privacy as intermediary party can see the transaction content. And finally could pave the way to a high cost, if intermediary charges a fee.

– **Virtual Channels** Virtual channels are introduced as an alternative to mitigate drawbacks in the previous approach. It allow two parties to send payments without interacting with intermediary parties between them. Here all intermediary parties along the route lock coins for a fixed period of time making a "Virtual"

private channel between the sender and receiver [25]. Virtual channels first were introduced for payment channels [26] and later extended to generalized state channels [22].

2.2.3 Drawbacks of State Channels

State channels introduce an efficient solution for users when they make frequent transactions between them. They drastically eliminate the time to finality, if both parties agree on the latest state and additionally remove transaction cost in Layer-1. But they are not favourable for mass adoption. The main shortcoming of state channels is "Predefined Participants". Users have to initiate a new channel each time when they want to make transfers with a new participant. This limits the usability of channels as it introduces addtional set-up cost.

Additionally, state channels assume liveness of participants throughout its lifetime. If a malicious participant tries to steal funds submitting an "out-dated" state, other users must monitor to "challlenge" them. There are solutions that have been introduced to alleviate these risks incorporating third-party watchers. But these remedies still would not attract users as it introduces another cost factor.

Channels also suffer from the "Data unavailabality" issue as off-chain transactions are not recorded on mainnet. Through the protocol itself, users are required to maintain the latest state but if a user is denied to maintain its liveness due to network failures or mechanical issues, security of his assets are relied on the honesty of other channel participants.

2.3 Sidechains

Sidechain [27] [28] is a separate blockchain, which runs in parallel to the mainnet which is connected to the underlying parent chain via a two-way peg bridge. "Two-Way" peg bridges enable sidechains to transfer assets between the sidechain and mainnet. Sidechains ease some of the drawback faced in the state channel protocols. It does not assume continuous liveness of users and removes constraints on predefined user sets.

Normally sidechains operate under a completely different blockchain scheme. In general they sacrifice strong assumptions on decentralization and security to achieve a high throughput. Based on the definition presented in [29], a sidechain should satisfy three conditions. Firstly, it has a native token pegged to a parent chain through a bridge, it does not require a token different from the mainnet to function and it

achieves the transaction finality based on its own consensus independent from the parent chain.

2.3.1 Architecture

Sidechain takes the same construction of a Layer-1 blockchain. But it may differ from its parent chain in several ways. In this case, the architecture of a sidechain shall be organized in three main components.

- **Block Parameters** Sidechain may design its block structure completely different to its parent chain. These design considerations will directly improve some characteristics of the system. Increasing the "Block Size" and reducing the "Block Time" will achieve a high throughput allowing fast transfers.

  But this will have strong implications on the decentralization of the sidechain as it will need validators with more sophisticated hardware to operate.

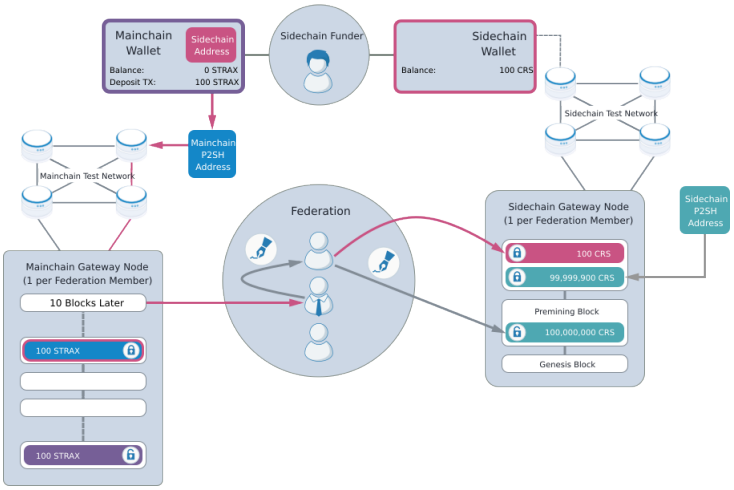
- **Consensus Algorithm** Sidechains can define its own consensus mechanism despite the parent chain implementation. A federated sidechain consensus is introduced in [30]. It presents a set of entities called *functionaries* who are in charge of the sidechain consensus. Proof of Authority (PoA) based consensus is proposed in [31] for its sidechains. A Non-Interactive Proof of Work (NIPoPoW) based consensus mechanism is discussed in [32] introducing the first trustless construction for proof-of-work sidechains. A suitable construction for a Proof of Stake (PoS) based sidechain is presented here in [33].

- **Bridge Construction** The bridge facilitates fund transfers between the sidechain and parent chain in a secure manner. It may be a smart contract in a EVM based blockchain or a federated system in Non-EVM based blockchain [29]. Figure 2 shows the execution of a federated peg sidechain.

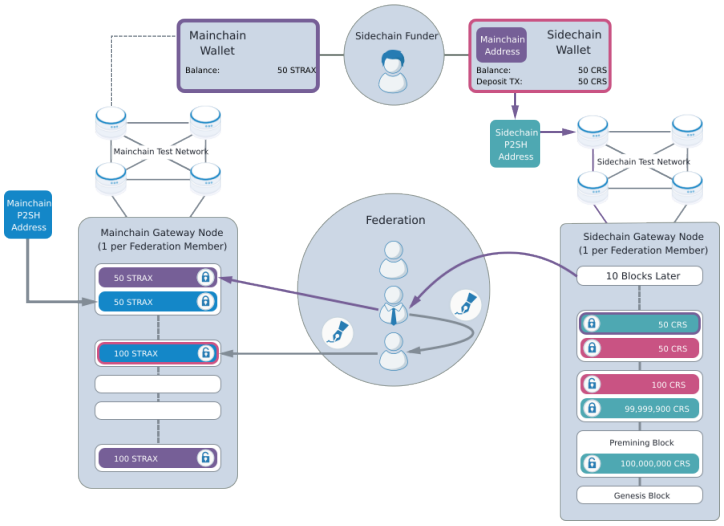
2.3.2 Drawbacks of Sidechains

A sidechain allows efficient transaction execution outside of the main blockchain. It eliminates issues inherent to state channel implementations. Futhermore it does not incur additional overhead on third-party watchers or liquidity requirements.

The major weakness of sidechains is lack of security. Due to its separate consensus, sidechain publish neither attestation nor transaction data in underlying blockchain.



(a) Depositing into a sidechain



(b) Withdrawing from a Sidechain

Figure 2: The role of the federation in a two-way federated peg sidechain

Source: `https://academy.stratisplatform.com`

Therefore users are not guaranteed with the Layer-1 security on sidechains. Consequently, Sidechain consensus shall be weakly assumed based on the consensus protocol used. Even under a PoW or PoS based consensus, they may not reach mainnet level of security due to the limited set of miners or validators who are vulnerable to varies types of attacks.

2.4 Plasma

Plasma [34] proposes a "blockchain of blockchains" to improve the scalability issue in public blockchains. Plasma-Chains (aka "Child-Chains") are considered as scaled down versions of the Layer-1 blockchains, which are rooted to the mainnet using smart contracts. It extends the concept introduced in "Side Chains" improving security and mitigating generalization issues of "State Channels" enabling unlimited user participation.

Initial proposal of Plasma [35] does not implement any chain but describes a specification with basic building blocks of a protocol. It enforces a tree of blockchains with cryptographic commitments in the root chain. These merkleized commitments ensure the security of assets in each of the child-chain.

2.4.1 Architecture

The idea of Plasma relies on Off-chain execution of transactions in child chains which has its own consensus protocol. But these child chains are still anchored to the parent chain (mainnet) via a smart contract (or bridge). These child chains periodically send a state commitment to the parent chain guaranteeing the integrity of the system.

The basic structure of a Plasma implementation shall be decomposed into three components [36].

- **Off-chain Computation** Plasma assumes not all transactions need to be verified in the maninet. Hence an Off-chain operator is given the responsibility of transaction ordering and execution at the cost of security. This necessarily introduces a trade-off between trustlessness and instant finality.

- **State Commitments** While increasing the throughput, Off-chain execution lacks transparency of the system. To overcome this issue, Plasma operator is required to publish state commitments on the mainnet periodically. Here state commitment takes the form of the root hash of a Merkle Tree. A merkle root provides the information on the latest off-chain state, allowing users to withdraw funds permissionlessly.

- **Exit Mechanism** The exit mechanism provides the methodology to withdraw funds from plasma chain to mainnet using state commitments. The exit strategy differs over the particular implementation of Plasma. If Plasma is based on a UTXO model, users have to provide a merkle proof verifying that the withdrawal

transaction is included in a block. If tokens are "Non-Fungible", the proof should attest the ownership of the token.

2.4.2 Variants of Plasma

Plasma has many realizations built on top of the basic concept in [35]. These implementations are specified on execution level, token types, consensus protocol and ledger classification (UTXO and Account based) etc. Here we present some plasma variants initially categorized in [37].

Plasma MVP

The first plasma version of Plasma is "Plasma MVP" [38] introduced by the authors of initial Plasma proposal. It proposes a model for simple payments transfers under a UTXO model. It further implements a challenge system and an exit mechanism to overcome malicious operation.

Plasma Cash

"Plasma cash" [39] was proposed to overcome problems in Plasma MVP. It leverages non-fungible tokens and sparse merkle trees to reduce the data storage and bandwidth requirements for users. Plasma cash improves the scalability of the system as users only need to keep track of their own tokens. But as it utilizes NFTs, tokens are in fixed denomination. As a consequence fractional token transfers are not possible reducing the flexibility and usability of the scheme.

Plasma Debit

"Plasma Debit" [40] try to improve issues in Plasma Cash on non-fungibility. The major difference of Plasma Debit is every token can be used as a payment channel between users or operators. It enables user transactions with both fungible and non-fungible tokens. But it still needs to keep track of one's tokens on a possible dispute by the centralized operator.

2.4.3 Drawbacks of Plasma

With the move of off-chain transaction execution, Plasma extends scalability aspect in Layer-1 blockchains. But it introduces many other issues into the system making it inapplicable as a Layer-2 scaling solution.

The major downside of Plasma is "Data Unavailability". When data is unavailable, users are unable to confirm the accuracy of state pledges issued by operators in the main chain. Accordingly trust in the system may be damaged, if a malicious operator refuses to respond to user requests for the transaction data that forms the basis of block commitments. This will inevitably lead to a "Mass Exit" scenario where users attempt to withdraw funds from Plasma in the presence of a fraudulent operator.

Additionally, in some Plasma variations, such as Plasma MVP, users must keep an eye on all system transactions to confirm the state. With the use of a sparse merkle tree model, Plasma Cash overcame this problem, but to maintain token security, users must log in at least once every two weeks [41].

2.5 Rollups

Learning through the pitfalls of previous Layer-2 scaling solutions, a new paradigm was introduced in a Github repository under the name of "roll_up" [42]. This well presented and already sophisticated protocol, described a cryptographic proof based Plasma type data structure with a "bundled-up" transaction posting strategy in Layer-1. The concept in Rollups removes the operator trust in Plasma, security implications of Sidechains and liveness assumptions in State Channels.

Rollups bring transaction execution off-chain like both Plasma and Sidechains solutions. But compared to Sidechains, Rollups still rely on the security from the mainnet as the settlement and consensus are still achieved through it. Rollup differs from Plasma, since Rollup operators will post compressed transaction data for state verification along with state commitments. This data availability enables users to re-execute all transactions in the system increasing trustlessness.

2.5.1 Architecture

Rollup is called a hybrid Layer-2 solution as it assumes Layer-1 security for consensus. The general mechanism of a Rollup consists of several operators with different responsibilities in the protocol. These operators are assigned with off-chain transaction aggregation and execution, validation and dispute resolution. Multiple perspectives on the implementation methods on these aspects have lead to various paradigms of Rollups.

Basically a Rollup aggregates user transactions, which are either directly or indirectly received in the mempool into a block. This block will be executed against the

latest state of the Rollup state and the resultant merkleized state root will be posted in the Layer-1 blockchain with transaction data.

This will be the latest state of the Rollup assuming the honest computation of the Rollup operator. The major variation of Rollups happens in the verification of this published Rollup state. One Rollup solution assumes Rollup operator is honest until proven guilty, resulting on Optimistic Rollups [43]. And the other solution requires operator to prove the validity of its commitments cryptographically; Zero-Knowledge Rollups [44].

2.6 Rationale for Rollups as a "Feasible" Layer-2 Scaling Solution

The Layer-2 solutions presented earlier aim to solve the problem of blockchain scalability. However, each solution offers different features based on unique underlying technologies and involves various trade-offs on fundamental blockchain principles. Therefore, it is crucial to assess and analyze these solutions to identify correct scaling solution to facilitate ever-increasing user adoption.

Analyzing a Layer-2 solution is a tricky task as every solution promises to be trustless, secure and economically feasible. Therefore it is necessary to classify the features of a scaling solution which could ultimately achieve the requirements of a feasible Layer-2 model.

An approach on creating a mental model for evaluating Layer-2 solutions has been proposed in [45], which presents a questionnaire. The framework is designed to guide users in assessing the advantages and limitations of various Layer-2 solutions based on a specific criteria. It asks four questions on the target solution, "*Who operates it?*", "*How is the data?*", "*What is the stack like?*" and "*How does it prepare for the worst?*". When decomposing these questions into technical terms, it concerns on the trustlessness, data availability, composability and the security of the corresponding technology.

Numerous studies have been conducted to evaluate the advantages and limitations of Layer-2 solutions [47, 48, 49]. One such study proposed a comprehensive framework for analyzing Layer-2 solutions as presented in [50]. This framework provides a structured approach to evaluating Layer-2 solutions covering most of the features of a permissionless Layer-2 system.

The evolution of Layer-2 solutions is illustrated in the figure 3. It demonstrates the gradual improvement of Layer-2 solutions from restrictable static participation to

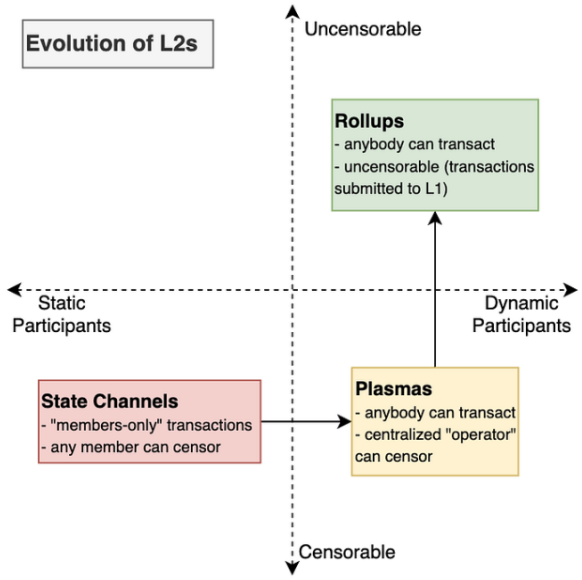


Figure 3: The Evolution of Layer-2 Solutions [46]

uncensorable and unlimited user participation solutions.

In Table 1 we compare major Layer-2 scaling solutions referring to the studies mentioned above. In terms of performance, State channels, Sidechains and Plasma could theoretically perform better than Rollups. But Rollups presents the best security model as it improves the data availability problem we saw in Plasma. Moreover Rollups do not assume any user or third-party to be live in the system to protect their assets.

One downside of Optimistic Rollups is its considerably slow withdrawal time. Because of the use of challenge-response protocols, it requires at least 1 week of a challenge period to reach finality. Furthermore, Rollups have the best composability aspect for EVM-compatible blockchains in Layer-1.

The Ethereum community has recognized the significant benefits that Rollups offer for scaling the system [51]. This recognition highlights the potential of Rollups to enhance system performance and represents an important step forward in the adoption of this technology.

Performance and Operations					
Aspect	Channels	Sidechains	Plama	Optimistic Rollups	ZK Rollups
Transaction Speed	Fast	Moderate	Moderate	Slow	Slowest
Transaction Cost	Very Low	Low	Very Low	Low	Low
On-chain Account Opening	Yes	No	No	No	No
Capital Efficient	No	Yes	Yes	Yes	Yes
Temporal Requirements					
Withdrawal Time	One Confirmation	One Confirmation	1 Week	1 Week	<10 minutes
Finality	Instant	N/A	One Confirmation	One Confirmation	< 10 minutes
Instant Transaction Confirmations	yes	No	No	No	No
Security Considerations					
Cryptographic Primitives Used	Standard	Standard	Standard	Standard	New
Liveness Assumption	Yes	Bonded	Yes	Bonded	No
Mass Exiting	No	No	Yes	No	No
Fund Freezing by Validators	No	Yes	No	No	No
Vulnerability to Crypto-economic Attacks	Moderate	High	Moderate	Moderate	Moderate
Composability & Privacy					
Smart Contracts	Yes, Limited	Yes, Flexible	Yes, Limited	Yes, Flexible	Yes, Flexible
EVM-bytecode Portable	No	Yes	No	Yes	Yes
Native Privacy Options	Limited	No	No	No	Full

Table 1: A Comparison of Layer-2 Scaling Solutions [49, 50]

3 State-of-the-Art

In this chapter we delve into latest variations of Rollups (3.1), further investigating Optimistic Rollup (3.2) and Zero-Knowledge Rollup (3.3) implementations. Within each subsection, we explore associated technologies such as dispute resolution (3.2.3) in Optimistic Rollups and Zero-Knowledge proofs (3.3.4) in ZK Rollups and introduce latest initiatives working in both avenues under "Related Projects" (3.2.4, 3.3.5).

Finally, we have conducted an extensive analysis (3.4) of the latest cost analysis on Rollup economics. In this analysis, we delve deeply into the cost structure of Layer-2 solutions, identifying the key cost factors involved. Additionally, we conclude this chapter by discussing the challenges (3.5) faced in Optimistic Rollups and exploring cutting-edge solutions found in the current literature.

3.1 Rollups

Rollups are hybrid Layer-2 solutions [52] which post their blocks to another blockchain and inherit the security from the underlying blockchain. In the context of Layer-2 solution for blockchains, Rollups are the leading contenders, offering capabilities that closely resemble those of a Layer-1 blockchain. Different architectural views have been put forward in the recent proposals of several Rollup architectures [53, 54]. These proposals leverage the modular architectural view of the blockchain.

In a traditional or so-called monolithic blockchain, all operational tasks are carried out by network operators. These resource consuming tasks have been viewed differently in a novel architecture called "modular blockchains". In modular blockchain design, resources such as computation, bandwidth and storage are dedicated into a stack of "Layers" (See Figure 4).

3.1.1 Flavours of Rollups

Different flavours of Rollups have been realized incorporating the layered structure of modular blockchains. The latest break down of Rollups are depicted in Figure 5.

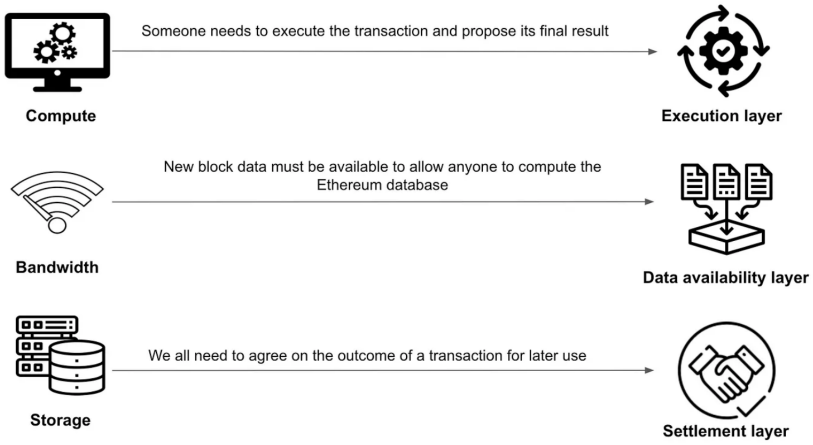


Figure 4: Layered View of Blockchain Tasks[55]

- **Smart Contract Rollups** Smart contract Rollups (SCR) are the first break through of this paradigm. Almost every current production level Rollups are different variants of SC Rollups. SCRs capitalize on bring execution layer off-chain reducing the computational burden off Layer-1. They operate using a trust-minimized bridge between the off-chain Rollup and Layer-1 blockchain.
- **Enshried Rollups** Enshried Rollups (ER) [56] could be considered as the ultimate goal of the Rollup paradigm and still a very futuristic concept. It implements the Rollup directly into the Layer-1 specification providing an inherited security.
- **Sovereign Rollups** Sovereign Rollups (SoR) [53] bring both execution and settlement layers, off-chain. It operates like a Layer-1 blockchain for settlement layer, allowing block posting and dispute resolution directly via the peer-to-peer network. In contrast to SC Rollups, SoRs do not implement a trust minimized bridge.
- **Settlement Rollups** Settlement Rollups (SeR) [54] are a different structure of Sovereign Rollups, where it implements two distinct Rollups for settlement and execution. In could have a sovereign or non-sovereign Rollup for settlement layer and a recursive Rollup for execution layer. Additionally they need a trust-minimized bridges between each layer.
- **Validium** Validium [57] moves data availability layer away from Layer-1, resulting a pure Layer-2 solution rather than being "hybrid". This adds security

implications to the system, but it attempt to assures the data availability by decentralizing data storage through guardians, data committees etc.

Smart Contract Rollup model is the most matured paradigm out of all above designs. Therefore we focus our state-of-the-art analysis on SC Rollup architectures. SC Rollups shall be divided into two main categories: Optimistic Rollups and Zero-Knowledge Rollups, which are explored in the following sections.

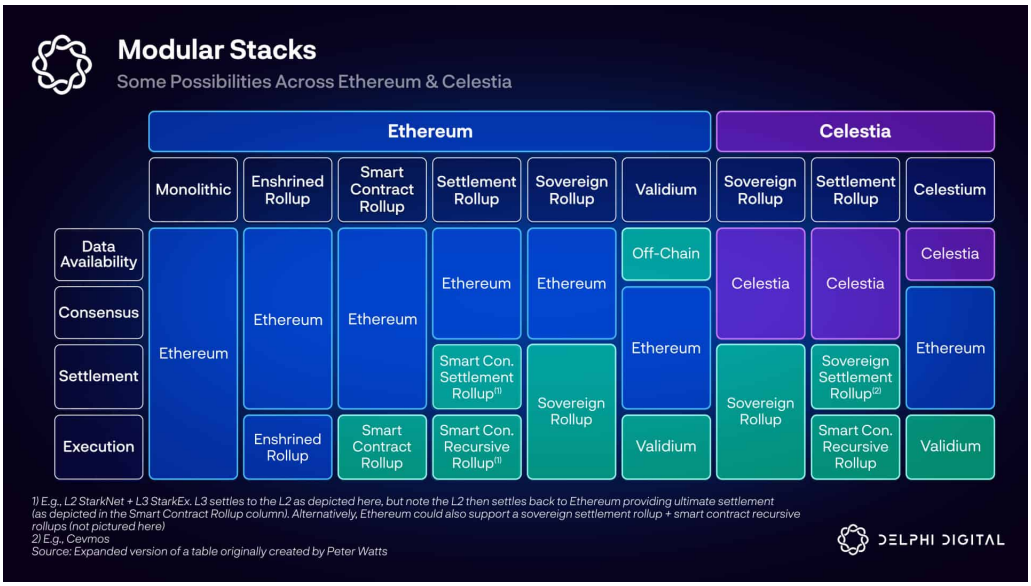


Figure 5: Flavours of Rollups

Source : "The Complete Guide to Rollups" by Delphi Digital

3.2 Optimistic Rollups

Optimistic Rollups [43] offer Layer-2 scaling with less computation in parent-chain by processing transactions off-chain. This improves the throughput of the system while maintaining the desirable security through Layer-1 mainnet. These constructions resemble Plasma [34], but trade the scalability of Plasma for a better security. Current implementations of Optimistic Rollups are already providing an EVM equivalent execution in their virtual machines. This EVM composability, enables a smooth migration for existing Web3 applications to achieve low gas fees and a high throughput.

Optimistic Rollups bundle-up batches of transactions off-chain before submitting in the underlying Layer-1 chain. This allows to spread the fixed costs occurring in

on-chain reducing fees for Layer-2 users. The term "Optimistic" is because it assumes off-chain transactions are all valid until proven wrong. And it does not publish a validity proof as done with Zero-Knowledge Rollups [44].

Instead of having a cryptographic proof on the correctness of transactions, Optimistic Rollups rely on a fraud-proving mechanism to detect malicious activities. For fraud detection, Optimistic Rollups require special “Watchers" to maintain system integrity. This leads to a liveness assumption of at least one active inspector in the system, which is referred as “Verifier’s Dilemma” 3.5.2.

3.2.1 The Vanilla Architecture

The architecture of an Optimistic Rollup could change with different implementations. But in high-level all implementations follow the basic concept. Following Layer-1 blockchains, Rollup system is democratized between various actors or components. This is a one distinct point where current Optimistic Rollups differ in their architectures.

In the vanilla Optimistic Rollup architecture, there are two main Layer-2 actors actively involved in the process, namely “Sequencer” (aka aggregators) and “Verifier” (aka Validators, Watchers, Challenger). Generally, transactions are collected by the sequencer which are received either directly by Layer-2 users or via an inter-chain transaction. An inter-chain transaction is done through a Layer-1 smart contract (bridge) and forwarded to the sequencer mempool.

The sequencer selects an arbitrary number of transactions, execute them and generate a new merkle tree for the updated ledger. He then publishes the compressed transaction data and the latest state root in the underlying blockchain, committing a collateral for their validity. Meanwhile, verifiers monitor all activities in the Layer-2 network and continuously check the correctness of the data published in Layer-1. If a verifier identifies a fraudulent state update in the data published by the sequencer, Rollup protocol enter the so-called “dispute resolution” phase.

Dispute resolution is a gas expensive process as it is executed in Layer-1 chain. The fraudulent party will be punished by slashing his collateral or fidelity bond. The sequencer stake a collateral, which will be slashed if the fraud proof fails to validate the integrity of state commitments. Verifiers equally stake a collateral that will be slashed if they challenge an honest sequencer. By utilizing these cypto-economic incentives and penalties, Optimistic Rollups have established a mechanism to ensure system integrity and safety.

Finally, the appended block with the latest state root will reach the “finality”

after a successful challenge period. Finality in blockchain refers to the point at which a transaction or block is confirmed and becomes permanently recorded on the blockchain. Once a transaction or block has achieved finality, it is considered to be fully validated and irrevocable.

3.2.2 Latest Architectural Trends in Optimistic Rollups

In this section, we will take a deep dive into the latest Optimistic Rollup architectures that are currently available in the ecosystem. The Layer-2 scaling landscape is currently dominated by Optimistic Rollup projects, and two leading projects, Optimism [58] and Arbitrum [59], have made significant contributions to the research and development of the Rollup concept.

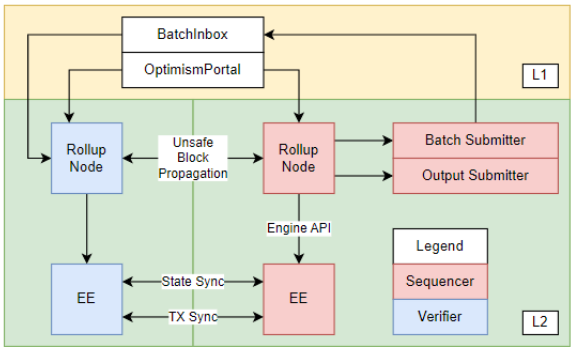
Optimism Rollup Architecture

Optimism's latest upgrade "Bedrock" went on live in 6th of June 2023. With this hard fork, Optimism promises much lower fess and faster speeds for transactions. The upgrade is designed to be the fisrt EVM-based rollup with "Ethereum level" of security. Bedrock is a complete redesign of the previous system to support its highly anticipated fault proof protocol [60] and to increase reliability.

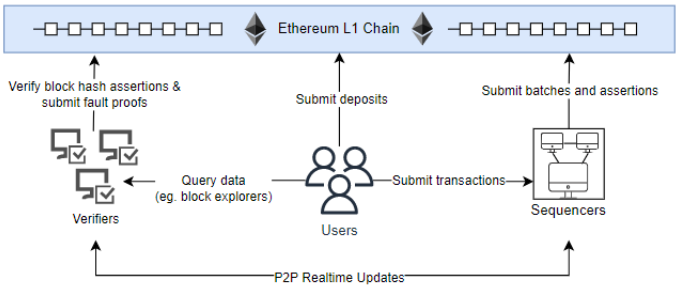
System Actors

- **Sequencer** The sequencer is the rollup node responsible for producing the Layer-2 blocks. Apart from block production it gossips out blocks to validators for verification.
- **Verfier** The Verifier, also referred as validator in the specification, is responsible for verifying the blocks by re-deriving output roots and comparing them against those submitted by the sequencer. In case of a mismatch, the verifier can perform a fault proof.
- **Batch Submitter** The batch submitter, also referred to as the batcher, is the entity submitting the Layer-2 sequencer data to Layer-1, to make it available for verifiers.
- **Output Submitter** The output submitter or proposer, submit the Layer-2 state commitments to the Layer-1. This state roots verified by the validators and will be contested with a fault proof in case of a mismatch.

This is a refined version of the system components we presented in the vanilla architecture. But in the actual Optimism implementation, batch submitter and output submitter roles are incorporated to the sequencer node. Even though this results back to the basic concept,they are expecting to democratize the sequencing role further by splitting as described in later versions.



(a) The System Architecture



(b) High-level construction of Optimism Rollup

Figure 6: The System Architecture and Construction of Optimism

Source: Optimism bedrock specification

The Transaction Lifecycle

The general overview of the transaction flow could be explained with the system diagram in the Figure 6. Firstly, users deposit funds in Layer-1 contract via on-chain transactions. The sequencer reads these deposit events and continuously generates new blocks in Layer-2. The sequencer then notifies the batch submitter about the new blocks and syncs with verifiers off-chain over the state and transaction blocks.

Once after it knows about the new block information, batcher submits the batches to the Layer-1 contract. Here verifiers have the capability of extracting transaction blocks

directly off-chain from the sequencer, but there is no guarantee if the data is going to be published in the mainnet, hence called "Unsafe blocks". The blocks produced by the sequencer are considered technically insecure, until they are confirmed on the Layer-1 blockchain. But once the batcher submits data into the Layer-1, verifiers can validate unsafe blocks he got over *peer-to-peer* connections and create fraud proofs based on the corresponding output roots submitted through the output submitter.

Deposits ($L1 \longrightarrow L2$)

Users first have to send transactions to the optimism portal, which is the bridge for Layer-1 to Layer-2 communications. Then the portal contract check for the transaction validity and charge for the Layer-2 gas usage. After it has passed the checks on gas levels and data correctness, portal contract emits an event to the Layer-2. The sequencer then catch this deposit event and create a block including the transaction.

Layer-2 Transactions ($L2 \longleftrightarrow L2$)

For transactions within Layer-2, users can initiate transactions to the transaction pool. Sequencer picks up these transactions from the mempool and includes it in its latest Layer-2 block. After it goes through the process described above in the lifecycle section, verifiers will decide on the validity of them.

Withdrawals ($L2 \longrightarrow L1$)

Users send a Layer-2 transaction to the L2 withdrawal contract. The withdrawal contract check on the validity of the transaction (fund availability etc.) and lock the funds. Based on the event emitted by the withdrawal contract, proposer submits an output proof to the Layer-1 with Layer-2 state root and withdrawals contract root. After the dispute challenge period expired, user can withdraw their tokens through the Layer-1 contract.

Arbitrum Rollup Architecture

As compared to the "Bedrock" upgrade from Optimism, Arbitrum's latest version is "Nitro"[61]. Nitro improves from Arbitrum's classic version in few ways. It further drives down on transaction costs by reducing the amount of data posted in Layer-1 using an advanced Calldata compression mechanism. While deploying an Ethereum

compatible gas pricing mechanism, it enhances interoperability with Layer-1 enabling a tighter synchronization.

System Actors

- **Sequencer** The sequencer is a full node who is responsible for submitting sequenced blocks in Layer-2. It is trusted for ordering incoming transaction honestly according to a First-Come-First-Serve (FCFS) policy. The sequencer's role is defined into a limited capacity. it does not possess power to prevent the chain from making progress, nor can deny a transaction getting into the Layer-2.

- **Validator** Validators are the nodes which allow the Rollup protocol to progress securely. Every validator can choose their own role.

  - **Active Validator** The Active validator participate in the Rollup protocol actively by proposing new Rollup blocks called "RBlocks", with a collateral being staked for it. A chain only need one honest active validator to process correctly, which is currently operated by the development team itself.

  - **Defensive Validator** The defensice validator carefully observes the chain as it operates with the submitted RBlocks. If an incorrect RBlock is proposed by an active validator, defensive validator intervenes by posting a correct RBlock or staking on a correct RBlock that is already available in the chain.

  - **Watchtower Validator** The watchtower validator simply watches the Rollup protocol and if an incorrect RBlock is proposed, it raises an alarm, so others get informed and could intervene. These nodes never actively participate in the protocol nor stake on RBlocks.

  In general defensive and watchtower nodes won't do anything except observing the chain. But they play a critical part on the fraud-proof progression of the system.

The Transaction Lifecycle

The transaction lifecycle starts with users submitting various type of transactions to the sequencer. The sequencer can receive them in two ways, directly or over Layer-1.

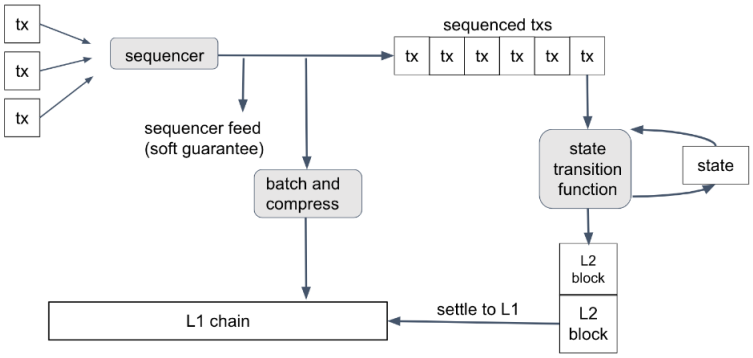


Figure 7: The System Architecture of Arbitrum Nitro[61]

Typically, under the assumption of an honest sequencer, users can submit the signed transactions directly to the sequencer. Alternatively, a user can submit transactions by signing and publishing a Layer-1 transaction to the Arbitrum's "Delayed Inbox" contract. The Delayed Inbox has two purposes, first it enable Layer-1 contracts to submit transactions in Layer-2 and secondly it mitigate censoring possibly incurred thorough a malicious sequencer.

Upon receiving a transaction, the sequencer will order them as they arrive (FCFS policy), execute it using the Arbitrum Nitro VM and instantly give a "soft" finality to the users. The sequencer will periodically batch a certain amount of transactions and post the compressed batch into Layer-1. With this process, transactions reach the hard finality. Users are given three varieties of finalities in Nitro architecture.

- **Soft finality** (≈1 second)
  soft finality is achieved through the sequencer feed. Any party can subscribe to the sequencer feed and compute the state transition based on the sequenced transactions. It is called soft finality because there is no guarantee if sequencer honestly bind them in a block in the said order.

- **Hard finality** (≈10 mins)
  Layer-1 finality or hard finality can be observed once the compressed batches reach Layer-1 contract. User can decompress these batches and run the state transition function and verify on the finality.

- **Certification** (≈ 7 days)
  The certification is achieved after a block has survived the dispute period aka a one week. The disputation period depends on the maximum gas allowed in a rolled up batch of transactions and other factors.

Once the sequencer post Layer-2 blocks in Layer-1, an active validator can challenge on the validity of the published blocks. This is the "Optimistic Rollup Protocol" presented in Nitro's white paper [61]. The Rollup protocol produces a chain of Rollup blocks ("RBlocks"), which are not the same as Layer-2 blocks. An RBlock typically encapsulates a sequence of Layer-2 blocks and are much less numerous than Layer-2 blocks. A validator can make an on-chain assertion on the chain's latest state, i.e. RBlock state.

In the happy(or common) case, an honest validator will assert on a RBlock and over the dispute window, no other validator will challenge the assertion. Then once the challenge period expires, RBlock get the final certification and user can withdraw funds from Layer-1. A dispute triggers if two validators assert on different RBlocks (on separate successors of the same parent RBlock). Once an assertion get challenged, protocol automatically step into the challenge sub-protocol. After the challenge, only one validator will be able to win the claim while the other party being penalized with on the staked bond.

3.2.3 Dispute Resolution

Historically the major difference of Optimism and Arbitrum was over their dispute resolution mechanism. Optimism's[1] approach to dispute resolution was naively runninig the entire disputed block in the Ethereum Virtual Machine (EVM). It is called "single round" dispute resolution as the challenge is solved after one EVM run on-chain. On the contrary, Arbitrum uses a "multi round" dispute resolution protocol. This leads to a longer finality in the case of a dispute, but consumes less gas compared to a single-round protocol.

Optimism has announced their next generation fault mechanism scheme "Cannon" [62][60] claiming the first "EVM-equivalent" [46] public fault proof implementation. With Cannon, Optimism has abandoned their single-round dispute resolution as it has turned to Arbitrum alike interactive scheme. Here the protocol decomposes the problem into two steps, first finding the MIPS instruction (MIPS are at the heart of Optimistic VM) such that the challenger and defender (sequencer in Optimism's case) agree on the memory(RAM and registers) of the minigeth program before that step, but disagree on the memory after it. And finally running that disagreed instruction in Layer-1. A detailed description on the proposed architecture is given in its specification [60].

[1] At the time of writing the fault proof process is currently undergoing major redevelopment as a side-effect of the OVM 2.0 upgrade and is therefore not active on the current Optimism system.

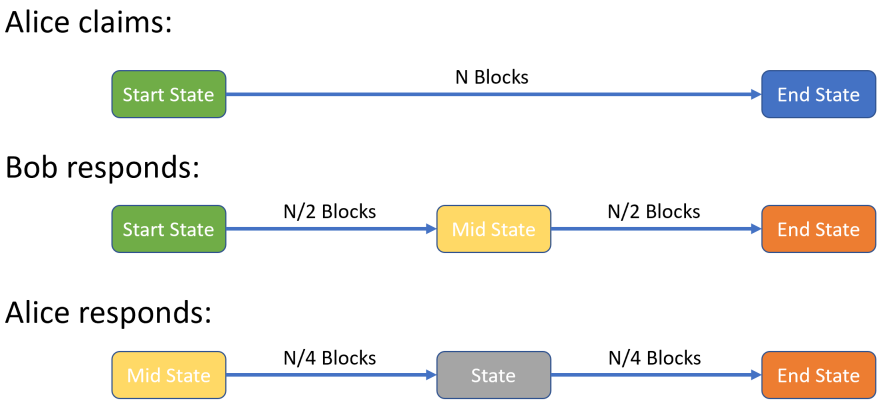


Figure 8: An Illustration of bisection of dispute resolution protocol

Arbitrum's latest version of the dispute resolution algorithm is presented in Nitro [61]. The challenge protocol operates in three phases. It first dispute over the blocks reducing the scope to a dispute over the creation of a single block. Second, challenge is carried out over the instructions (WAVM). Once the dispute is carried out recursively bisecting instruction sets, finally it converges to a dispute over execution of a one single WAVM instruction. In the final and third phase, one party (either challenger or defender) will produce a one-step proof to be carried out in Layer-1. A detailed illustration of a dispute resolution example is explained here in [47].

Say Alice and Bob are two validators in the Arbitrum network. Alice stake on the latest RBlock and Bob disagree with it and challenge Alice submitting a stake. Alice and Bob then play following steps before one of them appear as a honest player.

1. Alice claims result of N blocks starting with a finalized state and submitting a new end state.

2. Bob disagrees with Alice and claims on a different end state.

3. Bob now have to submit mid state half way through the N blocks.

4. Alice have to respond on her choice on the Bob's claim. Either if she agrees on start state to mid-state or mid-state to end-state.

5. Then again Alice has to submit the mid-state of the block set she find correct.

6. Finally theoretically after $\lceil \log_2 N \rceil$ rounds Alice and Bob achieve on a single block that they do not agree.

7. Then they bisect over the WAVM instructions re-running steps 1-6.

8. Alice or Bob call one-step proof verification contract on Ethereum.

In practice Arbitrum has improved the algorithm to be *d-way* dissection rather than bisection, where d = 400 [63]. Apart from it new protocol forces the participants who respond to their opponent's dissection to not only choose the challenging section but also to respond with a dissection of that segment into d subsections.

With these modifications the algorithm run time complexity reduces from roughly $2\log_2(NK)$ to roughly $\log_d(NK)$, by a factor of $2\log_2 d$ where, $K = instructions$ and $N = Blocks$.

3.2.4 Related Projects

Arbitrum

Arbitrum is a layer 2 scaling solution developed by Offchain Labs. The Arbitrum was founded in 2018 by three experts in the field on cryptography and blockchain technology. It is the leading player in the ecosystem with $2.3B TVL (Total Value Locked) and a market share over 51% at the time of writing.

Arbitrum is the only Optimistic Rollup with live a fraud proof mechanism. Arbitrum's sequencer is guaranteed to deliver fast finality given the user trust. In August 2022 Arbitrum released its latest Nitro technology, Arbitrum Nitro. Nitro comes with many improvements, such as higher throughput (7x-10x higher than pre-Nitro), an advanced calldata compression mechanism etc.

Arbitrum Roadmap

- Arbitrum initial development as a class project at Princeton - 2014
- Offchain Labs founded - Aug 2018
- Arbitrum One in mainnet - Aug 2021
- Arbitrum Nova in mainnet - Aug 2022
- Arbitrum One upgraded to Nitro - Aug 2022

Optimism

Optimism is a low-cost and lightning fast Ethereum Layer-2 blockchain with the second highest market share of 30% with a $1.29B TVL in escrow contracts on Ethereum. In November 2021, Optimism upgraded to OVM 2.0 to support EVM

equivalence. With this new design, Optimism was brought to one-to-one parity with Ethereum VM execution. As a part of the OVM 2.0 upgrade, optimism's fault proof mechanism is temporarily disabled. Therefore optimism users are expected to trust the sequencer to publish valid state roots to Ethereum. Optimism is planing to replace their single-round fault proof mechanism with a multi-round interactive fraud proof scheme. This is being developed in the Optimism Cannon project. Currently Optimism runs the sole sequencer on Optimism. It is planning on decentralizing the sequencer eliminating the Optimism's role enitrely from the protocol. Optimism plans on an economic mechanism incorporating an MEV scheme to create a competitive market for sequencing. In the second quarter of 2023 Optimsim plans to release its "Bedrock" upgrade. Bedrock comes with lower Layer-1 gas fees, efficient data compression, use of EIP 1559 for Layer-2 execution fees and reduced block periods.

Optimism Roadmap

- Introduced Optimistic Rollup - June 2019
- EVM Compatible Testnet- Sep 2020
- EVM Equivalent Mainnet - Oct 2021
- Open Mainnet - Dec 2021
- Bedrock - May 2023
- Next gen fault proof - 2023
- Sequencer decentralization - 2023

3.3 Zero-Knowledge Rollups

Zero-Knowledge (ZK) Rollup is the Rollup architecture described in the first proposal on Rollup paradigm [42]. But due to the complexity of implementation of ZK Rollups, Optimistic Rollups got ahead with the popularity. But with the recent announcements on tremendous progresses from ZK Rollup projects, they are again under the spotlight of blockchain enthusiasts.

Likewise Optimistic Rollups, ZK Rollups promise to increase the throughput by moving the execution part of the Ethereum to off-chain. Zero-Knowledge Rollups are considered the "holy-grail" of scaling solutions as it can batch up few thousand transactions with a minimum Layer-1 gas costs. ZK Rollup operation resembles the same architecture of an Optimistic Rollup except the verification. It derive the

security over Ethereum while publishing compressed transaction data as Calldata. It further maintains the Rollup state using smart contracts that act like anchors and are responsible for storing rollup blocks.

The validity of transactions in the Rollup is ensured by cryptographic proofs that are submitted to the Ethereum network for verification by the verifier contract. Compared to Optimistic Rollups, ZK Rollups offer much faster finality (in just a few seconds) through the use of cryptographic proofs. Once the verifier contract attests to the correctness of the proofs, transactions achieve hard finality. This improves scalability and throughput, making ZK Rollups an attractive option for those looking to improve transaction speed and efficiency.

3.3.1 The Vanilla Architecture

The general architecture of a Zero-Knowledge Rollup is almost same of its counter part, Optimistic Rollups. But the difference comes over the verification process. A typical flow of a ZK-Rollup can be seen as depicted in Figure 9.

Users initiate transactions and submit them to the sequencer. The sequencer create a batch of transactions in a single block and submit the latest "state-diff" to Layer-1 along with a validity proof. The state-diff contains all the changes made to the state as a result of the transactions in that batch. The proof and the state diff are verified by the verifier contract, confirming the correct execution of the transactions included in the block.

ZK-rollups architectures provide some favourable guarantees to users. They offer faster finality in seconds, system participants can never steal user funds as they are always secured by an on-chain verification contract. And they do not assume any strong assumptions such as liveness of at least one single honest validator as in Optimistic variation.

3.3.2 Latest Architectural Trends in Zero-Knowledge Rollups

Amidst undeniable security and faster finality guarantees, ZK Rollups are yet to reach the state of a "general-purpose" Rollup. The ZK proof unfriendliness of EVM and other various challenges in the core infrastructure [65], have been hindering ZK Rollups from achieving the so-called EVM equivalence [46]. Therefore, "Application-Specific" ZK Rollups have gained popularity in some specific niches, including simple payments [66], perpetuals trading [67] and gaming [68].

To accomplish the general Ethereum level execution, present ZK Rollup projects

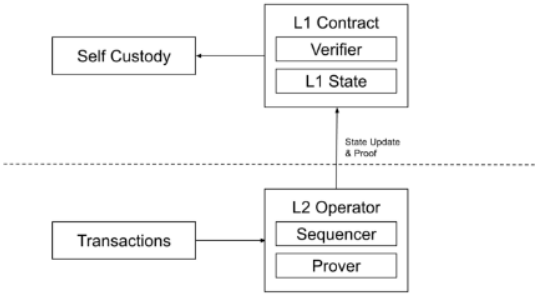


Figure 9: The Vanilla Architecture of a Zero-Knowledge Rollup[64]

are working towards implementing a "zkEVM". zkEVM is a general-purpose virtual machine that supports Zero-Knowledge proofs and has the potential to replace the current Ethereum Virtual Machine (EVM) in the future. Different types of zkEVMs are presented in [69], where it systematizes current zkEVM specifications according to the performance and EVM equivalence.

present realizations are developing on various trade-offs that they believe is the right-match for scaling Ethereum. These aspects consists, compatibility to current EVM ecosystem including programming language support and frameworks, capacity to facilitate EVM upgrades, performance of proofing systems etc.

These various choices have paved to a highly discussed categorization of zkEVMs such as full Etehreum equivalent, EVM equivalent, EVM compatible etc. At the time of writing some projects have self-claimed their implementations as EVM equivalent [70, 71].

In the following subsections, we present an EVM compatible and EVM equivalent ZK Rollup implementations in detail.

StarkNet ZK Rollup Architecture

StarkNet [72] is one of the leading ZK Rollup which provides an EVM-compatible execution. As it was the first project to adapt Zero-Knowledge proofs for Rollups, it has established a thriving ecosystem across various domains.

As with every other implementations, StarkNet introduces their own set of roles in the architecture. The system is composed of user accounts (or users), full nodes, a sequencer, a prover, a verifier and a core smart contract. So far the sequencer and the prover are centralized and controlled by the development team.

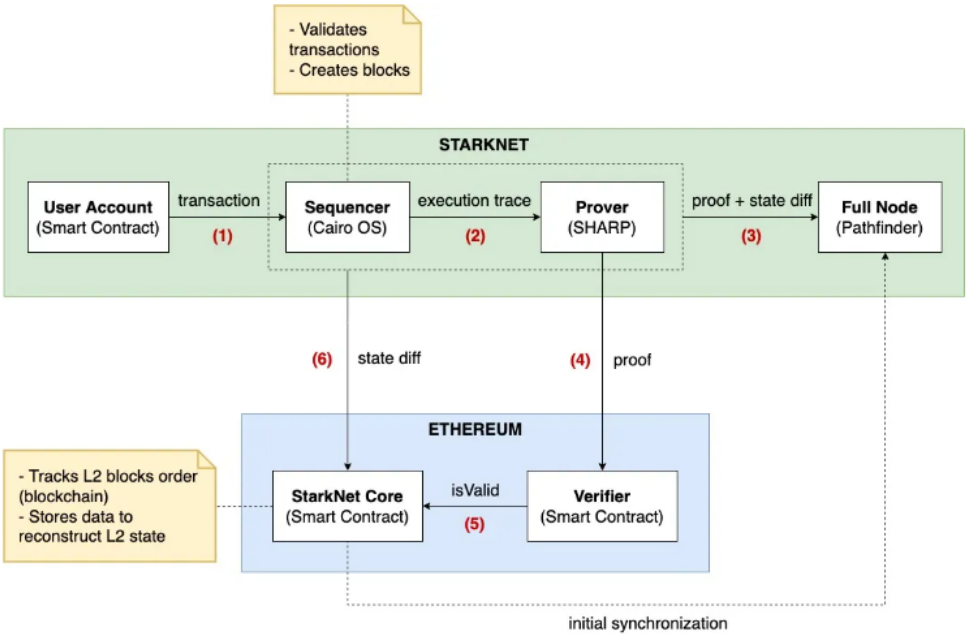


Figure 10: The System Architecture of StarkNet[73]

System Actors

- **Sequencer** Sequencer receives all the transactions, validates them and bundle them into blocks. For a sequencer to validate transactions it has to execute them using Cairo OS for smart contracts written in Cairo (the native smart contract language for StartkNet developed by StarkWare) [2]

- **Prover** Prover is responsible for generating cryptographic proofs that attest to the integrity of the computation performed by the Sequencer. For the prover to generate the validity proofs it requires to be given the "Execution Trace" of the computation performed by the sequencer.

- **Full Nodes** Full nodes keep the track of the blockchain recording all the transactions performed in the Rollups. They receive the information via a P2P network. A full node is capable of reconstructing the history of the Rollup by connecting to the Layer-1 and processing all the Layer-1 transactions associated with StarkNet.

- **Verifier** Verfier is a smart contract on Layer-1 which verifies the validity proofs

[2]Cairo is a language for creating STARK-provable programs for general computation.

submitted by the prover. The result of the validation is recorded in the core smart contract.

- **StarkNet Core** The core smart contract lives in Ethereum which receives changes to the Layer-2 global state when a new Layer-2 block is created and the verifier successfully validate a validity proof. The state transition is sent as "calldata" to save gas.

Transaction Lifecycle

A StarkNet transaction goes through several state transitions [74] during its lifetime until it reaches finality in Layer-1. In below we list down transaction states in the processed order.

1. NOT_RECEIVED: A transaction which has been sent by a user but has not reached the sequencer mempool is in the *not_received* state.
2. RECEIVED: Once the transaction is received in the sequencer mempool, it is marked as *received*.
3. PENDING: The transaction is executed successfully and added in the "Pending Block". The pending block accumulates new transactions until it has enough transactions to match up with the gas costs.
4. ACCEPTED_ON_L2: Once after the sequencer decides to close the pending block, it get accepted on Layer-2.
5. ACCEPTED_ON_L1: Validity proof is verified successfully in the Layer-1 smart contract and the transaction is accepted on-chain reaching the "Hard Finality".
6. REJECTED: The transaction is being rejected. This could happen in two scenarios where an assertion failed during the transaction execution by the sequencer or the block failed in verification step.

Scroll ZK Rollup Architecture

Scroll [75] is an another layer 2 scaling solution focusing on a fully EVM equivalent ZK Rollup. Scroll's zkEVM design is a combination of recent breakthroughs in both zero-knowledge proof generations and hardware acceleration [76]. It has two major components zkEVM, to attest validity of Layer-2 blocks and Layer-1 contracts to

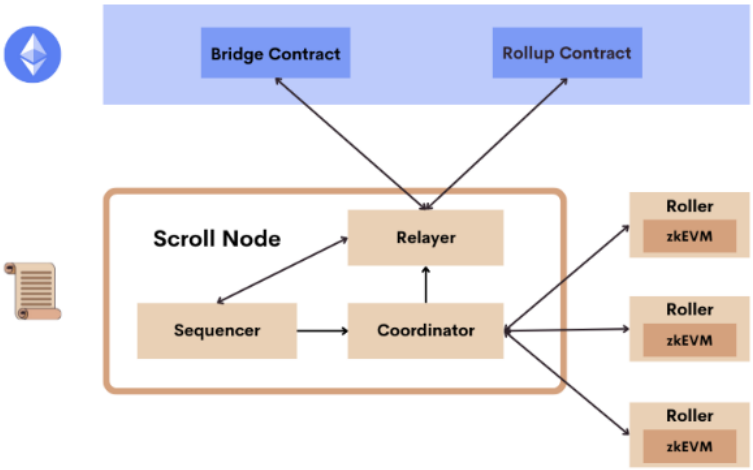


Figure 11: The System Architecture Scroll[77]

store states. The current version of the Scroll is composed of a centralized sequencer and a decentralized proving scheme. Their vision is to have a fully decentralized node scheme for both sequencing and proving.

System Actors

- **Sequencer** The sequencer is a part of the Scroll Node in Scroll's architecture. It retrieves a batch of transactions from Layer-2 transaction mempool, executes them and generate Layer-2 block with a state root.

- **Coordinator** The coordinator is notified once a new block is generated by the sequencer, and it selects a "Roller" randomly from the roller pool. Once it has selected a roller, the roller will be provided the execution trace (received by the sequencer) to generate a proof.

- **Relayer** The relayer monitors the state of the Layer-1 and Layer-2 contracts. Its primary task is to relay the messages between Layer-1 and Layer-2.

- **Roller** The roller are the "provers" in Scroll's architecture. They generate validity proofs for ZK Rollups. Rollers utilize hardware accelerators for this task (GPUs, FPGAs, ASICs etc.). A Roller first converts the execution trace to circuit witnesses and then generate proofs for each zkEVM circuit (zkEVM is

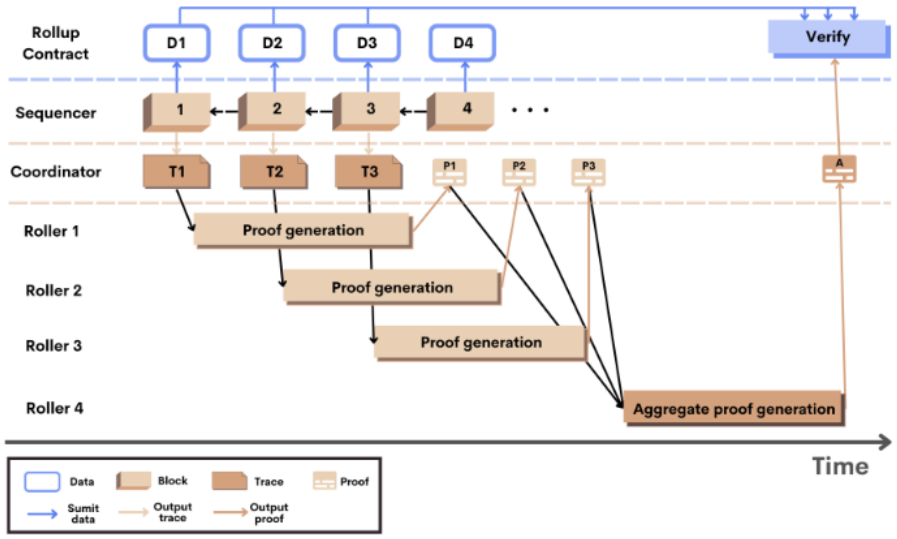


Figure 12: Scroll's Transaction Flow

Source : An Overview of Scroll's architecture Review[73]

consists of several modular circuits; EVM circuit, RAM circuit, storage circuit etc.). Finally it aggregate all sub proofs into a single block proof.

– **Layer-1 Contracts** Layer-1 smart contracts consists of two components, Rollup contract and Bridge contracts. Rollup contract receives the state roots of Layer-2 blocks and block data as "calldata". The Bridge contracts are responsible for managing asset transfers between the Ethereum and layer-2.

Transaction Lifecycle

Transaction lifecycle in scroll is summarized in Figure 12.

- The sequencer generates a Layer-2 block and submits the transaction data $\boldsymbol{D}$ to the Rollup contract in Layer-1.
- The Coordinator receives an execution trance $\boldsymbol{T}$ for the generated Layer-2 block, and randomly selects a Roller to generate a validity proof.
- The Roller generates a proof $\boldsymbol{P}$ and sends it back to the coordinator.
- The Coordinator selects another roller to aggregate k ($k = 3$ in the Figure 12) block proofs into a single aggregated proof $\boldsymbol{A}$.

- The Coordinator submits the aggregated proof to the Rollup contract to finalize Layer-2 blocks by verifying aggregated proof with state roots and transaction data already submitted by the sequencer.

During the transaction process, each Layer-2 block will go through three stages until it is finalized in underlying blockchain.

1. *Pre-committed* : Once the block is proposed by a sequencer and sent to the roller, Layer-2 block becomes to "precommitted" stage. This is a "Soft Finality" for users if they trust the sequencer.

2. *Committed* : This stage achieves once the transaction data is posted on the Rollup contract in Layer-1. This ensures the data availability of Layer-2 blocks promised by the sequencer in the previous stage.

3. *Finalized* : Once the validity proof corresponding to the Layer-2 block is verified, block is finalized in Layer-1. This is the "Hard Finality" and considered canonical block in the Scroll's chain.

3.3.3 Transaction Finality in ZK Rollups

Transaction finality of a ZK Rollup can be guaranteed once the validity proof is accepted by the verifier. The proof generation time varies based on the implementation and the validity proofing scheme.

In the case of an arbitrary smart contract, proof generation would be time consuming. But most of the implementations available at the moment in Layer 2 ZK-rollups space are application specific ZK Rollups. Proof generation for application specific Rollups are much faster than a zkEVM.

zkSync, a leading player in ZK Rollup space, claims to generate proofs approximately within 10 minutes [78]. So once after a batch of transactions are submitted to the sequencer, it could get into the finality in 10 to 15 minutes. But at the moment there is a problem with proof generation in terms of transaction batch size. Currently when a user sends a transaction, ZkSync's approach is to wait until the block has enough transactions to cover the Layer-1 gas costs. But they believe this will be mitigated once the system get congested.

Even though the finality takes 10 minutes to process, instant confirmations are promised by most of the ZK Rollup implementations. This so-called instant finality is purely based on the trust of the sequencer node. At the moment of writing, all Rollups implementations have a centralized sequencer run only by the development

team. But the users are given the liberty of waiting until full finality is achieved after max few hours.

In the future with decentralized sequencers, a staked bond or collateral will come into play when submitting validity proofs. This will allow systems to still achieve a instant finality (theoretically with mostly honest participants) in ZK Rollups. The validators or sequencers who are willing to participate in the protocol, will have to post a significant security bond to the mainnet.

Once a user submit his transaction to a sequencer, a consensus run by the validators provide a "sub second" confirmation to the user that his transaction will be added in the next upcoming zkSync block, signed by a super majority ($> 66\%$) of validators [79]. If the user's transaction is not included in the next block, security bonds of the intersection of the signers of the original receipt (which promises user for inclusion) and the signers of the new block will be slashed. The user will be compensated with the slashed tokens and the rest will be burned.

StarkNet has an improvement proposal to decrease the time to finality using "Checkpoints"[80]. Instead of waiting for a new block to reach Layer-1, state transition checkpoints are submitted at intervals of 1 minute and 1 hour. Every minute, a validity proof is generated, corresponding to all transactions that have occurred during the last 60 seconds. The validity proof and the state transition during this interval is submitted to Full Nodes to achieve a soft/weaker one minute finality. But in every hour, there is a validity proof derived from all validity proofs created in that period and submits to the verifier along with the state changes during the interval to create a "strong" finality.

3.3.4 Zero-Knowledge Proofs

Zero-Knowledge Proof(ZKP) is an interactive verification protocol where two actors, "Prover" and "Verifier" communicate each other on the validity of a cryptographic statement without revealing any additional information. ZKP framework shown in figure 13 is going through three phases in the process of prover convincing the verifier on the validity of his commitment [81].

- **Witness Phase** Prover generates a cryptographic witness on a computation or a statement and send it to the verifier.

- **Challenge Phase** Verifier challenges the validity of the witness asking questions.

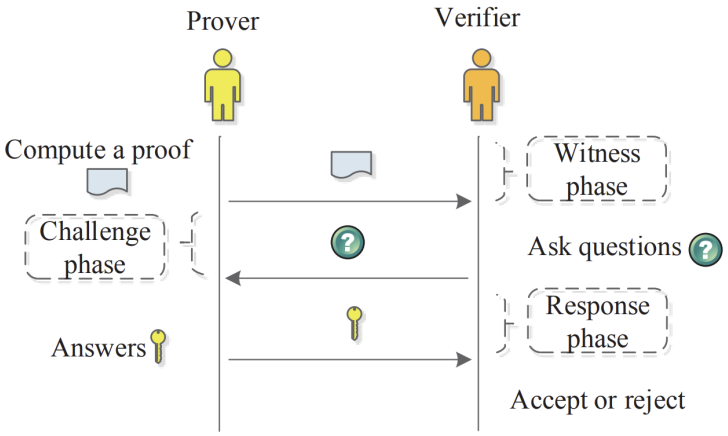


Figure 13: The Framework of Zero-Knowledge Proof [81]

Project	Proof Scheme
zkSync	Plonk[86], RedShift[87]
StarkNet	STARK, Deep-FRI[88]
Polygon zkEVM	STARK[89], SNARK[90]
Scroll	zkSNARK[91]

Table 2: Zero-Knowledge Proof Systems in ZK Rollups

– **Response Phase** Prover respond to verifier's challenges, on which verifier conclude on the correctness of the witness.

Furthermore a Zero-Knowledge proof system should satisfy three properties to fulfill its intended task. **Completeness**, that a honest verifier will always accept a true statement from a prover. **Soundness**, a malicious prover can not convince a verifier on his false statements except for a small probability. And **Zero-Knowledge**, that prover does not leak any additional information to prove his statement and verifier learns nothing except that validity of the statement.

ZK Proofs shall be divided into two main categories, interactive and non-interactive. Non-ineractive ZKPs are the main building blocks of ZK Rollups[82]. The most popular non-interactive ZKPs are SNARK (Succinct Non-Interactive Arguments of Knowledge)[83], STARK (Scalable Transparent ARgument of Knowledge)[84] and Bulletproof[85]. The table 2 displays a list of ZK Rollup projects and the corresponding proof systems they utilize.

3.3.5 Related Projects

StarkWare

StarkWare [92] is a pioneering company in Layer-2 ecosystem, one of its co-founders being an contributor to the original STARK whitepaper in 2018. StarkWare has two products; StarkNet and StarkEx. The StarkNet is a permissionless, general-purpose decentralized ZK Rollup network where users can write and deploy their own smart contracts, just like Ethereum. StarkEx is a standalone, customizable Layer-2 SAAS for exchanges that uses the STARK proof system in order to provide massive scaling. Even though it is still in its early stages it already has a huge ecosystem hundreds of projects being developed on it.

Going against other implementations StarkWare is developing a different ecosystem for development. They use the Cairo programming language for smart contract development instead of industry standard Solidity. Cairo is a turing-complete programming language which enables verifiable computation using STARKs. but Nethermind's Wrap project is building a traspiler to translate Solidity smart contracts to Cairo. StarkNet's roadmap[93] is basicaly based on three things; Performance, Cairo 1.0 release and Seamless Regenesis. In order to improve performance StarkNet is working on parallelization (Optimistic prallelization), rewrite cairo-VM in Rust "cairo-rs" and a new Sequencer to improve performance. Cairo 1.0 was released in Nov 2022 and Regenesis is planned next. During the regenesis StarkNet will go through a transition period and at the end of it will only allow Cairo 1.0 contracts and a new genesis block with existing state.

StarkNet Road Map

- StarkNet Alpha went live on mainnet - Nov 2021
- Cairo 1.0 release - Sep 2022
- Introduced The StarkNet Foundataion - Nov 2022
- StarkNet Token deployment on Ethereum - Nov 2022
- StarkNet Regenesis - 2023

zkSync

zkSync [94] by Matter Labs is another Layer-2 scaling project working on ZK Rollups. The company launched the first-ever public ZK Rollup prototype in January 2019 and was the first to implement recursive ZK Rollups on Ethereum. They introduced

world's first practical FPGA-based hardware acceleration for Zero-Knowledge proofs in 2020. They launched its zkEVM version called zkSync 2.0 in February 2022. The zkSync 2.0 is a highly efficient, Turing complete, SNARK-friendly virtual machine for executing smart contracts. zkSync 2.0 will support smart contract capabilities through Solidity and its native programming language, Zinc.

In October 2022, zkSync announced "Baby Alpha" release on mainnet. This includes deployment of end-to-end system on mainnet. During this stage users are not allowed to use the system as development teams are working on stress tests and security audits with top-tier auditors. At the end of 2022 zkSync plans to hit "Full Launch Alpha" making zkSync 2.0 available for all projects and users. With the "Baby Alpha" launch, zkSync 2.0 becomes the first zkEVM solution to go live on mainnet.

zkSync 2.0 Road Map

- zkSync 2.0 launch on a public testnet - Feb 2022
- Dynamic fees - Aug 2022
- Proof generation and validation - Oct 2022
- Baby Alpha - Oct 2022
- Full launch Alpha - Q2 2023

Polygon

Ploygon [95] is a Layer-2 scaling platform with both sidechain and rollup solutions. Polygon founded in 2017 as "Matic Network" by three developers. In 2021 they rename it as Ploygon but its token is still named as MATIC. Polygon has three ZK rollup solutions; Ploygon zkEVM, Polygon Zero and Polygon Miden. Polygon zkEVM (currently on public testnet) is the first open-source zkProver that provides complete EVM opcode equivalence and security. Developers can use existing contracts written in Solidity in polygon zkEVM. Polygon guarantees 90% fee reduction with on-chain data. Ploygon Zero (under development) is a Layer-2 scaling solution powered by Plonky2, a recursive SNARK proof that is 100x faster than existing alternatives. With Polygon Zero they claim to generate ZK proofs within 200 mili-seconds on a commodity laptop.

Polygon Miden (under development) is a general purpose ZK Rollups for arbitrary smart contracts. Miden relies on STARKs on proof generation. It is transparent (no trusted setup) and post-quantum secure. With Miden polygon assumes to achieve

1000+ TPS and 10,000+ TPS after sharding in Layer-1. It promises over 100x reduction on transaction fees with Ethereum. Polygon is an open source project, the complete sorce code is openly available for use. Their main project zkEVM is currently on a public testnet, they are planing to go live on mainnet in later 2023. In the mid 2022 Polygon open the code base, so it is an open-source project now.

Scroll

Scroll [96] is in the pursuit of building a native zkEVM Layer-2 solution for Ethereum. In October 2022, they announced an upgrade to its pre-alpha testnet enabling developers to write and deploy thier own contracts on Scroll using the same Ethereum developer tools. In Scroll's zkEVM is bytecode compatible with Ethereum. This is allows Scroll to get closer to be EVM equivalent. Scroll is yet to release their Alpha testnet, they are planing to implement a decentralized roller (prover) network and smoother developer experience with integration of EVM native developer tools.

Scroll Roadmap

- zkEVM PoC - Done
- zkEVM testnet - Up coming
- Layer-2 proof outsourcing - Up coming
- zkEVM mainnet - Up coming
- Decentralized sequencer and more afficient xkVm - Up coming

3.4 Rollup Cost Analysis

At the peak of the bull market in 2021 Ethereum got highly congested resulting in unprecedented transactions fees (Figure 14). It cost over 200$ to send a transaction using the Ethereum blockchain earlier in 2021[1]. Ever since this demand and supply dilemma has been alarming the crypto ecosystem. In order to facilitate mass adoption and exponential growth of user cases, the actions towards the scalability of the network such as "Layer-2" have been gaining momentum.

One of the key promises Layer-2 systems have made is "cheaper gas costs". With less computing and storage resources contained on the main chain, Layer-2 solutions can reduce the transaction fees drastically without compromising on security aspects.

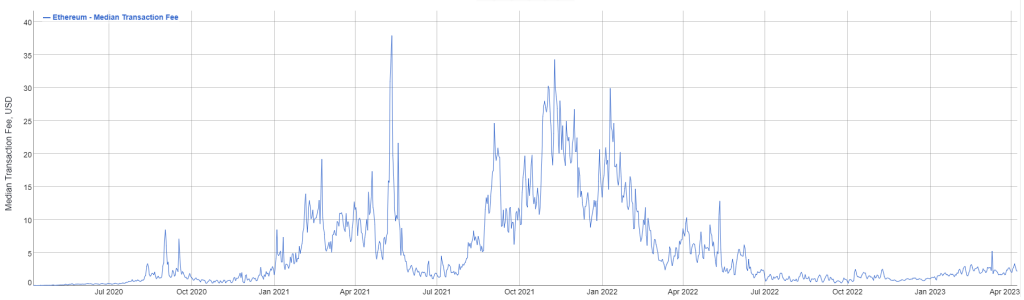


Figure 14: Ethreum Median Transaction Fee Chart

Source: `https://bitinfocharts.com/ethereum/`

3.4.1 Layer-2 Transaction Cost Analysis

As shown in the Figure 15 cost of a single Rollup transaction shall be break down into two main components, fixed costs and variable costs.

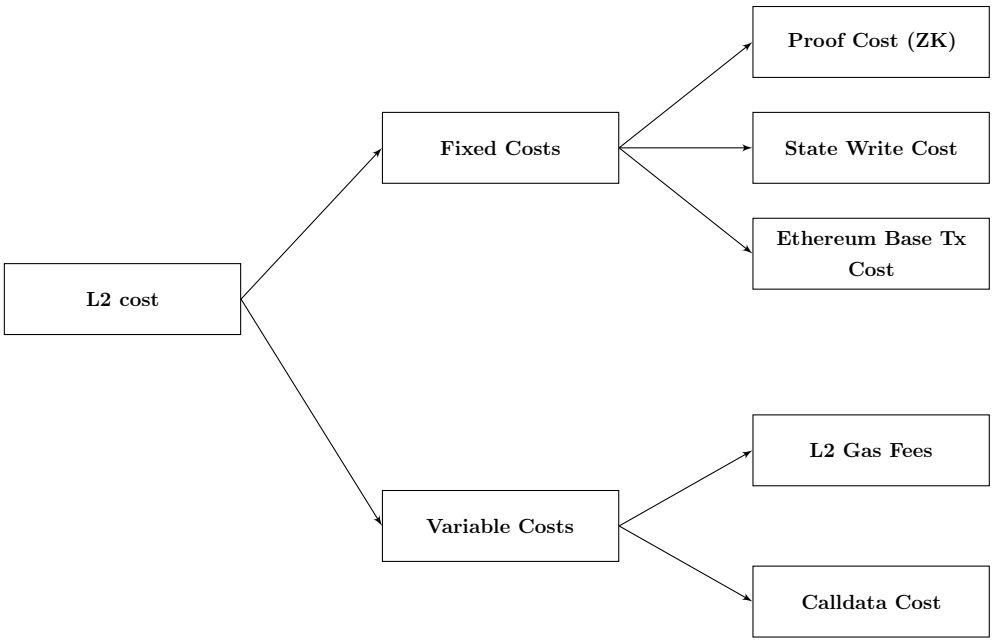


Figure 15: Anatomy of a Layer-2 Transaction Cost [97]

Fixed Costs

Fixed costs are gas fees which are deterministic in the context of a "Rollup block". Fixed costs consist of state write cost, Ethereum base transaction cost and proof cost (in Zero-Knowledge Rollups).

– **State write cost** The most prominent fixed cost is state write cost. It occurs when a Rollup writes data in the bridge contract. A Rollup may have to store several properties to secure integrity of the system and facilitate fraud proofs. It includes pre/post state roots, transaction roots and other variables for system management etc. These Rollup properties are stored in the bridge contract as "state variables".

  State variable storing cost is one of the most expensive opcode gas cost in Ethereum as they use the "storage" in EVM memory layout. After the Berlin upgrade state write costs were updated as shown below in the Table 3. This is reflected in G_{sstore} and $G_{coldsload}$ in the fee schedule (see Table 4) and costs 22,100 gas per store.

OPCODE	Before Berlin	After Berlin	
		Not Accessed	Accessed
SSTORE (zero to non-zero)	20000	22100	20000
SSTORE (non-zero to non-zero)	5000	5000	2900

Table 3: SSTORE Fee Schedule in EVM After Berlin Upgrade [98]

– **Ethereum base transaction cost** Every transaction in Ethereum has to pay a base fee ($G_{transaction}$) of 21,000 gas.

– **Proof cost** In Zero-Knowledge Rollups, sequencers submit blocks with a cryptographic proof. This proof size vary with the proofing mechanism (See Table 5). For example, ZK-Rollups pay circa 500,000 gas to verify a single ZK-SNARK proof on Ethereum, with ZK-STARKs requiring even higher fees [99].

Variable Costs

Variable costs of a Rollup transaction are consist of L2 gas fees and Calldata cost.

– **L2 gas fees** The L2 gas fees reflect the execution and storage fees in Layer-2. Every Layer-2 transaction will pay some execution fee. As shown in the Equation 1 this equals to the amount of Layer-2 gas ($l2\_gas\_used$) used by the transaction multiplied by the layer-2 gas price ($l2\_gas\_price$) attached to the transaction. This is exactly how fees work on the Ethereum network.

Name	Value	Description
$G_{coldsload}$	2100	Cost of a cold storage access.
$G_{transaction}$	21000	Paid for every transaction.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	16	Paid for every non-zero byte of data or code for a transaction.
$G_{keccak256}$	30	Paid for each KECCAK256 operation.
$G_{keccak256word}$	6	Paid for each word (rounded up) for input data to a KECCAK256 operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	2900	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.

Table 4: Gas consumption metrics associated with Rollup related computations (Fee Schedule - Berlin Version)

Proof system	Proof Size
Buletproof	$O(logM)$
Ligero (zk-SNARK)	$O(\sqrt{\lvert C\rvert})$
zk-STARK	$O((\log \lvert a\rvert)^2)$
Sonic (zk-SNARK)	O(1)

Table 5: Zero-Knowledge Proof Size Comparison [100]

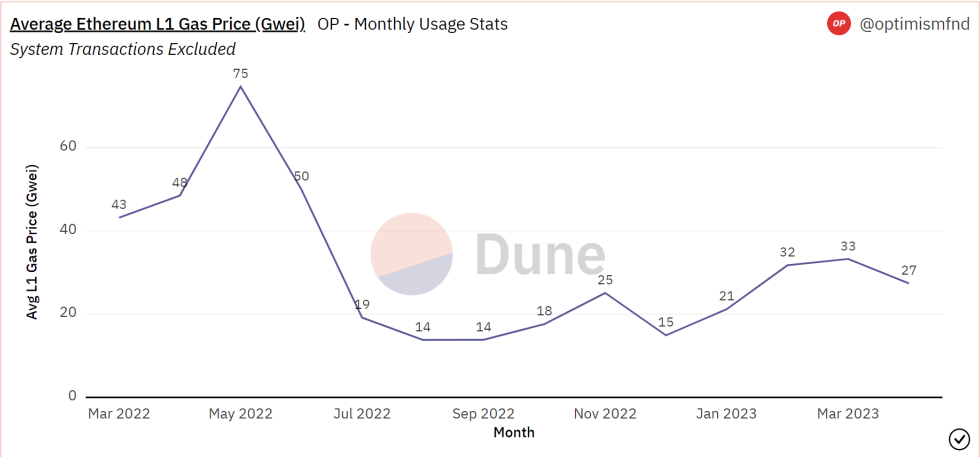


Figure 16: Average Ethereum Layer-1 Gas Price

Source: `https://dune.com/optimismfnd/Optimism`

But here the L2 gas prices (Figure 17) are extremely lower than Ethereum fees (Figure 16).

– **Calldata cost/ L1 Security cost** Calldata cost (sometimes referred to as L1 security cost) is the gas fees for posting transaction data in Layer-1. This is crucial to the security properties of the Rollup as it guarantees the data-availability via Ethereum. Therefore users have to pay for this data-posting costs in Ethereum. Calldata cost typically dominates the total cost of a transaction in Rollups. This fee is directly influenced by the Ethereum gas market ($l1\_gas\_price$).

$$l2\_gas\_cost = l2\_gas\_price \times l2\_gas\_used \tag{1}$$

$$calldata\_cost = l1\_gas\_price \times G_{calldata} \tag{2}$$

$$G_{calldata} = G_{calldatazero} + G_{calldatanonzero} \tag{3}$$

$$G_{calldatazero} = G_{txdatazero} \times N_{txdatazero} \tag{4}$$

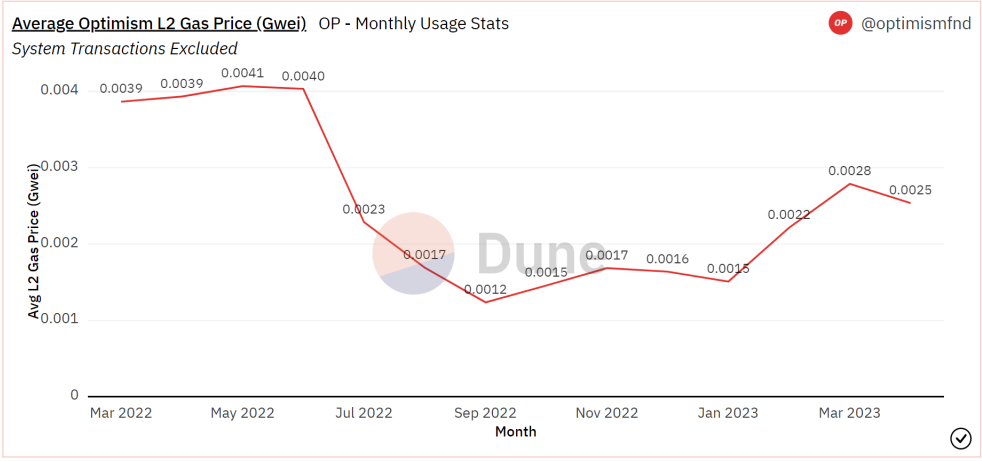


Figure 17: Average Layer-2 Gas Price in Optimism

Source: `https://dune.com/optimismfnd/Optimism`

$$G_{calldatanonzero} = G_{txdatanonzero} \times N_{txdatanonzero} \tag{5}$$

Usually, the fixed costs of the Layer-2 transaction do not change much on a transaction to transaction basis. What can have a magnitude of an impact on a per transaction basis is the price of posting call data to Ethereum. Equation 2 and 3 summarizes the L1 Calldata gas fee calculation. Calldata cost ($calldata\_cost$) is evaluated as the product of Ethereum gas price ($l1\_gas\_price$) and L1 Calldata gas cost ($G_{calldata}$).

L1 Calldata gas cost is derived as a sum of zero byte Calldata gas ($G_{calldatazero}$) and non-zero byte Calldata gas ($G_{calldatanonzero}$). Zero byte Calldata gas cost (in Equation 4) is determined by multiplying gas cost for posting a zero byte of data ($G_{txdatazero} = 4\ gas\ per\ byte$) by number of zero bytes in Calldata ($N_{txdatazero}$). likewise non-zero byte Calldata gas cost is calculated in Equation 5.

3.4.2 Impact of Transaction Batch Size

Rollup fixed costs are attached to a batch rather than a single Layer-2 transaction. Therefore user transaction fees are amortized over the size of the sequenced batch. The batch size increases when more transactions are sequenced into it. Accordingly fixed cost is dispersed over the number of transactions in the batch. As shown in figure 19, fixed cost per L2 transaction is reduced when more transactions are added

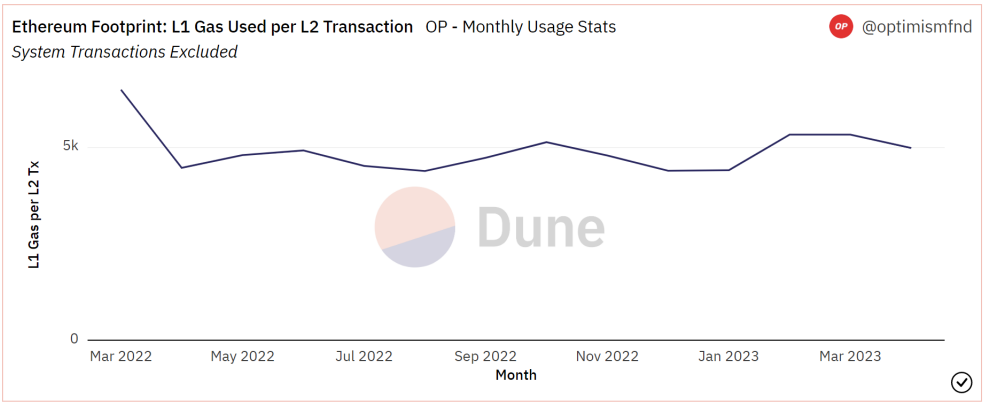


Figure 18: Layer-1 Gas Used per Layer-2 Transaction in Optimism

Source: `https://dune.com/optimismfnd/Optimism`

in the transaction batch.

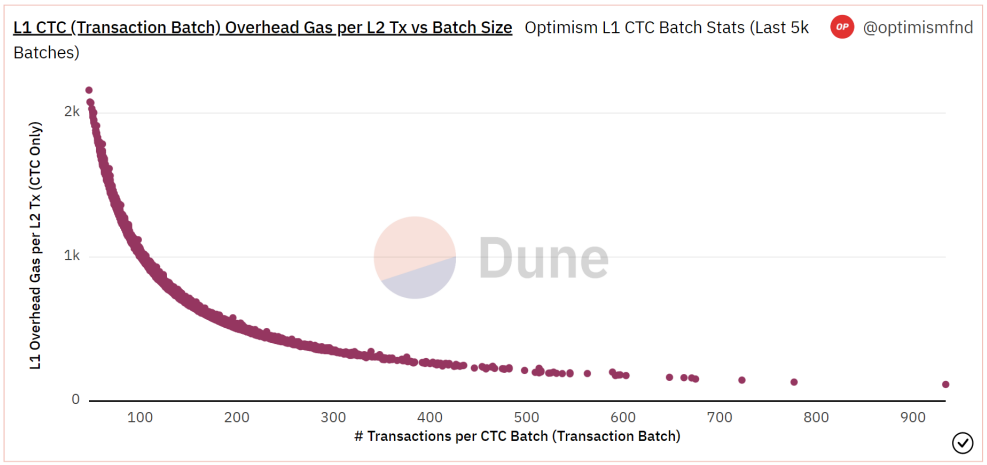


Figure 19: Fixed Cost vs. Batch Size

Source: `https://dune.com/optimismfnd/Optimism`

3.4.3 Transaction Savings from Layer-1 Security Costs

At the time of writing, Optimistic Rollup (Arbitrum and Optimism) transaction fees are in the range of $0.0001 \sim 0.0002$ ETH. Table 6 breaks down the current fee market in Layer-2 rollup projects. Based on the market prices, rollups on average tends to be 90% lower than the Ethereum. This margin is expected to fluctuate with

Project	send ETH fees ($)	Token swap fees ($)
Loopring	0.03	0.48
zkSync Lite	0.05	0.12
Arbitrum One	0.06	0.18
Polygon zkEVM	0.11	0.44
Boba Network	0.14	0.31
Optimism	0.15	0.22
StarkNet	0.19	0.48
Ploygon Hermez	0.25	-
Ethereum	0.81	4.04

Table 6: Transaction fee comparison in major Layer-2 Projects (April 2023)

Source: `https://l2fees.info/`

mainnet congestion.

Figure 20 highlights percentage transaction savings from Ethereum in Optimism and Arbitrum. Both rollups demonstrate almost 95% of transaction saving when compared to barely using Ethereum. In terms of rollup macroeconomics, Optimism's all time USD saving on Layer-1 gas fees rallies up to astronomical amount of 3.18$. This gas fee saving in Optimism is illustrated in Figure 21.

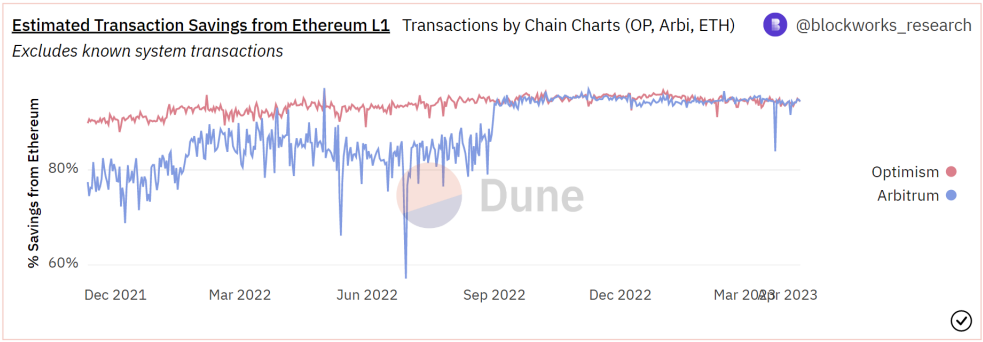


Figure 20: Estimated Transaction Savings from Layer-1

Source: `https://dune.com/blockworks_research/l2-comparison-dashboard`

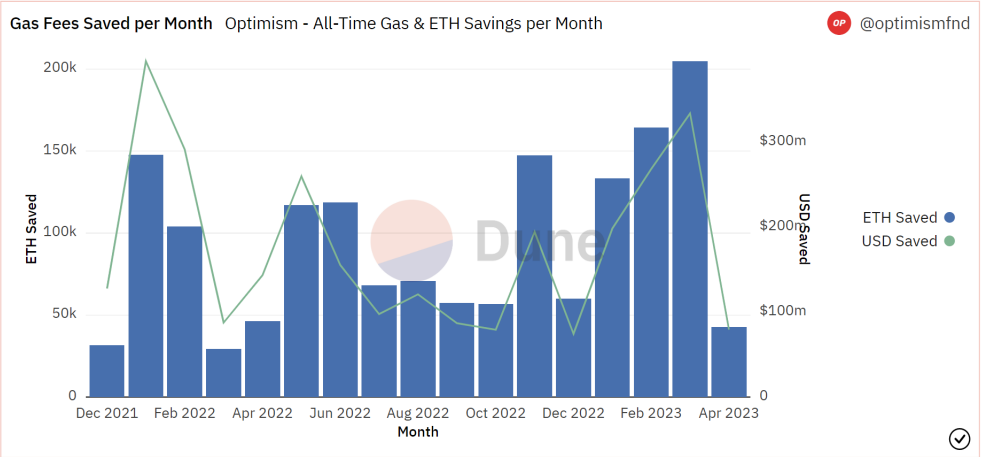


Figure 21: All-Time Gas Savings in Optimism

Source: `https://dune.com/optimismfnd/optimism-all-time-gas-fee-savings`

3.5 Challenges

In this subsection we discuss the challenges mainly encountered by Optimistic Rollups. Currently, Rollups in general operate under the control of a centralized sequencer, which poses a significant challenge. The next pivotal objective for Rollups is to establish robust decentralization within their protocols.

To address this, we explore various decentralization strategies presenting the cutting-edge approaches. Furthermore, we discuss two key challenges particular to Optimistic Rollups, namely fast withdrawals and verifier's dilemma.

3.5.1 Decentralization of Sequencer

Decentralization is a key principle of blockchain technology that involves removing central intermediaries and enabling a network of nodes to maintain and validate the blockchain. However, in the current state of Rollups, centralized sequencers are used to batch together off-chain transactions before submitting them to the blockchain for verification. This centralization means that the network is reliant on a single entity, which can be a point of failure or vulnerability.

Centralized sequencing with white-listed validation are the current state of affairs of Rollups. As a result, it is far from core principles of blockchain technology like decentralization and minimum assumption of trust. Centralized Rollups comes with advantages of efficiency on bug fixing, fast upgrades and reduced latency on

transaction execution. In contrary, decentralization adds up considerable overhead on protocol to reach consensus. Therefore the major challenge, Rollup face today is achieving decentralization without compromising the user experience.

The path for fully decentralization of Rollups mainly relies on two properties, degree of ownership of protocol upgradeability and decentralization of sequencer. As we discussed in section 5.2.2, protocol upgradeability shall be democratized through a proper implementation of a permissionless DAO. Current state of Rollups are not fully achieving the concept but progressively driving towards it.

Consequently the key bottleneck remains with the sequencer decentralization. The first known introduction on decentralization strategies were discussed in [52], where the author suggests a variety of possibilities for batch submission. It further refined in [101] dividing the role into sequencer appointment and transaction ordering policies. Below we list down several cutting-edge concepts for sequencer decentralization.

- **Based Sequencing** Based sequencing [102] is a sequencing strategy which utilizes the underlying Layer-1 for sequencing. It reaches the full liveness and decentralization of the Layer-1 but at the cost of throughput which is fundamentally limited by the transaction ordering and consensus of Ethereum.

- **Permissionless Leader Election** Traditional BFT and PoS consensus protocols already employs different strategies for block proposers. This can be further utilized for sequencer selection. Weighted round-robin algorithm used in tendermint [103] and random leader election in Gasper [104] utilizing RANDAO [105] are battle-tested examples.

  In [52] author points out staking as a drawback of PoS based leader election mechanisms as it leads to large amount of collaterals. EigenLayer [106] is a novel protocol with the concept of re-staking where stakers can impose additional slashing conditions on their staked ETH. This positions EigenLayer a pragmatic solution, as decentralization can be achieved via a quorum of ETH stakers who are willing to stake their ETH for the role of sequencing. The same idea of reusing stakes and validator set in Interchain Security in Cosmos [107] is presented in [108] which proposes a common sequencer network which could act like "sequencer as a service" for a set of Rollups.

  An Epoch based public lottery protocol is proposed in StarkNet's decentralization protocol [109] for leader election. Here time is divided into epochs which are further subdivided into slots. Protocol appoints a leader for each slot based on the stake distribution and randomness. A similar proposal [110] introduces

a multi-slot sequencer selection with a governance adjusted time factor which takes into account the reputation of sequencers. This protocol utilizes the honesty of available sequencers encouraging newcomers via proper alignment of incentives. Espresso [111] is another decentralized sequencer architecture whose promise to deliver Web2 performance with Web3 security. It is built on HotStuff protocol [112] which can reach throughput of 10,000 - 20,000 ERC-20 token transfers under good network conditions.

– **MEV auctions** Apart from leader election techniques based on governance and randomness, sequencer shall be elected via MEV auctioning [113] [114]. Here the highest bidder receives the slot for sequencing with the right to extract MEV from the corresponding block.

3.5.2 Verifier's Dilemma

Optimistic Rollups fundamentally assume of a one verifier who puts effort to achieve system incentives. But we can not always assume verifier will act accordingly all the time. This is called verifier's dilemma [115].

Verifier's dilemma occurs when sequencer always act honestly. The verification process incentivize verifiers to secure sequencer's collateral fully or partially, if he can successfully prove fraudulent submissions from the sequencer. But if the sequencer decides to not to act maliciously, verifiers are not able to gain anything from performing the verification. This leads verifiers to withdraw from the network, permitting a secure environment for sequencer attacks.

There are mainly two arguments on the verifier's dilemma in the literature, one argument is, it is critical to the security of the Rollups, hence should be treated accordingly[52]. The other states we should not consider it at all[116]. However, according to our knowledge, none of the Rollup implementations except Arbitrum [117] have deeply discussed about this topic.

The verifier's dilemma is a fairly complex problem, and solving it seems quite difficult. A mathematically aggressive game theory based explanation in [118] discusses on different strategies of sequencer and verifier on verification process in Optimistic Rollups. It shows through Nash Equilibrium, if sequencer attacks with a lower probability, it is assured that there is no verifier active to validate. And it further demonstrates increasing the number of verifiers, moreover discourage verifiers as it reduces the expected verification reward.

As a solution it mathematically proves Attention Challenge method [119] resolves

the dilemma by slashing verifiers who do not participate in the verification process. The mechanism motivates verifiers not by increasing the reward but by making it inevitable and inflicting considerable losses if not performed.

However it is not the most suitable solution as when sequencer does not make any infringement, there is always a loss for the verifier. This ends up in another game theory problem having trade-offs of operation at loss until the next violation of Rollup state.

3.5.3 Fast Withdrawals

One of the main drawback in the Optimistic Rollups is its long withdrawal time. When a user wants to withdraw tokens back in the Layer-1, it needs 7 days of time span for funds to be available on the main-chain aka Ethereum [120]. This constraining 7 days of period is the so-called "Challenge Period" in Optimistic Rollups.

The length of the challenge period is a critical design decision in Rollup architecture. It strongly connects with the overall security of the system and efficiency. A longer challenge period replicates increased security in the Rollup as it makes enough time for challenges despite congestion or malicious attacks. But it results in longer waiting times for batch finality resulting in poor user experience.

The current industry accepted 7-day challenge period is considered long enough to withstand any attack in the system. It makes attacks on Rollups extremely expensive for malicious parties. A plausible rationale on the 7-day challenge period is given in [121]. It mathematically shows one week time period is the optimal challenge period to make attacks economically infeasible.

The challenge Rollups face today is how to enhance the usability of Rollups while maintaining high security standards. We find two primary approaches in community to address this issue.

- **Fast withdrawals through liquidity providers** Layer-2 Bridge protocols such as HOP [122], Connext [123], Rubicon [124] with the use of Automated Market Makers (AMMs) provides a pricing mechanism for liquidity for fast withdrawals. When the user is willing to withdraw tokens/ETH before the hard finality (7-days), he could use liquidity providers for instant withdrawal under a fee [116].

  Current Rollups are already utilizing this methodology improving user experience allowing a competitive market [125] [126]. This works satisfactorily for fungible tokens with smaller token swaps. But it could be problematic for pro-

fessional traders as it would constitute a significant cost when they move huge volumes of liquidity back to Layer-1. Another drawback of liquidity provider solution is that they are only compatible with fungible tokens. Non-Fungible-Token (NFT) withdrawals can not be accelerated as they are unique and not interchangeable [127].

– **Dynamic challenge periods** In contrast to the fixed challenge periods, there are some proposals published on reducing the challenge period without compromising the security. A censorship oracle is intorduced in [128], which can indicate the Rollup protocol if it can reduce the challenge period based on the past activity in the underlying blockchain. It outputs a binary signal with high confidence, if there has not been censorship over the last N blocks or not. The rationale of this work is, if there are missing blocks in Ethereum chain, that could be a resultant of forking censorship threatening the safety of the Rollup. Accordingly, challenge period is not altered. This approach has a higher abstraction on the view of Rollup security through the block inspection in Layer-1.

  A more granular strategy is described in [129] considering the reputation of the sequencer and aggregated batch value. This proposal studies the time required to balance the aggregated batch value under a spamming attack by a malicious sequencer in the Layer-1 blockchain. If this time is less than the current challenging period, it is rational to consider a lower challenging period as it is not economically feasible for a malicious party to conduct such attack more than that time window. But it should not be lower than the minimum challenge period required for a challenger to submit a fraud proof. This minimum time period required to post a successful challenge is constrained by the network congestion and consensus failures in the Layer-1 chain. Through empirical results on Ethereum consensus failures, it suggests challenge period can be reduced to 23 hours while ensuring top-notch security of Rollups.

4 Implementation

In this chapter we present an architecture and develop a prototype of an Optimistic Rollup system for simple payments including a fraud proof mechanism. In the first section we investigate the architectural reasoning behind the inception of the Rollup paradigm. Then in the following sections we introduce system components of our Optimistic Rollup with a detailed description on the implementation.

4.1 Layer 2: A Paradigm Shift in Blockchain Architectures

Since its inception, blockchain architects have been facing the challenge of creating the optimal system. The design of a blockchain relies on some core properties, such as consensus, guaranteed security, data availability and transaction execution. Historically all popular blockchain designs such as Bitcoin and Ethereum, handled these tasks on a single layer, hence are called "monolithic blockchains".

Monolithic blockchains offer great benefits such as high security, transparency and data availability. In these blockchains, protocols reach consensus by re-executing transactions by all nodes in the network except the miner. Additionally it has introduced barriers on ordinary users, such as requirement of high capacity hardware equipment leading to a more centralized domain.

In order to attain a high degree of decentralization and minimize operational costs, monolithic blockchain systems have traditionally limited block times and block sizes. However, this approach has resulted in significantly low throughputs, leading to a major obstacle for scalability.

4.1.1 The Modular Blockchain Architecture

With the increased popularity of blockchain systems, monolithic blockchains has come to a point it can no-longer facilitate mass adoption due to its inability to scale and weaker performance. Solution to these challenges progressed to a new blockchain paradigm called "Modular Blockchains".

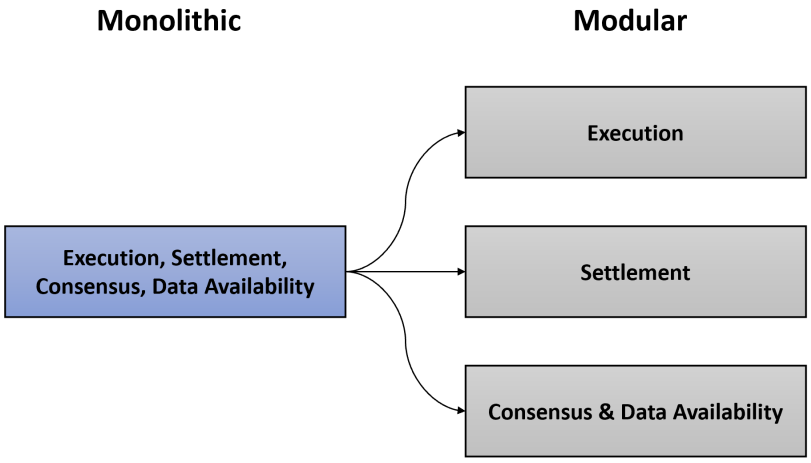


Figure 22: Monolithic vs Modular Blockchain Architectures

Source: celestia.org

Modular blockchain is based on an approach of a modular design. The basic idea is separating the tasks of a monolithic blockchain into several layers making a more flexible and robust system. The four main functions of a modular blockchain are execution, settlement, consensus and data availability. The data availability layer ensures transparency and accessibility of data, the settlement layer handles transactions and their finality, the consensus layer maintains the integrity of the network through a consensus mechanism, and the execution layer executes smart contracts and other applications on the blockchain. A single layer in a modular system may provide more than one functionality. For example, a modular blockchain that specializes in data availability also is required to combine consensus for ordering.

An optimized modular blockchain is a system where these functionalities are decoupled across specialized independent layers. The basic structure of a modular blockchain is shown in the figure 22. Here the execution layer sits at the top of the stack while a settlement layer bridging it with the consensus layer. The execution layer may publish its full blocks to the settlement layer, which may build its own blocks using one or more execution layer blocks and publish to the consensus layer.

Consensus and data availability layer provides the consensus over the transaction ordering and verifies the data availability.

Benefits of Modular Architecture

The "Blockchain Trilemma"[3] on decentralization, security and scalability shows that it can never achieve all aspects without making weak assumptions on one property. Optimizing on certain two qualities would leads to a trade-off in the third. However modular blockchains allow to scale systems without introducing any trust assumptions.

Modular designs introduce greater flexibility for blockchain operations. A modular blockchain may consist of separate chains that focus on settlement and data availability, while another on the execution. A consensus and data availability layer may benefit from scalability since transactions are processed in a separate layer. It only needs to enforce the validity of off-chain execution and guarantee the availability of off-chain data. An execution layer may benefit from extra security by leveraging the parent chain. This is how Layer-2s benefit from Ethereum's decentralization.

Drawbacks of Modular Architecture

Unlike the monoliths, modular blockchains can not guarantee on security. If the consensus and data availability layer is not effectively handling the security, modular chain is at risk. Therefore in general, modular blockchains use a monolithic as consensus and data availability layer. Another disadvantage of modular architecture is its incurred complexity. For example, Layer-2 systems need to have complex mechanisms such as fraud proofs and validity proofs to insure the correctness in execution layer. Architecting a secure and efficient modular blockchain requires a higher level of skills and scrutiny on the design process.

4.1.2 The Rollup Architecture

Recently, several new construction approaches have emerged for Rollup architectures. We identified the schematics of these concepts in the previous chapter 3. In this work we are following what is arguably the only implemented version of Rollups, namely "Smart Contract Rollups" (SCRs). In the coming chapters, Unless otherwise specified, we will refer to Smart Contract Rollups simply as "Rollups".

Since their conceptual inception, Smart Contract Rollups have become the industry standard. The SC Rollup design follow a basic modular approach compared to monoliths. They are basically consists of two layers, an execution layer and the parentchain. As shown in the figure 23 Layer-1 (aka Ethereum) offer security and data availability while Layer-2 provides the additional scalability through off-chain execution.

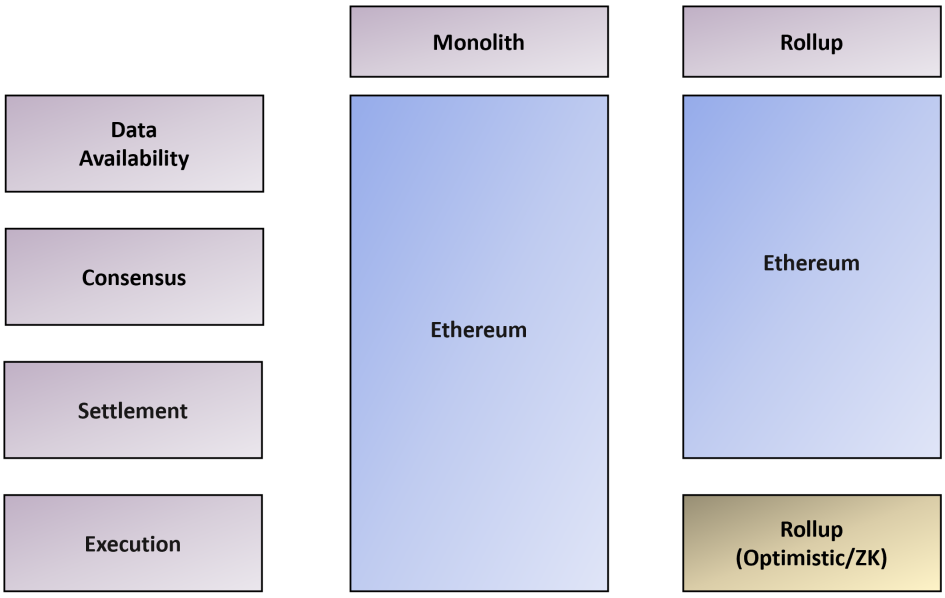


Figure 23: Difference of Rollup and Ethereum

Source: celestia.org

In this work we follow the same architecture, maintaining the two layer blueprint. The top-level system architecture of our Optimistic Rollup is shown in the figure 24 and a simplified sequence diagram of the Rollup execution is presented in 25. The sequencer and verifier are the two main components in the Layer-2. They are responsible for the secure transaction execution and verification. The bridge contract in Layer-1 cater the consensus and settlement to the process.

In the following subsections we present our approach on the main requirements of a Rollup architecture, namely data availability, settlement scheme and censorship resistance.

Data Availability

The sequencer batch up the transaction data and publish them in Layer-1 as "Calldata". This provides the data availability enabling all participants to rebuilt the state. Any user can re-execute the Rollup's state and verify the correctness of state transitions.

SCRs are technically described as "hybrid scaling solutions" because they derive the security properties such as data availability through a monolithic blockchain. Without data availability, verifiers can not construct fraud proofs to challenge invalid

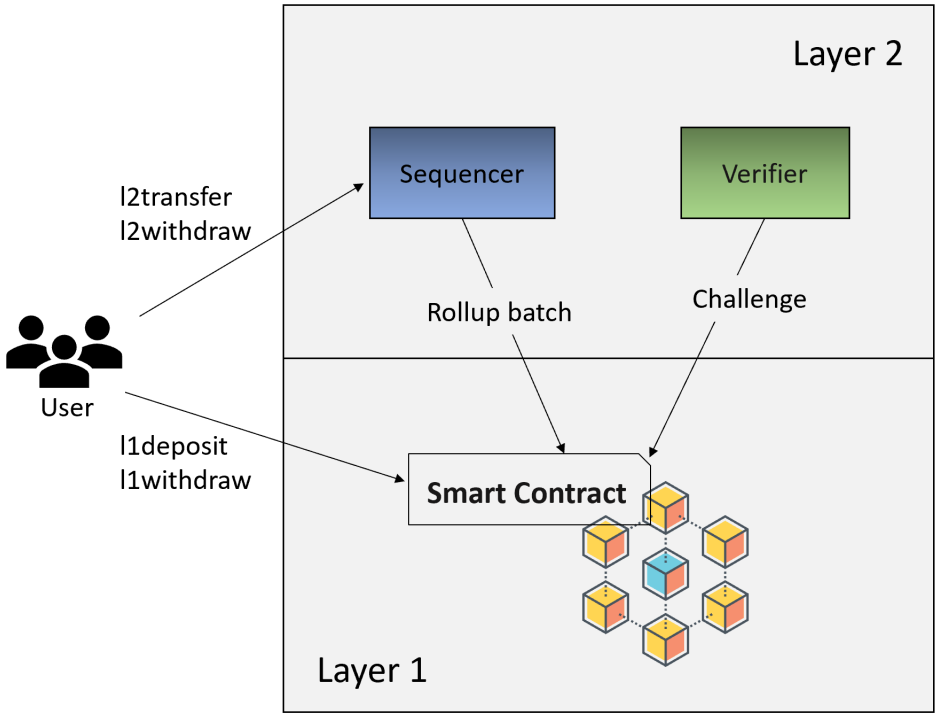


Figure 24: Rollup System Architecture

blocks by sequencers. Having Ethereum as the base layer, SC Rollup systems can avoid malicious behaviors without risking the system integrity.

Settlement

Settlement layer is again facilitated through the Ethereum network. It establishes security and finality to the Rollup canonical chain in case of a dispute. When the sequencer is malicious, a verifier can challenge the fraudulent batch by directly calling the bridge contract where the validity is decided.

Censorship Resistance

In an Optimistic Rollup, a centralized operator gets the responsibility of aggregation and submission of transactions. This introduces an undesired trust to the system as Rollup operators can censor users from participation.

A malicious sequencer could prevent users by going offline completely or by refusing to include transactions from certain users. Alternatively, they can prevent users from

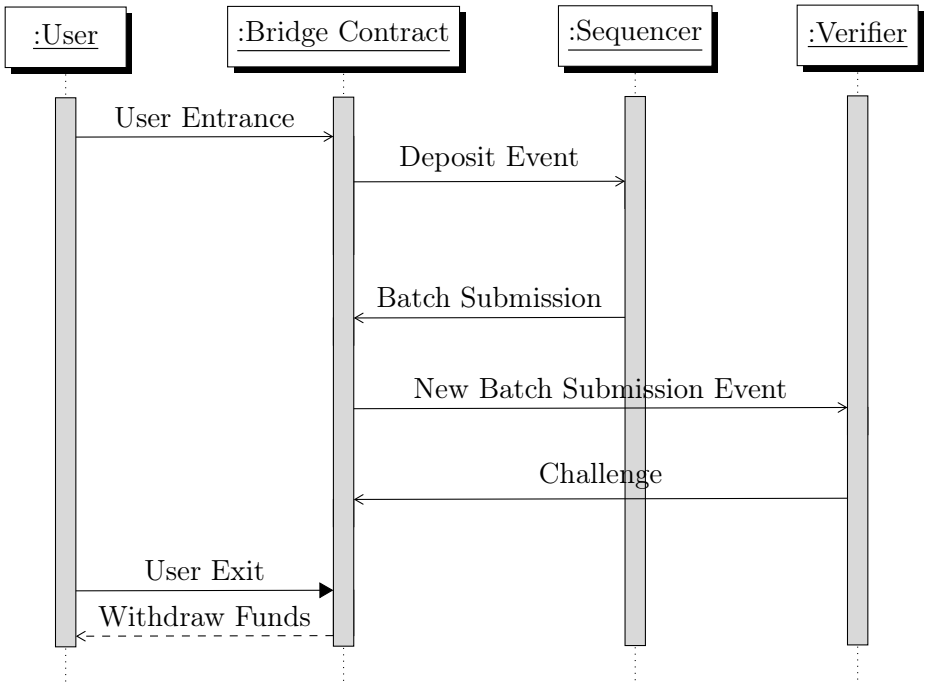


Figure 25: A Simplified Sequence Diagram of the Complete Rollup Execution

withdrawing funds back to Layer-1 by withholding the state data necessary to prove transaction inclusion in the batches.

Optimistic Rollup designs address the first issue by enabling users to submit their transactions directly on Layer-1, rather than going through the sequencer and potentially being censored. Here the sequencer is forced to include these transactions given a delay. But in this work we are not implementing such a mechanism assuming an honest behavior from the sequencer on user censorship. The latter is prevented by making sequencers to publish the data associated with the world state changes. If a sequencer goes offline or stops operating, another node can take up the responsibility by reproducing the world state through the published data. In the same way a user can generate proofs for their withdrawal transactions by generating merkle trees.

However our work only implement a single "trusted" sequencer where we make a guarantee on continuous block production without user censorship and users are always provided with Merkle proofs (Proof of Inclusion) for their withdrawals in Layer-1.

4.2 Transactions

Transactions are cryptographically signed messages which are recorded in a blockchain. An Ethereum transaction refers to an action initiated by an externally-owned account (EOA). For example if Alice sends Bob 1 ETH, Alice's account is debited and Bob's must be credited. This is called the "state change" in the blockchain. A transaction carries the necessary instructions to carry out such a state change.

There are two types of accounts in Ethereum, Externally-Owned Accounts and Contract Accounts. Externally-Owned Accounts (EOAs) are the accounts controlled by human users through public and private keys. The public key is the identifier of the account. Transactions are signed by the private key to prove ownership of the EOA. An EOA can be thought of as an individual's bank account that can be used to send funds using password verification. Contract Accounts (CAs) are the accounts containing code and identified by a public key. The code is commonly referred to as a smart contract and is an automated program that runs when it receives a transaction from another EOA or contract account.

4.2.1 Type of Transactions

In a blockchain there could be three ways a transaction could occur, between two externally-owned accounts, between two contract accounts and lastly between an EOA and a contract account. Payment transactions between users accounts are always between EOAs. Since our design focus only on simple payments, we captalize on the transactions between EOAs.

Following a basic payment system, our implementation introduces three transaction types to accomplish fund transfers between users. They are namely, deposit transactions (l1deposit), Layer-2 user transactions (l2transfer) and withdrawal transactions (l1/l2withdraw). We support two Layer-1 transactions and two Layer-2 transactions to realize above three transfer types. Layer-1 transfers consist of "l1deposit" and "l1withdraw", which resemble the user entry and exit of Rollup system. The Layer-2 transactions are "l2transfer" which is the user to user payment transaction in Rollup and "l2withdraw" which allows a user to stage a Layer-2 withdraw before he withdraw in Layer-1.

Deposit Transaction - "l1deposit"

The deposit transaction is the entry point to the rollup system (Figure 26). When a user wants to enter the Layer-2 Rollup, he has to deposit some Ether in the Layer-1

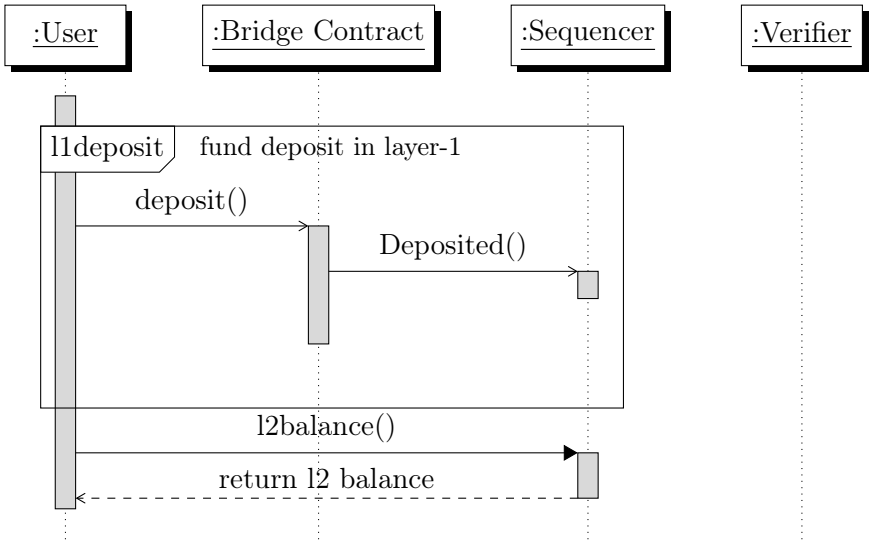


Figure 26: The Deposit Transaction - User Entrance to the Rollup System

bridge contract. Then this deposit (with a deduction for the Layer-2 costs) will be available in the Layer-2 for use in few minutes.

The deposits are resembled in the user wallet as "l1deposit". A deposit transfer triggers an event in the bridge contract which then informs the sequencer on the locked funds. Users are expected to wait until the sequencer captures the event from the Ethereum. This delay would vary based on the mainnet congestion.

Layer-2 User Transaction - "l2transfer"

Layer-2 user to user transfers or "l2transfer"s enable users to transfer funds without any delays related to Layer-1 block minting and costs.

Once the transaction is initiated, the receiver will receive funds in few minutes in Layer-2 as sequencer updates the world state of the Rollup. In general, users may consider transaction "soft-finality" as soon as an honest sequencer submits the batch to the Ethereum, where in which transaction status changes to "ACCEPTED_IN_L1". But the transaction "hard-finality" is achieved when the batch matures after the challenge period.

Withdrawal Transactions - "l2withdraw", "l1withdraw"

The withdrawal is the most critical transaction in a Rollup design. In case of an Optimistic Rollup, it is generally considered as a system drawback, since it takes at

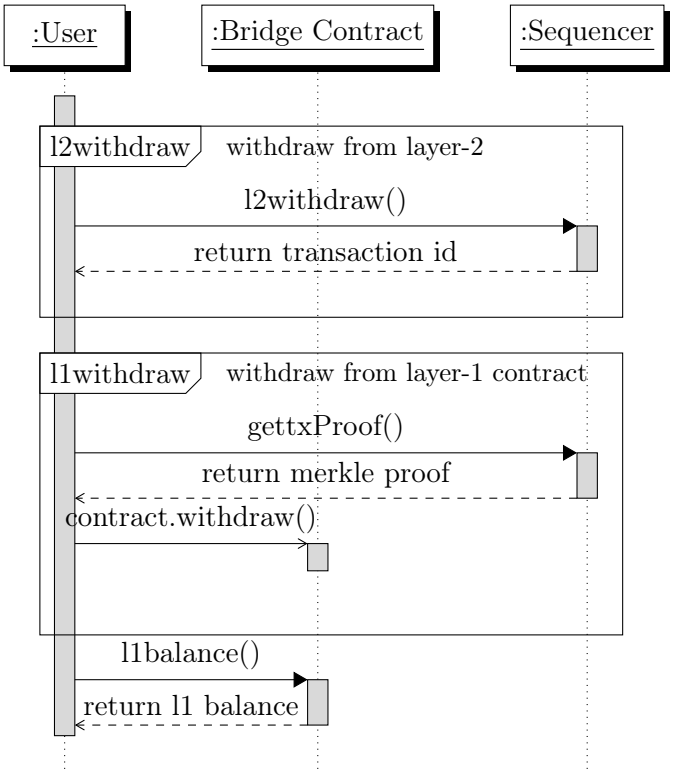


Figure 27: The Withdrawal transactions - User Exit from the Rollup

least 7 days to withdraw funds back in Layer-1 securely.

Withdrawals happens in two steps in our protocol. First a user has to make an Laye-2 withdrawal (l2withdraw) where it updates the world state on the Rollup. As this transaction gets finalized in Layer-1, user can trigger an Layer-1 withdrawal (l1withdraw) to secure funds in Ethereum. During an "l1withdraw", user wallet first requests a merkle proof (Proof of Inclusion, See 4.5.3) from the sequencer for the initial "l2withdraw". Then it submit this proof to the bridge contract which validates the transaction inclusion and eligibility.

Here we make an assumption of an honest sequencer, who will always provide Merkle proofs upon user requests. In case of a malicious sequencer who rejects to produce proofs, users may still recreate the Merkle proofs with the data available in Layer-1. Therefore we still guarantee the safe withdrawal from the system without any obstacles. The complete illustration of the withdrawal procedure is shown in figure 27.

```
{
   id: 2,
   sender: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8,
   target: 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC,
   type: 'l2transfer',
   value: 100000000000000000n,
   nonce: '1',
   timestamp: 1679642993121,
   status: 2,
   batchid: 1,
   errormsg: undefined,
   finality: 1679643934000
}
```

Figure 28: Layer-2 Transaction Format - Type I

```
{
   id: 2,
   type: 'l2transfer',
   timestamp: 1679642993121
}
```

Figure 29: Layer-2 Transaction Format - Type II

4.2.2 Transaction Architecture

A transaction in Ethereum includes many different data fields including sender/reciever addresses, signature, and gasLimit etc. And with newer changes like EIP-1559, it has introduced more fields such as "maxPriorityFeePerGas". In this work we use some of the basic necessary fields as used in Ethereum and introduce new fields to assist our Rollup protocol.

We prepare three different variants of a Rollup transaction. The first variant "Type I", shown in the figure 28 lives in the sequencer mempool throughout its lifecycle until it gets removed. The second variant "Type II" is the concise version of the first, which goes through transaction management cycles such as queuing, re-queuing within the sequencer data structures. Third version "Type III", is the last variant which goes in to the Layer-1 as compressed Calldata.

Layer-2 Transaction Format - Type I

The first version of a transaction that appears in the mempool contains all the necessary data for the user interface. This includes the transaction status shown

as “status”, and any error messages displayed to the user in case the transaction is rejected, known as “errormsg”.

Furthermore, it carries information such as “batchid”, the batch number in which the transaction got included and “finality” the timestamp transaction achieves the hard finality. These information facilitates the sequencer to identify and execute in case of a “re-batching” situation following a malicious batch.

below shows all the information included in a Rollup transaction. The granularity is differed based on the transaction type.

- **id** A unique index to identify the transaction
- **sender** The sender’s address, that will be signing the transaction
- **target** The receiver’s address.
- **type** Transaction type ($deposit, l2transfer, withdraw$)
- **value** Amount of ETH to transfer from the sender to receipt(denominated in WEI)
- **nonce** A sequentially incrementing counter which indicates the transaction number from the account
- **timestamp** The timestamp when transaction get included in the mempool
- **status** Current state of the transaction
- **batchid** Batch index in which transaction gets included
- **errormsg** Error message if transaction is rejected by the sequencer
- **finality** The timestamp on which the transaction is expected to reach hard-finality

Layer-2 Transaction Format - Type II

Second version of the transaction, named as “txref” (in the software architecture) which abbreviates to “transaction reference”, carries enough information for the sequencer to carry out necessary operations (sorting etc.) having a pointer to the full data.

Sequencer will include “txref” in its transaction queue and when it starts sequencing, it will extract the detailed version of the transaction by indexing from the mempool.

```
{
   id: 2,
   sender: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8,
   target: 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC,
   type: 'l2transfer',
   value: 100000000000000000n,
   nonce: '1',
   timestamp: 1679642993121,
}
```

Figure 30: Layer-2 Transaction Format - Type III

Mempool is implemented as a hashmap having a key-value combination with transaction id to transaction data. "texref" carries the transaction id (shown in fig 29), a unique number and this is indexed in mempool to withdraw full data.

Layer-2 Transaction Format - Type III

The final and the third version is the transaction format which goes to the Layer-1. As it directly interacts with Ethereum, it needs to be concise as much as possible to reduce Layer-1 Calldata gas costs. Therefore it carries only the exact information which is required to ensure a deterministic re-execution of transactions.

Transaction States

A transaction goes through several different state transitions from the moment it get initiated by the user until it is either rejected by the system or settled on the Ethereum. Below, we list down all transition states of a Rollup transaction in our protocol.

- **PENDING**: Transaction is received in the sequencer mempool and is waiting to be sequenced.
- **ACCEPTED_IN_L2**: Transaction is validated by the sequencer and waiting to be batched and compressed.
- **ACCEPTED_IN_L1**: Transaction is submitted in Layer-1 successfully. Data is available for verification.
- **FINALIZED**: Transaction is finalized in Layer-1 after a successful challenge period.

– **REJECTED**: Transaction is rejected by the sequencer.

4.2.3 Transaction Lifecycle

The transaction lifecyle includes the chain of states a transaction goes through starting from its creation until finality. The complete lifecycle of a layer-2 transaction (l2transfer) is depicted in figure 31.

The lifecycle starts when a user triggers a Layer-2 transaction through the wallet which then goes to the sequencer mempool via a peer-to-peer connection. Upon reception, sequencer creates a Layer-2 transaction with the received data and set the status as "PENDING".

In the second stage sequencer creates a batch of transactions and starts the validation process. In this phase either a transaction gets "REJECTED" if it fails to satisfy certain requirements (a valid nonce etc.) or become "ACCEPTED_IN_L2".

Once the batch is validated, sequencer starts the post processing phase. In this stage, the batch is encoded and compressed for better performance with respect to the gas costs in Ethereum. If the batch gets submitted in Layer-1 successfully, all the transactions in the batch goes to a new state "ACCEPTED_IN_L1". This is the "soft-finality" stage assuming an honest sequencing condition. Users are expected to accept this soft finality under a risk of re-ordering. In case of a batch submission fail to Layer-1, due to any network error or congestion, transactions will be re-queued back in the sequencer queue.

Once the batch is submitted to Layer-1, verifiers are expected to validate the submitted block within the "challenge period". Currently, most of the available Rollups in mainnet have set this period to 1 week (7 days) [120] [130].

If a verfier recognize a batch as "malicious", he can challenge it submitting a fraud proof. If verfier wins the challenge, state commitment chain will be "rolled back" in Layer-1 and the batch will be re-queued back in the sequencer queue. Therefore all transactions in this batch and all following transactions in later batches submitted in Layer-1 will go again into "PENDING" state.

If the batch lives through the challenge period, transaction will become "FINALIZED" in Ethereum. This is the final state of a transaction reaching the irreversible "hard-finality". Once a transaction has achieved this state users can consume it without any risks.

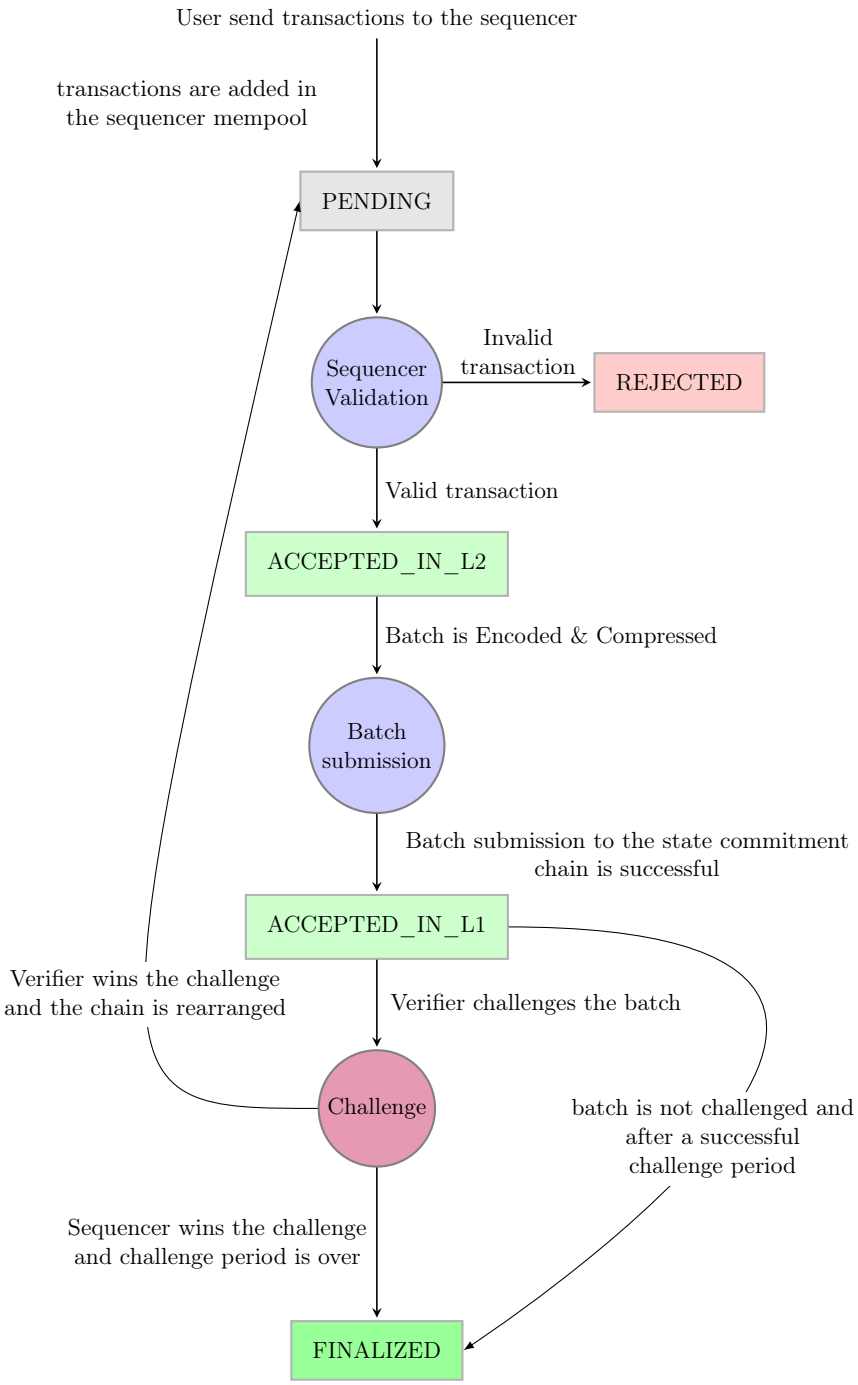


Figure 31: Transaction Lifecycle in Rollup System

4.3 The Sequencer

The sequencer is a single centralized entity whose task is to accept and execute Layer-2 transactions and then submit compressed batches to the Layer-1 blockchain. The responsibilities of a sequencer may differ on the Rollup architecture. Commonly, the sequencer is the gateway for users in Layer-2, sending direct signed transactions.

The sequencer design depends on the trust assumptions in the system. For e.g. Arbitrum's sequencer is trusted on transaction ordering. And it guarantees the sequencer has no capacity to censor any transaction in Layer-2 [131].

In general a sequencer has two ways of getting transactions. First through a direct peer-to-peer connection where users send their signed transactions to the sequencer. But if a malicious sequencer prevent a user from using the network, users are allowed to directly submit transactions to the Layer-1 which will be inserted in the next blocks given a delay.

As stated previously, this work is not performing a mechanism to mitigate censorship resistance. Henceforth we assume an honest behavior on transaction acceptance by the sequencer.

The sequencer is entitled to order and execute transactions following a First-Come-First-Serve policy. It gets transactions in two ways, one via peer-to-peer transactions for Layer-2 transfers and Layer-2 withdrawals and others through Ethereum events for deposits. Periodically sequencer will get these by listening to the Layer-1 events. Layer-1 events may consists of events from deposits and events related to the verification challenge. Both deposits and Layer-2 transactions are added in the transaction mempool of the sequencer.

As shown in the Figure 32 sequencer will start by creating a transaction batch. The number of maximum transactions is a parameter which should be decided on several inputs such as Layer-1 batch size, dispute challenge cost etc. Here we have set the maximum transactions in a batch to 5 considering the ease of demonstration but it can be easily changed for any reasonable value. Valid transactions will be then encoded and compressed. Compressed batch along with other mandatory values (Merkle roots) will be sent to Layer-1.

In a Rollup system we shall assume two ways of sequencer behaviors. The best case scenario is an honest behavior from the sequencer where the chain progresses without any dispute. In the worst case, sequencer compromises the blockchain integrity by behaving maliciously.

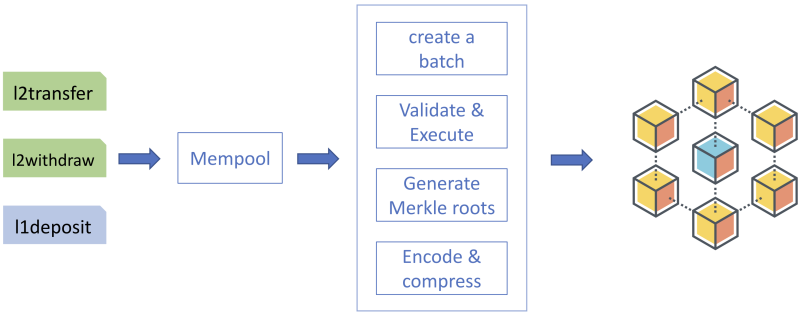


Figure 32: The Execution Structure of The Sequencer

4.3.1 State Root Generation

In Ethereum, the state root is a hash value that represents the current state of the blockchain. It is generated by combining the Merkle tree roots of all the account and storage trie data structures. The Merkle tree is a binary tree where each leaf node represents a hashed value of an account or storage trie. The tree is constructed by recursively hashing the leaf nodes and combining them to form parent nodes until a single root node, known as the state root, is obtained.

Since our Rollup only implements a simple payment system, Merkle tree generation shown in Figure 33 only involves account balance and nonce. Here we customize the Openzepplin's Merkle tree JavaScript package for our needs.

In the original code, the Merkle tree generation process includes sorting the leaf hash values before recursively generating the final root hash. However, to optimize the verification process and minimize gas costs, we made a modification by eliminating the sorting step from the algorithm. By removing the sorting operation, we avoid the need for heavy construction in Solidity in the bridge contract.

4.3.2 An Honest Sequencer

An honest sequencer is a fully functional component which operates with the intent of giving the best user experience by processing transactions in as safe and timely manner possible. We assume of a zero down-time or negligible interference with the operations. As mentioned above, sequencer will receive transactions through two ways. Direct peer-to-peer requests or via Ethereum events.

In the best case scenario sequencer will batch them as they receive based on the First-Come-Fisrt-Serve policy. Once a transaction request or a deposit event

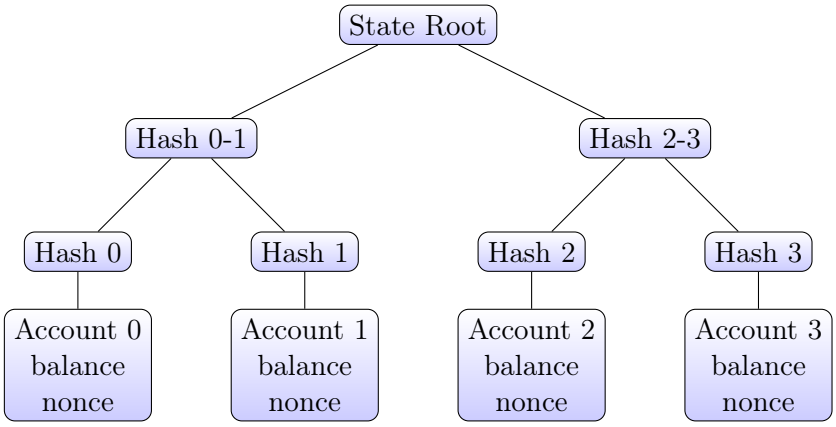


Figure 33: The Merkle Tree Construction for State Root Generation

is received, sequencer first will create a Layer-2 transaction (Figure 28) with the necessary information including a timestamp and a unique id. Then this transaction will be added in the mempool and a concise version of the transaction (Figure 29) is pushed into the transaction queue.

As we are assuming an honest sequencer, he will execute transactions in the received order, generate correct Merkle roots demonstrating the complete world state and then append it to the state commitment chain in Ethereum. The state commitment chain is the data structure in bridge contract which stores the state chain.

Once the transaction batch is submitted in the Layer-1, it enters the challenge period. Over this period verifiers all allowed to challenge the purity of the block. Since our sequencer is acting honestly, no verifier would challenge it. And consequently batch will get finalized in Layer-1 allowing secure fund withdrawals.

4.3.3 A Malicious Sequencer

In a malicious scenario, sequencer would likely be trying to manipulate the world state of the Layer-2 or prevent users from using the blockchain by censoring their transactions. In this work we only try to answer the first case and for the latter, we make an assumption on the honesty from the sequencer on censorship resistance.

Assume a malicious sequencer submits a faulty batch in the Layer-1. Then the verifiers start verification against the batch and corresponding merkle roots (state root, transaction root etc.). Verifier will ask the sequencer for the pre-state-map, which is the world state before the transaction batch is executed. Accordingly, verifier will execute transactions and get into a conclusion on the malicious batch.

Eventually, verifier will challenge the batch which will end up on the verifier's

favor. Here sequencer will loose its fidelity bond and state commitment chain will be rearranged.

4.4 The Verifier

In contrast to the Zero-Knowledge Rollups, Optimistic Rollups assume the "state" submitted by the sequencer is correctly executed unless challenged. Therefore in order to maintain optimal system performance, participants are expected to alarm the malevolence of the sequencer. These participants of the system who are assigned with this task are called "Verifiers".

One of the main assumptions, we made in an Optimistic Rollup system is the availability of a "single verifier" who act for the integrity of the Rollup. As in a Layer-1 (Ethereum etc.) blockchain, they are incentivized for the correct verification of the state commitment chain.

Even though this work is based on a centralized sequencer, our architecture enables multiple verifiers to actively challenge and interact with the Layer-1 bridge contract.

4.4.1 Verification Process

The primary task of a verifier is to listen on the events from Ethereum bridge contract. When the sequencer successfully append a sequenced batch in the state commitment chain, bridge contract will emit an event notifying the new update. Verifiers who are being triggered by this event then retrieve the "Calldata" related to this append call from Ethereum. This extracted data includes compressed transaction batch, pre-state root, post-state root, transaction batch root and the batch index.

The task of the verifier is to attest the post state submitted by the sequencer. This is executed by generating the Merkle state root over the post-world state, which is the resultant of executing the transaction batch upon the previous world state.

Verification process starts with the transaction batch decoding. In the sequencing process, sequencer encode the transaction data and then compress it. The verifier then decompresses the compressed transaction batch and decode before re-execution. After the raw transaction batch is reconstructed, verifier shall start the execution process.

Before the execution, verifier has to validate the precision of the previous state and the transaction batch. Accordingly verifier compute Merkle roots for the previous world state and transaction batch then verify with the values of sequencer submission.

After the accuracy of the transaction root and previous world state root are verified, verifier re-execute the transactions and generate the new post world state root. This state root should match with the retrieved post state from the sequencer. If any of the state roots i.e., previous state, post state or transaction state is mismatched with the calculated roots, verifier challenges the batch by submitting a "fraud proof".

4.5 The Bridge Contract

The bridge contract is responsible to act as the middle layer in a Rollup between Layer-1 and Layer-2. It facilitates user entrance and exit of rollup supporting asset transfers between layers. The integrity and security of the Rollup is held up by the bridge contract.

A bridge contract has several roles in a simple payment system. It should accept deposits from the users and allow users to exit from the Rollup. It should perform as a "*messenger*" between Layer-1 and Layer-2. And finally the bridge contract operate as the final referee in case of a dispute over a fraudulent batch submission.

In the following subsections we describe our approach in accomplishing aforementioned properties of a Rollup bridge contract.

4.5.1 Protocol Assumptions of the Layer-1 Blockchain

In designing a bridge contract, we rely on several assumptions that are made on the underlying blockchain[101].

- **Neutrality** Validators of the underlying blockchain are not colluding with the Rollup operators. Currently, Layer-1 blockchains are fairly decentralized networks, where colluding is extremely difficult. Hence it is correct to assume strong neutrality from Ethereum.

- **Eventual Delivery** A user transaction will be minted in the underlying blockchain given that he pays an appropriate base fee. Ethereum operates on an economic-incentive gas fee mechanism, ensuring that the transaction will eventually be added to an upcoming block.

- **Safety of Smart Contract** A smart contract is considered as a trustless immutable third party. It is a deterministic decentralized program which carries the same security as Ethereum. This enables a trust-minimized bridge creation for Layer-2.

4.5.2 User Entrance and Exit

The user entry to the Rollup is supported by "l1deposit" shown in figure 26. Here bridge contract emits an event indicating a valid fund deposit in Layer-1. User exit is not as trivial as the entrance. It consists of two transactions in Layer-2 and Layer-1 initiated in the given order (see figure 27). The high level view of the mechanism is discussed under Transactions 4.2 section.

The Layer-1 withdrawal transaction, "l1withdraw" is implemented in the bridge contract. Once after the Layer-2 withdrawal is finalized in Layer-1, users can claim their funds initiating a withdrawal request in bridge contract. This is generated through the user wallet. Our Layer-1 withdrawal is based on a proof of inclusion mechanism.

4.5.3 Proof of Inclusion

Proof of inclusion (PoI) is the methodology of verifying users have correctly withdrawn funds in Layer-2 and authorizing bridge contract to release funds back in Layer-1. PoI includes a proof that verifying the target Layer-2 withdrawal transaction is included in the corresponding batch.

If the Merkle root generated by the leaf transaction and proof is equal to the associated transaction root of the batch, the bridge contract can release funds to the receiver.

4.5.4 Authentication Process for Layer-1 Withdrawals

Before executing the Proof of Inclusion, a withdrawal transaction should go through various validation steps. This complete process is illustrated in figure 34.

It should first verify if the Rollup is in a stable state to allow withdrawals. If there is a dispute over any block in the system, withdrawal transactions will be rejected until consensus over the canonical chain.

In the next steps, the system will check the status of the batch to determine whether it has reached hard finality. It will also verify the transaction-level details, such as whether the claimed withdrawal has already been processed and whether the transaction is definitely a withdrawal transaction. Finally, the system will authenticate the proof of inclusion by comparing it against the transaction root.

After all of these checkpoints have passed successfully, the funds will be released to the owner, confirming the validity of the user's address.

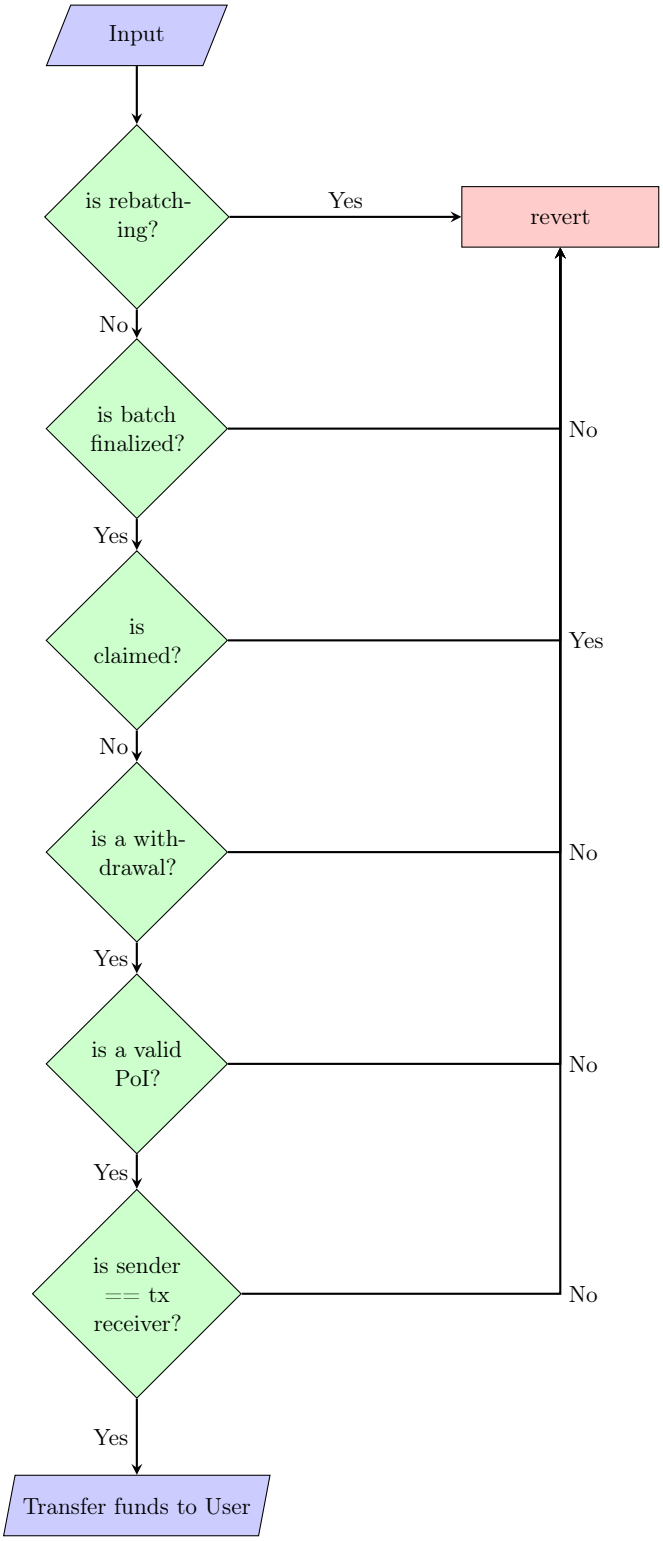


Figure 34: Authentication Process for Layer-1 Withdrawal

```
    struct stateCommitments {
        bytes32 prevStateRoot;
        bytes32 postStateRoot;
        bytes32 txRoot;
        bytes32 txhash;
        uint256 finality;
    }

    // state commitment chain
    stateCommitments[] scc;
```

Listing 4.1: State Commitment Chain

4.5.5 State Commitment Chain

The State Commitment Chain (SCC) is the canonical chain which preserves the world state of the Rollup. This is implemented as an array of structures in Solidity. Each structure contains the previous state root, the post state root, the transaction root, the transaction batch hash, and the timestamp indicating when the block is expected to reach hard finality (see listing 4.1).

Sequencer will append to the SCC by periodically submitting blocks with state roots and transaction data as Calldata. The transaction batch hash is calculated on-chain using keccak256 hash function and this will facilitate to authenticate transaction data when ever a verifier challenges a submitted block.

Additionally, append function will only accept a block if its immediate previous block's post state root matches with the previous state root of the new block, adhering to a strict chain integrity.

4.5.6 Dispute Resolution Protocol

In this work we implement a single round non-interactive dispute resolution protocol. The high level breakdown of the protocol shown in Algorithm 1 shall be decomposed into three main phases.

Firstly, a challenge should fulfill several properties to successfully contest a malicious batch. The challenger should submit enough collateral set by the protocol, the intended malicious batch should not have exceeded the challenge period and submitted transaction batch should resemble the same hash value in the SCC. These checks will be executed in the phase 1 of the challenge protocol.

The second phase is dedicated to data decompression. We use puff decompression

algorithm[132] to decompress data. Here, the data decompression results in UTF-8 format. This is converted back to a hex stream before the Application Binary Interface (ABI) decode.

In the third and final phase, transaction batch is re-executed to examine the correct world state of the batch. This process begins by validating the transaction root, ensuring that the generated transaction root aligns with the corresponding batch in the SCC. Subsequently, the previous state root is examined to confirm its correctness.

Finally the post state root is generated after executing all transactions in the decompressed batch. If an error detected in the process, necessary bridge contract properties, such as "reBatching", "batchId" will be changed appropriately. By setting "reBatching", it will stall all bridge contract calls both from users and sequencer, until the next valid batch is appended. And "batchId" will be updated with the malicious batch id in the context, which is expected to be re-batched in the next batch append call.

Consequently, all batches submitted after the contested batch, are revoked re-ordering the SCC. Then the protocol ends by slashing the malicious party and rewarding the winner. Half of the perpetrator's bond will be slashed and the other half becomes the incentive for the winner.

4.5.7 Rationale for Burning a Percentage of the Fidelity Bond

A percentage of the fidelity bond is burned without rewarding the whole amount as an incentive to the winner. This mechanism discourages malicious sequencers by "frountrunning" the protocol. In an optimistic case for a malicious party, he could challenge his own fraudulent batch in the canonical chain.

This will allow him to win the challenge and not to loss any collateral except the "frountruning" cost. To mitigate this vulnerability, protocol will slash substantial amount (here 50%) of the bond discouraging such attacks by the sequencer[116].

4.5.8 Streamlining State Commitment Chain Maintenance

One limitation of our implementation is it maintains the complete Layer-2 state within the Layer-1 bridge contract. As a result, our approach does not strictly constitute a Layer-2 blockchain but instead involves certain nodes directly interfacing with the bridge contract to facilitate chain progression.

Due to this approach, we have encountered a situation where a monotonically increasing canonical chain has been established, resulting in increased memory con-

sumption. However in the current Ethreum's perspective this does not impose any cost-related implications. But considering the multitude of applications being developed on Ethereum, it is reasonable to anticipate that the protocol may introduce a new gas scheme specifically tailored for storage[133].

Looking into the future and to gain rewards from releasing storage in Ethereum, we have implemented a maintenance mechanism to release matured data from the canonical Rollup chain (aka SCC). The process is designed in a way it shall not release any block data unless there is consensus over the canonical chain. The definition for maturity is three times the challenge period which is set when deploying the bridge contract. We hold an optimistic assumption that users will have released their withdrawal within this maturity period.

4.6 The User Wallet

Wallets are the client-side software for decentralized applications in blockchain. We have implemented a Command Line Interface (CLI) in Node.js for our Rollup prototype.

Figure 35 shows the full description of the current version of the wallet. This includes adding user data in the wallet, initiating Layer-1 deposits and withdrawals, Layer-2 user transactions and retrieving transaction status and balance.

This wallet is a simple prototype of a cryptographic wallet. Our implementation guarantees the user private keys are not revealed to Rollup nodes but not all security threats are considered in this work.

4.6.1 CLI Commands

Below we list down user commands of our CLI wallet.

- **add <key> <privateKey>** add command allows users to enter their "privateKey" with an identification "key" which is a unique id. The key is used identify the account when user has multiple private keys in the wallet.
- **show** Shows wallet data to the user.
- **l1deposit <key> <ether>** Deposit funds in Ethereum blockchain. Command need two inputs "key" to identify correct private key for signing and "ether" value.
- **l1balance <key>** Retrieve Layer-1 balance for the corresponding "key".

Algorithm 1 Dispute Resolution Protocol

Phase 1: Pre-validation of challenge properties
- check enough collateral for the challenge
- challenge should be within the challenge period
- challenged transaction batch should match the associated hash

Phase 2: Batch decompression
- Convert *UTF-8* data to a *hex* stream

Phase 3: Re-execution of transaction batch

if $tx\_root_{gen} != tx\_root_{scc}$ **then**
    $error\_detected \leftarrow true$ ▷ Validate transaction root
end if

if $preState\_root_{gen} != preState\_root_{scc}$ **then**
    $error\_detected \leftarrow true$ ▷ Validate pre-state root
end if

if !$error\_detected$ **then**
    foreach transaction **in batch do**
        execute transaction
    end for
    if $postState\_root_{gen} != postState\_root_{scc}$ **then**
        $error\_detected \leftarrow true$ ▷ Validate post-state root
    end if
end if

if $error\_detected$ **then**
    $rebatch \leftarrow true$
    State Commitment Chain is reordered
    Perpetrator's bond is slashed ▷ half of the bond is burned
    Winner is rewarded ▷ the other half of the bond is rewarded
end if

```
optimistic rollup prototype user wallet

Options:
  -h, --help                                    display help for command

Commands:
  add <key> <privateKey>                        Add a key and a private key to the wallet
  show                                          Show wallet data
  l1deposit <key> <ether>                       Desposit ether in Ethereum blockchain (L1).
  l1balance <key>                               Get l1 balance
  l2balance <key>                               Get l2 balance
  l2transfer <key> <target> <value> <nonce>     Submit l2 transaction to the sequencer
  l1withdraw <key> <id>                         Withdraw ether in Ethereum blockchain (L1)
  l2withdraw <key> <value> <nonce>              Submit l2 withdraw to the sequencer
  l2status <id>                                 Get transaction Status
  help [command]                                display help for command
```

Figure 35: The CLI Implementation of User Wallet

- **l2balance** <**key**> Retrieve Layer-1 balance for the corresponding "key".
- **l2transfer** <**key**> <**target**> <**value**> <**nonce**> Layer-2 user transactions. Inputs are "key" identifying the user private key, "target" receiver public address, "value" transfer amount and the correct "nonce". It returns the transaction "id" of the l2transfer.
- **l2withdraw** <**key**> <**value**> <**nonce**> Layer-2 withdrawal transaction. It takes three inputs, "key", "value" and "nonce". It returns the transaction "id" of the withdrawal.
- **l1withdraw** <**key**> <**id**> Layer-1 withdrawal transaction. The two inputs are "key" and "id", the transaction id of the corresponding l2withdraw. The user should first initiate a l2withdraw and then wait the challenge period until the transaction get finalized in Layer-1.
- **l2status** <**id**> Get the status of the associated Layer-2 transaction "id".

4.7 Rollup Implementation Details

We have successfully implemented our Rollup system using the latest libraries in JavaScript and Solidity. To facilitate testing and demonstration, we have incorporated Hardhat and utilize its local blockchain node.

The code for our Rollup system can be found in our GitHub repository at the following link: `https://github.com/dakilaserasinghe/oprollup`.

5 Discussion

In this chapter, we present a comprehensive analysis of our Optimistic Rollup implementation. We begin by examining the data compression technique employed in our work, drawing connections to similar works in existing implementations.

Subsequently, we delve into the security aspects of Optimistic Rollups, discussing the underlying assumptions embedded in our protocol. To assess the performance of our implementation, we perform a detailed analysis of scalability and usability. This includes a system performance and characteristic evaluation with a comparison with existing solutions.

Finally, we recognize limitations encountered during our work and outline potential avenues for future research for protocol enhancement.

5.1 Calldata Compression

In our analysis on Layer-2 gas costs in the State-of-the-Art Chapter, showed that majority of the gas costs in Rollups corresponds to the Layer-1 Calldata submission. Since EIP-2028 : "Transaction data gas cost reduction", Ethereum is already charging less on Calldata but this still accumulates to high costs in the context of Rollups. Primarily there are two ways of reducing the cost on Layer-1 Calldata.

1. Reduction of Calldata cost (gas/byte) in Layer-1.

2. Reduction of Calldata size.

Reducing the Calldata cost is not exactly a straightforward optimization as it requires fundamental protocol changes in the EVM code in Ethereum. But there are some improvements ahead of us such as EIP-4844 [134] and EIP-4488: Transaction Calldata gas cost reduction with total Calldata limit [135].

But the latter is in the scope of Layer-2 as its implementation is independent of Layer-1. Reducing the size of the foot-print in L1 primarily leads to "data-compression". Production level Rollups are already leading the research on compression. Optimism

and Arbitrum are compressing their Calldata using different general purpose compression algorithms. Arbitrum is using Brotli [136] and Optimism currently using Zlib [137] algorithm for data compression.

In [138] Optimism sums up their strategy on data compression algorithm selection. In this study they look at a variety of compression algorithms (Brotli [136], Zlib [137], Zstd (Zstandard) [139], LZW (Lempel–Ziv–Welch) [140], ZLE (Zero Length Encoding) [141]) and measure both the compression rate (compressed data size as a percentage of uncompressed data size) and estimated fee savings (assuming 40% of bytes in a transaction are of zero-bytes).

The results of this experiment (in Table 7) demonstrates higher compression ratio for batches of transactions rather than for individual transactions. The better results from batch level compression could be of several reasons. User transactions shall belong to same applications (Uniswap etc.) thus using identical function signatures and some transaction fields (gas fees, chain ID) could be similar across transactions. These similarities in transactions lead to significant performances in compression.

Replicating compression gains, batch compression shows better fee savings. In single transactions, it predominantly tends to remove only the zero-bytes. Since Ethereum already costs less gas for zero-bytes (4 gas) compared to non-zero bytes (16 gas), it results in less savings even after the compression. But when batched with thousands of transactions it shows notable savings with better non-zero byte compression.

Optimism's experiment identify Zlib, Zstd and Brotli algorithms all have the best compression capacity on transaction batches. But they decided to use Zlib given the better run time and availability on different programming languages. In the long run they expect to use Zstd to achive higher compression levels. Zstd demonstrates better compression with the use of a dictionary (37.31%).

According to Optimism's research, Brotli algorithm is slower compared to other two in achieving the same compression ratios. The trade-off between execution speed and compression ratio plays as the deciding factor on selecting the compression algorithm. But in contrast Arbitrum uses it in their stack [142] [143].

5.1.1 Calldata Compression in This Work

Our implementation is taking a different approach in terms of verification compared to the general implementations. In current rollups either they first agree on the mismatch at a granular level (Multi-Round Verification) and only run that operation in Layer-1 reducing the costs. Else it sends the whole batch to the Layer-1 for

Algorithm	Batches of Transactions		Single Transaction	
	Compression Rate (%)	Estimated Fee Savings (%)	Compression Rate (%)	Estimated Fee Savings (%)
Zlib	40.13	42.67	63.97	8.61
Zstd	40.12	42.68	62.87	10.18
LZW	62.24	11.09	68.29	2.44
Brotli	39.21	43.99	65.78	6.02
ZLE	59.37	15.18	59.37	15.18

Table 7: Calldata Compression Analysis in Optimism[138]

verification (Single-Round Verification).

In both above scenarios, operators (Sequencer, Verifier) agree on the data before they execute the final verification in Layer-1. In contrary we have implemented the system in such a way it doesn't require external parties to agree on the challenge before execution. Once a verifier sees a malicious batch, it can challenge it straight away. But this comes with a cost of need to decompress the transaction data in Layer-1.

In order to facilitate on-chain decompression, algorithm should be written in Solidity. We could not find any Solidity implementation of aforementioned algorithms for the use of our work. But we found a library [132] which is an implementation of the Puff decompression algorithm [144] in Solidity.

In our work, data compression is done using "Pako" [145], a high speed Zlib port to JavaScript and decompressed on-chain using Puff in case of a "Challenge".

Puff Decompression

Puff is a lightweight library that is specifically designed for decompression. It is used in some embedded systems or similar environments where memory is limited and speed is not critical.

Puff is based on the DEFLATE algorithm [146], which is a combination of Huffman coding [147] and LZ77 compression [148]. Huffman coding is used to compress the data by creating a variable-length code for each symbol in the data based on their frequency of occurrence. This allows more frequently occurring symbols to be represented by shorter codes, resulting in better compression. Puff uses a simplified version of the Huffman decoding algorithm that is optimized for low memory usage than Zlib.

# of Transactions	Compression Ratio (%)	Estimated Fee Savings (%)
1	15.58	34.80
2	9.64	50.63
3	7.65	61.65
5	5.55	71.94
7	4.51	77.58
10	4.03	80.58
30	2.90	85.66
50	2.75	86.97
75	2.56	87.48
100	2.53	87.54
200	2.42	88.09
500	2.33	88.63
1000	2.30	88.83

Table 8: Calldata Compression Ratio and Fee Savings

An Experiment on Our Calldata Compression Mechanism

An experiment was carried out to measure the data compression capacity and fee savings in this work. Compression ratio is derived as compressed batch size as a percentage of uncompressed batch size. Fee saving is estimated as Calldata gas cost saving as a percentage of uncompressed Calldata cost. In each experiment, the transaction batch is compressed with different number of transactions ranging from 1 to 1000.

$$Fee\_Savings = \frac{uncompressed\_cost - compressed\_cost}{uncompressed\_cost} \times 100 \tag{6}$$

$$Compression\_Ratio = \frac{compressed\_size}{uncompressed\_size} \times 100 \tag{7}$$

Table 8 presents the corresponding compression ratios and estimated fee savings as a percentage. Figure 36 demonstrate there is a diminishing rate of increase in fee savings and decrease in compression ratio after 200 transactions in a batch.

This indicates both fee savings and compression ratio plots approaches a plateau, suggesting that further increase in the transaction batch size is unlikely to achieve substantial increase or decrease in fee savings and compression respectively.

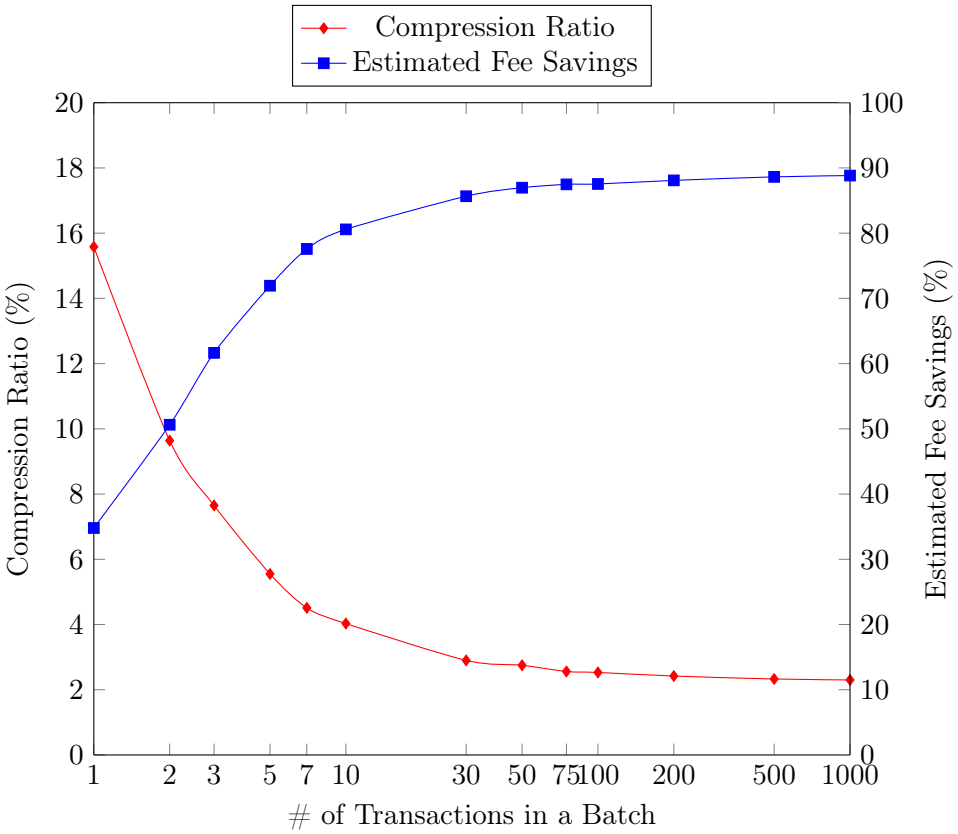


Figure 36: Calldata Compression Ratio and Fee Savings

5.2 Security

Rollups as a promising scaling solution for EVM based blockchains has shown an optimistic trajectory for addressing the scalability and high transaction costs. However adoption of rollups also raises new security concerns as they go beyond the fundamental assurances of a Layer-1 blockchain. In this section we study the common security implications of Rollup systems, analyze potential risks and discuss possible measures to mitigate them.

We expand the knowledge accumulated in [101] where authors systemize the knowledge on Layer-2 bridge validation. Furthermore we explore the literature on risk analysis on blockchain scaling solutions but primarily focusing on Rollups implementations.

We identify key assumptions and issues on current Rollups such as liveness assumption of operators, censorship resistance, transaction sequence ordering, upgrade keys, bridge vulnerabilities. We provide a detailed explanation on each topic and discuss latest approaches to alleviate them. In each section we compare our implementation on security threats and assumptions.

5.2.1 Centralization of Operators

According to the famous "scalability trilemma" [3], blockchain's security, decentralization and scalability shall never be achieved all at once. The concept of Rollups attempts to solve this problem gracefully. But deep down in the current Rollup architectures there are some aspects which bring the blockchain out of the scope of decentralization. An example of this is centralization of Rollup operators.

A centralized operation is not completely dreadful as we think. It can offer great efficiency and faster finality as there is a single point of coordination for processing. However, it also comes up with potential threats such as single point of failure and imposed capacity on manipulation and control.

In current Rollup landscape there are three actors involved in the Layer-2 operations, sequencer (aggregator), validator (executor) and verifier (challenger). The sequencer is tasked with transaction ordering, validators are submitting the L2 state executing the sequenced batches from sequencer and verifier is expected to examine sequencer submissions in Layer-1. Currently all available Rollup implementations are operating with a single sequencer which is run by respective production teams and a set of whitelisted validators chosen by them.

Given the rapidly developing nature of this paradigm this approach allows teams

to upgrade their protocols easily. But it comes up with key assumptions we make on the integrity and liveness of the operators.

Assumption 1: Liveness of Sequencer and Validators

Centralized sequencers add up an assumption on continuous operation without any downtime. But it has no guarantee and could go down anytime due to network traffic, new releases and other factors [149]. Once the sequencer goes off-line, block production stalls and assets get stranded. This undermines the system making user funds unusable in Layer-1.

But some rollups provide assurance on such cases through escape hatches [150]. Escape hatches allow users to recover their assets from the Layer-2 when the operators are offline. Here the Bride contract keeps a liveness signal from the sequencer. If this "keep-alive" signal is not received within a certain period of time, users can withdraw their assets through the bridge contract [151]. Design of a escape hatch is differed across rollups, but they fundamentally equal to the procedure stated above.

In the same way escape hatches are developed to allow users to withdraw their funds in case of validator failures. Users can exit the rollup trustlessly submitting either a merkle proof or a Zero-Knowledge proof of their assets.

In our implementation, sequencer is responsible for both transaction ordering and execution. We only assume the liveness of the sequencer and no exit mechanism is being implemented in the protocol.

Assumption 2: Liveness of 1-of-N Honest Verifiers

Optimistic Rollups are fundamentally based on the assumption of at least one "Active Honest" verifier in the system. If there is no verifier actively participating in Rollups, the whole security is at risk as it allows the sequencer to manipulate and steal user funds.

This assumption inevitably increases the risk of collusion of malicious sequencers and verifiers aka "Bribery Attack" [118]. However these attacks shall be curtailed by setting right economic incentives for verifiers, thus making "turn-it-in" economically lucrative than colluding. Additionally, risk can be further reduced by increasing the number of verifiers.

We discuss this issue further under "*Challenges*" in section 3.5.2.

Assumption 3: Censorship Resistance

Another assumption we make in general due to centralized sequencing is oprators will act honestly without censoring any party from participation. Censorship shall happen at two levels. In Layer-2 a malicious sequencer shall directly suppress particular user transactions from being added in the upcoming block. Secondly it could occur in underlying blockchain where Layer-1 miners are reluctant to add Layer-2 transactions in the Layer-1 blocks.

- **Layer-1 censorship resistance** A "miner-cartel" in Layer-1 blockchain could attack a targeted Rollup design by continuously resisting to add their transactions in the Layer-1 canonical chain. This resembles a 51(%) attack which has a very little probability in a blockchain like Ethereum.

  Risk of censorship resistance based on the first-price auctions shall be eliminated by the introduction of Proposer Builder Separation (PBS) in Ethereum [152]. Nonetheless the risk of censorship will not be completely vanished. There are two possible censorship attacks that could harm Rollups [153].

  - **Forking Censorship** Layer-1 validators may conspire to suppress blocks with Layer-2 challenges by forking the Layer-1 chain so challenges never become the part of the canonical chain.
  - **Block-Building Censorship** Layer-1 block builders are bribed to not to include challenges from verifiers in their blocks within the challenge period.

  In both above attacks Layer-2 sequencer is able to successfully bribe majority of validators in Layer-1. To carry out Block-Building censorship, malicious party has to bribe all builders/validators in Layer-1 to attack successfully. If one non-censoring builder includes a block in their blocks, given that block is chosen by the proposer, attack fails. But forking censorship only needs the majority validators to collude to attack. But making challenge periods long enough (7 days) we shall make these attacks economically non-viable.

- **Layer-2 censorship resistance** Layer-2 Rollups have introduced solutions for censorship, allowing users to submit transactions in Layer-1 bridge contract in case of a malicious sequencer [131]. Users can send L1 messages to the Bridge Contract where sequencers are forced to include them in the sequenced block given a certain delay.

Some Rollups use "split-batch" paradigm, where the action of submitting a batch of layer-2 transactions and the action of submitting a state root are done separately [52]. But this shall only mitigate censorship when sequencer is decentralized. When multiple sequencers publish batches in parallel, validators shall choose the next batch independently improving the censorship resistance. But again we should have enough distribution of operators to reducing collusion.

In this work we assume our sequencer to act honest with no risks on Layer-2 censorship and we further assume there is no risk of censorship in the underlying blockchain.

Assumption 4: Fair Sequencing and Execution

If a particular transaction gets precedence over another transaction (e.g. token swaps in DEXes) in a block, it may impose an economically boosting effect on one of them while acting conversely on the other. Therefore transaction ordering is an important factor in a Rollup as it could cause economic ramifications on users. But the centralized environment in current Rollups allow sequencer to manipulate transaction arrangement in order to maximize its profits. This is discussed in the literature as "Maximal Extractable Value" (MEV) [154].

Current Rollups with centralized sequencers promise not to extract profits through fair ordering (First-Come-First-Serve) and execution [61]. Users shall bear with it as long as they trust these development teams. But once the sequencer is decentralized this becomes a concern like in Ethereum [155]. Predominantly there are two views on MEV in Rollup landscape. Some view it as healthy for the system [156] and others consider it is harmful for users over its benefits [157].

Following this idea two leading Optimistic Rollups have taken different approaches, Where Optimism proposes a competitive market for transaction ordering inventing a MEV auction (MEVA) mechanism [114]. The winner of the auction has the right to reorder submitted transactions and insert its own transactions. Arbitrum has taken a MEV minimized approach as they propose a committee-based [61] sequencer based on distributed BFT algorithms [158] [159]. Here the Layer-2 MEV is only possible if more than 1/3 of the committee are malicious [160]. Both these approaches are in the roadmaps of respective projects but still not finalized on the protocol.

Fair ordering is a hot topic in Layer-2 research space. Apart from above solutions there are other techniques being discussed such as Threshold Encryption [161], Timelock Encryption [162] and Fair Sequencing Service [163] to minimize MEV in

Rollups. In our work we only assume of an honest sequencer with First-Come-First-Serve policy on transaction ordering despite the possibilities of MEV.

5.2.2 Upgradeability

Every protocol or standard has inherent faults and limitations, which highlight the need for continuous improvement. In order to ensure these improvements are performed transparently, it is essential to establish a proper framework for executing protocol upgrades and bug fixes. Within the current Rollup ecosystem, the majority of protocol upgrades are reliant upon a single trusted party [164]. However this centralized approach introduces a significant degree of risk, as compromised upgrade keys could potentially place all deposited assets in jeopardy.

In light of these concerns, it is imperative to explore alternative approaches that prioritize transparency and decentralized governance in order to enhance the overall security and resilience of the system. There are two strategies for the governance, permissioned and permissionless.

Security Councils

"Security Council" is a permissioned governance methodology which is already being favoured by several Rollup projects [165] [166]. The security council is a permissioned group of well-known members of the Ethereum community [167] who has the authority on participating multi-signature contract to govern upgrades.

When a new upgrade is announced, users are given the liberty to opt-out of the system within a timelock period (e.g 4 weeks [166]) should they not agree with it. But if it is a critical vulnerability, there should be a mechanism to accelerate the process making it instant. In such situations the security council is trusted to bypass the timelock for the safety of the protocol.

Decentralized Autonomous Organization (DAO)

Second approach is the introduction of Decentralized Autonomous Organisation for permissionless governance [168]. This is an improvement to the security council concept increasing decentralization of protocol upgrades. Here every upgrade proposal will be aggressively debated and governance token holder can vote on them to accept them into the protocol. Once the proposal passes, users are given time to withdraw assets from the system, if they are not in alignment with the majority.

The use of a DAO will completely decentralize the upgrading process as it does not need any permissioned party to carry out the procedure. But given the probability of critical bugs in the protocol, only enabling slow-moving DAO for governance is not desirable in the current state of Rollups. Although there is a risk of collusion in security council model, it is preferred to employ both strategies concerning the safety in case of an emergency [169].

We have not considered on upgradeability measures in our implementation. Therefore it is exposed to the risks discussed above.

5.2.3 Bridge Contract Vulnerabilities

Layer-2 solutions such as Rollups are complex protocols enriched with multiple interactions between various independent parties. This great complexity in the logic increases the likelihood of vulnerabilities in the Bridge Contract. Hence it is essential to examine security threats in Bridge Contracts.

Smart contract security vulnerabilities are well discussed in both academia and industry over the years [170, 171, 172]. It has been proposed various countermeasures to combat them such as code reviews, best practices and formal verification [173]. Considering the value composition of Rollup smart contracts, it is vital to incorporate independent third parties on bridge contract risk analysis to enhance the trust of users.

Security Audits

Security Auditing [174] is one way to detect these security flaws before publishing the contract. Independent third-party audits play a key role in reducing security threats and design errors in smart contracts.

Rollup projects are continuously conducting security audits before each major update to mitigate security errors in their code bases [175] [176].

Bug Bounties

Bug bounty programs are another way to mitigate security threats in smart contracts [177]. It encourages ethical hackers to inspect smart contract codes by offering financial rewards. Many Rollups are undertaking such programs to improve security in their protocols [178]. In reality these programs have helped developer teams to identify critical bugs that could have compromised the entire system [179].

In the context of our Rollup prototype, we have not done any security audits on the bridge contract. But we have followed the best practices in the process of smart contract development.

5.3 Scalability and Usability Analysis

Our implementation of Optimistic Rollups mainly focuses on identifying the design aspects and challenges on Rollup system design. In this section we analyse the scalability of the system and derive its capacity on production and research works based on the performance.

Layer-1 blockchains with conventional consensus algorithms are suffered either from the computational capacity or overheads of consensus protocol. Ethereum's move from proof of work to proof of stake, has not achieved much in the throughput.

5.3.1 Throughput of Ethereum

We collect the block data of Ethereum, for the period of 18th May 2023 to 18th June 2023. In average, Ethereum has 7k blocks per day with an average of 1000k daily transactions. Average block time measures only upto 12 seconds per block. This leads that Ethereum only perform with approximately 12 transactions per second.

$$\begin{aligned} Average\ Transactions\ per\ Block &= \frac{Average\ txs\ per\ day}{Average\ Blocks\ per\ day} \\ &= \frac{1,000,000}{7,000} \\ &\approx 143\ Transactions/Block \end{aligned}$$

$$\begin{aligned} Throughput &= \frac{Average\ Transactions\ per\ Block}{Average\ Block\ Time} \\ &= \frac{143}{12} \\ &\approx 12\ Transactions/Second \end{aligned}$$

5.3.2 Throughput of This Work

Theoretical Maximum Throughput

Considering the current capacity of Ethereum, we calculate the theoretical maximum throughput we can achieve in our Rollup system. Since 2021 August, maximum gas

limit is doubled up to 30 million gas per block. This allows more transaction inclusion in the Ethereum blocks.

- Max Gas limit: 30 Million gas/block
- Gas per Byte in L1 transaction: 16 gas/Byte
- Average block time: 12 sec/block

Suppose our Rollup use the maximum available gas in a Ethereum block. Then we shall derive the maximum block size of a Rollup block.

$$\begin{aligned} Block\ Size &= \frac{Max\ Gas\ Limit}{Gas\ per\ Byte} \\ &= \frac{30,000,000\ (Million\ gas/block)}{16\ (gas/Byte)} \\ &\approx 1,875,000\ Byte/block \end{aligned}$$

Based on the data compression experiment (in Figure 36), we assume a compression ratio of a 2.5% on the uncompressed transaction data. Then we derive the uncompressed size of the Rollup block.

$$\begin{aligned} Uncompressed\ Transaction\ Block\ Size &= \frac{1,875,000}{0.025} \\ &= 75,000,000\ Byte/block \end{aligned}$$

We experimentally evaluate the average Rollup transaction size of our work as 650 Bytes per transaction. The average number of transactions per block is the ratio of block size and the average Rollup transaction size.

$$\begin{aligned} Average\ Transactions\ per\ Block &= \frac{Block\ Size}{Average\ Rollup\ tx\ Size} \\ &= \frac{75,000,000\ (Bytes/Block)}{650\ (Bytes/Transation)} \\ &\approx 115,385\ (Transactions/Block) \end{aligned}$$

$$
\begin{aligned}
Throughput &= \frac{Average\ Transactions\ per\ Block}{Average\ Block\ Time} \\
&= \frac{115,385}{12} \\
&\approx 9,615\ Transactions/Second
\end{aligned}
$$

Our system has the potential to achieve an impressive throughput of 9000 transactions per second, representing a significant improvement over Ethereum's current throughput limitations.

While our system may focus on a specific use case rather than supporting general smart contracts, its enhanced throughput capabilities can still greatly benefit payment-oriented applications, enabling faster and more responsive financial transactions.

Practical Throughput

When we derive the theoretical throughput, we assume Rollup achieves to consume every Ethereum block. This is rather a very optimistic assumption which is very far from the reality.

In practical scenarios, the actual throughput of a system can be influenced by various factors, including competition from other Rollup implementations vying for space in Ethereum blocks. Therefore, relying solely on the theoretical maximum throughput may not accurately reflect the real-world performance of a system. It is important to consider the dynamic and competitive nature of the Ethereum ecosystem, where multiple solutions are contending for transaction inclusion.

In such environments, achieving and sustaining the highest possible throughput requires not only theoretical capacity, but also effective optimization, efficient resource allocation, and adaptability to changing network conditions.

Through a brief experiment, we conducted a practical assessment of the throughput in our system. Our findings indicate that the Rollup block time is primarily constrained by the execution of the sequencer run. The sequencer is responsible for sequencing transactions and executing them before appending them to the Bridge Contract. To accommodate other operations within the sequencer, we have set a period of 100 seconds. However, under operational conditions, it has been observed that the average block time can significantly increase, reaching up to 200 seconds per block. This highlights the impact of the sequencer's execution time on the overall system throughput and the need for optimization measures to address this constraint.

With several optimizations in event listening of Ethereum events (from 60 secs to 12 secs) and Sequencing period (from 100 secs to 20 secs), we were able to bring down the block time approximately to 20 seconds per block.

Despite efforts to optimize the Rollup system, the observed performance falls short in comparison to Ethereum's throughput. With only 20 transactions per block and a block time of 200 seconds, the Rollup system achieves a meager 0.1 transactions per second. Even with a reduced block time of 20 seconds, there is only a marginal improvement.

Unfortunately, these results demonstrate that the current optimizations are insufficient to meet the desired transaction throughput. Further enhancements and optimizations are necessary to significantly improve the Rollup system's performance and bridge the gap with Ethereum's transaction processing capabilities.

We only could conduct a brief experiment due to the time constraints of the thesis. To ensure a comprehensive evaluation of practical throughput, it is needed to conduct an extensive test that aggressively inject transactions into the Rollup bandwidth to test operational capacity.

A comprehensive analysis of the practical limitations of Rollup systems has been presented in [180]. The study involved conducting experiments and gathering transaction block data from existing Rollups over a period of one year. Such in-depth analysis requires significant time and resources to accurately assess the throughput of Rollup systems.

Further, It is needed to explore various corner cases and implement scenarios that push the boundaries of our system. This includes rigorous testing to minimize the sequence time as much as possible. By subjecting our system to these extreme conditions, we aim to identify any potential limitations and fine-tune our solution for optimal performance.

5.3.3 A Comparison with Existing Solutions

Rollup System Performance Evaluation

Table 9 shows a comparison of existing Layer-2 systems with our Rollup implementation. The maximum theoretical throughput of each mentioned Layer-2 systems are extracted by their respective white papers.

The significantly low throughput observed in our work cannot be ignored, and it calls for further analysis and improvement. This limitation stems from various factors such as the constraints imposed by the Rollup architecture, the execution time of the

sequencer, and the limited number of transactions processed per block.

Additionally, the block time intervals and transaction processing delays due to user interactions, contribute to the overall reduced throughput. To overcome this challenge, it is crucial to explore innovative optimizations, enhance the scalability of the system, and fine-tune various components to achieve higher transaction throughput and improve overall system performance.

System	Max Theoretical Throughput (tps)	Practical Throughput (tps)	L2 Approach	Finality
Polygon[95]	65,000	6,900	Side Chain/Plasma	3h (PoS), 7 days (Plasma)
Optimism[58]	200	110	Opt. Rollup	7 days
Arbitrum[59]	4,500	102	Opt. Rollup	7 days
ZKSync[94]	3,000	165	ZK Rollup	1 hour
This work	9,000	1[1]	Opt. Rollup	7 days

Table 9: A Comparison and Performance Evaluation with Existing Major Layer-2 Projects [180].

Rollup Characteristic Evaluation

We present a proper characteristic comparison in Table 10 comparing Rollup features and attributes with existing Optimistic Rollups. Here we evaluate our work with existing solutions in terms of its features and capacities. This analysis helps to assess the strengths and weaknesses of our implementation and determine its potential advantages or limitations compared to what is already available.

Properties	Arbitrum	Optimism	Fuel v1[181]	This Work
Purpose	Universal	Universal	Payments	Payments
Model	Account	Account	UTXO	Account
Data Availability	On-chain	On-chain	On-chain	On-chain
State Validation	Fraud proofs(MR)[2]	In Dev.	Fraud proofs (SR)[3]	Fraud proofs (SR)
Upgreadibility	SC and DAO[4]	Upgrade keys	Immutable	Upgrade keys
Centralized Sequence	Yes	Yes	Yes	Yes
Censorship Resistance	Self sequence	Self sequence	Self sequence	No mechanism
Security Audits	Yes	Yes	Yes	No
Rollup Stage	Stage 1	Stage 0	Stage 2	Stage 0

Table 10: A Characteristic Comparison with Existing Optimistic Rollups

[1]It is important to note that these limitations arise from the fact that our Rollup system has not undergone a comprehensive test that would encompass the entire network and utilization.

Rollup Maturity Assessment

With the ever-evolving number of Rollup projects in Layer-2 ecosystem, it has been really difficult to rate these projects due to the varied and complex routes they use to achieve a common goal. After several attempts, L2BEAT, an analytics website for Layer-2 scaling solutions for Ethereum, has presented a framework [182] to evaluate the "maturity" of Rollup solutions.

This framework gauge the level of Rollup into three stages 0 to 2, based on specific requirements and conditions that characterize each stage.

Stage 0 Requirements	
Does the project call itself a Rollup?	☑
Are L2 state roots posted on L1?	☑
Does the project provide Data Availability (DA) on L1?	☑
Is software capable of reconstructing the rollup's state open source?	☑
Stage 1 Requirements	
Does the project use a proper proof system?	☑
Are there at least 5 external actors that can submit a fraud proof?	☑
Can the users exit without the operator's coordination?	☒
Do users have at least 7 days to exit in case of unwanted upgrades (Security Council and governance excluded)?	☒
Is the Security Council properly set up?	☒
Stage 2 Requirements	
Is the fraud proof system permissionless?	☑
Do users have at least 30 days to exit in case of unwanted upgrades?	☒
Is the Security Council restricted to act only due to errors detected on chain?	☒

Table 11: Rollup Maturity Assessment of This Work

Our work has been assessed (See Table 11) within the framework and ranked in the initial stage, referred to as "Stage 0" of the maturity ranking. While our system meets certain criteria outlined in stages 1 and 2, it is important to note that the absence of mechanisms for decentralized protocol upgrades and secure user exit brings attention to the limitations of our system.

[2]Multi-Round fraud proof system
[3]Single-Round fraud proof system
[4]Security Council and Decentralized Autonomous Organization

5.3.4 Usability

The primary focus of this thesis is to implement a simple Rollup prototype and delve into the complexities of Rollup system design. Our research involves an in-depth analysis of both the design principles and practical considerations in Rollup systems.

Specifically, our work centers around the protocol design of an Optimistic Rollup system. We have dedicated considerable efforts to designing a lightweight protocol that can be easily integrated into any work which requires an Optimistic Rollup. We identify this protocol holds significant utility in research and development, allowing for testing in demanding Rollup environments that involve decentralized sequencing and aggressive bandwidth utilization, among other challenging scenarios.

It is important to note that our Rollup system may not be deemed secure for production usage and could be vulnerable to crypto-economic attacks. Therefore, it is strongly advised to refrain from deploying it without undergoing a thorough external third-party security audit. As a precautionary measure, our Rollup implementation is not recommended for production environments until it has undergone such an audit and necessary security enhancements have been implemented.

5.4 Limitations

Our Rollup work has certain limitations that should be acknowledged. These limitations include:

- Throughput Constraints: Despite our efforts to optimize the Rollup system, we have observed limited throughput in comparison to Ethereum's transaction processing capabilities. The number of transactions per block and the block time intervals significantly impact the overall system throughput. We identify the bottleneck in our Rollup architecture as the sequencer's dual responsibilities of sequencing and execution of transactions, coupled with direct user interface interaction. This arrangement has led to significant performance degradation, particularly impacting the block time intervals.
- Centralized Sequencing: Our Rollup implementation relies on centralized sequencing, wherein a central authority or entity is responsible for ordering and sequencing transactions. This introduces a centralized point of control and can impact the decentralization and trustlessness of the system.
- Limited Transparency of Transaction Data: Another limitation we have identified is the inherent trust placed on the transaction data submitted by the

sequencer within the sequenced batch. As the sequencer acts as the sole connection point for users, there is a lack of transparency regarding the transaction data within the network.

The effectiveness of the fraud proof mechanism heavily relies on the accuracy and completeness of the transactions to successfully recreate the state root. However, in the event that the transaction batch becomes corrupted or compromised, verifiers will be unable to submit a correct state root, rendering on-chain challenges ineffective.

- Security Considerations: Our Rollup implementation may have potential security vulnerabilities, making it susceptible to crypto-economic attacks. It is crucial to conduct a comprehensive external security audit to identify and address these vulnerabilities before considering production usage.

- Lack of an Escape Hatch: One another notable limitation in our Rollup systems is the absence of an escape hatch mechanism. An escape hatch serves as a safety mechanism that allows for exceptional circumstances or emergency situations to bypass the standard rules and protocols of the Rollup system.

  To mitigate this limitation, it is important to explore and design mechanisms that provide a controlled and secure escape hatch, enabling the system to handle exceptional cases while maintaining the overall integrity and security of the Rollup implementation.

- Lack of Extensive Testing: Although we have conducted experiments and evaluations, there might be scenarios or corner cases that have not been thoroughly tested. Further testing and analysis are required to uncover potential edge cases and ensure robustness.

- Censorship Vulnerability: Our Rollup implementation has limitations in terms of robustness against censorship compared to fully decentralized systems. It is important to note that all production Rollup systems have implemented measures to mitigate potential censorship by operators, typically by introducing a transaction inclusion delay in blocks.

- Time-Constrained Withdrawal Process: Our Rollup withdrawals depend on users submitting proof of inclusion for their withdrawal transactions within a matured batch. However, this approach introduces a level of trust in the sequencer, as users rely on the provided proof of inclusion. Additionally users

are expected to withdraw their funds within a time period (currently set as three times the Challenge Period) otherwise funds are lost and not accessible.

- Fraud Proof Limitation: Our Rollup system utilizes a single round proof system for validating the entire block on Layer-1. This is due to our approach on Rollup architecture as we do not implement a separate Layer-2 blockchain for the process. However, it is important to note that this approach is considered outdated and incurs higher costs when compared to the latest approaches in Optimistic Rollups.

- Developmental Stage: Our Rollup implementation should be considered as a work in progress or at an experimental stage. It may not yet possess the maturity or stability required for widespread production usage.

It is important to recognize these limitations as they help identify issues in the design of Rollup systems. By acknowledging these limitations, we can understand the areas that require improvement and focus our efforts on enhancing the robustness and efficiency of our Rollup solution. This continuous pursuit of improvement ensures that we can deliver a more resilient and efficient Rollup system.

5.5 Future Work

As part of a future work, we identify several ways to enhance our implementation as listed below.

- Sequence and Execution Separation: Our Rollup implementation faced a decrease in throughput due to the burden of multiple responsibilities on the sequencer. In line with the latest advancements in Rollup architectures, it is essential to separate the execution task from the sequencer and delegate it to a dedicated validator. By separating the execution from the sequencer, we can significantly improve the throughput of our Rollup system. The sequencer is no longer encumbered by the execution task, allowing it to focus solely on its sequencing responsibilities. This separation of duties enhances the efficiency and performance of the Rollup system, resulting in higher throughput and improved overall scalability.

- A Fast low-level Implementation of Protocol: Initially, we chose to implement the complete Rollup protocol in JavaScript, considering the ease of implementation.

However, this decision had a significant negative impact on the performance of our system.

To address this limitation and improve the system's performance, it is crucial to transition to a more low-level and efficient implementation using languages such as Rust, Go, or C++. By leveraging the capabilities and optimizations offered by these languages, we can achieve higher performance and better utilization of system resources.

- A mechanism for Censorship Resistance: It is important to develop a mechanism that allows users to submit transactions directly on Layer-1 in the event of a sequencer failure or explicit censorship. This would provide an additional layer of security and ensure that users can continue to interact with the system even in adverse conditions.

- Trust-minimized and Time-insensitive Withdrawal Process: Our current withdrawal mechanism relies on users submitting proof of inclusion for their withdrawal transactions within a matured batch. However, this approach introduces a level of trust in the sequencer, as users must rely on the provided proof of inclusion. This reliance on a trusted system poses potential challenges.

  Furthermore, the periodic deletion of state roots from the state commitment chain creates a time-sensitive withdrawal process. Users face pressure to withdraw their funds within a specific timeframe, which can be inconvenient and restrictive.

  To overcome these limitations, it is essential to enhance the withdrawal mechanism. This can be achieved by introducing a trust-minimized method that eliminates the need for a second user transaction on Layer-1. By implementing a more secure and efficient withdrawal process, users can withdraw their funds without relying on a trusted third party and without being constrained by time limitations.

- Trust Removal from Sequenced Transaction Data: In limitations sections we discussed the lack of transparency on transactions data submitted by the sequencer. This has introduced an unnecessary trust on the sequencer. To address this limitation, it is crucial to explore mechanisms such as decentralized transaction verification or cryptographic-proofs, that enhance the transparency and verifiability of the transaction data within the Rollup system.

By introducing measures to validate and cross-check the integrity of the sequencer's transaction data, we can mitigate the risks associated with relying solely on the sequencer's submissions, thereby enhancing the overall security and reliability of the system.

– An Escape Hatch: An escape hatch should be implemented to ensure the security of user funds in the event of an emergency within the Optimistic Rollup system. This escape hatch would serve as a fail-safe mechanism to protect user assets in unforeseen circumstances such as protocol vulnerabilities, critical failures, or external attacks.

  By incorporating an escape hatch, users can have peace of mind knowing that their funds are safeguarded and can be accessed even in the face of emergencies, further enhancing the overall security and reliability of the Optimistic Rollup system.

6 Conclusion

Rollups have emerged as a viable solution for the short-term, mid-term and probably long-term scaling solution for the Layer-1 blockchains. Since its emergence in 2018, the concept of Rollups has been a revolutionary force in blockchain scaling. The Ethereum community has identified the significance of Rollups as they have introduced a Rollup centric future roadmap for Ethereum.

Various implementation models of Rollups have been proposed both by the academia and industry, where they leverage the concepts of modular blockchain. Predominantly, Rollups are classified into two major categories, namely Zero-Knowledge (ZK) Rollups and Optimistic Rollups. ZK Rollups, incorporating Zero-Knowledge proofs, are considered as the holy-grail for scaling EVM based blockchains, as they are capable of providing a sub-second transaction finality with ultimate security of a Layer-1 blockchain. In contrast, Optimistic Rollups are less complex schemes with fraud proofs to secure the integrity of the protocol.

In this work we design and implement a prototype for an Optimistic Rollup system for simple payments. First we extensively discuss the current architectural aspects of Optimistic Rollup designs and provide a complete protocol of a Rollup for a simple payment system. By doing so we develop all Optimistic Rollup components, namely a sequencer, a verifier and the bridge contract providing a wallet as the user interface.

Further we deliver a comprehensive analysis on the Rollup economics, delving deep into the intricacies of Layer-2 transaction costs. As a result we identify the critical cost factors of Rollups and highlight data compression as a fundamental element in mitigating Layer-1 security expenses. Consequently, we investigate data compression methods employed in Optimistic Rollups and perform an experiment to evaluate the effectiveness of data compression in our study.

Our comprehensive evaluation of our Rollup system includes a thorough performance and characteristic assessment, comparing it to existing solutions. We measure scalability by analyzing the system throughput and also highlight the limitations of our work in comparison to available Optimistic Rollups.

Additionally, we analyze key security concerns in Optimistic Rollups, scrutinizing

the issues and assumptions made in our work while presenting potential solutions available in the ecosystem. Lastly, we explore primary challenges associated with Optimistic Rollups and discuss the latest research approaches addressing them.

We accomplish the primary objective of this thesis, by conducting an in-depth examination of the complete design process for a scalable Optimistic Rollup system tailored specifically for simple payments. With the employment of cutting-edge libraries and widely adopted industry standard frameworks, e.g. Hardhat, we create a seamlessly integrable Rollup prototype, facilitating smooth research and development processes.

7 Acknowledgments

I would like to express my deepest gratitude and appreciation to everyone who has supported and contributed to the completion of this thesis.

First and foremost, I would like to thank my examiner, Prof. Dr. Peter Thiemann and supervisor, Dr. Thi Thu Ha Doan, for their invaluable guidance, expertise, and continuous support throughout this research endeavor. Their insightful feedback and encouragement have been instrumental in shaping the direction and quality of this thesis.

I am indebted to my colleagues and friends who have been a constant source of support, motivation, and intellectual stimulation throughout this journey. Their encouragement, discussions, and collaborative spirit have contributed significantly to my personal and academic growth.

I would like to extend my heartfelt gratitude to Sandra Wollenburg for her invaluable support and assistance in proofreading. Her dedication and attention to detail have been instrumental in ensuring the accuracy and quality of this work.

I would like to extend my appreciation towards the whole blockchain community. Their continuous research and invaluable insights have been invaluable and have significantly enriched the content and the quality of this work.

Last but not least, I would like to express my heartfelt gratitude to my family for their unwavering love, encouragement, and belief in my abilities. Their support and understanding during the ups and downs of this academic pursuit have been my anchor and source of strength.

Finally, to the people of Sri Lanka who have contributed to the development and sustenance of the free education system, thank you for providing me with the opportunity to pursue my educational goals and reach the highest levels of academic achievement. I am forever grateful for the invaluable gift of education that you have bestowed upon me.

Bibliography

[1] "ETH fees over $200?." Available: https://www.reddit.com/r/ExodusWallet/comments/lfegg9/eth_fees_over_200/. [Online; last accessed 18-May-2023].

[2] "Visa fact sheet." Available: https://www.visa.co.uk/dam/VCOM/download/corporate/media/visanet-technology/aboutvisafactsheet.pdf. [Online; last accessed 18-May-2023].

[3] V. Buterin, "Why sharding is great: demystifying the technical properties." https://vitalik.ca/general/2021/04/07/sharding.html, 2021. [Online; last accessed 20-April-2023].

[4] D. Ding, X. Jiang, J. Wang, H. Wang, X. Zhang, and Y. Sun, "Txilm: Lossy block compression with salted short hashing," 2019.

[5] "BitcoinCash." Available: https://bitcoincash.org/. [Online; last accessed 18-May-2023].

[6] L. Luu, V. Narayanan, K. Baweja, C. Zheng, S. Gilbert, and P. Saxena, "Scp: A computationally-scalable byzantine consensus protocol for blockchains," *Cryptology ePrint Archive*, 2015.

[7] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, "Bitcoin-ng: A scalable blockchain protocol," 2015.

[8] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "Sok: Sharding on blockchain." Cryptology ePrint Archive, Paper 2019/1178, 2019. https://eprint.iacr.org/2019/1178.

[9] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol." Cryptology ePrint Archive, Paper 2016/1159, 2016. https://eprint.iacr.org/2016/1159.

[10] T. Zhou, X. Li, and H. Zhao, "Dlattice: A permission-less blockchain based on dpos-ba-dag consensus for data tokenization," *IEEE Access*, vol. 7, pp. 39273–39287, 2019.

[11] L. Cui, S. Yang, Z. Chen, Y. Pan, M. Xu, and K. Xu, "An efficient and compacted dag-based blockchain protocol for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4134–4145, 2020.

[12] A. Hafid, A. S. Hafid, and M. Samih, "Scaling blockchains: A comprehensive survey," *IEEE Access*, vol. 8, pp. 125244–125262, 2020.

[13] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, and R. P. Liu, "Survey: Sharding in blockchains," *IEEE Access*, vol. 8, pp. 14155–14181, 2020.

[14] "State Channels." Available: `https://ethereum.org/en/developers/docs/scaling/state-channels/`. [Online; last accessed 04-May-2023].

[15] J. Coleman, "State channels." Available: `https://www.jeffcoleman.ca/state-channels/`, 2015. [Online; last accessed 04-May-2023].

[16] "Payment Channels." Available: `https://en.bitcoin.it/wiki/Payment_channels`. [Online; last accessed 04-May-2023].

[17] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[18] "Raiden." Available: `https://raiden.network/`. [Online; last accessed 04-May-2023].

[19] N. Papadis and L. Tassiulas, "Blockchain-based payment channel networks: Challenges and recent advances," *IEEE Access*, vol. 8, pp. 227596–227609, 2020.

[20] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostakova, M. Maffei, P. Moreno-Sanchez, and S. Riahi, "Generalized channels from limited blockchain scripts and adaptor signatures." Cryptology ePrint Archive, Paper 2020/476, 2020. `https://eprint.iacr.org/2020/476`.

[21] A. Miller, I. Bentov, R. Kumaresan, C. Cordi, and P. McCorry, "Sprites and state channels: Payment networks that go faster than lightning," 2017.

[22] S. Dziembowski, S. Faust, and K. Hostáková, "General state channel networks." Cryptology ePrint Archive, Paper 2018/320, 2018. `https://eprint.iacr.org/2018/320`.

[23] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, and S. Riahi, "Bitcoin-compatible virtual channels." Cryptology ePrint Archive, Paper 2020/554, 2020. https://eprint.iacr.org/2020/554.

[24] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks." Cryptology ePrint Archive, Paper 2017/820, 2017. https://eprint.iacr.org/2017/820.

[25] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "Sok: Layer-two blockchain protocols." Cryptology ePrint Archive, Paper 2019/360, 2019. https://eprint.iacr.org/2019/360.

[26] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "Perun: Virtual payment hubs over cryptocurrencies." Cryptology ePrint Archive, Paper 2017/635, 2017. https://eprint.iacr.org/2017/635.

[27] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, vol. 72, pp. 201–224, 2014.

[28] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, and K.-K. R. Choo, "Sidechain technologies in blockchain networks: An examination and state-of-the-art review," *Journal of Network and Computer Applications*, vol. 149, p. 102471, 2020.

[29] S. D. Lerner, J. Álvarez Cid-Fuentes, J. Len, R. Fernàndez-València, P. Gallardo, N. Vescovo, R. Laprida, S. Mishra, F. Jinich, and D. Masini, "Rsk: A bitcoin sidechain with stateful smart-contracts." Cryptology ePrint Archive, Paper 2022/684, 2022. https://eprint.iacr.org/2022/684.

[30] J. Dilley, A. Poelstra, J. Wilkins, M. Piekarska, B. Gorlick, and M. Friedenbach, "Strong federations: An interoperable blockchain solution to centralized third-party risks," *arXiv preprint arXiv:1612.05491*, 2016.

[31] "Proof-of-Authority Consensus for Sidechains." Available: https://lisk.com/blog/posts/proof-authority-consensus-sidechains. [Online; last accessed 07-May-2023].

[32] A. Kiayias and D. Zindros, "Proof-of-work sidechains." Cryptology ePrint Archive, Paper 2018/1048, 2018. https://eprint.iacr.org/2018/1048.

[33] P. Gaži, A. Kiayias, and D. Zindros, "Proof-of-stake sidechains." Cryptology ePrint Archive, Paper 2018/1239, 2018. https://eprint.iacr.org/2018/1239.

[34] "Plasma." Available: https://ethereum.org/en/developers/docs/scaling/plasma/. [Online; last accessed 06-May-2023].

[35] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," *White paper*, pp. 1–47, 2017.

[36] "The Plasma Framework." Available: https://www.learnplasma.org/en/learn/framework.html. [Online; last accessed 07-May-2023].

[37] "Plasma World Map - the hitchhiker's guide to the plasma." Available: https://ethresear.ch/t/plasma-world-map-the-hitchhiker-s-guide-to-the-plasma/4333. [Online; last accessed 06-May-2023].

[38] "Minimal Viable Plasma." Available: https://ethresear.ch/t/minimal-viable-plasma/426. [Online; last accessed 06-May-2023].

[39] G. Konstantopoulos, "Plasma cash: Towards more efficient plasma constructions," 2019.

[40] "Plasma Debit: Arbitrary-denomination payments in Plasma Cash." Available: https://ethresear.ch/t/plasma-debit-arbitrary-denomination-payments-in-plasma-cash/2198. [Online; last accessed 06-May-2023].

[41] Ashwin Ramachandran, "The Life and Death of Plasma." Available: https://medium.com/dragonfly-research/the-life-and-death-of-plasma-b72c6a59c5ad#. [Online; last accessed 07-May-2023].

[42] barryWhiteHat, "roll up." https://github.com/barryWhiteHat/roll_up, 2018.

[43] "Optimistic Rollups." Available: https://ethereum.org/en/developers/docs/scaling/optimistic-rollups/. [Online; last accessed 06-May-2023].

[44] "Zero-Knowledge Rollups." Available: https://ethereum.org/en/developers/docs/scaling/zk-rollups/. [Online; last accessed 06-May-2023].

[45] "Four Questions To Judge Any Layer 2 Scaling Solution." Available: https://consensys.net/blog/blockchain-explained/four-questions-to-judge-any-layer-2-scaling-solution/. [Online; last accessed 06-May-2023].

[46] Optimism, "Introduction to evm equivalence." Available: https://medium.com/ethereum-optimism/introducing-evm-equivalence-5c2021deb306, Oct. 26, 2021 [Online].

[47] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, vol. 10, pp. 93039–93054, 2022.

[48] C. Sguanci, R. Spatafora, and A. M. Vergani, "Layer 2 blockchain scaling: a survey," 2021.

[49] A. Gangwal, H. R. Gangavalli, and A. Thirupathi, "A survey of layer-two blockchain protocols," *Journal of Network and Computer Applications*, vol. 209, p. 103539, 2023.

[50] Alex Gluchowski, "Evaluating ethereum l2 scaling solutions: A comparison framework." Available: https://blog.matter-labs.io/evaluating-ethereum-l2-scaling-solutions-a-comparison-framework-b6b2f410f955, June 12 2020 [Online]. [Online; last accessed 06-May-2023].

[51] Vitalik Buterin, "A rollup-centric ethereum roadmap." Available: https://ethereum-magicians.org/t/a-rollup-centric-ethereum-roadmap/4698, Oct 2020 [Online]. [Online; last accessed 06-May-2023].

[52] V. Buterin, "An Incomplete Guide to Rollups." https://vitalik.ca/general/2021/01/05/rollup.html, 2021. [Online; last accessed 31-October-2022].

[53] "Rollups as Sovereign Chains." Available: https://blog.celestia.org/sovereign-rollup-chains/. [Online; last accessed 16-May-2023].

[54] "An open, modular stack for EVM-based applications using Celestia, Evmos, and Cosmos." Available: https://forum.celestia.org/t/an-open-modular-stack-for-evm-based-applications-using-celestia-evmos-and-cosmos/89. [Online; last accessed 16-May-2023].

[55] Patrick Mccorry, "Modular Design and the Two Blockchains ." Available: https://stonecoldpat.substack.com/p/modular-design-and-the-two-blockchains. [Online; last accessed 16-May-2023].

[56] "Enshried Rollups." Available: https://www.reddit.com/r/ethereum/comments/vrx9xe/comment/if7auu7/. [Online; last accessed 16-May-2023].

[57] "Validium." Available: https://ethereum.org/en/developers/docs/scaling/validium/. [Online; last accessed 16-May-2023].

[58] "Optimism." https://www.optimism.io/. [Online; last accessed 05-November-2022].

[59] "Arbitrum." https://developer.offchainlabs.com/inside-arbitrum-nitro. [Online; last accessed 05-November-2022].

[60] ethereum optimism, "cannon." https://github.com/ethereum-optimism/cannon/, 2022.

[61] L. Bousfield, R. Bousfield, C. Buckland, B. Burgess, J. Colvin, E. W. Felten, S. Goldfeder, D. Goldman, B. Huddleston, H. Kalodner, F. A. Lacs, H. Ng, A. Sanghi, T. Wilson, V. Yermakova, and T. Zidenberg, "Arbitrum nitro: A second-generation optimistic rollup," 2022. Accessed: 2022-11-01.

[62] Optimism, "Cannon cannon cannon: Introducing cannon." Available: https://medium.com/ethereum-optimism/cannon-cannon-cannon-introducing-cannon-4ce0d9245a03, March 09, 2022 [Online].

[63] ETHGlobal, "Arbitrum inside arbitrum nitro," May 12 2022.

[64] "Ground Up Guide: zkEVM, EVM Compatibility & Rollups." Available: https://www.immutable.com/blog/ground-up-guide-zkevm-evm-compatibility-rollups. [Online; last accessed 06-May-2023].

[65] "Design challenges of zkEVM." Available: https://scroll.io/blog/zkEVM#design-challenges-of-zkevm. [Online; last accessed 06-May-2023].

[66] "Loopring protocol." https://loopring.org/#/protocol. [Online; last accessed 13-April-2023].

[67] "dydX." https://dydx.exchange/. [Online; last accessed 13-April-2023].

[68] "Immutable." https://www.immutable.com/. [Online; last accessed 13-April-2023].

[69] Vitalik Buterin, "The different types of ZK-EVMs." Available: https://vitalik.ca/general/2022/08/04/zkevm.html. [Online; last accessed 06-May-2023].

[70] "ConsenSys zkEVM." Available: https://consensys.net/zkevm/. [Online; last accessed 06-May-2023].

[71] "Benefits of Polygon zkEVM." Available: https://zkevm.polygon.technology/docs/introduction#benefits-of-polygon-zkevm. [Online; last accessed 06-May-2023].

[72] STARKNET, "What is starknet." Available: https://starknet.io/.

[73] David Barreto, "Starknet's-architecture review." Available: https://david-barreto.com/starknets-architecture-review/, June 21, 2022 [Online].

[74] StarkNet Documentation, "Transaction lifecycle." Available: https://docs.starknet.io/documentation/develop/Blocks/transaction-life-cycle/.

[75] "The Native zkEVM Scaling Solution for Ethereum." Available: https://scroll.io/. [Online; last accessed 06-May-2023].

[76] Y. Zhang, S. Wang, X. Zhang, J. Dong, X. Mao, F. Long, C. Wang, D. Zhou, M. Gao, and G. Sun, "Pipezk: Accelerating zero-knowledge proof with a pipelined architecture," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pp. 416–428, 2021.

[77] Scroll, "An overview of scroll's architecture." Available: https://scroll.mirror.xyz/.

[78] zkSync, "Transaction Finality." https://docs.zksync.io/userdocs/tech/#transaction-finality. [Online; last accessed 11-May-2023].

[79] zkSync, "Instant Confirmations." https://docs.zksync.io/userdocs/tech/#instant-confirmations. [Online; last accessed 11-May-2023].

[80] "Checkpoints for faster finality in starknet." Available: https://ethresear.ch/t/checkpoints-for-faster-finality-in-starknet/9633, May 26, 2021 [Online].

[81] X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, and X. Peng, "A survey on zero-knowledge proof in blockchain," *IEEE Network*, vol. 35, no. 4, pp. 198–205, 2021.

[82] J. Partala, T. H. Nguyen, and S. Pirttikangas, "Non-interactive zero-knowledge for blockchain: A survey," *IEEE Access*, vol. 8, pp. 227945–227961, 2020.

[83] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "Snarks for c: Verifying program executions succinctly and in zero knowledge." Cryptology ePrint Archive, Paper 2013/507, 2013. `https://eprint.iacr.org/2013/507`.

[84] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity." Cryptology ePrint Archive, Paper 2018/046, 2018. `https://eprint.iacr.org/2018/046`.

[85] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more." Cryptology ePrint Archive, Paper 2017/1066, 2017. `https://eprint.iacr.org/2017/1066`.

[86] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge." Cryptology ePrint Archive, Paper 2019/953, 2019. `https://eprint.iacr.org/2019/953`.

[87] A. Kattis, K. Panarin, and A. Vlasov, "Redshift: Transparent snarks from list polynomial commitments." Cryptology ePrint Archive, Paper 2019/1400, 2019. `https://eprint.iacr.org/2019/1400`.

[88] E. Ben-Sasson, L. Goldberg, S. Kopparty, and S. Saraf, "Deep-fri: Sampling outside the box improves soundness," 2019.

[89] "Introduction to STARK Recursion." Available: `https://zkevm.polygon.technology/docs/zkProver/intro-stark-recursion/`. [Online; last accessed 12-May-2023].

[90] "zk-SNARK Prover." Available: `https://zkevm.polygon.technology/docs/zkProver/overview/#zk-snark-prover`. [Online; last accessed 12-May-2023].

[91] Y. Zhang, S. Wang, X. Zhang, J. Dong, X. Mao, F. Long, C. Wang, D. Zhou, M. Gao, and G. Sun, "Pipezk: Accelerating zero-knowledge proof with a pipelined architecture," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pp. 416–428, 2021.

[92] "StarkWare." https://starkware.co/. [Online; last accessed 21-June-2023].

[93] StarkNetCC, "Starknet's roadmap: all things regenesis & cairo 1.0 - tom brand & uri kolodny - starknetcc lisbon." https://www.youtube.com/watch?v=1eEO9to4Z-A, 2022. [Online; last accessed 16-Nov-2022].

[94] "zkSync." https://zksync.io/. [Online; last accessed 21-June-2023].

[95] "Polygon Technology." https://polygon.technology/. [Online; last accessed 21-June-2023].

[96] "Scroll." https://scroll.io/. [Online; last accessed 21-June-2023].

[97] "Ethereum Rollup Call Data Pricing Analysis." https://forum.celestia.org/t/ethereum-rollup-call-data-pricing-analysis/141. [Online; last accessed 09-April-2023].

[98] HackMD, "Understanding gas costs after Berlin." https://hackmd.io/@fvictorio/gas-costs-after-berlin. [Online; last accessed 10-April-2023].

[99] "Zero-Knowledge proof verification costs." https://ethereum.org/en/zero-knowledge-proofs/#proof-verification-costs. [Online; last accessed 10-April-2023].

[100] S. Hohl, "Seminar innovative internet technologies: Zero knowledge proofs," 2022.

[101] P. McCorry, C. Buckland, B. Yee, and D. Song, "Sok: Validating bridges as a scaling solution for blockchains." Cryptology ePrint Archive, Paper 2021/1589, 2021. https://eprint.iacr.org/2021/1589.

[102] "Based rollups—superpowers from l1 sequencing." Available: https://ethresear.ch/t/based-rollups-superpowers-from-l1-sequencing/15016. [Online; last accessed 25-April-2023].

[103] J. Kwon, "Tendermint: Consensus without mining," *Draft v. 0.6, fall*, vol. 1, no. 11, 2014.

[104] V. Buterin, D. Hernandez, T. Kamphefner, K. Pham, Z. Qiao, D. Ryan, J. Sin, Y. Wang, and Y. X. Zhang, "Combining ghost and casper," 2020.

[105] "Randao: Under the hood." Available: https://blockdoc.substack.com/p/randao-under-the-hood. [Online; last accessed 25-April-2023].

[106] "Eigenlayer whitepaper." Available: https://docs.eigenlayer.xyz/overview/whitepaper.

[107] "Atom 2.0 whitepaper - everything you need to know." Available: https://blog.switcheo.com/atom2-whitepaper-review/. [Online; last accessed 25-April-2023].

[108] N. White, "Decentralized rollup sequencing as a service via interchain security." Available: https://forum.celestia.org/t/decentralized-rollup-sequencing-as-a-service-via-interchain-security/423. [Online; last accessed 25-April-2023].

[109] "Starknet decentralized protocol ii - candidate for leader elections." Available: https://community.starknet.io/t/starknet-decentralized-protocol-ii-candidate-for-leader-elections/4751. [Online; last accessed 25-April-2023].

[110] "The key role of sequencer decentralization." Available: https://threesigma.xyz/blog/optimistic-rollups-challenging-periods-reimagined-part-two. [Online; last accessed 25-April-2023].

[111] "The espresso sequencer." Available: https://hackmd.io/@EspressoSystems/EspressoSequencer. [Online; last accessed 25-April-2023].

[112] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus in the lens of blockchain," 2019.

[113] "The future of mev is suave." Available: https://writings.flashbots.net/the-future-of-mev-is-suave/#decentralized-block-building. [Online; last accessed 25-April-2023].

[114] Karl Floersch, "Mev auction: Auctioning transaction ordering rights as a solution to miner extractable value." Available: https://ethresear.ch/t/mev-auction-auctioning-transaction-ordering-rights-as-a-solution-to-miner-extractable-value/6788, Jan. 04, 2020 [Online]. [Online; last accessed 20-April-2023].

[115] L. Luu, J. Teutsch, R. Kulkarni, and P. Saxena, "Demystifying incentives in the consensus computer." Cryptology ePrint Archive, Paper 2015/702, 2015. https://eprint.iacr.org/2015/702.

[116] Georgios Konstantopoulos, "(almost) everything you need to know about optimistic rollup." Available: https://www.paradigm.xyz/2021/01/almost-everything-you-need-to-know-about-optimistic-rollup, Jan. 27, 2021 [Online]. [Online; last accessed 20-April-2023].

[117] Ed Felten, "The cheater checking problem: Why the verifier's dilemma is harder than you think." Available: https://medium.com/offchainlabs/the-cheater-checking-problem-why-the-verifiers-dilemma-is-harder-than-you-think-9c7156505ca1, Sep. 04, 2019 [Online].

[118] Aiden Park, "Optimistic rollup is not secure enough than you think — game theoretic approach for more verifiable rollup[en]." Available: https://medium.com/onther-tech/optimistic-rollup-is-not-secure-enough-than-you-think-cb23e6e6f11c, May. 20, 2021 [Online].

[119] Ed Felten, "Cheater checking: How attention challenges solve the verifier's dilemma." Available: https://medium.com/offchainlabs/cheater-checking-how-attention-challenges-solve-the-verifiers-dilemma-681a92d9948e, Sep. 05, 2019 [Online]. [Online; last accessed 20-April-2023].

[120] "Sending data between l1 and l2." https://community.optimism.io/docs/developers/bridge/messaging/#, 2023. [Online; last accessed 03-April-2023].

[121] DeGate Team, "A comprehensive analysis of arbitrum security mechanism." Available: https://medium.com/degate/a-comprehensive-analysis-of-arbitrum-security-mechanism-1a61cbc4fe20, June. 25, 2021 [Online]. [Online; last accessed 02-May-2023].

[122] "Hop: Send tokens across rollups." Available: https://hop.exchange/whitepaper.pdf. [Online; last accessed 25-April-2023].

[123] Connext, "Connext." https://github.com/connext, Aug 2022. [Online; last accessed 03-May-2023].

[124] "What is rubicon?." Available: https://docs.rubicon.finance/#what-is-rubicon. [Online; last accessed 02-May-2023].

[125] "Optimism ecosystem." Available: https://www.optimism.io/apps/bridges. [Online; last accessed 25-April-2023].

[126] "Fast withdrawals." Available: https://docs.fuel.sh/v1.1.0/Tools%20and%20Applications/Fast%20Withdrawals.html. [Online; last accessed 25-April-2023].

[127] "What are the major differences between zksync era and optimistic rollups (eg. arbitrum, optimism)?." Available: https://docs.zksync.io/zkevm/#what-are-the-major-differences-between-zksync-era-and-optimistic-rollups-eg-arbitrum-optimism. [Online; last accessed 02-May-2023].

[128] "Reducing challenge times in rollups." Available: https://ethresear.ch/t/reducing-challenge-times-in-rollups/14997. [Online; last accessed 02-May-2023].

[129] "Road to a dynamic challenging period." Available: https://threesigma.xyz/blog/optimistic-rollups-challenging-periods-reimagined-part-two. [Online; last accessed 02-May-2023].

[130] "Arbitrum glossary." https://developer.arbitrum.io/intro/glossary, 2023. [Online; last accessed 03-April-2023].

[131] "Arbitrum Sequencer." https://developer.arbitrum.io/sequencer. [Online; last accessed 23-March-2023].

[132] J. Adler, "Inflate implementation in Solidity." https://github.com/adlerjohn/inflate-sol, 2021. [Online; last accessed 13-April-2023].

[133] "Should storage be priced separately from execution?." Available: https://ethresear.ch/t/should-storage-be-priced-separately-from-execution/9101. [Online; last accessed 16-May-2023].

[134] D. F. d. Vitalik Buterin (@vbuterin), "EIP-4844: Shard blob transactions." https://eips.ethereum.org/EIPS/eip-4844#how-rollups-would-function, Feb 2022.

[135] A. D. a. Vitalik Buterin (@vbuterin), "EIP-4488: Transaction calldata gas cost reduction with total calldata limit." https://eips.ethereum.org/EIPS/eip-4488, Nov 2021.

[136] "Brotli." https://en.wikipedia.org/wiki/Brotli, Mar 2023. [Online; last accessed 13-April-2023].

[137] "Zlib." https://en.wikipedia.org/wiki/Zlib, Apr 2023. [Online; last accessed 13-April-2023].

[138] Optimism, "The Road to Sub-dollar Transactions, Part 2: Compression Edition." https://medium.com/ethereum-optimism/the-road-to-sub-dollar-transactions-part-2-compression-edition-6bb2890e3e92, Mar 2022. [Online; last accessed 13-April-2023].

[139] "Zstandard compression." https://en.wikipedia.org/wiki/Zstd, Mar 2023. [Online; last accessed 13-April-2023].

[140] "Lzw compression." https://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv%E2%80%93Welch, Mar 2023. [Online; last accessed 13-April-2023].

[141] vbuterin, "Zero length encoding." https://github.com/ethereum/pyethereum/blob/develop-snapshot/ethereum/compress.py, June 2015. [Online; last accessed 13-April-2023].

[142] "Compression in nitro." https://research.arbitrum.io/t/compression-in-nitro/20, Mar 2023. [Online; last accessed 13-April-2023].

[143] eleglenctic, "A short study relating calldata compression to gas savings." https://github.com/eleglenctic/friendly-invention, Mar 2022. [Online; last accessed 13-April-2023].

[144] M. Adler, "Puff." https://github.com/madler/zlib/tree/master/contrib/puff, Aug 2022. [Online; last accessed 13-April-2023].

[145] nodeca, "pako." https://github.com/nodeca/pako, Aug 2022. [Online; last accessed 13-April-2023].

[146] "Deflate." https://en.wikipedia.org/wiki/Deflate. [Online; last accessed 13-April-2023].

[147] "Huffman coding." https://en.wikipedia.org/wiki/Huffman_coding. [Online; last accessed 13-April-2023].

[148] "Lz77 and lz78." https://en.wikipedia.org/wiki/LZ77_and_LZ78. [Online; last accessed 13-April-2023].

[149] "Arbitrum network downtime." https://support.uniswap.org/hc/en-us/articles/7425285242509-Arbitrum-network-downtime. [Online; last accessed 20-April-2023].

[150] J. Gorzny, L. Po-An, and M. Derka, "Ideal properties of rollup escape hatches," in *Proceedings of the 3rd International Workshop on Distributed Infrastructure for the Common Good*, DICG '22, (New York, NY, USA), p. 7–12, Association for Computing Machinery, 2022.

[151] "Starknet Escape Hatch Research." `https://community.starknet.io/t/starknet-escape-hatch-research/1108`. [Online; last accessed 20-April-2023].

[152] Vitalik Buterin, "State of research: increasing censorship resistance of transactions under proposer/builder separation (pbs)." Available: `https://notes.ethereum.org/@vbuterin/pbs_censorship_resistance`, 2022 [Online]. [Online; last accessed 24-April-2023].

[153] Ed Felten, "reducing-challenge-times-in-rollups." Available: `https://ethresear.ch/t/reducing-challenge-times-in-rollups/14997`, Mar. 08, 2023 [Online]. [Online; last accessed 24-April-2023].

[154] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges," 2019.

[155] "Maximal extractable value (mev)." Available: `https://ethereum.org/en/developers/docs/mev/`. [Online; last accessed 20-April-2023].

[156] Vitalik Buterin. Available: `https://medium.com/@VitalikButerin/i-feel-like-this-post-is-addressing-an-argument-that-isnt-the-actual-argument-that-mev-auction-b3c5e8fc1021`, May. 26, 2020 [Online]. [Online; last accessed 24-April-2023].

[157] Ed Felten, "Mev auctions considered harmful." Available: `https://medium.com/offchainlabs/mev-auctions-considered-harmful-fa72f61a40ea`, May. 26, 2020 [Online]. [Online; last accessed 20-April-2023].

[158] M. Kelkar, S. Deb, S. Long, A. Juels, and S. Kannan, "Themis: Fast, strong order-fairness in byzantine consensus." Cryptology ePrint Archive, Paper 2021/1465, 2021. `https://eprint.iacr.org/2021/1465`.

[159] M. Kelkar, F. Zhang, S. Goldfeder, and A. Juels, "Order-fairness for byzantine consensus." Cryptology ePrint Archive, Paper 2020/269, 2020. `https://eprint.iacr.org/2020/269`.

[160] "Mev minimization: Arbitrum (offchain labs)." Available: https://www.mev.wiki/solutions/mev-minimization/arbitrum-offchain-labs. [Online; last accessed 20-April-2023].

[161] Shutter Network, "Rolling shutter: Mev protection built into layer 2." Available: https://blog.shutter.network/announcing-rolling-shutter/, Apr. 04, 2022 [Online]. [Online; last accessed 20-April-2023].

[162] N. Gailly, K. Melissaris, and Y. Romailler, "tlock: Practical timelock encryption from threshold bls." Cryptology ePrint Archive, Paper 2023/189, 2023. https://eprint.iacr.org/2023/189.

[163] Chainlink, "Creating a highly scalable and mev-resistant defi ecosystem using arbitrum and fair sequencing services." Available: https://blog.chain.link/arbitrum-and-chainlink-fair-sequencing-services/#reducing_mev_with_fair_sequencing_services, Dec. 09, 2021 [Online]. [Online; last accessed 20-April-2023].

[164] "Risk analysis." Available: https://l2beat.com/scaling/risk. [Online; last accessed 25-April-2023].

[165] "Polygon zkevm security measures: The journey toward a safe, decentralized mainnet beta." Available: https://polygon.technology/blog/polygon-zkevm-security-measures-the-journey-toward-a-safe-decentralized-mainnet-beta. [Online; last accessed 25-April-2023].

[166] "Upgrade mechanism." Available: https://docs.zksync.io/userdocs/security/#upgrade-mechanism. [Online; last accessed 25-April-2023].

[167] Matter Labs, "Keeping funds safe: a 3-factor approach to security in zksync 2.0." Available: https://blog.matter-labs.io/keeping-funds-safe-a-3-factor-approach-to-security-in-zksync-2-0-a70b0f53f360, May. 21, 2021 [Online]. [Online; last accessed 25-April-2023].

[168] "Arbitrum dao: A conceptual overview." Available: https://docs.arbitrum.foundation/concepts/arbitrum-dao. [Online; last accessed 25-April-2023].

[169] "The future of arbitrum governance." Available: https://docs.arbitrum.foundation/why-governance#the-future-of-arbitrum-governance. [Online; last accessed 25-April-2023].

[170] E. M. Sifra, "Security vulnerabilities and countermeasures of smart contracts: A survey," in *2022 IEEE International Conference on Blockchain (Blockchain)*, pp. 512–515, 2022.

[171] D. He, Z. Deng, Y. Zhang, S. Chan, Y. Cheng, and N. Guizani, "Smart contract vulnerability analysis and security audit," *IEEE Network*, vol. 34, no. 5, pp. 276–282, 2020.

[172] S. Sayeed, H. Marco-Gisbert, and T. Caira, "Smart contract: Attacks and protections," *IEEE Access*, vol. 8, pp. 24416–24427, 2020.

[173] "Smart contract formal verification." Available: `https://ethereum.org/en/developers/docs/smart-contracts/formal-verification/`. [Online; last accessed 25-April-2023].

[174] "Audits." Available: `https://ethereum.org/en/developers/docs/smart-contracts/security/#audits`. [Online; last accessed 25-April-2023].

[175] "zksync security audits." Available: `https://docs.zksync.io/updates/security-audits/`. [Online; last accessed 25-April-2023].

[176] "Polygon security audits." Available: `https://github.com/0xPolygonHermez/zkevm-rom/tree/main/audits`. [Online; last accessed 25-April-2023].

[177] "Bug bounties." Available: `https://ethereum.org/en/developers/docs/smart-contracts/security/#bug-bounties`. [Online; last accessed 25-April-2023].

[178] "Security first: Polygon zkevm mainnet beta." Available: `https://polygon.technology/polygon-zkevm/security-first`. [Online; last accessed 25-April-2023].

[179] "Critical bug in ethereum l2 optimism." Available: `https://cryptoslate.com/critical-bug-in-ethereum-l2-optimism-2m-bounty-paid/`. [Online; last accessed 25-April-2023].

[180] R. Neiheiser, G. Inácio, L. Rech, C. Montez, M. Matos, and L. Rodrigues, "Practical limitations of ethereum's layer-2," *IEEE Access*, vol. 11, pp. 8651–8662, 2023.

[181] "Fuel v1." `https://docs.fuel.sh/v1.1.0/Introduction/Welcome.html`. [Online; last accessed 21-June-2023].

[182] Luca Donno, "Introducing stages — a framework to evaluate rollups maturity." Available: https://medium.com/l2beat/introducing-stages-a-framework-to-evaluate-rollups-maturity-d290bb22befe, June 20, 2023 [Online]. [Online; last accessed 20-June-2023].