Entangled Rollups: Multi-chain Interoperability Without Bridges

ZKM Research*

Feb 2024 - Draft Version 1.1.2

Abstract

Interoperability in blockchains is often implemented using a trusted bridge, a separate centralized or partially decentralized intermediary which validates and transfers cross-chain messages. In this work, we implement an interoperability protocol by judiciously entangling the underlying primitives under standard security assumptions of zkRollups, leveraging our state-of-the-art recursive zkVM. The Entangled Rollup protocol is trustless, and a step toward addressing liquidity fragmentation as well as simplifying the user and developer experience as major adoption barriers of the multi-chain world.

1 Introduction

1.1 Motivation: multi-chain interoperability

The complex set of trade-offs in the security, deployment and interactions costs, applications, and community of different blockchain infrastructures has resulted in the rapid proliferation of blockchain infrastructures such as layer 1s, layer 2s, and app-chains. While this multi-layer world of blockchain infrastructures undeniably provide value to the blockchain industry and users, it had introduced significant challenges in terms of liquidity fragmentation as well as cost of on-boarding for developers and users.

Cross-chain bridges [14], such as Wormhole [4] and Axelar, try to address this issue by enabling cross-chain transfer of assets and general message passing by introducing multi-signing committees for validating the cross-chain transactions. However, high fees for users, costs of the network, centralization of the external committee, and introduction of wrapped assets has been the major blockers to their adoption.

Recently, several projects are developing zero knowledge (ZK) [11] bridging solutions, including Succinct Labs, zkIBC (Electron Labs), and zkBridge (Polyhedra Network). These initiatives utilize zkSNARKs [13, 7] to enhance bridge designs. Central to their success is a light client protocol for efficient blockchain interaction and state synchronization.

---

*Ming Guo, Lucas Fraga, Stephen Duan, Jeroen van de Graaf, Pavel Sinelnikov, Lynn Zhong, Peyman Momeni (e-mail: first name dot first letter of last name at zkm dot io)

Some works propose slightly different designs that integrate zkRollup [8, 9] concepts into bridges. This approach faces challenges such as the need for larger circuit sizes than rollups and reducing on-chain storage and computational overhead, which are key to the effective functionality of ZK bridges. While integrating zero-knowledge proofs (zkProofs) into bridge designs significantly enhances decentralization and security, it introduces computational challenges, primarily due to the larger circuit sizes required.

In this paper we go in a different direction by exploring the subsistence of zkRollup architectures. We propose the concept of "entangled rollups" which allows multi-chain interoperability without relying on a separate entity. This architecture addresses challenges such as liquidity fragmentation while introducing less complexity for developers and users to deploy and interact.

Entangled rollups are deployed on all blockchain infrastructures, and their states are synced through state-of-the-art recursive zero-knowledge proofs. It is worth mentioning that the vision for entangled rollups is not limited to interoperability and asset transfer as this design enables a wide range of multi-chain applications and protocols which can leverage access to underlying infrastructures and ecosystems.

1.2 Outline of the paper

In Section 2, we introduce the necessary building blocks of Entangled Rollups to the general reader. In Section 3 we present our new approach, which we subject to an informal security analysis in Section 4. Section 5 discusses the properties we obtain, Section 6 draws a comparison with similar projects, while Section 7 contains the conclusion.

2 Background

2.1 Smart Contracts

Smart contracts are decentralized applications on blockchain networks, designed for automatic and deterministic execution logic. Their code specifies the execution logic, leading to predictable and consistent outcomes without human intervention.

Running on decentralized blockchains rather than centralized servers, smart contracts offer impartial, efficient, and secure outcomes, free from tampering. Their operation without central authority eliminates single points of failure and lowers attack risks, making them suitable for automating agreements across multiple parties with benefits like reduced risk, cost savings, and increased transparency.

Introduced by Ethereum, smart contracts have become foundational in Web3, fueling developments in decentralized finance (DeFi) [15], non-fungible tokens (NFTs) [12], gaming and more applications demonstrating their vital role in evolving decentralized applications.

2.2 zkRollups

zkRollups are layer 2 solutions that enhance the scalability of layer 1s by batching off-chain transactions and proving the correct execution of the batched transactions using zero-knowledge proofs that can be further verified in a smart contract on the underlying layer 1.

On-chain verifiability of the generated ZKPs removes the absolute blind trust in sequencers and ensures the correct execution of batched transactions.

The main components for zkRollups solutions include:

- **On-chain Contracts** are essential for operation, including a primary contract for rollup blocks and a verifier contract for checking zero-knowledge proofs, securing transactions processed off-chain.
- **Off-chain Virtual Machine (VM)** handles transaction processing and state maintenance off-chain to create ZK proofs for state transitions that are verified on-chain using a smart contract.

zkRollups offer a secure, efficient scaling method by utilizing Ethereum for data integrity and security, thus ensuring on-chain data availability and validity of state changes. Therefore zkRollups have emerged as one of the most secure mechanisms for implementing L1-L2 interoperability.

Note that here (and throughout this paper) the zk-prefix refers to the succinctness property, and not to the privacy property of these techniques. Though technically speaking this is an abuse of terminology, this usage of the term 'zero knowledge' has become standard practice in the field.

2.2.1 Implementing zkRollups with zkMIPS

One of the most important applications for ZKM's off-chain VM, zkMIPS, is the design of zkRollups. zkMIPS is capable of producing zkProofs for the correct execution of any program inside a standard MIPS VM. The produced zkProof can be optionally converted to any smart-contract friendly format, allowing the final proof to be verified on-chain. In the specific case of zkRollups, this feature can be used to verify the correct execution of a program that validates block-transitions.

Given a MIPS program and a proper input to it, zkMIPS compiles the execution of this program under this input into a Plonky2 [1] proof. If the proof is destined to be verified on-chain, one final procedure can convert the Plonky2 proof into a Groth16 [10] proof. The final Plonky2 proof size and time are adjustable, while the Groth16 proof size and time depend on the statement (Plonky2 proof verification) being proved, meaning the final on-chain proof can be as small as we need it to.

To improve efficiency of on-chain proof verification we do the following: during off-chain proof generation both the MIPS program and its inputs are written in a succinct representation of the initial memory state over which the VM starts running. In practice, this means that any program can be verified on-chain given a proper representation of its initial state, allowing for any program to be verified on-chain with roughly the same amount of resources. in particular, the succinct memory representation is a Merkle root with memory pages as Merkle node and, in the case of Ethereum, the main on-chain verification resource is gas.

2.3 Cross-chain interoperability

Cross-chain technology enables interoperability between distinct blockchain networks, allowing for seamless transfer of data and assets. To accomplish this goal, it must address the challenge of different blockchains operating with unique rules and protocols. See [14] for a good example.

The key functionalities of cross-chain interoperable solutions include:

- **Data Sharing** enables cross-chain communication, essential for developing applications that integrating data from multiple blockchains.
- **Asset Swap** allows for the transfer of digital assets across blockchain platforms, increasing asset liquidity and flexibility, which greatly improve UX.
- **Scalability and Performance Enhancement** utilizes the strengths of various blockchains for improved functionality and performance, facilitating more efficient system throughput.

2.4 Interoperability Trilemma of Blockchain Bridges

In trying to achieve interoperability through cross-chain bridges, we want to reconcile three fundamental properties: trustlessness, extensibility, and generalizability. [6]

- **Trustlessness** refers to the need for cross-chain bridges to offer the same security guarantees as the underlying blockchain Layer 1, without introducing additional trust assumptions. This means that users do not need to place extra trust in any intermediaries or third parties in order to ensure the security and reliability of the entire system.
- **Extensibility** refers to the ability to connect and interact with other blockchain networks. This capability allows for the free flow of assets and data between different blockchains, thereby enhancing flexibility and efficiency of the entire blockchain ecosystem.
- **Generalizability** refers to the cross-chain bridge's capacity to handle more generic applications. This includes not only common transactions, but also a wide array of applications like smart contracts, NFT transfers, authentication, etc. By supporting a broader range of applications, bridges enhance their practical value.

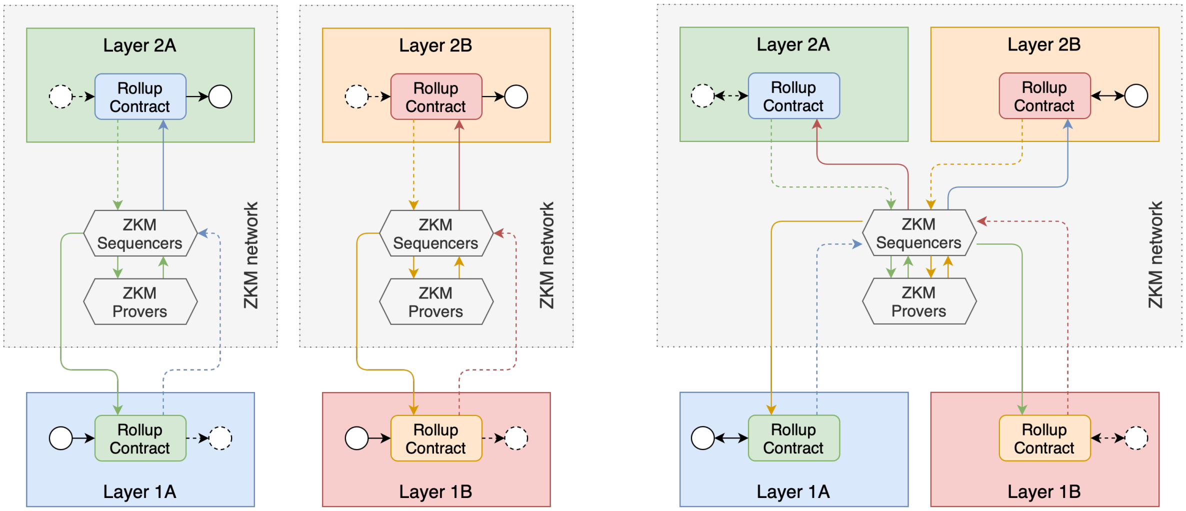
To achieve efficient and secure cross-chain bridge interoperability, one needs to find a proper balance between these three properties.

3 Our proposal: Entangled rollups

3.1 Key insight: entangling two rollups

Our key contribution is this: zkRollups can be seen as an interoperability mechanism between L1s and L2s, allowing us to implement the functionality of a cross-chain bridge but without creating one. To understand how this is possible, suppose we have two different blockchains called A and B respectively, each with two layers called Layer 1A/2A and Layer 1B/2B respectively. In addition to this, suppose there exist zkMIPS-based zkRollups from Layer 2A to Layer 1A and from Layer 2B to Layer 1B.

Since the protocol and entities that implement the rollup for A and B are similar, it is possible to entangle these rollups, meaning we can deposit from one chain to any other chain in the same rollup network. For instance, Layer 2A can withdraw funds to Layer 1B instead of to its ‘parent’ Layer 1A, as



(a) Two L1/L2 blockchains

(b) Two entangled L1/L2 blockchains

Figure 1: Entangling blockchains

well as Layer 1A can deposit funds to Layer 2B instead of to its 'child' Layer 2A. In the same way, Layer 2A can transfer funds to its counterpart Layer 2B through a withdrawal followed by a deposit, as well as Layer 1A can transfer funds to its counterpart Layer 1B through a deposit followed by a withdrawal.

The key component to this feature is the existence of a general-purpose proving mechanism common to all zkRollups, meaning a proof can be produced for one blockchain and verified on another. In this context, since every transaction on one Layer 2 will be eventually rolled-up to its respective Layer 1, the proof generated for any transaction in this chain can be accordingly rolled-up to any other chain.

This **Entangled Rollup** can be used to trigger actions on one chain based on actions that happened on another chain, using a zk proof for some action on the second chain to justify actions triggered on the first one. We call the chain where the **trigger action** occurred the **Source chain** (or Src chain/L1/L2), and the chain where the **result action** will occur the **Destination chain** (or Dest chain/L1/L2).

We assume that every party engaged in Dest L1 trusts the entities implementing its own zkRollup mechanism and therefore the zk proofs, so there is no need to verify the cross-chain transaction coming from Src L1. Note that the entities implementing the zkRollup will only bring this transaction to Dest L1 if it exists on Src L1. However, we cannot assume every party engaged in Dest L2 trust these L2 entities, so a cross-chain transaction arriving to Dest L1 must be verified on-chain, together with its Src L1 counterpart. These two proofs (from Src and Dest L1) can be combined and verified on-chain (Dest L2) in the same contract call that executes the zkRollup of the Dest L1 side of the cross-chain transaction.

One possible pair of actions that can be implemented is the minting (or unlocking) of some asset on Dest L1 based on the burning (or locking) of some asset on Src Layer 2, allowing for the implementation of cross-chain bridging features without any significant architectural modification. In this specific use case, the Dest L1 state validation algorithm must be modified to allow minting subjected to a successful Src Layer 2 state validation. To this end, the Src Layer 2 state validation proof generated during its rollup process can be used during the Dest L1 state validation proof generation to mint (or unlock) assets on the Dest L1.

The choice between burning/minting or locking/unlocking is up to the rollup designer, but impacts every cross-chain transaction involving the blockchain. The burning/minting approach can be implemented through a fixed burning address (e.g. a null address or any randomly generated address to which no private key is known) to send assets being transferred to other blockchains, and a special function on the rollup contract to mint assets being transferred from other blockchains. On the other hand, the locking/unlocking approach can be implemented through smart contracts responsible for holding assets transferred to other blockchains and releasing assets transferred from other blockchains.

We call a smart contract which implements the locking/unlocking functionality for cross-chains transactions a **Shadow Contract**, since it 'follows' the behavior of other blockchains involved in the Entangled Rollup network. Collectively, the set of Shadow Contracts from all blockchains involved in the Entangled Rollup act as a liquidity provider for the network. For this reason, we consider the locking/unlocking approach more didactic and choose it as the default for the rest of this document.

One advantage of Entangled Rollups over bridges is that no new nodes have to be created, as all off-chain nodes involved already exist and participate in the involved zkRollups by assumption. Besides, its functionality is completely general by design because transaction data is handled and proven off-chain by the same nodes that operate the rollups. In the remainder of this section we provide more details about Entangled Rollup, starting with an extensive step-by-step description of cross-chain transaction.

3.2 A step-by-step description

In a zkRollup architecture, off-chain sequencer and prover nodes sequence transactions and generate ZK proofs for every new L2 batch, respectively. In the Entangled Rollup architecture, these nodes continue to exist but now there are also relayer nodes to pass cross-chain transactions and ZK proofs to other blockchains involved in the transactions. In this setting, an L2-L2 cross-chain transaction is processed in the steps described below and illustrated in Figure 2.

1. *Src Account* triggers the withdraw procedure by submitting a transaction $\mathtt{tx}_1$ to *Rollup Contract* on *Src L2*.

   We require $\mathtt{tx}_1$ to be formatted in a way to specify *Dest L2*, the data that must be carried on the *Dest L2* transaction and the asset that must be released on the *Dest L1* transaction.

2. The withdraw transaction $\mathtt{tx}_1$ is processed, which requires that:
   (a) *ZKM Sequencers* read $\mathtt{tx}_1$ from *Rollup Contract* on *Src L2*.
   (b) *ZKM Provers* produce a proof $\mathtt{zk}_2$ for the block containing $\mathtt{tx}_1$.
   (c) *ZKM Sequencers* rollup $\mathtt{tx}_1$ to *Rollup Contract* on *Src L1* by sending $\mathtt{zk}_2$ to it.
   Rollup Contract updates the *Src L2* state if $\mathtt{zk}_2$ passes the on-chain verification.

3. *Rollup Contract* on *Src L1* concludes the withdraw procedure and triggers the cross-chain procedure by submitting a transaction $\mathtt{tx}_3$ to *Shadow Contract*.

   We require $\mathtt{tx}_3$ to be formatted in a way to represent any data attached to $\mathtt{tx}_1$.

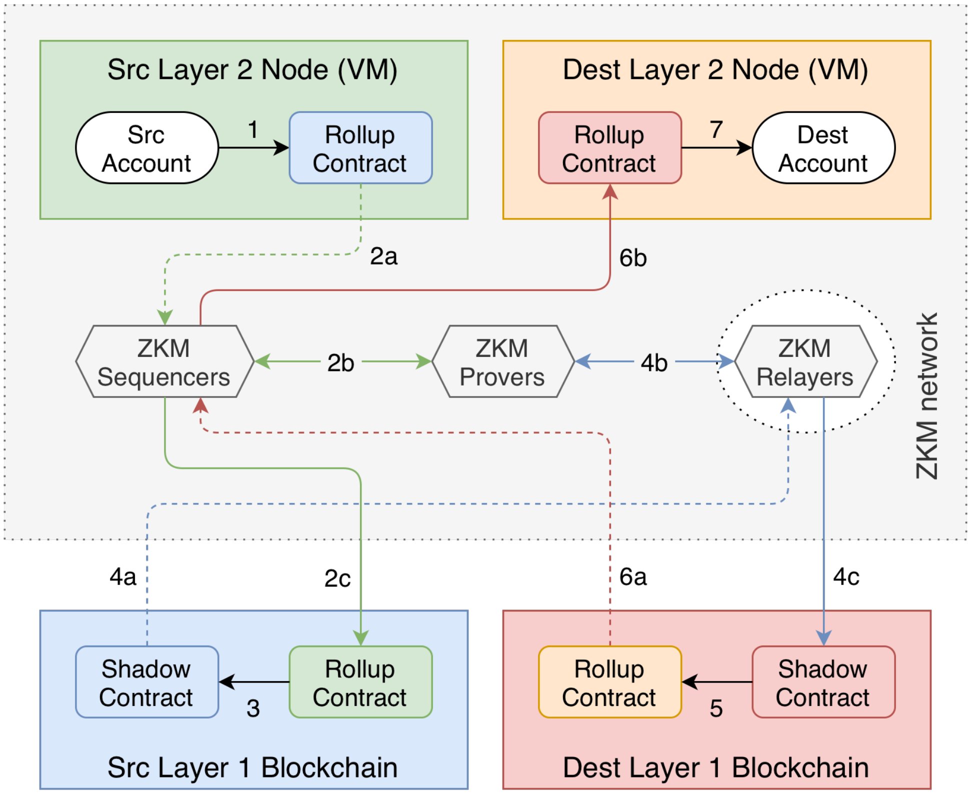


Figure 2: Entangled Rollup scheme

4. The cross-chain transaction is processed, which requires that:
   (a) *ZKM Relayers* read $\texttt{tx}_3$ from *Shadow Contract* on *Src L1*.
   (b) *ZKM Provers* produce a proof $\texttt{zk}_4$ for the block containing $\texttt{tx}_3$.
   (c) *ZKM Relayers* pass $\texttt{tx}_3$ and $\texttt{zk}_4$ to *Dest L1* via *Shadow Contract*.
5. *Shadow Contract* on *Dest L1* concludes the cross-chain procedure and triggers the deposit procedure by submitting a transaction $\texttt{tx}_5$ to *Rollup Contract*.

   We require $\texttt{tx}_5$ to be formatted in a way to represent any data attached to $\texttt{tx}_3$.
6. The deposit transaction $\texttt{tx}_5$ is processed, which requires that:
   (a) *ZKM Sequencers* read $\texttt{tx}_5$ from *Rollup Contract* on *Dest L1*.
   (b) *ZKM Sequencers* pass $\texttt{tx}_5$ to *Rollup Contract* on *Dest L2*.
7. *Rollup Contract* on *Dest L2* concludes the deposit procedure by submitting a transaction $\texttt{tx}_7$ to *Dest Account* on *Dest L2*.

   We require $\texttt{tx}_7$ to be formatted in a way to carry any data attached to $\texttt{tx}_5$.

What makes this design possible is zkMIPS proving architecture. By design, all input data (including the program being proven and its parameters) are encoded in a succinct representation of the initial state over which the input program is initialized, as described in Section 2.2.1. As a result, the final on-chain verifier must check two things: whether the correct transaction data is encoded into the right memory location, and the proof itself. This way, the zkMIPS on-chain verifier embedded in the rollup contract on Dest L1 can verify the correctness of $\texttt{tx}_3$, $\texttt{tx}_5$, their equivalence and a few more properties by retrieving public proof parameters stored in the contract. These public parameters must be updated every time a change happens in Src L2, Src L1 or Dest L2 state transition functions. For details of the properties these proofs must show and how they can be composed, see Appendix A.

4 Informal security analysis

The security of the asset transfer procedure involving ZK proofs, Rollup contracts, and Shadow Contracts across Source (Src) and Destination (Dest) chains, as outlined, depends on several foundational principles of blockchain security and cryptography. In essence, it is anchored in (a) the principles of ZK proofs for verification, (b) the transparent control of assets through Shadow Contracts, and (c) the rigorous adherence to asset conservation laws, all of which collectively form a robust framework for secure and verifiable asset transfers across blockchain networks. In this section we break down the key steps of the procedure described in Section 3, and assess its security.

Security of Src Chain Rollup Withdrawal Through ZK Proofs The use of a single proof system in steps 2b and 4b ensures the withdrawal from the Src Rollup is secured under the same cryptographic assumptions. This uniformity allows both L2 contracts to independently verify the proof without additional requirements. Thus, integrity of withdrawal is cryptographically linked to the ZK proof validity.

Security of Dest Chain Rollup Deposit Through ZK Proofs Similarly, the deposit into the Dest Chain Rollup steps 6a and 6b is secured by reusing the zkMIPS proof system. This shared proof ensures that the deposit is cryptographically consistent with the withdrawal, maintaining cross-chain asset integrity. As a consequence, the asset transfer is secure and verifiable, with deposit legitimacy tied to the proof integrity, thus preventing double-spending and asset manipulation during the transfer.

Control by Shadow Contract The fact that the native token is controlled exclusively by the Shadow Contract adds an additional layer of security. This transparent control within the decentralized framework ensures that unauthorized control of the token is impossible under standard cryptographic assumptions.

Withdrawal and Deposit Verification The procedure of withdrawal and deposit verification is a critical component of the asset transfer mechanism, ensuring the integrity and conservation of assets across different layers and networks. This mechanism involves several key steps and principles:

- **Verification procedure:** The Src Shadow Contract on Dest L2 plays a pivotal role in verifying the withdrawal of an asset from the Src L1 to the Shadow Contract on Src L2. This step is crucial for maintaining the traceability and integrity of the asset as it moves across layers.

- **Closed Loop of Asset Transfer:** By verifying the withdrawal before initiating the deposit, the system creates a closed loop that ensures every asset leaving the Src L1 has a corresponding entry on the Dest L1. This verification procedure prevents the duplication of assets and ensures that the total supply remains constant, adhering to the principle of asset conservation.
- **Conservation of Assets:** The principle of asset conservation is upheld through this meticulous verification procedure. By ensuring that each asset withdrawn is matched with an equivalent asset deposited, the system prevents the creation or destruction of assets in the transfer procedure. This is crucial for maintaining the balance and value of assets across different blockchain networks.
- **Security Implications:** The verification procedure ensures asset conservation and enhances the overall security of the asset transfer mechanism. By requiring proof of withdrawal before a deposit can be made, the system minimizes the risk of unauthorized or fraudulent transfers.

5 Desirable properties: back to the Interoperability Trilemma

Entangled Rollups represent a novel Layer 2 to Layer 2 interoperability protocol, effectively addressing the properties listed in the trilemma of cross-chain interoperability in blockchain technology.

Trustlessness: Entangled Rollups possess a trustless nature, a feature inherited from its zkRollup architecture. It relies solely on a shared sequencer, which we assume to be decentralized. This property is crucial to ensure that the protocol operates *without* the need for additional trust in a single authority or intermediary (bridge), and is fundamental to enhancing the security and integrity of cross-chain transactions.

Extensibility: In terms of extensibility, the Entangled Rollup Protocol benefits significantly from its shared sequencer pool. This property allows for scalable and efficient interactions between different Layer 2 platforms. The shared sequencer pool enables a seamless and streamlined procedure for managing and verifying transactions across multiple chains, thereby facilitating a more connected and interoperable blockchain ecosystem.

Generalizability: Our architecture is not confined to specific types of transactions or data. Instead, it offers a far-reaching range of applications:

- Entangled Rollups can bridge to any Ethereum-native token, including ERC20 and ERC721 tokens.
- Entangled Rollups allow compatibility with various blockchain standards and protocols. Whether it is Ethereum or other newer blockchain platforms, Rollup bridges can effectively facilitate the transfer of data and assets between these diverse systems.
- Entangled Rollups are capable of processing and transferring a variety of data types, including the execution of multi-chain smart contracts, the exchange of authentication information, the transfer of NFTs, among others.

To put it differently, ZKM's Entangled Rollups enable deploying smart contracts in all ecosystems in just one click. This enables developers to have access to liquidity, to users, and to unique technical features of the underlying ecosystems without going through the learning curve and costs of deployment on each of them separately. Moreover, Entangled Rollups can enable smart contracts of the same application on different ZKM nodes to sync their states with each other in a fast and efficient way, thus offering possibilities for designing new DeFi protocols.

6 Comparison with related interoperability approaches

Multi-chain rollup proposed by [5] introduces a zkRollup architecture which requires deploying the contracts on multiple L1 blockchains and L2 networks at the same time, thus is called a multi-chain zkRollup. In simple terms, a multi-chain rollup deployed on all ecosystems, and these instances of these rollups should be synced together as the ZK verification part for all instances is only done in a single primary chain.

Even though this design shares a similar vision with ZKM, the main difference is that zkProof verification of each ZKM instances are done independently whereas [5] is verifying all zkProofs in a primary chain and syncing them in an additional step. While this design may introduce optimizations, it introduces an additional synchronization round which requires two cross-chain messages through a light client network. The light client network itself should be powered by cross-chain message passing solutions such as LayerZero or Chainlink, which introduces additional security assumptions and significant delays.

Optimistic cross-chain orders proposed by [3] supports cross-chain trading, allowing users to swap assets across various blockchains through settlement oracles and cross-chain messaging protocols. The system's security largely depends on the architecture of the oracles used: centralized oracles introduce a single point of failure and reduced security, while decentralized, committee-based oracles enhance security but may slow down transactions, affecting user experience negatively. It also uses an off-chain Request For Quote system (RFQ) to set the initial prices for Dutch orders by gathering price quotes from multiple fillers. This approach could favor the execution of swaps at the lowest quoted price, which might open up arbitrage opportunities, raising questions about the quoting system's fairness, potentially favoring certain participants unfairly.

Compared to our solution, it encounters more pronounced challenges, such as heightened security risks, longer operational delays, and complications arising from performance issues and arbitrage-related pricing imbalances. Despite its pioneering method of facilitating cross-chain transactions, the reliance on oracles and the RFQ system might undermine [3]'s effectiveness and security, making it potentially less robust and efficient than our solutions.

Aggregation layer proposed by [2] aims to establish a universal state across all chains by employing recursive ZK proofs, which include proof aggregation, optimistic batch confirmation, and atomic cross-chain interactions. These batches, which are verified within minutes, are subsequently posted to the Ethereum blockchain at intervals ranging from 30 to 60 minutes. This infrequent posting schedule inherently causes delayed cross-chain messaging, leading to significant latency issues.

Optimistic batch confirmation is used as a strategy to alleviate these latency concerns. However, integrating multiple systems introduces potential for numerous unforeseen complications. For example, partial rollbacks could precipitate system failures; moreover, malicious transactions on blockchains characterized by low gas fees might engender transaction congestion on other blockchains. Consequently, considering the possibility of blockchain reorganizations (reorgs) subsequent to each proof generation, coupled with the substantial expense associated with generating these proofs, the security and practicality of such design is questionable.

7 Conclusion

In this paper we presented a novel multi-chain interoperability architecture called Entangled Rollups. The key idea of Entangled Rollup is the re-utilization of the zkRollup architecture, the foundation for designing Layer 2 to Layer 1 interoperability. By applying the same principles to different L1 and L2 one can guarantee that the integrity and security of transactions are maintained across layers and ensure that data and asset transfers are both secure and verifiable, without the need to introduce an additional trusted entity (bridge).

We want to emphasize that this first version presents one possible architecture for Entangled Rollups. We are aware that other, similar scenarios exist and intend to elaborate on them in future versions of this paper.

References

[1] Plonky2. https://github.com/0xPolygonZero/plonky2.

[2] Polygon aggregation layer. https://docs.polygon.technology/learn/agglayer/.

[3] Uniswapx. https://uniswap.org/whitepaper-uniswapx.pdf.

[4] Wormhole. https://wormhole.com.

[5] zklink - a multi-chain rollup infrastructure based on zero-knowledge technology. https://zk.link/zkLink-whitepaper.pdf.

[6] Bhuptani a. a new paradigm for crosschain communication. https://medium.com/connext/optimistic-bridges-fb800dc7b0e0. 2023.

[7] Thomas Chen, Hui Lu, Teeramet Kunpittaya, and Alan Luo. A review of zk-snarks. *arXiv preprint arXiv:2202.06877*, 2022.

[8] Rex Fernando and Arnab Roy. Poster: Wip: Account zk-rollups from sumcheck arguments. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 3594–3596, 2023.

[9] Vincent Gramoli, Len Bass, Alan Fekete, and Daniel W Sun. Rollup: Non-disruptive rolling upgrade with fast consensus-based dynamic reconfigurations. *IEEE Transactions on Parallel and Distributed Systems*, 27(9):2711–2724, 2015.

[10] Jens Groth. On the size of pairing-based non-interactive arguments https://eprint.iacr.org/2016/260.pdf.

[11] Jens Groth. Short non-interactive zero-knowledge proofs. In *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security*, pages 341–358. Springer, 2010.

[12] Logan Kugler. Non-fungible tokens and the future of art. *Communications of the ACM*, 64(9):19–20, 2021.

[13] Alexandre Miranda Pinto. An introduction to the use of zk-snarks in blockchains. In *Mathematical Research for Blockchain Economy: 1st International Conference MARBLE*, pages 233–249. Springer, 2020.

[14] Tiancheng Xie, Jiaheng Zhang, Zerui Cheng, Fan Zhang, Yupeng Zhang, Yongzheng Jia, Dan Boneh, and Dawn Song. zkbridge: Trustless cross-chain bridges made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 3003–3017, 2022.

[15] Dirk A Zetzsche, Douglas W Arner, and Ross P Buckley. Decentralized finance (defi). *Journal of Financial Regulation*, 6:172–203, 2020.

A Properties proven in cross-chain transactions

A.1 Properties proven on destination Layer 1

Given a Src L1 $\texttt{chain}^{\texttt{SrcL1}}$, a shadow contract address $\texttt{shadow}^{\texttt{SrcL1}}$, an asset $\texttt{asset}^{\texttt{SrcL1}}$, an amount $\texttt{amount}^{\texttt{SrcL1}}$ and a list of inputs $\texttt{input}^{\texttt{SrcL1}}$, the zkProof $\texttt{zk}_4$ from Page 7 must prove:

1. there exists $\texttt{tx}^{\texttt{SrcL1}}$ such that
    it withdraws $\texttt{amount}^{\texttt{SrcL1}}$
    of $\texttt{asset}^{\texttt{SrcL1}}$
    with $\texttt{input}^{\texttt{SrcL1}}$
    to $\texttt{shadow}^{\texttt{SrcL1}}$
    on $\texttt{chain}^{\texttt{SrcL1}}$;
2. $\texttt{tx}^{\texttt{SrcL1}}$ was included in some Src L1 block $\texttt{block}^{\texttt{SrcL1}}$;
3. $\texttt{block}^{\texttt{SrcL1}}$ was processed and generated some Src L1 state $\texttt{state}^{\texttt{SrcL1}}$.

In proving $\texttt{tx}^{\texttt{SrcL1}}$, ZKM Provers must provide the data that characterizes this transaction, namely $\texttt{amount}^{\texttt{SrcL1}}$, $\texttt{asset}^{\texttt{SrcL1}}$ and $\texttt{input}^{\texttt{SrcL1}}$. The algorithm that verifies this transaction characterizes $\texttt{chain}^{\texttt{SrcL1}}$ and $\texttt{shadow}^{\texttt{SrcL1}}$, and the public parameters for the zkProof that proves it are already stored in the rollup contract on Dest L1, so there is no need to provide this information. In proving $\texttt{block}^{\texttt{SrcL1}}$ and $\texttt{state}^{\texttt{SrcL1}}$, ZKM Provers must provide other transactions included in $\texttt{block}^{\texttt{SrcL1}}$ as well as any other data required for the state transition function of that specific zkRollup architecture.