Escrow IBC: An IBC based protocol for fast trust-minimized bridging between Optimistic Rollups

Yishay Harel, Omri Dagan, Lior Zilpa

yishay@dymension.xyz

March 21, 2023

Abstract

This paper presents escrow-IBC (eIBC), an Inter-Blockchain Communication (IBC) based protocol designed to facilitate trust-minimized and instantaneous token transfers across Optimistic Rollups. Optimistic Rollups are emerging as the most promising scaling solution for blockchain technology; however, they suffer from several issues, including the fraud dispute window period and the Verifier's Dilemma. eIBC leverages the IBC protocol to enable users to move their fungible assets instantly from a source rollup to a specialized settlement layer in a trust-minimized manner. This opens up an instant routing method to any destination rollup connected with the settlement layer, allowing for improved user experience and developer benefits.

1 Introduction

Optimistic Rollups,[1] have emerged as a popular scaling solution for blockchains, offering a range of benefits. However, they suffer from various drawbacks, including the fraud dispute window period which prevents instant transacting with other economic zones. Another significant issue is the Verifier's Dilemma,[2] which poses an economic challenge for those allocating resources to check the work of rollups. While RollApps (Dymension rollups) are integrating with the IBC[3] ecosystem via the Dymension Hub, moving assets between rollups and a consensus-full layer-1 network is currently slow due to the fraud-proof periods that disqualify the settlement layer from instantly trusting the rollup state, hampering the user and developer experience. The eIBC protocol proposes a trust-minimized and instantaneous method of transferring tokens between Optimistic Rollups by leveraging the IBC ecosystem. It allows users to move their fungible assets from a source rollup to a settlement layer, opening up an instant routing method to any destination rollup connected with the settlement layer. This paper describes the eIBC protocol and its potential to improve the user experience and developer benefits of Optimistic Rollups.

This paper is premised on the following assumptions:

1. Users are willing to pay an additional fee for instant withdrawal of tokens.
2. Verifying a rollup eliminates the risk of rollback for the verifier.
3. Verifiers who encounter fraud would submit a proof.

Based on these assumptions, the proposed solution involves enabling fast withdrawals for users willing to pay verifiers for assuming a rollback risk while ensuring users will not need to place trust in any intermediary entities. Additionally, as rollback risk for a verifier is removed with proper state verification, verifiers receive a 'free option' to profit from malicious sequencers via fraud proof submission.

2 eIBC

In this section, we introduce the eIBC protocol, outlining the roles of the two actors involved - Alice, a user on a RollApp, and Bob, a relayer in the IBC ecosystem. eIBC wraps a standard pending finalization IBC message from RollApp to settlement layer (which will confirm post dispute-window) in order to achieve instant trust minimized transferring. As a wrapper the protocol introduces a similar flow to the IBC protocol in which, Alice submits an eIBC transaction on the source RollApp, and Bob relays the message from the RollApp to the settlement layer where it's verified. If Bob decides to fulfill the request, he sends the required tokens to the eIBC middleware module on the settlement layer, which in turn instantly delivers the said tokens to Alice on the settlement layer and re-routes Alice's original pending IBC message to Bob's address.

Flow

1. Alice initiates an eIBC transaction by submitting a request on the source RollApp to withdraw a specific amount of tokens (e.g., 10 DYM) from her account.
2. Bob, a relayer in the ecosystem, receives the eIBC message from the RollApp and relays it to the settlement layer which softly verifies and escrows the underlying IBC message.
3. Bob, who also has tokens on the destination settlement chain, decides to fulfill Alice's eIBC request by providing the requested tokens (10 DYM) to the eIBC module on the settlement layer.
4. The eIBC module on the settlement layer delivers the requested tokens to Alice and re-routes the original pending finalization tokens to Bob.

Several key observations can be made regarding the proposed eIBC solution. First, assuming that Bob has verified the RollApp chain, he bears no added risk as an active relayer and liquidity provider. In the event of an invalid state transition by the sequencer, Bob is expected to submit a fraud proof to profit from the situation, while refraining from delivering the tokens to Alice to avoid exposure to potential rollback risks and associated damages. Second, as more market-making actors like Bob join the ecosystem, more RollApps are expected to be verified by other actors and economic forces. Nevertheless, the proposed solution requires Alice, as a user on the RollApp, to wait until a single liquidity provider like Bob decides to fulfill her token request on the settlement layer. However, assuming Alice is willing to pay an extra fee, it is expected that economic forces will eventually drive the ecosystem towards a more efficient scenario which will be driven by custom applications developed on top of the eIBC protocol.

A potential issue arises when Bob, which acts both as the eIBC message relayer and the liquidity provider has limited funds and must wait for them to become available on the settlement layer post-dispute window. Consequently, Bob could publicly invite individuals who trust his ability to properly verify RollApps and possess tokens on the settlement layer to pool liquidity with him, with the promise of sharing the withdrawal fee. Given this scenario, the following application could evolve on top of the eIBC protocol.

3 Application on eIBC

In this proposed application, the following actors are defined: Alice, a user on the RollApp; Bob, a relayer in the ecosystem and an experienced node operator; and Carrol, an investor looking to get a return on her capital. The flow of the solution is as follows:

1. Bob creates a public 'verifier liquidity pool' that utilizes the pooled tokens on the settlement layer to provide instant liquidity for eIBC users of a specific source RollApp. Bob runs a full node of the RollApp and is not concerned with invalid state updates, as he will actively submit a fraud-proof in any case of fraud.
2. Carrol, an investor looking for returns, decides to pool her funds with Bob's 'verifier liquidity pool' for the source RollApp.
3. Alice submits an eIBC transaction on the source RollApp, requesting to withdraw tokens. (e.g 10 DYM)
4. Bob relays the eIBC message from the RollApp to the settlement layer where it's softly verified.
5. Bob's liquidity pool can now provide liquidity to Alice by delivering the tokens required on the settlement layer to the eIBC module.
6. The eIBC module is able to distribute the escrowed tokens to Alice and re-route Alice's original pending finalization tokens to Bob's liquidity pool.
7. Carrol is able to withdraw profit from Bob's liquidity pool.

By utilizing Bob's services, the proposed solution ensures that Alice can receive her requested tokens instantly, from an aggregated source of liquidity. This is made possible by the existence of an application that provides the necessary liquidity, allowing the eIBC protocol to verify the delivery of tokens and distribute them accordingly.

The proposed solution also introduces 'Verifier Pools', an application of liquidity pools that depend on the verifier to mitigate risk. Despite offering profitable opportunities for node operators and investors, Verifier pools also entail risk for liquidity providers who must place trust in the verifier's ability and integrity. Utilizing these pools can facilitate a healthy liquidity market, enabling instant transactions between RollApps and increasing external verification rates, ultimately reducing fraud attempts.

The eIBC protocol addresses both the verifier dilemma and the fraud proof window, leveraging the shared bridging environment and the battle-tested IBC infrastructure to support cross-rollup bridging in a seamless, trust-minimized way.

---

[1] Konstantopoulos, G. (2021, January 27). (Almost) Everything you need to know about Optimistic Rollup. Paradigm Research. https://research.paradigm.xyz/rollups

[2] Felten, E. (2021, December 11). *The Cheater Checking Problem: Why the Verifier's Dilemma is Harder Than You Think*. Medium. https://medium.com/offchainlabs/the-cheater-checking-problem-why-the-verifiers-dilemma-is-harder-than-you-think-9c7156505ca1

[3] *What is IBC? | Developer Portal*. (n.d.). https://tutorials.cosmos.network/academy/3-ibc/1-what-is-ibc.html