HyperLoop: Rationally Secure Cross-chain Bridge

Supra Research

Abstract

In this work, we present Hyperloop, an efficient *multi-sig* blockchain bridge. We solve the 'bridge problem' with a committee requiring signatures from a simple majority of nodes non-interactively rather than an interactive protocol requiring a super-majority which is typical among most existing bridges. This yields an economic, computation and communication-efficient design wherein, with a smaller number of bridge nodes and for a given bridge value transfer limit, a smaller cumulative stake value may be employed for similar crypto-economic security.

Considering the assumption of honesty among nodes to be a stretch in real-world settings, especially when high-value financial assets are involved, we study the 'bridge problem' in more realistic *rational* and *rational-malicious* settings. Here, the bridge nodes, as rational parties, can deviate from the protocol arbitrarily for an economic incentive. We introduce a *sliding window* limiting mechanism that limits the total value that can be authorized by the bridge nodes in a given period of time. We introduce a new role of *whistle-blowers* to monitor the activities of bridge nodes, and raise complaints to a governance role, typically fulfilled by an Audit Decentralized Autonomous Organization (Audit-DAO), in case of malicious activity. Using game theory, we derive a robust incentive-penalty mechanism in these rational settings so that both the safety and liveness characteristics of the bridge are preserved and optimized.

1 Introduction

Blockchains have introduced "decentralized trust" across a variety of services. As the rise of the adoption of Decentralised Finance (DeFi) protocols demonstrates, services from the finance domain have been the "killer application" of blockchains. As per DefiLlama, as of 24th June 2023, around USD $45 billion have been locked on all DeFi protocols in all the blockchains put together. Though Defi has been a main driver, the seamless and trustless connectivity offered by blockchains is catalyzing adoption across various sectors as awareness of the technology grows.

As the circulation of assets and value represents the cumulative global liquidity of the world's financial systems, naturally, financial assets need to be bridged across blockchains. Cross-chain bridge protocols provide the infrastructure and functionality necessary to transfer digital assets across blockchains. Again, as per DefiLlama on 24th June 2023, roughly $61 million of value was bridged at the time of publication. This shows the utility and ubiquity of cross-chain bridges.

As the bridge protocols gain popularity, the attacks on them have also seen a rise. Chainanalysis reports that bridge attacks have hit USD 2 billion as of 2022 [9]. These attacks on bridges show that the bridges form the weakest link in the ecosystem. Hence the security of bridge protocols is fundamental and paramount in establishing confidence among dApp developers across blockchains to do high-volume asset transfers. Furthermore, regular hacks also hinder blockchain adoption as end users lose confidence in spite of seeing value in these services.

Decentralized cross-chain bridges are typically realized using a committee of "staked" nodes acting as a bridge between two chains: the source chain and the destination chain. These nodes may interact with both chains and each other to produce a collective witness or a threshold signature conveying the events of the source chain to the destination chain. Traditionally, it has been assumed that these nodes follow the honest-malicious model, where a certain number of nodes are assumed to be always honest. This threshold assumption may not always hold in the real world, and the parties may collude for economic reasons. This

makes many of the existing bridge mechanisms extremely vulnerable to collusion attacks. If a sufficient number of nodes realizing the bridge decide to collude, they can transfer funds to any address of their choice.

Rationality

In this work, we view bridge protocols from a more realistic *rational* model. So far, bridges have been studied in the *honest-malicious* model wherein a subset of nodes are always assumed to be incorruptible and honest. Naturally, this is far from realistic and the realities of real-world threat models. A detailed study and a rigorous analysis of bridges from the perspective of *economic rationality* is necessary and imminent. Hence we undertake this work of building a cross-chain bridge and studying it *from the lens of rationality*.

Our approach constitutes three sets of actors: *staked bridge* nodes, *whistle-blower* nodes, and a *monitoring* Decentralised Autonomous Organisation (Audit-DAO). We analyze and derive the values of the incentives and penalties in the bridge protocol to make inert any collusion amongst bridge nodes and make attempts to usurp the protocol uneconomical. The critical insight to meet this objective is to have a set of incentivized whistle-blower nodes constantly monitor the activities of the bridge nodes and raise complaints to a monitor Audit-DAO in the case of demonstrable discrepancies. Upon proven discrepancy, the network withholds the stake of the colluding bridge nodes. We then have non-collusion as an equilibrium strategy for the bridge nodes, further ensuring the protocol's security. The architecture of HyperLoop is provided in Figure 1.

The value of stakes of the bridge nodes naturally limits the maximum value of allowable transfers. If not, bridge nodes can easily collude and sign off an arbitrary transfer to themselves. Hence, there must be some limit to the value they can transfer. Typically bridge protocols use a notion of *rate-limiting* where a limit L is imposed on the allowed transfer in a window of time.

However, the bridge nodes can maneuver around this limit and time a peak transfer of L near the end of a window and another L at the beginning of the following window, essentially transferring $2L$ around the edges of consecutive windows. To avoid this maneuvering, one has to set the rate limit of L to $1/2$ of the expected value for the same stake value of the bridge nodes. A *sliding-window* mechanism overcomes this issue. Pertinent questions, then, are to determine the value limit and the window length. As we see in Section 7, the sliding-window value limit depends on the collective stake value of the bridge nodes and the number of Byzantine or malicious nodes we tolerate.

The second set of actors – incentivized whistle-blowers determine the length of the sliding window. There is a natural turnaround time for whistle-blowers to detect any collusion, and report to the Audit-DAO which validates and resolves disputes. Let T denote the worst-case of this time duration. This T is a precise fit to be our sliding-window length as any violation/collusion will then be reported. Whistle-blowers then collectively act as balancers for colluding bridge nodes.

Note that we limit our focus of *rationality* to the bridge nodes and whistle-blowers only, and assume that the governing Audit-DAO network is trusted, meaning it contains the required number of honest actors. The monitoring Audit-DAO does not come into the picture in the normal operation of the bridge protocol, and comes into operation only in the exceptional cases of malicious activity. With this trust assumption, we take the verdict of the monitoring Audit-DAO to be determinate, as is most practical from our analysis.

Contributions. The design of our multi-sig bridge, HyperLoop, brings in the following contributions:

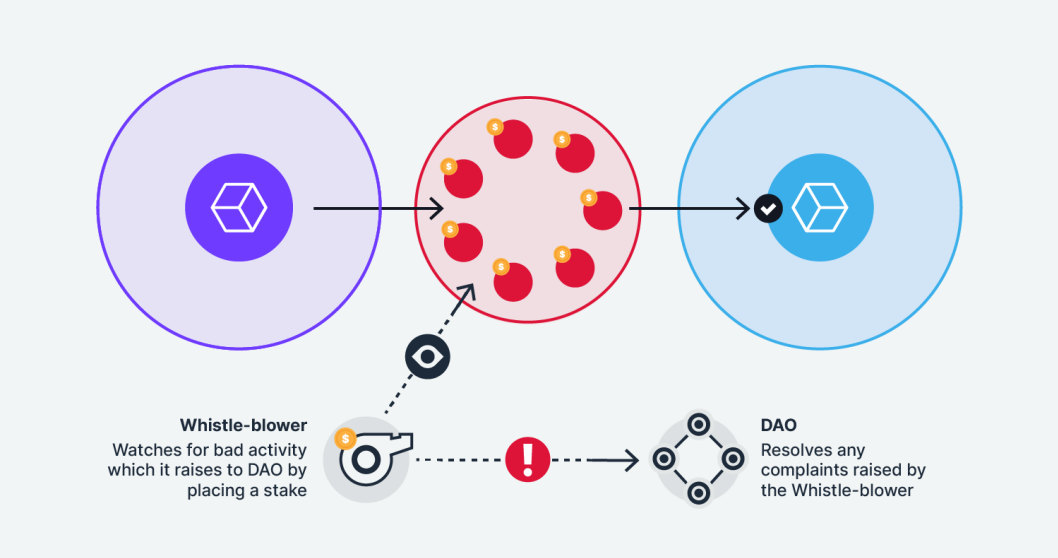

Figure 1 Architechture of the HyperLoop Bridge

- Using game theory, we derive an **incentive-penalty** mechanism and show that the non-collusion is the dominant strategy for the bridge nodes so that the crypto-economic security of the bridge protocol is assured. As far as we know, we are the first ones to analyze a bridge protocol in a much-needed rational setting.
- We realize a decentralized collusion detection mechanism by introducing an open participation of incentivized **whistle-blowers**, who can also be rational.
- We introduce a **sliding window** limit on high-value transfers so that the loss due to collusion is capped.
- We realize different **batching** strategies, through which we make our bridge design gas-efficient and scalable.
- We construct a bridge protocol which requires only a **simple majority** of bridge node signatures in a non-interactive way. This yields an economically efficient design with fewer bridge nodes and less cumulative stake value. In other words for the same cumulative stake value we can bridge higher values than the bridges requiring super-majority of signatures.

The rest of the paper is organized as follows. We contrast the related work in Section 2. We describe the system setup and adversary model in Section 3 and formalize the desired properties from a bridge in Section 4. We then present the HyperLoop protocol in Section 5 first in a simplified honest-malicious model and then in the rational model. We discuss the subtleties of this protocol in Section 6. We provide the game-theoretic analysis and the resultant stake value computation in Section 7, discuss allied topics of security in Section 8, and conclude the paper in Section 9.

2 Related Work

Before we introduce our approach, we briefly discuss a few well-known bridge protocols. Here, we discuss only decentralized bridge solutions for comparison and omit centralized solutions as they do not offer "decentralized trust" which is of our interest as well as the defining feature and value-proposition of blockchain systems.

Approaches to use assets like Bitcoin on other blockchains have of course already been developed in order to tap the economic potential of Bitcoin on different DeFi applications on other chains. The standard way was to deposit some Bitcoins in a custodian's account on Bitcoin's core chain, and the custodian would subsequently mint a matching amount of *wrapped Bitcoin (wBTC)* on the destination chain. These wBTC can then be used for different applications and, when needed, exchanged back for the original Bitcoin on the Bitcoin network, typically in a $1:1$ ratio. This involved trusting the custodian though the trust is often decentralized using a threshold-distributed version of the custodian. The decentralization may take the shape of a multi-sig or a threshold-signature generating protocol, where a minimum number of nodes must generate a signature on the source chain requests.

HyperLoop is a pairwise multi-sig bridge that is applicable when the destination chain can not directly validate the events on the source chain. In cases where the destination chain can indeed validate source-chain events, one does not need a multi-sig based protocol to achieve bridge functionality, a simpler *relaying* functionality suffices. Supra offers such a bridging solution – HyperNova [4], where the bridge nodes simply forward the information from the source to the destination chains and are validated on the destination chains. They also need to forward auxiliary information (like a set of public keys of the source chain consensus validators) to the destination chain to help verify the requests to settle.

Zero-knowledge proof-based Bridges fall into the category of *relay bridging*, as they do not require a confirmation from a set of bridge nodes. As is standard in the relay bridging mechanism, only one honest bridge node is required, and that too only for the liveness as the ZK proof serves as witness for any tamper resistance. However, these approaches are still in the initial stages involving high proof-generation times resulting in unacceptably high latency.

As HyperLoop is a pairwise bridge solution, it is appropriate to compare it against other pairwise bridges as opposed to interoperability-focused systems like Cosmos [2] and Polkadot [3] directly. So, we compare only the pairwise subchains of these systems or communication between a subchain and an external chain. A detailed comparison of these interoperability-focused systems as well as the protocols that attempt to connet to multiple chains at once like Axelar [8] and Zetachain [7] is scoped and presented in our IntraLayer approach [5]. Very briefly, Supra's IntraLayer approach positions Supra as the decentralized nexus by which external chains are connected via the HyperNova and HyperLoop bridge solutions.

The class of bridge protocols that we refer to and extend are typically called "middle chains" or "multi-sig" chains, where confirmation from a group of staked bridge nodes is sought for an event communicated between chains. As we progressively unfold the design and features of HyperLoop, a multi-sig chain, its economic viability, practical efficiency, and scalability are established.

We observe that a game-theoretic analysis of bridges in a more realistic rational setting is missing, and so apply and present from the same angle throughout this work. Subsequently, we remark on other (than game-theoretic analysis) differences of bridge protocols from HyperLoop.

Cosmos

Cosmos IBC [2] applies a hub-and-spoke architecture, with Cosmos Hub in the center and zones as the spokes. All the zones are independent, run Tendermint [6] consensus protocol and communicate with other zones via Cosmos Hub using Inter Blockchain Protocol (IBC). A zone could communicate to another zone directly (without the hub) provided the nodes of

the zone also run the light clients of the destination zone. The nodes of the zones also run the Tendermint light clients of the Cosmos Hub so that all the communication is bridged via the Cosmos Hub. The Hub does not host user transactions. There can be more than one Hub chain as well. The communication between zones works on the principles of *relaying* rather than a multi-sig approach of HyperLoop. So we compare it with our relay bridging solution - HyperNova [4].

Since Tendermint light clients cannot be run on the validators of external (to Cosmos) chains like Ethereum, it requires another network, a specialized Cosmos zone called a *peg zone*, to bridge across chains. Then, this peg zone serves as a multi-sig bridge between an external chain and Cosmos zones. For example, the Gravity Network/Bridge serves as a multi-sig bridge between Cosmos and Ethereum. Since this peg zone runs an instance of Tendermint Consensus protocol in a partially synchronous networking model, they need a super-majority rather than HyperLoop's simple majority.

Another distinguishing factor is latency. The communication from a Cosmos zone to an external chain has to go through the relay process of the Cosmos Hub and then the consensus protocol of the peg zone before reaching the destination chain. In contrast, there is no interactive distributed protocol in the HyperLoop bridge design, which facilitates a direct connection between a Cosmos-style 'peg zone' and a destination chain.

Polkadot

Polkdot [3] also makes use of a hub-and-spoke model with a Relay chain at the Hub. Like Cosmos, the Relay chain does not host any user transactions. Only the parachains (spokes) contain the user transactions. Unlike Cosmos, the Relay chain maintains the state and the corresponding Merkle root of all the parachains. It also provides finality to the blocks of the parachains.

The communication between parachains happens through the principles of relay bridging similar to Supra's HyperNova [4] solution. Similarly, the communication between Polkadot and external chains also follows the relay bridging method, as discussed in our HyperNova approach [4].

Wormhole

Wormhole [11] is a multi-sig bridge with exactly 19 (as of this writing) permissionned nodes in the network. The same network of nodes is used for connecting multiple chains. It requires 13 out of 19 nodes to sign off on cross-chain requests. The Wormhole philosophy trusts large validator companies that are heavily invested in the success of DeFi as a whole, rather than use tokenomics and open participation. HyperLoop differs fundamentally wherein it offers open participation and yet assures that the nodes do not collude in the rational setting.

LayerZero

The central thesis of LayerZero[10] is to require two *separate* entities, Oracles and Relayers, to conduct settlements from source chains to destination chains. They claim that as long as the interests of Oracles and Relayers conflict, collusion attacks on any transfers using this pair of entities are not possible. LayerZero claims the selection of these entities to be configurable, though Chainlink Oracle [1] appears to be a fixed choice for the Oracle role for all practical purposes. Another artifact of this separation is that even if one pair of Oracle-Relayer (x, y) is compromised, the apps configured to set either the Oracle or the Relayer to other than x or y are saved from any corresponding attacks, thus restricting the attack surface.

As mentioned ealier, we analyze the bridge protocols from a natural, rational model that has not been done before. In a rational model, the Relayer and the Oracle committees can collude and convince the destination chain of any transaction, minting currency. This leads to an undetected and perpetual attack on the system.

Layer 0's two-entity model yields significant power to each of the groups in terms of messing up the ordering, yielding to MEV attacks, and mounting delaying (leading to front running), back running, sandwiching and censorship attacks. In the LayerZero model, a DApp is motivated to run relayer nodes on their own to retain control and game the system covertly via these attacks. Thus, unsuspecting users can fall prey to such attacks if not adequately accounted for in the original security design.

Chainlink's Cross-Chain Inter-operability Protocol (CCIP)

The CCIP protocol consists of three actors: a *committing* Distributed Oracle Network (DON), an *executing* DON, and a *Risk Management Network (RMN)*. The committing DON monitors the events of an *OnRamp* smart contract on the source chain. It then bundles the bridge requests generating those events and computes a Merkle Root of this bundle. This Merkle root is then posted on to the destination chain along with a Quorum Certificate (QC) of the committing DON showing its Byzantine Fault Tolerant (BFT) consensus. The nodes in RMN also follow the bundle of bridge requests on the source chain, compute the Merkle root independently, and validate the committing DON posted Merkle root on the destination chain. Upon success, the RMN node 'blesses' this Merkle root or else it 'curses' in the case of disputes. The nodes in the executing DON post these bridge requests along with its Merkle proof against the blessed Merkle root on the destination chain, and then they get processed on the destination chain.

Remark on security. Note that the Merkle proof validation on the destination chain stops the processing of an illegitmate bridge request posted by a malicious executing DON node. Even if more than the threshold nodes in a commiting DON collude, along with a node in an executing DON, the cursing of RMN stops the processing of an illegitimate bridge request on the destination chain. Collusion attacks are only possible when the threshold nodes in the RMN collude with the threshold nodes of the committing DON along with the executing DON nodes. The responsibilities of the committing DON and the RMN are almost the same, and the value of this separation is unclear.

Remark on latency. Mainly the protocol has 3 steps: Merkle root posting by the committing DON after a consensus inside the commiting DON, Merkle root endorsement by the RMN, relay of the bridge request and its Merkle proof by an executing DON node. In comparison, HyperLoop has just one step without any interaction in the distributed protocol during normal (non-collusion/malicious) operations and, hence is faster.

Atomic Swap approaches

Atomic swaps are another method to bridge assets. After all, an asset transfer is essentially a swap of one currency with the target currency by the client. The client swaps currency, with the bridge acting as the second entity in the swap. Existing techniques include using HTLC - hashed time-lock contracts, adaptor signatures, etc. These techniques involve setting up shared keys between the parties to achieve atomicity. Each client needs to establish shared keys with the bridge, and simultaneous bridge requests from multiple clients can not be batched together, forcing the bridge to deal with each request independently. This results in

a significant overhead for setting up shared keys with each client and in overall latency in the system.

3 System Setup

In this section, we set up the preliminaries and definitions required for the rest of the paper.

Hyperloop is a pairwise bridge protocol that connects two blockchains enabling information and asset transfers between them. There are three actors in our protocol – a network of bridge nodes, a set of whistle-blower nodes, and a Decentralized Autonomous Organization (DAO) network. The bridge nodes run the clients of both the source and destination chains to follow the events on them. The whistle-blower nodes also run both the clients and constantly monitor and verify the actions of the bridge nodes. They raise complaints upon discrepancies to the monitor DAO, which is assumed to be trusted. The details of the realization of the DAO network are outside the scope of this paper.

The nodes (bridge and whistle-blower) can be classified into the following categories:

Honest nodes always follow the protocol steps and do not deviate from expected actions or responses of the defined protocol.

Rational nodes always take rational actions for economic incentives. A rational node may deviate from the expected actions or responses prescribed by the protocol. These nodes have a (rational) strategy to maximize the utility obtained by such a deviation. The strategy space may be known to other nodes in the protocol, however, the exact strategy or steps taken by the parties are not known to the other parties.

Malicious nodes can arbitrarily deviate from the protocol. The actions taken by the node need not maximize the node's utility nor result in an economic incentive. Nodes compromised by the adversary can act maliciously and are called Byzantine.

Throughout the paper, we use the word *node* to mean a bridge node unless otherwise explicitly stated.

Network Model

The bridge network is completely connected and asynchronous [12]. An asynchronous network entails that the messages sent by the honest parties are delivered to other honest parties eventually, and none of the messages are dropped. There is no bound on the message latency. Bridge nodes may have different views of the chain at any given instant due to the asynchrony of the network. All the bridge nodes are connected by authenticated point-to-point links and have access to a reliable broadcast channel.

Threat Model

Honest-Malicious Model. In this model, the protocol tolerates some nodes (up to a limit, say f) being malicious and the rest of the nodes are honest. Most of the distributed protocols till now – State Machine Replications (Consensus protocols), Bridge, and Broadcast protocols etc, have been studied under this model of Honest-Malicious.

Pure Rational Model. All the nodes of the protocol are rational in this model.

Rational-Malicious Model. The protocol tolerates some number (up to a limit, say f) of malicious nodes, and the rest are rational.

All the malicious nodes are assumed to be controlled by a single adversary. The adversary can corrupt up to f nodes when the network is initialized, and controls all communication channels such that messages can be arbitrarily delayed.

Notion of Time

Time is needed for handling *revertable bridge requests* (Section 4). Blocks of many blockchains typically carry wall-clock time-stamps. These approximated wall-clock block time-stamps can be used to specify the time factor for revertable requests. When the block time-stamps are not available, we use the number of blocks on the destination chain as the factor of time for reverts.

Staking

The bridge nodes are required to stake a predetermined amount of tokens. The external whistle-blower nodes are also required to include a stake amount at the time of raising a complaint. Upon a successful complaint, the stakes of the colluding bridge nodes are slashed and a portion of this, denoted by r, is paid as a reward to the corresponding whistle-blower node. The whistle-blower nodes are also paid a periodic reward for the verification process. The stake amount of complaints is withheld when the complaint is determined to be false. Section 7 expands up on the staking requirements.

4 Correctness Properties

We define the properties we expect cross-chain bridges to satisfy in this section.

We use the term *bridge requests* for the transactions on the source chain asking for a message transfer or asset transfer or requesting some service from the bridge network. The corresponding transaction posted on the destination chain by the bridge nodes is termed the *bridge response*. The bridge requests could come with an optional *revert* period τ_{rev}, indicating that the submitters of the requests expect their transactions on the source chain to be reverted in case the corresponding bridge responses are not posted on the destination chain within τ_{rev}. τ_{rev} is specified either as wall-clock time approximated by block time stamps or by the number of blocks on the destination chain. Such requests are termed *revertable bridge requests*, and the corresponding reverting transactions on the source chain are termed *reverted bridge requests*.

We use the following notation and definitions in describing the properties expected from the bridge:

$\langle \texttt{req}_1, \texttt{req}_2, \ldots, \texttt{req}_k \rangle$ indicates the bridge request transactions on the source chain respecting the total order from $\texttt{req}_1$ to $\texttt{req}_k$.

$\texttt{valid}(\texttt{req}, \texttt{res})$ is a relation that holds on a tuple – a bridge request $\texttt{req}$ and a bridge response $\texttt{res}$, only if they are valid and successful.

$\texttt{validRevert}(\texttt{req}, \texttt{rev})$ is a relation that holds on a tuple – a revertable bridge request $\texttt{req}$ and a reverted bridge request $\texttt{rev}$.

$\texttt{limit}(\texttt{req}_{i+1}, \texttt{req}_{i+2}, \cdots, \texttt{req}_{i+k})$ is a relation that holds on a sequence of bridge requests only if they satisfy a limit condition. Typically, this condition is used to limit the total value of the amount transferred in a time duration.

We now present the properties that we expect from the bridges and group them as the classical safety and liveness properties.

Safety

Validity.

- Every pair of non-revertable bridge request `req` and its bridge response `res` must satisfy `valid(req, res)`.
- Every pair of revertable bridge request `req` and its bridge response `res` satisfies `valid(req, res)` if and only if `res` happened within τ_{rev}.
- Every pair of revertable bridge request `req` and its reverted bridge request `rev` satisfies `validRevert(req, rev)` if and only if there is no bridge response `res` within τ_{rev} such that `valid(req,res)` holds.

Ordering.

- Let the ordered sequence of bridge requests be $\langle \texttt{req}_1, \texttt{req}_2, \cdots \rangle$, and let the ordered sequence of bridge responses be $\texttt{resp} = \langle \texttt{res}_1, \texttt{res}_2, \cdots \rangle$. Then, for every $1 \leq i$, if $\texttt{res}_i \in \texttt{resp}$ then $\texttt{valid}(\texttt{req}_i, \texttt{res}_i)$ holds, otherwise $\texttt{req}_i$ is a revertable bridge request.
- Let the ordered sequence of reverted bridge requests be $\langle \texttt{res}_1, \texttt{res}_2, \cdots \rangle$. Then, there must be an ordered subsequence of revertable bridge requests $\langle \texttt{req}_1, \texttt{req}_2, \ldots, \rangle$ such that $\texttt{validRevert}(\texttt{req}_i, \texttt{rev}_i)$ with $1 \leq i$.

Sliding Window Limiting. Let $\langle \texttt{req}_1, \texttt{req}_2, \ldots, \texttt{req}_k \rangle$ be the sequence of bridge requests in any duration of predetermined length, say T. Then, $\texttt{limit}(\texttt{req}_1, \texttt{req}_2, \ldots, \texttt{req}_k)$ holds.

Liveness

- For every non-revertable bridge request `req` there must exist a bridge response `res`.
- For every revertable bridge request `req`, there exists exactly one of the following (XOR relation):
  - Bridge response `res` within τ_{rev}, (XOR)
  - reverted bridge request `rev`.

5 HyperLoop Protocol

We describe HyperLoop, a multi-sig bridge protocol, give its specifications, and show how the mechanism design aspects preserve its safety and liveness properties.

We present the bridge protocol in the honest-malicious model first. Let n denote the number of bridge nodes that run the clients of both the chains and offer the bridge service. Let t be the number of Byzantine/malicious nodes we tolerate in the bridge network. Smart contracts are deployed on both the chains encoding the bridge functionality. The bridge nodes listen to the events from these smart contracts that result from client requests on the source chain and perform appropriate action/s on the destination chain. The basic workflow of the HyperLoop protocol is shown in Figure 2.

Note that there is no distributed protocol employed among the nodes of the bridge, they are just relaying the information after signing off on its validity. More details on this aspect are presented in Section 6.

Batching and non-batching channels. Bridging each and every bridge request separately and individually could be inefficient, especially when these contain small value transfers or

HyperLoop protocol

1. A user/client on the source chain submits a bridging request transaction by calling an appropriate function of the smart contract on the source chain.
2. The bridge request transaction is executed, emitting specific events on the source chain.
3. Bridge nodes detect these events. Then, they sign and submit a *response* transaction to the destination with a payload containing all the relevant information of the bridge request. Note that the bridge nodes need to hold accounts on the destination chain to submit such transactions.
4. The smart contract on the destination chain records these messages. Upon receiving response transactions from $t+1$ bridge nodes, it performs the appropriate bridge response action.

Figure 2 HyperLoop protocol steps

custom messages.. An efficient, scalable approach is to batch multiple requests together. We use 3 configurable parameters for batching:

`bTimeLimit` We batch bridge requests for the duration set by `bTimeLimit`. This avoids inordinately long waiting times.

`bValueLimit` We batch the bridge requests up to or immediately greater than the value of `bValueLimit`. If the sum of the values of the requests is greater than or equal to the `bValueLimit` then those requests are batched and forwarded. If a request is of value greater than `bValueLimit` then it is still admitted. Such a request of value greater than `bValueLimit` is immediately processed (a batch on its own) without waiting for other requests. This process allows for the prioritization of larger value requests to reduce the delay and latency they experience. Hence `bValueLimit` is a soft limit.

`bNumLimit` We cap the number of bridge requests to be batched by `bNumLimit`.

Batching helps amortize the costs of multi-sig verification on the destination chain among multiple bridge requests and hence lowers the bridging fees. However, batching increases the latency of serving bridge requests by a factor dependent on the batching parameters.

HyperLoop also provides a non-batching and faster way of communication. This is helpful for users who prefer faster processing of bridge requests and are ready to pay a higher fee. We introduce the abstraction of *channels* to separate the requests going in the batching or the non-batching route. The property (see Section 4) of ordering is provided only within these channels.

Reverts. The distributed protocol of HyperLoop is realized in an asynchronous setting, meaning there is no theoretical time bound on the submission of the corresponding response transactions on the destination chain. As we see in the following sections, we do incentivize all the actors for their respective works, and hence we have a highly performant design. However, there are scenarios in which the user expects the response transaction on the destination chain to happen within a specific time, and if not, they prefer reverting the transaction rather than waiting for it to settle. To cater to such scenarios, HyperLoop offers optional *revertable* bridge requests, wherein bridge requests come with a time parameter τ_{rev}. Here we require the bridge smart contracts on the destination chain ensure that the bridge requests are not honored past their τ_{rev}. In other words, if the response transaction posted by the bridge nodes lands in a block of the destination chain with a timestamp value more than τ_{rev}, then they fail.

For the revertable bridge requests, the additional steps of the protocol are shown in Figure 3.

HyperLoop protocol - reverts

5. For every revertable bridge request `req`, upon the destination chain adding a block with a timestamp greater than τ_{rev} of `req` without hosting its corresponding response transaction, the bridge nodes submit a *reverting* transaction on to source chain to revert `req`.
6. The smart contract on the source chain records the payloads of the reverting transactions. Upon receiving such reverting transactions by $t + 1$ bridge nodes, it performs the appropriate reverting operations on the source chain.

Figure 3 HyperLoop protocol steps for reverts

▶ **Theorem 1.** *HyperLoop protocol of Figures 2 and 3 with the batching and non-batching channels satisfies the Validity, Ordering and Liveness properties (of Section 4) in the Honest-Malicious model even in asynchronous networking conditions, so long as there is a simple-majority of honest bridge nodes.*

Rational Setting

We now consider the rational setting where the bridge nodes are rational and work to maximize their gains. In such a scenario, the rational bridge nodes can collude and sign off a non-existent bridge request, and mint money on the destination chain illegitimately. This is precisely why an incentive-penalty mechanism, which incentivizes good behavior and penalizes bad behavior, is needed. To enable such a mechanism we require the nodes to *stake* an amount, say S, so that slashing and penalization can be implemented. Depending on the nature of the bad activity the penalization could include suspension from participation for a time period, banning completely, or legal recourse in case of a permissioned setup.

Limiting through *sliding time-window*. Staking does not fully solve the problem as rational bridge nodes may collude to mint illegitimate assets on the destination chain greater than that of the staked amount. That is to say, the staked funds would cease to function as a deterrent since the reward might far outweigh the slashing penalties that might be imposed. Hence a limit on the bridge requests is necessary. HyperLoop imposes a value limit of L for a time window of length T. This time window is implemented as a sliding window as opposed to the non-sliding approach. If a value v has been approved for a source chain in a time t, HyperLoop disallows authorizing a transfer of value greater than $L - v$ in the next $T - t$ time. The limit or restriction is realized as a sliding window where the limit is imposed over the continuous time domain. This prevents the bridge nodes from transferring the value of $2L$ in the tail edge of one window and the starting edge of the next window. The relation between L and the stake S is presented in Section 7.

HyperLoop enforces this sliding window limit simultaneously at 3 places: bridge smart contracts on the source chain, the bridge nodes, and the bridge smart contracts on the destination chain. This layered security avoids the exploitation of loopholes (software vulnerabilities) at any one place. This is also better in terms of UX (user experience) because if a bridge request is not admitted owing to the crowded sliding window pipeline, then the client's request fails on the source chain itself.

The client then needs to resubmit this request, perhaps with a higher transaction fee on the source chain. The other option is to enforce the sliding window only at the destination chain. Then, the smart contracts on the destination chain need to buffer the requests in an unbounded way. Finally, this results in a worse UX as the bridge request submitter waits indefinitely for the response though his bridge request transaction has been finalized on the source chain.

Whistle-blowers & Audit-DAO. The rational bridge nodes are not incentivized to raise a valid complaint against other nodes. They can keep generating fraudulent transactions that mint tokens on the destination chain. These tokens may be equally shared amongst the colluding nodes. To solve this problem, we incentivize external nodes, *whistle-blowers*, to verify and provide proof of fraud transactions on either chain to break such collusion. The whistle-blower nodes constantly monitor the source and destination chains and the signatures generated by bridge nodes and raise complaints in the case of discrepancies as shown in Figure 4.

HyperLoop protocol - Whistle-blowers guard Validity

If the bridge nodes sign off a value greater than L, or sign transactions without a valid source chain request, or in any way try to break the Validity property of the protocol, the whistle-blower nodes raise a complaint to the Audit-DAO network by piggy-backing a stake β.

Figure 4 HyperLoop protocol - Whistle-blowers guard Validity

At times, bridge nodes may opt to collude together, breaking the liveness property (see Section 4) by not submitting the required transactions on to the destination chain so that no action corresponding to the bridge request is taken. This is especially true for non-revertable requests and therefore the revertable requests have a built-in τ_{rev} to address this. However, the problem still manifests in the opposite direction when the reverted requests need to be submitted by the bridge nodes, and they either do not submit them or delay them indefinitely. To solve these issues, we introduce a notion of a *timeout* and the rule for whistle-blowers as shown in Figure 5.

HyperLoop protocol - Whistle-blowers guard Liveness

If $t+1$ number of bridge nodes do not submit the required transactions corresponding to a non-revertable request within *timeout* then the whistle-blower nodes raise a complaint to the Audit-DAO network by piggy-backing a stake β. Similarly, if $t+1$ number of bridge nodes do not submit the required reverting transactions corresponding to a revertable-request req within *timeout* when the corresponding response does not happen within τ_{rev} of req, the whistle blowers raise a complaint.

Figure 5 HyperLoop protocol - Whistle-blowers guard Liveness

The first node that generates the valid complaint is rewarded. This reward could go up to the sum total stakes of all the colluding bridge nodes. The piggy-backed stake β on the complaint prevents any of them from generating invalid complaints and false alarms since the stake is withheld upon invalid complaint.

Nuanced network model. Note that the introduction of *timeout* limits us to a synchronous networking setting. In practice, because the timeouts are in the order of tens of minutes or even hours, we find that this synchronous setting in the *coarse* granularity of time and the asynchronous setting in the *fine* granularity of the time to be perfectly viable. We term this as a *mixed networking condition*.

The Audit-DAO network consists of independent nodes with access to the source and destination chain states, and the transactions signed by the bridge nodes. Upon a valid complaint from a whistle-blower, the Audit-DAO network may request that each bridge node respond to it. If no valid response is obtained within the required window of time, the stake of the bridge nodes is confiscated, and in the case of permissioned setting, the bridge nodes are publicly banned from the system. The first whistle-blower node which raises a valid complaint, is rewarded.

We now informally state the correctness of the HyperLoop protocol, shown in Figure 2, in the rational setting. We present a more game-theoretic treatment in Section 7.

▶ **Theorem 2.** *HyperLoop protocol of Figures 2 to 5 that includes the batching and non-batching channels, and the enforcement of the sliding-time window satisfies all the correctness properties of Section 4 in a mixed networking condition in the rational and rational-malicious settings.*

6 Discussion on HyperLoop Protocol

Honest simple majority and honest super majority. We observe that in almost all of the existing bridge networks, a super majority threshold is employed, with the assumption that there exists at least a super majority of honest nodes in the bridge network. We understand that it is an inspiration from *Byzantine Agreement* [14] and *Byzantine Fault Tolerant State Machine Replication (BFT SMR)* [13] protocols.

We remark that this need not be the case, mainly because the bridge requests are already ordered owing to the execution of BFT SMR protocol on the source chain. These bridge requests also have a unique place on the source blockchain – typically with a block number and the transaction index inside the block. When we have this unique placement of the requests on the source chain, a Byzantine Agreement or a BFT SMR protocol becomes redundant. We observe that there is no possibility of equivocation in the bridge network. All that is required is a confirmation or an endorsement from a simple majority of nodes in the bridge. The bridge nodes simply relay requests from the source chain to the destination chain while preserving the unique indexing information. The bridge smart contracts on the destination chain then need to receive such relay messages only from a majority of bridge nodes before taking the corresponding action on the destination chain. The simple majority requirement then guarantees the inclusion of one honest node's endorsement (in the honest-malicious model), which is sufficient to take the required response on the destination chain.

By doing away with the roles of proposers and aggregators, the bridge protocol yields to a simpler game-theoretic analysis, as presented in Section 7.

Threshold signature. An accountability incentive structure needs to be in place for bridge nodes to reward honesty and penalize malicious activity. As the standard designs of threshold signature cryptography do not efficiently provide accountability, we opted not to use it. Additionally, we do not have any aggregators in the bridge protocol itself. Instead, the

signatures are individually verified on the destination chain until a simple majority threshold is reached.

An argument can be made that this approach of validating individual signatures is not scalable. Consider the alternatives which typically include a role of an aggregator in the protocol. Here the aggregator receives signed endorsements of bridge requests from the bridge nodes, aggregates them into a multi-sig along with a bit-map or a threshold signature, and submits to the destination chain as one single transaction. As mentioned above, owing to accountability, we need to go with a multi-sig approach. The validation on the destination chain smart contract would typically involve aggregating the public keys (using bit-map) and then validating the aggregate multi-sig with this aggregated public key.

This approach is expensive regarding gas costs as the public key aggregation and aggregated signature validation will be performed on the destination chain's virtual machine. In HyperLoop protocol the signature validation occurs external to the blockchain's virtual machine along with the transaction validation and is hence more cost-effective.

This alternate approach also yields higher latency as we now have an interactive protocol with aggregators to account for.

Adding or removing bridge nodes. Typically a notion of *epochs* is introduced and the bridge nodes are added or removed only at these epoch boundaries. Because of our protocol is non-interactive and also does not necessarily need a distributed key generation protocol, the bridge nodes may be added or removed seamlessly without halting the bridge operations at all. All that needs to be done is to update the destination chain smart contracts with the public keys of the added or removed nodes.

With the change in the bridge node set, the sliding window value limit also changes. Because the sliding window value limit is enforced on the destination chain smart contract as well, this change may also be seamless.

Though it is not a requirement from the protocol, updating the bridge nodes across epoch boundaries can be a convenience. In our intralayer [5] paper we plan to utilize the epochs of our Layer 1 blockchain to update the membership of the bridge nodes.

Use-cases and architecture. Hyperloop is essentially a cross-chain message-passing protocol. Many kinds of cross-chain applications may be built on top of this message-passing layer. These applications include:

- cross-chain smart contract function calls,
- lock-and-mint wrapped asset bridging,
- stable-coin bridging with liquidity pools,
- NFT bridging,
- cross-chain DeFi apps like lending, derivatives, etc

7 Game-Theoretic Reward and Stake Analysis

In this section, we design the rewards and penalties for the different parties to prevent collusion. We present a basic analysis for setting the rewards for each party after discussing different collusion scenarios. We first consider a purely rational model and then extend it to a rational-malicious model.

The (rational) bridge nodes may collude at any point amongst themselves for an economic incentive, i.e. if the payoff from a deviation from the protocol is higher than the penalties for doing so. They may collude to even endorse a non-existent bridge request on the source chain and mint money higher than the penalties for doing so. This breaks the Validity property stated in Section 4.

However, with the sliding window property from Section 4, the bridge nodes can not transfer a value more than L in a time window of size T. This is enforced by HyperLoop's sliding window limit as discussed in Section 5. Similarly, we also assume that the Ordering property is always maintained.

We also require that the turn-around time of the complaint resolution be bounded by the sliding-window length of time T. Otherwise, multiple fraudulent requests can be encashed on the destination chain by the time the penalties are applied to the colluding bridge nodes.

We consider three scenarios for the bridge nodes:

Permissionless Any node may join the bridge network and offer services by staking an amount S. The staking amount is confiscated after being caught doing unauthorized/illegitimate transfers.

Permissioned Nodes submit publicly verifiable identities, stake a value S, and enter into legal contracts before offering services. Any deviation from the protocol by the permissioned nodes is publicly visible when connecting public blockchains. Such malicious activities are penalized by confiscating the stake, the responsible node operators are banned, and legal action is taken if needed. Each node incurs a negative pay-off of value denoted by γ due to banning. This is the total pay-off they may obtain from future service offerings and the repercussions of legal recourse.

Dual committee approach Bridge network partitioned into two committees – one *permissionless* and the other *permissionned*.

7.1 Design of the stake and reward value

We now compute the bounds on the rewards and stakes of the nodes to join the bridge network. Due to the enforcement of the sliding window value limit of L, we assume that no collusion can extract more than L. Here, we assume that the whistle-blower nodes are guaranteed to detect fraud, raise a complaint to the DAO, and resolve it within the sliding window time limit of T.

Permissionless bridge network. Any $t+1$ or more nodes can collude to endorse an illegitimate or non-existing bridge request. Per HyperLoop's design such colluding nodes will be caught, and they will lose their stake value of S. So, collusion is not profitable, and the total pay-off is negative if the total gain is less than the total loss to the nodes, i.e., if:

$$L < (t+1)S. \tag{1}$$

In other words, the stake of each node should be at least $\frac{L}{t+1}$ to make the pay-off from collusion negative.

A_1 \ A_2	*collude*	*honest*
collude	$\left(\frac{L}{2}-S, \frac{L}{2}-S\right)$	(f, f)
honest	(f, f)	(f, f)

Table 1 Pay-off matrix of parties in the a two-party bridge in the permissionless setting

The protocol introduces a game between the bridge nodes with two strategies - collude or act honestly. Throughout this section, we illustrate the game's strategies and their pay-offs using a simple two-node example. A general case for n nodes should follow a similar analysis.

Consider the case of just two bridge nodes, wherein both nodes must endorse a bridge request for the corresponding action to be taken on the destination chain. Table 1 depicts the pay-offs for the different strategies. If one of the nodes is honest, collusion does not occur, and they obtain a positive pay-off, say f, typically the reward for their services. Under collusion, they can transfer a value $v \leq L$ together, and each can obtain a maximum pay-off of $\frac{L}{2}$. However, both the nodes lose their stake of S, making the pay-off $\frac{L}{2} - S$ for collusion. Since we set $S > \frac{L}{2}$, the pay-off would be negative for the colluding nodes and from Table 1, to make honest behaviour an equilibrium and a dominant strategy, we need $\frac{L}{2} - S < \mathsf{f}$. For a general case with n nodes, collusion occurs only when at least $t+1$ of the nodes decide to collude. Following a similar pay-off matrix, for honesty to be an equilibrium condition, we need

$$\frac{L}{t+1} - S < \mathsf{f}. \tag{2}$$

Permissioned bridge network. Any $t+1$ nodes can authorize the transactions. Upon unauthorized transfer and identification, each colluding node will lose its stake S and also incur a negative pay-off of γ (for losing future service offerings and legal recourse). Similar to the previous case, pay-off from collusion is negative if

$$\frac{L}{t+1} < S + \gamma \tag{3}$$

and honesty is an equilibrium and collusion is not profitable if

$$\frac{L}{t+1} - S - \gamma < \mathsf{f}. \tag{4}$$

Usually $\gamma \gg S$. In such cases, S can even be set to zero.

A_1 \ A_2	*collude*	*honest*
collude	$\left(\frac{L}{2} - S - \gamma, \frac{L}{2} - S - \gamma\right)$	(f, f)
honest	(f, f)	(f, f)

Table 2 Pay-off matrix in permissionned setting

Consider again the simple case of just two nodes. Apart from confiscating the stake, the nodes are banned upon collusion, incurring a negative pay-off of γ. The pay-off matrix is depicted in Table 2. The condition where honesty is more profitable that collusion is $\mathsf{f} > \frac{L}{2} - S - \gamma$. where the pay-off upon honest behavior is f. The dominant strategy for the bridge nodes then is *not to* collude and behave honestly. Thus, non-collusion is a Nash equilibrium in the game as long as the inequality of (3) is satisfied.

Dual committees - permissioned & permissionless bridge network. The HyperLoop bridge may be realized using two committees, one permissioned and the other permissionless. Assume the committees have threshold access structures (n_1, t_1+1), (n_2, t_2+1). A transaction is authorized if t_1+1 number of nodes in the permissionless committee and t_2+1 nodes in the permissioned committee sign it. Let S_1, S_2 be the stakes of permissionless and permissioned nodes, respectively. The pay-off from collusion is negative if

$$L < ((t_1+1) \cdot S_1) + ((t_2+1) \cdot (S_2 + \gamma)). \tag{5}$$

Following the similar analysis of pay-off matrix as in Table 2, honesty is an equilibrium if $\frac{L}{t_1+t_2+2} - S_b - (b-1)\gamma < \mathsf{f}$, where the node belongs to the committee $b \in \{1,2\}$ with committee 2 being the permission committee.

Any node that wishes to raise a complaint against the bridge nodes first must provide a stake of value β before making the complaint. If the complaint is valid, up to the sum total of the stakes of all the colluding bridge nodes is given as a reward along with the stake β. However, if the whistle-blower node is rational, they may not raise a complaint; they may instead approach the colluding bridge nodes to share the transferred value. We resolve this in Section 7.2

Securing Liveness.

Note that the bridge nodes do not gain anything by breaking Liveness by delaying indefinitely the submission of the corresponding transaction. Regarding asset transfers, the funds stay locked in the bridge smart contract on the source chain. However, the non-access of the funds to its owner is the main problem. To solve these issues, *timeout* was introduced, and a value of θ is applied as a penalty on all the bridge nodes that haven't submitted a transaction before the timeout when t+1 transactions are not received on the destination chain. The DAO network enforces enforces punitive measures via the complaints raised by the whistle-blower nodes. The pay-off matrix for a simple two bridge node setting is shown in Table 3, which makes it clear that not delaying is the dominant strategy and, hence the Nash equilibrium.

A_1 \ A_2	*delay*	*post*
delay	$(-\theta, -\theta)$	$(-\theta, \mathsf{f})$
post	$(\mathsf{f}, -\theta)$	(f, f)

Table 3 Pay-off matrix for the reverts

To summarise, we have the following variables inlcuding rewards and penalties in the HyperLoop protocol.

Stake value	S
Rewards for following the protocol	f
Penalty for breaking Liveness	$-\theta$
Penalty for breaking Validity	$-S-\gamma$
Whistle-blower stake while raising a complaint	β
Whistle-blower reward for a successful complaint	r
Complaint resolution turnaround time	T
Transfer limit in a sliding time-window of duration T	L

Table 4 Variables in the HyperLoop's incentive-penalty system

We now state the security of HyperLoop when all the bridge nodes are rational and there is at least one honest whistle-blower. For the sake of conceptual clarity, we consider only the permissionless setting. However, the following theorems can be extended in a straightforward manner to the aforementioned permissioned and the dual-committee settings.

▶ **Theorem 3.** *Let there be at least one honest whistle-blower node, and let all the bridge nodes be rational. The HyperLoop protocol of Figures 2 to 5 that includes the batching and*

non-batching channels, and the enforcement of the limiting sliding-time-window, yields a Nash equilibrium wherein all the correctness properties of Section 4 are satisfied in a mixed networking condition as long as the inequality (1) is satisfied.

Rational-Malicious model. An interesting model is the rational-malicious model wherein we assume an adversary which can control up to t nodes in the bridge network. The adversary has a bribing budget of ω to induce the rational parties to collude. When the adversary controls t nodes, he needs to induce only one more node for collusion to occur by offering a bribe of ω. Along with the bribe, the adversary may also forego all the transferred value, up to L, to this node. Upon complaint, the adversary will lose the stakes of their t nodes (tS), ω, and the value transferred maliciously (limited by L); so a total of $tS + \omega + L$. The colluding rational node will lose its stake S, but gains the bribe ω and the maliciously transferred/minted value L.

So, the inequalities corresponding to (1) and (3) are modified respectively to:

$$L + \omega + \mathsf{f} < S \tag{6}$$

$$L + \omega + \mathsf{f} < S + \gamma \tag{7}$$

$$L + \omega + \mathsf{f} < S_b + (b-1)\gamma \tag{8}$$

Here incase of a two-committee system, the node belongs to the committee $b \in \{1, 2\}$ where committee 2 is the permissioned one.

The inequalities of (6) – (8) ensure that collusion is not the Nash equilibrium. Suppose the stakes are set to $S > \frac{L}{t+1}$ (but not satisfying (6)), honesty is no longer the Nash equilibrium. However, even in case of a collusion, the users or the system will not lose any value. Therefore, the total loss to the system is L and the total stake withheld is $(t+1)S$. The system does not lose any value if $S > \frac{L}{t+1}$.

▶ **Theorem 4.** *Let there be at least one honest whistle-blower node. Suppose the adversary has a bribe budget of ω and controls t bridge nodes. Let the rest of the bridge nodes be rational. The HyperLoop protocol of Figures 2 to 5 that includes the batching and non-batching channels, and the enforcement of the limiting sliding-time-window, yields a Nash equilibrium wherein all the correctness properties of Section 4 are satisfied in a mixed networking condition as long as the inequality (6) is satisfied.*

7.2 Collusion scenarios with whistle-blower nodes

So far, we expected to have at least one honest whistle-blower node. Given that HyperLoop's whistle-blower node can be anybody in the world, we feel it is a practical assumption to make. Note that Supra itself can run one whistle-blower node to safeguard its bridges. However, for the sake of completeness, we now consider the scenario when all the whistle-blower nodes and all the bridge nodes are rational.

Correctness is broken. Let the number of whistle-blower nodes be denoted by N. When bridge nodes are colluding, all the whistle blower nodes can choose to collude with the bridge nodes for a share in the total malicious transaction value. As N whistle-blowers join $t+1$ bridge nodes, the share of each node can be up to $\frac{L}{N+t+1}$ in one go. One may think that setting the whistle-blower reward to be more than $\frac{L}{N+t+1}$ solves the issue, but it does not.

Since no whistle-blower reports the collusion, the colluding bridge nodes can continue to maliciously transfer a value of L every time window T. This behavior can continue indefinitely without anybody catching it because all whistle-blowers collude. The colluding whistle-blowers keep getting their share and are incentivized against raising a complaint

because once they raise a complaint, their perpetual stream of income of value $\frac{L}{N+t+1}$ for every T window vanishes. So, no finite reward for the whistle-blower can incentivize whistle-blowers to raise a complaint. Hence, the properties of Validity and Liveness would be broken when all the whistle-blower nodes are rational.

We use the discount rate assumption, under which we can still find honest behaviors as the dominant strategy. We assume that the whistle-blowers find the share they get for collusion in the current time window to be more valuable than the same share they get for the subsequent future time windows. For the discount rate, we associate a diminishing function $\delta : \mathbb{N}^+ \mapsto [0, 1]$ to model this perceptible value by the whistle-blower nodes. The domain of δ represents the indexes of the time window starting from the current one, which is 1. The range represents the weight factor of the value. The assumption is stated below

▶ **Definition 5.** *For the whistle-blower nodes, for the current time-window, the weight is* 1*, i.e.,* $\delta(1) = 1.0$ *and for the subsequent time-windows, we have* $\delta(i) > \delta(i+1)$*, for* $i \in \mathbb{N}^+$*.*

With the assumption of Definition 5, the inequality (9) characterizes the reward for the whistle-blowers so that the honest behaviour of raising complaints when bridge nodes collude still remains a dominant strategy.

$$r > \frac{L}{N+t+1} \sum_{i=1}^{\infty} \delta(i) \tag{9}$$

Rational-malicious model. Consider now the collusion with rational whistle-blowers when there are t malicious nodes and the rest of the bridge nodes are rational. Now the adversary has a bribe budget of ω and needs to collude with only one rational bridge node (as explained before). So the factor of t vanishes and ω gets added to L in (9). In case of a collusion event, the whistle-blower nodes get a share of $\frac{L+\omega}{N+1}$. Hence, to have at least one whistle-blower node raise a complaint on a malicious behaviour of bridge nodes as the dominant strategy, the whistle-blower reward should satisfy (10).

$$r > \frac{L+\omega}{N+1} \sum_{i=1}^{\infty} \delta(i) \tag{10}$$

There is no threshold structure on the whistle-blower nodes. We need (any) one whistle-blower node for flagging malicious behavior of bridge nodes. Hence, a rational-malicious model does not apply to the whistle-blower nodes.

We now state the security of our approach through the following theorems:

▶ **Theorem 6** (Security in the Pure Rational Model)**.** *Suppose all the bridge nodes and all the whistle-blowers are rational. Also, suppose that all whistle-blowers hold diminishing valuation (discount rate) of pay-offs as stated in Definition 5.*

Then the HyperLoop protocol of Figures 2 to 5 that includes the batching and non-batching channels, and the enforcement of the limiting sliding-time-window, yields a Nash equilibrium wherein all the correctness properties of Section 4 are satisfied in a mixed networking condition as long as the equations (1) and (9) are satisfied.

▶ **Theorem 7** (Security in Rational-Malicious Model)**.** *Suppose the adversary has a bribe budget of* ω *and controls* t *bridge nodes. Let the rest of the bridge nodes be rational. Let all the whistle-blowers be rational. Also suppose that all whistle-blowers hold diminishing valuation assumption as stated in Definition 5.*

Then the HyperLoop protocol of Figures 2 to 5 that includes the batching and non-batching channels, and the enforcement of the limiting sliding-time-window, yields to a Nash equilibrium wherein all the correctness properties of Section 4 are satisfied in a mixed networking condition as long as the equations (6) and (10) are satisfied.

8 Discussion on the Game-theoretic aspects

Simple majority vs Super majority. We noted in Section 6 that many protocols use a super majority while HyperLoop uses a simple majority threshold. We now further elaborate on the nuances of this.

In both settings, the compromise/collusion of $t+1$ bridge nodes leads to a compromise of the system. The relations between the sliding-time-window value limit and the stakes as expressed in the inequalities (1) - (10) stays the same in both the threshold settings. So, with respect to the individual bridge nodes there is no difference. The individual nodes stake a value of S, get a reward f for honest behavior, and get slashed S when caught colluding. This is so in both the rational and rational-malicious models.

From Supra's point of view, the simple majority setting makes the protocol efficient for the following reasons:

1. For every transfer, only a simple majority of bridge nodes are rewarded with a fee rather than a super majority. This leads to a cost-efficient reward scheme for Supra. The benefits also percolate to the users because their bridge-requests fees feed the rewards for a simple majority of nodes rather than a super majority.
2. To enforce a transfer limit L in a sliding time window, the HyperLoop protocol requires one-third fewer nodes than in a super majority protocol. The cumulative staked value to manage is also one-third less.

Lazy whistle-blowers. Collusion is an unusual event amongst bridge nodes, and this does not often occur, particularly with banning policies in place to impose punitive measures for misbehavior. To motivate the whistle-blowers to consistently monitor for collusion, we periodically reward them by requesting the submission of proof of validation. This validation request may be generated randomly over the registered whistle-blowers. However, all the whistle-blower nodes may collude, and only one among them may be checking (or not) the validity of transactions and subsequently sharing the results with others. Dealing with this scenario requires a much more nuanced approach, which is out of the scope of this paper. However, note that as long as at least a single whistle-blower node checks the validity (which they are rationally motivated to), the correctness properties of the protocol are satisfied.

Code vulnerability attacks. The focus of this paper has been to analyze the bridge protocol in a rational setting that resulted in considering various collusion scenarios. However, most bridge hacks are due to the exploitation of code vulnerabilities in the smart contracts and code of the bridge software, as well as leaks or a compromise of the bridge nodes' private nodes.

We understand that no analyses of the protocol and its features can safeguard against these attacks. These vulnerabilities can be eliminated or, to a large extent, minimized by developing high-quality software code using various means of testing, fuzzing, formal verification, auditing, etc. Similarly, to safeguard against key compromise attacks, we practice industry standards regarding cryptography and operational security.

Insurance mechanism. The Hyperloop protocol mitigates attacks like collusion and malicious behaviors using cryptographic techniques. However, any attacker who hacks into

the smart contracts of source and destination chains can steal the user assets since the sliding window checks on the smart contracts are vulnerable to such attacks. To handle such a case, we introduce insurance pools.

A certain percentage of the client's request fee may be collected and contributed towards an insurance 'pool'. The contribution may also be a fixed value per transaction. This insurance pool covers the client's assets in case of a vulnerability attack. A similar insurance pool may also be maintained on the destination chain to protect the protocol from minting coins utilizing a vulnerability. This insurance mechanism assures the clients and the liquidity providers alike and attracts further liquidity, eventually leading to economies of scale and reducing request fees for the clients.

9 Conclusion

We have designed HyperLoop, a non-interactive multi-sig bridge protocol with novel and distinguishing contributions to the state-of-the-art in cross-chain communications:

- HyperLoop uses a simple majority threshold rather than a super majority threshold yielding an efficient protocol that also enjoys the advantage of requiring a lesser cumulative staked value in order to enforce the same limits of bridge transfers compared to the status quo.
- The stakes, rewards, and penalties for the bridge nodes and whistle-blowers are designed such that the collusion is not profitable, and hence honest node behavior is the Nash equilibrium of the system.

References

1 Chainlink oracle. https://chain.link/data-feeds.
2 Cosmos network. https://cosmos.network/.
3 Polkadot network. https://polkadot.network/whitepaper/.
4 Supra hypernova. https://supraoracles.com/news/HyperNova-Consensus/.
5 Supra intralayer. https://supraoracles.com/intralayer-product/.
6 Tendermint. https://tendermint.com/.
7 Zeta chain. https://www.zetachain.com/.
8 Axelar Network. https://axelar.network/axelar_whitepaper.pdf, 2022.
9 Chainanalysis Report. https://tinyurl.com/rk4zs679, 2022.
10 Layer Zero. https://layerzero.network/, 2022.
11 Wormhole. https://wormhole.com/, 2022.
12 Christian Cachin, Klaus Kursawe, Anna Lysyanskaya, and Reto Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 88–97, 2002.
13 Tobias Distler. Byzantine fault-tolerant state-machine replication from a systems perspective. *ACM Computing Surveys (CSUR)*, 54(1):1–38, 2021.
14 Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. STOC '88, page 148–161, New York, NY, USA, 1988. Association for Computing Machinery.