Proof of Diligence: Cryptoeconomic Security for Rollups

Peiyao Sheng
University of Illinois Urbana-Champaign, IL, USA
Witness Chain, NJ, USA

Ranvir Rana
Witness Chain, NJ, USA

Senthil Bala
Witness Chain, Bengaluru, India

Himanshu Tyagi
Witness Chain, Bengaluru, India

Pramod Viswanath
Princeton University, NJ, USA
Witness Chain, NJ, USA

Abstract

Layer 1 (L1) blockchains such as Ethereum are secured under an "honest supermajority of stake" assumption for a large pool of validators who verify each and every transaction on it. This high security comes at a scalability cost which not only effects the throughput of the blockchain but also results in high gas fees for executing transactions on chain. The most successful solution for this problem is provided by optimistic rollups, Layer 2 (L2) blockchains that execute transactions outside L1 but post the transaction data on L1.

The security for such L2 chains is argued, informally, under the assumption that a set of nodes will check the transaction data posted on L1 and raise an alarm (a fraud proof) if faulty transactions are detected. However, all current deployments lack a proper incentive mechanism for ensuring that these nodes will do their job "diligently", and simply rely on a cursory incentive alignment argument for security.

We solve this problem by introducing an incentivized watchtower network designed to serve as the first line of defense for rollups. Our main contribution is a "Proof of Diligence" protocol that requires watchtowers to continuously provide a proof that they have verified L2 assertions and get rewarded for the same. Proof of Diligence protocol includes a carefully-designed incentive mechanism that is provably secure when watchtowers are rational actors, under a mild rational independence assumption.

Our proposed system is now live on Ethereum testnet. We deployed a watchtower network and implemented Proof of Diligence for multiple optimistic rollups. We extract execution as well as inclusion proofs for transactions as a part of the bounty. Each watchtower has minimal additional computational overhead beyond access to standard L1 and L2 RPC nodes. Our watchtower network comprises of 10 different (rationally independent) EigenLayer operators, secured using restaked Ethereum and spread across three different continents, watching two different optimistic rollups for Ethereum, providing them a decentralized and trustfree first line of defense. The watchtower network can be configured to watch the batches committed by sequencer on L1, providing an approximately 3 minute (cryptoeconomically secure) finality since the additional overhead for watching is very low. This is much lower than the finality delay in the current setup where it takes about 45 minutes for state assertions on L1, and hence will not delay the finality process on L1.

2012 ACM Subject Classification Security and privacy → Distributed systems security

Keywords and phrases blockchain, rollup, game theory, security

Digital Object Identifier 10.4230/LIPIcs.AFT.2024.5

6th Conference on Advances in Financial Technologies (AFT 2024).
Editors: Rainer Böhme and Lucianna Kiffer; Article No. 5; pp. 5:1–5:24
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Related Version *Full Version*: https://arxiv.org/abs/2402.07241

Acknowledgements This work was conducted when all authors were working at Witness Chain.

1 Introduction

L1 Scalability

Blockchain scalability, via improving throughput, latency and transaction fees, is a crucial component of blockchain adoption [18, 30]. Improving the core L1 consensus protocols is a key direction to scalability [8, 17, 48, 10]; as an example, Ethereum upgraded consensus from the longest chain protocol in the proof-of-work (PoW) setting to the GHOST protocol [38] with Casper finality gadget [7] in the proof-of-stake (PoS) setting recently, improving its throughput potentially by three orders of magnitude and reducing 99.95% energy demand [1, 29]. Changes to L1 via upgrading the consensus protocol is a wholesale change (hard fork), requiring significant social consensus around the process and is time-consuming (e.g., Ethereum's upgrade to PoS from PoW took seven years [9]). However the scalability challenges have persisted with increased computational demands and data storage outstripping the upgraded capabilities, with the result that resulting gas fees have not seen any significant reduction. More scaling techniques that maintain equivalent decentralization, known as horizontal scaling, are necessary.

L2 Economy

L2 solutions have emerged as a significant breakthrough, with potential to increase transaction processing speed, cut down costs, and enhance the overall capacity of blockchains. Rollups work by processing transactions outside the L1 chain and then posting the transaction data and state commitments back to it. Established rollup platforms such as Optimism [46] and Arbitrum [23] accommodate a vast ecosystem (e.g., Arbitrum L2 supported more transactions than Ethereum L1 itself in February 2023 [52]). At the same time, new entities [40, 41, 45, 44] are entering the scene, introducing new infrastructure and optimizing their functionalities for specific applications. These new schemes differ from traditional rollups in their targeted functionalities and optimization techniques. Moreover, driven by the demand for customizable and accessible layer 2 solutions, the concept of "Rollups-as-a-Service" is gaining traction [22, 42, 43, 39], allowing a broader range of participants to create and utilize their own rollup strategies.

Rollup security

With the rapid adoption and reliance on rollups, there is an increasing need to address the critical security concern. Current rollup strategies secure the L2 states by requiring asserters to execute and commit L2 block data to the L1 blockchain (Figure 1). These asserters, often staked and centralized, can be objectively slashed when their asserted states are proven to be incorrect. As depicted in the upper path of Figure 1, the system allows applications to independently verify the states committed on L1 and initiate disputes, serving as the secondary line of defense. These disputes are typically addressed and resolved via fraud proofs. However, a basic vulnerability remains despite the security layer provided by staked asserters and fraud proofs: the lack of incentives for actively watching the rollup. In the normal path, there is no guarantee that these applications monitor the asserted states consistently and effectively: how to ensure vigilance during normal path when attention

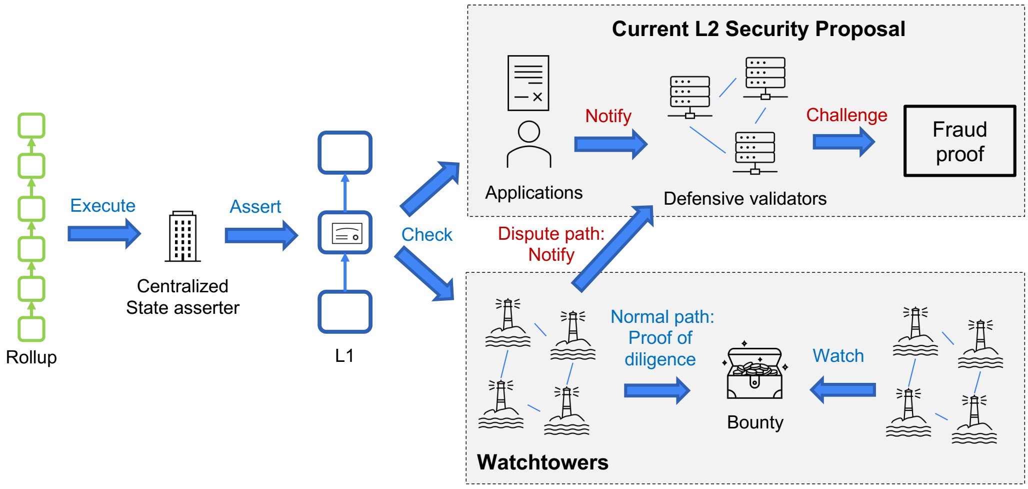

Figure 1 Watchtowers are added to the current L2 security workflow to guard normal path security.

might diminish due to the absence of apparent threats? In other words: *who is watching the watchers?* This problem was identified sharply by the inventors of Aribtrum [16], however a systematic solution has remained open since then.

Watchtowers: the first line of defense for rollups

In this paper we propose a "rational watchtower pool", a group of workers *incentivized* to constantly watch the transactions *in the normal path*. The lower part of Figure 1 illustrates how watchtowers operate independently, interacting with the existing rollup system only when an incorrect state is identified. At that point, they sound an alarm, much like what applications typically do. However, watchtowers are incentivized to stay vigilant at all times. Their role, serving as the first line of defense for rollups, is crucial for identifying potential faults that might otherwise go unnoticed. To ensure that the watchtower fulfill their "watching responsibilities" *diligently*, they must provide what we refer to as "proof of diligence", a prerequisite for earning incentives.

Proof of diligence

Specifically, the duties of watchtowers entail that for new L2 state assertions to be integrated into the L2 ledger, watchtowers must execute the transactions and validate these new assertions. For a watchtower to demonstrate their diligence, their evidence must meet two criteria: (1) those who don't process the transactions should only be able to generate the proof with a negligible likelihood; (2) a proof produced by one watchtower shouldn't be valid for another. These standards ensure only the diligent watchtowers can generate valid proofs, and prevent multiple watchtowers from presenting identical proofs, thus promoting individual effort. To satisfy these conditions, each watchtower computes a verifiable random function (VRF) using the commitment of transaction execution trace. As every watchtower possesses a unique VRF key pair, they generate proofs independently and submit proofs on-chain, allowing public verification. As illustrated in Figure 2, other watchtowers can recompute this proof, ensuring watchtowers presenting false proofs are identified and penalized during disputes.

Incentive framework for watchtowers

For watchtowers to operate effectively, a carefully designed incentive mechanism is vital. This mechanism should consider two parts: positive incentives for continuous monitoring and negative incentives to punish undesirable actions. Positive rewards are allocated using a "bounty mining" scheme. Watchtowers calculate their proof of diligence to determine their eligibility for these rewards. To optimize efficiency, the criteria to obtain the bounty is set by a specific threshold, narrowing down the list of potential recipients. Only the selected winners need to submit their VRF outcomes as their proof of diligence. On the other hand, the negative incentives are implemented through staking. To engage in the protocol, watchtowers must deposit stakes, which are slashable upon detection of misconduct. Monitoring watchtower behavior introduces the problem of "watching the watchtowers". Hence, the incentive structure offers additional rewards for the verification of the proof of diligence. Watchtowers who spot inconsistent proofs can challenge them, earning a reward if their challenge is successful. We rigorously analyze the protocol as a non-cooperative game [26], examining the potential actions of watchtowers: being diligent or lazy. By appropriately configuring these incentives, our findings show that a strategy where all watchtowers diligently monitor rollups is the unique Nash equilibrium, leading to an effective frontline defense for rollups.

Extended mechanism design

Moreover, we extend the action space for watchtowers to encompass potential collusion strategies. Here, several watchtowers might form a collusion to exchange execution results or decide on a mutual random outcome driven by the benefits of saving computational costs. Our findings indicate that when factoring in such cooperative actions, the equilibrium tends to favor collusion, nullifying the role of watchtowers. To counteract this, we introduce design enhancements to disrupt collusive behaviors. We design a whistleblower scheme allowing any colluder to secretly expose collusion in exchange for compensation. Our game-theoretic analysis demonstrates that the whistleblower system encourages betrayals against the collusion. Since this reporting remains confidential, while the act of betrayal is noticeable, the whistleblower can not be individually attributed. Consequently, the collusion will not be initiated at the first place.

System design

By integrating the proof of diligence and incentive mechanism, the design enforces watchtowers to maintain their essential roles in guarding the security of rollups in normal path. The network consists of a pool of watchtowers that are allocated to one of the participating rollups randomly, with the allocation changing over time. The system utilizes a staking mechanism to ensure sybil resistance and fair allocation of positive incentives; the stake is also used as a bond that can be utilized for negative incentives. On-chain contracts perform the broad activities of watchtower registration, rollup status monitoring, and disbursement of incentives. The off-chain client comprises a wrapper to the L2 full node that fetches intermediate information to mine the bounty. The system is implemented on the Optimism Bedrock stack [47] that watches Optimism and Base rollups on the Goerli testnet. Note that since watchtowers operate independently of the optimistic rollup framework, the system is capable of monitoring multiple optimistic rollup chains simultaneously. We use Eigenlayer (EL) [14] as the staking mechanism and build our registration functionality associated with it. Our implementation adds a very low compute cost to an L2 full node with a low transaction fee that can be adjusted on a sliding scale.

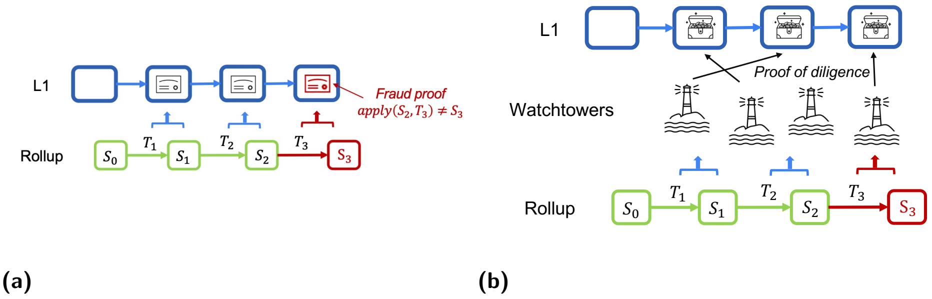

Figure 2 (a) Optimistic rollup model and (b) Watchtower model.

The subsequent sections provide a detailed breakdown of our investigative approach and findings. In Section 2, we outline the system's model and explore various threat models. In Section 3, we compare prior work on related topics with our solution. Section 4 presents an in-depth view of our developed watchtower protocol. Further, in Section 5, we extend the model to allow collusions and conduct a thorough cooperative game theoretic analysis of the security aspects and the incentive compatibility associated with the enhanced protocol. Section 6.1 demonstrates our system design and implementation details, the evaluations results from the live system are presented in Section 6. Concluding the paper, Section 7 engages in a comprehensive discussion, bringing to the fore the pivotal learning from our research and suggesting future directions.

2 Security Models and Definitions

2.1 Rollup Model

Rollups enhance the blockchain's efficiency by handling transaction execution off the main chain (L1). Present rollup techniques rely on either the validity proofs or fraud proofs to ensure security. Validity proofs, used in schemes known as zk-rollups, apply sophisticated cryptographic techniques to validate every batch of L2 transactions. On the other hand, fraud proofs, employed by optimistic rollups, come into play only when a fault is spotted in the execution process. Our focus is on enhancing the security of optimistic rollups (Figure 2a). In this context, the watchtower pool has been introduced to monitor the normal-path operations – that is, the process when transactions are presumed to be valid unless challenged.

In our system, there exists an L2 chain employing optimistic rollup scheme. The scheme involves an untrusted *asserter*, responsible for processing the L2 blocks and submitting the updated states assertion on L1 chain. L1 chain is guaranteed to be secure and live. We assume the data of L2 blocks are stored on L1 directly or through a trusted data availability service, hence the data is ensured to be retrievable. Furthermore, the system consists of a group of *defensive validators*, whose role is activated when there is a need to produce a fraud proof, which is verifiable on chain. We consider a *live* rollup system operator, who coordinates the issuance of L2 blocks (typically through a role called sequencer) and provides rewards for asserter, validators, and watchtowers. The goal of the operator is to ensure the security of rollup states with minimal cost.

2.2 Watchtower Model

Independent of the rollup infrastructure, our model incorporates a pool of n registered entities known as *watchtowers* (Figure 2b), denoted as $\{W_1, W_2, \cdots, W_n\}$. Each watchtower W_i has deposited relative stake α_i into the system, where $\sum_i^n \alpha_i = 1$ and the total amount of stake is $\mathcal{S}$. These watchtowers are modeled as individual *rational adversary*, where the term "rational" implies that they operate with the purpose of optimizing a known payoff function. And they have the ability to methodically process all potential scenarios and select strategies that provides the best outcomes.

Even though rational participants will not deviate from the protocols without cause, they might engage in attacks – either intentionally or inadvertently – if the expected rewards justify such actions. We identify several potential attacks in this framework, emphasizing the nuanced strategies rational actors might employ.

Lazy watchtower

Since watchtowers are also required to execute the entire batch of L2 transactions, they can earn rewards by performing their duties diligently. This role closely resembles that of the asserter in the original optimistic rollup system, who posts computation results in exchange for rewards. As a result, one prominent challenge the watchtower design must confront is the "lazy watchtower" problem. This issue arises because of two main reasons: (1) rational watchtowers may submit arbitrary responses if the results lack verification process, and (2) they might opt out of protocol participation if the associated costs outweigh potential rewards. In essence, the watchtowers must provide a form of evidence for their work and the protocol must offer sufficient incentives to encourage participants to actively and consistently perform tasks.

Collusion attack

Rational entities might form collusion where several parties conduct coordinated actions based on a common agreement. However, it's essential to note that, despite the existence of any agreement, colluding parties maintain individual autonomy and continue to prioritize their self-interest. Within the scope of collusion attacks, we consider adaptive adversaries who can adjust their collusion strategies in response to protocol developments.

In our system, we assume that adversaries are limited by computational constraints, so that they are not able to break the security of necessary cryptographic primitives. It's important to highlight that rational adversaries, guided by their payoff functions, are considered weaker threats compared to Byzantine adversaries. Since their profit-driven actions exclude certain strategies. In the discussion section (Section 7) we explore the trade-off between different levels of security and associated costs.

2.3 Preliminaries

We introduce fundamental primitives utilized throughout the paper. Other notations are summarized in Table 1.

Verifiable random functions

Our protocol use verifiable random functions (VRFs), providing two functions to generate and verify proofs. $\mathsf{VRF}_{sk}(x)$ processes an input x and returns two values (d, π_d): a normalized hash digest d and a proof π_d. The value $d \in [0, 1)$ is uniquely determined by the input x

Table 1 Summary of notations.

Term	Definition	Term	Definition
n	Number of registered watchtowers	c_T	Cost per transaction batch
W_i	Watchtower i	c_V	Cost of resolving disputes
α_i	Relative stake of watchtower W_i	R_B	Normal path reward for watchtowers
$\mathcal{S}$	Total amount of stake in the system	R_C	Dispute path reward for watchtowers
S	Blockchain state	R_w	Whistleblower protocol reward
T	A batch of L2 transactions	$\phi(\alpha_i)$	Bounty mining threshold
E	Execution trace	t	Deposit for participating in collusion
r_S	State assertion	h	Rent in diligent collusion
r_E	Merkle root of execution trace	n_c	The size of the colluding group

and a secret key sk, and is indistinguishable from a random value to anyone that does not know sk. $\mathsf{verifyVRF}_{pk}(\pi_d, d, x) \in \{\mathsf{true}, \mathsf{false}\}$ takes the proof π_d input and allows anyone who knows the public key pk to verify whether d is the correct value computed from x and sk.

Merkle trees

Merkle trees are a fundamental data structure in cryptography, summarizing lists of items, such as transactions or states, by concatenating their cryptographic hashes at various levels of the tree. We provide the function $\mathsf{Merklize}(L) \to r$ that generates a Merkle root r from a list of items L. It involves the construction of a Merkle tree or Patricia trees [28, 51] for L and then produces the root hash that represents the entire list of items. $\mathsf{MerkleProof}(r, l, L) \to p$ is used to generate a Merkle proof p associated with a leaf node l in a tree rooted at r. This proof consists of the minimal amount of information needed to confirm the presence of the specific leaf node l within the tree constructed from L. Furthermore, there exists a validation function $\mathsf{verifyMerkleProof}(r, l, p) \in \{\mathsf{true}, \mathsf{false}\}$ that takes in a root r, a leaf l, and a Merkle proof p, then returns a boolean value indicating whether p demonstrates that l is indeed part of the Merkle tree rooted at r.

State transitions

We consider a general model for L1 blockchain, which keeps the latest states set S and transactions organized as blocks within hash-based chains. The function $\mathsf{apply}(S', T) \to (S, E)$ represents the process of applying a list of transactions T to a prior state S' to yield a new state S and an execution trace E. The execution trace is a detailed record of all intermediate states during the transition from S' to S, serving as a reference for verification. Another function $\mathsf{validate}(r, r', L) \in \{\mathsf{false}(r), \mathsf{false}(r'), \mathsf{false}(r, r')\}$ resolves a conflict between two Merkle roots r and r' calculated from the same list of items L. The output represents the subset of roots that are proven to be invalid. There are different ways to implement this function, such as using L1 as a trusted third party to provide ground truth or executing an interactive verification game (IVG) to identify the incorrect root.

Game theory

The concepts about game theory utilized in our analysis are all defined in the book [26]. We first analyze the rollup security with watchtowers as a non-cooperative game in strategic form (Def. 4.2, [26]), and examine the dominance of strategies (Def. 3.10 and 4.6, [26]) to find the pure strategy Nash equilibrium (Def. 4.17 and Def. 5.3, [26]) for watchtowers. We also discuss an enhanced protocol considering cooperative game (Chapter 15, [26]), where more than one Nash equilibria exist and Pareto efficiency (Def. 15.7, [26]) is considered.

3 Background and Prior Work

The dialogue surrounding optimistic rollups originated in community discussions on the Ethereum research forum [2], where developers and researchers shared insights about potential scalability solutions. One of the earliest detailed presentations came from the Optimism team, who outlined the foundational framework of optimistic rollups and their theoretical implications [21]. The theoretical cornerstone for optimistic rollups was further strengthened through scholarly research. Pioneering works such as Arbitrum [19] and TrueBit [49] introduced essential concepts related to off-chain computation and dispute resolution.

Subsequent discussions began to address broader implications in contexts such as data availability [3, 53, 34], sequencer risks [27, 32], and validator behavior [20, 16, 33, 15, 13]. Specifically, among these topics, validator behavior is most pertinent to this paper. While the basic rollup design discusses an elementary incentive structure to motivate asserters to post correct states, verifiers may lack incentives to monitor the system's states diligently, such a situation is known as the verifier's dilemma problem [24]. To address this issue and enhance system security, the authors of Arbitrum [16] proposed an attention game in which verifiers who fail to participate may be punished. TrueBit [20] also suggested an enhanced mechanism to select a pool of validators, requiring them to submit a proof of independent work [15] to enforce verification. Additionally, some research [33] provides game-theoretic analyses of collusion risks in incentivized computation outsourcing, proposing mitigation strategies, though not completely resolving the issue. In the field of verifiable outsourced cloud computing, [13] investigates the dynamics between clients and workers within smart contract frameworks, demonstrating a preference for honest behavior under certain conditions. Although they propose a "traitor contract" to eliminate collusion, this is limited to a two-party context. Furthermore, we contend that this proposed solution might not be effective, as the contract's output could compromise the traitor's secrecy. Another study [35] examines cooperation in N-person prisoner's dilemma scenarios, with institutional arrangements akin to smart contracts. Differing from our focus, this work considers collusion through bargaining rather than a leader-based approach. A new design of the attention game was proposed in a recent paper [25] to find the optimal number of validators that minimizes failure probability. The design only provides probabilistic security and requires modification in the underlying rollup protocol to ensure deterministic security. As a comparison, we provide a plug-and-play solution that can be used for any off-chain compute resource with minor modifications.

4 The Watchtower Protocol

4.1 Proof of Diligence

In our protocol, we focus on a single task that watchtowers undertake: verifying the updated state assertion r_S of the L2 blockchain, a state assertion is the Merkle root of the state tree, calculated by $r_S = \mathsf{Merklize}(S)$. The states S on a blockchain consist of a list of key-value pairs, such as the account address and the account balance. Formally, given the latest validated state S' and a sequence of transactions T, the responsibility of the watchtowers is to verify r_S by executing the computations specified in Algorithm 1.

Algorithm 1 Function for Watchtowers.

```
function CHECKSTATE($S', T, r_S, \alpha_i$)
    $(S, E) \leftarrow \mathsf{apply}(S', T)$
    $r'_S = \mathsf{Merklize}(S)$
    $r_E = \mathsf{Merklize}(E)$
    $(d, \pi) = \mathsf{VRF}_{sk}(r'_S | r_E)$
    if $d < \phi(\alpha_i)$ then
        submit proof of diligence $(d, \pi)$
    if $r_S = r'_S$ then
        Return true
    else
        Return false
```

Algorithm 1 represents the process of assessing whether the proposed state assertion r_S is indeed consistent with the ledger history. The watchtower calling the checkState function first applies the transactions T to the initial state S', which returns the new state S and an execution trace E. The watchtower then calculates the Merkle root of the legitimate state S and compares the result r'_S with the posted r_S. If the verification process succeeds and no faults are found, the watchtower considers the new state S as validated. Otherwise, the watchtower is expected to raise an alarm to the rollup scheme, activating defensive validators for dispute resolution and fraud proof creation.

Besides, the watchtowers utilize the execution trace E to construct a Merkle root r_E. Since r_E is not available anywhere and can only be derived by executing the transactions, it serves as evidence of their work and diligence. They compute a VRF using r_E and r'_S, incorporating their secret keys. It is assumed that the corresponding public keys were disclosed during the registration phase. The VRF produces (d, π); the digest d is subsequently used to allocate rewards for diligent watch. Accompanied by the proof π, the watchtower submits this as the *proof of diligence*, which satisfies the following properties:

- **Verifiability.** Given a keypair (pk, sk), for any input x, if $(d, \pi) \leftarrow \mathsf{VRF}_{sk}(x)$, then $\mathsf{verifyVRF}_{pk}(\pi, d, x) = \mathsf{true}$.
- **Uniqueness.** Given a keypair (pk, sk), for any input x, if $(d, \pi) \leftarrow \mathsf{VRF}_{sk}(x)$, no one, including the key owner, can produce a different $d' \neq d$ and the associated proof π' such that $\mathsf{verifyVRF}_{pk}(\pi', d', x) = \mathsf{true}$.
- **Pseudorandomness.** For a given input x and public key pk the output d is indistinguishable from a truly random string to anyone who does not possess the private key sk.

The verifiability ensures that once the proof is posted on the chain, it can be verified by every watchtower using r_S, r_E, and the public key pk. This process is referred to as "watching the watchtowers". Other watchtowers will use their own r_S, r_E values to check the proof. If they detect any inconsistencies, they invoke the validate function to resolve conflicts on the chain. After a predefined challenge period t_C, the protocol concludes that the remaining proofs are correct. The uniqueness and pseudorandomness imply that only diligent watchtowers can generate a valid proof, and the proof generated by one watchtower cannot be used by others.

4.2 Incentive Design

To ensure that watchtowers execute their verification duties with diligence, it's imperative to institute a carefully designed incentive mechanism. First of all, this mechanism mandates that all enrolled watchtowers use their stakes as the deposit. This deposit acts as a form of commitment to honest service – any verified misbehavior results in the slashing of their deposit.

Bounty mining

Understanding that watching operations incur costs, represented as c_T, in the process of executing all transactions in T, we introduce a bounty mining scheme to motivate the watchtowers to perform the verification (specified in Algorithm 1, line 4-7). The process of bounty mining can be analogous to the process of committee or leader selection in some blockchains [8, 11], where a VRF is computed to determine bounty winners. Formally, a diligent watchtower W_i with relative stake α_i who performs the transaction execution can generate a proof of diligence (d, π) on the task. Then with a probability $\phi(\alpha_i)$, W_i finds that its proof satisfies a certain condition (specified in Eq. 1), allowing them to receive a bounty by publishing the proof. A system parameter θ is defined to control the probability that a party with all stake wins the bounty.

$$\Pr[\mathsf{VRF}_{sk_i}(r_S|r_E).d < \phi(\alpha_i)] = \phi(\alpha_i) = 1 - (1-\theta)^{\alpha_i} \tag{1}$$

The amount of bounty each winner who submits a valid proof (d, π) can collect is a constant value R_B. If any watchtower identifies an incorrect proof, the validate function will be invoked to resolve the dispute. In this process, both watchtowers are required to publish their r_S, r_E values. The winner of the challenge will receive a constant reward of R_C, and a compensation distributed among all winners for the cost of dispute resolution c_V. The rewards setting satisfies the following conditions:

$$R_B > \frac{c_T}{\phi(\alpha_0)},\ R_C > c_T \tag{2}$$

where α_0 is the unit stake fraction, hence $\alpha_0 \le \min\{\alpha_i\}_{i\in[1,n]}$. To ensure that the slashed deposit is enough to pay for the cost of dispute resolution and rewards, we require that

$$\alpha_0 S \ge c_V + (n-1)R_C \tag{3}$$

In summary, the entire protocol works as follows.

1. When a new state assertion r_S is published, a bounty timer t_1 starts. Each watchtower recomputes the assertion r'_S and generates the execution trace root r_E by applying transactions T to the old states S', then computes $(d, \pi) = \mathsf{VRF}_{sk}(r'_S, r_E)$.
2. If the assertion is incorrect ($r_S \ne r'_S$), the watchtower notifies the defensive validators of the rollup to initiate a challenge.
3. If a watchtower wins the bounty, the watchtower submits proof of diligence (d, π) before t_1.
4. When a watchtower observes a proof of diligence submitted by other watchtowers, it verifies the proof using the other's public key and the execution trace root r_E calculated by itself. If the proof is incorrect, the watchtower calls validate interface to resolve the dispute. The cost of triggering validate is denoted as c_V shared among all watchtowers who call the function.
5. If no validate is triggered before t_1 expires, the rollup operateor concludes that the asserted execution trace root is correct. Validated bounty winners receive R_B as reward each.
6. If validate is triggered, the winning parties receive a reward R_C and a compensation for the shared cost, and the losing parties lose all the stake.

4.3 Incentive Analysis in Non-Cooperative Games

We first consider the proof of diligence protocol as a non-cooperative game denoted as PoD-Game, where different watchtowers optimize their individual payoff without any mutual agreement for cooperation. Watchtowers can adopt one of two possible strategies: diligent or lazy. Diligent watchtowers execute transactions honestly and report proof of diligence when the condition is met. In contrast, lazy watchtowers opt for generating a random result as the new state assertion, incurring negligible cost. Moreover, these lazy watchtowers might submit a fake proof, computed from the random root, to deceitfully claim the bounty. Notably, we consider only those lazy *and deceitful* watchtowers, as non-deceitful lazy watchtowers are indistinguishable from non-participants in impact. A default constraint of our incentive mechanism is that the payoff for diligent behavior is always positive, which dominates the non-participating strategy. Therefore, we omit this trivial case in the subsequent analysis.

Let $u_i^a(n_d)$ be the expected payoff function of watchtower W_i given that watchtower W_i choose action $a \in \{d, l\}$, where d and l represent diligent and lazy strategies respectively, and there are n_d diligent watchtowers in total. According to the protocol, for all $i \in [1, n]$, these payoff functions are defined as follows:

$$u_i^d(n_d) = \begin{cases} \phi(\alpha_i)R_B + R_C - c_T & n_d < n \\ \phi(\alpha_i)R_B - c_T & n_d = n \end{cases} \tag{4}$$

$$u_i^l(n_d) = \begin{cases} \phi(\alpha_i)R_B & n_d = 0 \\ -\alpha_i S\phi(\alpha_i) & n_d > 0 \end{cases} \tag{5}$$

Note that a lazy watchtower would attempt to mimic genuine probability to submit proofs, otherwise the proof submission frequencies that can happen with negligible probability can be used to detect malicious behaviors. By comparing the payoff for different strategies, we observe that the diligent strategy dominates the lazy strategy for all watchtowers; formally, we have the following theorem.

▶ **Theorem 1.** *The diligent strategy in the PoD-Game is a dominant strategy for all watchtowers.*

Proof. For every watchtower W_i, given an arbitrary strategy vector s_{-i} containing all others' actions. Let n_d denote the number of watchtowers that are diligent in s_{-i}, we compare the payoff of two strategies for W_i below.

- Case 1: If $n_d = 0$, due to Eq.2, $u_i^d(1) = \phi(\alpha_i)R_B + R_C - c_T > \phi(\alpha_i)R_B = u_i^l(0)$.
- Case 2: If $0 < n_d < n-1$, due to Eq.2, $u_i^d(n_d+1) = \phi(\alpha_i)R_B + R_C - c_T > -\alpha_i S\phi(\alpha_i) = u_i^l(n_d)$.
- Case 3: If $n_d = n-1$, due to Eq.2, $u_i^d(n) = \phi(\alpha_i)R_B > -\alpha_i S\phi(\alpha_i) = u_i^l(n-1)$. ◀

Theorem 1 implies that the game has a unique Nash equilibrium since we can eliminate the strictly dominated lazy strategy and get a unique strategy vector of all diligence; this follows directly from Cor.4.37 [26].

▶ **Corollary 2.** *The unique Nash equilibrium in PoD-Game occurs when every watchtower is diligent.*

In conclusion, in a setting without cooperation, our proof of diligence protocol ensures that rational watchtowers will always work diligently, providing the first line of defense for the rollup system. In practice, this setting can model many situations, such as when watchtowers cannot communicate with each other, or when there is a public reputation system where participating in any cooperation would be detected and deteriorate reputations.

5 Cooperative Games and The Enhanced Protocol

In the PoD-Game, we observe that if all watchtowers choose to be lazy and agree to use a common random r_E and r_S to compute their proofs, they will receive a higher payoff than in the all-diligent equilibrium. However, this strategy did not get chosen since any party can deviate from this "unreliable collusion" to achieve higher utility, while lazy parties end up losing all their stakes.

Collusion can be reinforced by adding specific punishment mechanisms. In the context of rollups, smart contracts are the most viable tools for enforcing agreements or promises of such cooperation. Beyond the previously mentioned lazy collusion strategy, other strategies may improve payoff through cooperation, like sharing the costs of diligence. Additionally, the process of forming a colluding group can vary. For instance, in a leader-based method, a watchtower might take the initiative to create a colluder contract [13], setting conditions for joining and outlining actions to be taken, thereby allowing others to join. Conversely, in a leader-less method, watchtowers can choose to join a collusion group and negotiate group strategy collectively [35].

In this section, we explore the space of collusion, concentrating on two main leader-based strategies applicable to most relevant settings. Our primary findings reveal that establishing mutual agreements enforced by smart contracts makes lazy actions more beneficial. To counteract this, we propose setting up Whistleblower contracts, which encourage colluders to betray their collusion, thereby eliminating the lazy equilibrium.

5.1 Lazy Collusion

One possible strategy that watchtowers might employ to solidify the collusion is to require all colluders to deposit a certain amount of stakes into the collusion. If a colluder posts a proof computed from different execution roots, they will lose the collusion deposit.

We assume any party is capable of initiating such collusion, and we refer to that party as the leader. A leader will specify the amount of stake t that each newly joined colluder needs to contribute. Since the collusion is motivated by the benefit of being lazy, we term this strategy "lazy collusion." We observe that watchtowers choosing not to join the collusion perceive the same game as PoD-Game, and therefore, they are likely to adopt a diligent strategy. If such independent watchtowers exist, lazy collusion will not gain the expected advantage by not computing the results. Consequently, collusion will only be effective when all watchtowers participate in it. Specifically, the process of forming lazy collusion unfolds as follows:

1. A watchtower initiates collusion by placing a deposit of t. The watchtower also releases a randomly chosen r'_E.
2. Other watchtowers may join the collusion by placing a deposit of t. If n watchtowers join the collusion before t_{lc}, the collusion is formed. Otherwise, watchtowers get back their deposits.
3. During the watching phase, all colluders are required to calculate the proof of diligence using $\mathsf{VRF}_{sk}(r_S, r'_E)$, where r_S is the state root posted by the asserter.
4. If a colluder becomes a winner, the collusion protocol will check whether the winner's proof is calculated from r'_E, if not, the winner is considered a traitor and will lose t.
5. At the end of the collusion, colluders who do not betray the collusion receive $t + n_t t/(n_c - n_t)$, where n_c is the size of colluding group, n_t is the number of traitors. If all colluders betray, everyone gets back their deposit t.

We are considering two possible actions that all colluders (including the leader) can take, given the collusion strategy selected by the leader: obey and betray. Colluders who choose to obey the strategy follow the leader's instructions to submit a response calculated from the specified r_E, while those who choose to betray may submit something different, driven by personal interest. The game induced by lazy collusion is denoted as LC-Game. Let $u^a_{l_i}(n_o)$ be the expected payoff function of the i-th colluder $W_{l_i}(i \in [1, n_c])$. $a \in \{o, b\}$ represents for the action chosen by W_{l_i}, with o as obey and b as betray. There are n_o colluders who choose to obey the collusion strategy. According to the protocol, $n_c = n$, so we let $l_i = i$ for simplicity, and the payoff functions can be written as follows:

$$u^o_i(n_o) = \begin{cases} -\alpha_i \mathcal{S}\phi(\alpha_i) + \frac{(n-n_o)t}{n_o} & n_o < n \\ \phi(\alpha_i)R_B & n_o = n \end{cases} \tag{6}$$

$$u^b_i(n_o) = \begin{cases} \phi(\alpha_i)R_B - c_T & n_o = 0 \\ \phi(\alpha_i)R_B + R_C - c_T - t & n_o > 0 \end{cases} \tag{7}$$

Now when there exists such lazy collusion, we compare the payoff of colluders with different actions and observe that obeying the group strategy dominates the betrayal for all colluders when the deposit t is high enough, formally, we have the following theorem.

▶ **Theorem 3.** *The "obey" strategy in the LC-Game is a dominate strategy for colluder W_i if the following conditions hold:*

$$t > R_C - c_T \tag{8}$$

$$t > \frac{n-1}{n}\left(\alpha_i \mathcal{S}\phi(\alpha_i) + \phi(\alpha_i)R_B + R_C - c_T\right) \tag{9}$$

Proof. For every colluder W_i, given an arbitrary strategy vector s_{-i} containing all others' actions. Let n_o denote the number of colluders that obey the collusion in s_{-i}, we compare the payoff of two strategies for W_i below.

- Case 1: If $n_o = n - 1$, due to Eq. 8, $u^o_i(n) = \phi(\alpha_i)R_B > \phi(\alpha_i)R_B + R_C - c_T - t = u^b_i(1)$.
- Case 2: If $0 < n_o < n - 1$, due to Eq. 9 , $u^o_i(n_o + 1) = -\alpha_i \mathcal{S}\phi(\alpha_i) + \frac{(n-n_o-1)t}{n_o+1} > \phi(\alpha_i)R_B + R_C - c_T - t = u^b_i(n_o)$.
- Case 3: If $n_o = 0$, due to Eq. 9, $u^o_i(1) = -\alpha_i \mathcal{S}\phi(\alpha_i) + (n-1)t > \phi(\alpha_i)R_B - c_T = u^b_i(n)$. ◀

Then we consider the game combining PoD-Game and LC-Game. It starts with an initiation phase where watchtowers may choose to initiate a contract to become the collusion leader. If all watchtowers join the same collusion contract, which is the only case where collusion will be formed successfully, and the conditions specified in Eq.8 and 9 are satisfied, this sub-game LC-Game can be eliminated according to Theorem 3, with the payoff induced by the dominant strategy. Otherwise, independent watchtowers will follow the PoD-Game and iterative elimination can be applied with Theorem 1. Observing that the payoff derived from LC-Game is higher than PoD-Game, we find the game has two Nash equilibria but only the lazy collusion strategy is Pareto efficient.

▶ **Corollary 4.** *In PoD-Game that allows lazy collusion, there are two Nash equilibria: (1) all watchtowers are independently diligent (2) all watchtowers are collusively lazy. The second equilibrium is Pareto efficient.*

Table 2 DC-Game: The game induced by diligent collusion.

Payoff $\begin{pmatrix} u_{d_1} \\ u_{d_i} \end{pmatrix}$		follower	
		All join	Not all join
leader	Obey	$\begin{pmatrix} \phi(\alpha_{d_1})R_B - c_T + (n_C - 1)h \\ \phi(\alpha_{d_i})R_B - h \end{pmatrix}$	
	Betray	$\begin{pmatrix} \phi(\alpha_{d_1})R_B - c_T + R_C - \frac{c_V}{n-n_C+1} - t \\ -\alpha_{d_i}\mathcal{S}\phi(\alpha_{d_i}) + \frac{t}{n_C-1} \end{pmatrix}$	
	Cheat	$\begin{pmatrix} \phi(\alpha_{d_1})R_B + (n-1)h \\ \phi(\alpha_{d_i})R_B - h \end{pmatrix}$	$\begin{pmatrix} \phi(\alpha_{d_1})R_B + (n-1)h \\ \phi(\alpha_{d_i})R_B - h \end{pmatrix}$

5.2 Diligent Collusion

Moreover, watchtowers might choose to remain diligent while seeking to form a collusion to share the execution costs. In this scenario, the leader initiating the collusion carries out the computations and commits its solution to the group. Anyone wishing to join the collusion is required to contribute a fee of $h < c_T$ to access the results. Subsequently, all colluders utilize the execution root, as calculated by the collusion leader, to generate proof of diligence and claim bounties. We refer to this strategy as "diligent collusion". The process for forming such a diligent collusion unfolds as follows:

1. A watchtower initiates collusion by placing a deposit of t. The watchtower also commits a computed r'_E and specifies a rent $h < c_T$.
2. Other watchtowers may join the collusion by paying the rent of h. Then the committed r_E is revealed.
3. During the watching phase, all colluders are required to calculate the proof of diligence using $\mathsf{VRF}_{sk}(r_S, r'_E)$.
4. If the proof a non-leader colluder submits is recognized as a faulty proof, the leader will lose t, and others will get back $t/(n_c - 1)$, where n_c is the size of the colluding group.
5. If the proof provided by leader gets accepted, the leader gets $t + (n_c - 1)h$.

In the game of diligent collusion, we observe that the leader has three possible actions: obey, betray, and cheat. "Obey" implies that the leader will diligently compute the transaction execution root and share it with others. "Betray" suggests that the leader might commit a random output to the colluding group while submitting a correct proof for its own benefit. And "cheat" represents the scenario in which the leader lazily commits and submits the same random output. The other colluders may only choose to follow what the leader commits, since they pay the rent; in other words, there is no benefit to join the collusion if they plan to choose another action. Table 2 lists the payoff functions of DC-Game, the game induced by the diligent collusion strategy. u_{d_1} and $u_{d_i} (i \in [2, n_c])$ represent for the payoff function for the leader and other colluders. Note that if the leader chooses to cheat, its payoff is highly influenced by whether all watchtowers join the collusion. If there exist independent watchtowers, they must choose the diligent strategy as the PoD-Game implies, then all colluders will be punished by the proof of diligence protocol. Therefore, it is evident that the following theorem holds:

▶ **Theorem 5.** *DC-Game has no pure strategy Nash equilibrium when* $t > R_C - c_V/(n-1) - h$.

Proof. We denote the strategy profile in DC-Game as $\{a, n_c\}$, where $a \in \{o, b, c\}$ is the action chosen by leader, representing obey, betray and cheat, n_c is the number of other watchtowers who choose to join the collusion. Firstly, the condition $t > R_C - c_V/(n-1) - h$

implies that $\forall n_c \in [2, n], u_{d_1}(\{o, n_c\}) > u_{d_1}(\{b, n_c\})$, hence betray is strictly dominated by obey. We then observe that $u_{d_1}(\{o, n\}) < u_{d_1}(\{c, n\})$, indicating that $\{o, n\}$ is not a Nash equilibrium, as in this case the leader can achieve a higher payoff by switching to cheat. Additionally, independent watchtowers receive a higher payoff if they join the collusion when the leader chooses to obey, as this spreads the execution cost across the entire colluding group. Conversely, when the leader opts to cheat, if all watchtowers join the collusion, switching to be independently diligent achieves a better payoff. However, if not all watchtowers join, obey becomes the more beneficial strategy for the leader. Consequently, there doesn't exist any pure strategy that is a Nash equilibrium. ◀

Theorem 5 indicates that even if we take DC-Game into consideration, the pure strategy Nash equilibria of the full game remain the same. Hence we have the following property.

▶ **Corollary 6.** *In PoD-Game that allows lazy and diligent collusion, there are two Nash equilibria: (1) all watchtowers are independently diligent (2) all watchtowers are collusively lazy. The second equilibrium is Pareto efficient.*

5.3 Enhanced Protocol with Whistleblower

The incentive for colluders to establish collusion lies in the potential to mine bounties with less effort. However, less effort always correlates with the likelihood of faulty proofs. As analyzed in Section 5.1, watchtowers are incentiviced to join the lazy collusion. In lazy collusion, the benefits derived from betraying the collusion (by being diligent in computations) do not suffice to offset the loss of the collusion deposit. Thus, the principles for eliminating the collusion are to: (1) offer rewards for identifying and reporting collusion, and (2) provide compensation for the losses incurred from betraying the collusion. We refer to such colluders, who disclose information about collusion to the rollup system, as "whistleblowers". In response, we have specifically designed the whistleblower protocol as follows:

1. The rollup operator establishes a whistleblower bounty R_w and declares that the first whistleblower will be eligible for the reward.
2. Any individual can place a deposit of d and submit the correct r_E to assume the role of a whistleblower.
3. The protocol invokes the validate interface to resolve the dispute. If the whistleblower succeeds, they receive $R_w + d$ in return. Otherwise, a loss results in the forfeiture of their deposit.

To ensure the payoff of whistleblower is better in all above collusion games, we derive the following condition to determine rewards.

▶ **Lemma 7.** *With the additional action "report" that each watchtower can choose, the strategy that all watchtowers obey the lazy collusion is no longer a Nash equilibrium for watchtower i if*

$$R_w > \phi(\alpha_i)R_B + c_V + \alpha_i \mathcal{S}\phi(\alpha_i) + c_T \tag{10}$$

Proof. We first consider the impact that the additional action *report* brings to LC-Game. First, the whistleblower, by adhering to the collusion agreement to submit proof, will not be subject to punishment by the slashing rule of the collusion. However, the payoffs of other colluders will be reduced due to the exposure of the collusion, hence the changes in outcomes are detectable and might be used to augment the collusion deposit. Even though, the act of reporting cannot be traced back to an individual colluder. Therefore, any colluder may switch to report to gain higher payoff from the whistleblower protocol, since

$$\begin{aligned} u_i^r(n) &= -\alpha_i \mathcal{S}\phi(\alpha_i) - c_V + R_C - c_T + R_w && (11) \\ &> \phi(\alpha_i)R_B + R_C > \phi(\alpha_i)R_B = u_i^o(n) && (12) \end{aligned}$$

This further discourages other colluders from participating in the collusion, because the payoff $\hat{u}_i^o(n)$ they get in the existence of whistleblower becomes

$$\hat{u}_i^o(n) = -\alpha_i \mathcal{S}\phi(\alpha_i) < \phi(\alpha_i)R_B - c_T - c_V/2 + R_C = u_i^d(n) \quad (13)$$

As a result, joining and obeying the lazy collusion is not a Nash equilibrium.

Then we discuss the impact of the whistleblower scheme and the elimination of the lazy equilibrium on DC-Game. We denote the strategy profile in DC-Game as $\{a, n_c, w\}$, where $a \in \{o, b, c\}$ is the action chosen by the leader, n_c is the number of watchtowers in the collusion, and $w \in \{\mathsf{true}, \mathsf{false}\}$ represents whether there exists a whistleblower. First, all strategies with $w = \mathsf{false}$ are not Nash equilibria by the same analysis in Theorem 5. Next, since the existence of a whistleblower will not lower the payoff of a leader who chooses to obey, betrayal is still strictly dominated by obedience. Then we consider the strategy where a whistleblower exists in the collusion group. In this case, if the leader opts to cheat, it's always better to switch to obedience. However, if the leader chooses to obey, the whistleblower is better off choosing not to report. Consequently, there doesn't exist any pure strategy in the subgame DC-Game that is a Nash equilibrium. ◀

The introduction of a whistleblower protocol changes the payoff dynamics of lazy collusion, as any colluder can expose the collusion for a higher payoff. Knowing this, the leader may not choose to initiate lazy collusion in the first place and the full game reach back the state where diligent strategy is the only Nash equilibrium.

▶ **Corollary 8.** *In PoD-Game that allows lazy and diligent collusion, if whistleblower contract exists, there is a unique equilibrium that all watchtowers are independently diligent.*

Cryptoeconomic security and parameter selection

To provide a clear benchmark for evaluation and decision-making for different security needs, we discuss how to choose parameters that ensure both cryptoeconomic security and compatible incentives.

First, we normalize the execution transaction costs to $c_T = 1$. As an example, assume there are $n = 10$ watchtowers with equal stakes; then $\phi(\alpha_i) = \phi(\alpha_0) \simeq 0.2$ (Eq. 1) when $\theta = 0.9$. Eq. 2 then gives the bound on rewards: $R_B > 5$ and $R_C > 1$, which are affordable as the normal-path incentives the rollup operator needs to provide. Eq. 3 calculates the minimum stake $\alpha_0 \mathcal{S} > 100009$, assuming that $c_V = 100000 \gg c_T$.

To induce lazy collusion, a leader in LC-Game will set the stake $t > 18514$ according to Eq. 9. In LC-Game, for any $t \geq 0$, the condition in Theorem 5 always holds. To eliminate all collusion strategies, according to Eq. 10, the reward for the whistleblower should be set to $R_w > 120572$.

If we increase the number of watchtowers to $n = 100$, we see that the bounty R_B increases accordingly to at least 44, which is still low. The minimum stake required for each watchtower does not change significantly, and R_w decreases to 102281.

Under the rational adversary assumption, our protocol guarantees a unique pure strategy Nash equilibrium where all watchtowers are diligent. Beyond this, cryptoeconomic security requires that when an attack occurs, the cost of launching the attack exceeds the maximum profit. Therefore, when designing the actual parameters (e.g. n, $\mathcal{S}$) for a practical system, we can utilize signals on the inherent value of transactions [12] to adapt the security requirements of watchtowers. For example, considering the current average transaction fee on Ethereum is approximately \$3 and the average L2 batch size is around 200, we simplify the model by assuming all transactions are executed on L1 to resolve disputes, which incurs the $c_V \approx \$600$ and $c_T \approx \$0.006$. (Using other more complex dispute resolution methods can reduce this cost.) Then the normal path reward for watchtowers can be estimated as:

$$R_B > 5 \times \$0.006 = \$0.03 \text{ per batch.}$$

Next, we calculate the number of batches produced by some L2 chains per year. For example, Optimism processes approximately 400,000 transactions per day, translating to about 2,000 batches per day or 700,000 batches per year. Given the probability that a watchtower wins the bounty is 0.2 and the annual percentage yield (APY) from external investment vehicles is around 6%, the condition to incentivize stakers with the estimated return rate is:

$$\frac{(5-1) \times 0.006 \times 0.2 \times 700{,}000}{600 + \text{tx value}} > 6\%.$$

In other words, a watchtower would be willing to secure approximately \$56,000 worth of transactions. And if the application aims to incentivize watchtowers to secure higher value transactions, the rewards should be increased accordingly.

Additionally, the minimum stake that each watchtower needs to post should include the transaction value. Notably, n determines the normal operational overhead of our protocol, which does not directly determine the security. It can be chosen with a trade-off between stake decentralization and operational cost.

6 Implementation and Evaluation

6.1 System design

The protocol described so far is a general framework for proving diligence in a computing platform with anytrust guarantees. We describe an implementation of these proofs subsequently in the context of optimistic rollups (ORs) as the compute platform and Eigenlayer [14] as the underlying staking platform under a non-collusion setting. The complete system is termed a watchtower network and serves the following implementation goals:

1. *Low compute overhead*: Watching an OR state involves executing all transactions of the rollup. Overhead is termed as any resource cost on top of the bare minimum rollup state execution. Our implementation minimizes this overhead to lower a watchtower's resource costs.
2. *Modular implementation*: The rollup ecosystem has a lot of tech stacks for full nodes ranging from general OP-stack to specialized implementations for DeFi, such as LayerN. Our modular implementation can be used on any rollup stack with minimal modifications.
3. *Low gas fees:* Large gas fees on settlement layers such as Ethereum can make watching prohibitively expensive. Our implementation scales down L1 gas costs and makes it an adjustable feature for the rollup.

The implementation is split across the functional domains of the rollup, settlement, payment, and staking layers. For simplicity, we assume the settlement, payment, and staking layers sit on the same ledger. However, it can be easily expanded to independent networks if desired. Figure 3 shows the binding of the functional components with the two

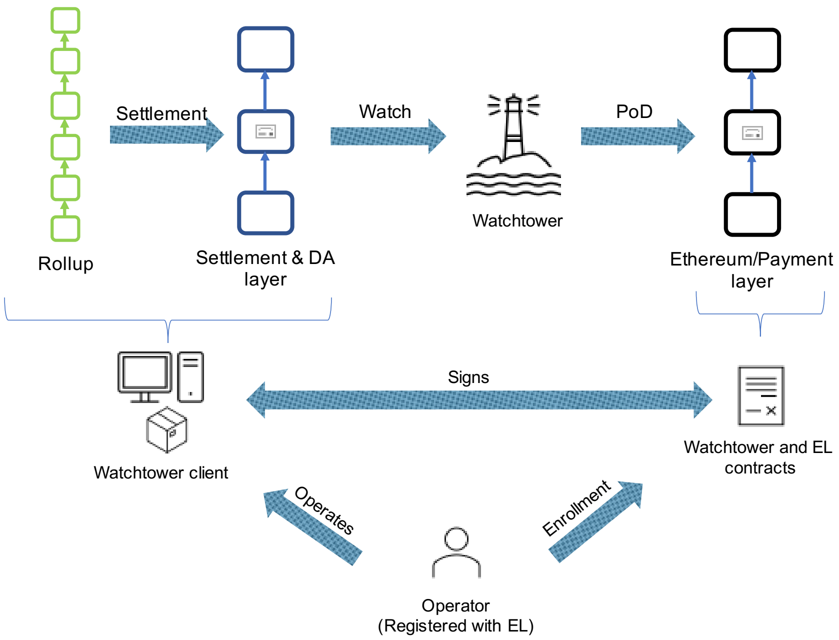

Figure 3 Watchtower client executes the rollup and observes the commitments on settlement layer, it posts bounty and flags on the payment/stake layer.

architectural components: the Watchtower client running on a server and smart contracts running on Ethereum. We outline the details of the two architectural components below. Our implementation draws from the modules in section 6.1 to build a watchtower on the Optimism Bedrock stack [50]. We test the implementation on the Ethereum Goerli testnet to watch OP-goerli and Base-goerli. We evaluate the system as per our quantitative implementation goals as described in section 6.1: *Compute overhead*, and *gas costs*. We adapt the modules to fit the existing rollup stacks and deploy them using an update-optimized architecture to make them evolve with the rollup ecosystem. We go over these details in Appendix A[1].

6.2 On-chain Contracts

The contracts are written in Solidity and deployed on Ethereum Goerli via a UUPS proxy architecture [37] to enable future updates. We deploy three contracts derived from section 6.1: *OperatorRegistry, BountyManager*, and *AlertManager*. The deployed contracts can be found at [4].

BountyManager contract implements a hash minimum across watchtowers to ensure a single bounty winner within an epoch. The payment pool and Rollup registry contract are replaced with a simple bounty count to measure contribution and a chain-id to point to a rollup. Dispute resolution contracts will utilize L2 fraud proofs once they evolve.

Optimizations. Immutable data like *diligenceProof* are stored as calldata variables to avoid storage costs. Proof outputs as set to a fixed size using *keccak256* to ensure consistency in storage requirements. Hash minima is calculated by the winning party to balance gas costs. We use *Mappings* instead of arrays to store hash data to reduce gas costs. We utilize audited

[1] Appendix is in the full version of the paper: `https://arxiv.org/abs/2402.07241`.

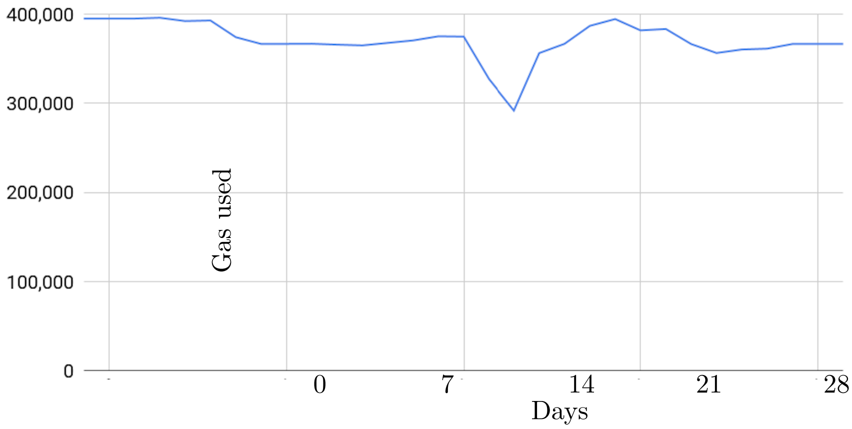

Figure 4 Gas usage of *MineBounty* operation over 4 weeks of deployment.

and optimized ECDSA OpenZeppelin libraries [36] to verify the authenticity of signed proofs to reduce contract risk. The contracts emit *NewBountyClaimed* and *NewBountyRewarded* for efficient notifications to off-chain clients.

Gas usage. A bounty mining transaction consumes 380K gas on average as shown in figure 4. As a reference, this is of the same order as a token swap operation on Uniswap. Two significant contributors to gas usage are (a) Proof storage for reward reimbursement and (b) Verifying the authenticity of proofs. This gas fees can be reduced in the future by introducing an appropriate aggregation layer for submitting bounty mining transactions.

6.3 Off-chain Client

The off-chain clients are implemented to support OP-Bedrock as a rollup execution engine. Two clients implemented in Golang enable the Watchtower client. A bounty mining client contains the bounty mining and transaction generator modules. This client is supported by a State extraction client that contains the settlement layer module with an RPC connection to a full node. Alert management and reconfiguration manager modules are left as future work to be implemented once the rollup stack evolves. We describe the implementation details of the Go clients below:

Optimizations. We employ an event-based trigger to the PoD generation. The watchtower client listens to emitted events on the *L2OutputOracle* Contract to receive real-time data updates from the contract; this is much more efficient than polling mechanisms. Generating execution trace requires storing the state roots after each transaction; this operation involves re-execution and hence is limited to just the penultimate block in an L2 epoch to avoid re-executing the whole epoch.

Resource utilization. The L2 full node and watchtower client are implemented on different machines. We run the L2 full node on a machine with 4 cores and 16GB of RAM and the watchtower client on a machine with 2 cores and 4GB RAM. Table 3 lists the resource costs in running the watchtower client. We observe that the client consumes minimal resources natively and has a minimal resource usage overhead on the L2 node. The client has no I/O since its context is stored in memory.

Table 3 Watchtower client usage stats.

L2 node	**CPU(%)**	**Mem(MB)**	**I/0(KB/s)**	**NW(KB/s)**
Before mining	0.7-25	5306	700	10-150
During mining	0.7-25	5307	700	10-150
Watchtower client	**CPU(%)**	**Mem(MB)**	**I/0(KB/s)**	**NW(KB/s)**
Before mining	0.1	15	0	1
During mining	0.1 - 10	15	0	1

We observe CPU usage burst to 10% on the watchtower client when the rollup proposer on L1 output Oracle proposes a new block. We observe a similar burst of 25% on the L2 goerli node when op-node sends a block to op-geth to execute the transactions and update the state.

We can utilize this capacity for additional off-chain computing to redistribute some on-chain contract operations to an off-chain module. A proposed approach is to perform proof aggregation on-chain; this implementation is left for future work.

7 Discussion

Proof of Diligence ensures that watchtower network executes all transactions on the rollup diligently. Further improvements down the line will enhance security, enable generalized applications, and allow for efficient trade-offs between delay and stake. We describe such improvements below:

7.1 Enabling Cryptoeceonomically Secure Watchtower Applications

The design described so far ensures that watchtowers are independently verifying transactions on rollups. The verification results can be utilized for attesting to any event on the rollup. These attestations are cryptoeconomically secured by the watchtower's stake locked with EigenLayer. We summarize the design here:

- **Configurable execution event trace**: Applications can subscribe to the Watchtower network to get verifiable updates on their transactions' life cycle. The events emitted from these transactions will be added to the bounty to ensure the execution and can be challenged for cryptoeconomic security.
- **Application event tracing**: Watchtowers can trace the whole life cycle of transactions pertaining to an application, starting from being sequenced by the sequencer to being ordered on L1 to being asserted into the state. A different level of cryptoeconomic security will accompany each of these stages.
- **Dispute resolution and cryptoeconomic security**: Events pertaining to the subscribed application are attested by the watchtower and are bound to be included in the next bounty. As enforced by the proof of diligence, other watchtowers will ensure that these attestations are correct. If these attestations are exchanged in private, An application/agent consuming this attestation can contest its correctness in the future by showing a mismatch between the watchtower's attestation and the mined bounty.

Besides applications on the rollup, the incentivized watchtower network holds significant potential for broader applications in general verifiable computing. Our future work will explore how the Proof of Diligence protocol can be extended to various domains such as AI inference [6], cloud computing [13], and blockchain light client protocols [31]. For example, to monitor and verify AI inference tasks, each watchtower can independently re-compute the inference results from the provided input data and model parameters, raising an alert if discrepancies are found. A recent work [31] demonstrates the use of watchtowers as a monitoring service to secure proof of stake blockchain states for resource-limited light clients, ensuring any invalid states are detected. This approach can be extended to a wider range of blockchain applications where light clients are prevalent. Watchtowers provide a robust layer of security and accountability.

7.2 Enhancing Security

The current system design ensures that Proof of Diligence works under a static adversary that can form static collusion. Resistance against a stronger dynamic adversary requires assumptions on rational independence of watchtowers and the privacy of whistleblower contracts. These assumptions can be enforced through system design by enabling random rotation of watchtowers in the pool and ensuring the privacy of the whistleblower.

Watchtower rotation. The rotation of watchtowers across different rollups in the pool is essential for security against an adaptive adversary. Watchtowers can be periodically rotated in small batches across rollups in a random and staggered manner reminiscent of the cuckoo rule [5]. The rotation can be made more efficient by utilizing two techniques: (a) *utilizing modularity*: we are designing an efficient reconfiguration protocol for watchtowers rotating between rollup two rollups sharing similar modules - such as two rollups running the OP stack; (b) *stateless clients*: watchtower rotation through the reconfiguration manager can be made very efficient by removing the need to transfer state. The witness chain team is developing a stateless client architecture that removes the need to download state when reconfiguring to a new rollup.

Private Whistleblower contracts. The interactions of the whistleblower with the whistleblower contract are private to ensure that they can't be used within the collusion contract. We are designing a system to ensure these inputs stay private to the collusion contract by deploying a whistleblower contract upon request post the bounty mining period. This ensures that the address of the whistleblower contract is not static and cannot be referenced in the collusion contract. Alternate design solutions include privacy-enhancing contract structures such as Aleo.

References

1 The merge. https://ethereum.org/en/roadmap/merge, 2023. Accessed: 2023-02-04.

2 John Adler. Minimal viable merged consensus. https://ethresear.ch/t/minimal-viable-merged-consensus/5617, 2019. Accessed on Oct 17, 2023.

3 Mustafa Al-Bassam, Alberto Sonnino, and Vitalik Buterin. Fraud and data availability proofs: Maximising light client security and scaling blockchains with dishonest majorities. *arXiv preprint*, 2018. arXiv:1809.09044.

4 Anonymous. Proof of diligence contracts. https://goerli.etherscan.io/address/0x1BF313AADe1e1f76295943f40B558Eb13Db7aA99.

5 Baruch Awerbuch and Christian Scheideler. Towards a scalable and robust dht. In *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 318–327, 2006.

6 Suma Bhat, Canhui Chen, Zerui Cheng, Zhixuan Fang, Ashwin Hebbar, Sreeram Kannan, Ranvir Rana, Peiyao Sheng, Himanshu Tyagi, Pramod Viswanath, et al. Sakshi: Decentralized ai platforms. *arXiv preprint*, 2023. arXiv:2307.16562.

7 Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint*, 2017. arXiv:1710.09437.

8 Jing Chen and Silvio Micali. Algorand. *arXiv preprint*, 2016. arXiv:1607.01341.

9 CoinTelegraph. Ethereum upgrades: A beginner's guide to eth 2.0. https://cointelegraph.com/learn/ethereum-upgrades-a-beginners-guide-to-eth-2-0, 2020. Accessed: 2023-02-04.

10 George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and tusk: a dag-based mempool and efficient bft consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 34–50, 2022.

11 Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part II 37*, pages 66–98. Springer, 2018.

12 Soubhik Deb, Robert Raynor, and Sreeram Kannan. Stakesure: Proof of stake mechanisms with strong cryptoeconomic safety. *arXiv preprint*, 2024. arXiv:2401.05797.

13 Changyu Dong, Yilei Wang, Amjad Aldweesh, Patrick McCorry, and Aad Van Moorsel. Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 211–227, 2017.

14 EigenLabs. Eigenlayer, 2023. Accessed: 2024-01-27. URL: https://www.eigenlayer.xyz/.

15 Dankrad Feist. Proofs of custody. https://dankradfeist.de/ethereum/2021/09/30/proofs-of-custody.html, 2021. Accessed on Oct 17, 2023.

16 Ed Felten. Cheater checking: How attention challenges solve the verifier's dilemma. https://medium.com/offchainlabs/cheater-checking-how-attention-challenges-solve-the-verifiers-dilemma-681a92d9948e, 2019. Accessed on Oct 17, 2023.

17 Bingyong Guo, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. Dumbo: Faster asynchronous bft protocols. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 803–818, 2020.

18 Abdelatif Hafid, Abdelhakim Senhaji Hafid, and Mustapha Samih. Scaling blockchains: A comprehensive survey. *IEEE access*, 8:125244–125262, 2020.

19 Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S Matthew Weinberg, and Edward W Felten. Arbitrum: Scalable, private smart contracts. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1353–1370, 2018.

20 Julia Koch and Christian Reitwiessner. A predictable incentive mechanism for truebit. *arXiv preprint*, 2018. arXiv:1806.11476.

21 Georgios Konstantopoulos. How does optimism's rollup really work? https://research.paradigm.xyz/optimism, 2021. Accessed on Oct 17, 2023.

22 Caldera Lab. Caldera: The rollup platform. https://caldera.xyz/, 2023. Accessed on Oct 17, 2023.

23 Offchain Labs. Arbitrum rollup. https://arbitrum.io/rollup, 2018. Accessed on Oct 17, 2023.

24 Loi Luu, Jason Teutsch, Raghav Kulkarni, and Prateek Saxena. Demystifying incentives in the consensus computer. In *Proceedings of the 22Nd acm sigsac conference on computer and communications security*, pages 706–719, 2015.

25 Akaki Mamageishvili and Edward W Felten. Incentive schemes for rollup validators. In *The International Conference on Mathematical Research for Blockchain Economy*, pages 48–61. Springer, 2023.

26 Michael Maschler, Shmuel Zamir, and Eilon Solan. *Game theory*. Cambridge University Press, 2013.

27 Patrick McCorry, Chris Buckland, Bennet Yee, and Dawn Song. Sok: Validating bridges as a scaling solution for blockchains. *Cryptology ePrint Archive*, 2021.

28 Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

29 Juhi Mirza. Ethereum 2.0 transactions per second: Ethereum will reach 100,000 tps after upgrade, says vitalik buterin. *Gfinity Esports*, August 2022. URL: https://www.gfinityesports.com/cryptocurrency/ethereum-2-transactions-per-second/.

30 Gianmaria Del Monte, Diego Pennino, and Maurizio Pizzonia. Scaling blockchains without giving up decentralization and security: A solution to the blockchain scalability trilemma. In *Proceedings of the 3rd Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pages 71–76, 2020.

31 Niusha Moshrefi, Peiyao Sheng, Soubhik Deb, Sreeram Kannan, and Pramod Viswanath. Unconditionally safe light client. *arXiv preprint*, 2024. arXiv:2405.01459.

32 Shashank Motepalli, Luciano Freitas, and Benjamin Livshits. Sok: Decentralized sequencers for rollups. *arXiv preprint*, 2023. arXiv:2310.03616.

33 Mahmudun Nabi, Sepideh Avizheh, Muni Venkateswarlu Kumaramangalam, and Reihaneh Safavi-Naini. Game-theoretic analysis of an incentivized verifiable computation system. In *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, pages 50–66. Springer, 2020.

34 Kamilla Nazirkhanova, Joachim Neu, and David Tse. Information dispersal with provable retrievability for rollups. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, pages 180–197, 2022.

35 Akira Okada. The possibility of cooperation in an n-person prisoners' dilemma with institutional arrangements. *Public Choice*, 77(3):629–656, 1993.

36 OpenZeppelin. OpenZeppelin-contracts. URL: https://github.com/OpenZeppelin/openzeppelin-contracts.

37 Hadrien Croubois Santiago Palladino, Francisco Giordano. Erc-1967: Proxy storage slots. https://eips.ethereum.org/EIPS/eip-1967, April 2019.

38 Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*, pages 507–527. Springer, 2015.

39 Altlayer Team. Altlayer: A decentralized interlayer for rollups. https://altlayer.io/, 2023. Accessed on Oct 17, 2023.

40 Base Team. Base. https://base.org/, 2023. Accessed on Oct 17, 2023.

41 BNB Chain Team. opbnb: High-performance optimistic layer 2 solution for bnb smart chain. https://opbnb.bnbchain.org/en, 2023. Accessed on Oct 17, 2023.

42 Conduit Team. Conduit. https://conduit.xyz/, 2023. Accessed on Oct 17, 2023.

43 Eclipse Team. Eclipse. https://www.eclipse.builders/, 2023. Accessed on Oct 17, 2023.

44 LayerN Team. Layer n: Ethereum's financial superlayer. https://www.layern.com/, 2023. Accessed on Oct 17, 2023.

45 Linea Team. Linea. https://linea.build/, 2023. Accessed on Oct 17, 2023.

46 Optimism Team. Optimism. https://www.optimism.io/, 2020. Accessed on Oct 17, 2023.

47 Optimism Team. Optimism bedrock stack, 2023. Accessed: 2024-01-27. URL: https://community.optimism.io/docs/developers/bedrock/.

48 The Diem Team. Diembft v4: State machine replication in the diem blockchain. https://developers.diem.com/papers/diem-consensus-state-machine-replication-in-the-diem-blockchain/2021-08-17.pdf, 2021. Accessed on April 19, 2023.

49 Jason Teutsch and Christian Reitwießner. A scalable verification solution for blockchains. *arXiv preprint arXiv:1908.04756*, 2019.

50 The Optimism Collective. The Optimism Monorepo. URL: https://github.com/ethereum-optimism/optimism.

51 Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

52 Sage D. Young. Layer 2 network arbitrum surpasses ethereum in daily transactions. *CoinDesk*, February 2023. URL: https://www.coindesk.com/tech/2023/02/22/layer-2-network-arbitrum-surpasses-ethereum-in-daily-transactions/.

53 Mingchao Yu, Saeid Sahraei, Songze Li, Salman Avestimehr, Sreeram Kannan, and Pramod Viswanath. Coded merkle tree: Solving data availability attacks in blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 114–134. Springer, 2020.