GOAT Network Economic Beigepaper 2.0

Economic Design for a Bitcoin-Native zkRollup

1 Introduction and Motivation

Scaling blockchain systems often requires moving computation off-chain, but this introduces new economic and incentive challenges. In many rollup designs today, critical operational roles – such as sequencing transactions or proving state correctness – are concentrated in the hands of a few entities. For example, a single sequencer might monopolize transaction ordering and fees, and a single prover (often the rollup operator itself) bears the heavy cost of generating zero-knowledge proofs (zk-proofs) for each batch. This cost concentration can lead to centralization and unfair burden: only actors with significant resources can participate in high-cost roles, and they capture outsized rewards (in the case of sequencers) or incur outsized expenses (in the case of provers and data publishers). Such centralization undermines the trust-minimization goal of rollups and creates single points of failure (e.g. if the sole prover goes offline, the rollup halts).

In particular, despite recent technical progress, Bitcoin rollups face systemic economic design gaps:

- No foundational incentive model exists for critical protocol roles (sequencer, prover, publisher, challenger, etc.)
- A 2-week withdrawal (peg-out) latency significantly reduces capital efficiency and user experience.
- Insufficient incentive alignment and punitive mechanisms for challengers, thereby weakening the security of the dispute resolution process.

This paper seeks to address these challenges. **Avoiding cost concentration and improving decentralization** is a key motivation for our design. We seek to distribute both responsibilities and rewards across a broad set of participants. Prior work has noted that splitting duties can help prevent unnecessary centralization. The GOAT Network (our Bitcoin L2 project) emphasizes widespread participation of sequencers and provers via a PoS mechanism to ensure decentralization and censorship-resistance. In a nutshell, the GOAT network uses Proof-of-Stake (PoS) mechanism combined with BitVM2 fraud-proof technology to create a more balanced and fair system. In essence, we aim for a design where no single operator consistently bears all the costs or reaps all the rewards. Instead, through rotation and cross-subsidization of roles, every operator in the network takes turns performing each role thereby spreading out costs and benefits over time.

This white paper outlines how rotating roles and carefully crafted incentives can improve cost efficiency, fairness, and permissionless in a Bitcoin-native zero-knowledge rollup (zkRollup). We discuss why concentrating high-cost tasks like proving or data publishing in one entity is problematic, and how our rotation mechanism mitigates that. We also introduce a multi-round challenge mechanism (using 1-of-n honest assumptions) to reduce redundant work, wherein one randomly selected challenger in each round is responsible for the verification and challenge (if needed). The economic logic behind this selective challenge – including bonding, rewards, and the option to act or pass – is analyzed.

2 System Roles and Cost/Benefit Tradeoffs

Our rollup design involves several distinct roles, each with different responsibilities, costs, and incentives. Understanding these roles and their tradeoffs is crucial before we describe how to rotate and reward them fairly:

- **Sequencer:** The sequencer collects user transactions, orders them into a block (or batch), and typically submits the transaction data and updated state root to Layer 1. This role is low in computational cost (basically running a node and sorting transactions), but high in responsibility. The sequencer benefits by collecting transaction fees and potentially extracting MEV (Maximal Extractable Value) from ordering. In many rollups today, a single sequencer can earn substantial fees but also could censor transactions if malicious. In our PoS model, sequencers must put up a stake (bond) and can be slashed for malicious behavior, aligning their incentives with honest operation. The primary cost a sequencer faces is any Layer 1 fee for publishing data (if that duty isn't separated out), and the opportunity cost of their stake.
- **Prover (BitVM2 Operator):** The prover generates zk-proofs attesting to the correctness of the new state after executing a batch of transactions. This role carries a high computational cost – proving can require significant time and specialized hardware (e.g. generating a SNARK or STARK proof). The prover typically doesn't directly collect fees from users; instead, they might be compensated via a portion of transaction fees or protocol rewards for the service of proof generation. In a traditional zkRollup, every state update must come with a validity proof. If only one or a few parties can afford to run provers, it creates a chokepoint and centralization risk. The prover's benefit is any reward or subsidy for proof generation, but if the rewards are insufficient, the cost could outweigh the benefit, discouraging participation. We address this by sharing prover duties among many and cross-subsidizing them from sequencer revenue. Note that in the BitVM2 design, the prover's work is handled by a role called Operator. In addition, BitVM2 operators are responsible for exchanging PeggleBTC with users for native BTC and initiating reimbursement requests from the Committee.
- **Publisher (Data Availability Provider):** This role publishes the batch data and state commitments to the Layer 1 chain. In many implementations, the sequencer also handles data publication, but we distinguish it here to highlight the cost. Publishing transaction data (calldata) to L1 is often the single largest expense in rollup operation – it directly depends on L1 gas costs and the size of the batches. The publisher ensures data availability so that anyone (especially challengers) can reconstruct state and verify transactions. The cost for the publisher is the L1 gas fees paid for posting data; the benefit is usually a share of transaction fees to offset these costs. If one entity is always the publisher, they bear a heavy financial burden especially during times of high L1 fees. In our model, we assume this cost is covered by the sequencer of the block (who uses part of the collected fees to pay L1 costs). We allow the publishing role to be rotated or subsidized so that no single operator constantly pays out of pocket.
- **Challenger/Verifier:** The challenger plays a critical role in an optimistic challenge model. Any participant can act as a challenger by watching the rollup's state updates and submitting **a fraud proof** if they detect an invalid state transition. In our system, a challenger must run a full (or at least validating) node to recompute state roots and check proofs. The cost for challengers is the runtime and resources to verify each batch off-chain (executing the transactions or checking a proof). Typically, challengers are not paid unless they actually catch fraud – monitoring is a voluntary expense undertaken in order to keep the system honest or to earn a

reward in the rare case of fraud. When a fraudulent state is posted, a challenger can initiate a dispute; if the challenge is successful, the malicious actor's stake is slashed and awarded to the challenger as a reward. For example, we will slash a malicious sequencer's staked BTC and give a portion to the challenger who proved the fraud. In our design, challengers also have to put up a bond when challenging to prevent spam or frivolous disputes – a false challenge (if the state was actually valid) means the challenger loses their deposit. Thus, challengers are economically incentivized to monitor carefully and only challenge when they are certain an invalid state has been proposed Each of these roles has different cost/benefit profiles, and a naive design might allocate them to different dedicated entities. For instance, one could imagine a system with a set of sequencers (who profit), a separate set of provers (who incur costs and hopefully receive fees), and open participation of challengers (who only occasionally earn rewards). However, such separation can lead to misaligned incentives and centralization pressures. The prover and publisher roles, being cost-intensive, might end up performed only by the rollup project team or a few large players, while the sequencer role could be coveted by profit-seeking actors. This imbalance can cause the whole system to gravitate towards centralized control or to become unsustainable if cost-bearing roles aren't adequately compensated.

The key idea in our economic design is to unify these roles under a single umbrella of "universal operator" ("operator" in short) and have operators rotate through all roles. All operators are required to stake tokens in the L2, and they become eligible (and obligated) to perform any of the roles assigned to them in a given round. By doing so, we ensure that:

– No operator is only a sequencer (earning fees) or only a prover (paying costs) all the time. Over the long run, each operator will experience both the revenue-generating and cost-incurring duties.
– Incentives are balanced out: an operator knows that if this epoch they spend resources proving or publishing, the next epoch they are likely to get to sequence a block and earn fees (or vice versa). This cross-subsidization smooths out the profit/loss volatility of any single role.
– It lowers barriers to entry. Even if running a prover node is expensive, an operator can justify it because they are not doing it continuously – e.g. a smaller operator could participate knowing they only need to produce a proof periodically, not for every block. In the meantime, they earn from other roles. This enables more participants to join the network, improving decentralization.
– It provides redundancy and robustness: if one operator fails or goes offline, the system can assign the role to another in the next rotation. There is no permanent reliance on a specific single sequencer or prover.

In summary, our system transforms the various actors (sequencers, provers, challengers, etc.) into **a unified set of staked operators** who share all responsibilities. This is somewhat analogous to how in some decentralized protocols. Our model extends these ideas with an explicit rotation scheme to guarantee fairness and prevent specialization from re-introducing centralization.

3 Rotation Mechanism for Universal Operators

To implement the above idea, we introduce a rotation mechanism that assigns roles to universal operators on a rotating or random schedule. The rotation functions as a role assignment randomly drawn from a pool of N staked operators. Operators are randomly selected for different roles (if needed): one operator is the Sequencer (ordering transactions), another the Prover (generating a zk-proof if needed), another the Publisher/Committer (posting the state root to L1), and another the Challenger (verifying that state), and so on. These assignments are determined on L2 using stake-weighted random selection, without involving L1 transactions for each rotation. **The principle is that roles are rotated frequently to prevent any single operator from continuously holding a privileged or costly position.** We opt for the random one.

Rotation based on stake weight is one practical approach: if an operator has 5% of the total stake, over a long period they should produce about 5% of the blocks as sequencer, generate 5% of the proofs, etc. This is analogous to leader election in some delegated systems. In fact, other researchers have discussed stake-weighted rotation for sequencers; Polynya suggests that a set of sequencers could be elected and rotate between them every X period of time, perhaps in proportion to performance or stake. Similarly, in a BlockBeats analysis, it's noted that pledge (stake) weights can determine which sequencer gets to propose blocks, without needing a complex consensus among them. We extend this concept to all operator roles, not just sequencing.

Cross-Subsidization via Rotation: By rotating roles, we achieve a form of cross-subsidization. The profits from sequencing can be used to subsidize the costs of proving and publishing. Concretely, when an operator is the sequencer for a block, they collect user fees (and possibly block rewards or MEV) – this is income. In another block, the same operator might be the prover, who has to spend computing resources to generate a proof – this is an expense. Because every operator does both, the net reward over time can be balanced. Ideally, the fee structure is set such that over a full cycle (where an operator has been sequencer X times, prover Y times, etc.), an honest and efficient operator comes out with a reasonable profit. The protocol may top-up rewards for costly roles if needed (for instance, using a portion of a token issuance or an ecosystem fund to pay provers). Many rollup projects consider token incentives to cover infrastructure costs; for example. In our model, because the roles are unified, any such token-based subsidies effectively benefit all operators and thereby indirectly subsidize the expensive tasks. However, for sustainability we adjust the fees and the economic design on frequency to not rely on token to sustain the network.

Another benefit of rotation is **improved fairness and neutrality**. With equal-opportunity sequencing, users get a more neutral ordering service (reducing risk of censorship or preferential treatment). No single sequencer can consistently extract MEV from all blocks – they only do so on their turns, and competition in stake ensures they don't misbehave (a malicious sequencer would get slashed and lose their turn). This design aligns with credible neutrality: it's akin to having a diverse committee of block producers rather than one monopolist. In practice, even if sequencers are selected by stake, requiring them to rotate prevents an "extended monopoly" that could have externalities. Short rotation intervals (even every block) avoid long monopolization by one party at the cost of a bit more overhead (handover coordination), whereas longer rotations improve efficiency but increase centralization risk over that period. Our stance is to favor short rotations – potentially at every block or every few blocks – to minimize any single operator's dominance at any time.

Performance and reliability considerations: A potential challenge in a rotation scheme is that not all operators have equal capabilities. One prover might be slower than another. If a "slow" prover happens to be assigned, it could delay proving for that block. However, since our design doesn't require immediate proof per block (see the section on publishing frequency), a slower proof does not halt the chain immediately – it just means the final validation for that block lags. We can mitigate this by enforcing timeouts: if a prover hasn't produced a proof within a certain window, another operator might be allowed to take over (and the failing prover could be penalized for missing their turn). Similarly, if the assigned sequencer fails to post a block in time, the system can skip to the next sequencer in line (and slash the defaulter's stake or slash their chance to sequence next time). These measures ensure liveness despite rotation. Just to cite another idea that is in the literature, BlockBeats notes a related idea for provers: you could allow anyone to step in and submit a proof if the designated prover fails, but only the designated prover is entitled to the reward. This is a graceful fallback – others might step in altruistically to keep the system going (perhaps to secure their own funds or the network's state) even if they won't be paid, but the expectation is the scheduled prover will normally do the job to earn their reward. We incorporate a similar principle: the rotation assignment gives the exclusive right to earn the role's reward to a specific operator, but does not preclude others from stepping in if something goes wrong. This greatly reduces redundant work under normal conditions (since only the assigned prover works on the proof), but still protects against downtime.

To summarize, the rotation mechanism in our PoS zkRollup works as an algorithmic scheduler for roles, distributing work and rewards among all staked operators. This achieves decentralization by numbers (many participants share the workload) and by time-sharing (no single participant continuously controls any function). It addresses the problem of cost concentration by ensuring that the expensive tasks (like proving or paying L1 fees) are borne by everyone in turn, not shoved onto one entity permanently. By blending the roles, we prevent siloed incentives; each operator's self-interest (maximizing their long-term rewards) aligns with the network's interest (having all roles performed honestly). An operator who tries to game the system in one role (say, a sequencer including invalid transactions to gain MEV) will be caught and punished by others (when those others act as challengers), losing not just future sequencing opportunities but also their stake – a strong disincentive to misbehave.

4 Multi-round Challenge Mechanism

In a traditional optimistic rollup, any participant can act as a challenger and verify the state, and if they find an error, submit a fraud proof. This provides security under the assumption that at least one honest participant will catch any fraud (the classic "1-of-n honest assumption"). As long as one honest participant exists, fraud can be detected. However, this design presents a few issues.

- **Resource Wastage from Redundant Computations:** If 100 challengers validate a reimbursement and it turns out to be valid, the efforts of 99 challengers are redundant. If each challenger trusts only their own computation, it results in wasted computational resources, including:
  - The KickOff Phase: Checking the longest chain (including both L1 and L2) for validity;
  - The Disprove Phase: Verifying through public input and the operator's proof.
- **Insufficient Incentives Due to Lack of Rewards for Challenges:** In the absence of fraud, challengers are likely to spend a significant amount of time monitoring the challenge system without receiving any rewards. This lack of incentives could lead to honest nodes lacking motivation to participate in challenges, and in some cases, a scenario where no challengers step forward may arise.
- **Long Challenge Period:** A 2-week challenging period significantly reduces capital efficiency.

GOAT Network introduces a multi-round challenge mechanism to enhance BitVM2 design, meaning that challengers are chosen in a fixed sequence (schedule) rather than relying on anyone to jump in. In practice, this can be implemented as multiple rounds of random challenger assignment when a challenger is needed. For example, the system might select twelve distinct challengers in a day – two hours a challenger in sequence – each drawn at random from the operator pool.

- If the first challenger does not initiate a challenge (whether because the state is valid, or they failed to detect an invalid state, or they colluded), the opportunity passes to the second, and so on.
- The selected challengers must stake a small amount of BTC on Layer 2.
- If previously selected challengers did not kick off a challenge during their rounds, and someone else successfully challenges (i.e. catches a fraud) in the current round, the BTC staked by the malicious operator who submitted the invalid state, along with the BTC staked by any previously assigned challengers who failed to act, will be slashed and awarded to the successful challenger who proves the fraud.

This randomized round-robin of challengers greatly increases the scrutiny on each proof while still avoiding duplicate work. Essentially, instead of all N watchers verifying in parallel (wasting effort) or none verifying (verifier's dilemma), the protocol forces one verifier at a time, but gives multiple chances to catch a bad block before it's finalized. Importantly, the system remains permissionless — any operator can still initiate a challenge.

This mechanism **improves security by sequential redundancy** – if a malicious operator manages to corrupt or bribe one challenger, they must still contend with the next randomly-chosen challenger, and the next. The probability of twelve (or even three) independent randomly selected challengers all being dishonest or colluding is extremely low. Thus, the chance that a fraudulent state slips through three rounds is far smaller than in a single-round model. In effect, the design is enforcing multiple independent 1-of-n assumptions in series: e.g. if each challenger has a 95% chance of being honest, only three sequential checks yields ¿99.9% chance an honest verifier catches the fraud at or before the third round. This means practically, **fraud will be detected much faster**, long before a final resort of an open public challenge is needed.

Under this scheme, finality can be achieved sooner. In a standard optimistic rollup, one must wait about two weeks for absolute finality because any watcher could present a late fraud proof. In the enhanced BitVM2, by the time the scheduled challenger rounds are done (e.g. 1 day), one can be highly confident in the block's validity (given the multiple independent checks). The protocol could then declare economic finality after a few rounds, shortening the waiting period significantly. Only in the extremely unlikely event that all designated challengers failed to act would the system fall back to a full open challenge (allowing anyone to step in). This open challenge would serve as a last safety net, but it should almost never be needed in practice – it's there primarily to uphold the security model in worst-case scenarios, ensuring that even if the sequence of assigned challengers were all compromised (an extremely low probability), the system is still secure if any other honest participant exists. In normal operation, the bad actor would be caught by one of the sequential challengers before it ever reaches an open challenge. Thus, the "1-of-n honest" assumption is preserved and even strengthened: we still only require one honest party eventually, but by giving multiple vetted parties a turn to inspect, we virtually guarantee one finds any fraud early. This reinforces security without the costs of everyone verifying everything.

To illustrate, consider a block with an invalid state root. In the old model, the block proposer might hope no one notices for 14 days; challengers have no assigned duty, so each might assume someone else will handle it, risking a missed fraud. In our model, Challenger A is assigned in the first 2-hour window – they will definitely check (or be slashed), so the fraud is likely caught immediately. Even if Challenger A were malicious or offline, in the second 2-hour window Challenger B (random new person) will check. It's highly improbable that a proposer can silence or corrupt both A and B (who were unpredictable choices). By the third 2-hour window, Challenger C checks, etc. By structuring the challenge phase as "random + random + random", the protocol minimizes the chance of undetected fraud and eliminates any rational reason to delay challenges. Each challenger only has a short window and knows someone else will do it if they don't, so there is no benefit in waiting – they must act or lose out. Griefing attacks (like challenging honest blocks just to delay finality) are curtailed as well, because a challenger who posts an unwarranted challenge loses their bond and the block simply proceeds after proving correctness.

In summary, the multi-round challenge mechanism means every zkp checkpoint is audited by multiple eyes in turn, not just assumed valid until timeout. It "focuses many eyes one at a time where needed", achieving the effect of constant vigilance without requiring all watchers to run in parallel. This clever scheduling improves both security and liveness: honest transactions finalize faster on average, and fraudulent ones are caught much sooner than under the optimistic challenge model.

5 Incentive Compatibility and Anti-Bribery Mechanisms

The enhanced BitVM2 economic model is designed to be incentive-compatible, meaning every participant's best interest is to follow the honest protocol rules. This is achieved through a carefully balanced combination of rewards for correct behavior and steep penalties for misbehavior or negligence.

As discussed, role rotation and cross-subsidization play a critical role in smoothing out the profit and loss volatility associated with any single role. By distributing responsibilities and earnings fairly across sequencers, provers, challengers, and publishers, the system promotes greater fairness, neutrality, and participation.

To enforce honest behavior, GOAT Network uses **Bonding and Slashing (Security Deposits):** Every operator must lock up a significant bond (stake) that can be slashed for misbehavior. The sequencer/publisher of a block implicitly puts their stake at risk when they post a new state – if that state is proven fraudulent, "slashing the full stake of malicious proposers" is carried out. Likewise, as discussed in the previous section, challengers post a smaller bond that can be slashed if they abuse the system.

The system aligns incentives such that the equilibrium behavior is honest. As described in the document, a challenger faced with the decision to challenge or not effectively has: if they challenge a bad state, they keep their bond and gain a bounty; if they don't challenge a bad state, eventually they lose their bond. Conversely, if they challenge a good state, they lose their bond; if they correctly do nothing on a good state, they keep their bond (and likely got a small fee reward). Therefore, **the best strategy is to only challenge when the state is invalid and to never challenge when it's valid.** This makes the protocol robust against rational adversaries.

Eliminating Bribery and Ransom Attacks: A noteworthy benefit of the committed role assignment and bonding is that it closes off opportunities for bribery or extortion that might exist in a naive design. In an open challenge system, a block proposer could attempt to bribe all potential challengers out-of-band ("I'll pay you to stay quiet") or a challenger could extort the proposer ("Pay me or I will challenge your block"). In BitVM2's enhanced model, these vectors are neutralized. Since the challenger is randomly picked and unknown until the block is published, a malicious proposer cannot know whom to bribe ahead of time. By the time they know the challenger's identity, that challenger is already locked into the protocol and has more to lose by taking a bribe. As the analysis explains, any "out-of-band bribe from the proposer" would be far smaller than the slashing risk the challenger faces. For example, a proposer might offer 1 BTC under the table, but the challenger would lose, say, a 5 BTC bond if caught colluding – not a good deal. Thus no rational challenger would accept such a bribe. Similarly, challengers cannot effectively ransom the system because if they falsely accuse a valid block to extort the proposer, the challenger simply gets slashed and dismissed by the contract for future roles – the proposer's state will be vindicated by a validity proof and the challenger loses their deposit. There's no mechanism for a challenger to hold a valid block hostage without suffering financially either through slashing or through the elimination of future important roles. Furthermore, **the protocol structure commits challengers to their role** (they don't get to opt out without penalty). If a challenger did nothing in hopes of extracting a later bribe ("I won't challenge if you pay me later"), they'd be slashed for failing their duty once the fraud is proven by someone else. In effect, challengers are pre-committed to either act honestly or be punished. This removes any bargaining power they might have had to demand payment outside the protocol. Through hefty slashing and automatic role assignment, **the system enforces honest behavior and makes dishonesty unrewarding**. As noted, "any potential bribe is far smaller than the slashing risk", so collusion doesn't pay.

In addition, because all operators are also periodically challengers, each knows that a culture of bribery would undermine the value of their own stake. The **committee of operators collectively has an interest in rejecting bribery**, since any successful bribe undermines security for everyone and could devalue the system. And with random selection, an attacker cannot systematically target all honest parties – there's always a chance the next challenger is an honest one who cannot be bribed or is outside the attacker's influence.

To summarize the incentive design: **honest behavior is strongly rewarded (fees, block rewards, fraud bounties)** while any deviation leads to a net loss (slashed stake or lost revenue). The strategy that "maximizes payoff" for each rational participant is to follow the protocol exactly. This creates a self-enforcing economic equilibrium supporting the protocol's security.

6 Comparison with the Standard BitVM2 Model

To appreciate the improvements, let's compare the **standard BitVM2 economic model** (or a generic optimistic rollup model) with the **enhanced model** described above:

Role Structure:

- **Standard:** BitVM2 primarily focuses on a trust-minimized bridge protocol between Bitcoin and any sidesystems, assuming the sidesystems are trusted. However, it is not a full-fledged rollup solution, offering no detailed design for rollup participants. Specifically, the challenger may not get its bonus as expected since the Bitcoin miner may apply a front-run attack on the challenger's Disprove transaction.

- **Enhanced:** GOAT Network introduces a complete Bitcoin-native zkRollup architecture with a well-defined role structure. It leverages a set of universal operators who rotate roles among sequencers, operators, challengers, etc., preventing any single actor from monopolizing rewards or being burdened with ongoing costs. This role rotation scheme ensures fairness, decentralization, and system resilience — if one actor fails, others seamlessly take over in the next round, avoiding bottlenecks or censorship risks. For the specific front-run attack, GOAT BitVM2 replaces the staker or rewards receiver to Committee, and executes the rewarding or slashing on GOAT Network between Committee and challengers.

Challenger Participation:

- Standard: Anyone can challenge during the 2-week window, but there's no **structured incentive to do so promptly**. This can lead to the verifier's dilemma – if everyone assumes someone else will verify, an invalid state might slip by or only be caught at the last minute. Additionally, many parties might duplicate verification for safety, which is wasteful.
- Enhanced: Only a **designated challenger** verifies each block, and they are incentivized with rewards and bound by slashing to do it reliably. There's no ambiguity about who should act – the protocol assigns responsibility. This eliminates the verifier's dilemma because someone is always incentivized to check every block, and it cuts out redundant work since others can trust the assigned challenger (backed by their stake). The chain is secure assuming just that the assigned challenger is honest each time (statistically ensured by random selection) rather than assuming an amorphous set of many watchers.

Challenge Process & Finality:

- **Standard:** Uses a lengthy **open challenge period** (e.g. 2 weeks) during which funds are locked and users must wait for finality. This delay exists to give any watcher time to notice and act. It significantly slows down user experience (withdrawals or inter-chain operations are delayed).
- **Enhanced:** By using **multi-round challenge mechanism**, the effective challenge period can be much shorter (e.g. less than 1 day) for high assurance. Honest blocks can be confirmed faster because once they pass the assigned checks, the probability of any undiscovered fraud is astronomically low. Thus, the perceived finality for users is faster – potentially accelerating withdrawals and reducing the capital lock-up time. If fast finality is required (e.g., for smaller withdrawals), the protocol can even dynamically shorten the challenge rounds, relying on the high likelihood an honest checker is present early. In short, users get their confirmations sooner, improving UX and capital efficiency. Only in rare cases of dispute would finality be delayed, and even then the dispute is resolved by an on-chain fraud proof that likely completes faster than a multi-day interactive protocol thanks to BitVM2's efficient proofs.

Security and Bribery Resistance:

- **Standard:** Relies on the game-theoretic assumption that someone honest will check and challenge, but if challengers are not properly incentivized, an attacker might exploit that (e.g., by attempting a fraud when defenders seem inactive, or by bribing multiple parties). There have been theoretical attacks where an attacker with enough capital could bribe or overwhelm honest verifiers if the incentive structure is weak.

– **Enhanced:** The security model has been tightened. There is always a bonded party on duty, meaning an attacker must contend with at least one honest staker every block. The **slashing conditions** and randomization make bribery or collusion extremely difficult and unprofitable. Dishonest majority scenarios are mitigated by the one-of-n assumption plus stake distribution – even if the majority of the committee failed, one honest challenger is enough. And if absolutely nobody honest was watching (very unlikely given incentives), the fallback of periodic zk-proofs would catch the issue eventually, though at that point large rollbacks might be needed. However, practically the combination of incentives means the network is always being actively defended by someone with something to lose. Compared to the previous model, which might have had latent watchers, this model has active, accountable watchers. It's like moving from a neighborhood watch to an assigned security guard on duty at all times – and if that guard fails, another takes their place immediately. This results in **faster fraud detection** and higher confidence in system integrity at any given time.

To encapsulate these differences, consider that the **previous BitVM2 economic model** faced challenges with "delay in the optimistic challenge, lack of incentive to challenge, high cost to challenge, long finality times" (as noted in the task description). The **enhanced model** fixes each of these:

– **Delay in optimistic challenge:** mitigated by splitting the challenge period into proactive rounds (no need to wait idly for 14 days; checks happen promptly each day or even per hour) and by potentially reducing the total challenge window.
– **Lack of incentive to challenge:** solved by assigning challengers explicitly and rewarding them for their service (no one is relying on altruism; challengers are paid and bonded to do the job).
– **High cost to challenge:** reduced since only one party does the work and they receive coverage for their costs (through fee share and bounty). Other parties don't incur cost, and the one who does is compensated. Also, using BitVM2 tech, the cost of proving fraud is manageable on-chain, so a challenger isn't deterred by exorbitant gas fees.
– **Long finality times:** improved by narrowing the challenge scope and leveraging optimistic assumptions for most blocks. In many cases, finality is reached as soon as the designated challenger signs off (which could be within one day) rather than waiting in weeks.

Finally, it's worth noting that the enhanced model is **practically more permissionless and fair**. All operators have equal opportunity to earn and equal responsibility to secure. It's a "**cooperative of stakers**" instead of a service run by one operator with external auditors. This not only improves robustness but also aligns with the ethos of trust-minimization. As the document concludes, by avoiding cost concentration and balancing incentives, the rollup can be "scalable, secure, and permissionless" – often called the blockchain trilemma – and BitVM2's economic design is crafted to achieve that.

7 Conclusion

In this paper, we presented a comprehensive economic design for a BTC-native Rollup that integrates BitVM2 and a multi-round challenge mechanism to achieve both high security and cost-efficiency. The core innovation is the **rotation and cross-subsidization of operator roles – transforming the typically siloed roles of sequencer, prover, publisher, and challenger into duties shared by all staked operators in a fair schedule.** This rotation ensures that no single operator is overburdened by costs (like proof generation or L1 fees) without also reaping rewards, and conversely, no operator can monopolize rewards without also contributing to the heavy lifting. By aligning the incentives across roles, the design prevents centralization and fosters a healthy, distributed set of operators, each economically motivated to keep the system running honestly.

We addressed the motivation to avoid cost concentration: high-cost roles like provers and publishers, if left to only a few, create central points of failure and economic inefficiency. Our design spreads these costs out. An operator might spend resources proving a batch today, but tomorrow they get to be sequencer and earn fees – over time the expenses and incomes balance. This **economic flywheel** keeps operators engaged and the network **sustainable**. In effect, the system self-subsidizes expensive operations by sharing the revenue from other operations, much like a cooperative.

We also introduced a novel multi-round challenge mechanism, where a randomly-chosen challenger in each round verifies each ZKP published in L1. This enhanced mechanism maintains the powerful "1-of-n honest" security premise of optimistic rollups – that the presence of a single honest verifier is enough to secure the system – but it **practically guarantees** that an honest verifier is found quickly by selecting challengers in sequence. It combines the scalability of optimistic approaches (rarely needing expensive proofs) with additional assurance layers and eventually the finality of zk-proofs at checkpoints. The result is a BitVM2 protocol that is economically robust: it incentivizes widespread participation (many operators can cooperate because they know they will all share costs and revenues fairly), it responds to fraud swiftly and with certainty, and it minimizes wasteful expenditure of resources.

In summary, the enhanced BitVM2 economic design creates a virtuous cycle (an "economic flywheel) where honest operators are continually rewarded, the network stays permissionless and well-audited, and users benefit from lower costs and faster finality. It represents a thoughtful synthesis of game theory and engineering – ensuring that security is not achieved at the cost of liveness or economy, but rather all aspects reinforce each other. This design can be visualized as a comparison between the old and new models, highlighting how the **Enhanced Model** introduces rotating duties, assigned challengers, and layered verification to dramatically improve upon the **Standard Model's** static roles and open challenge period (see Figure 1 below for a conceptual comparison). With these innovations, BitVM2 moves closer to the ideal of a scalable, trust-minimized rollup that inherits L1 security while operating with the efficiency of an L2 service.

References

1. GOAT BitVM2 White Paper (GOAT Network, 2025)
2. Discussion on sequencer decentralization and rotation (Polynya, 2023)
3. Rollup economics and roles (Delphi Digital, 2023; Chaisomsri, 2023)
4. Optimistic vs zkRollup cost considerations
5. Rotation-based prover mechanisms (Scroll, StarkNet plans)
6. Optimistic rollup security model (BitcoinCorleone, 2021)