Fraud Proofs Are Broken

… but we can fix them.

Optimistic rollups aim to inherit Ethereum's security through fraud proofs. The reasoning is straightforward: a single honest validator can prove all dishonest validators are liars, and thus enforce the truth. Since anyone can be a validator (including you), securing any optimistic rollup is just a matter of setting up a node to run on your laptop. No need to trust anyone.

The underlying assumption is that these fraud proof algorithms are permissionless. We know of three algorithms that aim for permissionless interactive fraud proofs. In no particular order: Cartesi's Permissionless Refereed Tournaments

(PRT), Arbitrum's Bounded Liquidity Delay

(BoLD), and OP Stack's Fault Proof System

(OPFP).

They all have issues. They're vulnerable, in different ways, toSybil attacks that undermine the safety, settlement speed and/or decentralization of the fraud proof system.

Our discussion focuses on the theoretical properties of the proposed algorithms, rather than the current development stages of any specific protocol that may or may not have training wheels. If the underlying algorithm isn't resistant to Sybil attacks, the training wheels can never be safely removed. Now, fraud proofs with a permissioned set of validators are certainly much better than no fraud proofs at all; it's quite acceptable as an intermediary stage. However, we need to keep researching better algorithms.

I'm a contributor in the Cartesi ecosystem, which provides an optimistic rollups solution. I'm part of the team developing the next iteration of our fraud proof protocol, mentioned above.

This post has three purposes:

- Establish a set of criteria to analyze fraud proofs.

Quoting Vitalik, the ecosystem's standards need to become stricter.

- Articulate why none of the candidates put forth so far meet these criteria.

Including ours.

- Call for open and public research collaboration between the optimistic rollups teams.

This is a personal frustration of mine. Our attempts at public communication with these teams haven't been successful at eliciting substantive academic discourse on the vulnerabilities and tradeoffs for existing fraud proof algorithms.

This post is not

about providing a solution to Sybil attacks in fraud proofs, nor is it about proposing a new algorithm. Furthermore, this post is concerned only with interactive

fraud proofs; non-interactive fraud proofs have different challenges and criteria.

Criteria

Optimistic rollups generally operate under two key assumptions: the base layer works (but can be arbitrarily censored for up to a week) and the presence of at least one honest validator. Under these assumptions, we need to design a protocol that allows anyone to participate while being resistant to Sybil attacks. Sybil attacks are the antagonist of this story, an ever-present adversary any permissionless protocol must face.

We propose three properties for analyzing permissionless interactive fraud proofs: safety, promptness and decentralization. These properties should be viewed as spectra rather than binary. Loosely, a permissionless algorithm is:

- Safe

if it can reject all false states under the original assumptions;

- Prompt

if settlement cannot be delayed substantially (or indefinitely);

- Decentralized

if validating doesn't require a lot of resources like hardware and funds.

In an ideal world, we want the honest validator to win any dispute in a timely manner using nothing more than a toaster.

We'll assume the reader is generally familiar with fraud proofs. We'll call the set of honest validators the hero

(worst case scenario is just one validator), and the set of dishonest validators working in tandem the adversary

. We'll use the terms validator

and player

interchangeably.

What's at stake

A dispute puts the total value locked (TVL) of a rollup at stake. An incorrect states that gets accepted can be extraordinarily profitable for the adversary. However, the same is not true for the hero – the hero gets nothing of the TVL by winning.

This financial imbalance puts the hero at an incredible disadvantage. At the limit, the adversary should be willing to burn a TVL's worth of resources. What's worse, different attackers may trustlessly coordinate through smart contracts and pool their resources.

We cannot assume the hero can match a TVL's worth of resources, not even when teaming up with others. We must assume the adversary has significantly more resources than the hero, given what's at stake. It's a lot harder to design good algorithms under these constraints.

Canetti et al.

Traditional fraud proof protocols are based on thework of Canetti. We'll highlight a few important concepts. The basic idea is quite simple.

Consider a setup with only two players. First, these players agree upon an initial state and a state-transition function (STF). They'll run the computation on their machine by successively applying the STF to the initial state, and propose a final state to the blockchain. These proposals are called claims

; they're a commitment to the final state of a computation. If the players agree, all is well. If not, the system starts a verification game. A verification game consists of two phases: the bisection phase

and the one-step proof phase

. The bisection phase is an interactive binary search over the whole computation, trying to find where the players first disagree. This disagreement is called the divergence

. Once the divergence is found, the blockchain applies the STF once (a step

), and eliminates the adversary.

Now consider a setup with many players. Naively extending the algorithm described above, there's an attack where the adversary posts the honest claim but lose the game on purpose by playing dishonestly. This makes evident the fragility of Canetti with respect to Sybils: it doesn't reveal lies, it reveals liars. We can't group players with the same claim into teams. Each player must represent themselves and only themselves for all parts of the dispute, even when there are others with the same claim. Players' signatures have to be part of the message.

Canetti proposes a few ways to make verification games with many players, taking this fragility into consideration. Unfortunately, none of those approaches scale well on the number of players. This makes the algorithm unsuited for permissionless fraud proofs since they'd be susceptible to Sybil attacks.

For example, the first fraud proof protocols of Cartesi and Arbitrum (based on Canetti) would fail the promptness property if made permissionless without further changes. Ed Felten wrote a post describing delay attacks, which are a type of Sybil attacks that compromises the promptness of the protocol. In short, there's an attack where settlement is delayed linearly on the number of Sybils, which can be a very long time indeed.

We need to design algorithms that can scale on the number of players, and thus be resistant to Sybil attacks. Otherwise, we won't be able to make protocols permissionless.

Permissionless Refereed Tournaments (PRT)

The PRT algorithm was developed by Cartesi. The paper can be foundhere. The algorithm is focused on application-specific rollups, which have different requirements when compared to shared chains. This has influenced the design of PRT: it's important that anyone can validate a rollup in a laptop with minimal stakes. (We'd argue this is also true for shared chains; ideally, the hero shouldn't need a supercomputer or a gazillion dollars.)

The key contribution of PRT is computation hashes

. It addresses the fragility of Canetti described above, allowing players to be grouped into teams. This enables a more efficient tournament structure.

Computation hashes

Claims in PRT are a stronger kind of commitment. Instead of a commitment just to a final state, claims are a commitment to the entire path of the computation (i.e.

a computation hash

). A computation hash is Merkle tree where each leaf is a hash representing the state at every state transition along the computation.

Bisections in PRT require a valid Merkle proof that the intermediary state is consistent with the computation hash. This way, an adversary that posts the honest computation hash cannot lose the game on purpose (except by inaction). As such, PRT can allow anyone to bisect any claim, since the Merkle proofs guarantee the bisection is honest. There's a fundamental shift: PRT can reveal lies, instead of just liars. Players' signatures need no longer be part of the message.

Therefore players can be grouped into teams and fight together. With this property, PRT creates a bracket-style tournament between claims, matching them pairwise and eliminating half the Sybils at each bracket level. This means the amount of work required of the hero is logarithmic in the number of Sybils

; PRT scales well on the number of players.

[

have you tried logarithms?

872×894 88.3 KB

](https://ethresear.ch/uploads/default/original/2X/1/1afc14e3db541a2c67ed0bf8b0221edcf44c96ba.jpeg)

To make a claim, players need to post a stake. After a claim is made, there's no need to post further stakes. Bisections are free, except for gas costs. Since the adversary needs to deposit one stake to create one Sybil, and half the Sybils are eliminated at each iteration, launching a Sybil attack is exponentially expensive.

The hero needs only fight one match at a time, which means they'll only ever need one computer. Furthermore, they need to deposit only a single stake. This means the decentralization property of PRT is great. Additionally, given the assumptions of optimistic rollups, the correct claim will always be enforced, which means PRT is safe.

On promptness, delay grows logarithmically on the number of Sybils, which is nice since logarithms grow very slowly. Unfortunately, this logarithmic delay has high constants: each single match still takes about a week to complete. The logarithm of Sybils multiplied by a week is quite slow. It's not fatally slow, but it's still quite slow.

At this point, PRT has reached a trilemma of safety, promptness and decentralization. A rollup can choose:

- promptness and safety

by increasing stakes (harming decentralization);

- promptness and decentralization

by reducing the maximum base layer censorship (harming safety);

- safety and decentralization

by making stakes small and keeping the maximum base layer censorship at seven days (harming promptness).

Ideally, we'd want to have the three properties at the same time, but for PRT something will have to give. We believe the second option is acceptable for chains that don't have a lot of TVL. But since the other algorithms were designed with high TVL in mind, for this comparison we'll pick the third option.

PRT

Safety

Promptness

Decentralization

Multi-stage disputes

The PRT paper describes two setups: single-stage and multi-stage. In practice, if the state-transition function is a single virtual machine (VM) step

, the single-stage setup is prohibitive for large computations. Generating a computation hash with every state transition in a computation is extremely slow. We introduced the multi-stage setup to address this issue.

The idea is to make the initial computation hash sparse, where the number of state transitions between each Merkle leaf is greater than one. This means we can't resolve the divergence between two sparse computation hashes. The blockchain would need to perform too many state-transition functions.

However, we can reduce the search space to the sparseness of that commitment. With a smaller search space, we have reduced the dispute to a smaller computation; we can recursively use the same method, but with a denser computation hash since the computation smaller. The base case is a fully dense computation hash that can be resolved normally. Players still need to post a stake at every nested tournament they wish to submit a claim, otherwise generating a Sybils would be cheap.

This technique is clever, but introduces delays. In a two-stage setup, there's a strategy where the adversary can delay settlement for the logarithm squared

of the number of Sybils, making it strictly worse than the single-stage setup.

The description and analysis of PRT above consider the faster single-stage setup. Although prohibitive if the state-transition function is a single virtual machine step, we can leverage ZK proofs and change the state-transition function from one VM step to several VM steps. This makes the single-stage setup possible in practice. Note that this is still an interactive fraud proof; it's just a change to its state-transition function.

Check Carsten Munk's presentation here about proving the execution of our virtual machine in ZK, using RISC Zero. Our benchmarks with low-end hardware are promising. This increases the hardware requirements from one laptop to one laptop with one GPU. Our ongoing implementation (called Dave) currently uses the multi-stage setup. We're moving to the single-stage setup because it has better promptness. We're also engaging in more research to create an algorithm that achieves all three properties simultaneously.

Bounded Liquidity Delay (BoLD)

In December 2022, Arbitrum announced their protocol was vulnerable to delay attacks in a permissionless setting. In the same post, they announced they had a solution, and that they'd post it soon. In August 2023, they released a preliminary paper, along with an implementation, naming it BoLD. The full paper is yet to be released. (One of the motivations of this text is also a call for teams to prioritize these discussions.) We look forward to the full release; we believe it will benefit the ecosystem as a whole.

BoLD was developed independently from PRT (PRT was published first), but both algorithms reached similar techniques. In particular, BoLD also uses computation hashes and the recursive disputes technique with sparse computation hashes. We've described these two techniques in the PRT section above. We highlight that players need to post a stake to enter a challenge in BoLD.

Unlike PRT, instead of setting up a tournament with brackets, in BoLD all claims fight each other simultaneously. Because of this and a novel clock technique, the settlement time is constant, regardless of the number of Sybils. The current implementation has a 16 day settlement time. As such, the promptness property of BoLD is great. But there are trade-offs.

The core issue of BoLD is safety, compromised by an attack we call proof of whale

. It's a kind of Sybil attack where a better funded adversary can win the dispute against the hero by exhausting the hero's resources until they have no more resources to keep playing the game.

Reiterating, winning a dispute is extraordinarily profitable for the adversary, but not for the hero. At the limit, the adversary should be willing to burn resources approaching the TVL of the rollup. It's reasonable to assume the attacker is significantly better funded than the hero.

[

poor seal

1024×1024 267 KB

](https://ethresear.ch/uploads/default/original/2X/2/2e390d880e5a7c2b330739c88e01e3550641a3a8.jpeg)

The hero needs to spend three different kinds of resource throughout a dispute: compute, blockspace, and stakes. Since each of these can be acquired with money, proof of whale

can be seen as exhausting the hero financially

. However, these resources are on different dimensions, and will be analyzed separately.

We'll describe three variants to proof of whale

, each exhausting a different resource.

Variant one: Zerg Rush the hero's servers

The first variant is an attack on the computational resources of the hero. The worst-case work required of honest parties is linear in the number of Sybils. Since the time is constant (one week) but the work is linear on the number of Sybils, hardware requirements grow linearly with the number of stakes the adversary is willing to burn.

This attack is hard to defend against, because the hero must be able to scale his computational power dynamically, or always dimension it with the worst case scenario in mind. Furthermore, the adversary needs no extra computers to pull this off since they can completely fabricate the states without turning on a VM.

Nevertheless, the adversary still has to burn one stake to create one Sybil. If the cost of stakes is irrelevant, then this attack is trivial to pull off. Increasing the stake to just match the cost of renting one laptop makes this attack as costly for the adversary as it is for the hero. In this case, if the adversary has more stakes than the hero has computers, then a wrong claim will win. Further increasing the stake increases the hero's advantage.

Increasing the stake as to dominate the cost of hardware does mitigate this attack. However, it's harder for players to participate when stakes are higher, compromising the decentralization of the algorithm.

Variant two: Zerg Rush the hero's access to blockspace

The second variant is an attack on the hero's access to blockspace. Blockspace can be acquired by expending funds (think paying ether for transaction costs), but it's not the only way. If a player is a block producer in the base layer (or has a generous friend who is), they will have blockspace access without expending funds for transaction costs. In any case, blockspace is not free; players have a limited amount of it. As such, instead of modeling access to blockspace as funds, it's better to think about blockspace as a total budget for transactions that each player consume along a dispute. This attack is not

about denying the hero's access to blockspace (i.e.

censorship), but forcing the hero to use up their budget.

All Sybils created by the adversary must be fought simultaneously by the hero. Each Sybil that eagerly participates in the bisection phase forces the hero to respond in kind. Playing the bisection phase to its completion, each eager Sybil forces both the adversary and the hero to include a number of transactions equal to the logarithm of the size of the computation. These transactions consume the budget of both adversary and hero.

If the adversary has more blockspace access than the hero, then the adversary might be able to exhaust the hero's blockspace access, depending on how many stakes the adversary is willing to burn. If the cost of stakes is irrelevant and the adversary has access to more blockspace than the hero, then a wrong claim will win.

Increasing the stake to just match the cost of blockspace access gives the hero a two-fold advantage; the adversary has to pay one additional stake plus blockspace (the equivalent of two stakes), whereas the hero has to pay only one additional blockspace (the equivalent of one stake). Further increasing the stake-to-blockspace ratio increases the hero's advantage, to the point where the advantage about equals this ratio.

Increasing the stake as to dominate the cost of transaction fees does mitigate this attack. However, it's harder for players to participate when stakes are higher, compromising the decentralization of the algorithm.

It's difficult to defend against this attack for several reasons. The adversary can use smart contracts to submit many claims and bisections at the same time, making their use of blockspace more efficient than the hero's. Furthermore, depending on the magnitude of the whale, fees may skyrocket as blockspace becomes scarce, reducing the hero's access to blockspace. Finally, if the adversary has the support of a block producer in the base layer, they can include transactions without having to pay (possibly skyrocketing) fees.

Variant three: Zerg Rush the hero's funds

If the stake is large, dominating both the cost of hardware and the cost of transaction fees, the adversary should use the third variant of proof of whale

. The third variant hinges on the multi-level challenge setup that we described on previous sections. BoLD is set to use three levels of nested challenges.

Reiterating, players need to post a stake at every challenge they wish to submit a claim, otherwise it would be cheap to generate Sybils. Every claim made at a non-leaf challenge may potentially spawn a sub-challenge. Take a two-stage setup: if the adversary launches a Sybil attack posting N

claims at the first challenge, BoLD will have to spawn N

second-level sub-challenges. This means the hero will have to submit N

claims in the second level, one claim for each spawned sub-challenge. Posting N

claims requires N

stakes. Therefore, if the adversary has more stakes than the hero, then the adversary will win.

Note that using the strategy described in the third variant also triggers the first and second variants. That is, proof of whale

forces the hero to have simultaneously more hardware and

more blockspace and

more stakes than the adversary has stakes.

Otherwise, the hero will lose the dispute. This makes the system behave not unlike majority token voting, though possibly biased towards the hero depending on the setup. We'll go into more details about the hero's advantage in the next section.

Proof of whale

is extremely hard to defend against because of the financial imbalance between hero and adversary. To make things worse, different attackers may trustlessly coordinate through smart contracts and pool their resources.

BoLD

Safety

Promptness

Decentralization

The hero's advantage

The proof of whale

issue was touched on this post at Arbitrum's research forum. In it, Arbitrum researchers mention a behavior of the protocol not described in the preliminary paper: BoLD has sub-challenges requiring a "mini-stake" instead of a full stake. We take their comment to mean the hero still needs stakes linear on the number of Sybils, but the constant significantly favors the hero. This constant is defined by the ratio between the stakes of each challenge level.

However, the protocol setup has three levels of challenge (according to the preliminary paper). If the mini-stakes of the two final levels are the same, then we're back to the original attack, shifted down one level: the adversary joins the root challenge once – paying a single full stake – and then joins the second challenge in earnest – paying one mini-stake for each Sybil. The hero must match the number of mini-stakes on the third level.

It seems the optimal strategy for BoLD is to chose the same ratio between levels one and two, and levels two and three. This ratio should be as high as possible to foil proof of whale

variant three. Furthermore, the smallest stake should to be high enough to foil proof of whale

variants one and two.

Let's try some numbers. Suppose we want the hero to have an ten-fold resource advantage over the adversary. As such, we should choose a ratio of 10

between the stakes of each level. Suppose $1,000

is just enough to cover for hardware costs and block access fees for any challenge. To keep the ten-fold advantage, we should set the smaller stake to about $1,000 * 10

, otherwise the proof of whale

variants one and two give the adversary a better advantage than the target of ten. This means the stakes for the top challenge should be set to ($1,000 * 10) * 10 * 10

, which is $1,000,000

. This makes the system quite centralized, since few would be able to participate.

Nevertheless, BoLD is still unsafe. Large shared chains have several billion dollars in TVL; that's what at stake here. If the adversary is willing to burn one billion dollars to win several billion, then the hero needs to have resources in the hundred millions (one tenth of one billion), sitting in the bank just to protect against whales. Ten is way too small of an advantage considering what's at stake.

However, if we increase the advantage, we significantly hurt decentralization. For example, if we set the advantage to one hundred (which may not even be safe enough), the first level stake becomes ($1,000 * 100) * 100 * 100

, which is $1,000,000,000

. We believe there's no choice of parameters that makes the system reasonable.

An improvement on BoLD

We suggest an improvement on BoLD: leveraging ZK proofs at the state-transition function to eliminate the use of sub-challenges, in the same way PRT is moving towards. This eliminates proof of whale

variant three. Nevertheless, ZK is a huge engineering effort that has trade-offs of its own that must be considered carefully. Let's call this improvement BoLD++.

BoLD++ settles in constant time like BoLD, so it has great promptness. Furthermore, the hero only ever needs one stake, addressing the third variant of the proof of whale

attack.

However, BoLD++ is still susceptible to variants one and two of the proof of whale

attack. Setting the stake high makes this attack prohibitively expensive, but unfortunately it compromises decentralization. In this setup, BoLD++ is quite centralized but not fatally so.

BoLD++

Safety

Promptness

Decentralization

OP Stack's Fault Proof System (OPFP)

The Optimism specs can be found here. In particular, the fault proof system is describedhere. The implementation is here. It has just been deployed to testnet.

In the traditional conception of fraud proofs, dishonest players can say the correct thing but lose on purpose. Both PRT and BoLD use computation hashes to address this. OPFP has a different approach. It allows claims to be bisected multiple times by anyone, introducing a directed acyclic graph (DAG) of claims in which anyone can add more vertices. Claims are added by bisecting an existing claim in the DAG.

Disputes in OPFP are comprised of many games that anyone can create and play. All games are played simultaneously, which makes the dispute finish in constant time. Each game has its own DAG, starting with a root claim that asserts the result of the rollup. There are two main ways honest validators will play these games:

- Protect the game with the correct root claim.

There's only one game with the correct root claim.

- Expose all other games as faulty.

A game is played by countering claims. But since claims are countered mainly by adding child claims to the DAG, we're entering the territory of countering counters all the way down to the one-step proof. Countering a counter uncounters the original claim. In the end, in order for a claim to be considered uncountered, all of its children must be countered. To

consider a claim countered, it's enough that a single one of its children is uncountered. The hero protects the correct game by countering the faulty claims that have countered the correct claims. The hero exposes faulty games by countering its root claims, and countering all the counters to their counters.

[

I bet the Optimism developers play mono blue in MtG

1667×1167 785 KB

](https://ethresear.ch/uploads/default/original/2X/d/d57ff7c4e33092f6f1019e4ac46c6ff36e583570.jpeg)

It's clever, though a tad convoluted. We recommend reading OPFP's specs, where this is explained in more details. A detail we should highlight is that posting a claim (through bisections or otherwise) requires posting a bond. This bond starts small, and grows exponentially as claims get closer to the leaf. The final value of this bond was set to cover the worst-case cost of a one-step proof.

Proof of Whale 2: Electric Boogaloo

OPFP, however, suffers from a similar proof of whale

attack as the original BoLD.

The attack is straightforward. The adversary creates a script that bisects the correct claim in earnest. This can be improved by creating a smart contract that bisects many times, amortizing the base cost of an Ethereum transaction. This can be further improved by choosing the bisections in a way that the hero has to always compute a new state (in the worst case, the adversary can force the hero to post the entire computation on-chain). From time to time, the adversary also creates a new game by posting a faulty root claim. The hero has to counter all moves of either kind if they want to win.

Since countering a claim requires compute, blockspace and bonds, proof of whale

forces the hero to have simultaneously more hardware and

more blockspace and

more bonds than the adversary has bonds.

Otherwise, the hero will lose the dispute. This makes the system behave not unlike majority token voting.

Unlike BoLD, we believe it's not possible to tune the parameters to give an advantage to the hero. The hero has no relevant advantage over the adversary; if the adversary is somewhat better funded than the hero, the adversary will win. It's reasonable to assume the attacker is significantly better funded than the hero given what's at stake.

Furthermore, we believe a winning adversary can recover both their bonds and the hero's bonds, since the adversary's claim will resolve correctly and the hero's incorrectly. This makes a proof of whale

even more profitable for the adversary.

OPFP

Safety

Promptness

Decentralization

Conclusion

Some algorithms perform better than others, but it's clear we're not in an ideal situation. Reiterating, the ecosystem's standards need to become stricter. Here's the tally:

Algorithm

Safety

Promptness

Decentralization

PRT

BoLD

BoLD++

OPFP

This is a call for collaboration. As a community, we need to fix fraud proofs. Furthermore, our individual implementations will be beneficial to all of us in a multi-prover future.

We have a couple of promising ideas that attempt to improve fraud proofs. We'll publish them soon as a public good to the ecosystem, and we hope it can start a conversation.

Let's nurture the Infinite Garden

together. It all starts with openness. The ecosystem can only benefit from collaboration.

Afterword

I've been supported by a remarkable group of people in the creation of this article.

I extend my deepest gratitude to:

- Augusto Teixeira and Diego Nehab, for their crucial collaboration in writing the article.
- Pedro Argento and Stephen Chen, my co-conspirators in developing the fraud proof system.
- Brandon Isaacson, Carsten Munk, Felipe Argento, Guilherme Dantas and Milton Jonathan, for their insightful feedback and thorough review.
- Donnoh and Willem Olding, for their gracious involvement in reviewing our work.

I also extend my gratitude toCartesi for funding the research and for providing the environment where the development could take place. We invite the reader to join us on our Discord, where we continuously engage in public research and debate these topics constantly.