Training Wheels

Software is known to have bugs. A decentralised network must have mechanisms to mitigate, respond and fix bugs – or it will struggle to gain adoption.

For L2's with bridged funds, this presents an especially difficult challenge – the source of truth for assets is L1. If a bug is exploited and the funds leave the L1 portal contract, the rollup has no claim on them.

This problem is compounded when upgrades occur via public social consensus – the bug must be disclosed publicly, before it can be patched.

This document proposes adding validating training wheels to the rollup, via a signature scheme to give greater confidence in the security of the network in the first 12 months, especially while the zk proving system is not deemed to be secure.

BLS Validating Nodes

Aztec is a ZK rollup, it uses a zero-knowledge proof to verify a state transition.

Bugs in the circuit logic or proving system are exploitable to produce a mismatch between the correct state transition and the proof being made – an attacker can supply a valid proof for an invalid state transition.

To mitigate the effects of bugs in the cryptography, the block transition function on L1, can be modified to also perform a BLS signature aggregation check from an honest majority of stakers. Effectively add a type of consensus that can only agree or disagree with the ZK proving system.

BLS attestations will catch bugs in the proving system and rollup circuits and public kernel, but will not catch bugs in the private kernel circuit – these run on a users device.

Stakers should compute the state transition for a block by processing each transaction in order to the protocol specification (later on by processing the ACIR code of the rollup / public transaction circuits). A diverse set of execution clients should exist to increase security here. NB these are not proving clients, just execution clients.

Stakers attest to their computed state transition via a signature. This signature uses a homomorphically additive signature scheme like BLS for aggregation. The sequencer will collect attestations and aggregate them for transmission with a block.

Additional checks are performed by the L1 rollup contract to validate a block:

1. A signature aggregation check using BLS. Stakers sign over the block transition.
2. A check that the resultant signature is from more than 50% of the staked signatures
3. Validation of the zk proof as normal

Estimated Gas Costs

Geometry defined a trick to reduce the gas cost of BLS verification to a fixed gas cost to verify a signature of ~16k gas + 150 gas per signer. This is achieved by pre-computing parts of the aggregated signing key to save computation at verification time. The aggregated key is pre-computed when stakers enter or exit the system.

Assuming additional EVM overhead to the quoted numbers, a conservative gas cost for an aggregated signature of 1000 signers is ~1,000,000 gas.

This cost can be subsidised by higher block rewards while the validating training wheels are active.

Prior to network launch, the foundation will declare the minimum staking thresholds required to activate the network for launch.

Example Staking Launch Criteria

Stake required

Main-net

= 32 ETH equivalent

Canary Network

= 5 Eth equivalent

Main-net

= 1000 validators

Canary network

= 100 validators

For main-net this means the cost of an attack will require ~$32M assuming an honest majority.

The additional checks imposed by BLS validating training wheels should only be active for a period of 1 year. Normal upgrades can optionally increase this time limit at the communities discretion.

Processing Valid Blocks

The following scenarios are possible:

1. If the staking network signature is invalid, the bock is invalid.
2. If the staking network signature is valid, but less than X% of the stake, the block is invalid.
3. It the staking network signature is valid, greater than X% of the stake, and the zk proof is invalid, the block is invalid.
4. It the staking network signature is valid, greater than X% of the stake, and the zk proof is valid, the block is valid.

If no valid block is produced for X hours, the network will enter emergency mode.

Emergency Mode

Whilst the rollup is in emergency mode the following measures are activated:

1. The validation threshold for signatures for a valid block is increased e.g from 50% to 75%.
2. An emergency upgrade path is enabled, detailed below.
3. Any active upgrade proposals can not proceed until the rollup exits emergency mode.

If block production resumes, the rollup will exit emergency mode, but the emergency upgrade path will remain active.

Emergency Upgrade Path Details

Emergency Upgrade Path

The fast upgrade path is baked into the L1 contracts and allows for:

1. Bypassing the normal upgrade path for critical bugs while emergency mode is active. Upgrades can change:
2. Rollup Logic
3. Verifier Address
4. Token Address
5. Rollup Logic
6. Verifier Address
7. Token Address
8. Automatic de-activation after 7 days hard coded
9. Pausing the rollup

Triggering a fast upgrade is governed by a faster consensus group than the slow upgrade path e.g:

- a multisig
- a security council with soulbound tokens
- a permissioned validator set
- tbc

Tradeoffs

Negatives

Forcing a majority of stakers to re-compute the state transition for each block, will harm network throughput and create redundant work.

Further research must be done on the effect this will have on throughput, it should not be a permanent fixture.

Positives

A bug in the rollup circuit or proving system can not be exploited, without acquiring a majority of the stake in this model.

Emergency mode can only be enabled by a bug not at will of something like a multisig, making it more resilient to:

- wrench attacks
- regulatory changes

Mitigation for bug types

Type

Code

Impact

Exploit Timescale

BLS Training Wheels Mitigation

Message Boxes

Solidity

Loss of L1 Funds

Hours

Rollup Solidity bug

Solidity

Total Loss of Funds + Liveness Failure

Hours

Kernel Circuit logic bug

Noir / Barrentenberg

Total Loss of Funds + Liveness Failure

1 - 30 Days

Verifier bug

Solidity

Total Loss of Funds + Liveness

1 - 30 Days

Rollup Circuit Bug

Barrentenburg / Noir

Total Loss of Funds + Liveness Failure

1 - 30 days

Public Kernel Circuit Bug

Barrentenburg / Noir

Loss of Funds + Liveness Failure

1 - 30 days