Optimism Blob Bug Fix Review

I conducted a targeted four-day review of a series of changes made in response to a bug in the Optimism fraud proof challenger library. The bug in question is related to the fact that the Optimism fraud proof system does not correctly use roots of unity. During this review, my primary focus was on the mathematics of the KZG commitments and the blob specification in order to ensure that the new code does not have any other issues or conflicts with blob implementations. In addition to this I have reviewed changes to the fraud program and the data flows in the Optimism codebase as well as the L1 smart contract preimage oracle. The review was done on a PR from a temporary repo responsive to security issue GHSA-r3cr-h2xh-wh4qs on Github.

Disclaimer: Security review is performed on a best effort basis and some bugs may remain. Moreover in this codebase only a very small part directly related to the change was reviewed.

About The Team

Aleph_v is a former mathematician and current cryptographer and security researcher with 7 years of experience performing security reviews and finding critical bugs for clients across the blockchain and cryptography industries.

Contact: Twitter - https://twitter.com/alpeh_v or Telegram - aleph_v

Summary of Findings

There were no bugs of note discovered in this review. The changes to the codebase were small and focused on a simple change to the functionality which replaces the incorrect polynomial evaluation points with roots of unity. As part of the review the following were checked in detail:

1. The mathematics of the blob commitments were checked in detail. Due to the specification the mathematics of blob commitments is restricted compared to general purpose KZG fewer things can go wrong. I reviewed the codebase and

consensus specifications to ensure no ambiguity in the construction of the roots or other issues with the polynomials.

2. I reviewed the changes to the go codebase in detail and those changes seem to be correct in the ways they create, validate and pass data between different parts of the application.
3. The onchain PreImageOracle.sol was reviewed for how it handles and validates blob data, the assembly seems to be correct, however it can load arbitrary data and requires the offchain fraud proof program to only access valid keys. I checked how the fraud proof loads data and found no instances where invalid blob commitments or non roots of unity could be used in preimage lookups.

Findings

Summary: Critical: 0, High: 0, Medium: 0, Low: 0

Findings are presented chronologically:

-

Notes

Very low severity, performance notes, or other non-issue comments

- The documentation for the changes is incorrect in that it describes the blob as a commitment to a polynomial which interpolates data at roots of unity of degree 4096 e.g.: $P(w_0) = a_0, P(w_1) = a_1$ ... $P(w_{4068}) = a_{4068}$, but in fact, blobs use the roots of unity in a bit reversed binary order (a fact which is not in the 4844 EIP but is actually only in the Ethereum consensus specs). However the code implements the bit reversal correctly.