Report

v. 1.0

Customer

Polyhedra

Cryptography Audit

Polyhedra Whitepaper

26th May 2023

Report prepared by

ABDK Consulting

Polyhedra report

Dmitry Khovratovich

May 2023

1 Introduction

We were asked to review the whitepaper of Polyhedra, which describes a proof of reserves protocol adapted for multiple assets. The main difference to existing PoR protocols is that all assets have a price; every user may have negative balance in some assets; it is additionally asserted that every user has a non-negative balance when all debts are taken into account with prices.

2 Protocol

2.1 Notation

- The negative balance (debt) in asset j (from 1 to N) for user i (from 1 to M) is denoted by non-negative variable $d_{i,j}$. The sum of all debts in asset j is denoted by $d_{\Sigma,j}$.
- The positive balance (equity) in asset j for user i is denoted by non-negative variable $e_{i,j}$. The sum of all equities in asset j is denoted by $e_{\Sigma,j}$. The debt and equity values are organized into a $M \times 2N$ matrix where rows correspond to users and columns to assets.
- The price of asset j is denoted by p_j.
- Commitments for debt and equity columns for each asset: $C_j^D = comm(\{d_{i,j}\}_{1\le i\le M})$, $C_j^E = comm(\{e_{i,j}\}_{1\le i\le M})$
- The Merkle tree whose leafs are hashes of individual balances $L_i = H(\{d_{i,j}, e_{i,j}\}_{1\le j\le N})$ is denoted by T and its root by R_T.

2.2 Statement

Throughout the whitepaper the full statement asserted by the protocol is never explicitly formulated. We have reconstructed it as follows.

The protocol asserts the following for user i:

1. The public inputs for the statement are
   - Summary balances $\{d_{\Sigma,j}, e_{\Sigma,j}\}_{1\le j\le N}$
   - Individual balance $\{d_{i,j}, e_{i,j}\}_{1\le j\le N}$.
   - Merkle tree root R_T.
   - Commitments for debt and equity columns for each asset: $\{C_j^D, C_j^E\}_{1\le j\le N}$.

2. The statement is

$$H(\{d_{i,j}, e_{i,j}\}_{1 \le j \le N}) \text{ is a leaf } i \text{ in a tree with root } R_T; \tag{1}$$

$$\forall j \; d_{i,j} \text{ is an } i\text{-th element in commitment } C_j^D; \tag{2}$$

$$\forall j \; e_{i,j} \text{ is an } i\text{-th element in commitment } C_j^E; \tag{3}$$

$$\forall i \sum_j p_j(e_{i,j} - d_{i,j}) = b_i \ge 0. \tag{4}$$

where b_i is the total *capital* of the user.

The protocol also asserts, for the sake of simplicity, that all the debt and equity values as well as the capital values are in certain range (l_j and l bits respectively).

3 Correctness

We have checked that an honest prover can convince a user of the statement, provided that it holds. The latter is non-trivial: a change in prices may make certain capitals negative, which makes it impossible to prove the statement.

4 Soundness

There is no formal proof of soundness of the protocol. Due to the time constraints, we have not attempted to make one. Regarding the statements we have formulated, the protocol seems to be sound. However, it is not certain that these statements are sufficient for the proof of reserves as a business goal.

5 Zero knowledge

There is no formal proof of zero knowledgeness of the protocol. Due to the time constraints, we have not attempted to make one. Regarding the statements we have formulated, the protocol seems to be zero knowledge.

5.1 Per user masking

Let $f(X)$ be the polynomial that encodes all users' balance for a given asset. The encoding is performance over the NTT domain $\mathbb{H}$, i.e., the i-th user's balance is $f(\omega^i)$ where ω is the root of unity over $\mathbb{H}$.

The masking polynomial is a zero polynomial at ω^i, constructed as $g_i(X) := (a_i X + b_i)(X - \omega^i)$ where a_i and b_i are sampled from the transcript. It is therefore crucial that the transcripts are distinct for various invocations of the protocol. The final polynomial whose commitment that user i receives is $\tilde{f}_i(X) := f(X) + g_i(X)$. The commitment and the corresponding opening can be derived respectively, due the the additive homomorphic property of KZG.

Overall, since ω^j is not a root for $g_i(X)$ for $i \neq j$ with overwhelming probability, $\tilde{f}_i(\omega^j) = f(\omega^j) + g_i(\omega^j) \neq f(\omega^j)$ for $i \neq j$. This ensures no information is extracted from $\tilde{f}_i(X)$ other than $f(\omega^i)$.

6 Issues

6.1 Major issues

1. There is no rigorous statement to be asserted by the protocol. Such a statement should explicitly describe public data, witness data, and the relation that is satisfied. For instance, the statements

listed in the whitepaper do not mention commitments.

2. As all balances are private, it is easy for Prover to create additional rows to the database and thus significantly shifts the summary balances it is going to prove. For example, let

   - Alice deposit 100 of token A of price 1;
   - Bob deposit 50 of token B of price 2;
   - Alice borrows 40 of token B using her token A deposit as a collateral.
   - Bob borrows 80 of token A using his token B as a collateral.

   Overall we have

   - Alice: $A : (D = 0, E = 100)$ $B : (D = 40, E = 40)$
   - Bob: $A : (D = 80, E = 80)$ $B : (D = 0, E = 50)$
   - Sum: $A : (D = 80, E = 180)$ $B : (D = 40, E = 90)$.

   We see that the Prover has to show publicly the reserves of 100 in token A and of 50 in token B. Now Prover inserts a fake user Charlie with debt 100 in A and equity 50 in B, which satisfies the balance condition:

   - Alice: $A : (D = 0, E = 100)$ $B : (D = 40, E = 40)$
   - Bob: $A : (D = 80, E = 80)$ $B : (D = 0, E = 50)$
   - Charlie: $A : (D = 100, E = 0)$ $B : (D = 0, E = 50)$
   - Sum: $A : (D = 180, E = 180)$ $B : (D = 40, E = 140)$.

   Now the Prover only has to show the reserves of 100 in token B.

   This may cause problems if token A is more volatile than token B and/or is less liquid. Therefore Prover can lie about the ratio of such assets that they hold.

3. (related to 2.) It is not guaranteed that Merkle tree are the same as witness rows: there may be leafs that do not correspond to valid witness values, and there can be witness data not being part of the tree.

6.2 Minor issues

1. In the protocol detailed description it is inconvenient to distinguish between positive and negative balances (debt and equity), as it effectively doubles the number of variables. Consider a uniform approach with the distinction between odd and even columns being made only in the solvency check.

2. The authors motivate partitioning the values into individual bits with the idea that the MSM reduces to simple addition. However, we note that it is also possible to precompute $[x]G$ for various small x and generator G and use those in the MSM. Therefore it is possible to split not into bits but into chunks, and verify that decomposition with some sort of a lookup argument.

3. There is a simpler way to check the sumcheck condition:

   - Commit to $w_{j,k}, d_j, h_j$;
   - Generate challenge v;

- Check that the following poly evaluates to 0 at v:

$$f'(X) = \sum_i \gamma^i \left(\sum_k 2^k w^D_{j,k}(X) - v_H(v) h^D_j(X) - v g^D_j(X) - d_{\Sigma,j}/n \right. \tag{5}$$

$$\left. + \gamma^N (\sum_k 2^k w^E_{j,k}(X) - v_H(v) h^E_j(X) - v g^E_j(X) - e_{\Sigma,j}/n) \right) \tag{6}$$

- Verifier can compute a commitment to f'_7 since they know v_H and can evaluate it at v.

4. The techniques listed in Challenge 3 (pages 3-4) are not well defined. Consider writing details rigorously.
5. A lot of variables are undefined: G, τ, n are examples. Consider doing a separate notation section.
6. It is unclear what is exactly asserted in Challenge 4, and what should be concealed from a user. Consider elaborating.
7. It is unclear if the large FFT domain technique (Challenge 5) is used eventually, or it is just mentioned.
8. The RLE technique (Challenge 6) makes sense, but it does not seem to be used. Consider clarifying.
9. In challenge 7, $f^D_j(X) + m^D_i(X)$ should be $f^D_j(X) + m^D_j(X)$.
10. In Prover step 2, "There are in total $(L + \ell - 1)$ polynomials" should be $(2L + \ell - 1)$.
11. In Prover step 3 it does not seem that you need both α and δ as just one of them would suffice.
12. In Prover step 3 it is unclear how $(M + 4)$ is derived.
13. In Prover step 4 "we squeeze a hash function H from the sponge" should be reformulated.
14. In Prover step 4 item 2, sample u_j, v_j, w_j and compute R_j.
15. In Prover step 6 the linear combination polynomial should be explicitly provided.
16. In Prover step 6 it should be x instead of β when polynomials are listed.
17. In Prover step 7 the linear combination polynomial should be explicitly provided.
18. In Prover step 7 it should be x instead of v when polynomials are listed.
19. The BN254 curve only deliver 110 bits security. Using a large SRS may also reduce security.
20. We recommend to re-randomized the external SRS via a small ceremony.

6.3 Alternative design

The protocol argues relations over 64 bits integers. It uses bit decomposition's to ensure the witnesses are in the right range; and use 254 bits field, BN254's scalar field, to store witnesses. Due to bit decomposition, the vast majority of the witnesses are in bits, and therefore allows for efficient KZG commitment.

The alternative design one may consider is

- use Goldilocks field, which is sufficient for all witnesses in this protocol;
- use FRI for commitment scheme;

- use Plookup (or other lookup arguments) to build range proofs.

Our brief analysis suggest this method improves the memory by a factor of ≈ 64: Goldilocks is $4\times$ smaller than BN254's scalar field; Plookup allows for, for instance, 16 bits table look up, and therefore reduces polynomial degree by $16\times$.

On the other hand, it is not clear how it effects run time. We can no longer use Lagrange KZG, nor FK22's full domain opening trick; however, overall the number of witness is reduced by $16\times$, and the field operations becomes $\approx 17\times$ faster.

An additional consideration is that if the proof needs to be verified on chain, rather than by the user, then a recursive proof needs to be applied to convert the proof to BN254 curve.