Zether and Zeto Compare

| Items | Zether | Zeto |
| --- | --- | --- |
| Introduction | Anonymous Zether is a private value-tracking system, in which an Ethereum smart contract maintains encrypted account balances. | a UTXO based privacy-preserving token toolkit using Zero Knowledge Proofs |
| Features | 1. Account Based 2. Sigma-bullet proof zkp 3. group for anonymous | 1. UTXO based 2. Zk-snark zkp 3. Nullifier for anonymous |
| Register Step | Need to register the account on Zether to use.  gas cost: 139,090 | No need to register. (have an optional kyc module). |
| Deposit Step | Store the amount encrypted in the Zether smart contract.  gas cost: 234,286 | Store the UTXO hash on the Zeto smart contract.  gas cost: 583,904 |
| Withdraw Step | Minus the amount encrypted in the Zether smart contract.  gas cost: 1,559,476 | Mark the UTXO hash status to be SPENT  gas cost: 1,290,544 |
| Transfer Step | Change the account amount based on the user input.  gas cost: 4,071,736 (group 4)  gas cost: 5,350,637 (group 8)  gas cost: 8,320,434 (group 16) | Change the old UTXO hash status and record the new UTXO hash.  gas cost: 1,768,599 ( round 1)  gas cost: 2,732,332 ( round 20)  gas cost: 3,310,311 (round 100) |
| Recovery Step | Read the account amount from the smart contract and try to decrypt the amount, A for loop with the max amount. So slow for large amounts in the system. | Need the client to record the UTXO in some backup storage and can’t recover from the chain. |
| drawback | 1. Epoch Limit, a user only can spend once every epoch time, limit the concurrent. 2. When a user spends, other users send money to him, which can make the user spend failure. 3. The big amount used in the system can slow the client. 4. A dead project not maintained for no new commit for two years. | 1. Complex UTXO management, UTXO merge, UTXO spend choose.   (After 9 merges of 10 UTXOs into one, the time consumed on the Arbitrum Sepolia testnet was 56,724 ms, with a gas usage of 37,568,943.)   1. Recovery UTXO needs some complex system design. |
| Concurrent | 1. A user spend: only one spend per epoch time 2. A user receives: any number of transfers. | 1. A user spends: the UTXO numbers he has. 2. A user receive: any number of transfers. |
| Support Chains | Ethereum (layer1)  Polygon (sidechain)  Arbitrum (layer2 - op-rollup) | Ethereum (layer1)  Polygon (sidechain)  Linea (layer2 - zk-rollup)  Arbitrum (layer2 - op-rollup) |
| Public Input of ZKP |  |  |

zeto Public inputs(The items are the **public inputs** for the circuit. These are inputs that will be visible and accessible in the verification process):

### 1. anon\_enc\_nullifier\_non\_repudiation.circom

Public Inputs:

* nullifiers: Used to verify the nullifier of the input UTXO to prevent double-spending.
* root: The root of the Merkle tree, used to verify the validity of the UTXO.
* enabled: Flags whether the current input UTXO is enabled.
* outputCommitments: Commitment values of the output UTXO.
* encryptionNonce: Random nonce used for encryption.

### 2. anon\_enc\_nullifier.circom

Public Inputs:

* nullifiers: Used to verify the nullifier of the input UTXO to prevent double-spending.
* root: The root of the Merkle tree, used to verify the validity of the UTXO.
* enabled: Flags whether the current input UTXO is enabled.
* outputCommitments: Commitment values of the output UTXO.
* encryptionNonce: Random nonce used for encryption.

### 3. anon\_enc.circom

Public Inputs:

* inputCommitments: Commitment values of the input UTXO.
* outputCommitments: Commitment values of the output UTXO.
* encryptionNonce: The random nonce used for encryption.

### 4. anon\_nullifier\_kyc.circom

Public Inputs:

* nullifiers: The nullifier of the input UTXO.
* utxosRoot: The root of the UTXO Merkle tree.
* enabled: Indicates whether the input UTXO is enabled.
* identitiesRoot: The root of the identity verification Merkle tree.
* outputCommitments: Commitment values of the output UTXO.

### 5. anon\_nullifier.circom

Public Inputs:

* nullifiers: The nullifier of the input UTXO.
* root: The root of the Merkle tree.
* enabled: Indicates whether the input UTXO is enabled.
* outputCommitments: Commitment values of the output UTXO.

### 6. anon.circom

Public Inputs:

* inputCommitments: Commitment values of the input UTXO.
* outputCommitments: Commitment values of the output UTXO.

### 7. check\_hashes\_value.circom

Public Inputs:

* outputCommitments: Hash commitment values of the outputs.
* outputValues: Values of the outputs.
* outputSalts: Salts of the outputs.
* outputOwnerPublicKeys: Public keys of the output owners.

### 8. check\_inputs\_outputs\_value.circom

Public Inputs:

* outputCommitments: Hash commitment values of the outputs.

### 9. check\_nullifier\_value.circom

Public Inputs:

* nullifiers: Nullifiers of the input.
* root: Merkle tree root.
* enabled: Indicator of whether the input is enabled.
* outputCommitments: Commitment values of the output.

### 10. check\_nullifiers.circom

Public Inputs:

* nullifiers: Nullifiers of the input.
* inputCommitments: Commitment values of the input.

zether zkp public inputs:

**All commitments (BA, BS, A, B, T\_1, T\_2, etc.)** ensure the verifier can check consistency without seeing the underlying values.

**The challenge (c) and response values (s\_\*)** allow the verifier to check that the prover correctly formed the proof based on the secret values.

**Inner product proof (ipProof)** ensures the prover has correctly applied the polynomial relations without exposing private data.

**BA**:

* This is a Pedersen commitment to the vectors aL (the binary decomposition of the value being proven) and aR (complement of aL).

**BS**:

* This is another Pedersen commitment, but this time to the vectors sL and sR, which are random vectors used to blind the original vectors aL and aR.

**A**:

* This is a Pedersen commitment to the intermediate vectors a and d, which are used in polynomial evaluations in the proof.

**B**:

* This commitment is to vectors b and c, and it also includes vectors e and f. These values are part of the recursive polynomial expansion.

**T\_1, T\_2**:

* These are Pedersen commitments to the intermediate coefficients t1 and t2 of the polynomial t(x) used in the range proof.

**CnG**

* the encrypted amounts

**C\_0G**

* commitments related to the original balances

**y\_0G**

* the public keys involved

**C\_XG**:

* the commitment to the difference in values

**tHat**:

* + **Definition**: tHat is the result of evaluating the polynomial t(x) at the challenge value x. It is the main term that connects the commitments to the underlying values.

**mu**:

* + **Definition**: mu is the blinding factor used in the final proof, which ensures that the randomness used in the commitments remains hidden.

**c**:

* + **Definition**: c is the challenge value, which is derived by hashing various commitments. This challenge forces the prover to construct the proof after the challenge has been issued, ensuring adaptive soundness.

**s\_sk, s\_r, s\_b, s\_tau**:

* + **Definition**: These are the response values corresponding to the prover’s secret key (s\_sk), randomness (s\_r), balance transfer (s\_b), and final blinding factor (s\_tau).

**Inner Product Proof (ipProof)**:

* + **Definition**: The inner product proof is a special proof that shows that the prover knows two vectors whose inner product is equal to a specific value. It is a compact and efficient way to verify range proofs and other statements without revealing the vectors themselves.