Tusima White Paper

Author: Tusima team

Date: 13 June 2023

Twitter: @TusimaNetwork

Summary

Blockchain technology has enabled significant progress in DeFi (DeFi). However, complete data transparency has led to asset theft incidents and compromised sensitive information. This hinders the growth of the industry. To address these issues, future DeFi must prioritize both privacy and auditability. This would allow fund institutions to prove that they only conduct authorized private investment transactions while enabling investors to carry out anonymous transactions with hidden amounts. Regulators can also audit user data without accessing original data while maintaining privacy and oversight.

Tusima prioritizes user privacy through the use of zero-knowledge proof and homomorphic encryption technology for identity and transaction value. Additionally, it utilizes rollup technology and DA acceleration scheme with off-chain TEE trusted hardware to achieve high TPS, making it an ideal choice for DeFi applications that require fast on-chain transactions.

Introduction

Financial transactions involve two types of privacy: anonymity and confidentiality. An anonymous donation to a non-profit organization means that the donor's identity will not be revealed, but the amount donated is known. However, when withdrawing money from a bank, your withdrawal amount remains confidential while your identity as an account holder can still be identified by those behind you in line.

Although the blockchain's public nature has fostered trust among all participants, it poses a challenge for scaling DeFi commercially. The transparency of all data on the chain makes it difficult to protect user data privacy. If this information were maliciously mined and exploited, it could seriously jeopardize user privacy.

Presently, the available privacy protection solutions for blockchain are primarily categorized into three groups: those based on mixed transactions, those based on L1 native chain architecture, and those based on cryptography.

- The privacy protection solution based on mixed transactions, such as Dash, blocks the connection between sender and receiver by mixing the transaction information of the participants. It only realizes the untraceability of assets, but does not hide key information of the sender, receiver and amount of the transaction.
- The privacy protection solution based on L1 native chain architecture, such as Oasis Network, Secret Network, Iron Fish, Manta Network, etc., starting from the blockchain architect, transforms the structure so that nodes in the blockchain could maintain different ledger information. It effectively avoids leakage of the user privacy data, but it is not compatible with the largest existing Ethereum ecology, and difficult to build in the ecology.
- The privacy protection solution based on cryptography uses cryptography technology to protect the privacy of transaction participants, among which the zero-knowledge proof algorithm provides the highest protection for transaction information. The more prominent cases include ZCash and Aztec. Although they can achieve complete privacy protection, they are not suitable for complex scenarios because of the UTXO model.

This paper proposes a new privacy protection solution called Tusima, which addresses the limitations of existing solutions. Tusima is based on an account model and uses zero-knowledge proof and homomorphic encryption technology to protect user identity and transaction amount in blockchain. The solution also includes DA acceleration through rollup expansion technology and off-chain TEE trusted hardware, allowing for tens of thousands of transactions per second (TPS). This makes it suitable for high TPS scenarios in decentralized financial applications.

The key innovations of this solution:

- A general privacy protocol based on Account Model
- The bottom layer is innovatively developed based on the ZK-Rollup framework technology, which has the same level of security as Ethereum and other Layer1 blockchains and is compatible with the EVM ecology.
- Based on L2 technology, compared with the L1 application solution (e.g. Tornado cash), it has the characteristics of extremely low handling fees, higher performance, and more convenient cross-chain ecology.
- Utilizing technologies such as recursive proof and offline accelerated computing, especially the TEE-based DA hardware acceleration function, the Tusima protocol can achieve the privacy proof of millisecond-level computing, and the TPS can reach more than 10,000.

- Tusima's DA offline privacy computing hardware acceleration solution adopts the confidential computing of TEE's trusted hardware acceleration solution, and the speed is increased by 10 to 15 times. It has been tested and verified by Huawei and has gained the international EAl4+ security certification.
- Tusima privacy transactions adopt a double authentication of homomorphic encryption and range proof, which realizes independent and controllable access to user privacy data, and also facilitates the auditing needs of regulatory authorities providing an on-chain privacy supervision window.

Background

ZK rollup

Rollup1 is a layer-2 scaling solution for Ethereum that proposes processing transactions off-chain in batches and verifying the results on-chain. ZK-Rollup2, on the other hand, is a Rollup that utilizes zkSnarks technology to achieve scalability. With zkSnarks3, instead of transmitting a user's signature data on-chain, we transmit proof that Multisig verification and other transaction verification checks have been correctly performed off-chain.

ZK Proof

zkSnarks use polynomials to create an arithmetic circuit from pending proof problems. This circuit is then converted into a Rank-1 Constraint System (R1CS)4 that zkSnarks can verify. The circuit has inputs and outputs, with public and private inputs. Private inputs are not visible externally. After executing the circuit, a proof is generated that proves the validity of the calculation process from input to output. By treating the circuit as an equation and using its solution as a private input, we can prove our ability to solve it without revealing the result. This feature of zkSnark is useful in Rollup for verifying off-chain calculations' validity.

Rollup in Tusima uses a Merkle tree to manage the balance information of all users. The root of this tree represents the state of Rollup, which we keep in the smart contract on chain. To update this state, a Snark Proof must be submitted that proves a valid off-chain state transition.

In Tusima's Rollup, we use a snark circuit to process a batch of off-chain transactions ($tx1, tx2 ... txn$). We provide the old Merkle root ($root1$) as public input to the snark circuit and the transactions as private input. The circuit outputs the new Merkle tree root ($root2$) and the corresponding Proof. This proof verifies that the transformation from $root1$ to $root2$ is correct, namely:

$$root1 \ + \ f(tx1, \ tx2 \ldots txn) \ = \ root2 \tag{1}$$

Finally, we submit these parameters to the smart contract on chain.

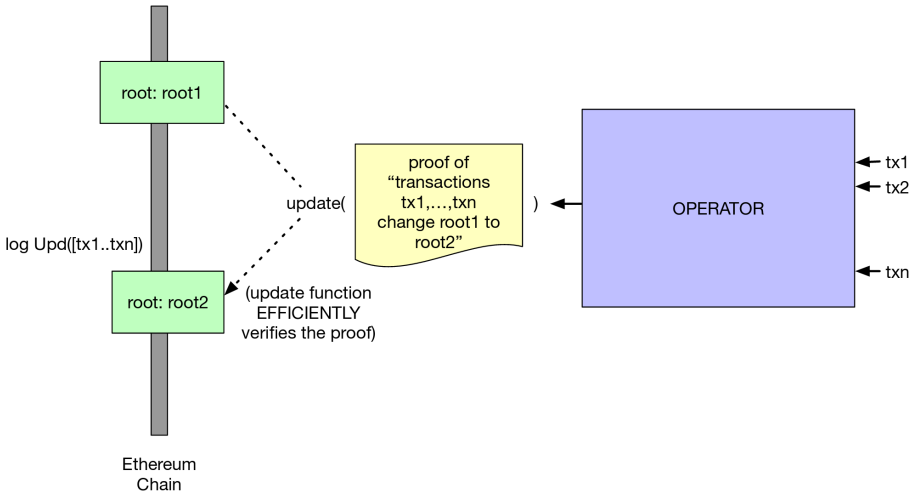

Figure 1: 7d4da5ee9980537e.png

- *root*1
- *root*2
- Proof
- Exclude signed and compressed transaction data (used for solving data availability problems)

In Tusima smart contract in L1 uses Snark-generated Verifier program to verify the Proof. Upon successful verification, the new Merkle root is deemed correct and replaces the old one.

Address Obfuscation

The Fisher-Yates shuffle algorithm mimics real shuffling by generating a random permutation of a finite set, such as an array. Each possible permutation is equally likely to occur, and the algorithm is highly efficient. The steps for calculating this algorithm are as follows:

1. Write down the numbers from 1 to N
2. Take a random number k from 1 to the remaining numbers (including this number)
3. Starting from the low bit, get the k-th number (this number has not been taken out), write it in the last bit of a separate list
4. Repeat step 2 until all numbers are taken out
5. The sequence written in step 3 is now a random arrangement of the original numbers

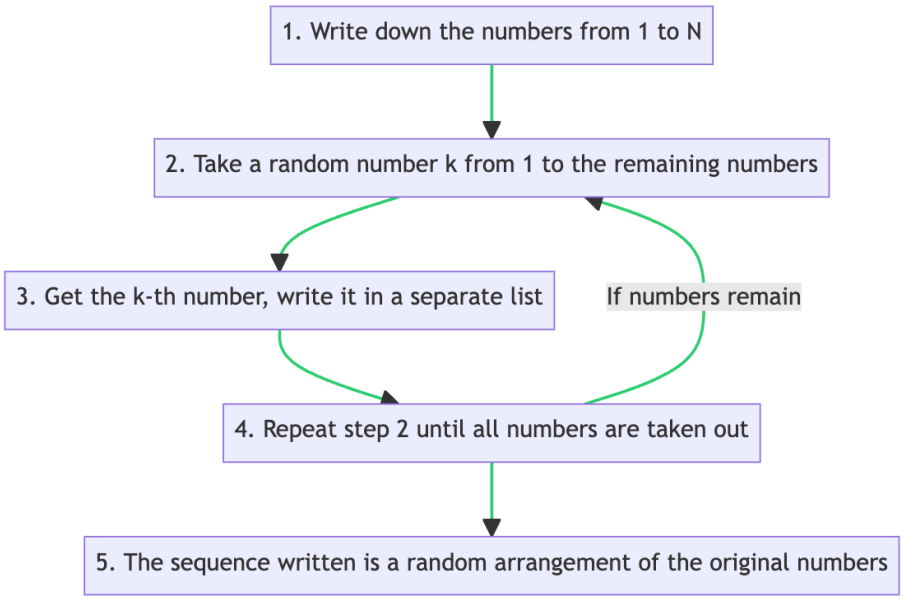

Figure 2: address.png

Range Proof

The account model requires the use of Range Proof as a cryptographic tool to verify the legitimacy of Confidential Transactions (CT). The Bulletproof Protocol5, proposed by Dr. Benedikt Bünz who also proposed the Zether protocol, is a popular algorithm for Range Proof. One advantage of Bulletproof is that it does not require a setup process. However, it relies on Inner-product Argument which is not compatible with circuit-based zero-knowledge proof algorithms. To reduce ECC related operations in our required range proof, we should consider using Sigma Protocol instead.

Encryption System

We will use Twisted Elgamal, which is an ECC-based public key cryptosystem, to encrypt the account balance. To enable homomorphic addition, we will place the plaintext on the index.

For the message $m $ in the p space, the ciphertext under pk, the encryption algorithm chooses $ r _p$, and calculates $C = (X = pkr, Y = grhm)$.

The Twisted Elgamal algorithm is as secure and efficient as the standard Elgamal algorithm. However, it has an additional attractive property of homomorphic addition. This means that for any value under the public key pk, you can perform homomorphic addition as follows.

$$(m_1, r_1), (m_2, r_2) \in M \times R$$
$$\mathbf{Enc}(pk, m_1; r_1) + \mathbf{Enc}(pk, m_2, r_2) = \mathbf{Enc}(pk, m_1 + m_2; r_1 + r_2)$$

Twisted Elgamal is friendly to zero-knowledge proofs because observing the r value in the encrypted space constitutes a promise.

Sigma Protocol

The Sigma protocol6 is utilized to demonstrate knowledge of values in a specific relation. This abstract protocol enables the satisfaction of a given relation through interactive zero-knowledge verification between P and V. Bulletproof, Groth16, and Plonk are all derived from this interactive Sigma protocol.

Tusima Architecture

Common solutions for private transactions include ZCash7, Tornado Cash8, Monero, Dash, etc. All of these except for Tornado Cash are based on the UTXO model. ZCash and Tornado Cash use zkSnarks for privacy while the latter uses currency mixing principles. On the other hand, Monero and Dash utilize ring signature technology.

Privacy protocol based on account model

Tusima is different from other projects in several ways. Firstly, it uses an account model instead of the UTXO model and supports general-purpose EVM (zkEVM) to ensure compatibility with Ethereum. Secondly, Tusima employs the Elgamal asymmetric homomorphic encryption algorithm for privacy protection, allowing users to decrypt their own transaction data while keeping others' data hidden. Thirdly, it utilizes an address obfuscation algorithm to generate a set of transaction addresses that conceal real addresses and participating accounts. Finally, Tusima uses Range Proof to verify calculation accuracy with nodes.

The main features of Tusima project can be summarized as follows.

- Tusima uses an account model instead of the UTXO model. ZK rollup is an expansion technology of Ethereum that supports transfers between users and general-purpose EVM, also known as zkEVM. Since Ethereum's EVM operates on an account model, Tusima follows suit to ensure full compatibility with Ethereum. This decision increases the potential for adding a zkEVM solution in the future.
- Tusima achieves privacy through the use of an asymmetric homomorphic encryption algorithm called Elgamal. This algorithm is based on a public-private pair and implements homomorphic addition, which Tusima uses to encrypt and calculate user balances. The asymmetry of this algorithm

allows users to decrypt their own transaction data while preventing them from accessing private data.

- Tusima generates a transaction address set using an address obfuscation algorithm. This hides the real addresses and the participating accounts in the transaction.
- Tusima employs Range Proof to demonstrate the accuracy of its calculations to the nodes.

Protocol Security

Tusima utilizes ZK-Rollup as its underlying architecture and employs homomorphic encryption, range proof technology, and address obfuscation to ensure transaction amount and address privacy.

Tusima uses ZK technology twice to ensure security. The first ZK, called range proof, proves that the total balance change in the set is 0. This means users prove to the rollup L2 node that they have not done anything illegal. The second ZK, called zkSnark, verifies the user's signature and range proof before calculating the new Merkle tree root in its circuit program.

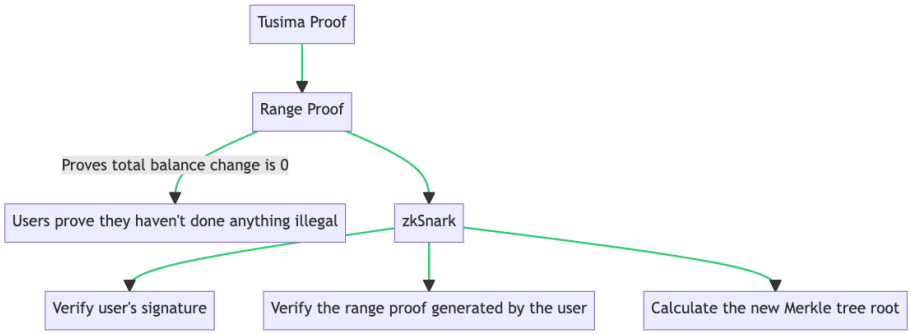

Figure 3: Tusima Protocol Security

The Rollup L2 node creates a Proof using a circuit program and sends it, along with the updated Merkle tree root, to the Ethereum L1 for verification. If the verification is successful, it proves that neither the user nor the node acted maliciously.

Efficiency

Tusima has improved its operating efficiency by implementing two measures: parallel computing and off-chain DA (Data Availability) with TEE trusted execution environment.

Parallel Computing

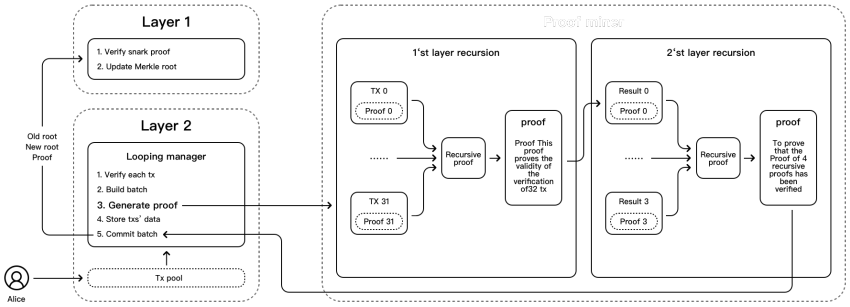

Figure 4: tusimaproofminer.png

Tusima is a solution that heavily relies on ZK algorithms, and the efficiency of these algorithms is just as important as their functional design. Tusima aims to solve algorithmic efficiency issues from two angles.

Firstly, by using parallel computing, the Rollup circuit will be split into two stages: The first stage will handle private transactions and its proof to obtain the output of the first stage Rollup circuit. The second stage will then process multiple parallel first-level Rollup circuits' proof outputs to get the final external circuit's proof output. This output will be submitted to the first layer for verification and updating by the Rollup contract of that layer. Both Plonk and Halo2 support recursive proofs; however, Plonk requires an updatable setup process based on Kate commitment while Halo2 eliminates this need with its inner production argument-based approach.

Secondly, Tusima plans to outsource ZKP proofs but faces information leakage challenges in doing so. Therefore, generating efficient proofs while protecting original witnesses are top priorities for Tusima's outsourcing solution. To achieve this goal, Tusima uses an MPC scheme-based privacy proxy protocol supporting PIOP framework-based proof systems and polynomial commitment schemes using KZG or inner product multiplication.

Finally, Tusima's economic model adopts ideas similar to POW promoting common growth between projects and miners alongside designing its proof outsourcing solution.

Off-chain DA and TEE Trusted Execution Environment

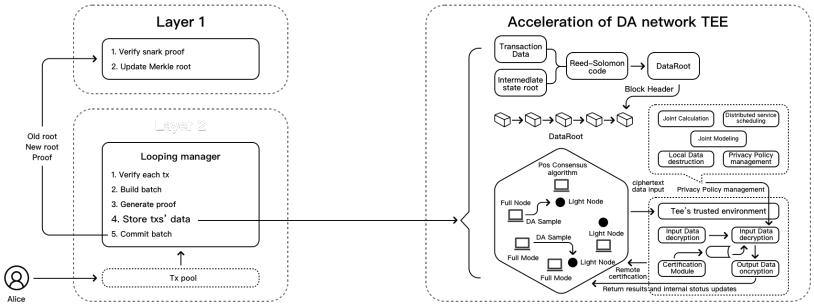

Figure 5: tusimada.png

Tusima has adopted off-chain DA to enhance performance. This involves transferring transaction data from on-chain to off-chain, which saves resources that can be used to improve the operation efficiency of L2. Tusima designed a DA network based on TEE trusted execution environment, which greatly improves efficiency without compromising security. The nodes of Tusima's DA network run on hardware with improved security and computing performance. This provides Rollup with more module options and 10000+ TPS performance. Additionally, it uses a dedicated layer for processing DA after modular decoupling, reducing computing power and improving throughput while skipping consensus bottlenecks under the same security conditions as Layer1. Finally, it allows for choosing a modular blockchain for each application.

Technology Architecture

1. Hierarchical module structure

Tusima modules are categorized into L1, L2, and Client. Below, we will provide a detailed introduction to each of them.

- L1 consists of smart contracts mainly of the following two:
  - Rollup

    Rollup is the main contract on the first layer, and its main functions are:

    * Register, i.e. registering a new account with L2.
    * Deposit
    * Withdraw
  - Forge, the Forge function is called by the L2 Operator (the L2 node program) to build a new Batch (the L2 block). During this process, the smart contract will conduct the following:

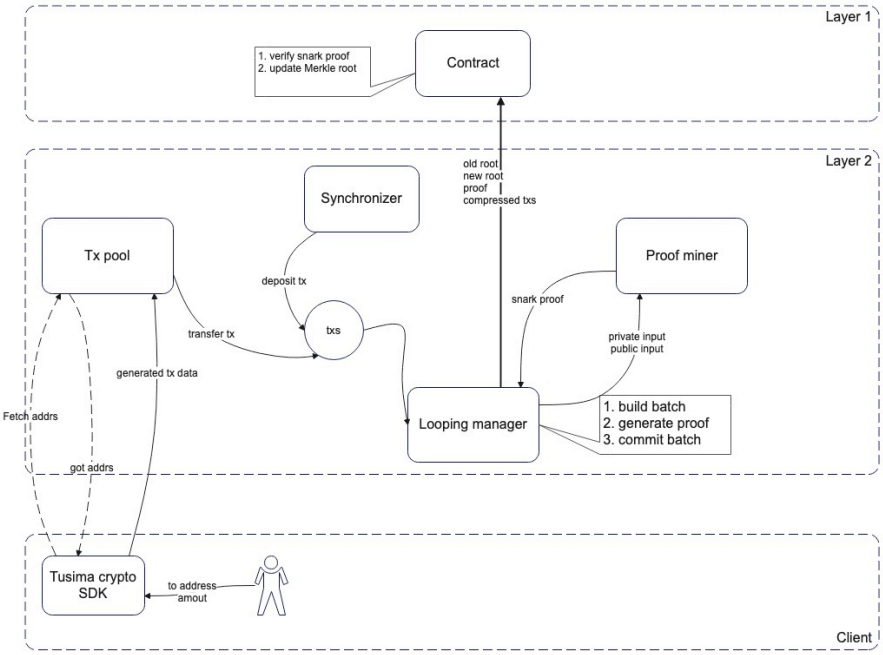

Figure 6: Hierarchical module structure

* Verify that the Proof submitted by the Operator is legal
* Verify that the old Merkle root is correct
* Verify that the on-chain transaction of the previous block is consistent with the on-chain transaction processed in the Operator circuit through hash verification
* If the above verifications are all passed, update the state on the chain, i.e. record the new Merkle tree root
* Save compressed off-chain transaction data to provide data validity.

– RollupPob

RollupPob is an Operator management contract, which will determine who should mine through an auction (Build batch on L2).

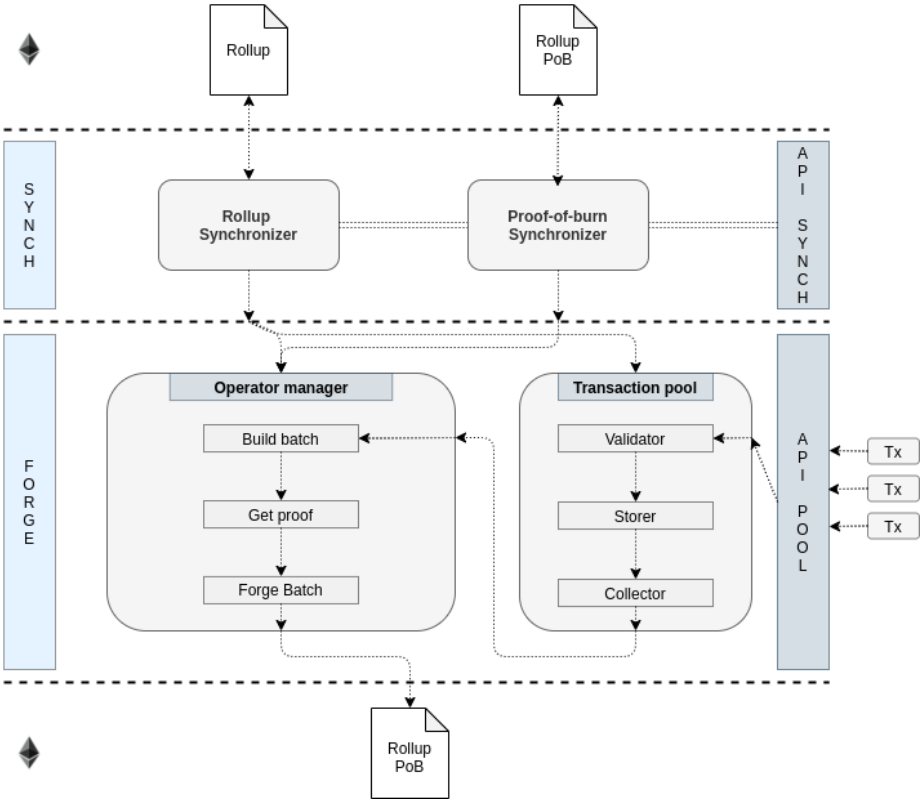

Figure 7: L2 Architecture

- L2 is composed of Operator nodes. Operator is the core module of Tusima, as introduced below.

  As a Rollup node, Operator can synchronize data and also cast batches

(Forge batches). So Operator consists of two modules:

  - Synchronizer

    Synchronizer manages all data from L1. Its main purpose is to continuously monitor `Rollup.sol` and `RollupPoB.sol` contracts to keep the database consistent with the state of L1.

  - Operator Manager

    The standard workflow for Operator Manager is as follows:

    * Interact with the Synchronizer module to check if the database is up to date.
    * Check if the operator's wallet address is the current minting batch node (i.e. Winner).
      · If the Operator is Winner: Wait for the arrival of the starting block that this Operator has minting rights, and then execute the minting.
      · If the Operator is not the Winner: If the current Winner has not performed the minting within a certain period of time or has not performed the minting correctly, the minting right will be transferred to the next Operator, and the Operator must always prepare for the arrival of this moment.
    * When minting time has come, check again that the database is fully synced.
    * Request transactions to be packaged from the transaction pool, build and mint batches.
    * Send zk-snark inputs to the server-proof module to request the corresponding zk-proof.
    * After getting the zk-proof, send the minting transaction to the smart contract of the first layer.
    * Monitor whether the transaction is executed successfully.
    * After the transaction is successfully executed, it enters the waiting synchronization state (i.e. it returns to the first step).

- Client is a wallet client, we call it Tusima Wallet.

  As a terminal of Tusima, Tusima Wallet will provide all users with the interactive operation of Rollup. Tusima Wallet is similar to Metamask, but adds the interactive operation of Tusima's L2 network. Its main functions include:

  - The basic wallet function, i.e. HD Wallet. In a nutshell, it is a process of deriving countless private keys from the mnemonic, and then deriving the address from the private key. HD Wallet was proposed by BIP4410.
  - Organize and initiate transactions including registration, recharge, cash withdrawal, transactions, private transactions, cross-chain transactions, etc.

We have packaged multiple SDKs for wallet use, including:

- Layer1 SDK, which mainly provides operations that interact with Layer1, such as recharge, cash withdrawal, etc.
- Layer2 SDK, which mainly provides operations that interact with Layer2, such as transactions, private transactions, cross-chain transactions, etc.
- Crypto SDK, which mainly provides cryptographic algorithms needed for private transactions, such as Sigma protocol, Elgamal, Fisher–Yates Shuffle, etc.

2. Program execution flow

Transaction Classification

Tusima transactions can be categorized as either on-chain or off-chain. In this chapter, we will use the following symbols to represent each type of transaction:

- Transaction tx
- On-chain transaction $tx1$ (L1)
- Off-chain transaction $tx2$ (L2)

On-Chain Transactions

Tusima Rollup has on-chain Deposit and Withdraw transactions. However, Withdraw is divided into two steps: off-chain destruction and on-chain extraction. The remaining transactions are off-chain. On-chain transactions will modify the state both on and off the chain. Similar to off-chain transactions, on-chain ones will also be included in the circuit to generate a proof before being submitted to the chain. Here's how it works:

To ensure consistency between the transactions entered into the circuit and those on the chain, we must perform hash verification. The hash operation is as follows:

$$hash_{tx} = h(tx) \begin{cases} tx_1^1 & h({}^1tx_1) \\ tx_2^1 & h(hash_{tx_1^1+tx_2^1}) \\ tx_3^1 & h(hash_{tx_2^1+tx_3^1}) \\ ... & ... \\ tx_n^1 & h(hash_{tx_{n-1}^1+tx_n^1}) \end{cases}$$

- Generate on-chain transaction hash $hashtxn11 = h(txn1)$
- Generate the corresponding hash in the circuit $hashtxn12 = h(txn1)$
- Compare it in an on-chain contract

$$verify(hash_{tx_n^1}^1, hash_{tx_n^1}^2) \begin{cases} hash_{tx_n^1}^1 & = & hash_{tx_n^1}^2 & \text{success} \\ hash_{tx_n^1}^1 & \neq & hash_{tx_n^1}^2 & \text{failed} \end{cases}$$

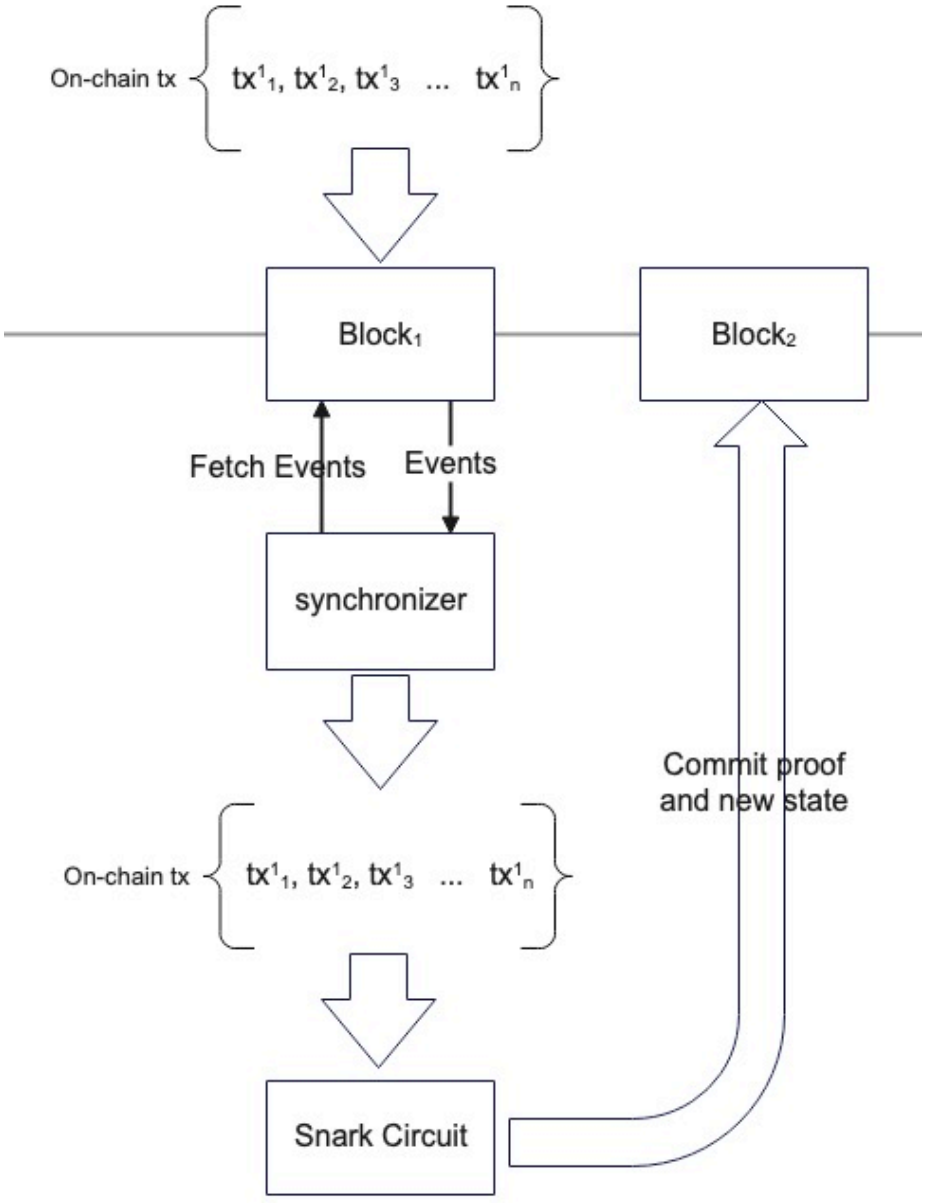

Figure 8: Tusima Flow

Off-chain transactions

Users must submit all off-chain transactions to the Operator node at L2, which will then input them into the circuit program. The program calculates a new Merkle tree root and generates proof, which is submitted to the smart contract of L1. If the smart contract passes verification by the Verifier program, it updates the old Merkle root with the new one. This finalizes the off-chain transaction and makes it irreversible. To summarize:

- Users submit off-chain transactions to Operator node at L2
- Transactions are inputted into circuit program
- New Merkle tree root is calculated and proof generated
- Proof is submitted to smart contract of L1
- Smart contract undergoes verification by Verifier program
- Old Merkle root is updated with new one if verification passes
- Off-chain transaction becomes irreversible

Detailed transaction type

The transaction types are divided into the following five types.

- Deposit (on-chain transaction)
- Transfer (off-chain transaction)
- Deposit (off-chain transaction)
- Private transfer (off-chain transaction)

  A private transaction is an off-chain transaction. Before initiating a transaction, the user first encrypts the transaction data using a homomorphic encryption algorithm, and then proves the validity of the transaction to the Operator node.
- Withdraw needs to be done in two steps:
  - Destroy off the chain, i.e. Withdraw off chain

    This is actually an off-chain transaction whose To address is 0x0, which can also be understood as burn.
  - Withdraw on chain

    Once the specified amount of assets are destroyed on L2, we can extract them on the chain. To do this, we must prove to the contract that they have been destroyed off-chain. This is done using Merck Ershu instead of Snark.

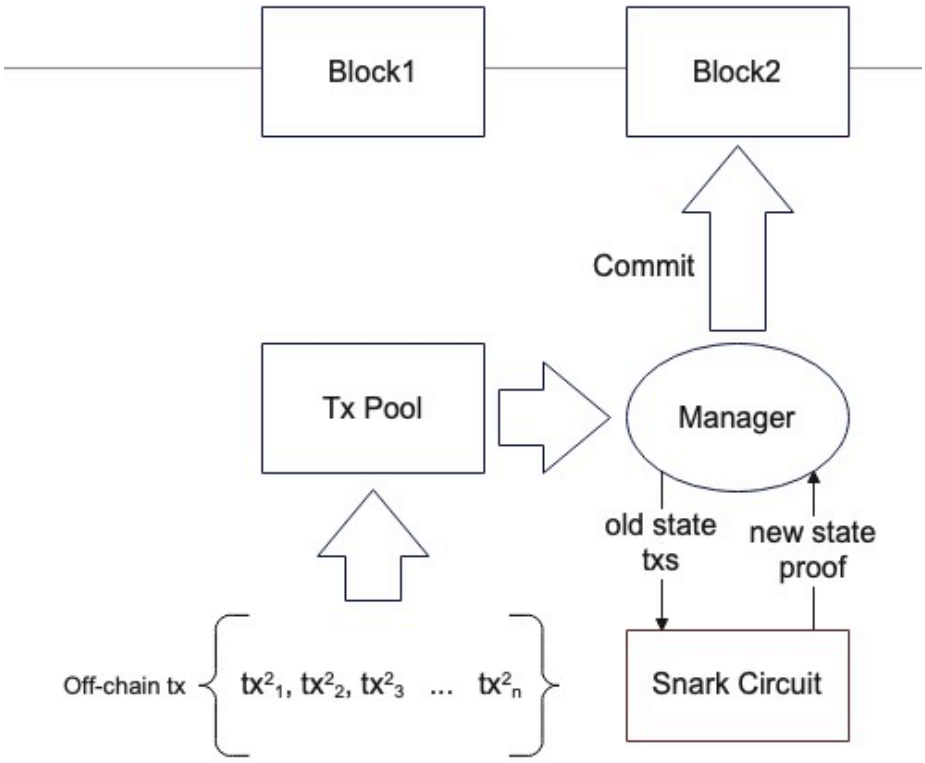

Figure 9: Tusima Flow

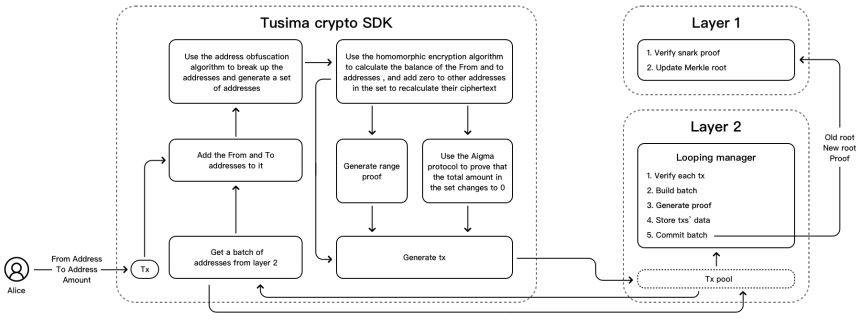

Figure 10: Tusima SDK

Core Algorithm

> Note: The operator symbols used in the framework represent processing logic only and do not reflect the actual data format. The underlying data format is determined by the Crypto Primitive employed.

Tusima transactions are categorized into three types: Deposit, Transfer, and Withdraw. Each transaction type has a corresponding proof agreement. Let's take the Transfer transaction as an example. Its structure is as follows:

$(C1\Delta, C1\Delta, \ , Cn\Delta, address1, address2, \ , addressn)$

We need to verify if the following relation holds true: The encryption result is correct, the user possesses the private key of the account and the account balance falls within range V, and finally, the total amount of changes in all accounts adds up to zero.

$$
\begin{aligned}
\mathcal{R}_{\text{transfer}} = \{ & \\
& (C_i, C_i^{\Delta}, T_i, Y_i, pk_i)_{i\in[1,n]}, g, h; (b_i^{\Delta}, b_i^*)_{i\in[1,n]}, b', sk, (r_i, \overline{r_i}, r_i^*)_{i\in[1,n]} : \\
& (C_{i,L}^{\Delta} = pk_i^{r_i} \wedge C_{i,R}^{\Delta} = g^{r_i} h^{b_i^{\Delta}})_{i\in[1,n]} \wedge \\
& (Y_i / C_{i,R}^{\Delta} = g^{r_i^* - r_i}) \vee \\
& (Y_i / T_i = g^{r_i^* - \overline{r_i}} \wedge pk_i = g^{sk_i} \wedge T_i = (C_{i,R} + C_{i\,R}^{\Delta}) / (C_{i,L} + C_{i\,L}^{\Delta})^{sk_i^{-1}} g^{\overline{r_i}})_{i\in[1,n]} \wedge \\
& \sum_{i=1}^{n} b_i^{\Delta} = 0 \wedge (Y_i = g^{r_i^*} h^{b_i^*} \wedge b_i^* \in \mathcal{V})_{i\in[1,n]} \\
& \}
\end{aligned}
\tag{2}
$$

definition:

$$
\mathcal{R}_{\text{transfer}} = \mathcal{R}_{\text{transfer, validEnc}} \wedge \mathcal{R}_{\text{transfer, range}} \wedge \mathcal{R}_{\text{transfer, Com}} \wedge \mathcal{R}_{\text{transfer, Sum}}
$$

in

$$
\mathcal{R}_{transafer,valide\mathcal{E}nc} = (C_i^{\Delta}, pk_i)_{i\in[1,n]}; (r_i, b_i^{\Delta})_{i\in[1,n]} : C_{i,L}^{\Delta} = pk_i^{r_i} \wedge C_{i,R}^{\Delta} = g^{r_i} h^{b_i^{\Delta}}
$$

$$
\mathcal{R}_{transafer,\mathcal{C}om} = ((C_i, C_i^{\Delta}, Y_i, T_i, pk_i)_{i\in[1,n]}, g, h; (r_i^*, r_i, b_i^{\Delta}, b_i', sk_i)_{i\in[1,n]} : Y_i / C_{i,R}^{\Delta} = g^{r_i^* - r_i} \vee (Y_i / T_i = g^{r_i^* - \overline{r}_i} \wedge p
$$

$$
\mathcal{R}_{\text{transafer, range}} = \left((Y_i)_{i\in[1,n]}, g, h; (r_i^*, b_i^*)_{i\in[1,n]}\right) : Y_i = g^{r_i^*} h^{b_i^*} \in \mathcal{V}
$$

$$\mathcal{R}_{transfer,\mathcal{S}um} = ((C_i^{\Delta})_{i\in[1,n]}), g; (b_i^{\Delta}, r_i)_{i\in[1,n]}: \sum_{n=1}^{n} C_{i,R}^{\Delta} = g^{\sum_{i=1}^{n} r_i}$$

Related Sigma Protocols:

Validated sigma protocol for $\mathcal{R}$

P selects $ri,\ bi\Delta \leftarrow R \quad P$, calculates $AC, L\Delta = pki\ ri,\ AC, R\Delta = g\ rih\ bi\Delta$, will $ A_{C, L}^{}{}$, $AC, R\Delta$ sents to V

V chooses $c \leftarrow R \quad p$, and sends it to P for a random challenge

P calculates $zrsi = \ ri + cri,\ zbi\Delta = \ bi\Delta + cbi\Delta$, sends to V, the calculation equation is verified by V

$pkizri = ACi, L\Delta(Ci, L\Delta)c$

$gzrihzbi\Delta = ACi, R\Delta(Ci, R\Delta)c$

If the verification is successful, the Statement is established

- Sigma protocol for $\mathcal{R}$ verification

  $\mathcal{R}_{\text{transfer, sum}}$

The interaction between P and V is as follows:

P selects $\alpha_{\text{sum}} \leftarrow \mathbb{Z}_p$, and calculates $A_{\text{Sum}} = g^{\alpha_{\text{sum}}}$, and sends A_{Sum} to V.

V chooses $c \leftarrow \mathbb{Z}_p$ and sends to P as a random challenge.

P calculates $\mathbb{z}_{\mathbb{sum}} = \alpha_{sum} + c(\sum_{i=1}^{n} r_i)$ and sends to V

V is calculated if it satisfies: $g^{\mathbb{z}_{\mathbb{sum}}} = A_{Sum}(\sum_{i=1}^{n} C_{i,R}^{\Delta})^c$, then accepts.

P selects $z_{r_i}, z_{r_i^*}, z_{b_i^{\Delta}}, c_0, \alpha_{r_i^*}, \alpha_{b_i'}, \alpha_{\overline{r_i}}, \alpha_{sk_i} \leftarrow R \in \mathbb{Z}_p$, calculates the commitment value as follows

$$A_{C_{i,0}^{\Delta}} = (C_{i,R}^{\Delta})^{-c_0} g^{z_{r_i}} h^{z_{b_i^{\Delta}}} A_{Y_{i,1}} = g^{z_{r_i^*}} h^{\alpha_{b_i'}} A_{T_i} = g^{\alpha_{\overline{r}_i}} h^{\alpha_{b_i'}} A_{pk_i} = g^{\alpha_{sk_i}} A_{T_i(C_{i,R}+C_{i,R}^{\Delta})^{-1}} = [(C_{i,L}+C_{i,L}^{\Delta})^{-1}]^{\alpha_{sk_i}^{-1}} g^{\alpha_{\overline{r}}}$$

V selects $c \leftarrow R \quad p$ in the challenge space, and sends it to P as the challenge value

P calculates $c_1 = c c_0 $ *zri** = * ri** + $c1$*ri**, $zbi\ =\ bi\ + c1bi$, $z_{\overline{r}_i} = \alpha_{r_i'} + c_1\overline{r}_i$, $zski = \ ski + c1ski$, $zsk - 1 = \ sk - 1 + c1ski - 1$ and sends to V

V is calculated as the following equation

$$
\begin{aligned}
g^{z_{r_i}} h^{z_{b_i^{\Delta}}} &= A_{C_{i,R}^{\Delta}} (C_{i,R}^{\Delta})^{c_0}, \\
g^{z_{r_i^*}^{'}} h^{z_{b_i^{'}}} &= A_{Y_{i,1}} (Y_i)^{c_1}, \\
g^{z_{\overline{r}_i}} h^{z_{b_i^{'}}} &= A_{T_i} (T_i)^{c_1}, \\
g^{sk_i} &= A_{pk_i} (pk_i)^{c_1}, \\
[(C_{i,L} + C_{i,L}^{\Delta}]^{z_{sk_i^{-1}}} g^{z_{\overline{r}_i}} &= A_{T_i (C_{i,R} + C_{i,R}^{\Delta})^{-1}} [T_i (C_{i,R} + C_{i,R}^{\Delta})^{-1}]^{c_1}.
\end{aligned}
\tag{3}
$$

If the above equality holds, the Statement is accepted.

Experimental data

Twisted Elgmal:

Data size	Encryption	decryption
100K	512 s	1.25s
1M	522 s	9.03s
10M	553 s	82.03s
100M	568 s	387.87s

Range Proof:

Operation	Native	Circuit
Prove	18ms	/
Verify	10ms	2.02s

ZK Rollup Circuit:

Operation	Constraints	Time
Poseidon hash function	600	0.005s
Ecdsa verification	93470	0.522s
Range proof verification	339803	2.021s
Transaction in L2	9087683	2.930s
Deposit transaction in L2	9087672	2.927s
Withdraw transaction in L2	9087683	2.930s

Operation data

Since the testnet's release in October 2022, it has garnered significant attention from the industry and users alike. As of January 31st, 2023, there have been

64,854 successful fund transfers and recharges to addresses on the network. Additionally, 42,932 addresses have completed all tests and received TSBT. The project community has grown substantially with over 23K Twitter followers and a Discord community of 29K members. Testnet-related activities are widely disseminated through various channels including more than 30k tweets and over 60 industry media distributions. Community members have also initiated more than 300 Tusima testnet tutorials covering fifteen national languages.

Reference

1. Rollup https://ethereum.org/en/developers/docs/scaling/
2. ZK-Rollup https://ethereum.org/en/developers/docs/scaling/zk-rollups/
3. zkSnarks https://chriseth.github.io/notes/articles/zksnarks/zksnarks.pdf
4. r1cs https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649
5. Bunz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short Proofs for Confidential Transactions and More. Proceedings - IEEE Symposium on Security and Privacy 2018-May, 315–334 (2018)
6. Sigma https://users-cs.au.dk/~ivan/Sigma.pdf
7. ZCash https://z.cash/
8. Tornado cash https://tornadocash.eth.link/
9. Elgmal https://link.springer.com/chapter/10.1007/BFb0054019
10. BIP44 https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki