Handbook

Blockchain Layer 2 Scaling Solution

Layer 2 research team

HPB underlying technical team

January 14, 2022

Table of Contents

List of Tables

List of Figures

1. Blockchain Layer 2 Scaling Solution

Blockchain Layer 2 [7] refers to the framework or protocol built on top of the existing blockchain system. The main goal of these designs is to solve the low transaction speed and scaling difficulties faced by major cryptocurrency networks. For example, Bitcoin and Ethereum are still unable to process thousands of transactions per second (TPS), which is not conducive to their further development. Before these networks can be effectively adopted and used more widely, higher throughput is required. In this case, the term "Layer 2" refers to multiple solutions to the scalability problem of the blockchain, the two main examples are Bitcoin Lightning Network and Ethereum Plasma.

Technical speaking, the Layer 2 protocols create a framework in which blockchain transactions and processes can be performed independently of the first layer (main chain). For this reason, these technologies can also be referred to as "off-chain" scaling solutions. For example, the Lightning Network is based on a state channel, which is basically an additional channel that performs blockchain operations offline, and then reports them to the main chain. On the other hand, the Plasma framework is composed of side chains, which are essentially small blockchains arranged in a tree structure.

One of the main advantages of using an off-chain solution is that the main chain does not need to undergo any structural changes, because layer 2 is added as an additional layer. The security of the main chain itself is independent of layer 2. Therefore, layer 2 solutions have the potential to achieve high throughput without sacrificing network security. In other words, most of the work performed by the main chain can be transferred to the second layer. Therefore, while the main chain (layer 1) provides security, the second layer provides high throughput, capable of executing hundreds or even thousands of transactions per second.

There are some existing layer 2 design solutions on the market such as zk-Rollup [2], Plasma [6], Nocust [3]. They all introduce transaction custody to process user transactions offline. These schemes are dedicated to moving a large amount of computing from the main chain to layer 2, but most schemes do not take into account the privacy of users. The user's balance and transaction data are transparent in these systems, which is not conducive to the user's privacy. However, it is difficult to have both scalability and privacy at the same time, because increasing the privacy situation of the system will increase the amount of data and calculations, thereby reducing the throughput of the blockchain and increasing transaction costs. Scalability and privacy are the two sides of the coin and require appropriate trade-offs.Table 1 compares the three mainstream layer 2 expansion solutions with the solution we proposed from the four perspectives of security, applicability, privacy, and functional evaluation.

Our project is committed to finding a balance between privacy and expansibility, so as to increase privacy for the expansion schemes in the current market without completely reducing their application value. Like other expansion solutions, we have introduced a transaction escrow party in the system. The transaction escrow packages the transaction and gives a proof of the calculation, and sends the compressed transaction information and proof to the block volume. In our current design, transaction escrow is semi-trusted: Since the transaction escrow will provide proof of the correctness of its own calculations, users do not need to believe that transaction escrow will calculate transactions correctly, but users need to believe that transaction escrow will keep their transaction information confidential.

2. Privacy-preserving Layer 2 Scaling Solution

Similar to other Layer 2 expansion solutions, we also introduce a "semi-trusted" transaction escrow into the system. The transaction escrow updates the users' privacy balance based on the transaction. The update calculation is trustless, that is, the transaction escrow cannot Change the user data at will, and the user balance data calculated based on the transaction data is always correct. This correctness is guaranteed by

Table 1. Comparison of differences of Layer-2 Scaling Solution [1]

Scheme		Plasma [6]	ZK-Rollup [2]	State Channel [5]	Our Scheme
Security	No Mass Exit Assumption	✗	✓	✓	✓
	No Watch Tower Assumption	✗	✓	✗	✓
	Cryptographic Primitives	Standard	New	Standard	Standard
Usability	Withdraw Time	1 week	10 minutes	1 confirmation	1 confirmation
	Transaction Finalization Time	1 confirmation	1 confirmation	Instant	1 confirmation
	User Verification	✗	✗	✓	✓
Privacy	Confidential Transaction	✗	✗	✗	✓
	Confidential User Address	✗	✗	✗	✗
	Anonymity Set	Small	Large	Small	Medium
Performance	Throughput	1k–9k TPS	2k TPS	1–∞ TPS[a]	-[b]
	Cost of Transaction	Very low	Very low	Low	-
	No Collateral Required	✓	✓	✗	✓

[a] Depends on the scheme.
[b] The performance of scheme depends on the later tests.

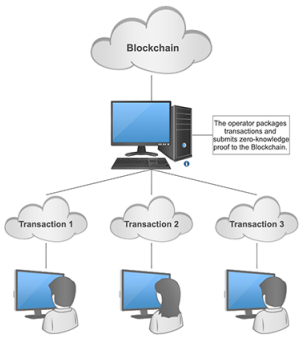

Fig. 1. Layer 2 Work Flow

the zero-knowledge proof technology. But users need to assume that transaction escrow will not leak data, because transaction escrow has all transaction information and user balances. Fig. 1 shows the system workflow. The overall process of the system is as follows:

1. The user signs the transaction and sends it to the transaction escrow
2. transaction custody packs transactions within a period of time, makes the updated balance of each user as a cryptographic commitment (commitment), calculates zero-knowledge proofs, and sends the commitments and proofs to the underlying blockchain.
3. Promises and zero-knowledge proofs are verified online by smart contracts. After verification is passed, the related transactions are completed.

Compared with other Layer 2 expansion solutions, our solution enhances the privacy of users. Considering that complete privacy (anonymous transactions) requires a greater computational burden, we weighed

the privacy and feasibility and decided to achieve user transaction data privacy first. Follow-up research on this basis will achieve anonymous transactions.

2.1 User Transfer Working Flow

Register When a user registers, he must first send a transaction request to the transaction escrow, including the user's pre-deposited quota and the user's signature. The transaction escrow returns to the user a random value and a Pedersen promise of a random value and signs the information. Then the user directly interacts with the smart contract, pre-stores a certain amount of balance, and stores the promise of random value.

Transfer When the user makes a transaction, the transaction and signature are sent to the transaction escrow. After the transaction escrow verifies the legality of the transaction, if this is the user's first transaction in an epoch, the transaction escrow will return the following information to the user:

1. A random number t and its Pedersen promise
2. Pedersen's commitment to the user's remaining amount and transaction value
3. The Pedersen commitment list of the transaction values of all transactions completed by the user in this epoch.
4. Signature of the above information

If this is not the first transaction sent by the user in an epoch, the operator only needs to send 2, 3, and 4.

The user checks the validation of the transaction list and then returns the user signature to the Pedersen commitment list of the transactions.

Update status on chain The transaction is hosted at the end of the epoch, and the transaction list of all users who want to send transactions during this epoch time period will be collected, as well as the promises of these transactions and the signatures from the users. Transaction custody calculates the user's new balance based on these transactions. Finally, the transaction escrow sends these data to the smart contract:

User index	Transaction list and signatures	Randomness and its signature	Balances Commitments	d
u_1	$\mathfrak{T}_1^r,\ \sigma_1$	$c_{t_1},\ \sigma_{t_1}$	c_1	d_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
u_i	$\perp,\ \perp$	$c_{t_i},\ \perp$	c_i	d_i
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
u_N	$\mathfrak{T}_N^r,\ \sigma_1$	$c_{t_N},\ \sigma_{t_N}$	c_N	d_N
Operator's signature: σ_o				
Zero-Knowledge proof: π				

Among them, d is a public value used to hide the user's balance, which satisfies $b = t - d$, b is the user's balance, and t is a random value shared by the transaction escrow and the user. For each user who did not select a random value of t_i for online and transaction escrow in this epoch, the smart contract updates the user's balance commitment as his/her balance commitment in the previous epoch and returns the value sent to this user. Therefore, if user u_j sends user u_i worth C_{ji} to this user, then user u_j balance commitment is updated to $c_j = c_j \cdot C_{ji}^{-1}$

User Withdrawal The user sends a withdrawal request to the smart contract. The withdrawal amount is $b = t - d$, t is the random value currently shared with the transaction escrow, and d is the public value. The user also submits the signature of the operator for t.

3. Technical Overview

Notations We use the following notations

- Op : Operator
- SC : Smart Contract
- r : epoch number
- pk_i : the public key of u_i
- b_i : the balance of u_i
- c_i : the commitment for the balance of u_i
- t_i : the randomness of u_i
- c_{t_i} : the commitment for t_i of u_i
- v_{ij} : the transaction value that u_i sends to u_j
- σ : signature
- B : the total balance of the smart contract
- Tx'_{ij}:$(pk_i, pk_j, v_{ij}, \mathsf{Null}, r, n, \sigma_{ij})$: the plaintext that a user sends to the operator
- Tx_{ij}:$(pk_i, pk_j, v_{ij}, \mathsf{c_{ij}}, r, n, \sigma_{ij})$: the confidential transaction that the operator sends to a user
- CTx_{ij} : (pk_j, c_{ij}) the short representation for the transaction that u_i sends to u_j
- $\mathcal{T}r$: transaction list in plaintext
- $\mathcal{CT}_r$: confidential transaction list
- H : hash function

3.1 Work Flow

Registration To register an account, a user u_i sends a request Reg consisting of signature σ_{b_i} on its balance b_i to the operator during an epoch r. The operator returns a random t_i with its commitment proofs and signature on these values. Further, the user verifies all the details and accordingly sends a registration request to the smart contract along with a deposit transaction. The smart contract verifies the request and registers the user accordingly. After a send/receive transaction, the user's balance becomes private in later epochs. Fig. 2 shows the user registration steps.

Transfer Fig. 3 depicts the transferring steps. When the user makes a transaction, the transaction and signature are sent to the transaction escrow. After the transaction escrow verifies the legality of the transaction, if this is the user's first transaction in an epoch, the transaction escrow will return the following information to the user:

1. A randomness t and its Pedersen commitment
2. The Pedersen commitments for user's transactions and balance
3. The confidential transaction list of this epoch
4. A signature for the above information

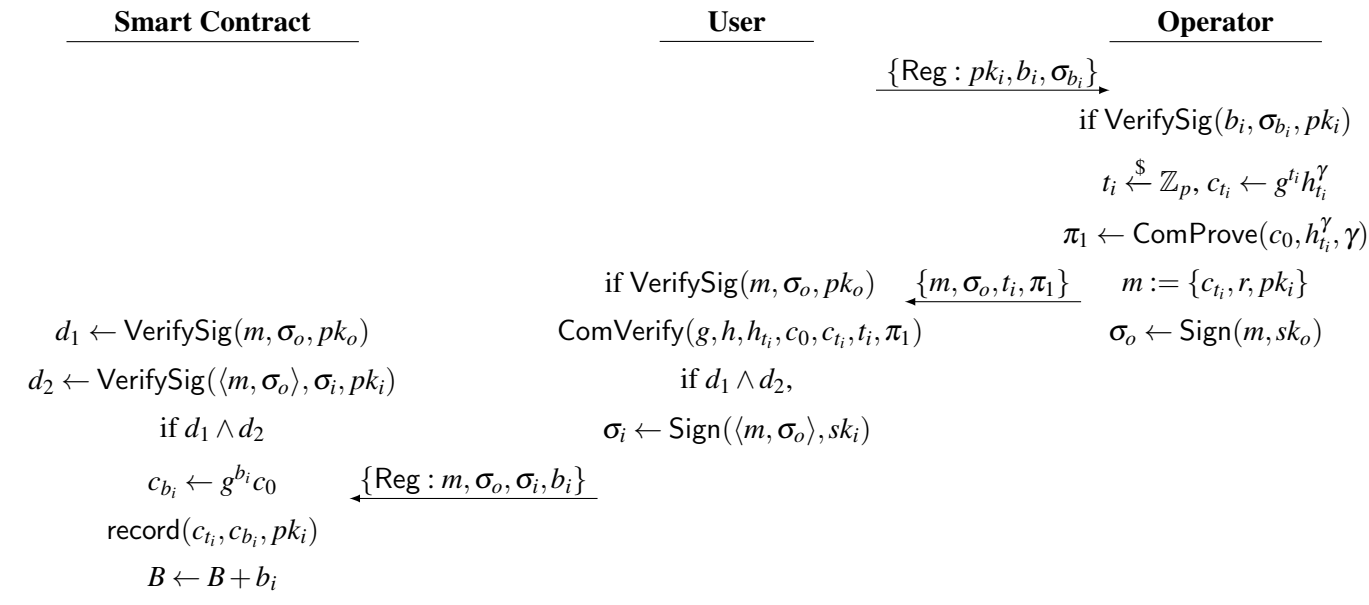

Fig. 2. User Registration

Update states on blockchain At the end of the epoch, the operator collects all the transaction lists from users, computes the updated balances, and sends the states to the blockchain.

User index	Transacion list and signature	randomness and its signature	balance commitment	d
u_1	$\mathcal{CT}_1^r,\ \sigma_1$	$c_{t_1},\ \sigma_{t_1}$	c_1	d_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
u_i	$\perp,\ \perp$	$c_{t_i},\ \perp$	c_i	d_i
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
u_N	$\mathcal{CT}_N^r,\ \sigma_1$	$c_{t_N},\ \sigma_{t_N}$	c_N	d_N
The signature of the operator: σ_o				
zero-knowledge proof: π				

d is a public value that is to hide the user's balance, it satisfies $b = t - d$, b is the user's balance$t$ is randomness agreed by user and operator. For every user u_i who did not agree on a randomness with the operator in this epoch, smart contract updates this user's balance commitment as his/her balance commitment in the previous epoch and for the other users who have sent a transaction with value C_{ji} to this user, it updates their balance commitments as $c_j = c_j \cdot C_{ji}^{-1}$

Withdraw The user sends a withdrawal request to the smart contract $b = t - d$, at the same time, sends the signature of the operator for t.

4. Arithmetic Circuit and Qudratic Arithmetic Program QAP

The calculation of the user balance update by the operator is represented by an arithmetic circuit C, and the arithmetic circuit is further represented by the Quadratic Arithmetic program [4].

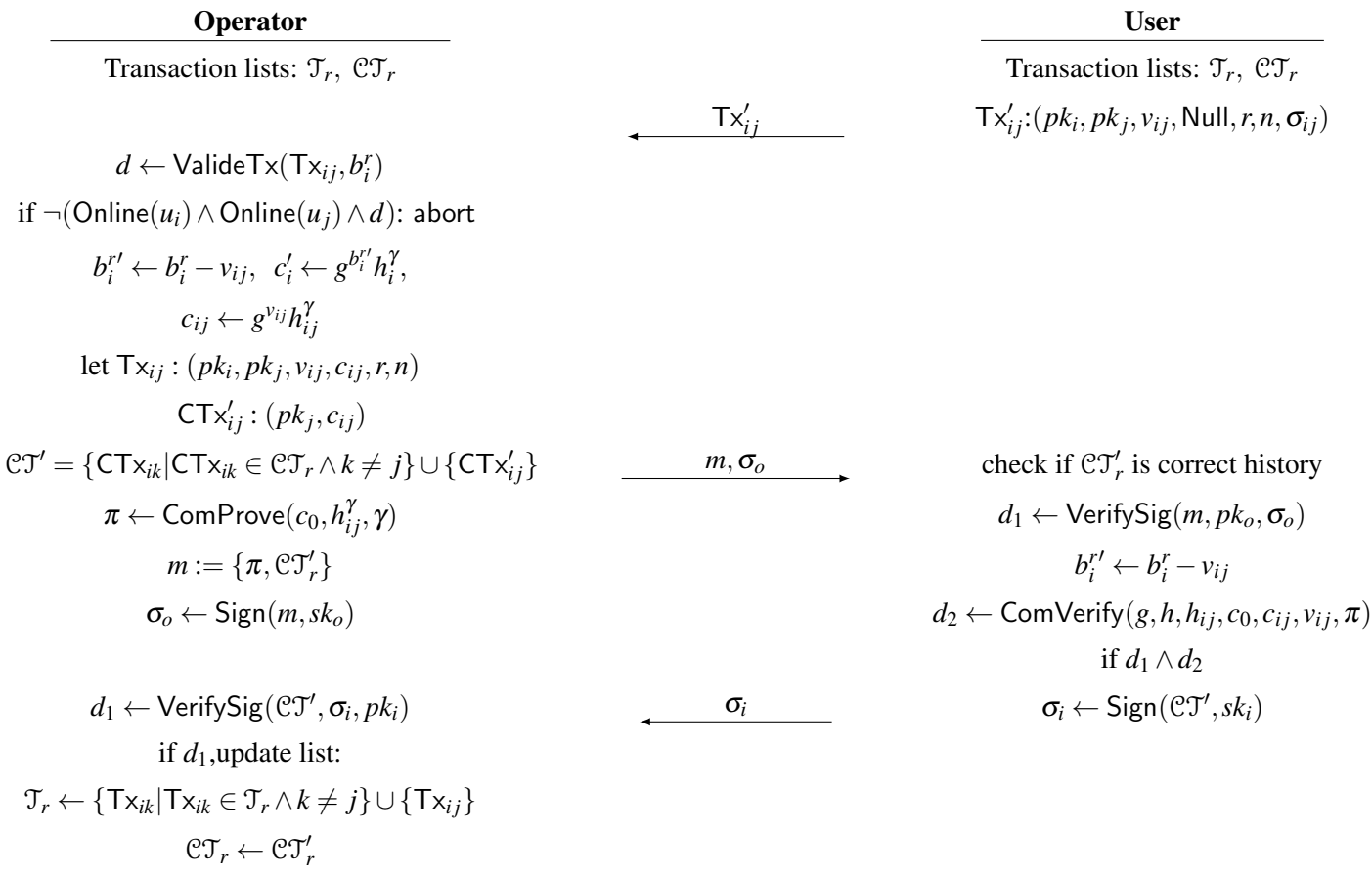

Fig. 3. Transfer

arithmetic circuit Arithmetic circuit C The calculations that need to be done include

1. Update user balance from transaction data
   $b'_i = b + \sum_{j \neq i} v_{ji} - \sum_{j \neq i} v_{ij}$
   Where v_{ji} is the transaction value received by the user in the current epoch, and v_{ij} is the transaction value sent by the user.
2. User transaction value, range proof of user balance, that is, to ensure that these values are positive and within the correct range. The Token value stored in the
3. smart contract is consistent with the user's behavior.
4. Calculate $b_i = s_i - t_i$.

QAP An arithmetic circuit can be expressed as a quadratic polynomial:

$$\sum_{i=1}^{n} a_i u_i(X) \cdot \sum_{i=1}^{n} a_i v_i(X) = \sum_{i=1}^{n} a_i w_i(X) + h(X)z(X)$$

Among them, a_i is the value of each wire of the circuit, including the balance of all users, transaction value, all random values, etc., $u_i(X), v_i(X), w_i(X)$ are Polynomials calculated from the circuit arithmetic structure, and $h(X), z(X)$ are Polynomials calculated from the circuit wiring structure. X stands for the label of circuit gates.

The idea of the algorithm is that we promise to hide private data such as all user balances, transaction values, and all random values by Pedersen, and other wire values of the circuit are promised to hide by Pedersen vector, and finally give an arithmetic circuit that meets all the above values (That is, the QAP equation is satisfied), which proves that the calculation performed by the transaction escrow is correct.

5. Core Algorithm Overview

5.1 Pre-computing

1. set the total number of users Ncircuit compiler computes the polynomials of x $u_i(x), v_i(x), w_i(x), h_i(x)$.
2. For every epoch, the smart contract needs to compute (N^2+N) new generators h_i of Pederson Commitment.

5.2 The computation of the operator

After the user sends the transaction within a period, the transaction custody collects the following data:

- The commitment of each user's balance, the commitment of value sent by each user to other users, these commitments are marked with $\{c_i\}_{i=1}^{k}$.
- The Pedersen Vector promise of other privacy values in the QAP circuit, expressed in c_l.
- Commitment to the value disclosed in the circuit $\{c_i\}_{l+1}^{n}$

After having these values, the zero-knowledge proof can be calculated next to Commit. The specific steps are as follows:

1. takes the coefficient of $h(x)$ as a Pedersen vector promise, and the result obtained is denoted by c_h.
2. Combine all public input values $((g, h, \{h_i\}_{i=1}^{k}, \{c_i\}_{i=1}^{k}, c_l, c_h, \boldsymbol{g}), \{a_i\}_{i=l+1}^{n})$ do hashing, and the result obtained is the challenge x_1, calculate the result of the circuit QAP polynomial $\{u_i(x_1), v_i(x_1), w_i(x_1), h_i(x_1)\}_{i=1}^{n}$
3. Transaction custody selects random numbers t_u, t_v, t_w and calculates
   $c_u = g^{\sum_{i=1}^{k} a_i u_i(x_1)} h^{t_u}$, $c_v = g^{\sum_{i=1}^{k} a_i v_i(x_1)} h^{t_v}$, $c_w = g^{\sum_{i=1}^{k} a_i w_i(x_1)} h^{t_w}$
   Set $\boldsymbol{b} = \{u_i(x_1)\}_{i=1}^{k}, \boldsymbol{b} = \{v_i(x_1)\}_{i=1}^{k}, \boldsymbol{b} = \{w_i(x_1)\}_{i=1}^{k}$ as input respectively, compute the algorithm shown in Fig. 4 three times.

Input: $(\{h_i\}_{i=1}^n, \{c_i\}_{i=1}^k, g, h, c \in \mathbb{G}; \{a\}_{i=1}^k, \boldsymbol{b} \in \mathbb{Z}_p^n; \gamma, t \in \mathbb{Z}_p)$ Output: (accept/reject)

Operator input:$(\{h_i\}_{i=1}^n, g, h, \boldsymbol{a}, \boldsymbol{b}, \gamma, t)$
Smart contract input:$(\{h_i\}_{i=1}^n, g, c, \{c_i\}_{i=1}^n)$

(a) Operator chooses randomness $\alpha, \beta, t \xleftarrow{\$} \mathbb{Z}_p$, compute

$$c_0 = h^\gamma, \quad \tau = \prod_{i=1}^n h_i^{b_i}, \quad \Omega = \tau^\gamma h^{-t},$$

$$d_1 = h^\alpha, \quad d_2 = \tau^\alpha h^\beta \text{ and sends } (c_0, \Omega, d_1, d_2) \ \mathcal{V}$$

(b) Compute the hash of $(c_0, \Omega, d_1, d_2 \{h_i\}_{i=1}^n, g, c, \{c_i\}_{i=1}^n, \boldsymbol{b})$ as challenge x

(c) Operator computes $\theta_1 = \alpha - x\gamma, \theta_2 = \beta + xt$

(d) Smart contract computes $\tau = \prod_{i=1}^n h_i^{b_i}$, only the following equations hold, then accept

$$c = \frac{\prod_{i=1}^n c_i^{b_i}}{\Omega} \quad \wedge \quad d_1 = c_0^x h^{\theta_1} \quad \wedge \quad d_2 = \Omega^x \tau^{\theta_1} h^{\theta_2}$$

Fig. 4. The inner product algorithm for user balance Pedersen commitments

4. the operator chooses s_u, s_v, s_w and computes

$$c_{\boldsymbol{u}} = g^{\sum_{i=k+1}^{l} a_i u_i(x_1)} h^{s_u}, \quad c_{\boldsymbol{v}} = g^{\sum_{i=k+1}^{l} a_i v_i(x_1)} h^{s_v}$$

$$c_{\boldsymbol{w}} = g^{\sum_{i=k+1}^{l} a_i w_i(x_1)} h^{s_w}$$

$$c_{hz} = g^{h(x_1) z(x_1)} h^{s_h}$$

set c_l as c_a, set $\{u_i(x_1)\}_{i=k+1}^l, \{v_i(x_1)\}_{i=k+1}^l, \{w_i(x_1)\}_{i=k+1}^l$ as vector $\boldsymbol{b}$ compute the following algorithm three times, the corresponding $c_{\boldsymbol{ab}}$ are $c_{\boldsymbol{u}}, c_{\boldsymbol{v}}, c_{\boldsymbol{w}}$. and set c_h as vector c_a, vector b is $\{z(x_1), x_1 z(x_1), x_1^2 z(x_1), ..., x_1^{n-2}\}$, set c_{hz} as the corresponding $c_a b$ and run the algorithm shown in Fig. 5.

5. to Prove, verifies whether the QAP equation holds, that is

   (a) compute

$$c_a = c_u \cdot c_{\boldsymbol{u}} \cdot g^{\sum_{i=l+1}^{n} a_i u_i(x_1)}, c_b = c_v \cdot c_{\boldsymbol{v}} \cdot g^{\sum_{i=l+1}^{n} a_i v_i(x_1)}, c_c = c_w \cdot c_{\boldsymbol{w}} \cdot g^{\sum_{i=l+1}^{n} a_i w_i(x_1)} \cdot c_{hz}$$

   (b) The algorithm shown in Fig. 6 is used to verify that the value of c_c is the product of c_a, c_b promised values.

6. Demo

In order to evaluate the usability of the Layer 2 extended protocol, we give a simple implementation in *Go*. Fig. 7 shows the performance. The experiment was conducted on a 2×24 Xeon 2.4 GHz core.

7. Summary

The scalability of the blockchain and the privacy after expansion are both problems that the blockchain application must solve. The goal of this solution is to achieve user privacy based on the technology of the blockchain expansion solution Layer 2. To this end, we have built an efficient Commit-and-Prove zero-knowledge proof protocol. This solution can realize user transaction value privacy and user balance privacy

The operator computes the hash of $(\boldsymbol{g},u,h,c,\boldsymbol{b})$ as challenge x. Compute $c = c_{\boldsymbol{a}}c_{\boldsymbol{ab}}^{x}$, $r = r_a + r_{ab}x$, $u = g^x$ and set $(\boldsymbol{g},u,h,\boldsymbol{a},\boldsymbol{b},c,r)$ as input

Zero-knowlege proof:

Input: $(\boldsymbol{g} \in \mathbb{G}^n, u,h,c \in \mathbb{G}; \boldsymbol{a},\boldsymbol{b} \in \mathbb{Z}_p^n, r \in \mathbb{Z})$ output: (accept/reject)

Operator input$(\boldsymbol{g},u,h,c,\boldsymbol{a},\boldsymbol{b})$
Smart contract input$(\boldsymbol{g},u,h,c,\boldsymbol{b})$
If $n = 1$ $(\boldsymbol{a} := \{a_1\}, \boldsymbol{g} := \{g_1\})$:

(a) operator chooses randomness $\alpha_1, \alpha_2 \xleftarrow{\$} \mathbb{Z}_p$, computes and sends $d = g_1^{\alpha_1} u^{\alpha_1 b_1} h^{\alpha_2}$ to the smart contract

(b) the operator computes the hash of $(\boldsymbol{g},u,h,c,\boldsymbol{b},d)$ as challenge x and send to smart contract

(c) the operator computes $\theta_1 = \alpha_1 - xa_1, \theta_2 = \alpha_2 - xr$, send θ_1 and θ_2 to the smart contract

(d) if $c^x g_1^{\theta_1} u^{b_1\theta_1} h^{\theta_2} = d$ accept, otherwise reject.

if $n > 1$:

(a) if $n' = \frac{n}{2}$, operator choose randomness $r_1 \xleftarrow{\$} \mathbb{Z}_p$ and $r_2 \xleftarrow{\$} \mathbb{Z}_p$. and computes L,R as follows, sends L,R to the smart contract

$$L = \boldsymbol{g}_{[n':]}^{\boldsymbol{a}_{[:n']}} \cdot u^{\langle \boldsymbol{a}_{[:n']}, \boldsymbol{b}_{[n':]} \rangle} \cdot h^{r_1} \in \mathbb{G}$$

$$R = \boldsymbol{g}_{[:n']}^{\boldsymbol{a}_{[n':]}} \cdot u^{\langle \boldsymbol{a}_{[n':]}, \boldsymbol{b}_{[:n']} \rangle} \cdot h^{r_2} \in \mathbb{G}$$

(b) the operator computes the hash of $\boldsymbol{g},c,\boldsymbol{b},L,R$ as challenge x

(c) the operator and smart contract both compute

$$\boldsymbol{g}' = \boldsymbol{g}_{[:n']}^{x^{-1}} \circ \boldsymbol{g}_{[n':]}^{x} \in \mathbb{G}^{n'}$$
$$\boldsymbol{b}' = x^{-1}\boldsymbol{b}_{[:n']} + x\boldsymbol{b}_{[n':]} \in \mathbb{Z}_p^{n'}$$
$$c' = c \cdot L^{x^2} \cdot R^{x^{-2}} \in \mathbb{G}$$

(d) the operator computes:

$$\boldsymbol{a}' = x\boldsymbol{a}_{[:n']} + x^{-1}\boldsymbol{a}_{[n':]}$$
$$r = r + x^2 r_1 + x^{-2} r_2$$

(e) runs this zero-knowledge proof iteratively with $((g'),u,h,c',(a'),(b'),r)$ as the input

Fig. 5. Verify Pedersen vector commitments inner product algorithm

Inpute: $(g,h,c,c_a,c_b \in \mathbb{G}; a,b,r_a,r_b,t \in \mathbb{Z}_p)$ Output: accept or reject by the smart contract

$\mathcal{P}$'s input:(g,h,a,b,r_a,r_b,t)
$\mathcal{V}$'s output:(g,h,c_a,c_b,c)

i. $\mathcal{P}$ chooses $\alpha,\beta,r_1,r_2,s_0,s_1$ and compute $d_1 = g^{\alpha}h^{r_1}$, $d_2 = g^{\beta}h^{r_2}$, $c_0 = g^{\alpha b+\beta a}h^{s_0}$, $c_1 = g^{\alpha\beta}h^{s_1}$. $\mathcal{P}$ send (d_1,d_2,c_0,c_1) to $\mathcal{V}$

ii. $\mathcal{P}$ compute the hash of $(g,h,c_a,c_b,c,d_1,d_2,c_0,c_1)$.

iii. $\mathcal{P}$ compute $\theta_a = \alpha - ax$, $\theta_b = \beta - bx$, $\theta_1 = r_1 - r_a x$, $\theta_2 = r_2 - r_b x$ $\theta_{ab} = x^2 t - xs_0 + s_1$ send $\theta_a,\theta_b,\theta_1,\theta_2$ to the smart contract

iv. The smart contract check $c_a^x g^{\theta_a} h^{\theta_1} = d_1$, $c_b^x g^{\theta_b} h^{\theta_2} = d_2$, $g^{\theta_a\theta_b} h^{\theta_{ab}} c_0^x = c^{x^2} c_1$, accept if all the equation hold, otherwise reject

Fig. 6. Verify Pedersen Commitment

on a scalable blockchain that supports smart contracts. We have tested the proposed scheme, and the test results prove that the scheme can provide privacy attributes, but to achieve feasibility, optimization and improvement need to be made in the preprocessing stage of zero-knowledge proof (refer to Libsnark's preprocessing code library for details).

References

[1] V. Buterin. An incomplete guide to rollups. URL https://vitalik.ca/general/2021/01/05/rollup.html.

[2] A. Gluchowski. Zk rollup: scaling with zero-knowledge proofs. Matter Labs, 2019. URL https://pandax-statics.oss-cn-shenzhen.aliyuncs.com/statics/1221233526992813.pdf.

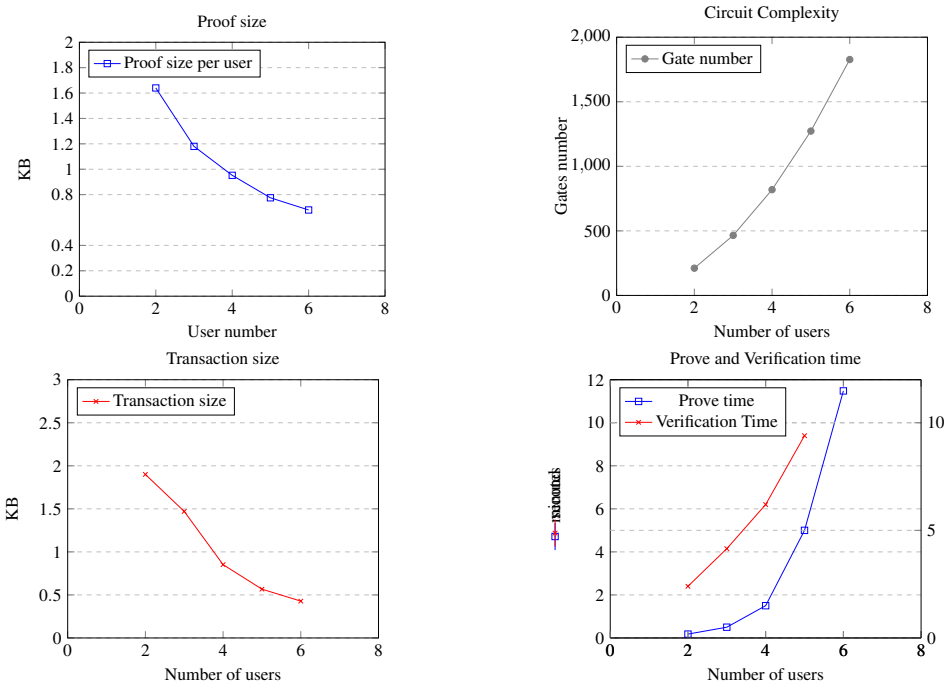

Fig. 7. Experimental results

[3] R. Khalil, A. Zamyatin, G. Felley, P. Moreno-Sanchez, and A. Gervais. Commit-chains: Secure, scalable off-chain payments. Cryptology ePrint Archive, Report 2018/642, 2018. https://ia.cr/2018/642.

[4] E. L. Lawler. The quadratic assignment problem. *Management science*, 9(4):586–599, 1963.

[5] A. Miller, I. Bentov, R. Kumaresan, and P. McCorry. Sprites: Payment channels that go faster than lightning. *CoRR abs/1702.05812*, 306, 2017.

[6] J. Poon and V. Buterin. Plasma: Scalable autonomous smart contracts. *White paper*, 2017.

[7] J. Stark. Making sense of ethereum's layer2 scaling solutions: state channels, plasma, and truebit. *URl: https://medium. com/l4-media/making-sense-of-ethereums-layer-2-scalingsolutions-state-channels-plasma-and-truebit-22cb40dcc2f4*, 2018.