Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 31(2): 729–753.
DOI: 10.3934/era.2023036
Received: 29 August 2022
Revised: 24 October 2022
Accepted: 02 November 2022
Published: 22 November 2022

Research article

A blockchain-based privacy-preserving transaction scheme with public verification and reliable audit

Shuang Yao[1,2] **and Dawei Zhang**[1,2,*]

[1] Department of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China

[2] Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing 100044, China

* **Correspondence:** Email: dwzhang@bjtu.edu.cn.

Abstract: With the continuous development of Internet of Things, finance, big data and many other fields, blockchain has been widely used in these areas for transactions, data sharing, product traceability and so on. Numerous assets have appeared in the blockchain, and there are some levels of conflicts among privacy protection of these assets, transaction transparency and auditability in blockchain; so how to provide privacy preserving, make public verifications and audit the encrypted assets are challenging problems. In this paper, we propose a privacy-preserving transaction scheme with public verification and reliable audit in blockchain. First, we provide privacy preserving of transaction contents based on homomorphic encryption. It is flexible, as we decouple user identity and transaction contents. Then, we propose and design a multiplicative zero-knowledge proof with formal security analysis. Furthermore, several verification rules are defined by us in the scheme, such as balance verification and multiplicative verification based on the proposed multiplicative zero-knowledge proof. Our scheme enables reliable and offline auditing for each transaction, and we aggregate the zero-knowledge proofs to save the ledger space. Finally, we make a security analysis of our proposal in terms of transaction confidentiality, public verification and audit reliability, and we give a performance analysis of the proposed scheme.

Keywords: blockchain; privacy-preserving; verification; zero-knowledge proof; audit

1. Introduction

Blockchain, as a type of decentralized and public computational paradigm using multi-party consensus, provides new solutions for data security and information sharing in many scenarios. Increasingly numerous assets have gradually appeared in the blockchain amid blockchain's wide application in various field such as the Internet of Things, smart grids and so on [1,2]. For example, many products' information is processed by blockchain for product traceability in the Internet of Things. Some blockchain-based data sharing schemes are also designed for sensitive information such as medical data and so on, that needs both privacy and some levels of data sharing [3–5]. Effective evaluation of privacy risk and ensuring privacy have always attracted broad attention [6–9]. In addition, many blockchain-based privacy preserving payment mechanisms for the Internet of Things have also been constructed to provide efficient and decentralized transactions [10, 11]. Therefore, how to achieve privacy of transaction contents, making monetary assets and data assets hidden from observers, and how to achieve public verification of transactions to ensure monetary assets and data assets satisfy transaction rules are crucial and have been focused on.

Traditional ledger-based transaction schemes in blockchain, such as Bitcoin, etc., lack of privacy. All transaction information, including transaction values that are permanently recorded on the blockchain is public, and it can be obtained by attackers for malicious using and spreading. Therefore, in order to hide transaction contents to make blockchain-based transactions more reliable, many cryptographic solutions have been used to offer privacy enhancing schemes in cryptocurrency which is based on the public blockchain. For example, Monero achieves hiding of transaction amounts by using Pedersen commitments. It also uses the homomorphic property of commitments and Bulletproofs to verify transactions. Zcash introduces one time encryption to protect transaction contents privacy and uses zero-knowledge Succinct Non-interactive ARgument of Knowledge (zk-SNARK) to ensure the transaction compliance. However, these solutions provide strong privacy guarantees that give users potential to circumvent regulatory controls, such as money laundering without authorities, evasion, fraud and many illicit activities that create many regulatory concerns. Enforcing reliable auditing in a blockchain-based transaction system is crucial [12], and especially in a system that offers privacy protection of transaction information, it is more challenging and essential.

Therefore, there are many challenging concerns about blockchain transaction privacy, effective auditing and public verification, as we mentioned above. More concretely, in terms of data assets such as the quantity of goods in supply chains, and sensitive information of patients in medical data sharing, many schemes do not pay attention to the public verification for data compliance while preserving privacy. For monetary assets in the unspent transaction output (UTXO) model, there is a lack of flexible transaction schemes that can both preserve privacy and achieve auditing of a transaction amount for a single transaction. How to simultaneously preserve privacy, keep a public ledger and reliably audit is challenging. Also, as there are extra leger space requirements in the UTXO model with the generation of transaction outputs and deletion of transaction inputs, how to save storage space of ledger and achieve efficiency gains for the user should be taken into consideration. Aiming to address these challenges, we focus on designing and constructing an efficient blockchain-based privacy preserving transaction scheme with public verification and reliable auditing. The main contributions of our paper are summarized as follow shows:

- We propose a privacy-preserving transaction scheme in blockchain. Our scheme offers privacy preserving both for monetary assets and data assets based on homomorphic encryption. We decoupled transaction identity information from transaction contents for the convenience of combining with different blockchain identity privacy protection schemes, which is more flexible.
- We propose and design a multiplicative zero-knowledge proof to prove the encrypted values (C_1, C_2, C_3) corresponding to (v_1, v_2, v_3) satisfy multiplicative relationship $v_1 \cdot v_2 = v_3$. It can

be widely used in blockchain based financial applications, blockchain based supply chains and many other scenarios to achieve data compliance and preserve privacy. We give formal security analysis of the proposed multiplicative zero-knowledge proof.

- We achieve public verification of hidden transaction contents based on zero-knowledge proof in our privacy preserving transaction scheme. We define several types of verification rules. For monetary assets, it achieves the balance verification relied on the signature of knowledge. For data assets, it achieves multiplicative verification by applying the proposed multiplicative zero-knowledge proof, which can also be used to save transaction computation and storage cost in the specific scenario in UTXO model.
- We also achieve reliable auditing of hidden transaction contents. In our scheme, we introduce the auditor. It can audit transaction values of each transaction instead of total transaction amounts, which is different from many existing schemes. There is also a verification of the audit zero-knowledge proof to ensure the audit reliability.
- We give formal security analysis of our blockchain-based privacy preserving transaction scheme. We also aggregate the balance proofs and audit proofs to save the ledger space. We implement the proposed scheme and evaluate its performance, and then we make a functional comparison between our scheme and others.

The rest of the paper is organized as follows. The related work is presented in Section 2. We give a brief introduction about background knowledge in Section 3. In Section 4, we present the proposed multiplicative zero-knowledge proof. We present our blockchain-based privacy-preserving transaction scheme in Section 5. Section 6 gives the security analysis of the proposed scheme. In Section 7, we give the performance analysis of the proposed scheme. Conclusions are drawn in Section 8.

2. Related work

Blockchain is a new concept that involves a consensus mechanism and distributed data storage. It was put forward as Bitcoin [13] in 2008. All transactions in Bitcoin are public and transparent. It cannot satisfy the confidentiality requirement of some applications. In 2014, Monero [14], which is a cryptocurrency deriving from Bitcoin, was proposed. It uses linkable ring signature, stealth address and RingCT to hide sensitive information of transactions such as transaction contents and user identities. Other cryptocurrencies that focus on privacy protection are Zerocash [15] and Zerocoin [16]. Zerocash leverages encryption and zk-SNARKs [17] to achieve strong privacy guarantees of transactions. Zerocoin provides strong user anonymity and coin security based on RSA accumulators and non-interactive zero-knowledge proofs. Mimblewimble [18] is also a privacy-enhancing cryptocurrency using confidential transactions [19] which is based on the Pedersen commitments [20] to hide transaction amount. Though these solutions achieve privacy protection of blockchain, neither of them satisfies the auditability, which is not compatible with illegal behaviors and is essential in financial applications.

In [21], the first distributed ledger system with auditing is proposed. In this system, commitments are used to hide transaction amount. They also provide a rough audit about the sums of transaction values. However, it needs some auditors to keep online and make queries to the system users to achieve audit, which leads auditors and all users to communicate with each other sequentially and significantly reduces the efficiency. In [22], the authors achieve an advance zero-knowledge ledger by proposing an efficient range-proof technique based on the improved inner product based zero-knowledge proofs. The reducing of proof size greatly improves the system efficiency. In [23], a private, authenticated and auditable blockchain is proposed. It achieves privacy protection and auditability in terms of user identity and transaction contents based on additive homomorphic encryption and BBS group signature. In [24], the authors propose a decentralized system framework using the blockchain and IPFS system to provide high security for sharing and exchanging the multimedia file system. They use the secure authentication protocol which is based on zero-knowledge proofs to guarantee multimedia data user privacy. In [25], the authors achieve anonymity of users and privacy of transaction amount. As for regulation, the system can regulate the total amount of transactions in a certain time. Also, there are some auditable solutions based on the account model [26–28].

Table 1. Functional comparison between our scheme and others.

Scheme	TM	TC	BV	MV	DIC	AR	AoET
[14]	UTXO	Yes	Yes	No	No	No	No
[15]	UTXO	Yes	Yes	No	No	No	No
[18]	UTXO	Yes	Yes	No	Yes	No	No
[23]	UTXO	Yes	Yes	No	No	Yes	Yes
[25]	UTXO	Yes	Yes	No	No	No	No
Ours	UTXO, data assets	Yes	Yes	Yes	Yes	Yes	Yes

We give the analysis and functional comparison between our scheme and other comparable schemes in Table 1 in aspects of transaction model (TM), transaction confidentiality (TC), balance verification (BV), multiplicative verification decoupled user identity and transaction contents (DIC), audit reliability (AR) and audit of each transaction (AoET). In summary, as we can see in Table 1, the above papers provide various privacy protections in terms of both identity and transaction contents, and they rarely achieve precise auditing of transactions, which is essential in financial applications. In particular, they mainly focus on transfer transactions, as blockchain has been widely applied in supply chains, data sharing and many other fields; and it is also quite necessary to provide efficient verifications for those scenarios with both monetary assets and data assets, which has been ignored.

3. Preliminaries

In this section, we introduce some related techniques that are used in this paper.

3.1. UTXO model

At present, there are many decentralized payment systems, such as Bitcoin, RSCoin [29], Fabcoin in Hyperledger fabric [30] and so on, that are based on the UTXO model, in which each transaction is formed by a set of inputs and a set of outputs. It is different from the traditional account model used by Ethereum, where the transaction value is specified and moved from one account to another. The UTXO model is shown in Figure 1. It represents some amount of monetary assets that have been authorized by one user to be spent by another. Details of monetary assets' flowS in transactions with the UTXO model are recorded in the blockchain ledger.

3.2. Pedersen commitment

Pedersen commitment is used to achieve transaction confidentiality in Bitcoin. It can be described as follows.

- $setup(1^{\lambda})$: This algorithm takes the security parameter λ as input, and it generates the cyclic group $\mathbb{G}$ with q order. G is the generator of group $\mathbb{G}$. H is the random element of $\mathbb{G}$. It outputs the public parameter $pp = \{\mathbb{G}, G, H, q\}$.
- $Cm(pp, v)$: This algorithm takes the public parameter pp, commitment c, the value v and the blind element r as input. It computes $c = rG + vH$ as the commitment of v.
- $Open(pp, c, v, r)$: This algorithm takes the public parameter pp, commitment c, the value v and the blind element r as input. It checks whether $c = rG + vH$ holds or not.

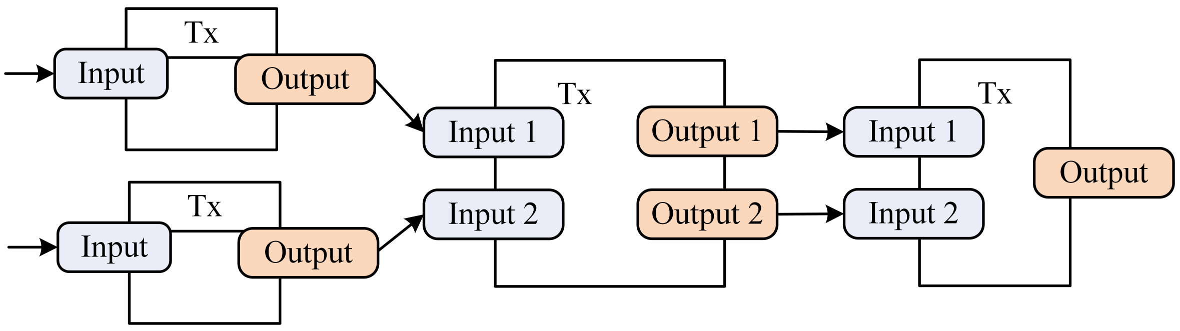

Figure 1. UTXO model

3.3. Hard problems and complexity assumptions

Definition 1. ***(Discrete logarithm (DL) problem)****. Let $\mathbb{G}$ be a cyclic group. Given a random instance (P, aP), where $P \in \mathbb{G}$, and $a \in \mathbb{Z}_p^*$, computation of a is computationally hard by a polynomial time algorithm. The probability that a polynomial time algorithm A can solve the DL problem is defined as $Adv_A^{DL}(\lambda)$.*

Definition 2. ***(Discrete logarithm assumption)****. For any probabilistic polynomial time algorithm A, $Adv_A^{DL}(\lambda)$ is negligible; that is, $Adv_A^{DL}(\lambda) \leq \epsilon$, for some negligible function ϵ.*

3.4. A variant of ElGamal encryption

There is a homomorphic encryption based on ElGamal encryption called twisted ElGamal [28], which is zero-knowledge friendly. Given a cyclic group $\mathbb{G}$ with order q, let P and H be two random generators of $\mathbb{G}$. So, $pp = \{\mathbb{G}, P, H, q\}$. Then, it consists of the following algorithms:

keygen: It takes pp as input and randomly chooses $x \in \mathbb{Z}_q^*$ as secret key. It computes public key $Y = xP$, and then it outputs (X, Y).

enc: It takes the public key Y and message m as input. It randomly chooses $s \in \mathbb{Z}_q^*$, computes $C_1 = sP$, $C_2 = mH + sP$ and outputs $C = \{C_1, C_2\}$.

dec: It takes the ciphtertext C and secret key as input. It computes $mH = C_2 - x^{-1} \cdot C_1$ to obtain m.

3.5. *Non-interactive zero-knowledge proof*

A non-interactive zero-knowledge (NIZK) proof [31] is a protocol that the prover can use to convince the verifier that it indeed has the knowledge of a secret value by some public information without revealing the secret value. The non-interactive zero-knowledge proof has properties of completeness, soundness, and zero-knowledge [32]. We introduce a non-interactive zero-knowledge proof that is the signature of knowledge of the discrete logarithm (SKDL) [33,34]. Let $\mathbb{G}$ be a cyclic group. $P, G \in \mathbb{G}$. A pair $(c, s) \in \{0, 1\}^k \times \mathbb{Z}_n^*$ satisfying $c = H_0(P, Y, sP + cY)$ is a signature of the knowledge of the discrete logarithm of $Y \in \mathbb{G}$ to the base P. It is denoted as $SKDL\{(a) \mid Y = aP\}$. It is as follows:

(1) The prover randomly chooses $r \in \mathbb{Z}_q^*$, then it computes $T = rP$, $c = H_0(P, Y, T)$ and $s = r - ca$. The prover sends (c, s) to the verifier.
(2) The verifier verifies whether $c = H_0(P, Y, sP + cY)$ holds. If the equation holds, it means that the prover knows the knowledge of the discrete logarithm of Y to the base P.

4. Proposed multiplicative zero-knowledge proof

Our proposed multiplicative zero-knowledge proof aims to convince the verifier that v_3 encrypted in C_3 is actually the product of v_1 and v_2, encrypted respectively in C_1 and C_2, i.e., $v_1 \cdot v_2 = v_3$. It mainly contains three steps that are as follows:

setup: Let $\mathbb{G}$ be a cyclic group with q order, where q is λ bits. P and H are two random generators of $\mathbb{G}$. Then, the public parameter is $pp = \{\mathbb{G}, P, H, q\}$.

prove: The prover randomly chooses $s_1, s_2, s_3 \in \mathbb{Z}_q^*$, and then it computes $C_1 = v_1H + s_1P$, $C_2 = v_2H + s_2P$ and $C_3 = v_3H + s_3P$. The prover randomly chooses $y_1, y_2, y_3, s'_1, s'_2, s'_3 \in \mathbb{Z}_q^*$, and then it computes $d_1 = y_1H + s'_1P$, $d_2 = y_2H + s'_2P$, $d_3 = y_3H + s'_3P$ and $d_4 = y_2C_{21} + s'_4P$. The prover sends the generated C_1,C_2,C_3,d_1,d_2,d_3,d_4 to the verifier. The verifier randomly chooses a challenge $c \in \mathbb{Z}_q^*$ and returns it to the prover. Then, the prover computes $u_1 = y_1 + v_1c$, $u_2 = y_2 + v_2c$, $u_3 = y_3 + v_3c$, $\theta_1 = s'_1 + s_1c$, $\theta_2 = s'_2 + s_2c$, $\theta_3 = s'_3 + s_3c$ and $\theta_4 = s'_4 + (s_3 - s_1v_2)c$. The prover sends the generated u_1,u_2,u_3,θ_1,θ_2,θ_3,θ_4 to the verifier.

verify: The verifier computes $d'_1 = \theta_1P + u_1H - cC_1$, $d'_2 = \theta_2P + u_2H - cC_2$,$d'_3 = \theta_3P + u_3H - cC_3$, $d'_4 = \theta_4P + u_2C_1 - cC_3$, and then it checks whether $d'_1 = d_1$, $d'_2 = d_2$, $d'_3 = d_3$ and $d'_4 = d_4$ holds. If the above equations hold, it outputs 1. Otherwise, it outputs 0.

According to the above steps, the prover proves that C_1, C_2, C_3 are encrypted values of v_1, v_2, v_3 satisfying $v_1 \cdot v_2 = v_3$. In addition, the above proof can turn to be non-interactive by applying the Fiat-Shamir heuristic [35]. Particularly, there are some applications in blockchain for the proposed multiplicative zero-knowledge proof to be used in variants of scenarios, no matter for monetary assets and data assets. We give explanations about it in Section 7.

Theorem 1. *The proposed multiplicative proof is a zero-knowledge proof under the Discrete logarithm assumption, which means that it satisfies correctness, zero knowledge (can be simulated) and a proof of knowledge (has an extractor).*

We prove it through Lemmas 1–3.

Lemma 1. *The proposed multiplicative zero-knowledge proof satisfies correctness.*

Proof of Lemma 1. If the prover follows the computation steps specified for it, we have the following.

$$\begin{aligned} d'_1 &= (s'_1 + s_1c)P + (y_1 + v_1c)H - c(v_1H + s_1P) \qquad (4.1)\\ &= y_1H + s'_1P = d_1 \\ d'_2 &= (s'_2 + s_2c)P + (y_2 + v_2c)H - c(v_2H + s_2P) \qquad (4.2)\\ &= y_2H + s'_2P = d_2 \\ d'_3 &= (s'_3 + s_3c)P + (y_3 + v_3c)H - c(v_3H + s_3P) \qquad (4.3)\\ &= y_3H + s'_3P = d_3 \\ d'_4 &= y_2C_1 + v_2cC_1 + (s'_4 + (s_3 - s_1v_2))P - c(v_3H + s_3P) \qquad (4.4)\\ &= y_2C_1 + s'_4P + (v_1v_2cH - v_3cH) \\ &+ (v_2s_1cP - v_2s_1cP) + (s_3cP - s_3cP) = d_4 \end{aligned}$$

As we can see from the above equations, Eqs (4.1)–(4.4) hold. Therefore, the verifier always accepts the proof, and then the proposed multiplicative zero-knowledge proof satisfies correctness.

Lemma 2. *The proposed multiplicative zero-knowledge proof can be simulated under the Discrete logarithm assumption.*

Proof of Lemma 2. We describe a simulator that can outputs the proof. It randomly chooses a set of values v_1, v_2, v_3 and computes $C_1 = v_1H + s_1P$, $C_2 = v_2H + s_2P$, $C_3 = v_3H + s_3P$. The distribution of these values generated by the simulator is indistinguishable from the distribution output by the prover. In the remainder of the simulation, it does not assume knowledge of v_1, v_2, v_3.

The simulator randomly chooses a challenge $c \in \mathbb{Z}_q^*$ and u_1, u_2, u_3, θ_1, θ_2, θ_3, θ_4. It computes $d_1 = \theta_1P + u_1H - cC_1$, $d_2 = \theta_2P + u_2H - cC_2$, $d_3 = \theta_3P + u_3H - cC_3$ and $d_4 = u_2C_1 + \theta_4P - cC_3$ that satisfy Eqs (4.1)–(4.4). Moreover, these values have the same distribution as those in the real proof. The simulator outputs c,u_1,u_2,u_3,θ_1,θ_2,θ_3,θ_4,d_1,d_2,d_3,d_4 that are indistinguishable from the real proof in the multiplicative proof. Therefore, the proposed multiplicative zero-knowledge proof can be simulated under the Discrete logarithm assumption.

Lemma 3. *The proposed multiplicative zero-knowledge proof has an extractor.*

Proof of Lemma 3. Suppose there exits an extractor that enables one to rewind a prover in the multiplicative proof we proposed above to the point before it generates c. To the challenge value c, there is $(u_1, u_2, u_3, \theta_1, \theta_2, \theta_3, \theta_4)$. For challenge value $c' \neq c$, the prover responds with $(u'_1, u'_2, u'_3, \theta'_1, \theta'_2, \theta'_3, \theta'_4)$. If the prover is convincing, then all Eqs (4.1)–(4.4) hold.

So, we have $\Delta c = c - c'$, $\Delta u_1 = u_1 - u'_1$, and Δu_2,Δu_3,$\Delta\theta_1$,$\Delta\theta_2$,$\Delta\theta_3$,$\Delta\theta_4$ are similar with Δu_1. Considering Eq (4.1), we have $\Delta cC_1 = \Delta\theta_1P + \Delta u_1H$, so let $v_1^* = \Delta u_1/\Delta c$ and let $s_1^* = \Delta\theta_1/\Delta c$. Similarly, from Eqs (4.2)–(4.4), we obtain v_2^*, s_2^*,v_3^*, s_3^* and $s^* = \Delta\theta_4/\Delta c$. We have $(v_1^*v_2^* - v_3^*)H = (s_3^* - s^* - v_2^*s_1^*)P$. Therefore, the extractor obtains a Discrete logarithm problem solution $log_PH = (s_3^* - s^* - v_2^*s_1^*)/(v_1^*v_2^* - v_3^*)$. Therefore, the proposed multiplicative zero-knowledge proof has an extractor.

5. Proposed blockchain-based privacy-preserving transaction scheme

5.1. Overview

We propose a blockchain-based transaction scheme with privacy-preserving that enables reliable auditing and different verification rules. There are four roles in our scheme that are described as follows:

- Trusted Center: It initializes the whole scheme.
- Users: It includes payer and payee that involves in the blockchain based transactions. It also contains users that transact, share and store data assets through blockchain.
- Validator: It verifies whether proposed encrypted transactions satisfy verification rules.
- Auditor: It audits encrypted transactions in the scheme.

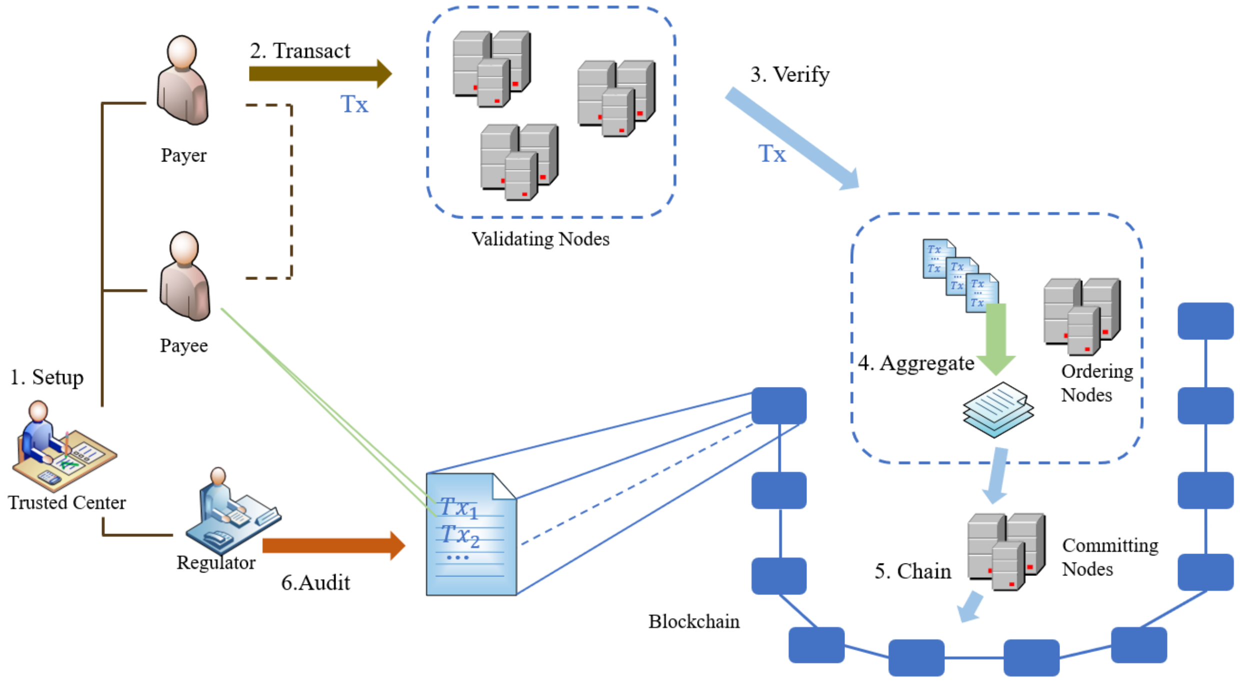

Figure 2. Overview of our scheme.

As we can see in Figure 3, the transaction overflow of our privacy preserving transaction scheme is summarized as follows:

(1) Setup: The trusted center makes an initialization and generates an audit key pair for auditor.
(2) Transact: Users generate transactions, and they send transactions to validators.
(3) Verify: Validators receive transaction and verify whether it satisfies verification rules and audit reliability.
(4) Aggregate: Balance and audit zero-knowledge proofs in transaction are aggregated and sent to committing nodes.
(5) Chain: committing nodes make verifications of the aggregated information. If they pass verifications, transactions are committed to the blockchain.

(6) Audit: The auditor audit transaction contents. It does not need to be online all the time and can achieves audit transaction contents of each transaction.

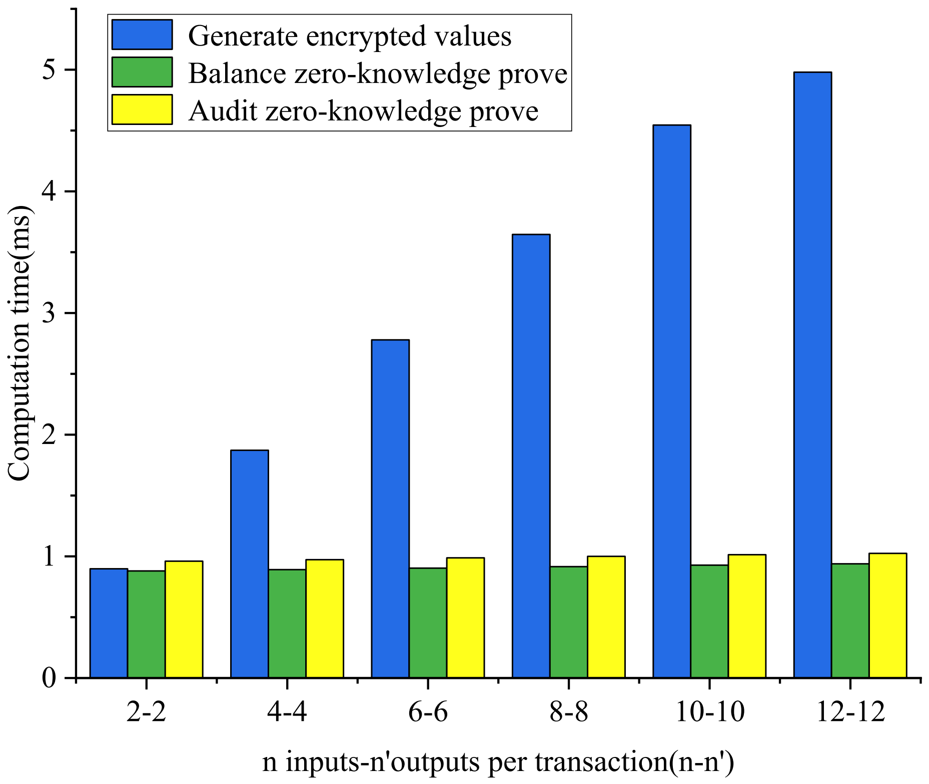

Figure 3. Computation time comparison in transact phase with increasing inputs and outputs.

Notations in our paper are summarized in Table 2. In our scheme, transaction tx is used to record the encrypted payment process between payers and payees for monetary assets, and it is used to record the encrypted data transaction for data assets. Transactions are finally recorded in the ledger of the blockchain. The structure of transaction tx is $tx = \{tx.in, tx.out, tx.data, \pi_{bl}, \pi_{rp}, \pi_{pro}, \pi_{au}\}$. $tx.in$ is the encrypted inputs of the transaction, and $tx.out$ is the encrypted outputs of the transaction. $tx.data$ is the encrypted data of data assets. π_{bl} is the balance proof generated by users for balance verification. π_{rp} is the range proof to prove the transaction value is in a certain range $[0, v_{max}]$, where v_{max} is a system parameter. π_{pro} is the multiplicative proof that can prove transaction values satisfy product relationship, and π_{au} is the audit proof to prove the auditor can reliably audit the transaction.

More concretely, $tx.in$ includes n inputs of a transaction such that $tx.in = \{C_i^{in} \mid C_i^{in} = \{C_{1i}^{in}, C_{2i}^{in}\}, i \in [1, n]\}$. The value of each input C_i^{in} is v_i^{in}. $tx.out$ includes n' outputs of a transaction and the change C_c, which can be presented as $tx.out = \{C_j^{out}, C_c \mid C_j^{out} = \{C_{1j}^{out}, C_{2j}^{out}\}, j \in [1, n'], C_c = \{C_{1c}, C_{2c}\}$. The value of each output C_j^{out} is v_j^{out}, and the change value is v_c. $tx.out$ includes encrypted data $tx.data = \{C_1 = \{C_{11}, C_{21}\}, C_2 = \{C_{12}, C_{22}\}, C_3 = \{C_{13}, C_{23}\}, ...\}$, where C_1, C_2, C_3 are encrypted data of some values v_1, v_2, v_3.

Table 2. Notations.

Symbols	Descriptions
λ	Security parameter
pp	Public parameters
$\mathbb{G}$	A cyclic group
tx	Transaction
$tx.in$	Transaction encrypted inputs
$tx.out$	Transaction encrypted outputs
$tx.data$	Transaction encrypted data
C_i^{in}, C_j^{out}	Encrypted inputs and outputs
C_1, C_2, C_3	Encrypted data assets

5.2. Security model

Our scheme is designed to satisfy the security requirements of transaction confidentiality, public verification and audit reliability.

Definition 3. *(Transaction confidentiality). Transaction confidentiality means the plaintext of transaction contents such as payment value or data assets cannot be obtained by an attacker in our system.*

We define the transaction confidentiality of our scheme by the following transaction confidentiality experiment. The adversary $\mathcal{A}$ is a user in the system, and it has the UTXO that belongs to him.

$$|Pr\left[b = b' : \begin{array}{c} pp \leftarrow setup(1^\lambda); \\ (X, Y) \leftarrow keygen(pp); \\ (\{ptx.rmdr_0, ptx.rmdr_1\}) \leftarrow \mathcal{A}_1(pp, Y); \\ b \overset{R}{\leftarrow} \{0,1\}; \\ tx.out^* \leftarrow tx(pp, ptx.rmdr.Y) \\ \pi_{au}^* \leftarrow au(pp, ptx.out, \pi_{pau}, Y); \\ b' \leftarrow \mathcal{A}_2^{O}(tx.out^*, \pi_{au}^*) \end{array} \right] - \frac{1}{2}| \leq negl(\lambda),$$

in which the definitions of the oracles O_{pre} and O_{GenCT} are as follows:

- O_{pre}: On input $((C_i^{in}, v_i^{in}, s_i^{in}), v_\rho)$, run $ptx \leftarrow pretx(pp, C_i^{in}, v_i^{in}, s_i^{in}, v_\rho, Y)$ and store $\{(C_i^{in}, v_i^{in}, s_i^{in}), v_\rho, Y, ptx\}$ into the list L.
- O_{GenCT}: On input $(ptx.rmdr)$, search L, run $tx.out \leftarrow tx(pp, ptx.rmdr, Y)$ and $\pi_{au} \leftarrow au(pp, ptx.out, \pi_{pau}, Y)$, and then return $tx.out$ and π_{au}.

Public verification means that transactions in our scheme can be publicly verified by validators to satisfy various verification rules. We design two types of verification rules, and they are transaction balance and transaction multiplicative relationship that are defined as follows.

Definition 4. *(Transaction balance). For monetary assets, it satisfies balance verification such that the sum of inputs' values is equal to the sum of outputs' values.*

We define the transaction balance of our scheme by the following transaction balance experiment. The adversary $\mathcal{A}$ is a user in the system, and it has the UTXO that belongs to him.

$$Pr\left[\begin{array}{cc} & pp \leftarrow setup(1^\lambda); \\ veribl(pp,\pi_{bl}) = 1 \wedge & (X,Y) \leftarrow keygen(pp); \\ \sum_1^n v_i^{in} \neq \sum_{j=1}^{n'} v_j^{out} + v_c^{out}: & (tx.in, tx.out, v_i^{in}, \pi_{bl}) \leftarrow \mathcal{A}^O(pp,Y) \end{array}\right] \leq negl(\lambda),$$

in which the definitions of the oracles O_{pre} and O_{bal} are as follows:

- O_{pre}: On input $((C_i^{in}, v_i^{in}, s_i^{in}), v_\rho)$, run $ptx \leftarrow pretx(pp, C_i^{in}, v_i^{in}, s_i^{in}, v_\rho, Y)$ and store $\{(C_i^{in}, v_i^{in}, s_i^{in}), v_\rho, Y, ptx\}$ into the list L.
- O_{bal}: On input $ptx.rmdr$, run $tx.out \leftarrow tx(pp, ptx.rmdr, Y)$, search L to find the corresponding π_{pbp} and P_b, then run $\pi_{bl} \leftarrow bl(pp, \pi_{pbp}, P_b)$, and return $tx.out$ and π_{bl}.

Definition 5. *(Transaction multiplicative relationship). For data assets, the validator can publicly verify whether some values v_1, v_2, v_3 satisfy multiplicative relationship such as $v_1 \cdot v_2 = v_3$.*

We define the transaction multiplicative relationship of our scheme by the following transaction multiplicative relationship experiment. The adversary $\mathcal{A}$ is a user in the system.

$$Pr\left[\begin{array}{cc} & pp \leftarrow setup(1^\lambda); \\ veripro(pp,\pi_{pro}, C_1, C_2, C_3) = 1 & (X,Y) \leftarrow keygen(pp); \\ \wedge v_3 \neq v_1 \cdot v_2: & (v_1, v_2, v_3, C_1, C_2, C_3, \pi_{pro}) \leftarrow \mathcal{A}^O(pp,Y) \end{array}\right] \leq negl(\lambda),$$

in which the definitions of the oracles O_{pro} are as follows:

- O_{pro}: On input v_1, v_2, v_3, run $(C_1, C_2, C_3) \leftarrow tx(pp, v_1, v_2, v_3, Y)$ and $\pi_{pro} \leftarrow pro(pp, v_1, v_2, v_3, C_{21}, C_{22}, C_{23})$, and return C_1, C_2, C_3 and π_{pro}.

Definition 6. *(Audit reliability). Audit reliability means they can be reliably audited by the auditor.*

We define the audit reliability of our scheme by the following audit reliability experiment. The adversary $\mathcal{A}$ is a user in the system and it has the UTXO that belongs to him.

$$Pr\left[\begin{array}{cc} & pp \leftarrow setup(1^\lambda); \\ veriau(pp,\pi_{au}, C_{forge}) = 1: & (C_{forge}) \leftarrow \mathcal{A}(pp, v_j^{f,out}, Y^f); \end{array}\right] \leq negl(\lambda)$$

5.3. Description of the proposed scheme

It consists of six phases, including Setup, Transact, Verify, Aggregate, Chain and Audit.

Setup: In the setup phase, the trusted center generates public parameters and audit key pair. First, it executes the **setup**(1^λ) algorithm, where λ is the security parameter. $\mathbb{G}$ is a cyclic group which is q order, where q is λ bits. P and H are two random generators of $\mathbb{G}$. H_0, H_1, H_2 and H_3 are hash functions that satisfy $H_0 := \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_q$, $H_1 := \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_q$, $H_2 : \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_q$, $H_3 := \mathbb{G} \times \underset{2n'+2}{......} \times \mathbb{G} \rightarrow \mathbb{Z}_q$. Second, it executes the **keygen**(pp) algorithm. It randomly chooses $x \in \mathbb{Z}_q$ as the audit secret key X, and then it computes the audit public key $Y = x \cdot P$. At last, the trusted center outputs the audit public key Y and the public parameters $pp = \{\mathbb{G}, P, H, q, H_0, H_1, H_2, H_3\}$.

Transact: In the transact phase, the payee and the payer generate transaction that preserves privacy of the transaction contents that can be audited by the auditor. In addition, they also generate proofs to ensure the transaction satisfy verification rules and reliable audit. In this phase, they provide balance proof that ensures the sum of outputs is equal to the sum of inputs, range proof that ensures the transaction value is greater than zero, multiplicative proof that ensures that some transaction data satisfies the multiplicative relationship and audit proof that guarantees the audit reliability. In this phase, there are five algorithms that are described as follows:

(1) The $\mathbf{pretx}(pp, C_i^{in}, v_i^{in}, s_i^{in}, v_\rho, Y)$ algorithm is executed by the payer. It takes as input the public parameters pp, transaction inputs C_i^{in}, value v_i^{in}, randomness s_i^{in}, transfer value v_ρ and the audit public key Y. It outputs the pre-transaction ptx as the following shows:

- The payer selects n inputs C_i^{in} of total value $v = \sum_{i=1}^{n} v_i^{in} \geq v_\rho$. Let pre-transaction input be $ptx.in = \{C_i^{in} \mid i \in [1, n]\}$. It generates n' outputs of total value $v_\rho = \sum_{j=1}^{n'} v_j^{out}$. Let the pre-transaction remainder be $ptx.rmdr = \{v_j^{out} \mid j \in [1, n']\}$.
- The payer computes the change value $v_c^{out} = v - v_\rho$. Let the change value be $ptx.chg = v_c^{out}$. It randomly selects randomness of the change value $s_c^{out} \in \mathbb{Z}_q$. It computes $C_{1c}^{out} = s_c^{out} Y$ and $C_{2c}^{out} = s_c^{out} P + v_c^{out} H$. Let $C_c^{out} = \{C_{1c}^{out}, C_{2c}^{out}\}$, and it stores C_c^{out} in $tx.out$.
- The payer generates the pre-transaction balance proof π_{pbp}. It randomly chooses $r_a \in \mathbb{Z}_q$ and computes $s_s^{in} = -\sum_{i=1}^{n} s_i^{in} + s_c^{out}$. It computes $X_a = s_s^{in} P$, $R_a = r_a P$, $e_a = H_0(R_a, X_a)$ and $\sigma_a = r_a + e s_s^{in}$. So, the pre-transaction balance proof $\pi_{pbp} = \{\sigma_a, e_a, R_a, X_a\}$.
- The payer computes the pre-transaction audit proof π_{pau}. The proof can be described as $SKDL\{(v_c^{out}, s_c^{out}) : C_{1c}^{out} = s_c^{out} Y \wedge C_{2c}^{out} = s_c^{out} P + v_c^{out} H\}$, which ensures that this transaction can be reliably audited. It randomly chooses $s_c^{out'} \in \mathbb{Z}_q$ and $v_c^{out'} \in \mathbb{Z}_q$, then it computes $R_{1c} = s_c^{out'} Y$, $R_{2c} = s_c^{out'} P + v_c^{out'} H$, $\tilde{c}_p = H_1(R_{1c}, R_{2c}, C_c^{out})$, $\sigma_{c,1} = s_c^{out'} + \tilde{c}_p s_c^{out}$ and $\sigma_{c,2} = v_c^{out'} + \tilde{c}_p v_c^{out}$. So the pre-transaction audit proof is $\pi_{pau} = \{\sigma_{c,1}, \sigma_{c,2}, R_{1c}, R_{2c}, \tilde{c}_p\}$.

The payer outputs the generated pre-transaction $ptx = \{ptx.in, ptx.out, \pi_{pbp}, \pi_{pau}\}$, where $ptx.out = \{ptx.chg, ptx.rmdr\}$.

(2) The $\mathbf{tx}(pp, ptx.rmdr, Y)$ algorithm is executed by the payee. It takes as input the public parameters pp, pre-transaction remainder $ptx.rmdr$ and the audit public key Y. It generates the transaction outputs $tx.out$, balance randomness P_b and range proof π_{rp} as the following shows: The payee checks whether $\sum_{i=1}^{n} v_i^{in} = \sum_{j=1}^{n'} v_j^{out} + v_c^{out}$ holds. If it does not hold, it aborts. Otherwise, the payee executes the $\mathbf{txenc}(pp, v_i^{in}, Y)$ algorithm, which is twisted ElGamal encryption. This algorithm randomly chooses $s_j^{out} \in \mathbb{Z}_q$ and computes $C_{1j}^{out} = s_j^{out} Y$ and $C_{2j}^{out} = s_j^{out} P + v_j^{out} H$, and then it stores them to $tx.out$. The payee computes $s_s^{out} = \sum_{j=1}^{n'} s_j^{out}$ and the balance randomness $P_b = s_s^{out} P$, and then the payee executes the Bulletproofs [36] to generate range proof $\pi_{rp} = \{\pi_{rp_c}, \pi_{rp_j} \mid j \in [1, n']\}$. For data assets such as $v_1, v_2, v_3 (v_3 = v_1 v_2)$, it generates C_1, C_2, C_3 by $\mathbf{txenc}(pp, v_1, v_2, v_3, Y)$ in the same way, and it stores them in $tx.data = \{C_1, C_2, C_3\}$.

(3) The $\mathbf{bl}(pp, \pi_{pbp}, P_b)$ algorithm is executed by the payer and payee. It takes as input the public parameters pp, pre-transaction balance proof π_{pbp} and balance randomness P_b. It generates balance proof π_{bl} as the following shows:

- The payee computes $e_a' = H_0(R_a, X_a)$, and then it verifies whether $\sigma_a P = R_a + e_a' X_a$ holds. If it does not hold, the payee aborts. Otherwise, the payee randomly chooses $r_b \in \mathbb{Z}_a$, computes $R_b = r_b P$, $\triangle R = R_a + R_b$ and $\bar{X} = X_a + P_b$. It calculates $e = H_0(\triangle R, \bar{X})$ and computes $\sigma_B = r_b + e s_s^{out}$. The payee sends these generated σ_B and P_b to the payer.
- The payer computes $\triangle R = R_a + R_b$, $\bar{X} = X_a + P_b = x_s P$, $e = H_0(\triangle R, \bar{X})$, $\sigma_A = r_a + e s_s^{in}$ and $\sigma = \sigma_A + \sigma_B$. Therefore, the generated balance proof is $\pi_{bl} = \{\sigma, e, \triangle R, \bar{X}\}$.

(4) The $\mathbf{pro}(pp, v_1, v_2, v_3, C_{21}, C_{22}, C_{23})$ algorithm is executed by the user. It proves that some encrypted transaction values v_1, v_2, v_3 satisfy the product relationship $v_1 v_2 = v_3$. It takes as input the

public parameters pp, $C_{21} = v_1H + s_1P$, $C_{22} = v_2H + s_2P$ and $C_{23} = v_3H + s_3P$ that are encrypted values of v_1, v_2, v_3. It generates multiplicative proof π_{pro} as the following shows:

- The user randomly chooses $y_1, y_2, y_3, s'_1, s'_2, s'_3 \in \mathbb{Z}_q$, and then it computes $d_1 = y_1H + s'_1P$, $d_2 = y_2H + s'_2P$, $d_3 = y_3H + s'_3P$ and $d_4 = y_2C_{21} + s'_3H$. It computes $c = H_2(d_1, d_2, d_3, d_4, C_{21}, C_{22}, C_{23})$.
- It computes $u_1 = y_1 + v_1c$, $u_2 = y_2 + v_2c$, $u_3 = y_3 + v_3c$, $\theta_1 = s'_1 + s_1c$, $\theta_2 = s'_2 + s_2c$, $\theta_3 = s'_3 + s_3c$ and $\theta_4 = s'_3 + (s_3 - s_1v_2)c$. So, the multiplicative proof π_{pro} is $\pi_{pro} = \{c, u_1, u_2, u_3, \theta_1, \theta_2, \theta_3, \theta_4\}$.

(5) The **au**$(pp, ptx.out, \pi_{pau}, Y)$ algorithm is run by the payee. It takes as input public parameters pp, a remainder $ptx.rmdr$, the pre-transaction audit proof π_{pau} and the audit public key Y. It outputs the audit proof π_{au} as the following shows:

- The payee randomly chooses $s_j^{out'} \in \mathbb{Z}_q$ and computes $R_1 = R_{1c} + \sum_{j=1}^{n'} R_{1j} = R_{1c} + \sum_{1j}^{n'} s_j^{out'} Y$, and then it randomly selects $v_j^{out'} \in \mathbb{Z}_q$ and computes $R_2 = R_{2c} + \sum_{2j}^{n'} R_{2j} = R_{2c} + \sum_{2j}^{n'}(s_j^{out'}P + v_j^{out'}H)$.
- It calculates $\tilde{c} = H_3(R_1, R_2, tx.out)$ and $\sigma_{j,1} = s_j^{out'} + \tilde{c}s_j^{out}$, $\sigma_{j,2} = v_j^{out'} + \tilde{c}v_j^{out}$, where v_j^{out} is the output value, and s_j^{out} is the random number.
- It computes $\bar{\sigma} = \sigma_{c,1} + \sum_{j=1}^{n'} \sigma_{j,1}$ and $\sigma' = \sigma_{c,2} + \sum_{j=2}^{n'} \sigma_{j,2}$. So, the audit proof π_{au} is $\pi_{au} = \{\bar{\sigma}, \sigma', R_1, R_2, \tilde{c}\}$.

Finally, the payee sends the transaction to the validating nodes.

Verify: In the verify phase, validating nodes are responsible for verifying whether the transaction meets some requirements that we defined. There are four verifying algorithms that are described as the following shows:

(1) The **verirp**$(pp, tx.out, \pi_{rp})$ algorithm takes as input the public parameters pp, transaction output $tx.out$ and the range proof π_{rp}. It uses the Bulletproofs [36] to verify whether the transaction output is in a certain range $[0, v_{max}]$. The detailed Bulletproofs can be seen in [36].

(2) The **veribl**(pp, π_{bl}) algorithm takes as input the public parameters pp and balance proof π_{bl}. It verifies whether the transaction satisfies the balance property as the following shows: It computes $e' = H_0(\triangle\ R, \bar{X})$, and then it checks whether $e' = e$ and $\sigma P = \triangle R + e\bar{X}$ hold. If they hold, it outputs true which means that the transaction satisfies balance property.

(3) The **veripro**(pp, π_{pro}) algorithm takes as input the public parameters pp and the multiplicative proof π_{pro}. It verifies whether these encrypted transaction values satisfy product relationship $v_1v_2 = v_3$. It computes $d'_1 = \theta_1P + u_1H - cC_{21}$, $d'_2 = \theta_2P + u_2H - cC_{22}$,$d'_3 = \theta_3P + u_3H - cC_{23}$, $d'_4 = \theta_4P + u_2C_{21} - cC_{23}$ and $c' = H_2(d'_1, d'_2, d'_3, d'_4, C_{21}, C_{22}, C_{23})$, and then it checks whether $c' = c$ holds. If it holds, it outputs true which means that these encrypted transaction values satisfy product relationship.

(4) The **veriau**(pp, π_{au}) algorithm takes as input the public parameters pp and audit proof π_{au}. It verifies whether the transaction can be reliably audited as the following shows: It computes $R'_1 = \bar{\sigma}Y - \tilde{c}C_{1c}^{out} - \sum_{j=1}^{n'} \tilde{c}C_{1j}^{out}$, $R'_2 = \sigma'H + \bar{\sigma}P - \tilde{c}C_{2c}^{out} - \sum_{j=1}^{n'} \tilde{c}C_{2j}^{out}$ and $\tilde{c}' = H_3(R'_1, R'_2, tx.out)$. It checks whether $\tilde{c} = \tilde{c}'$ holds. If this equation holds, it outputs true, which means that the transaction can be reliably audited.

Aggregate$(\sigma_k, \triangle\ R, \sigma'_k, \bar{\sigma}_k, R_{1k}, R_{2k})$: In the aggregate phase, the ordering nodes takes as input the balance signature σ_k, balance randomness $\triangle\ R$, audit signature $\sigma'_k, \bar{\sigma}_k$, and audit randomness R_{1k}, R_{2k}, it aggregates m transactions' balance signature and audit signature, where $k \in m$. The ordering nodes compute $\sigma_{Agg} = \sum_1^m \sigma_k$, $R_{Agg} = \sum_1^m \vartriangle R_k$, $\sigma'_{Agg} = \sum_1^m \sigma'$, $\bar{\sigma}_{Agg} = \sum_1^m \bar{\sigma}_k$, $R^1_{Agg} = \sum_1^m R_{1k}$ and $R^2_{Agg} = \sum_1^m R_{2k}$. Therefore, the aggregated message is $info_{Agg} = \{\sigma_{Agg}, R_{Agg}, \sigma'_{Agg}, \bar{\sigma}_{Agg}, R^1_{Agg}, R^2_{Agg}\}$.

Chain$(info_{Agg}, \bar{X}_k, tx.out_k, e_k, \tilde{c}_k)$: In the chain phase, the committing nodes take as input the aggregated message $info_{Agg}$, public randomness $\bar{X}_k$, transaction outputs $tx.out_k$, hash value e_k corresponding to each transaction and balance challenge value $\tilde{c}_k$. They verify the correctness of the aggregated message $info_{Agg}$ by checking whether $\sigma_{Agg}P = R_{Agg} + \sum_k e_k\bar{X}_k$, $\bar{\sigma}_{Agg}P = R^1_{Agg} + \tilde{c}_k C^{out}_{1c} + \sum_{j=1}^{n'} \tilde{c}_k C^{out}_{1j}$ and $\sigma'_{Agg}H + \bar{\sigma}_{Agg}P = R^2_{Agg} + \tilde{c}_k C^{out}_{2c} + \sum_{j=1}^{n'} \tilde{c}_k C^{out}_{2j}$ hold. If these two equations hold, it outputs true, then committing nodes add transactions that have been verified onto the ledger and the updated ledger is Λ.

Audit$(pp, X, tx.out)$: In the audit phase, the auditor takes as input the public parameters pp, audit secret key X and transaction outputs $tx.out$, and it computes $v^{out}_j H = C^{out}_{2j} - X^{-1}\dot{C}^{out}_{1j}$ and auditing transaction by comparing $v^{out}_j H$ with the pre-computed bH, where $b \in [0, v_{max})$.

6. Security analysis

6.1. Transaction confidentiality

Theorem 2 (Transaction confidentiality)**.** *Our scheme satisfies transaction confidentiality, if the twisted ElGamal algorithm is IND-CPA secure, and the audit proof π_{au} is zero-knowledge.*

Proof of Theorem 2. We prove it via the following games. Let Win_i denote the probability that the adversary $\mathcal{A}$ wins the $Game_i$.

$Game_0$: We proceed with the transaction confidentiality experiment defined in Section 5.2. The challenger C and the adversary $\mathcal{A}$ interact as the following shows:

(1) C computes $pp \leftarrow setup(\lambda)$ and $(X, Y) \leftarrow keygen$. It returns the generated pp and Y to $\mathcal{A}$.
(2) $\mathcal{A}$ queries O_{Pre} and O_{GenCT}. C answers these queries. On input $((C^{in}_i, v^{in}_i, s^{in}_i), v_\rho)$, run $ptx \leftarrow pretx(pp, C^{in}_i, v^{in}_i, s^{in}_i, v_\rho, Y)$ and store $\{(C^{in}_i, v^{in}_i, s^{in}_i), v_\rho, Y, ptx\}$ into the list L. On input $(ptx.rmdr)$, search L, run $tx.out \leftarrow tx(pp, ptx.rmdr, Y)$ and $\pi_{au} \leftarrow au(pp, ptx.out, \pi_{pau}, Y)$, and then return $tx.out$ and π_{au}.
(3) $\mathcal{A}$ chooses $\{ptx.rmdr_0, ptx.rmdr_1\}$. C randomly selects $b \in [0, 1]$ and computes $tx.out^* \leftarrow tx(pp, ptx.rmdr_b, Y)$, $\pi^*_{au} \leftarrow au(pp, ptx.rmdr_b, ptx.chg, \pi_{pau}, Y)$. It returns the generated $\{tx.out^*, \pi^*_{au}\}$ to $\mathcal{A}$.
(4) $\mathcal{A}$ generates the guess b' of b. If $b = b'$, it wins the experiment.

Therefore, we have $Adv_{\mathcal{A}}(\lambda) = Pr[Win_0] - \frac{1}{2}$.

$Game_1$: $Game_1$ is similar to $Game_0$ except that the audit proof π_{au} is generated by simulator $S = (S_1, S_2)$. S_1 generates the trapdoor τ, and then S_2 takes τ as input without any proof. It outputs the simulated proof π_{au}. Therefore, the proof generated by S_2 is the same as the proof computed in $Game_1$. The probability that $\mathcal{A}$ wins $Game_1$ satisfies

$$|Pr[Win_1] - Pr[Win_0]| \leq negl(\lambda). \tag{6.1}$$

As we can see in Lemma 1, we have $Pr[Win_1] \leq negl(\lambda)$.

Lemma 4. *If the twisted ElGamal algorithm is IND-CPA secure, then for all PPT adversary $\mathcal{A}$, we have $Pr[Win_1] \leq negl(\lambda)$.*

Proof of Lemma 4. Suppose that there is a PPT adversary $\mathcal{A}$ that wins $Game_1$ with non-negligible advantage, and then we can contruct algorithm $\mathcal{B}$ that can break the IND-CPA secure property of the twisted ElGamal algorithm. $\mathcal{B}$ simulates $Game_1$ as the following shows:

(1) $\mathcal{B}$ computes $pp \leftarrow setup(\lambda)$ and $(X, Y) \leftarrow keygen(pp)$. It uses S_1 to generate the trapdoor τ, and then it returns them to $\mathcal{A}$.

(2) $\mathcal{A}$ queries the oracle O_{Pre} and the oracle O_{GenCT}. The challenger C answers these queries.
O_{Pre}: $\mathcal{A}$ makes this query with $(C_i^{in}, v_i^{in}, s_i^{in}, v_\rho)$. C receives this query, and then it executes $ptx \leftarrow pretx(C_i^{in}, v_i^{in}, s_i^{in}, v_\rho, Y)$. It stores $(C_i^{in}, v_i^{in}, s_i^{in}, v_\rho, Y, ptx)$ in the list L.
O_{GenCT}: $\mathcal{A}$ makes this query with $(ptx.rmdr)$. C receives this query, and then it executes $tx.out \leftarrow tx(pp, ptx.rmdr, Y)$. It takes the trapdoor τ generated by S_2, and it outputs simulated π_{tr}. It returns $tx.out$ and π_{tr} to $\mathcal{A}$.

(3) $\mathcal{A}$ selects two pre-transaction remainders $\{ptx.rmdr_0, ptx.rmdr_1\}$. $\mathcal{B}$ sends $\{ptx.rmdr_0, ptx.rmdr_1\}$ to its challenger C. $\mathcal{B}$ receives $C_j^{out*} = \{C_{1j}^{out*}, C_{1j}^{out*}\}$, where C_j^{out*} is the encrypted value that is obtained by encrypting $ptx.rmdr_b$ using audit public key Y. Let $tx.out^* = \{C_j^{out*}\}$. $\mathcal{B}$ takes the trapdoor τ as input. It outputs the simulated audit proof π_{tr}^*. $\mathcal{B}$ returns $tx.out^*$ and π_{tr}^* to $\mathcal{A}$ as challenge.

(4) $\mathcal{A}$ generates b' as the guess of b, then $\mathcal{B}$ returns the guess generated by $\mathcal{A}$.

We can see that $\mathcal{B}$ successfully simulates the $Game_1$, so it can break the IND-CPA secure property of twisted ElGamal algorithm with the same advantage. We prove the Lemma 4.

To sum up, we prove that if the twisted ElGamal algorithm is IND-CPA secure, and the audit proof π_{au} is zero-knowledge, our scheme satisfies transaction confidentiality.

6.2. *Public verification*

6.2.1. Balance verification

Theorem 3 (Balance verification)**.** *Our scheme enables transaction balance verification, which means that outputs of the transaction and the inputs of the transaction are equal, if the Discrete logarithm assumption holds.*

Proof of Theorem 3. Suppose that there is a PPT adversary $\mathcal{A}$ that wins the transaction balance experiment we defined in Section 3 with non-negligible advantage, and then we can construct algorithm $\mathcal{B}$ that can solve the Discrete logarithm problem with the same advantage. Let $pp = (\mathbb{G}, P, H, q, H_0)$. (P, H) is the instance of $\mathcal{B}$'s Discrete logarithm problem, where P and H are two random generators of $\mathbb{G}$. $\mathcal{B}$ simulates the experiment as the following shows:

(1) $\mathcal{B}$ computes $pp \leftarrow setup(\lambda)$ and $(X, Y) \leftarrow keygen(pp)$. It returns the generated public parameters pp and the public key Y to $\mathcal{A}$.

(2) $\mathcal{A}$ queries oracles O_{Pre} and O_{GenBal}. These oracles answer these queries.
O_{Pre}: $\mathcal{A}$ makes this query with $(C_i^{in}, v_i^{in}, s_i^{in}, v_\rho)$. C computes $(ptx) \leftarrow pretx(pp, C_i^{in}, v_i^{in}, s_i^{in}, v_\rho, Y)$, and then it stores $(C_i^{in}, v_i^{in}, s_i^{in}, v_\rho, Y, ptx)$ into the list L.
O_{GenBal}: $\mathcal{A}$ makes this query with $(ptx.rmdr)$. C receives this query and computes $tx.out \leftarrow tr(pp, ptx.rmdr,$
$Y)$. It selects L to find the corresponding (π_{pbp}, P_b), and then it computes $\pi_{bp} \leftarrow bl(pp, \pi_{pbp}, P_b)$. It returns $tx.out$ and π_{bp} to $\mathcal{A}$.

(3) $\mathcal{A}$ obtains complete transaction information that includes transaction inputs $tx.in = \{C_i^{in} | C_i^{in} = \{C_{1i}^{in}, C_{2i}^{in}, i \in [1, n]\}\}$, transaction outputs $tx.out = \{C_j^{out}, C_c^{out} | C_j^{out} = \{C_{1j}^{out}, C_{2j}^{out}\}, j = [1, n'], C_c^{out} = \{C_{1c}^{out}, C_{2c}^{out}\}\}$ and transaction balance information $\pi_{bl} = \{\sigma, e, \triangle, \bar{X}\}$. $\mathcal{B}$ rewinds e_2 and σ_2. Therefore, we have:

$$Y\sigma - e(C_{1c}^{out} + \sum_{j=1}^{n'} C_{1j}^{out} - \sum_{i=1}^{n} C_i^{in}) \tag{6.2}$$

$$= Y\sigma_2 - e_2(C_{1c}^{out} + \sum_{j=1}^{n'} C_{1j}^{out} - \sum_{i=1}^{n} C_i^{in})$$

$$e(C_{2c}^{out} + \sum_{j=1}^{n'} C_{2j}^{out} - \sum_{i=1}^{n} C_{2i}^{in}) - \sigma P \tag{6.3}$$

$$= e_2(C_{2c}^{out} + \sum_{j=1}^{n'} C_{2j}^{out} - \sum_{i=1}^{n} C_{2i}^{in}) - \sigma_2 P$$

Let $x_s^* = (\sigma - \sigma_2)/(e - e_2)$, and then $\bar{X}^* = x_s^* G$ can be regarded as the transaction public balance excess value. We have

$$x_s^* Y = C_{1c}^{out} + \sum_{j=1}^{n'} C_{1j}^{out} - \sum_{i=1}^{n} C_{1i}^{in} \tag{6.4}$$

$$x_s^* G = \bar{X}^* = C_{2c}^{out} + \sum_{j=1}^{n'} C_{2j}^{out} - \sum_{i=1}^{n} C_{2i}^{in} \tag{6.5}$$

If $\sum_{i=1}^{n} v_i^{in} \neq \sum_{j=1}^{n'} v_j^{out} + v_c^{out}$, then we have

$$x_s^* G = \bar{X}^* = C_{2c}^{out} + \sum_{j=1}^{n'} C_{2j}^{out} - \sum_{i=1}^{n} C_{2i}^{in} \tag{6.6}$$

$$= (\sum_{i=1}^{n} v_i^{in} - v_c^{out} - \sum_{j=1}^{n'} v_j^{out})H + (s_s^{out} - s_s^{in})G$$

So, we have $(\sum_{i=1}^{n} v_i^{in} - v_c^{out} - \sum_{j=1}^{n'} v_j^{out})H = (s_s^{out} - s_s^{in} - x_s^*)P$. Therefore, $\mathcal{B}$ can take $log_P H = (s_s^{out} - s_s^{in} - x_s^*)/(\sum_{i=1}^{n} v_i^{in} - v_c^{out} - \sum_{j=1}^{n'} v_j^{out})$ as the solution of the Discrete logarithm problem.

Thus, if the Discrete logarithm problem is hard to solve, our scheme satisfy the transaction balance property.

6.2.2. Multiplicative verification

Theorem 4 (Multiplicative verification)**.** *Our scheme enables multiplicative verification, which means that our scheme is able to prove and verify some encrypted values v_1, v_2, v_3 satisfy product relationship $v_1 \cdot v_2 = v_3$, if the Discrete logarithm assumption holds.*

Proof of Theorem 4. Suppose that there exists a PPT adversary $\mathcal{A}$ that can break the multiplicative verification property with non-negligible advantage, and then we can construct algorithm $\mathcal{B}$ that can solve the Discrete logarithm problem with the same advantage. Let $pp = (\mathbb{G}, P, H, q, H_0)$. (P, H) is the instance of $\mathcal{B}$'s Discrete logarithm problem, where P and H are two random generators of $\mathbb{G}$. $\mathcal{B}$ simulates the experiment as the following shows:

(1) $\mathcal{B}$ computes $pp \leftarrow setup(\lambda)$ and $(X, Y) \leftarrow keygen(pp)$. It returns the generated public parameters pp and the public key Y to $\mathcal{A}$.
(2) $\mathcal{A}$ queries the O_{pro} oracle with $(v_1, v_2, v_3, C_{21}, C_{22}, C_{23})$. C computes $\pi_{pro} \leftarrow pro(pp, v_1, v_2, v_3, C_{21}, C_{22}, C_{23})$. It returns π_{pro} to the adversary $\mathcal{A}$.
(3) $\mathcal{A}$ obtains the transaction information (C_{21}, C_{22}, C_{23}) and multiplicative proofs $\pi_{pro} = \{c, u_1, u_2, u_3, \theta_1, \theta_2, \theta_3, \theta_4\}$. $\mathcal{B}$ rewinds c', u'_1, u'_2, u'_3,θ'_1,θ'_2,θ'_3 and θ'_4. Therefore, we have

$$\theta_1 P + u_1 H - cC_{21} = \theta'_1 P + u'_1 H - c'C_{21} \tag{6.7}$$

$$\theta_2 P + u_2 H - cC_{22} = \theta'_2 P + u'_2 H - c'C_{22} \tag{6.8}$$

$$\theta_3 P + u_3 H - cC_{23} = \theta'_3 P + u'_3 H - c'C_{23} \tag{6.9}$$

$$u_2 C_{21} + \theta_4 P - cC_{23} = u'_2 C_{21} + \theta'_4 P - c'C_{23} \tag{6.10}$$

Let $v_1^* = (u_1 - u'_1)/(c - c')$, $s_1^* = (\theta_1 - \theta'_1)/(c - c')$,$v_2^* = (u_2 - u'_2)/(c - c')$, $s_2^* = (\theta_2 - \theta'_2)/(c - c')$,$v_3^* = (u_3 - u'_3)/(c - c')$, $s_3^* = (\theta_3 - \theta'_3)/(c - c')$ and $s^* = (\theta_4 - \theta'_4)/(c - c')$. Then, we have $v_3^* H + s^* 3P = v_1^* v_2^* H + (v_2^* s_1^* + s^*)P$. If $v_1^* v_2^* \neq v_3^*$, we have $(v_1^* v_2^* - v_3^*)H = (s_3^* - s^* - v_2^* s_1^*)P$. $\mathcal{B}$ can take $log_P H = (s_3^* - s^* - v_2^* s_1^*)/(v_1^* v_2^* - v_3^*)$ as the solution of the Discrete logarithm problem.

Thus, if the Discrete logarithm problem is hard to solve, our scheme satisfies multiplicative verification.

6.3. Reliable audit

Theorem 5 (Reliable audit)**.** *Transactions in our privacy-preserving transaction scheme can be reliably audited.*

Proof of Theorem 5. Suppose that trading parties (payee and payer) may construct a fake to escape audit. The adversary's malicious actions can be roughly summarized as the following two types:

(1) The adversary $\mathcal{A}$ randomly chooses $Y' \in \mathbb{G}, Y' \neq Y$ to generate encrypted transaction outputs instead of using audit public key Y. It computes $C_{1j}^{out'} = s_j^{out} Y'$, $C_j^{out'} = \{C_{1j}^{out'}, C_{2j}^{out}\}$. Therefore, validating nodes can verify it as the following shows:

$$R'_1 = \tilde{\sigma} Y - \tilde{c} C_{1c}^{out} - \tilde{c} \sum_{j=1}^{n'} C_{1j}^{out'} \tag{6.11}$$

$$= s_c^{out'} Y + \sum_{j=1}^{n'} s_j^{out'} Y + \tilde{c} s_c^{out} Y$$

$$+ \tilde{c} \sum_{j=1}^{n'} s_j^{out} Y - \tilde{c} s_c^{out} Y' - \tilde{c} \sum_{j=1}^{n'} s_j^{out} Y'.$$

We can see that $Y' \neq Y$, so $R_1' \neq s_c^{out'}Y + \sum_{j=1}^{n'} s_j^{out'}Y$ and $R_1' \neq s_c^{out'}Y + \sum_{j=1}^{n'} s_j^{out'}Y'$. Therefore, we have $R_1' \neq R_1$. Besides, hash functions are collision-resistant, so we get $\tilde{c}' \neq \tilde{c}$.

(2) The adversary $\mathcal{A}$ randomly chooses $v_j^{out'} \neq v_j^{out}$ to generate encrypted transaction outputs instead of using the real transaction value v_j^{out}. It computes $C_{2j}^{out'} = s_j^{out}P + v_j^{out'}H, C_j^{out'} = \{C_{1j}^{out}, C_{2j}^{out'}\}$. Therefore, validating nodes can verify it as the following shows:

$$\begin{aligned}
R_2' &= \sigma' H + \bar{\sigma} P - \tilde{c} C_{2c}^{out} - \tilde{c} \sum_{j=1}^{n'} C_{2j}^{out'} \\
&= v_c^{out'} H + s_c^{out'} P + \sum_{j=1}^{n'} v_j^{out'} H + \sum_{j=1}^{n'} s_j^{out'} P \\
&+ \tilde{c} \sum_{j=1}^{n'} v_j^{out} H - \tilde{c} \sum_{j=1}^{n'} v_j^{out'} H \\
&= R_{2c} + \sum_{j=1}^{n'} v_j^{out'} H + \sum_{j=1}^{n'} s_j^{out'} P \\
&+ \tilde{c} H \sum_{n=1}^{n'} (v_j^{out} - v_j^{out'})
\end{aligned} \tag{6.12}$$

We can see that $v_j^{out'} \neq v_j^{out}$, so $R_2' \neq R_{2c} + \sum_{j=1}^{n'} v_j^{out'}H + \sum_{j=1}^{n'} s_j^{out'}P$ that is $R_2' \neq R_2$. Therefore, we get $\tilde{c}' \neq \tilde{c}$.

In summary, the probability of the audit proof information forged by the adversary $\mathcal{A}$ that can pass the verification is negligible. Therefore, our scheme satisfies transaction auditability.

7. Performance analysis

In order to evaluate the performance of our proposed scheme, we implement the prototype of the proposed privacy preserving transaction scheme which mainly focuses on the transaction layer without considering the differences of consensus mechanisms. This makes our privacy preserving transaction scheme more feasible for different blockchain systems. Our implementation is in Golang language on a laptop with 8GB of RAM, an Intel Core i7-8500U 2.00GHz. The elliptic curve we used is secp256k1, and the hash function is sha256.

According to Table 3, we give an evaluation of the computation time about each step of the main phase in our proposed privacy preserving transaction scheme. We take the most frequently used 2 inputs-1 outputs as instance. As we can see from Table 3, computation times in each phase such as setup, transact, verify and audit are all in milliseconds. The total time is approximate 7.65 ms. It is practical and feasible for low frequency transaction scenarios.

Table 3. Computation time of the main phase of our proposed scheme in milliseconds.

Phase	Step	Time (ms)
Setup	Setup	0.232
Transact	Generate encrypted outputs	0.439
	Generate balance proofs	0.877
	Generate multiplicative proofs	1.308
	Generate Audit proofs	0.953
Verify	Balance proofs verification	0.349
	Multiplicative relationship verification	1.810
	Audit proofs verification	1.244
Audit	Audit	0.438

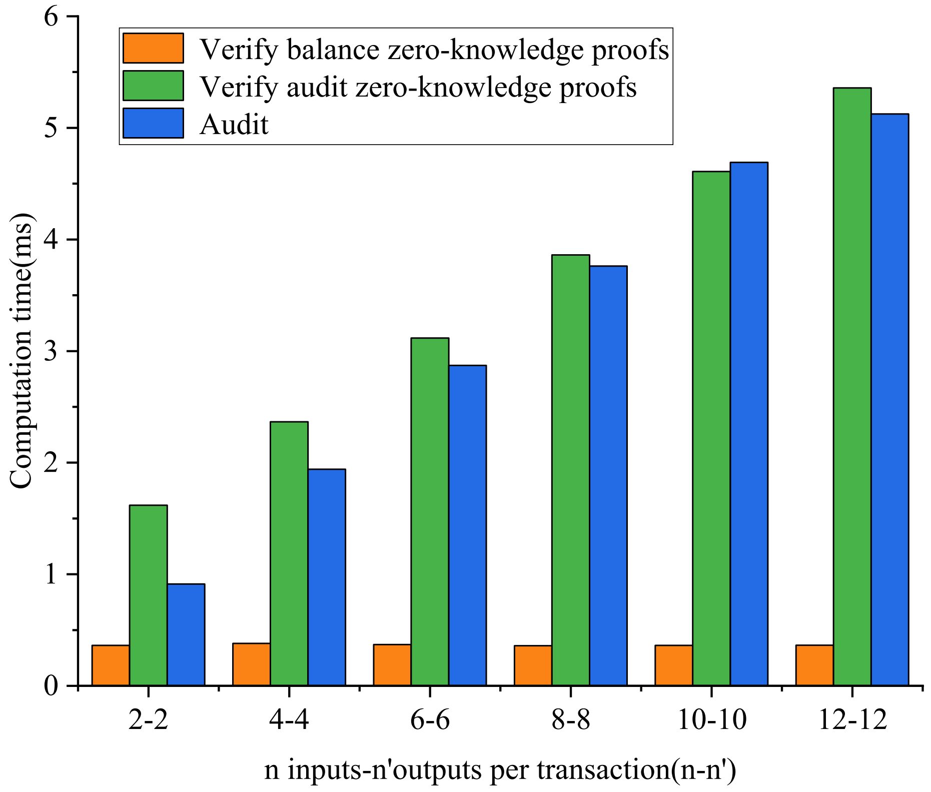

Figure 4. Computation time comparison in verify and audit phase with increasing inputs and outputs.

In Figures 3 and 4, we also evaluate our privacy preserving transaction scheme's time costs in transact, verify and audit phases with increasing inputs and outputs. According to Figure 3, as the number of inputs and outputs grows from 2-2 to 12-12 in one transaction, the balance zero-knowledge prove time and audit zero-knowledge prove time are approximately 0.9 and 1.0 ms with no obvious increasing. In Figure 4, the balance zero-knowledge proofs verification time requirements is kept approximate 0.4 ms as the number of inputs and outputs increasing from 2-2 to 12-12. Though the time of generating encrypted values grows from 0.8 to 4.9 ms in Figure 3, and the time of verifying audit zero-knowledge proofs and auditing time are increasing from 1.6 to 5.4 ms and 0.9 to 5.1 ms respectively in Figure 4, they are still within milliseconds.

Figure 5 presents the verification time comparison before and after aggregation, and Figure 6 presents the block size comparison before and after aggregation. According to Figure 5, the verification time linearly grows from 4.9 to 21.0 ms as the number of inputs and outputs is set to be 2-2, 4-4, 6-6, 8-8, 10-10, 12-12 respectively when there is no aggregation of balance proofs and audit proofs. However, in our proposed privacy preserving transaction scheme, we aggregate the balance proofs and audit proofs, which greatly shortens the verification time, as it approximately grows 3.8 to 7.5 ms when the number of inputs and outputs is set to be 2-2, 4-4, 6-6, 8-8, 10-10, 12-12, respectively. For the reason that we replace the multiplication operation with the faster add operation of group in our aggregation algorithm, the verification time has no obvious growth. Therefore, our aggregation algorithm makes the transaction verification more efficient. As we can see in Figure 6, the growth rate of block size has been significantly slowed as the number of transactions in a block after we make aggregation of the audit proofs and balance proofs. Thus, the aggregation technique reduces the storage size of proof at least 50% of the size before optimization. It effectively saves the ledger space.

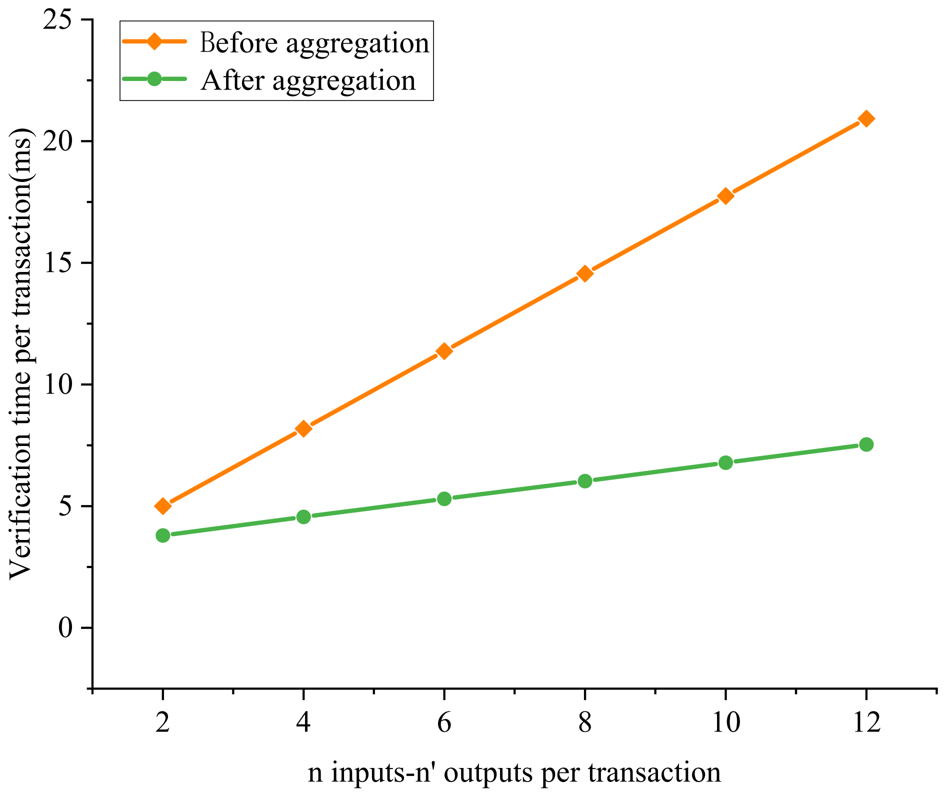

Figure 5. Verification time comparison before and after aggregation.

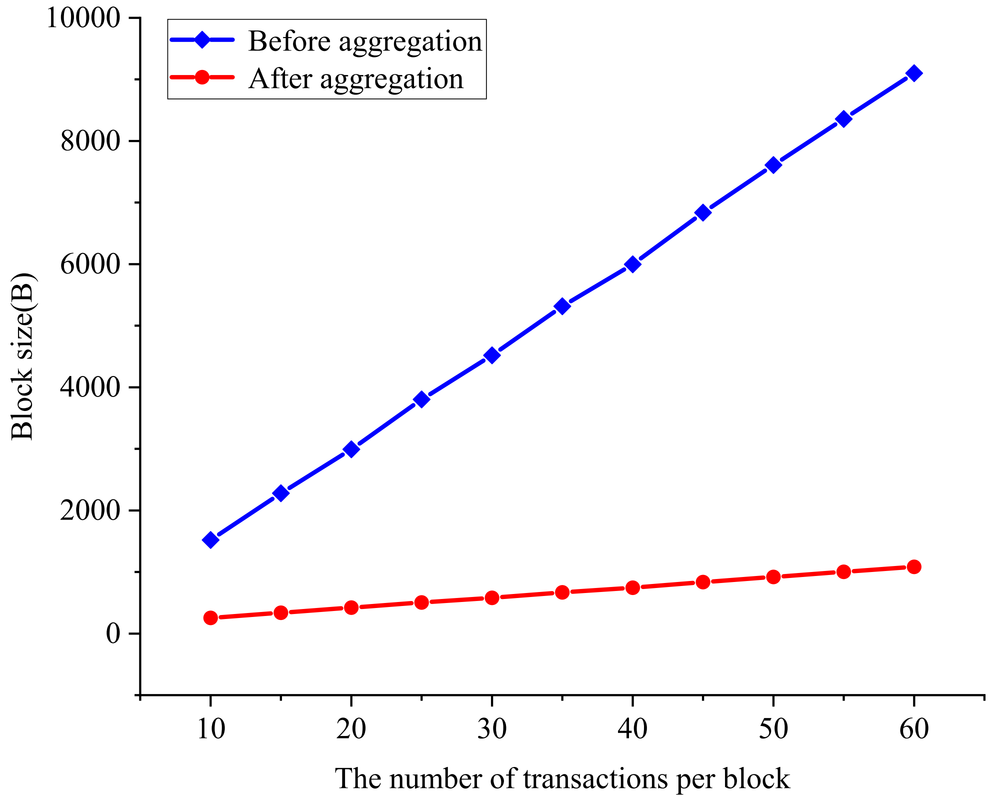

Figure 6. Block size comparison before and after aggregation.

Our scheme has functional advantages. In particular, there are several applications in blockchain for the proposed multiplicative zero-knowledge proof to be used in some specific scenarios. For monetary assets in UTXO model, if there are k outputs with the same value v for a user and the total amount of them is $sum = v \cdot k$, it needs to computes k encrypted values that $C_1 = \{C_{11} = s_1Y, C_{21} = vH + s_1P\}, ..., C_k = \{C_{1k} = s_kY, C_{2k} = vH + s_kP\}$, and it needs to store k encrypted values $C_1, C_2, ..., C_k$ in the leger. However, by using the proposed multiplicative zero-knowledge proof, it only needs to compute two encrypted values C_v, C_k and only stores these two ciphertexts in the leger without influencing the transaction balance and reliable audit. It is obvious that using the proposed multiplicative zero-knowledge proof achieves space savings of ledger and efficiency gains for the user. For data assets such as those in supply chain, suppose that the quantity of goods is r, the unit price of goods is v, and the total amount is $t = v \cdot r$. r, v and $t$need to record in chain with privacy preserving. We can compute $C_v = \{C_{1v} = s_vY, C_{2v} = vH + s_vP\}$,$C_r = \{C_{1r} = s_rY, C_{2r} = rH + s_rP\}$, and $C_t = \{C_{1t} = s_tY, C_{2t} = tH + s_tP\}$. This hides the transaction information, and then the multiplicative zero-knowledge proof ensures $t = v \cdot r$ to be public verified by validators in blockchain without revealing t, r and v.

8. Conclusions

In this paper, we propose a privacy preserving transaction scheme with public verification and reliable audit in blockchain. Our scheme not only provides confidentiality for transaction contents in a more flexible way by decoupling user identity and transaction contents, but also defines several verification rules that makes full use of validators in blockchain. It enables balance verification for monetary assets, and then we design a multiplicative zero-knowledge proof with security analysis, which can be potentially used in blockchain based financial applications, supply chains and so on. Then, validators can optionally multiplicative verification of data assets to ensure the data compliance by applying the proposed multiplicative proof. In addition, our proposal enables the auditor to make precise audit of each transaction which audit reliability is guaranteed by publicly verifying the audit proof. Security analysis shows that the proposed scheme satisfies the security requirements we defined. Performance analysis indicates that its computation cost is in milliseconds, and the aggregation effectively saves the storage space. Also, how to construct a more efficient range-proof is still to be taken into consideration.

Acknowledgments

This paper was supported by National Natural Science Foundation of China (Grant no. U21A20463).

Conflict of interest

The authors declare there is no conflicts of interest.

References

1. Y. Cao, F. Jia, G. Manogaran, Efficient traceability systems of steel products using blockchain-based industrial Internet of Things, *IEEE Trans. Ind. Inf.*, **16** (2019), 6004–6012. https://doi.org/10.1109/TII.2019.2942211
2. L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, et al, Creditcoin: a privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles, *IEEE Trans. Intell. Transp. Syst.*, **19** (2018), 2204–2220. https://10.1109/TITS.2017.2777990
3. S. J. Lee, J. C. Chew, Y. J. Liu, C. Y. Chen, Y. K. Tsai, Medical blockchain: data sharing and privacy preserving of EHR based on smart contract, *Int. J. Inf. Secur. Appl.*, **65** (2022), 103117. https://doi.org/10.1016/j.jisa.2022.103117
4. H. Huang, P. Zhu, F. Xiao, X. Sun, Q. Huang, A blockchain-based scheme for privacy-preserving and secure sharing of medical, *Comput. Secur.*, **99** (2020), 102010. https://doi.org/10.1016/j.cose.2020.102010
5. S. Purohit, P. Calyam, L. M. Alarcon, R. N. Bhamidipati, HonestChain: consortium blockchain for protected data sharing in health information systems, *Peer-to-Peer Netw. Appl.*, **14** (2021), 3012–3028. https://doi.org/10.1007/s12083-021-01153-y
6. W. Wang, J. Song, G. Xu, Y. Li, H. Wang, C. Su, Contractward: automated vulnerability detection models for ethereum smart contracts, *IEEE Trans. Network Sci. Eng.*, **8** (2020), 1133–1144. https://doi.org/10.1109/TNSE.2020.2968505
7. X. Liu, J. Liu, S. Zhu, W. Wang, X. Zhang, Privacy risk analysis and mitigation of analytics libraries in the android ecosystem, *IEEE Trans. Mob. Comput.*, **9** (2020), 1184–1199. https://doi.org/10.1109/TMC.2019.2903186

8. W. Wang, Y. Shang, Y. He, Y. Li, J. Liu, BotMark: automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors, *Inf. Sci.*, **511** (2020), 284–296. https://doi.org/10.1016/j.ins.2019.09.024

9. W. Wang, M. Zhao, J. Wang, Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network, *J. Ambient Intell. Hum. Comput.*, **10** (2010), 3035–3043. https://doi.org/10.1007/s12652-018-0803-6

10. Y. Zhang, J. Wen, The IoT electric business model: using blockchain technology for the Internet of Things, *Peer-to-Peer Netw. Appl.*, **10** (2017), 983–994. https://doi.org/10.1007/s12083-016-0456-1

11. D. Gabay, K. Akkaya, M. Cebe, Privacy-preserving authentication scheme for connected electric vehicles using blockchain and zero knowledge proofs, *IEEE Trans. Veh. Technol.*, **69** (2020), 5760–5772. https://doi.org/10.1109/TVT.2020.2977361

12. L. Xue, D. Liu, J. Ni, X. Lin, S. X. Shen, Enabling regulatory compliance and enforcement in decentralized anonymous payment, *IEEE Trans. Dependable Secure Comput.*, **2022** (2022). https://doi.org/10.1109/TDSC.2022.3144991

13. S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system, *Decentralized Bus. Rev.*, **2008** (2008), 21260. Available from: https://www.belegger.nl/Forum/Upload/2017/10425916.pdf.

14. Monero: a secure, private, untraceable cryptocurrency, 2021. Available from: https://www.getmonero.org/.

15. B. E. Sasson, A. Chinesa, C. Garman, M. Green, I. Miers, E. Tromer, et al., Zerocash: decentralized anonymous payments from bitcoin, in *2014 IEEE Symposium on Security and Privacy*, IEEE, (2014), 459–474. https://doi.org/10.1109/SP.2014.36

16. I. Miers, C. Garman, M. Green, D. A. Rubin, Zerocoin: anonymous distributed e-cash from bitcoin, in *2013 IEEE Symposium on Security and Privacy*, IEEE, (2013), 397–411. https://doi.org/10.1109/SP.2013.34

17. E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, M. Virza, SNARKs for C: verifying program executions succinctly and in zero knowledge, in *Annual Cryptology Conference*, Springer, Berlin, Heidelberg, **8043** (2013), 90–108. https://doi.org/10.1007/978-3-642-40084-1_6

18. G. Fuchsbauer, M. Orrù, Y. Seurin, Aggregate cash systems: a cryptographic investigation of mimblewimble, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, Cham, **11476** (2019), 657–689. https://doi.org/10.1007/978-3-030-17653-2_22

19. G. Maxwell, Confidential transactions. Available from: https://www.weusecoins.com/confidential-transactions/.

20. P. T. Pedersen, Non-interactive and information-theoretic secure verifiable secret sharing, in *Annual International Cryptology Conference*, Springer, Berlin, Heidelberg, **576** (1991), 129–140. https://doi.org/10.1007/3-540-46766-1_9

21. N. Narula, W. Vasquez, M. Virza, zkLedger: privacy-preserving auditing for distributed ledgers, in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, (2018), 65–80. Available from: https://www.usenix.org/conference/nsdi18/presentation/narula.

22. R. Singh, A. D. Dwivedi, R. R. Mukkamala, W. S. Alnumay, Privacy-preserving ledger for blockchain and Internet of Things-enabled cyber-physical systems, *Comput. Electr. Eng.*, **103** (2022), 108290. https://doi.org/10.1016/j.compeleceng.2022.108290

23. H. T. Yuen, PAChain: private, authenticated & auditable consortium blockchain and its implementation, *Future Gener. Comput. Syst.*, **112** (2020), 913–929. https://doi.org/10.1016/j.future.2020.05.011

24. S. Dhar, A. Khare, R. Singh, Advanced security model for multimedia data sharing in Internet of Things, *Trans. Emerging Telecommun. Technol.*, **2022** (2022), e4621. https://doi.org/10.1002/ett.4621

25. K. Wüst, K. Kostiainen, V. Čapkun, S. Čapkun, Prcash: fast, private and regulated transactions for digital currencies, in *International Conference on Financial Cryptography and Data Security*, Springer, Cham, **11598** (2019), 158–178. https://doi.org/10.1007/978-3-030-32101-7_11

26. S. Malik, V. Dedeoglu, S. Kanhere, R. Jurdak, Privchain: provenance and privacy preservation in blockchain enabled supply chains, preprint, arXiv:2104.13964.

27. P. Chatzigiannis, F. Baldimtsi, Miniledger: compact-sized anonymous and auditable distributed payments, in *European Symposium on Research in Computer Security*, Springer, Cham, **12972** (2021), 407–429. https://doi.org/10.1007/978-3-030-88418-5_20

28. Y. Chen, X. Ma, C. Tang, H. M. Au, PGC: decentralized confidential payment system with auditability, in *European Symposium on Research in Computer Security*, Springer, Cham, **12308** (2020), 591–610. https://doi.org/10.1007/978-3-030-58951-6_29

29. G. Danezis, S. Meiklejohn, Centrally banked cryptocurrencies, preprint, arXiv:1505.06895.

30. E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, et al., Hyperledger fabric: a distributed operating system for permissioned blockchains, in *Proceedings of the Thirteenth EuroSys Conference*, (2018), 1–15. https://doi.org/10.1145/3190508.3190538

31. S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof systems, *SIAM J. Comput.*, **18** (1989), 186–208. https://doi.org/10.1137/0218012

32. M. Blum, P. Feldman, S. Micali, Non-interactive zero-knowledge and its applications, in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, (2019), 329–349.

33. J. Camenisch, M. Stadler, Efficient group signature schemes for large groups, in *Annual International Cryptology Conference*, Springer, Berlin, Heidelberg, **1294** (1997), 410–424. https://doi.org/10.1007/BFb0052252

34. F. Hao, *Schnorr Non-interactive Zero-knowledge Proof*, *Tech. Rep.*, 2017. Available from: https://www.rfc-editor.org/rfc/rfc8235.

35. A. Fiat, A. Shamir, How to prove yourself: practical solutions to identification and signature problems, in *Conference on the Theory and Application of Cryptographic Techniques*, Springer, Berlin, Heidelberg, **263** (1986), 186–194. https://doi.org/10.1007/3-540-47721-7_12

36. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, G. Maxwell, Bulletproofs: short proofs for confidential transactions and more, in *2018 IEEE Symposium on Security and Privacy (SP)*, IEEE, (2018), 315–334. https://doi.org/10.1109/SP.2018.00020