Mathematics

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

AIMS Mathematics, 9(3): 6183–6206.
DOI: 10.3934/math.2024302
Received: 09 November 2023
Revised: 03 January 2024
Accepted: 12 January 2024
Published: 02 February 2024

Research article

BPA: A decentralized payment system that balances privacy and auditability

Le Gao, Junzhe Zhang, Jiaxin Yu, Yin Tang and Zhiqiang Zeng*

School of Electronic and Information Engineering, Jiangmen,529020, China

* **Correspondence:** Email: zhiqiang.zeng@outlook.com.

Abstract: The rapid development of blockchain transactions highlights the importance of privacy protection (including anonymity and confidentiality) and underscores the necessity for auditability. Some schemes, such as PGC and Minilеdger, support privacy protection and auditability. However, they only offer incomplete privacy protection (i.e., supporting anonymity or confidentiality exclusively). In response to these issues, we propose a scheme that achieves partial anonymity, confidentiality, auditability, and traceability. By integrating a variant of Pedersen commitments and randomizable signatures, we achieve partial anonymity for users and the auditability of transactions, thereby protecting user privacy under audit conditions. Based on the twisted ElGamal encryption algorithm and specially constructed zero-knowledge proofs, we achieve confidentiality of transaction amounts under legal and regulatory conditions. System test results indicate that this scheme effectively meets the above requirements. The feasibility of this scheme is confirmed through system testing, comparative analysis, and security analysis.

Keywords: blockchain; cryptocurrencies; auditable; confidential transactions; privacy protection
Mathematics Subject Classification: 94A62, 91B99

1. Introduction

Blockchain [1], a distributed ledger technology, is distinguished by its decentralized and tamper-resistant nature. Compared to traditional centralized storage models, information stored on blockchains is considered more authentic and reliable, thereby resolving trust issues. Since its introduction by Satoshi Nakamoto in 2008, blockchain has attracted much attention from researchers in academic and industry circles. Its inherent attributes of openness, verifiability, and programmability have found widespread application in fields such as finance, public services, and digital copyright. Nevertheless, cryptocurrency remains one of the primary use cases of blockchain technology, enabling rapid value transfer through a reliable distributed ledger. As of 2023, over 151,000 articles related to cryptocurrency [2] have covered more than 9,111 different cryptocurrencies, with a market capitalization exceeding USD 1.19 trillion [3].

Although cryptocurrencies offer numerous benefits, they also pose significant challenges to privacy protection. Because the blockchain serves as an append-only public ledger, it is collectively upheld by all entities within the system in a manner that is both verifiable and indelible. Consequently, transactions conducted on the blockchain are inherently transparent, granting any node within the network the ability to access all transactions without restrictions. This level of accessibility can allow various participants, including cybercriminals and data miners, to obtain sensitive information, such as user identities and transaction amounts, which they may exploit for illicit purposes.

To address this challenge, Bitcoin [4] introduced pseudonymity as a means of protecting user identities. Users generate new addresses using their public keys in each transaction, thereby avoiding the exposure of personal information. However, this anonymity relies on the assumption that addresses are unrelated to real-world identities. The emergence of various de-anonymization attacks indicates that pseudonymity is insufficient to provide adequate anonymity [5]. To overcome this limitation, researchers have proposed numerous robust anonymity schemes, such as Zcash [6], CoinJoin [7], and Monero [8]. Zcash uses zero-knowledge proof technology to hide transaction participants or amounts. CoinJoin employs a mixing mechanism that combines multiple transactions from different users into a single transaction, making it challenging for external observers to link transaction addresses to specific users. Monero, through the application of ring signature technology, mixes the public key of the transaction participant with a collection of random public keys, followed by signing the message, thereby obfuscating the information of the transaction participant. These schemes aim to enhance anonymity and confidentiality, resolve the limitations of pseudonymity, and alleviate the risks associated with de-anonymization attacks, thereby strengthening user privacy protection.

Nonetheless, overemphasizing anonymity and confidentiality may also introduce new challenges. Since cryptocurrencies are not recognized as legal tender or subject to government regulation, law enforcement agencies have difficulty ascertaining the destination or amount of transactions. This lack of transparency hampers the capacity of regulatory bodies to monitor and investigate the cryptocurrency market effectively, thus facilitating illegal activities like drug trafficking, money laundering, and terrorism. These activities could damage the reputation of financial institutions and pose a significant threat to the economic system. Consequently, there is growing concern that unregulated anonymous payments could increase the incidence of illegal activities.

To address this challenge, some have proposed integrating auditability into blockchain systems. Auditability is a critical attribute of any financial system, with traditional financial institutions implementing policies such as know your customer (KYC) [9] and anti-money laundering (AML) [10] to ensure regulatory compliance. To achieve similar regulatory functions in blockchain systems, some government agencies have proposed the concept of central bank digital currency [11]. The advantage of this currency is that it is issued by the government and protected by law, thereby promoting financial stability and government oversight. However, this approach also centralizes power excessively, and user privacy cannot be guaranteed.

In order to strike a balance between privacy and auditability, we incorporate user identities into transactions. Each transaction includes a user's identity credentials, ensuring that the regulator can recover user identities upon detecting illicit transactions. We enhance user anonymity by randomizing identity credentials, ensuring the confidentiality of user balances and transaction amounts through encryption, as well as adhering to regulatory policies through zero-knowledge proofs. The contributions of this paper are summarized in the following three paragraphs.

We propose BPA, a decentralized payment scheme that achieves partial anonymity, confidentiality, auditability, and traceability for cryptocurrency transactions. This scheme permits users who have obtained identity credentials from issuers to initiate transactions. After being verified by validators, these transactions are uploaded to the blockchain. Regulators can interact with validators to audit the compliance of transactions. For violating users, regulators can recover the user's identity through interaction with the issuers.

We have developed a formal security model for BPA schemes by considering regulatory compliance and privacy protection, and we demonstrate that it meets the expected security objectives and ensures compliance with relevant laws and regulations.

We implemented BPA in C++ and tested it on the Windows platform. The experimental results show that our scheme is efficient and has low computational overhead.

The rest of this article is organized as follows. Section 2 reviews related work on privacy and auditing in cryptocurrencies. In Section 3, we introduce some background knowledge. We describe our scheme's system model, security model, and system algorithm in Section 4. We present a detailed implementation plan for BPA in Section 5. In Section 6, we provide a performance evaluation and functional comparison. Finally, we conclude in Section 7.

2. Related work

2.1. Decentralized payment system that ensures privacy protection

Due to the lack of privacy in cryptocurrencies such as Bitcoin and Ethereum, privacy protection has become a significant research topic in recent years. Maxwell [7] initiated research on confidential transactions, proposing a decentralized confidential payment scheme that uses Pedersen commitment [12] to hide transfer amounts, and that uses range proofs to ensure transaction correctness. Mimblewimble [13] improves on Maxwell's work by reducing the signature size. Additionally, van Saberhagen proposed CryptoNote [14], which uses traceable ring signatures [15] to hide transaction participants and one-time keys to prevent the double spending of coins. Monero is a cryptocurrency based on the CryptoNote protocol that uses RingCT [16] to provide transaction privacy within small anonymity sets, hiding transaction destinations and amounts. CoinJoin is a mixing mechanism that enhances anonymity but requires a centralized server in the system. Quisquis [17] is a scheme similar to Monero but with smaller transaction sizes. Zerocoin [18] is a scheme proposed by Miers et al. that uses accumulators and non-interactive zero-knowledge (NIZK) [19] proofs to ensure user transaction anonymity, but it does not support direct transfers and hides transaction amounts. Zerocash [20] is a scheme developed by Sasson et al. that hides the flow of transactions and the amounts through nested commitments, zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKS) [21], and Merkle trees [22].

2.2. Decentralized payment system that supports auditing

To address the regulatory issues arising from privacy protection, some scholars have proposed the following solutions: Garman et al. [23] introduced auxiliary data to Zerocash in order to establish accountability, which a dedicated trust agency supervises. However, the type of audit it implements is limited, and the authority of the trust agency is pervasive. Solidus [24] adopts a bank-like structure but does not conceal the identities of transaction participants. It permits validators to disclose the number of transfer users. Li et al. introduced Traceable Monero [25], which balances user anonymity and accountability based on Monero by employing verifiable encryption techniques. ZkLedger [26] is a distributed ledger system that offers robust privacy and auditing support for multiple audits, but high storage costs hinder its practicality. MiniLedger [27], on the other hand, is a scheme that leverages accumulators based on ZkLedger to aggregate transactions, thereby conserving storage space. PG [28] is a scheme for auditable confidential payments based on the account model. It conceals the transaction amount by using twisted ElGamal encryption and incorporates range proofs through the use of Bulletproofs technology [29]. Regarding user identity, PGC employs the same pseudonymous mechanism as Bitcoin. It offers three accountability methods: Limiting the amount sent or received, paying a specified tax, and disclosing the transaction amount. Zether [30] is a scheme for account-based smart contracts on Ethereum that achieve K-anonymity and preserve amount confidentiality. It introduces Σ-Bullets to enhance Bulletproofs. BlockMaze [31], on the other hand, employs a dual-balance model and a two-step transfer mechanism to conceal user balances, transfer amounts, and transaction identities, using zk-SNARKs to achieve transaction correctness.

Androulaki et al. [32] introduced a privacy-preserving auditable token management system. Their proposed scheme uses the unspent transaction output (UTXO) model in a permissioned blockchain and allows a group of reviewers to review participants, thereby gaining access to all designated participant information. However, they only target business-to-business scenarios. The work of Damgård et al. [33] addressed the problem of balancing accountability with privacy. They proposed a new architectural design of an "identity layer", integrating self-sovereign identity management with transactions, ensuring compliance with regulatory requirements such as KYC/AML. Nevertheless, the use of secure multiparty computation [34] among the auditors (i) limits the distribution of trust and (ii) requires all parties to be online at the same time. Islam et al. [35] achieved self-manageable authentication on the chain through the use of dynamic decentralized identifiers, but their design still revealed the user's transaction amount. Platypus [36] is a scheme that merges the electronic cash transaction processing approach with the fund management model based on the account model. It introduces features such as anonymous payments and anonymous payment budgeting while imposing limits on the total balance of users. However, its payment protocol necessitates user interaction to complete transactions. The UTT [37] is based on the UTXO model, and it represents coins generated by transactions by using homomorphic commitments. The bank is responsible for re-randomizing and signing the commitment through the use of a verifiable random signature. The sender must prove that the sum of the input coins is equal to the sum of the output coins to ensure the correctness of the transaction. UTT enforces a monthly anonymous budget to control the upper limit of anonymous transfers. Recently, Xue et al. [38] proposed a scheme for supervised anonymous payments in the UTXO model. It calculates the total transfer amount and the number of transfers without revealing links between transactions. It allows regulatory agencies to restore user identities when users violate regulations; however, this plan exposes the transaction amount. Lin et al. [39] have developed a transaction system based on the UTXO model that provides anonymity, confidentiality, and auditability through the use of cryptographic accumulators and homomorphic encryption. Managers can generate traceable anonymous public keys and transaction ciphertexts for users, open transaction ciphertexts, and track long-term public keys. However, this grants managers a significant amount of power.

3. Preliminaries

Notations. In this article, given positive integers denoted by n, we use $[n]$ to represent the set $\{1, ..., n\}$. Z_p represents the integers modulo p. Z_p^* is the inverse set in Z_p. We denote three prime-order p cyclic groups as G_1, G_2, G_3 and the generators as g_1, g_2, g_3, respectively. We use $[a, b]$ for $a, b \in Z_p$ to represent the integer set $\{a, a+1, ..., b-1, b\}$. We use $x \overset{\$}{\leftarrow} S$ to indicate that x is randomly and uniformly sampled from set S.

3.1. Cryptographic assumptions

Definition 3.1. (Discrete logarithm assumption [40]) Suppose that G is a cyclic group generated by g. We define $\lambda \in N$ as the security parameter, and negl(λ) represents the negligible function. Then, for the probabilistic polynomial-time (PPT) adversary A, the probability of obtaining a from (g, g^a) is less than negl(λ), where (g, g^a) belongs to G and a belongs to Z_p.
Definition 3.2. (Decisional Diffie-Hellman assumption) Suppose that G is a cyclic group generated by g, g^a, g^b, g^c. Then, for the PPT adversary A, the probability of distinguishing (g, g^a, g^b, g^c) from (g, g^a, g^b, g^{ab}) is less than negl(λ).
Definition 3.3. (Divisible decisional Diffie-Hellman assumption) Suppose that G is a cyclic group generated by g, g^a, g^b, g^c. Then, for the PPT adversary A, the probability of distinguishing (g, g^a, g^b, g^c) from $(g, g^a, g^b, g^{a/b})$ is less than negl(λ).
Definition 3.4. (Bilinear group) Use three prime-order p cyclic groups G_1, G_2, G_t to define bilinear pairing. e is the mapping $e : G_1 \times G_2 \to G_t$. If e satisfies bilinearity, non-degeneracy, and computability, then the mapping is a bilinear pairing. A tuple $(G_1, G_2, G_t, p, g_1, g_2, e)$ that contains such a bilinear mapping e is defined as a bilinear group. To improve efficiency, we use the type-3 bilinear pairing, which requires that $G_1 \neq G_2$, and there is no computable isomorphism ψ such that $\psi(G_1) = G_2$. The multiplication symbol is used throughout the paper to denote G_1G_2.

3.2. "Double" Pedersen commitment

The Pedersen commitment scheme [12] is a two-party protocol between a sender and a receiver. In the "commit" phase, the sender commits a certain value, m, to the receiver by sending a commitment. In the "open" phase, the sender can reveal the commitment by providing m and r. The receiver can verify whether the value they received matches the value the sender committed in the commitment phase by using this information. The double Pedersen commitment is a variant that introduces re-randomization to the original Pedersen commitment. The randomized commitment is unrelated to the original commitment but commits to the same value m. Its structure is as follows:

- **DCM.Setup**(1^λ): Input security parameter 1^λ, select $(g_1, g_2) \leftarrow G_1 * G_2, \mathbf{g_1} \leftarrow g_1^y, \mathbf{g_2} \leftarrow g_2^y$, output public parameter $pp = (G_1, G_2, g_1, g_2, \mathbf{g_1}, \mathbf{g_2})$.
- **DCM.Com**($m; r$): Input message $m \in Z_p$ and random number $r \in Z_p$, calculate $(cm_1, cm_2) \leftarrow (\mathbf{g_1}^m g_1^r, \mathbf{g_2}^m g_2^r)$, output $cm = (cm_1, cm_2)$.
- **DCM.Open**(cm, m, r): If $cm \leftarrow (\mathbf{g_1}^m g_1^r, \mathbf{g_2}^m g_2^r)$, output "1"; otherwise, output "0".
- **DCM.Rerand**($cm; r$): Input commitment $cm = (cm_1, cm_2)$ and random number $r \in Z_p$, calculate randomized commitment value $(cm_1', cm_2') \leftarrow (cm_1 \cdot g_1^r, cm_2 \cdot g_2^r)$, output randomized commitment $cm' = (cm_1', cm_2')$.

3.3. *Randomizable signatures*

The randomizable signatures [41] that David Pointcheval suggested in 2016 were used in this study. This type of signature has the advantage of being more efficient and growing nonlinearly. It comprises a signer and a verifier. During the signing phase, the signer utilizes the private key to sign the message *cm*. In the verification phase, the verifier can use the signer's public key to verify that the signature was generated correctly. During the randomization phase, the verifier can randomize *cm* and the signature, ensuring that the randomized signature can still pass verification. This article's randomizable signature comprises five polynomial-time techniques, which are as follows:

- **PS**.**Setup**(1^λ): Input security parameter 1^λ, select $(g_1, g_2) \leftarrow \mathrm{G}_1 * \mathrm{G}_2$, output public parameter $pp = (\mathrm{G}_1, \mathrm{G}_2, g_1, g_2)$.
- **PS**.**KeyGen**(pp): Input public parameter pp, randomly generate $x \leftarrow \mathrm{Z}_p$, output $(sk, vk) \leftarrow (g_1^x, g_2^x)$.
- **PS**.**Sign**($sk, cm_1; u$): Randomly generate $u \leftarrow \mathrm{Z}_p$, calculate $(\sigma_1, \sigma_2) \leftarrow (g_1^u, (sk \cdot cm_1^u))$, output $\sigma = (\sigma_1, \sigma_2)$.
- **PS**.**Verify**($vk, (cm_1, cm_2), \sigma$): Input vk, cm_1, cm_2, σ, check whether $e(cm_1, g_2) = e(g_1, cm_2)$ and $e(\sigma_2, g_2) = e(\sigma_1, vk \cdot cm_2)$. If all pass, output "1"; otherwise, output "0".
- **PS**.**Rerand**($\sigma; r, u$): Input signature σ opened as (σ_1, σ_2), random values $r, u \in \mathrm{Z}_p$, output the new random commitment $\sigma' \leftarrow (\sigma_1^u, (\sigma_2 \cdot \sigma_1^r)^u)$.

3.4. *Twisted ElGamal encryption*

In 1985, Tahir Gamal introduced the widely accepted ElGamal encryption method. The process primarily involves encryption and decryption phases. During encryption, anyone can use the public key to encrypt the message *m* to generate ciphertext. In the decryption stage, only users with the corresponding private key can decrypt the ciphertext and obtain the message *m*. The encryption algorithm employed in this study is a variant known as twisted ElGamal [28]. It supports multiplicative homomorphic encryption, distinguishing itself from conventional ElGamal encryption techniques. Twisted ElGamal encompasses four distinct algorithms:

- **TE**.**Setup**(1^λ): Input security parameter 1^λ, select $h \leftarrow \mathrm{G}_3^*$, output public parameter $pp = (\mathrm{G}_3, p, g_3, h)$. Randomness and message space are denoted by Z_p.
- **TE**.**KeyGen**(pp): Input pp, select $sk \leftarrow \mathrm{Z}_p$, let $pk = g_3^{sk}$, output (pk, sk).
- **TE**.**Enc**($pk, m; r$): Calculate $X = pk^r, Y = g_3^r h^m$, output $C = (X, Y)$.
- **TE**.**Dec**(sk, C): Parse $C = (X, Y)$, calculate $h^m = Y/X^{sk^{-1}}$, recover m from h^m.

3.5. *Zero-knowledge proof*

The concept of zero-knowledge proof consists of two identities: The prover and the verifier. The role of the prover is to persuade the verifier of the veracity of specific statements without divulging any insight into their underlying rationale. For example, it is conceivable to prove that two ciphertexts encrypt the same message without revealing the actual message. Nevertheless, participating in such proofs' interactive process is often associated with substantial computation costs and may even become infeasible in some cases. By employing the Fiat-Shamir transformation, any public-coin honest-verifier zero-knowledge proof can be converted into NIZK under the random oracle model, thus eliminating the need for interaction. In NIZK, 'R' represents the relationship between statement x and witness w, while 'L' represents the language containing all statements in 'R'. NIZK comprises four key algorithms: **Setup**, **CRSGen**, **Prove**, and **Verify**.

- **NIZK**.**Setup**(1^λ): Input security parameter 1^λ, output public parameter pp.
- **NIZK**.**CRSGen**(pp): Input pp, output common reference string crs.
- **NIZK**.**Prove**(crs, x, w): Run by the prover. Output proof π.
- **NIZK**.**Verify**(crs, x, π): Run by the verifier. Verify the proof π generated by the prover and return result 1/0.

4. BPA overview

This section provides a high-level overview of the BPA system and presents the security model required for our proposed scheme. BPA is a confidential payment system operating under the account model, incorporating supervisory and privacy protection measures. The system is characterized by its partial anonymity, confidentiality, auditability, and traceability. It aims to protect users' personal information while avoiding the issue of excessive centralization of power, all in compliance with relevant laws and regulations.

4.1. Role

Users: In this system, users can serve as transaction senders and receivers. Each user maintains their confidential account and must interact with the issuer to generate personal identification credentials to conduct transactions.

Issuer: The issuer has its own public and private key pair (Ipk, Isk), which is responsible for generating users' identity credentials and assisting regulatory authorities in tracking transactions for compliance. Apart from the issuer, no other parties are privy to users' real identities.

Regulator: The regulator has its own public and private key pair (Rpk, Rsk) and is responsible for regulating blockchain transactions. If the validator detects a fraudulent transaction, it can request that the issuer disclose the user's identity. Additionally, the regulator may proactively interact with users to ensure compliance with relevant laws and regulations.

Validator: The validator has its own public and private key pair (Vpk, Vsk) and is responsible for maintaining the integrity and security of the blockchain while verifying user-initiated transactions. Only verified transactions are recorded on the blockchain. If the validator receives a fraudulent transaction from a user, it reports it to the regulator for appropriate action.

4.2. System model

As depicted in Figure 1, after initializing the system, our proposed scheme has the following three phases. During the register phase, each participant in the system initially needs to create their account. The transaction sender must also register with the issuer to acquire personal identity credentials. In the transaction phase, to hide both parties' information, the sender randomizes its identity credentials and the public keys of both parties and masks the actual amount by generating corresponding encryption for the transaction amount. The transaction must also include the sender's serial number to prevent double-spending attacks. Moreover, each transaction must be accompanied by a corresponding zero-knowledge proof to ensure validity and compliance. The validator is responsible for verifying each transaction; if the transaction is deemed valid, it will be recorded on the blockchain and the balances of both parties will be updated. In the audit and tracking phase, regulators can ask the validator about any transaction to complete the audit. For violating users, regulators can recover the user's identity through interaction with the issuer.

Through this design, the issuer is aware of the user's real identity but does not know the transaction destination; the validator knows the transaction destination but does not know the user's identity; the regulator needs to initiate applications to both the validator and the issuer separately in order to complete the transaction audit and track non-compliant users. Our scheme applies to those transaction systems in real life that require the involvement of governments or institutions. For example, currency transactions between different countries or currency circulation between banks.

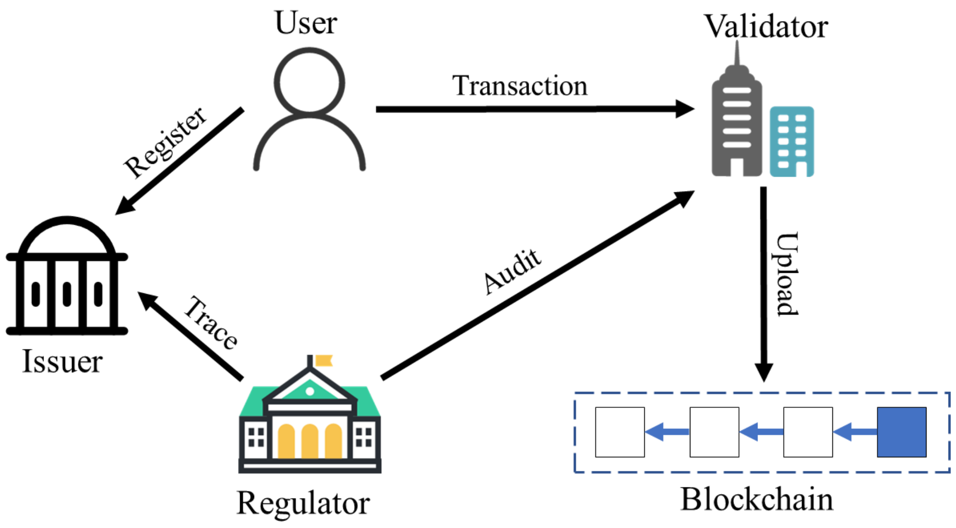

Figure 1. System model diagram.

4.3. System algorithm

Our BPA transaction system is primarily divided into the following algorithms:

- **Setup**(1^{λ}): Input security parameter λ and obtain the related public parameters pp. The trusted authority uses this algorithm to establish the entire system.
- **CreateAccount**(val, sn): Given the input of the value val and serial number sn, output user key pair (pk, sk) and encrypted amount C. A user executes this algorithm to create an account.
- **GetBalance**(sk, C): Given the input of the user private key and encrypted balance C, the algorithm output is the amount val. A user runs this algorithm to obtain an account balance.
- **CreateID**(pk, m): Given input of the public key pk and personal information m; the algorithm generates a commitment-signature pair (cm, σ). Through this algorithm, a user obtains personal identity credentials (cm, σ) from the issuer.
- **Send**$(pk_a, sk_a, pk_b, v, cm_a, \sigma_a)$: Given input of the sender's key pair (pk_a, sk_a), personal identity credentials (cm_a, σ_a), transfer amount v and receiver public key pk_b, output transaction tx_{send}. A user executes this algorithm to generate a transaction.

- **VerifySend**(tx_{send}): Given input of a transaction tx_{send}, return "1" if the transaction is valid; otherwise, return "0". The validators use this algorithm to verify a transaction.
- **JustifySend**(Rpk, x_1, x_2): Given input of the regulator public key Rpk and auxiliary information x_1, x_2, output $C_{Rpk} \leftarrow$ **TE.Enc**($Rpk, x_1 \| x_2; r$). The validators use this algorithm to respond to the regulator's audit requirements.
- **AuditSend**(tx_{send}, z): Given input of a transaction tx_{send} and supplementary information z, return "1" if the transaction is valid; otherwise, return "0". The regulator uses this algorithm to audit transactions.
- **Trace**(pk, cm, σ): Given input of the user public key pk_a and personal identity credentials (cm, σ), the issuer checks whether cm and σ are valid and belong to pk. If the verification succeeds, output the user's personal information, m; otherwise, output 0. The issuer uses this algorithm to provide the personal information of illegal users to the regulator.

4.4. Security model

This section presents the security model of the BPA scheme. Specifically, we focus on attacks at the transaction layer, while attacks at the network layer or other layers are beyond the scope of this paper.

For a decentralized payment system that balances privacy and auditability, the required attributes include anonymity, confidentiality, auditability, and traceability. As mentioned in the preceding text, we define $\lambda \in N$ as the security parameter, and $\mathrm{negl}(\lambda)$ represents the negligible function. Subsequently, we formally define these attributes by implementing a game involving adversaries A, a challenger CL, and the BPA oracle O^{BPA}, where A is a polynomial-time adversary. A can send different types of queries to CL, including **CreateAccount**, **CreateID**, and **Send** queries. After performing integrity checks on the queries, CL forwards the queries to O^{BPA}, which maintains the ledger, executes the queries according to the BPA scheme, and outputs the resulting transactions. In this way, A can elicit the behavior of honest users and learn the public output. Notably, A cannot obtain the private input when generating transactions. A can also send a transaction to CL, which is called an insertion query. After the integrity check, CL forwards the transaction to O^{BPA}.

GAME Anonymity: For anonymity, we define the property through ledger indistinguishability; that is, the ledger does not reveal any information about the user other than the public information on the ledger. CL, A, and two oracles O_0^{BPA}, O_1^{BPA} participate in the game.

Initialization: CL randomly selects a bit $b \in \{0, 1\}$ and initializes O_0^{BPA} and O_1^{BPA}. For $i \in \{0, 1\}$, O_i^{BPA} maintains a ledger L_i.

Query: A has the capability to submit distinct queries, such as **CreateAccount**, **CreateID**, and **Send**, to two different oracles. On each occasion, A submits two queries, Q and Q', both of which share the same type and public information with CL. CL provides two views of the ledger L_0, L_1 to A with a random order, i.e., $L_{left} = L_b, L_{right} = L_{1-b}$, where $b \in \{0, 1\}$. If the queries are **CreateAccount**, **CreateID**, and **Send** queries, then, after the soundness checks of the two queries, CL sends Q to O_0^{BPA} and Q' to O_1^{BPA}. If the query is an insert query, then Q is sent to O_{left}^{BPA}, and Q' is sent to O_{right}^{BPA}.

Guess: After the queries, A needs to determine whether the ledger it sees is $L_{left} = L_0, L_{right} = L_1$, indicating that $b = 0$, or $L_{left} = L_1, L_{right} = L_0$, i.e., $b = 1$. A returns a bit b' to C, which is A's guess.

If $b' = b$, then A wins the game; otherwise, A fails in the game. The anonymity property requires that A's advantage can be considered negligible, i.e., $|Pr[b' = b] - \frac{1}{2}| \leq negl(\lambda)$.

GAME Confidentiality: Confidentiality requires that others cannot determine the hidden amounts within confidential transactions apart from the transaction parties. The game involves CL, A, and an oracle O^{BPA}.

Initialization: CL selects two amounts v_0 and v_1 as challenge values, randomly selects a bit $\beta \in \{0, 1\}$, generates a transaction tx for v_β, and initializes O^{BPA} , which maintains a ledger L.

Query: Adversary A has the capability to submit queries to O^{BPA}, where O^{BPA} simulates the behavior of honest users. O^{BPA} executes the queries and provides A with a view of the ledger L.

Guess: After querying, A generates a guess value β'. If $\beta' = \beta$, then A wins the game.

Confidentiality necessitates that adversary A's advantage in this game can be considered negligible. That is to say, the mathematical representation of this as follows: $|Pr[\beta' = \beta] - \frac{1}{2}| \leq negl(\lambda)$.

GAME Auditability: For auditability, we require that no PPT adversary A can deceive the regulator into accepting false audit results. The game involves CL, A, and an oracle O^{BPA}.

Initialization: CL initializes O^{BPA} , which maintains a ledger L.

Query: Adversary A can interact with oracle O^{BPA} by sending different types of queries to CL, which proxies the queries to O^{BPA}. O^{BPA} performs queries and provides A with a view of the ledger L, simulating the behavior of an honest party. O^{BPA} can use the **Audit** algorithm in BPA to audit transactions.

Output: At this stage, A outputs a pour transaction tx.

If tx violates the regulatory policy but passes the audit conducted by the regulator, A wins. If A can win the above game with no more than a negligible probability, then BPA achieves auditability.

GAME Traceability: The proposed BPA achieves traceability if the regulator properly traces any users in violation. The game involves CL, A, and an oracle O^{BPA}.

Initialization: CL initializes O^{BPA} , which maintains a ledger L.

Query: Adversary A has the capability to interact with the oracle O^{BPA} by sending queries of varying types to CL, which then proxy these queries to O^{BPA}. O^{BPA} executes the queries and provides A with a view of the ledger L, thereby simulating the behavior of honest parties. O^{BPA} can utilize the tracing algorithm within BPA to track users who violate regulations.

Output: At this stage, A outputs a pour transaction tx.

If the transaction tx violates regulatory policies but passes the verification conducted by the validator, the regulator is unable to recover the user's identity, or if the recovered user has not registered with the issuer, then A emerges victorious. BPA achieves traceability if A can win the above game with no more than a negligible probability.

5. BPA: A decentralized payment system that balances privacy and auditability

5.1. System components

In this section, we will elaborate on the construction of the BPA system. To enhance the scheme's comprehensibility, we illustrate the process by using the transaction between User A and User B as an example, following a sequence of *initialization*, *register*, *transaction*, *audit*, *and tracking*. The main symbols involved are depicted in Table 1.

Table 1. Parameters in the system.

Acronym	Definition	Acronym	Definition
pp	Public parameters	C_a, C_b	Initial encrypted balance
pk_a, sk_a	User A's public key and private key	sn_a, sn_b	User's initial serial number
pk_b, sk_b	User B's public key and private key	$pk_a^{'}, pk_b^{'}$	User's randomized public key
Rpk, Rsk	Regulator public key and private key	x_1, x_2	Auxiliary information
Vpk, Vsk	Validator public key and private key	C_1	Encrypted transaction amount
Ipk, Isk	Issuer public key and private key	C_2	Encrypted transaction amount
m	User A's true identity	C_3	Encrypted auxiliary information
cm, σ	User A's credential	tx_{send}	User A's transaction
$cm^{'}, \sigma^{'}$	User A's randomized credential	Π	Zero knowledge proof

Initialization: In the initialization phase, the trust authority executes the **Setup** algorithm; choose an elliptic curve group $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ of order q. g_1, g_3, h are generators of group $\mathbb{G}_1$, g_2 is a generator of group $\mathbb{G}_2$, and, for each of them, the discrete logarithm of an element concerning the other one is unknown. The corresponding bilinear mapping is denoted as $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$. The trust authority chooses a random number $y \in \mathbb{Z}_p$ and calculates $\mathbf{g_1} = g_1^y$ and $\mathbf{g_2} = g_2^y$. The system's public parameters $pp = \{g_1, g_2, g_3, \mathbf{g_1}, \mathbf{g_2}, h\}$ are derived through the aforementioned computations. After that, the trust authority randomly selects $x, Rsk, Vsk \in \mathbb{Z}_p$, x denotes the parameter for the issuer, and Rsk denotes the regulator's private key, Vsk denotes the validator's private key; the trust authority then calculates the regulator public key $Rpk = g_3^{Rsk}$, issuer key pairs ($Ipk = g_2^x, Isk = g_1^x$), and validator public key $Vpk = g_3^{Vsk}$.

Register: During the register phase, users execute the **createAccount** and **createID** algorithms. First of all, both User A and User B independently select their respective amounts $v_a, v_b \in \mathbb{Z}_p$ and initial serial numbers $sn_a, sn_b \in \{0, 1\}^n$ (e.g., $n = 256$); run **TE**.**KeyGen**(pp) to generate their user key pairs (pk_a, sk_a) and (pk_b, sk_b) and compute **TE**.**Enc**($pk_{a/b}, v_{a/b}; r$) to generate the initial encrypted balances C_a, C_b for the User A and User B.

In order to initiate a transaction, User A also needs to interact with the issuer to generate an identity credential. Specifically, given the input pk_a and true identity m, the issuer runs **DCM**.**Com**($m; r$) to generate a commitment $cm = (cm_1, cm_2)$. Randomly generate $u \in \mathbb{Z}_p$; run **PS**.**Sign**($Isk, cm; u$) to generate a signature σ. Finally, return the identity credential (cm, σ) to User A. Now, User A's account includes ($sn_a, pk_a, sk_a, C_a, cm, \sigma$); User B's account includes (sn_b, pk_b, sk_b, C_b).

Transaction: In the transaction phase, User A executes **Send** to generate a transaction, and a validator runs **VerifySend** to validate whether the transaction is correct.

User A inputs the identity credentials (cm, σ), transfer amount v, their key pair (pk_a, sk_a), and User B's public key pk_b. The **Send** algorithm first checks whether $v \in V$ and $(v_a - v) \in V$. If not satisfied, it returns as false. Otherwise, perform the following steps:

(1) Open cm as (cm_1, cm_2), randomly generate $x_1, x_2 \in \mathbb{Z}_p$, and generate randomized public keys $pk_a' \leftarrow pk_a^{x_1}$ and $pk_b' \leftarrow pk_b^{x_1}$ from both parties' public keys. Run $cm' =$ **DCM**.**Rerand**($(cm_1, cm_2); x_1$) and $\sigma' =$ **PS**.**Rerand**($\sigma; x_1, x_2$) to obtain a randomized credential.

(2) Randomly generate $r, r' \in \mathbb{Z}_p$ and run $C_1 =$ **TE**.**Enc**($pk_a, v; r$), $C_2 =$ **TE**.**Enc**($pk_b, v; r$), $C_3 =$ **TE**.**Enc**($Vpk, x_1 \| x_2; r'$). C_1, C_2 are encrypted ciphertexts for the transaction amount, while C_3

is an encrypted ciphertext for auxiliary information. To ensure transaction compliance while safeguarding user privacy, User A must generate the corresponding zero-knowledge proofs Π. The zero-knowledge proofs include L_{equal}, L_{right} and $L_{solvent}$.

$$L_{equal} = \{(pk_a, pk_b, C_1, C_2) | \exists sk_a, r, v \ s.t. C_1 = \mathbf{TE.Enc}(pk_a, v; r) \wedge C_2 = \mathbf{TE.Enc}(pk_b, v; r)\}$$

$$L_{right} = \{(pk_a, C_1) | \exists r, v \ s.t. C_1 = \mathbf{TE.Enc}(pk_a, v; r) \wedge v \in V\}$$

$$L_{solvent} = \{(pk_a, C_a, C_1) | \exists sk_a \ s.t. (pk_a, sk_a) \in R_{key} \wedge \mathbf{TE.Dec}(sk_a, C_a - C_1) \in V\}$$

The L_{equal} proof demonstrates that C_1 and C_2 are correctly generated and encrypt the same amount v, the L_{right} proof shows that C_1 is accurately generated, and the transaction amount v falls within the correct range. The $L_{solvent}$ proof validates that User A's public key is accurately generated and the resulting balance after the transaction is within the correct range.

(3) Finally, User A outputs a transaction $tx_{send} = (sn_a, cm', \sigma', pk'_a, pk'_b, C_1, C_2, C_3, \Pi)$.

When the validator receives a transaction tx_{send}, open the transaction as $(sn_a, cm', \sigma', pk'_a, pk'_b, C_1, C_2, C_3, \Pi)$, and then verify the following:

(1) Run $x_1 \| x_2 \leftarrow \mathbf{TE.Dec}(Vsk, C_3)$ to obtain x_1 and x_2; restore cm', σ' by running the following:
$$\mathrm{cm}' \cdot \left(g_1^{-x_1}, g_2^{-x_1}\right) = \left(\mathrm{cm}'_1 \cdot g_1^{-x_1}, \mathrm{cm}'_2 \cdot g_2^{-x_1}\right)$$
$$= \left(\mathrm{cm}_1 \cdot g_1^{x_1} \cdot g_1^{-x_1}, \mathrm{cm}_2 \cdot g_2^{x_1} \cdot g_2^{-x_1}\right)$$
$$= (\mathrm{cm}_1, \mathrm{cm}_2) = \mathrm{cm}$$
$$\sigma' = (\sigma'_1, \sigma'_2) = \left(\sigma_1^{x_2}, \left(\sigma_2 \cdot \sigma_1^{x_1}\right)^{x_2}\right)$$
$$\left(\sigma_1^{x_2 \cdot \frac{1}{x_2}}, \left(\sigma_2 \cdot \sigma_1^{x_1}\right)^{x_2 \cdot \frac{1}{x_2}} \cdot \sigma_1^{-x_1}\right) = (\sigma_1, \sigma_2) = \sigma$$
and check whether $\mathbf{PS.Verify}(Ipk, cm, \sigma) = 1$.

(2) Calculate $pk_a \leftarrow pk'^{1/x_1}_a$, $pk_b \leftarrow pk'^{1/x_1}_b$ by using x_1; check whether $\mathbf{NIZK.Verify}(crs, tx_{send}, \Pi) = 1$.

(3) Check whether sn_a is a new serial number by using pk_a.

(4) If all of the above are verified, output "1". The validator uploads the transaction to the blockchain and updates User A's balance as $C_a = C_a - C_1$, and User B's balance as $C_b = C_b + C_2$.

Audit and tracking: At this stage, the regulator can initiate an audit request to the validator at any time. Upon receiving the request, the validator runs **JustifySend** to respond. The regulator executes **AuditSend** to audit the transaction. If the transaction violates any rules, the **Trace** operation is executed to conduct tracking.

When the validator receives an audit request, input of the regulator public key Rpk, and the auxiliary information x_1 and x_2 of the transaction tx_{send} results in the output $C_{Rpk} \leftarrow \mathbf{TE.Enc}(Rpk, x_1 \| x_2; r)$. The validator sends C_{Rpk} to the regulator.

Upon receiving the information, the regulator runs $x_1\|x_2 \leftarrow \mathbf{TE.Dec}(Rsk, C_{Rpk})$, decrypts tx_{send} based on x_1 and x_2 to obtain (cm, σ, pk_a, pk_b), and checks whether $\mathbf{NIZK.Verify}(crs, tx_{send}, \Pi) = 1$ and $\mathbf{PS.Verify}(Ipk, cm, \sigma) = 1$. The regulator can audit the number of transactions and the total transaction amounts for each user to check if they exceed the specified limits. If surpassed, the regulator may request tracing from the issuer. The issuer then provides the user's true identity as related to the respective transactions.

5.2. NIZK instantiation

As described in the previous section, our approach utilizes L_{equal}, L_{right}, and L_{solvent}, with their corresponding NIZK implementations being Π_{equal}, Π_{right}, and Π_{solvent}, respectively. We define $\pi_{\text{leg}} := \Pi_{\text{equal}} \circ \Pi_{\text{right}} \circ \Pi_{\text{solvent}}$, where $\circ$ denotes sequential composition. Based on the NIZK properties of conjunction statements, we can prove that π_{leg} is a valid proof system. Next, we will provide separate descriptions for these three categories.

5.2.1. NIZK implementation of L_{equal}

Since our scheme has security in the indistinguishability under the chosen plaintext attack (IND-CPA), L_{equal} can be rewritten as

$$\{(pk_1, pk_2, X_1, X_2, Y) \mid \exists r, v \text{ s.t. } X_i = pk_i^{r_i} \wedge Y = g_3^r h^v \text{ for i} = 1, 2\}.$$

Afterward, we convert it into a specific implementation through the Sigma protocol, as shown in Table 2:

Table 2. Sigma protocol of L_{equal}: Proof of knowledge of two twisted ElGamal encryptions with the same encrypted value under different public keys.

$$\begin{array}{ccc}
 & (pk_1, pk_2, X_1, X_2, Y) & \\
P & & V \\
w = (r, v) & & \\
a, b \overset{R}{\leftarrow} Z_p & & \\
A_1 \leftarrow pk_1^a, A_2 \leftarrow pk_2^a & \xrightarrow{A_1, A_2, B} & \\
B \leftarrow g_3^a h^b & & \\
 & \xleftarrow{e} & e \overset{R}{\leftarrow} Z_p \\
z = a + er & & \\
l = b + ev & \xrightarrow{z, l} & \text{check if} \\
 & & pk_1^z = A_1 X_1^e \\
 & & pk_2^z = A_2 X_2^e \\
 & & g_3^z h^l = BY^e
\end{array}$$

Finally, we were able to achieve the NIZK proof of L_{equal} by transforming its Sigma protocol into Π_{equal} using the Fiat-Shamir transformation.

5.2.2. NIZK implementation of L_{right}

In the BPA system, L_{right} is defined as follows:

$$\left\{(pk, X, Y) \mid \exists r, v \text{ s.t. } X = pk^r \wedge Y = g_3^r h^v \wedge v \in V\right\}.$$

Through analysis, the above definition can be decomposed as follows:

$$L_{\text{enc}} = \left\{(pk, X, Y) \mid \exists r, v \text{ s.t. } X = pk^r \wedge Y = g_3^r h^v\right\},$$

$$L_{\text{range}} = \left\{Y \mid \exists r, v \text{ s.t. } Y = g_3^r h^v \wedge v \in V\right\}.$$

It is straightforward to verify that $L_{\text{right}} \subset L_{\text{enc}} \wedge L_{\text{range}}$. The last component Y can be seen as a Pedersen commitment of value v under commitment key (g, h), whose discrete logarithm $\log_g h$ is unknown to all users. Note that each instance $(pk, X, Y) \in L_{\text{right}}$ has a unique witness.

Next, we give the specific protocol in Table 3.

Table 3. Sigma protocol of L_{enc}: Proof of knowledge of twisted ElGamal ciphertext.

	(pk, X, Y)	
P		V
$w = (r, v)$		
$a, b \overset{R}{\leftarrow} \mathbb{Z}_p$		
$A \leftarrow pk^a, B \leftarrow g_3^a h^b$	$\xrightarrow{A, B}$	
	$\xleftarrow{e}$	$e \overset{R}{\leftarrow} \mathbb{Z}_p$
$z_1 = a + er$		
$z_2 = b + ev$	$\xrightarrow{z_1, z_2}$	check if
		$pk^{z_1} = AX^e$
		$g_3^{z_1} h^{z_2} = BY^e$

For L_{range}, we directly referred to [29], Section 4.2 to complete the protocol.

Through the Fiat-Shamir transformation, we transform the Sigma protocol of L_{right} into Π_{right}, realizing the NIZK proof of L_{right}.

5.2.3. NIZK implementation of L_{solvent}

In the BPA system, L_{solvent} is defined as follows:

$$\{(pk, C_a, C) \mid \exists \text{ sk s.t. } (pk, sk) \in R_{\text{key}} \wedge \textbf{TE.Dec}(sk, C_a - C \in V)\}.$$

Referring to the example in PGC, we decompose it into $L_{\text{solvent}} = L_{\text{ddh}} \circ L_{\text{right}}$, where L_{ddh} is defined as follows:

$$L_{\text{ddh}} = \left\{(g_1, h_1, g_2, h_2) \mid \exists\, w \text{ s.t. } \log_{g_1} h_1 = w = \log_{g_2} h_2\right\}.$$

In $L_{\mathrm{ddh}}, g_1 = g_3^{r_1}h^m/g_3^{r_2}h^m, h_1 = pk^{r_1}/pk^{r_2}$, and $g_2 = g_3, h_2 = pk$. The specific protocol of L_{ddh} is shown in Table 4.

Table 4. Sigma protocol of L_{ddh}: Knowledge proof for the discrete logarithm equation.

	(g_1, g_2, h_1, h_2)	
P		V
$a \overset{R}{\leftarrow} \mathrm{Z}_p$		
$A_1 \leftarrow g_1^a, A_2 \leftarrow g_2^a$	$\xrightarrow{A_1, A_2}$	
	$\xleftarrow{e}$	$e \overset{R}{\leftarrow} \mathrm{Z}_p$
$z = a + ew$	$\xrightarrow{z}$	check if $g_1^z = A_1 h_1^e$ $g_2^z = A_2 h_2^e$

The construction of L_{right} has been given in the previous subsection. Through the Fiat-Shamir transformation, we transform the Sigma protocol of L_{ddh} into Π_{ddh}, realizing the NIZK proof of L_{ddh}. Let $\Pi_{\mathrm{solvent}} = \Pi_{\mathrm{ddh}} \circ \Pi_{\mathrm{right}}$, which yields the NIZK proof of L_{solvent}, where $\circ$ represents the conjunction symbol.

5.3. Security analysis

In this section, we prove the security of the BPA scheme.

Theorem 1. *If the zero-knowledge proof scheme is indeed zero-knowledge, the encryption scheme is IND-CPA secure, the commitment scheme is statistically hiding, and the signature scheme used is existentially unforgeable under the chosen message attack (EUF-CMA); then, BPA achieves anonymity.*

Proof. The anonymity of our scheme can be demonstrated by the indistinguishability of the ledger; that is, the ledger does not reveal any information about the user other than what is publicly disclosed. We constructed a series of games, i.e., $Game_{real}$, $Games$ 1–3, and $Game_{sim}$, where $Game_{real}$ is the real world and $Game_{sim}$ is the simulated world. We will show that the difference between the advantage of A in $Game_{real}$ and $Game_{sim}$ is negligible. □

Game 1. In the game $Game$ 1, after sampling $b \in \{0/1\}$, CL modifies $Game_{real}$ by using simulator S to simulate the generation of user accounts. CL uses S to generate the serial number sn, the user's public key pk, private key sk, and the user's encrypted balance C. CL sends the public parameters to A and initializes two BPA oracles, O_0^{BPA} and O_1^{BPA}. Since A can directly generate the corresponding encrypted balance by using the public key, the difference between $Game$ 1 and $Game_{real}$ is zero.

Game 2. $Game$ 2 is the same as $Game$ 1, except that random strings are used to generate commitments and signatures instead of sender information. Expressly, A submits a **CreateID** query, where the public key was generated from the previous **CreateAccount**. Each oracle generates commitments and signatures in the following two steps. First, the plaintext to be committed is replaced by a random string selected from the plaintext space; second, a corresponding signature is generated for the replaced random string. If A has an advantage of η in the signature algorithm's EUF-CMA experiment and an advantage of ϵ_1 in the commitment algorithm's hiding property, then the difference between *Games* 2 and 1 is at most $\eta + \epsilon_1$.

Game 3. In *Game* 3, CL generates the corresponding encrypted ciphertext by replacing plaintext with randomly chosen strings. Specifically, for the **Send** query, a randomly chosen string from the plaintext space replaces the plaintext for generation of the corresponding encrypted ciphertext. The generated serial numbers sn are also replaced with randomly chosen strings of the same length. If A has an advantage of ϵ_2 in the encryption algorithm's IND-CPA experiment, the difference between *Games* 3 and 2 is at most ϵ_2.

Game$_{sim}$. In *Game$_{sim}$*, CL replaces the public key by using randomly chosen strings. Specifically, for the **Send** query, A randomly generates a string from the plaintext space to replace the user's public key, and S simulates the proof generation within the zero-knowledge proof. As the zero-knowledge proof is indeed zero-knowledge, the distribution of the generated proof remains the same. Assuming that A has an advantage of ϵ_3 in the discrete logarithm assumption, the difference between *Game$_{sim}$* and *Game* 3 is at most ϵ_3.

Due to the uncorrelated display response and ledger of A within the *Game$_{sim}$*, A's advantage in *Game$_{sim}$* is 0. Therefore, A's advantage in *Game$_{real}$* is

$$Adv(A) \leq \eta + \epsilon_1 + \epsilon_2 + \epsilon_3. \tag{5.1}$$

Theorem 2. *Assuming that twisted ElGamal encryption is IND-CPA secure, the randomizable signature is EUF-CMA secure, and NIZK has zero-knowledge properties, our BPA scheme achieves confidentiality.*

Proof. We prove the scheme through the following game, where S_i represents the probability of adversary A winning game i. □

Game 0. CL runs $pp \leftarrow \textbf{Setup}(1^\lambda)$ and sends pp to A. A makes multiple inquiries about O^{BPA}, and CL responds according to the rules. A chooses sender pk_a^*, receiver pk_b^*, sender commitment cm^*, sender signature σ^*, and two amounts v_0, v_1 as challenge values, where v_0, v_1 are both within the valid range. CL randomly selects a bit β, calculates $tx^* \leftarrow \textbf{Send}(pk_a^*, sk_a^*, pk_b^*, v_\beta, cm^*, \sigma^*)$, and then sends tx^* as the challenge value to A. After this, A can still make multiple inquiries about O^{BPA}, and CL responds according to the rules, but with the following exceptions: (i) Refuse to destroy honest user inquiries about pk_a^* or pk_b^*; (ii) If the accumulated inquiry amount v_{sum} causes $v_a - v_0 - v_{sum}$ or $v_a - v_1 - v_{sum}$ to be out of the valid range, refuse to answer. Finally, A outputs the guess value β'; if $\beta' = \beta$, it means that A wins this game.

Game 0 perfectly simulates the real situation. We can see that

$$Adv(A) = |Pr[S_0] - 1/2|. \tag{5.2}$$

Game 1. It is similar to *Game* 0, except that CL is randomly guessed at the beginning; that is, randomly select $s, r \in [Q_{honest}]$ and generate the corresponding pk_s, pk_r. Q_{honest} is the plaintext space. If A selects $pk_a^*/pk_b^* \neq pk_s/pk_r$, CL will abort the challenge. Assuming that W is the probability that CL does not abort the challenge, it is easy to get that $Pr[W] = 1/Q_{honest}(Q_{honest} - 1)$. The difference between *Games* 1 and 0 is

$$Pr[S_1] = Pr[S_0] \cdot Pr[W]. \tag{5.3}$$

Game 2. It is similar to *Game* 1, except that the initial parameters of the zero-knowledge proof are randomly generated in the **Setup** phase and the zero-knowledge proof is generated in simulation mode. CL can randomly generate Π, which directly reduces the zero-knowledge property of NIZK so that the following can be obtained:

$$|Pr[S_2] - Pr[S_1]| \leq negl(\lambda). \tag{5.4}$$

Game 3. Let E denote the case in which A generates a new valid confidential transaction between pk_s and pk_r. *Game* 3 is the same as *Game* 2, except that CL aborts if E occurs. By the differential Lemma, we have

$$|Pr[S_3] - Pr[S_2]| \leq E. \tag{5.5}$$

By directly reducing the EUF-CMA security of randomizable signatures, we conclude that $Pr[E] \leq negl(\lambda)$. We now prove that no PPT adversary has a non-negligible advantage in *Game* 3.

Theorem 3. *Assuming the IND-CPA security of the twisted ElGamal encryption component, all PPT adversaries A satisfy that* $|Pr[S_3] - Pr[W]/2| \leq negl(\lambda)$.

Proof. Assuming that there is a PPT adversary A with a non-negligible advantage in *Game* 2, we can construct an adversary B with the same advantage to break the IND-CPA security of the twisted ElGamal encryption component. Given the challenge (pp, pk_a, pk_b), B simulates *Game* 3 as follows: □

(1) Setup: B runs $pp \leftarrow$ **Setup**(1^λ) and sends pp to A. B randomly selects two indexes $s, r \in [Q_{honest}]$.

(2) Pre-challenge query: During the entire experimental process, A can adaptively query. B answers these queries by maintaining two lists T_{honest} and $T_{corrupt}$, both of which are initially empty.

- In the i-th query, B randomly selects an initial balance $\widetilde{v}$, randomly selects an information value $\widetilde{m}$ and a serial number sn, and performs the following operations:
  - If $i \neq s$ or r, B runs **CreateAccount**$(\widetilde{v}, sn)$ to obtain (pk, sk) and encrypted balance $\widetilde{C} \leftarrow$ **TE**.**Enc**$(pk, \widetilde{v})$, runs **CreateID**$(pk, \widetilde{m})$ to obtain (cm, σ), and then records $(pk, sk, \widetilde{C}, \widetilde{v}, cm, \sigma, sn)$ in T_{honest}.
  - If $i = s$ or r, B sets $pk = pk_s$ or pk_r, calculates $\widetilde{C} \leftarrow$ **TE**.**Enc**$(pk, \widetilde{v})$, runs **CreateID**$(pk, \widetilde{m})$ to obtain (cm, σ), and then records $(pk, \perp, \widetilde{C}, \widetilde{v}, \perp, \sigma, sn)$ in T_{honest}.
  - Finally, B returns $(pk, \widetilde{C}, \sigma, sn)$ to A.

- A queries with public key pk, signature σ, and initial encrypted balance $\widetilde{C}$. B records $(pk, \perp, \widetilde{C}, \perp, \perp, \sigma, sn)$ in $T_{corrupt}$.
- A uses pk in T_{honest} for this query. If $pk = pk_s$ or pk_r, then B aborts. Otherwise, B returns the associated sk and $\widetilde{cm}$ to A and moves the corresponding entry to $T_{corrupt}$.
- A performs a transaction query $(pk_a, pk_b, v, cm, \sigma)$ restricted by $pk_a \in T_{honest}$. Let $\widetilde{C_a}$ be the encrypted account balance of pk_a. B performs the following operations:
  (a) randomly generate $r, r' \in Z_p$ and run $C_1 \leftarrow$ **TE.Enc**$(pk_a, v; r), C_2 \leftarrow$ **TE.Enc**$(pk_b, v; r), C_3 \leftarrow$ **TE.Enc**$(Vpk, x_1\|x_2; r')$;
  (b) simulate the corresponding zero-knowledge proof Π;
  (c) if $pk_a/pk_b \neq pk_s/pk_r$, open cm as (cm_1, cm_2), randomly generate $x_1, x_2 \in Z_p$, hide both public keys as $pk'_a \leftarrow pk_a^{x_1}, pk'_b \leftarrow pk_b^{x_1}$, run $cm' \leftarrow$ **DCM.Rerand**$((cm_1, cm_2); x_1)$, $\sigma' \leftarrow$ **PS.Rerand**$(\sigma; x_1; x_2)$, and obtain the randomized identity credentials (cm', σ'); otherwise, randomly generate cm' and use the randomizable signature oracle to generate the signature σ';

  B updates the associated account status and returns tx to A.
- A performs a display inquiry $tx = (sn, cm', \sigma', pk'_a, pk'_b, C_1, C_2, C_3, \Pi)$, restricted by $pk'_a, pk'_b \in T_{honest}$. If event E occurs, then B directly aborts. Otherwise, B processes as follows:
  (a) if the participants are pk_s and pk_r and $tx \in T_{tx}$, then B returns the corresponding v;
  (b) else, let $pk_a/pk_b \neq pk_s/pk_r$; B decrypts C_a/C_b with pk_a/pk_b and sends the result to A.
- O_{inject}: A submits the confidential transaction tx. If **VerifySend**$(tx) = 1$, then B inserts $(tx, \perp)$ into T_{tx} and updates the associated account status. Otherwise, B ignores it.

(3) Challenge: A chooses the sender pk_a^*, receiver pk_b^*, commitment cm^*, and two transmission values v_0 and v_1 as challenges, requiring that $pk_a^*, pk_b^* \in T_{honest}$ and v_0 and v_1 both constitute valid transactions. If $(pk_a^*, pk_b^*) \neq (pk_s, pk_r)$, B aborts. If $cm^* = cm$, B aborts. Otherwise, B submits (v_0, v_1) to its own challenger and receives $C^* = (C_1^*, C_2^*)$, which is v_β-encrypted under (pk_a^*, pk_b^*). Let sn^* and C^* be the serial number and encrypted balance of pk_j^*, respectively. B prepares as follows:

  (a) randomly generate $x_1, x_2, r \in Z_p$, and obtain $C_3 \leftarrow$ **TE.Enc**$(Vpk, x_1\|x_2; r)$;
  (b) generate the corresponding Π^*;
  (c) query the randomizable signature oracle to obtain the signature σ^* under cm^*.

  B inserts $(tx^*, \perp)$ into T_{tx}, updates the account and status, and then sends tx^* as a challenge to A.

(4) Guess: Finally, A outputs its guess β' for β.

If B aborts when simulating A's challenger in *Game* 3, it will continue to interact with its challenger and output a random guess for β. Otherwise, it forwards A's output to its challenger. It is easy to see that B's simulation of *Game* 2 is perfect. Therefore, B's advantage over the encryption component of the twisted ElGamal is $|(1 - Pr[W])/2 + Pr[S_3] - 1/2| = |Pr[S_3] - Pr[W]/2|$, assuming the IND-CPA security of twisted ElGamal, which can be ignored in λ. This proves Theorem 3.

From the above games, we can further deduce that $|Pr[S_1] - Pr[W]/2| \leq negl(\lambda)$. Putting all of the above together, we have that $|Pr[S_0] - 1/2| = |Pr[S_1] - Pr[W]/2|/Pr[W] = negl(\lambda)$. This proves Theorem 2.

Theorem 4. *Our scheme's auditability properties can be derived directly from the soundness and zero-knowledge properties of the adopted NIZK proof system.*

Theorem 5. *Assuming that the signature scheme is EUF-CMA security and the zero-knowledge proofs are sound, our BPA scheme achieves traceability.*

Proof. Our transaction scheme contains identity credentials, so when a transaction does not comply with the policy, the validators will pass the transaction information to the regulator, and the regulator will obtain the user's true identity from the issuer based on the user's identity credentials. If the adversary A can forge a signature and pass verification, A can be exploited to break the unforgeability of the underlying signature scheme. □

6. Experimental analysis

To evaluate the performance of BPA, we conducted simulations and analysis on a computer with an Intel$^{(R)}$ Core$^{(TM)}$ i7-11700K with 3.60 GHz and 16 GB RAM. We chose to use the Barreto-Naehrig curve as the elliptic curve, whose order is 254-bit length. The experiment was implemented in the C++ programming language, and the MIRACL cryptographic library was employed to implement bilinear pairing, the dot product, and other operations.

Compared with PGC, we have added the algorithms of **CreateID** and **Trace**, which are used for regulation compliance. Users need to include identity credentials in transactions so that regulators can restore the user's identity if they discover illegal transactions. We conducted simulations and analyses of the algorithms in both BPA and PGC. We ran BPA and PGC 10 times and obtaining the average computational cost as the final result to maintain rigor. The experimental results are shown in Figures 2 and 3.

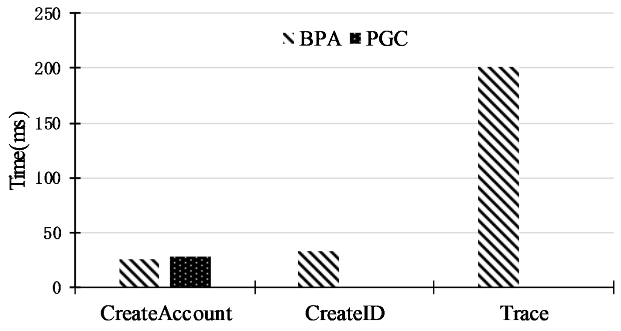

Figure 2. Computational cost of registration and tracking algorithms.

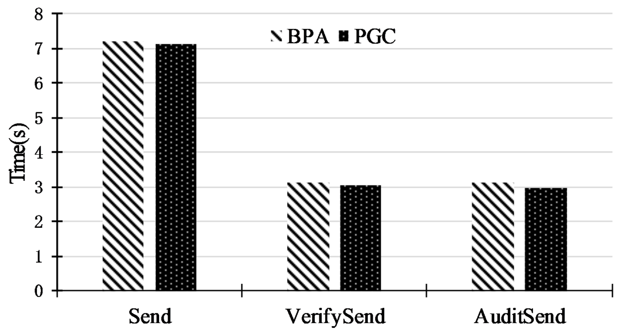

Figure 3. Computational cost of transaction and audit algorithms.

During the initialization phase, our scheme takes 75.22 s, while PGC takes 79.23 s. Due to the long time, it is not listed in Figure 2; however, compared to PGC, our scheme can start faster.

During the register phase, our protocol requires users to execute the **CreateAccount** algorithm to generate the initial user data, incurring a computational cost of 25.92 ms. Compared to PGC, an additional execution of **CreateID** is required to obtain the identity credentials. The issuer must compute the relevant commitments and signatures and return them to the user, incurring a computational cost of 33.12 ms. In the register phase, the total computational cost of our protocol amounts to 59.04 ms, while the PGC protocol totals 28.52 ms. Despite the increased computational overhead, as each user only executes **CreateID** once, the time cost is acceptable.

During the transaction phase, similar to PGC's process, our scheme requires the user to run the **Send** algorithm to generate a transaction, and the validator runs the **VerifySend** algorithm to verify the transaction and upload the verified transaction to the blockchain. Our **Send** algorithm incurs a computational cost of 7.19 s, whereas PGC's **CreateCTx** algorithm costs 7.14 s. Additionally, our **VerifySend** algorithm costs 3.13 s, while PGC's **VerifyCTx** algorithm costs 3.03 s. The increase in computational cost is due to our scheme's additional implementation of user privacy protection through the randomization of user public keys and identity credentials. As depicted in Figure 3, our experimental results indicate that the increase in computational cost is negligible after providing additional privacy protection.

During the audit and tracking phase, within PGC, the computational cost of the **Audit** algorithm amounts to 2.95 s. In our proposed scheme, the regulator primarily executes the **AuditSend** algorithm and the **Trace** algorithm. The **AuditSend** algorithm is utilized to validate transaction authenticity, incurring a computational cost of 3.12 s. In detecting illicit transactions, the regulator executes the **Trace** algorithm to ascertain the genuine identity of the violating user, incurring a computational cost of 200.61 ms. Consequently, the regulatory entity's total computational cost for the audit and tracking phase amounts to 3.32 s.

The experimental results show that the designed BPA can effectively generate user transactions and the regulator's audit process. Additionally, regulators can restore a user's identity with minimal outlay if suspicious transactions are detected.

Next, we compare this work with popular cryptocurrencies and well-known documents based on the blockchain account model. We conducted a comparative analysis, contrasting anonymity, confidentiality, auditability, and traceability, as depicted in Table 5. Bitcoin employs pseudonymity techniques to conceal transaction addresses, yet the amount of each transaction is public. Given that the blockchain records all transactions, the flow of funds can be traced by using on-chain analysis tools. The entire transaction history becomes exposed once a transaction address is identified or associated. Like Bitcoin, Ethereum transactions are public, although users use addresses rather than directly identifying information. There are also issues with on-chain analysis and address reuse. Monero is committed to providing an elevated level of privacy and anonymity. It employs techniques such as stealth addresses, ring signatures, and privacy-centric hard forks to obscure user information; it also utilizes cryptographic methods to ensure the confidentiality of transaction data. Transactions involving Monero are notably challenging to audit and trace, as its transaction records are designed to be untraceable and unlinkable. It can be observed that, among popular cryptocurrencies, either strong privacy is achieved or a certain degree of traceability is provided. However, they have not simultaneously realized the four attributes above. PGC successfully ensured transaction confidentiality and offered multiple audit functions, but it lacked support for anonymity. Solidus facilitated transactions through banks, where the bank knew the transaction amount but not the counterparty's identity. Zether protected user privacy through ring signature technology and achieved transaction confidentiality but lacked regulatory compliance. In contrast, our scheme simultaneously realizes all four of the attributes above.

Table 5. Comparison with existing schemes.

Scheme	Anonymity	Confidentiality	Auditability	Traceability
Bitcoin	◖	○	○	◖
ETH	◖	○	○	◖
Monero	●	●	○	○
PGC	◖	●	●	○
Solidus	◖	◕	◖	○
Zether	●	●	○	○
This program	◕	●	●	●

○: nonsupport; ◖ : medium; ◕: strong; ●: support;

7. Conclusions

In summary, we have introduced BPA, an account-based transaction system that successfully balances partial anonymity, auditability, and traceability, allowing users to conduct private transactions. Our approach allows the issuer to know the user's identity while ensuring the validators' transparency. In the case of violations, the cooperation between the validators, regulators, and issuer will facilitate auditing, preventing power concentration. We use zero-knowledge proofs to ensure transaction validity and regulatory compliance. Experimental results show that our scheme achieves better functionality with a similar level of efficiency. For future work, we aim to decentralize regulators and integrate them in Ethereum.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This research was funded by National Key R & D Program of China, grant number 2022YFC3303200 and Guang-dong Province Teaching Reform, grant number GDJX2020009.

Conflict of interest

The authors declare no conflict of interest.

References

1. Z. B. Zheng, S. A. Xie, H. N. Dai, X. P. Chen, H. M. Wang, Blockchain challenges and opportunities: A survey, *Int. J. Web. Grid. Serv.,* 2018, 352–375. https://doi.org/10.1504/IJWGS.2018.095647
2. *Google Scholar*. Available from: `https://scholar.google.com/`.

3. *CoinMarketCap*. Available from: `https://coinmarketcap.com/`.
4. R. B. Grinberg, Bitcoin: An innovative alternative digital currency, *J. Amer. Math. Soc.,* 2012. http://dx.doi.org/10.1090/S0894-0347-1992-1124979-1
5. E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, S. Capkun, Evaluating user privacy in bitcoin, *Financ. Cryptogr. Data Secur.,* **7859** (2013), 34–51. https://doi.org/10.1007/978-3-642-39884-1_4
6. D. Hopwood, S. Bowe, T. Hornby, N. Wilcox, *Zcash protocol specification*, GitHub: San Francisco, CA, USA, **4** (2016), 220.
7. G. Maxwell, *Coinjoin: Bitcoin privacy for the real world*. Available from: `https://bitcointalk.org/index.php?topic=279249`.
8. S. F. Sun, M. H. Au, J. K. Liu, T. H. Yuen, Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero, *Comput. Secur.,* **10493** (2017), 456–474. https://doi.org/10.1007/978-3-319-66399-9_25
9. C. M. Christensen, T. Hall, K. Dillon, D. S. Duncan, Know your customers' jobs to be done, *Harvard Bus. Rev.,* **94** (2016), 54–62.
10. J. Ferwerda, The economics of crime and money laundering: Does anti-money laundering policy reduce crime?, *Rev. Law. Econ.,* **5** (2009), 903–929. https://doi.org/10.2202/1555-5879.1421
11. R. Auer, R. Böhme, The technology of retail central bank digital currency, *J. Amer. Math. Soc.,* 2020. http://dx.doi.org/10.1090/S0894-0347-1992-1124979-1
12. T. P. Pedersen, *Non-interactive and information-theoretic secure verifiable secret sharing*, CRYPTO 1991: Advances in Cryptology, Springer, Berlin, Heidelberg, **576** (2001), 129–140. https://doi.org/10.1007/3-540-46766-1_9
13. *Mimblewimble*, Tom Elvis Jedusor. Available from: `https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.txt`.
14. N. Van Saberhagen, CryptoNote v 2.0, *J. Amer. Math. Soc.*, 2013.
15. E. Fujisaki, K. Suzuki, *Traceable ring signature*, PKC 2007: Public Key Cryptography, Springer, Berlin, Heidelberg, **4450** (2007), 181–200. https://doi.org/10.1007/978-3-540-71677-8_13
16. T. H. Yuen, S. Sun, J. K. Liu, M. H. Au, M. F. Esgin, Q. Z. Zhang, *Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security*, Financial Cryptography and Data Security, Springer, Cham, **12059** (2020), 464–483. https://doi.org/10.1007/978-3-030-51280-4_25
17. P. Fauzi, S. Meiklejohn, R. Mercer, C. Orlandi, *Quisquis: A new design for anonymous cryptocurrencies*, ASIACRYPT 2019: Advances in Cryptology, Springer, Cham, **11921** (2019), 649–678. https://doi.org/10.1007/978-3-030-34578-5_23
18. I. Miers, C. Garman, M. Green, A. D. Rubin, *Zerocoin: Anonymous distributed e-cash from bitcoin*, In: 2013 IEEE Symposium on Security and Privacy, IEEE, Berkeley, CA, USA, 2013, 397–411. https://doi.org/10.1109/SP.2013.34
19. A. De Santis, S. Micali, G. Persiano, *Non-interactive zero-knowledge proof systems*, CRYPTO 1987: Advances in Cryptology, Springer, Berlin, Heidelberg, **293** (1988), 52–72. https://doi.org/10.1007/3-540-48184-2_5

20. E. B. Sasson, A. Chiesa, 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, Berkeley, CA, USA, 2014, 459–474. http://dx.doi.org/10.1109/SP.2014.36
21. N. Bitansky, R. Canetti, A. Chiesa, E. Tromer, *From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again*, ITCS '12: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, 2012, 326–349. https://doi.org/10.1145/2090236.2090263
22. M. Szydlo, *Merkle tree traversal in log space and time*, EUROCRYPT 2004: Advances in Cryptology, Springer, Berlin, Heidelberg, **3027** (2004), 541–554. https://doi.org/10.1007/978-3-540-24676-3_32
23. C. Garman, M. Green, I. Miers, *Accountable privacy for decentralized anonymous payments*, FC 2016: Financial Cryptography and Data Security, Springer, Berlin, Heidelberg, 2017, 81–98. https://doi.org/10.1007/978-3-662-54970-4_5
24. E. Cecchetti, F. Zhang, Y. Ji, A. Kosba, A. Juels, E. Shi, *Solidus: Confidential distributed ledger transactions via PVORM*, CCS '17: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, 701–717. https://doi.org/10.1145/3133956.3134010
25. Y. Li, G. Yang, W. Susilo, Y. Yu, M. H. Au, D. X. Liu, *Traceable monero: Anonymous cryptocurrency with enhanced accountability*, In: IEEE Transactions on Dependable and Secure Computing, IEEE, **18** (2019), 679–691. http://dx.doi.org/10.1109/TDSC.2019.2910058
26. N. Narula, W. Vasquez, M. Virza, *zkLedger: Privacy-Preserving auditing for distributed ledgers*, In: Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI '18), USA, 2018, 65–80.
27. P. Chatzigiannis, F. Baldimtsi, *Miniledger: Compact-sized anonymous and auditable distributed payments*, ESORICS 2021: 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, M. H. Au, *PGC: decentralized confidential payment system with auditability*, ESORICS 2020: Computer Security, Springer, Cham, **12308** (2020), 591–610. https://doi.org/10.1007/978-3-030-58951-6_29
29. 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, San Francisco, CA, USA, 2018, 315–334. https://doi.org/10.1109/SP.2018.00020
30. B. Bünz, S. Agrawal, M. Zamani, D. Boneh, *Zether: Towards privacy in a smart contract world*, FC 2020: Financial Cryptography and Data Security, Springer, Cham, **12059** (2020), 423–443. https://doi.org/10.1007/978-3-030-51280-4_23
31. Z. Guan, Z. Wan, Y. Yang, Y. Zhou, B. Huang, *BlockMaze: An efficient privacy-preserving account-model blockchain based on zk-SNARKs*, IEEE Transactions on Dependable and Secure Computing, IEEE, **19** (2020), 1446–1463. https://doi.org/10.1109/TDSC.2020.3025129
32. E. Androulaki, J. Camenisch, A. D. Caro, M. Dubovitskaya, K. Elkhiyaoui, B. Tackmann, *Privacy-preserving auditable token payments in a permissioned blockchain system*, AFT '20: Proceedings of the 2nd ACM Conference on Advances in Financial Technologies, 2020, 255–267. https://doi.org/10.1145/3419614.3423259

33. I. Damgård, C. Ganesh, H. Khoshakhlagh, C. Orlandi, L. Siniscalchi, *Balancing privacy and accountability in blockchain identity management*, CT-RSA 2021: Topics in Cryptology, Springer, Cham, **12704** (2021), 552–576. https://doi.org/10.1007/978-3-030-75539-3_23

34. O. Goldreich, Secure multi-party computation, *Manuscript Preliminary Version*, 1998, 78–110.

35. M. M. Islam, M. K. Islam, M. Shahjalal, M. Z. Chowdhury, Y. M. Jang, *A low-cost cross-border payment system based on auditable cryptocurrency with consortium blockchain: Joint digital currency*, In: IEEE Transactions on Services Computing, IEEE, **16** (2022), 1616–1629. https://doi.org/10.1109/TSC.2022.3207224

36. K. Wüst, K. Kostiainen, N. Delius, S. Capkun, *Platypus: A central bank digital currency with unlinkable transactions and privacy-preserving regulation*, CCS '22: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, 2022, 2947–2960. https://doi.org/10.1145/3548606.3560617

37. A. Tomescu, A. Bhat, B. Applebaum, I. Abraham, G. Gueta, B. Pinkas, et al., Utt: Decentralized ecash with accountable privacy, *Cryptology ePrint Archive,* 2022, in press. Available from: `https://ia.cr/2022/452.`

38. L. Xue, D. Liu, J. Ni, X. Lin, X. S. Shen, *Enabling regulatory compliance and enforcement in decentralized anonymous payment*, IEEE Transactions on Dependable and Secure Computing, IEEE, **20** (2022), 931–943. https://doi.org/10.1109/TDSC.2022.3144991

39. C. Lin, X. Huang, J. Ning, D. He, *Aca: Anonymous, confidential and auditable transaction systems for blockchain*, IEEE Transactions on Dependable and Secure Computing, IEEE, **20** (2022), 4536–4550. https://doi.org/10.1109/TDSC.2022.3228236

40. A. Menezes, *The discrete logarithm problem*, Elliptic Curve Public Key Cryptosystems, Springer, Boston, MA, **234** (1993), 49–59. https://doi.org/10.1007/978-1-4615-3198-2_4

41. D. Pointcheval, O. Sanders, *Short randomizable signatures*, CT-RSA 2016: Topics in Cryptology, Springer, Cham, **9610** (2016), 111–126. https://doi.org/10.1007/978-3-319-29485-8_7

AIMS Press