KAIME : Central Bank Digital Currency with Realistic and Modular Privacy

Ali Dogan[1,2] and Kemal Bicakci[1]

[1] Computer Engineering Department and Informatics Institute, Istanbul Technical University, Istanbul, Turkiye kemalbicakci@itu.edu.tr
[2] Informatics and Information Security Research Center (BILGEM), TUBITAK, Kocaeli, Turkiye doganali@tubitak.gov.tr

Keywords: CBDC, privacy, zero knowledge proofs, threshold, anonymity set, realistic privacy, regulation compliance, AML, KYC, CFT

Abstract. Recently, with the increasing interest in Central Bank Digital Currency (CBDC), many countries have been working on researching and developing digital currency. The most important reasons for this interest are that CBDC eliminates the disadvantages of traditional currencies and provides a safer, faster, and more efficient payment system. These benefits also come with challenges, such as safeguarding individuals' privacy and ensuring regulatory mechanisms. While most researches address the privacy conflict between users and regulatory agencies, they miss an important detail. Important parts of a financial system are banks and financial institutions. Some studies ignore the need for privacy and include these institutions in the CBDC system, no system currently offers a solution to the privacy conflict between banks, financial institutions, and users. In this study, while we offer a solution to the privacy conflict between the user and the regulatory agencies, we also provide a solution to the privacy conflict between the user and the banks. Our solution, KAIME [*] has also a modular structure. The privacy of the sender and receiver can be hidden if desired. Compared to previous related research, security analysis and implementation of KAIME is substantially simpler because simple and well-known cryptographic methods are used.

1 INTRODUCTION

Blockchain technology has gained popularity with the emergence of cryptocurrencies. Many people have started to adopt and use these cryptocurrencies. Motivated by the prevalence and success of blockchains, there is a race between central banks for the development of Central Bank Digital Currency (CBDC). CBDCs could be a revolution in terms of payment systems worldwide. Several central banks, including the Swedish central bank [21] and the Bank of England [6], have shown interest in developing their own digital currencies. The People's Bank of China [27] has already begun testing the digital yuan. Moreover, several

[*]The name given to the first banknote issued by the Ottoman Empire

central banks, in collaboration with the BIS, have described the key concepts and characteristics of a CBDC. However, this revolution also brings problems, such as protecting private life and harmonizing regulations. How digital currencies can balance privacy and regulation is one of the focuses of recent research.

Many people are worried that introducing Central Bank Digital Currencies (CBDCs) may result in the central bank having continuous access to transactional data, making it a "panopticon." This concern is not unique to CBDCs and has also been expressed regarding first-generation cryptocurrencies like Bitcoin and Ethereum, which are only pseudonymous. To address this, privacy-enhanced cryptocurrencies such as ZCash [22] and Monero [24] were developed to provide a higher level of anonymity by hiding the value of transactions and making them unlinkable. However, this anonymity could also attract those who wish to use these systems for illegal activities, such as money laundering and financing terrorism. As a result, privacy-preserving systems using such techniques may pose challenges in regulatory compliance settings.

Related Work. Chaum introduced the initial framework for anonymous electronic cash in his work [11], which emphasized protecting the sender's anonymity while revealing the recipient's identity and the amount of money transferred. With this system, a user can obtain a coin from a bank by creating a distinctive serial number and obtaining a blind signature to keep the serial number concealed from the bank. The user can then unblind the signature and use the coin for payments. When a merchant receives payment, they can deposit the coin at the bank, which will verify whether the serial number has been utilized previously. If the serial number is already used, the payment is rejected; if not, it is accepted. Camenisch et al. [10] introduced a method of electronic payment based on tokens, where the bank can impose specific regulations such as payment limits for individual users. Despite this, the privacy of those who send transactions is maintained; however, the recipient's identity and the payment amount are revealed.

Another related work PRCash is more relevant to our solution that addresses the privacy conflict [25], presents a solution that utilizes ZKPs to enable efficient implementation of a receiving limit for anonymous transactions within a specific time interval or epoch. Additionally, the regulation mechanism of PRCash requires linking multiple transactions within a time limit, which can potentially compromise user privacy.

Androulaki et al. presented a token management system that is both privacy-preserving and auditable [2]. Their proposed system employs a UTxO (Unspent Transaction Output) model in a permissioned blockchain. Their solution is tailored for business-to-business scenarios and does not provide a comprehensive approach to regulatory compliance.

Gross et al. proposed a modified version of Zerocash to create a "privacy pool" for CBDC [16]. This modified Zerocash protocol [22] can ensure the privacy of CBDC transactions by hiding the identities of the transacting parties while maintaining the integrity of the CBDC system. It utilizes proofs of inclusion in a Merkle tree to verify transactions. This means the system uses a Merkle tree data structure to efficiently prove that a transaction is valid and that its inputs have not been previously spent.

Wüst et al. introduced Platypus, a privacy-preserving and centralized payment system [26]. Platypus is not decentralized, which means it cannot continue to function effectively in the event of a single point of failure.

Tomescu et al. proposed a decentralized payment system known as UTT, which relies on a Byzantine fault-tolerant infrastructure [23]. Additionally, UTT limits the amount of money that can be anonymously sent monthly.

PEReDi [17] provides support for regulatory compliance, including Know Your Customer (KYC), Anti-Money Laundering (AML), and Combating Financing of Terrorism (CFT) requirements. In the PEReDi, a committee of several authorities can revoke privacy or trace transactions from a specific user. The committee does so by decrypting the ciphertext stored in the ledger. Both users must be online for the transaction to occur on PEReDi.

The comparison of KAIME and related works is given in Table 1 and Table 2. Lee's framework has been expanded [19], and new comparable features have been added to the table.

Contributions. The paper presents the following contributions:

1. To the best of our knowledge, we propose a CBDC system that does not only address the privacy conflict between the user and regulatory agencies but also resolves the privacy conflict between the bank and the user by including all stakeholders (users, banks, financial institutions, regulatory agencies, central bank) for the first time. This system also supports regulatory mechanisms such as KYC, AML, and CFT, which are critical requirements that should be included in a CBDC system.
2. In KAIME, sender and receiver privacy can be added or removed as features from the system depending on the requirements. This adds modularity to our solution.
3. Since simple and known cryptographic algorithms are used, security analysis and implementation of KAIME is much easier than other related works. In addition, the zero-knowledge proofs can work without needing a trusted party.

2 OVERVIEW

In this section, we present a summary of our solution. We begin by discussing our motivation and our requirements. Next, we describe our system model and then give the details of the cryptographic techniques we have employed to develop our solution.

2.1 Motivation

In a report by the Swiss National Bank [12], "mass surveillance" is specifically identified as a potential risk associated with a CBDC. This underscores the

Reference	UTxO or Account Based	Sender Privacy	Receiver Privacy	Transaction Privacy	Crytographic Technique
[25]	UTxO	Yes	Yes	Yes	ZKP , ElGamal Enc., Ped. Com.
[2]	UTxO	Yes	Yes	Yes	VRF, ElGamal Enc., PS Sig., Ped. Com.
[16]	UTxO	Yes	Yes	Yes	Commitment, ZKP
[26]	Account	Yes	Yes	Yes	Commitment, ZKP
[23]	UTxO	Yes	Yes	Yes	MPC, Commitment, ZKP
[17]	Account	Yes	Yes	Yes	MPC, ZKP, PS Sig., Elgamal Enc.
KAIME	**Account**	**Optional**	**Optional**	**Yes**	**Elgamal Enc., ZKP, MPC, Anon. Set**

Table 1. The first column shows whether the system is UTxO or account based. The last column shows the cryptographic techniques used. The other columns show whether the sender, receiver, and transaction details are hidden.

importance of ensuring strong privacy protections. Furthermore, a survey conducted by the European Central Bank [5] revealed that privacy was considered the most critical aspect of a CBDC.

While CBDCs are expected to provide a critical feature, such as privacy, CBDCs must accommodate some regulatory requirements for financial stability and government security. Regulatory requirements for CBDCs are the enforcement of anti-money-laundering (AML), know-your-customer (KYC), and counter-financial-terrorism (CFT) [1]. On the other hand, this contradicts the objective of enhancing payment privacy.

There is a suggestion that this conflict can be resolved by allowing anonymous payments up to a specific limit per unit time [4]. Previous works have proposed this idea [25], [15], [26]. The idea does not meet the requirements. Government officials may not mind evading a \$100 tax, but when it comes to a criminal or murderer, payment information is critical. Various suggestions for solving this conflict are summarized in the related work section. These solutions include various cryptographic techniques such as zero-knowledge proof, commitment scheme, threshold cryptography, and blind signature. In [3], authors stated that these solutions do not explicitly address the privacy conflict between stakeholder groups (merchants, banks and payment providers, government). In the article, Auer et al. mentioned not only the privacy conflict between the user and the government but also the high level of conflict between other groups. They have also divided the situations in which the user's data should be accessed and the stakeholder who wants to access it, layer by layer.

Based on the motivation to provide both the privacy of users and regulatory requirements and the idea of bringing other stakeholders into the system, our first aim is to design a system in which a person suspected by the regulatory agencies can track all transactions retrospectively and provide this tracking by exceeding

References	Regulation Mechanism	Solution to Privacy Conflict Between User- Reg. Agen.	Solution to Privacy Conflict Between User- Fin. Ins.	For CBDC?
[25]	Balance Limit	Yes	No	No
[2]	Single Reg. Agency	Yes	No	No
[16]	Balance Limit	Yes	No	Yes
[26]	Balance Limit	Yes	No	Yes
[23]	Balance Limit	Yes	No	Yes
[17]	More than One Reg. Agency	Yes	No	Yes
KAIME	**More than One Reg. Agency**	**Yes**	**Yes**	**Yes**

Table 2. The table compares the related work dealing with the privacy conflict and our solution. The second column shows under what conditions and by whom the regulation mechanism is executed. The third and fourth column show for which stakeholders a solution to the privacy conflict is offered. The last column shows whether the papers were written for CBDC purposes.

the threshold number. Our second goal is to include banks and companies that use financial data in the system and to solve the privacy conflict between them and the user.

CBDC can be recorded in a distributed ledger using blockchain technology. This technology is used to ensure that CBDCs are traded in a secure, transparent, and reliable manner. Blockchain technology can help prevent fraudulent or misleading transactions as transactions are recorded irreversibly. In addition to such benefits, we use a permissioned blockchain to easily access the transaction details of the stakeholders, except the users in the system, and to prevent a single point of failure.

2.2 Balance Between Soft and Hard Privacy

Auer et al. divided the privacy methods in CBDC systems into three [3]. These are hard privacy, soft privacy, and privacy with a balance between soft and hard. The stakeholders in the system have been divided into shells according to the monitoring status of the transactions and the request to review the transactions.

Hard privacy argues that all stakeholders in the system cannot see the transactions and that only the person with the private key can see the plaintext, that is, the user. Unfortunately, this will lead to the disappearance of regulatory mechanisms and is undesirable for CBDCs. On the other hand, soft privacy addresses the ability of payment information to move freely between different parties yet still protects it from external attacks through point-to-point encryption. While a system like this can be highly effective in terms of efficiency, its privacy features will not differ from those of current payment networks. As a result, it may not meet the privacy needs of users who are particularly concerned about protecting their information.

The innermost ring of stakeholders divided into rings is the banks. Auer et al. argued that banks should see the transaction details in the balance between soft and hard privacy. We disagree with this view, but we are developing a solution where banks can see the transaction details if they receive approval from the user. They also said that hard privacy techniques could be used for other stakeholders. We use hard privacy techniques between regulatory agencies and users in KAIME; we use a technique that converges to soft privacy, although we cannot say precisely soft privacy between the bank and the user.

2.3 Security and Privacy Requirements

In this section, we define the privacy and security requirements that should be in KAIME.

Transaction Integrity. It should not be possible for any person to transact on behalf of someone else and change their balance. Following a successful transaction between two users, it is imperative to update the accounts of both parties accurately, taking into account all relevant parameters. The transaction must occur even if the receiving party is offline. The balance increases and decreases on the sender, and receiver side must be the same.

Regulatory Mechanism. Regulation mechanisms such as KYC, AML, and CFT should be included in the system. Regulatory agencies should be able to see the details of the process and review them retrospectively when needed. In order for these mechanisms to be quickly processed, the sender should not be stored encrypted in the ledger.

Bank and Financial Institutions Tasks. The duties of these institutions in traditional systems should also be provided in the solution. The user should be able to share the details of the past transaction with the institution without deceiving the institution. However, the institution cannot monitor past transactions without user permission.

Identity and Transaction Privacy. When a transaction is given, the recipient and the transaction value should not be detected in cases other than auditing. In addition, the user balance should be kept encrypted in the ledger, and the balance should not be detected.

Unlinkability Given a transaction, the ownership of the assets used by the current transaction should not be linked to past transactions. It should not be possible to connect the receiver to another payment in the same system where the sender or receiver is located.

Accountability Once a sender has made a payment, she should not be able to deny it later.

2.4 Stakeholders & Roles

In this section, we describe the entities involved and their respective roles. We would like to point out that the central bank, banks, and regulatory agencies are responsible for the operation of the blockchain.

- **Central Bank:** The digital currency is issued by the central bank, which is accountable for the monetary policy and has the authority over the monetary supply at any point. However, the central bank has no control over the status of all users' accounts and lacks trust when it comes to privacy due to the possibility of mass surveillance. This means that the central bank cannot disclose the transferred values associated with a particular transaction. For the role of the central bank, we refer to [12].

- **Users:** As with any digital currency system, users of the system can take on the role of either the sender or the recipient when participating in a transaction involving digital currency. Users have no choice against regulatory agencies to protect the privacy of their past transactions. If the regulatory agencies decide that the user is a potential criminal, they can abort the user's privacy with the help of threshold cryptography. However, users have the ability to allow banks and financial institutions to review transactions.

- **Banks and Financial Institutions:** Banks are responsible for making the user registration process. In the traditional banking system, banks also have various responsibilities, such as giving a credit score to the user and determining a credit card limit. In order to perform these functions, banks need to learn the balance and past transactions of the user. They can perform this operation cryptographically in line with the user's consent. Likewise, for financial institutions to fulfill their duties in the traditional system, they need to access the user's transaction details. The user can share transaction details with financial institutions upon request.
- **Regulatory agencies:** Our approach involves entrusting a group of authorized institutions, which we call regulatory agencies, with the task of conducting different audit procedures required for ensuring regulatory compliance. Regulatory agencies can access the data of the user's transactions in case of doubt by joint decision. They can translate the encrypted transaction data into plaintext with the help of threshold cryptography and access the transaction details.

2.5 High-level Overview

Each user within the system possesses a wallet that is used for storing their current balance, the encryption private key of the user sk_U, the signature private key of the user, and the public key of regulatory agencies pk_R. sk_U is used to access the plaintext of the encrypted balance of the user in the ledger and create zero-knowledge proofs, pk_R is used in threshold Elgamal encryption [14] [20] for regulatory agencies.

User registration is done by banks. The currency issuance function is performed by the central bank. The amount of v encrypted with the user's public key pk_U is added to the user's encrypted balance in the ledger with homomorphic encryption.

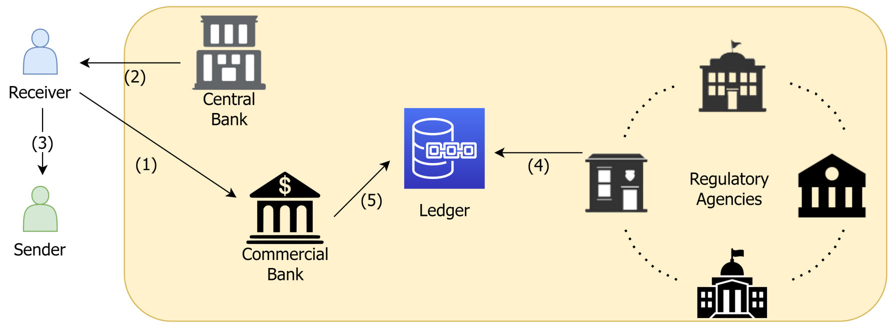

Fig. 1. The System consists of commercial banks (or any financial institutions)that are responsible for user registration and traditional bank tasks, the central bank that is responsible for currency issue and monetary policy, and regulatory agencies that are responsible for regulatory compliance. All entities are responsible for executing the validity of transactions and the blockchain network. The direction of the arrows and the numbers in the figure do not indicate a specific order. The purpose of the arrows is to show the functions that take place between the entities. (1) represents User Registration, (2) represents Currency Issue, (3) represents Payment, (4) represents Abort Transaction Privacy, and (5) represents Abort Transaction Privacy for Bank.

KAIME is built on an account-based system, similar to Zether [8]; the balances of the users are encrypted on the ledger. In the payment function, the sender encrypts the amount v that he wants to send with the public key of the receiver and encrypts with his public key. Then, the sender proves that the encrypted balance in the ledger is more than the amount he wants to send while performing the payment transaction. The sender also encrypts the amount v to be sent with the public key of the regulatory agencies. He also provides proof that the v values in the ciphertexts are the same similar to the currency issue function. However, due to the requirement of verifying that three different ciphertexts can be decrypted to the same value, she generates two proofs of equality. After checking the validity of the proofs, the balance of the sender's account decreases homomorphically (over the ciphertext), and the balance of the receiver increases homomorphically. Finally, the ledger is updated.

In KAIME, other stakeholders, banks, and financial institutions can access the user's past transactions and balance by getting approval from the user. The details are explained in Section 3. With past transaction details, banks benefit from various usage areas such as credit scores.

Regulatory agencies can run the Abort Transaction Privacy function to de-anonymize the user and examine past transactions. By exceeding the number of thresholds t similar to [17], the user's balance on the ledger can be accessed, and his past transactions can be accessed.

3 SYSTEM DETAILS

In the following section, we will provide a more detailed explanation of our solution. Our approach involves the integration of various cryptographic techniques as fundamental components. For additional clarity, we have included further information about these techniques in Appendix.

3.1 System Initialization

We use a group G with generator g in our solution $G = \langle g \rangle$. The following steps are performed by the entities involved in the initialization process.

Regulatory Agencies. An encryption keypair $(pk_R,\, sk_R) = (g^{sk_R},\, sk_R)$ for Threshold Elgamal encryption is generated by the regulatory agencies, and the public keys are made available to the system setup. We apply Shamir Secret Sharing to the private key so that a single agency does not have the authority to see the transaction details. In KAIME, there are n regulatory agencies and we will display the private key of each of them with $sk_{R,i}$ and the threshold number of regulatory agencies required to construct a private key is t. This means that in order to see the details of the transactions, at least t agency agencies must reach an agreement.

Central Bank. The central bank is registered in the ledger, and the signature key pair is generated like a user. These keys will be used for validation in the currency issue function.

Banks and Financial Institutions. Banks and financial institutions will register in the system as a user. They are responsible for the operation of the blockchain and have access to the user's ciphertext.

3.2 User Registration

By verifying the KYC step, the user is registered to the system through the bank. Then, the user needs to generate two key pairs. One is for signature; the other is to keep the balance in the ledger as encrypted and increase the received balances homomorphically encrypted balance.

The bank then creates the ElGamal ciphertext with a balance of 0 using the encryption public key pk_U created by the user for encryption and saves it to the ledger. The bank performs the same operation for the public key of regulatory agencies. For simplicity, we have only shown the operation performed with the user's public key.

$$(c_{U,1}, c_{U,2}) = (g^r,\, g^0 \cdot pk_U^r)$$

The bank also adds the proof that the plaintexts of the ciphertexts are 0 (see Appendix).

3.3 Currency Issue

The central bank encrypts the value v, which it wants to issue, with the public key of the user and the public key of the regulatory agencies. Then, the central bank creates an equality proof (see Appendix) that the values in these two ciphertexts are the same and signs the transaction and proofs, and then sends these to the ledger. The smart contract updates the ledger by checking the validity of the proofs and signature. For simplicity, we have only shown the operation performed with the user's public key.

Assume $(c_{U,1},\, c_{U,2})$ represents the encrypted current balance of the user in the ledger, and $(c'_{U,1},\, c'_{U,2})$ represents the encrypted value of the value to be issued by the central bank. After the request is authenticated, the ciphertexts are added, and the user's balance is updated.

3.4 Payment

To start the payment process, the sender must first have the public key of the receiver and the public key between the receiver and the bank. This initial step can be accomplished with a QR code. When the sender wants to send v value to the receiver, he encrypts v under the public keys of the receiver, the sender, between bank and receiver, between bank and sender, and regulatory agencies.

To ensure that three ciphertexts are decrypted to the same value, the sender creates two proofs of equality (see Appendix). In order to prevent any possibility of creating value out of thin air and verify that the sender has sufficient balance in her account, she also adds two range proofs. This enables us to effectively merge ElGamal encryptions with Bulletproofs-based range proofs [9]. To prepare these range proofs, he needs to keep track of the random values that were used while encrypting the amounts. However, the user does not need to store these random values, he refreshes the ciphertext in the ledger before performing the operation. In doing so, it uses a customized proof of equality. In this way, he will have the random value in the ciphertext in the ledger. This method was first used in PGC [13]. Finally, the sender signs the transaction with the private key and sends the proofs and ciphertexts to the blockchain.

Before starting any payment, the sender obtains his encrypted balance from the ledger and decrypts it using his secret key to see his balance. In order to send value v to the receiver, the sender computes the following:

1. The sender encrypts the value v with his public key, the sender's public key, and regulator agencies' public key $c^1 = \mathrm{Enc}(v,\, pk_s)$, $c^2 = \mathrm{Enc}(v,\, pk_r)$, $c^3 = \mathrm{Enc}(v,\, pk_r)$.
2. The sender creates two proofs of equality. One shows that the plaintexts in c^1 and c^2 are the same, and the other shows that the plaintexts in c^2 and c^3 are the same. In this way, it shows that all ciphertexts commit to the same message.
3. The sender creates range proofs to show that he has sufficient balance and the encrypted value is between 0 and $2^{32} - 1$.

4. The sender signs the ciphertexts and the proofs. Then he sends these to the ledger.

After proofs are verified, the sender's encrypted balance decreases homomorphically, and the receiver's encrypted balance increases homomorphically.

3.5 Abort Transaction Privacy

Regulatory agencies apply the abort privacy transaction function on the transaction or balance related to their shared ElGamal encryption keys to see the content of transactions they consider suspicious. Let $(c_{R,1},\, c_{R,2}) = (g^r,\, g^b \cdot pk_R^r)$ is the user's balance encrypted with the regulatory agencies public key.

We denote that $sk_{R,i}$ is the private key of decryption for i-th agency and $pk_{R,j} = g^{sk_{R,i}}$ is the corresponding public key share. Regulatory agencies can access plaintext using these public keys. Details are described in the Appendix to preserve the integrity of the paper. Also, they can apply the same function not only for the balance but also for accessing the transaction details.

3.6 Abort Transaction Privacy for Bank and Financial Institutions

When the user wants to receive service from the bank or institution, the institution that will provide the service needs the detail of the user's past transactions. The user can give the encryption private key to the bank in order to present the contents of the encrypted transactions on the ledger to the bank, but in such a scenario, the bank will have the ability to see the future transactions of the user.

Firstly, a bank or financial institution creates a one-time public key for this function and sends this to the user. The user encrypts the balance and values of all previous transactions with this public key. After this step, the user creates equality proof for all past encrypted transactions and the encrypted texts it creates with the one-time public key and sends it to the bank. The reason for creating this proof is to prevent the user from cheating the bank. After the bank has verified the proofs, it can access the user's transaction values and balance.

4 TRUST ASSUMPTION

Our paper does not address protection for network-based deanonymization attacks, such as linking an IP address to multiple transactions. Clients who wish to protect themselves against such attacks can employ measures.

We make the assumption that the clients engage in communication through secure channels, and all cryptographic operations employed conform to the standard definitions of their security: It is assumed that signatures are unforgeable, zero-knowledge proofs provide soundness and are zero-knowledge, and encryption is CPA-secure.

We assume that regulatory agencies do not want to see the transaction details arbitrarily. They run the abort privacy transaction function only for people and transactions they think are suspicious.

5 ANONYMITY

In this section, we will give an anonymous version of our solution. This version not only hides the transferred amount but also hides the receiver. However, it comes at the expense of additional costs. It is worth noting that the size of the zero-knowledge proof required for a transfer will increase linearly the size of the anonymity set. A similar solution was used in Zether [8]. Also, the sender is hidden in Zether. We do not hide the sender to make the regulation mechanism easy.

Because of the limited amount of available space, We will only introduce it as an overview. An anonymous transaction allows a sender who wishes to send a value v to a receiver with a public key pk_r, to conceal both the identity of the receiver among a larger set of users with public keys $\{pk_1, pk_2, ...pk_n\}$, as well as the transferred value v. The sender sends $3n$ ciphertexts, and all of them encrypt 0 except three. Only three ciphertexts represent the real transaction; the rest are fake transactions. Since the sent values in the fake transaction are 0, the balance of the sender and the users in the anonymity set does not change.

By using ring signatures and ZKPs, both the sender, the receiver, and the transaction details can be hidden. However, we do not recommend hiding the sender so that the regulation mechanism can work better, although it may differ according to the requirements.

6 SECURITY ANALYSIS

6.1 Transaction Integrity

Firstly, we will address the integrity of our solution. As our system operates as a digital currency platform, it is imperative that only authorized users can spend balances. That balance cannot be spent more than once.

Inspired by the Platypus [26], we define the "Transaction Forgery Game" and then prove that an adversary can win this game with negligible probability.

Definition. (Transaction Forgery Game) The game consists of an adversary called A and a challenger known as C. The challenger C has the advantage of having access to an oracle O, which simulates the behavior of honest participants within the system. The game is played in a particular sequence which is described as follows:

1. C starts the system and all cryptographic primitives.
2. After this, the adversary A has the ability to create private keys and corresponding accounts with a balance of his choice. Once A has created these accounts, the oracle O confirms their validity by registering. Additionally, the adversary A has the option to request the oracle O to set up more users in the system with balances chosen by A.
3. The adversary A has the freedom to create and submit any transaction they wish and also interact with any account managed by the oracle O. This

means A can send or receive transactions from any account within the game, regardless of whether it was created by A or not.

4. To succeed in the game, the adversary A must create a transaction that can be accepted by the oracle O. This can be achieved by either creating a transaction in which A has no control over either the sender or the recipient account, A has no control over the sender account, or creating a transaction with a value greater than the initial balance set up by A.

Claim. It is impossible for an adversary A to win the transaction forgery game with a negligible probability.
Sketch of Proof. We can prove this claim with various cases. Even if the adversary has the balance information in the accounts, A cannot create a valid transaction because he does not have private keys. Suppose the adversary also has signed transactions from oracle O. In that case, A will not be able to create a valid signature, as we assume that the digital signature algorithm we use is EUF-CMA.

If the attacker creates a transaction value more than the balance of the accounts he created, it contradicts our assumptions. We assume that range proof has a soundness property.

6.2 Regulation Integrity

Given that our solution incorporates regulatory mechanisms, it becomes imperative for us to examine the integrity of these mechanisms. Therefore, we give the following claim:

Claim. It is impossible for any user to generate a transaction violating the regulatory mechanism.
Sketch of Proof. The proof of this claim is covered by the soundness feature of the zero-knowledge proofs we use. Suppose the user does not encrypt the transaction values using the public key of Regulatory Agencies. In that case, the transaction will not be valid during the transaction verification and will not be written to the ledger.

6.3 Privacy Against Regulatory Agencies

Although our solution empowers regulatory agencies to monitor transactions, it requires some restrictions, as mentioned in the previous sections.

Claim. A single regulatory agency could not see the transaction contents and user balance.
Sketch of Proof. Our solution's use of threshold encryption confirms this claim. Since the private keys that will convert the encrypted texts into plaintext are shared, a single regulatory agency cannot see the content of the plaintexts correctly.

6.4 Privacy Against Banks

If the user permits the bank, the bank can see the transaction details. However, the bank should not be able to see the details of the transactions without obtaining permission to ensure privacy between the user and the bank.

Claim. A bank could not see the transaction contents and user balance without obtaining approval from the user.
Sketch of Proof. The user's transactions are encrypted on the ledger. Even if the bank can access these encrypted texts, the bank cannot see the plaintext because the ElGamal encryption is CPA-secure.

7 CONLUSIONS

This study showed that cryptographic protocols, as in related works, are an effective tool for providing privacy and regulation to CBDCs at the same time. In addition, our study also addressed the privacy between the user and the banks and showed that cryptographic protocols protect this privacy. It also enabled banks to fulfill their tasks. Future work may focus on further refining these protocols and better protection of privacy and regulation of CBDCs.

ACKNOWLEDGEMENTS

We want to thank the researchers at the TÜBİTAK BİLGEM Blockchain Technologies Unit and TÜBİTAK BİLGEM for supporting.

References

1. S. Allen, S. Čapkun, I. Eyal, G. Fanti, B. A. Ford, J. Grimmelmann, A. Juels, K. Kostiainen, S. Meiklejohn, A. Miller, et al. Design choices for central bank digital currency: Policy and technical considerations. Technical report, National Bureau of Economic Research, 2020.
2. E. Androulaki, J. Camenisch, A. D. Caro, M. Dubovitskaya, K. Elkhiyaoui, and B. Tackmann. Privacy-preserving auditable token payments in a permissioned blockchain system. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 255–267, 2020.
3. R. Auer, R. Böhme, J. Clark, and D. Demirag. Mapping the privacy landscape for central bank digital currencies. *Communications of the ACM*, 66(3):46–53, 2023.
4. E. C. Bank. Exploring anonymity in central bank digital currencies. *In Focus*, 4:1–11, 2019.
5. E. C. Bank. Eurosystem report on the public consultation on a digital euro. 2021.
6. U. Bank of England. Central bank digital currency. opportunities, challenges and design. *URL: https://www. bankofengland. co. uk/-/media/boe/files/paper/2020/centralbank-digital-currency-opportunities-challenges-and-design. pdf*, 2020.
7. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
8. B. Bünz, S. Agrawal, M. Zamani, and D. Boneh. Zether: Towards privacy in a smart contract world. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers*, pages 423–443. Springer, 2020.
9. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE symposium on security and privacy (SP)*, pages 315–334. IEEE, 2018.
10. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Balancing accountability and privacy using e-cash. In *Security and Cryptography for Networks: 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006. Proceedings 5*, pages 141–155. Springer, 2006.
11. D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of Crypto 82*, pages 199–203. Springer, 1983.
12. D. Chaum, C. Grothoff, and T. Moser. How to issue a central bank digital currency. *arXiv preprint arXiv:2103.00254*, 2021.
13. Y. Chen, X. Ma, C. Tang, and M. H. Au. Pgc: pretty good decentralized confidential payment system with auditability. *Cryptology ePrint Archive*, 2019.
14. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.
15. C. Garman, M. Green, and I. Miers. Accountable privacy for decentralized anonymous payments. In *Financial Cryptography and Data Security: 20th International Conference, FC 2016, Christ Church, Barbados, February 22–26, 2016, Revised Selected Papers 20*, pages 81–98. Springer, 2017.
16. J. Gross, J. Sedlmeir, M. Babel, A. Bechtel, and B. Schellinger. Designing a central bank digital currency with support for cash-like privacy. *Available at SSRN 3891121*, 2021.
17. A. Kiayias, M. Kohlweiss, and A. Sarencheh. Peredi: Privacy-enhanced, regulated and distributed central bank digital currencies. *Cryptology ePrint Archive*, 2022.

18. K. Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In *Public Key Cryptography*, volume 2274, pages 48–63. Springer, 2002.
19. Y. Lee, B. Son, S. Park, J. Lee, and H. Jang. A survey on security and privacy in blockchain-based central bank digital currencies. *J. Internet Serv. Inf. Secur.*, 11(3):16–29, 2021.
20. T. P. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology—EUROCRYPT'91: Workshop on the Theory and Application of Cryptographic Techniques Brighton, UK, April 8–11, 1991 Proceedings 10*, pages 522–526. Springer, 1991.
21. S. Riksbank. The riksbank's e-krona pilot. *Sveriges Riksbank*, 2020.
22. E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pages 459–474. IEEE, 2014.
23. A. Tomescu, A. Bhat, B. Applebaum, I. Abraham, G. Gueta, B. Pinkas, and A. Yanai. Utt: Decentralized ecash with accountable privacy. *Cryptology ePrint Archive*, 2022.
24. N. Van Saberhagen. Cryptonote v 2.0 (2013). *URL: https://cryptonote. org/whitepaper. pdf. White Paper. Accessed*, pages 04–13, 2018.
25. K. Wüst, K. Kostiainen, V. Čapkun, and S. Čapkun. Prcash: fast, private and regulated transactions for digital currencies. In *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, pages 158–178. Springer, 2019.
26. K. Wüst, K. Kostiainen, N. Delius, and S. Capkun. Platypus: A central bank digital currency with unlinkable transactions and privacy-preserving regulation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2947–2960, 2022.
27. W. Zhao. Chinese state-owned bank offers test interface for pboc central bank digital currency. *CoinDesk. Accessed July*, 7:2022, 2020.

APPENDIX

Homomorphic Elgamal Encryption

The difficulty of solving the discrete logarithm problem is ensuring the security of the Elgamal encryption scheme. The encryption consists of the following three algorithms:

1. **KeyGen.** Assuming that p is a prime number and g is a generator of $\mathbb{Z}_p^*$. Then private key sk is randomly selected $sk \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and public key $pk = g^{sk}$ is calculated.
2. **Encryption.** To encrypt the v value, a random r is selected $r \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and c is calculated.

$$(c_1, c_2) = (g^r, g^v \cdot pk^r)$$

3. **Decryption.** To decrypt the ciphertext, c_2/c_1^{sk} is calculated.

$$g^v = c_2/c_1^{sk}$$

Then, the value b is found with brute force.

Shamir Secret Sharing

Shamir Secret Sharing is used to share a secret. To share the secret s with n parties, $n-1$ random numbers $(s_1, s_2, \ldots, s_{n-1})$ are selected and $P(x) = s + s_1x + s_2x^2 + \cdots + s_{n-1}x^{n-1}$ is created. $(x_i, P(x_i))$ is given to each party and $P(x_i)$ represents the sharing secret value. The secret s is recreated using interpolation:

$$s = \sum y_i \cdot \prod_{j \neq i} \frac{-x_j}{x_i - x_j}$$

$\prod_{j \neq i} \frac{-x_j}{x_i - x_j}$ is the Lagrange coefficient. We represent it with λ_i. For t parties to generate the secret instead of n parties, the polynomial's degree $t-1$ is selected.

Threshold ElGamal Encryption

Suppose the ElGamal private key sk is distributed to n parties. That is,

$$sk = \sum sk_i \lambda_i$$

To decrypt a ciphertext, i-party publishes $c_1^{sk_i}$, and the proof is generated in order to demonstrate the honest contribution of the party. g^v is calculated after summing the values from the parties.

$$g^b = c_2/\prod c_1^{sk_i \lambda_i}$$

b is found by applying brute force to g^b.

Range Proof

Bulletproofs [9] are utilized for the range proof in our solution. The definition of the range-proof relation is as follows:

$$\{(g, pk, c_2;\, v, r) : c_2 = g^v \cdot pk^r \land v \in [0,\, 2^{32} - 1]\}$$

g, pk and c_2 are open parameters, v and r are witness values. With range proof, a prover can prove that the value of v in a ciphertext is greater than 0 and less than $2^{32} - 1$.

Fiat-Shamir Technique

Fiat-Shamir is a technique used to make an interactive protocol non-interactive [7]. This technique uses an algorithm that generates a result using a hash function instead of a traditional protocol where two parties (prover and verifier) share information and interact with each other. The Fiat-Shamir technique eliminates interactivity in the proofs described in this section.

Proof of 0 Encryption

The definition of the Proof of 0 Encryption relation is as follows:

$$\{(g,\, pk,\, c_2,\, c_1;\, r) : c_1 = g^r \land c_2 = g^0 \cdot pk^r\}$$

With this proof, the prover proves that the value in the ciphertext is 0. The proof consists of the following 2 algorithms:

1. **Proof Generation.** Verifier generates and computes the following:
   (a) $u \overset{\$}{\leftarrow} \mathbb{Z}_p^*$
   (b) $a_1 \coloneqq g^u$
   (c) $a_2 \coloneqq pk^u$
   (d) $C \coloneqq \text{hash}(g,\, \mathbb{Z}_p^*,\, a_1,\, a_2,\, c_1,\, c_2)$
   (e) $w \coloneqq u + C.r \mod p$
   Then prover sends $a_1,\, a_2,\, C, w$ to the verifier.
2. **Verification.** The verifier checks for the following equations:
   (a) $C = \text{hash}(g,\, \mathbb{Z}_p^*,\, a_1,\, a_2,\, c_2)$
   (b) $g^w = a_1 \cdot c_1^C$
   (c) $pk^w = a_2 \cdot c_2^C$

Proof of Equality

In the ElGamal encryption, Kurosawa demonstrated that it is possible to use the same random values to encrypt data for multiple ciphertexts [18]. This idea is applied in our solution to enhance the efficiency of the proof of equality. The definition of the Proof of Equality relation is as follows:

$$\{(g,\, pk,\, c_1, c_2^1,\, c_2^2;\, v, r) : c_2^1 = g^v \cdot pk_1^r \wedge c_2^2 = g^v \cdot pk_2^r\}$$

Proof of Equality shows that two ciphertexts commit to the same plaintext. The proof consists of the following two algorithms:

1. **Proof Generation.** The verifier generates and computes the following:
   (a) $u \overset{\$}{\leftarrow} \mathbb{Z}_p^*$
   (b) $a_1 \coloneqq g^u$
   (c) $a_2 \coloneqq (pk_1/pk_2)^u$
   (d) $C \coloneqq \text{hash}(g,\, \mathbb{Z}_p^*,\, a_1,\, a_2,\, c_1,\, c_2^1,\, c_2^2)$
   (e) $w \coloneqq u + C.r \mod p$

   Then prover sends $a_1,\, a_2,\, C, w$ to the verifier.

2. **Verification.** The verifier checks for the following equations:
   (a) $C = \text{hash}(g,\, \mathbb{Z}_p^*,\, a_1,\, a_2,\, c_1,\, c_2^1,\, c_2^2)$
   (b) $g^w = a_1 \cdot c_1^C$
   (c) $(pk_1/pk_2)^w = a_2 \cdot (c_2^1/c_2^2)^C$