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 [34] and the Bank of England [7], have shown interest in developing their own digital currencies. The People's Bank of China [44] 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 [35] and Monero [41] 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 [14], 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. [13] 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 [42], 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 [3]. 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 [24]. This modified Zerocash protocol [35] 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 [43]. 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 [40]. Additionally, UTT limits the amount of money that can be anonymously sent monthly.

PEReDi [26] 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 [29], 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 [15], "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
[42]	UTxO	Yes	Yes	Yes	ZKP , ElGamal Enc., Ped. Com.
[3]	UTxO	Yes	Yes	Yes	VRF, ElGamal Enc., PS Sig., Ped. Com.
[24]	UTxO	Yes	Yes	Yes	Commitment, ZKP
[43]	Account	Yes	Yes	Yes	Commitment, ZKP
[40]	UTxO	Yes	Yes	Yes	MPC, Commitment, ZKP
[26]	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 [6] 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 [5]. Previous works have proposed this idea [42], [21], [43]. 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 [4], 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?
[42]	Balance Limit	Yes	No	No
[3]	Single Reg. Agency	Yes	No	No
[24]	Balance Limit	Yes	No	Yes
[43]	Balance Limit	Yes	No	Yes
[40]	Balance Limit	Yes	No	Yes
[26]	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 [4]. 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 the system.

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 sides 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.

Non-repudiation. Once a sender has made a payment, he should not be able to deny it later.

Transaction Privacy. When a transaction is given, 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.

2.4 Stakeholders & Roles

In this section, we describe the entities involved and their respective roles.

– **Central Bank:** The central bank issues the digital currency, 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 in privacy because of the possibility of mass surveillance. This means it is unable to reveal the transferred values linked to a certain transaction. For the role of the central bank, we refer to [15].

- **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. In addition, users have the ability to allow banks and financial institutions to review transactions.
- **Financial Institutions and Banks:** 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, financial institutions need to access the user's transaction details to fulfill their duties in the traditional system. 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 $sk_{e,u}$, the signature private key of the user $sk_{s,u}$, and the public key of regulatory agencies pk_a. $sk_{e,u}$ is used to access the plaintext of the encrypted balance of the user in the ledger and create zero-knowledge proofs, pk_a is used in threshold Elgamal encryption [19] [32] 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 $pk_{e,u}$ is added to the user's encrypted balance in the ledger with homomorphic encryption.

KAIME is built on an account-based system, similar to Zether [11]; 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

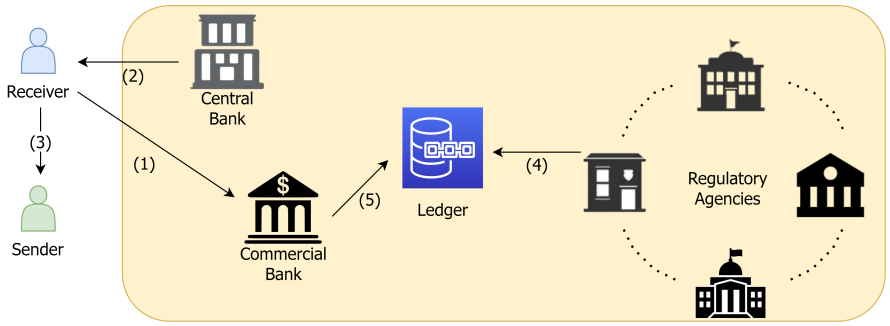


Fig. 1. The System consists of commercial banks (or any financial institutions)that are responsible for user registration and traditional bank tasks, the central bank that is responsible for currency issue and monetary policy, and regulatory agencies that are responsible for regulatory compliance. All 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.

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, he 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 deanonymize the user and examine past transactions. By exceeding the number of thresholds t similar to [26], the user's balance on the ledger can be accessed, and his past transactions can be accessed.

3 PROTOCOLS

In this section, the protocols used in KAIME are specified. The algorithms used in the protocols are described in Appendix.

3.1 System Initialization

In this protocol, the public keys of the central bank and banks are recorded on the blockchain and the public key of the regulatory agencies is created. The public keys $pk_c,\ pk_b$ of the central bank and banks are recorded on the blockchain. No specific algorithm for signing is specified in Chapter **??**. For signature, a signature algorithm with elliptic curve-based EUF-CMA security (such as ECDSA, EdDSA) can be used.

- **Central Bank:** The central bank runs KeyGen algorithm for signature and gets output as
$$(pk_c,\ sk_c) = (g^{sk_c},\ sk_c). \tag{1}$$
- **Commercial Banks:** A commercial bank runs KeyGen algorithm for signature and gets output as
$$(pk_b,\ sk_b) = (g^{sk_b},\ sk_b). \tag{2}$$
- **Regulatory Agencies:** Each regulatory agencies runs the distributed key generation algorithm with threshold parameters t for Homomorphic ElGamal Encryption and gets output as
$$(pk_a,\ sk_{a_i}) = (\prod_{i=1}^{n} g^{sk_{a_i}},\ sk_{a_i}) \quad \text{for} \quad 1 \leq i \leq n. \tag{3}$$

3.2 User Registration

This protocol is required to create an account on the system for a user. A 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.

User runs KeyGen algorithm for signature and gets output as

$$(pk_{s,u},\ sk_{s,u}) = (g^{sk_{s,u}},\ sk_{s,u}). \tag{4}$$

Then, the user runs KeyGen algorithm for Homomorphic ElGamal Encryption and gets output as

$$(pk_{e,u},\ sk_{e,u}) = (g^{sk_{e,u}},\ sk_{e,u}). \tag{5}$$

By verifying the KYC step, the user is registered to the system through the bank. The Bank encrypts 0 using the user's $pk_{e,u}$, runs Σ-E0 and signs this data and sends it to the blockchain.

Bank runs encryption algorithm with $pk_{E,U}$ gets output as

$$\phi = \texttt{Enc}(0,\ pk_{e,u}) = (g^r,\ g^0 \cdot pk_{e,u}^r). \tag{6}$$

Then, the bank creates proof π_{E0} with running Σ-E0.

$$\pi_{E0} = \Sigma\text{-E0}(\phi,\ r,\ pk_{e,u}) \tag{7}$$

Lastly, signs datas and sends to blockchain.

$$\sigma = \texttt{Sign}(\text{onboarding},\ pk_{e,u},\ pk_{s,u},\ \phi,\ \pi_{E0};\ sk_b) \tag{8}$$

Once the proof and signature are verified, the registration process is completed.

3.3 Currency Issue

Only the central bank can use user registration protocol and it is required to issue currency. 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.

$$\phi_u = \texttt{Enc}(v,\ pk_u) = (g^r,\ g^v \cdot pk_u^r) \tag{9}$$

$$\phi_r = \texttt{Enc}(v,\ pk_a) = (g^r,\ g^v \cdot pk_a^r) \tag{10}$$

Then, the central bank creates proof π-EE1 that the values in these two ciphertexts are the same.

$$\pi_{EE1} = \Sigma\text{-EE1}(pk_{e,u},\ pk_a,\ r) \tag{11}$$

Then, the central bank signs the datas.

$$\sigma = \texttt{Sign}(\text{issue},\ pk_{e,u},\ \phi_u,\ \phi_r,\ \pi_{EE1};\ sk_c) \tag{12}$$

Lastly, sends these to the ledger. The ledger is updated by checking the validity of the proofs and signature. The issued amount is added to the user's encrypted balance.

3.4 Payment

To start the payment process, the sender must first have the receiver's public key and the public key between the receiver and the bank. Assume User1 is sender and User2 is receiver. (In this section, to avoid notation complexity, the ElGamal Encryption public key of the users is shown as $pk_{e,u} = pk_u$.)

Then User1 accesses his old encrypted balance from the ledger and decrypts it.

$$\phi_{\text{o}} := (g^{r_{\text{o}}},\ g^b \cdot pk_1^{r_{\text{o}}}) \tag{13}$$

$$b := \texttt{Dec}(\phi_{\text{o}},\ sk_1) \tag{14}$$

User1 creates new ciphertext with same b to track the random value for creating range proof.

$$\phi_{\text{n}} := \texttt{Enc}(b, pk_1) = (g^{r_{\text{n}}},\ g^b \cdot pk_1^{r_{\text{n}}}) \tag{15}$$

User1 encrypts the value v it wants to transfer with its own public key pk_1, User2's public key pk_2 and the regulator agencies' public key pk_a.

$$\phi_1 := \texttt{Enc}(v, pk_1) = (g^r,\ g^v \cdot pk_1^r) \tag{16}$$

$$\phi_2 := \texttt{Enc}(v, pk_2) = (g^r,\ g^v \cdot pk_2^r) \tag{17}$$

$$\phi_a := \texttt{Enc}(v, pk_a) = (g^r,\ g^v \cdot pk_a^r) \tag{18}$$

User1 then runs two Σ-EE1. This proofs show that the plaintexts of 3 ciphertexts are equal to each other.

$$\pi^1_{EE1} = \Sigma\text{-EE1}(pk_1,\ pk_2,\ r) \tag{19}$$

$$\pi^2_{EE1} = \Sigma\text{-EE1}(pk_2,\ pk_a,\ r) \tag{20}$$

User1 also adds two range proofs to prevent the possibility of creating value out of thin air and to verify that there are sufficient funds in his account.

$$\pi^1_{\text{bullet}} = \texttt{Bulletproof}(\phi_1,\ r) \tag{21}$$

$$\pi^2_{\text{bullet}} = \texttt{Bulletproof}(\phi_n - \phi_1, r^{n-r}) \tag{22}$$

User1 runs Σ-EE2.

$$\pi_{\text{EE2}} = \Sigma\text{-EE2}(\phi_o,\ \phi_n,\ sk_1) \tag{23}$$

Finally, User1 signs the datas.

$$\begin{aligned}\sigma = \texttt{Sign}(\text{payment},\ pk_1,\ pk_2,\ \phi_n,\ \phi_1,\ \phi_2,\ \phi_a,\ \pi^1_{EE1},\\ \pi^2_{EE1},\ \pi^1_{\text{bullet}},\ \pi^2_{\text{bullet}},\ \pi_{EE2};\ sk_{s,1})\end{aligned} \tag{24}$$

Lastly, sends these to the ledger. The ledger is updated by checking the validity of the proofs and signature. The issued amount is subtracted to the User1's encrypted balance and the amount is added to the User2's encrypted balance.

3.5 Abort Transaction Privacy

Users use the public key of the regulators $pk_{E,R}$ when performing the transfer protocol and the central bank currency issue protocol.

Regulatory agencies apply the abort transaction protocol on the transaction or balance related to their shared ElGamal Encryption keys to see the content of transactions they consider suspicious. However, for this to happen, a sufficient number of regulatory agencies must reach a consensus.

3.6 Abort Transaction Privacy for Bank or Financial Institutions

When the user wants to receive service from the bank or institution, the institution that will provide the service needs the details 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 ElGamal Encryption public key for this function and sends it to the user.

$$(pk'_b,\ sk'_b) = (g^{sk'_b},\ sk'_b). \tag{25}$$

The user encrypts the balance and values of all previous k transactions with the public key.

$$\phi'_i = \texttt{Enc}(v_i,\ pk'_b) = (g^{r_i},\ g^{v_i} \cdot pk_b^{'r_i}) \quad \text{for} \quad 1 \leq i \leq k \tag{26}$$

After this step, the user creates Σ-EE2 for all past encrypted transactions ϕ_i 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.

$$\pi_{EE2_i} = \Sigma\text{-EE2}(\phi'_i,\ \phi_i,\ sk_{e,u}) \quad \text{for} \quad 1 \leq i \leq k \tag{27}$$

After the bank has verified the proofs, it can access the user's transaction values and balance.

4 ANONYMITY

In this section, we will give an anonymous version of the KAIME. This version not only hides the transferred amount but also hides the sender and receiver. However, it comes at the expense of additional costs. The complexity of the zero-knowledge proofs required for transfer will increase linearly , but with the method [18] proposes, the complexity will increase logarithmically. The process' complexity can be diminished using this method. A similar solution was used in Zether [11].

Since we do not recommend hiding the sender and recipient so that the regulatory mechanism can work better, we will only introduce it as an overview. An anonymous transaction allows a sender who desires to send a value v to a receiver with a public key pk_r, to hide 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, both the sender, the receiver, and the transaction details can be hidden.

5 IMPLEMENTATION

In the scope of this study, we have implemented cryptographic algorithms described in Appendix using the Go programming language. To evaluate the performance of these implementations, we provide the corresponding test results

Algorithm	Prover (ms)	Verifier (ms)	# of uses in a TX
Σ-EE1 (ed25519)	0.130	0.243	2
Σ-EE2 (ed25519)	0.201	0.156	1
Range Proof (ed25519)	32.209	18.072	2
Σ-E0 (ed25519)	0.121	0.252	-
ElGamal Encryption (ed25519)	0.147	-	3
Σ-EE1 (P256)	1.216	2.309	2
Σ-EE2 (P256)	1.989	1.503	1
Range Proof (P256)	292.965	121.516	2
Σ-E0 (P256)	0.672	1.749	-
ElGamal Encryption (P256)	1.083	-	3

Table 3. Performance Evaluation

in Table 3. Furthermore, the open-source tests and implementations of these algorithms can be accessed via GitHub[‡].

In Table 3. The "Algorithm" column specifies the cryptographic operation being measured, while the "Prover" and "Verifier" columns show the time it takes for the prover and verifier to complete the operation in milliseconds. Additionally, the "Number of uses in a TX" column indicates how many times each operation is utilized within a transaction. Since zero proof is not used in the transaction, it is shown with "-" in the column. The outcome produced by an i7-1165g7 @2.80GHZ computer with 16GB RAM.

6 SECURITY ANALYSIS

Before giving security analyses, we make some assumptions.

- The entities engage in communication through secure channels.
- All cryptographic operations are implemented as described in previous sections.
- Entities store their secret keys securely.
- Regulatory agencies do not want to see the transaction details arbitrarily.
- Regulatory agencies run the abort privacy transaction protocol only for people and transactions they think are suspicious.

6.1 Transaction Authentication

After zero knowledge proofs are created in the system, all data must be signed.

Claim. An adversary cannot create valid transaction if he has not have the related secret key.
Sketch of Proof. An adversary cannot generate a valid signature, since digital signature algorithm we use is EUF- CMA.

[‡]https://github.com/midmotor/kaime_cbdc_proof_test

6.2 Transaction Integrity

In the system, the balances of the sender and receiver increase and decrease homomorphically. It must be guaranteed that the balance increases and decreases by the same amount.

Claim. The user cannot send transactions value more than the balance he has. ”

Sketch of Proof. The claim's proof is covered by the soundness feature of the Bulletproof. Suppose the attacker wanted to send more than his balance and prepared the encrypted text in this way. In that case, the transaction will not be valid during the transaction verification since Bulletproof does not verify.

6.3 Balance Adequacy

Since the system operates on encrypted balances, balance adequacy cannot be checked with simple if conditions.

Claim. The user cannot send transactions value more than the balance he has.
Sketch of Proof. The claim's proof is covered by the soundness feature of the Bulletproof. Suppose the attacker wanted to send more than his balance and prepared the encrypted text in this way. In that case, the transaction will not be valid during the transaction verification since Bulletproof does not verify.

6.4 Negative Transaction Value

Since the transaction takes place encrypted, negative value control cannot be made.

Claim. The user cannot create a negative transaction value.
Sketch of Proof. The claim's proof is covered by the soundness feature of the Bulletproof. Suppose the adversary wanted to create negative value and prepared the encrypted text in this way. In that case, the transaction will not be valid during the transaction verification since Bulletproof does not verify.

6.5 Regulation Mechanism

Given that our solution incorporates regulatory mechanisms, it becomes imperative for us to examine the correctness 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 claim's proof is covered by the soundness feature of Σ-EE1. Suppose the adversary 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 since Σ-EE1 does not verify.

6.6 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.

6.7 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 circumstance 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.

6.8 Banks and Users Relationship

In the system, banks sometimes need the user's past transactions to perform their functions. In such a case, the user should not deceive the bank.

Claim. When the bank requests the user's past transactions, the user cannot change the detail of a transaction he has made.
Sketch of Proof. The claim's proof is covered by the soundness feature of Σ-EE1. Suppose the adversary does not encrypt the transaction values with one time public key of the bank. In that case, the transaction will not be valid during the transaction verification since Σ-EE1 does not verify.

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.

Sigma protocols are used instead of zkSNARKs for zero knowledge proofs in the system. Performance results show that processing times in discrete logarithm-based proofs took milliseconds with Σ-Protocols. As future work, these proofs

can be implemented with zkSNARKs and their performance times can be compared. Also, the system can be implemented on a permissioned blockchain that supports smart contracts. Transaction per Second (TPS) will give more detailed results about the performance of the system.

Additionally, the system does not include the offline payment function required for digital currencies to replace cash in the real world. Offline functionality can be added to the system by combining it with offline payment methods in the literature.

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. K. M. Alonso et al. Zero to monero. *Zero to monero*, 2020.
3. 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.
4. 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.
5. E. C. Bank. Exploring anonymity in central bank digital currencies. *In Focus*, 4:1–11, 2019.
6. E. C. Bank. Eurosystem report on the public consultation on a digital euro. 2021.
7. 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.
8. 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.
9. E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *Annual cryptology conference*, pages 90–108. Springer, 2013.
10. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 327–357. Springer, 2016.
11. 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.
12. 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.
13. 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.
14. D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of Crypto 82*, pages 199–203. Springer, 1983.
15. D. Chaum, C. Grothoff, and T. Moser. How to issue a central bank digital currency. *arXiv preprint arXiv:2103.00254*, 2021.
16. H. Chung, K. Han, C. Ju, M. Kim, and J. H. Seo. Bulletproofs+: Shorter proofs for a privacy-enhanced distributed ledger. *IEEE Access*, 10:42067–42082, 2022.
17. Y. G. Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–458, 1994.

18. B. E. Diamond. Many-out-of-many proofs and applications to anonymous zether. Cryptology ePrint Archive, Paper 2020/293, 2020. https://eprint.iacr.org/2020/293.
19. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.
20. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.
21. 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.
22. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32*, pages 626–645. Springer, 2013.
23. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Providing sound foundations for cryptography: On the work of shafi goldwasser and silvio micali*, pages 203–225. 2019.
24. 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.
25. L. C. Guillou and J.-J. Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *Advances in Cryptology—CRYPTO'88: Proceedings 8*, pages 216–231. Springer, 1990.
26. A. Kiayias, M. Kohlweiss, and A. Sarencheh. Peredi: Privacy-enhanced, regulated and distributed central bank digital currencies. *Cryptology ePrint Archive*, 2022.
27. C. Komlo and I. Goldberg. Frost: Flexible round-optimized schnorr threshold signatures. Cryptology ePrint Archive, Paper 2020/852, 2020. https://eprint.iacr.org/2020/852.
28. K. Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In *Public Key Cryptography: 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002 Paris, France, February 12–14, 2002 Proceedings 5*, pages 48–63. Springer, 2002.
29. 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.
30. G. Maxwell and A. Poelstra. Borromean ring signatures. *Accessed: Jun*, 8:2019, 2015.
31. T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Annual international cryptology conference*, pages 31–53. Springer, 1992.
32. 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.
33. 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.

34. S. Riksbank. The riksbank's e-krona pilot. *Sveriges Riksbank*, 2020.
35. 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.
36. C.-P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology—CRYPTO'89 Proceedings 9*, pages 239–252. Springer, 1990.
37. B. Schoenmakers. Lecture notes cryptographic protocols. *LectureNotes. pdf*, 2022.
38. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
39. D. Shanks. Class number, a theory of factorization, and genera. In *Proc. Symp. Math. Soc., 1971*, volume 20, pages 415–440, 1971.
40. 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.
41. N. Van Saberhagen. Cryptonote v 2.0 (2013). *URL: https://cryptonote. org/whitepaper. pdf. White Paper. Accessed*, pages 04–13, 2018.
42. 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.
43. 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.
44. W. Zhao. Chinese state-owned bank offers test interface for pboc central bank digital currency. *CoinDesk. Accessed July*, 7:2022, 2020.

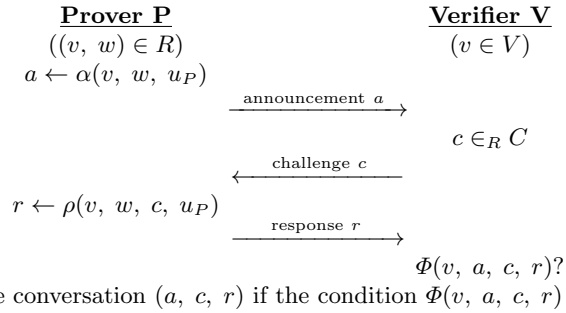
APPENDIX

Σ-Protocols

Σ-Protocols, are cryptographic techniques that allow a prover to convince a verifier that they possess knowledge of a value x satisfying a specific relation, all without showing any additional information about x [23]. Σ-Protocols find utility in the development of various cryptographic applications, credential protocols, e-voting protocols and group/ring signatures. In the proposed system, they were used to ensure privacy. One commonly known example of Σ-Protocols is Schnorr's protocol [36], designed for demonstrating knowledge of a discrete logarithm. The concept of Σ-Protocols generalizes not only Schnorr's protocol but also the identification protocols of Guillou-Quisquater [25] and Okamoto [31], along with numerous other protocols known in the literature.

More formally, consider a binary relation $R = (v; w) \subseteq V \times W$, where $v \in V$ represents the shared input for both the prover and the verifier, and $w \in W$ represents a witness, serving as the prover's private input. The language $L = \{v \in V : \exists w \in W \ (v;\ w) \in R\}$ is defined as the set of inputs v in V for which there exists a witness w in W such that $(v;\ w) \in R$ [37].

These proof systems possess three essential properties. First, completeness is that if the verifier and prover follow the protocol correctly, the verifier will accept the proof. Second, for any x and a pair of accepting conversations on input x represented as (a, c, z) and (a, c', z') with $e \neq e'$ there exists an efficient method to compute w such that $(x, w) \in R$. This is called the special soundness. Third, special honest-verifier zero knowledge (SHVZK) asserts that expressed through the existence of a specialized simulator known as the SHVZK simulator. Given a statement x and a challenge c, the simulator outputs (a, z) in a manner such that (a, c, z) forms an accepting 3-message transcript for x and is indistinguishable from a transcript produced by an honest prover when the challenge is c. Its definition is given below.



Accept the conversation $(a,\ c,\ r)$ if the condition $\Phi(v,\ a,\ c,\ r)$ is satisfied.

Fig. 2. Σ-Protocol for R.

Definition 1. *A Σ-Protocol for a relation R is a communication protocol involving a prover P and a verifier V, outlined in Figure 2, and adhering to the following three principles:*

- ***Completeness:** If both the prover P and verifier V adhere to the protocol, V always accepts.*
- ***Special Soundness:** There exists an efficient probabilistic polynomial time algorithm E (extractor) that, given any verifier output v and two accepting conversations $(a;\ c;\ r)$ and $(a;\ c;\ r)$ where $c \neq c$, can compute a witness w such that $(v;\ w) \in R$.*
- ***Special Honest-Verifier Zero-Knowledgeness:** There exists an efficient algorithm S (simulator) that, given any verifier output v in the language L and any challenge c, generates conversations $(a;\ c;\ r)$ with the same probability distribution as conversations between an honest prover P and verifier V for the given input v and challenge c.*

Why Σ-Protocols was Chosen (zkSNARKs vs Σ-Protocols)

Non Interactive Zero Knowledge (NIZK) proofs can be divided into two main categories: zk-SNARKs and Σ-Protocols with Fiat-Shamir. Both approaches have different efficiency characteristics, advantages and disadvantages. However, each has different characteristics in terms of factors such as computational cost, security level and overall complexity. This variety allows users to choose one that suits their specific application needs.

Zk-SNARKs can be theoretically applied to prove algebraic expressions, for example they can represent the circuit via Quadratic Arithmetic Programs (QAPs) [22]. However, the circuit for calculating a single exponential number in a group G usually consists of thousands of gates. QAP-based zk-SNARKs increase the computational cost of the prover linearly with the size of the circuit and require the creation of a reliable common reference sequence. For this reason, zk-SNARKs are generally considered inefficient when it comes to proving algebraic expressions. On the other hand, Sigma protocols can be used to represent discrete logarithm information with a fixed number of exponents and can therefore be more efficient. Since there are discrete logarithm statements in the system, Σ-Protocols were used in the proofs because they are much more advantageous.

Proof of Encryption Equality-1 (Σ-EE1)

Proof of Encryption Equality-1 (Σ-EE1) proves that the plaintexts of homomorphic ElGamal encryption encrypted with two different public keys pk_1 and pk_2 are equal to each other. $\phi_1 = (\phi_{1,L}, \phi_{1,R})$ represents the value encrypted with pk_1, $\phi_2 = (\phi_{2,L}, \phi_{2,R})$ represents the value encrypted with pk_2. In ElGamal encryption, Kurosawa demonstrated that employing identical random values for encrypting data across multiple ciphertexts is feasible [28]. This concept is integrated into our solution to optimize the efficiency of Σ-EE1. Therefore,

$\phi_{1,L} = \phi_{2,L}$. The sender uses 2 different Σ-EE1 for three different ciphertexts when preparing the transaction. It shows that the plaintexts of the encrypted ciphertexts with the public keys of the receiver, sender and regulatory agencies are equal to each other. In this way, the same amount that is deducted from the sender's balance increases in the receiver's balance, and regulatory agencies are provided with access to the transaction content in case of doubt.

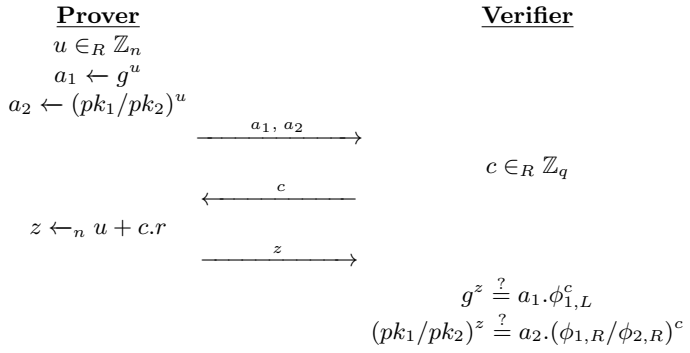


Fig. 3. Proof of Encryption Equality-1 (Σ-EE1)

Proposition 1. *The protocol depicted in Figure 3 is a Σ-Protocol designed for the relation:*

$$\{(g,\ pk_1,\ pk_2,\ \phi_{1,L},\ \phi_{1,R},\ \phi_{2,R};\ v,\ r) : \phi_{1,L} = g^r \wedge \phi_{1,R} = g^v \cdot pk_1^r \wedge \phi_{2,R} = g^v \cdot pk_2^r\}$$

Proof. Completeness evidently holds, as

$$g^z = g^{u+c.r} = g^u.g^{c.r} = g^u.(g^r)^c = a_1.\phi_{1,L}^c \tag{28}$$

and

$$(pk_1/pk_2)^z = (pk_1/pk_2)^{u+c.r} = (pk_1/pk_2)^u.((pk_1/pk_2)^r)^c = a_2.(\phi_{1,R}/\phi_{2,R})^c \tag{29}$$

For special soundness, we assume that there are two accepting conversations $(a_1,\ a_2,\ c,\ z)$ and $(a_1,\ a_2,\ c',\ z')$ with $c \neq c'$. Then we have:

$$\begin{aligned} & g^z = a_1.\phi_{1,L}^c, \quad g^{z'} = a_1.\phi_{1,L}^{c'} \\ \Rightarrow\ & g^{z-z'} = \phi_{1,L}^{c-c'} \\ \Leftrightarrow\ & \phi_{1,L} = g^{\frac{z-z'}{c-c'}} \end{aligned}$$

Therefore, the witness r holding $\phi_{1,L} = g^r$ is obtained as $r = (z - z')/(c - c')$. Similarly,

$$
\begin{aligned}
& (pk_1/pk_2)^z = a_2.(\phi_{1,R}/\phi_{2,R})^c, \quad (pk_1/pk_2)^{z'} = a_2.(\phi_{1,R}/\phi_{2,R})^{c'} \\
\Rightarrow\ & (pk_1/pk_2)^{z-z'} = (\phi_{1,R}/\phi_{2,R})^{c-c'} \\
\Leftrightarrow\ & (\phi_{1,R}/\phi_{2,R}) = (pk_1/pk_2)^{\frac{z-z'}{c-c'}}
\end{aligned}
$$

Lastly, in order to demonstrate the property of special honest-verifier zero-knowledgeness, it is essential to compare the distributions of conversations with an honest verifier (following the protocol and utilizing the witness x as input) with the simulated conversations that deviate from the protocol, using only the common inputs.

$$
\begin{aligned}
& \{(a;\ c;\ z) : u \in_R \mathbb{Z}_n;\ a_1 \leftarrow g^u;\ a_2 \leftarrow (pk_1/pk_2)^u;\ z \leftarrow_n u + cr\}, \\
& \left\{(a;\ c;\ z) : z \in_R \mathbb{Z}_n;\ a_1 \leftarrow g^z.\phi_{1,L}^{-c};\ a_2 \leftarrow (pk_1/pk_2)^z.(\phi_{1,R}/\phi_{2,R})^{-c}\right\},
\end{aligned}
$$

with given challenge c. The distributions are the same, and each conversation occurs with a probability of precisely $1/n^2$. Since the protocol is special honest-verifier zero-knowledgeness, it is also honest-verifier zero-knowledgeness □

Proof of 0 Encryption (Σ-E0)

Proof of 0 Encryption (Σ-E0) proves that the plaintext corresponding to the ciphertext is equal to 0. This is used by user's commercial bank to create the person's address in the ledger after the KYC step completed.

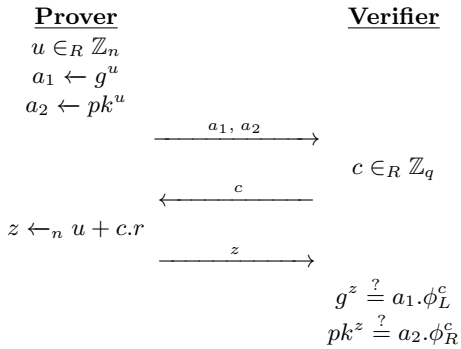


Fig. 4. Proof of 0 Encryption (Σ-E0)

Proposition 2. *The protocol depicted in Figure 4 is a Σ-Protocol designed for the relation:*

$$\{(g,\ pk,\ \phi_L,\ \phi_R;\ r) : \phi_L = g^r \wedge \phi_R = g^0 \cdot pk^r\}$$

Proof. Completeness evidently holds, as

$$g^z = g^{u+c.r} = g^u.g^{c.r} = g^u.(g^r)^c = a_1.\phi_L^c \tag{30}$$

and

$$pk^z = pk^{u+c.r} = pk^u.pk^{c.r} = pk^u.(pk^r)^c = a_2.\phi_R^c \tag{31}$$

For special soundness, we assume that there are two accepting conversations $(a_1,\ a_2,\ c,\ z)$ and $(a_1,\ a_2,\ c',\ z')$ with $c \neq c'$. Then we have:

$$\begin{aligned} & g^z = a_1.\phi_L^c, \quad g^{z'} = a_1.\phi_L^{c'} \\ \Rightarrow\ & g^{z-z'} = \phi_L^{c-c'} \\ \Leftrightarrow\ & \phi_L = g^{\frac{z-z'}{c-c'}} \end{aligned}$$

Therefore, the witness r holding $\phi_L = g^r$ is obtained as $r = (z - z')/(c - c')$. Similarly,

$$\begin{aligned} & pk^z = a_2.\phi_R^c, \quad pk^{z'} = a_2.\phi_R^{c'} \\ \Rightarrow\ & pk^{z-z'} = \phi_R^{c-c'} \\ \Leftrightarrow\ & \phi_R = pk^{\frac{z-z'}{c-c'}} \end{aligned}$$

Lastly, in order to demonstrate the property of special honest-verifier zero-knowledgeness, it is essential to compare the distributions of conversations with an honest verifier (following the protocol and utilizing the witness x as input) with the simulated conversations that deviate from the protocol, using only the common inputs.

$$\begin{aligned} & \{(a;\ c;\ z) : u \in_R \mathbb{Z}_n;\ a_1 \leftarrow g^u;\ a_2 \leftarrow pk^u;\ z \leftarrow_n u + cr\}, \\ & \left\{(a;\ c;\ z) : z \in_R \mathbb{Z}_n;\ a_1 \leftarrow g^z.\phi_L^{-c};\ a_2 \leftarrow pk^z.\phi_R^{-c}\right\}, \end{aligned}$$

with given challenge c. The distributions are the same, and each conversation occurs with a probability of precisely $1/n^2$. □

Proof of Encryption-2 (Σ-EE2)

When the sender wants to create a transaction, the sender must have the random value r of the ciphertext in the ledger. But unfortunately this is not possible as other transactions occur that change the user's state. For this reason, before sending a transaction, the user must retrieve the ciphertext from the ledger and create a new ciphertext by updating the random value. While doing this, it must show that it encrypts the same plaintext. For this reason, the system requires Σ-EE2. $\phi = (\phi_L,\ \phi_R)$ represents new ciphertext, $\phi' = (\phi'_L,\ \phi'_R)$ represents old ciphertext.

Proposition 3. *The protocol depicted in Figure 5 is a Σ-Protocol designed for the relation:*

$$\{(g,\ pk,\ \phi,\ \phi';\ r,\ v,\ x) : \phi_R = g^v \cdot pk^r \wedge \phi'_R = g^v \cdot pk^{r'}\}$$

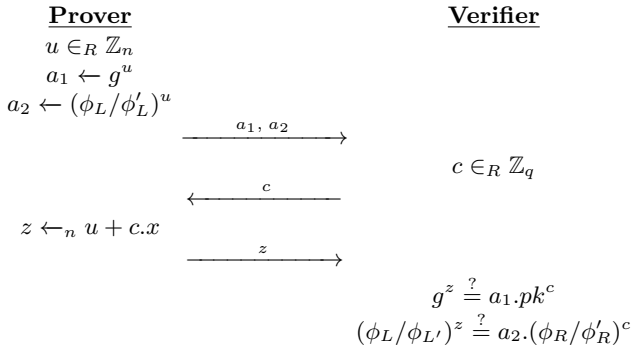


Fig. 5. Proof of Encryption-2 (Σ-EE2)

Proof. Completeness evidently holds, as

$$g^z = g^{u+c.x} = g^u.g^{c.x} = g^u.(g^x)^c = a_1.pk^c \tag{32}$$

and

$$(\phi_L/\phi'_L)^z = g^{(r-r').z} = g^{(r-r').u}.g^{(r-r').c.x} = a_2.pk^{(r-r')c} = a_2.(\phi_R/\phi'_R) \tag{33}$$

For special soundness, we assume that there are two accepting conversations $(a_1,\ a_2,\ c,\ z)$ and $(a_1,\ a_2,\ c',\ z')$ with $c \neq c'$. Then we have:

$$\begin{aligned} & g^z = a_1.pk^c, \quad g^{z'} = a_1.pk^{c'} \\ \Rightarrow\ & g^{z-z'} = pk^{c-c'} \\ \Leftrightarrow\ & pk = g^{\frac{z-z'}{c-c'}} \end{aligned}$$

Therefore, the witness x holding $pk = g^x$ is obtained as $x = (z - z')/(c - c')$. Similarly,

$$\begin{aligned} & (\phi_L/\phi_{L'})^z \stackrel{?}{=} a_2.(\phi_R/\phi'_R)^c, \quad (\phi_L/\phi_{L'})^{z'} \stackrel{?}{=} a_2.(\phi_R/\phi'_R)^{c'} \\ \Rightarrow\ & (\phi_L/\phi_{L'})^{z-z'} = (\phi_R/\phi'_R)^{c-c'} \\ \Leftrightarrow\ & \phi_R/\phi'_R = (\phi_L/\phi_{L'})^{\frac{z-z'}{c-c'}} \end{aligned}$$

Lastly, in order to demonstrate the property of special honest-verifier zero-knowledgeness, it is essential to compare the distributions of conversations with an honest verifier (following the protocol and utilizing the witness x as input) with the simulated conversations that deviate from the protocol, using only the common inputs.

$$\begin{aligned} & \{(a;\ c;\ z) : u \in_R \mathbb{Z}_n;\ a_1 \leftarrow g^u;\ a_2 \leftarrow (\phi_L/\phi'_L)^u;\ z \leftarrow_n u + cx\}, \\ & \left\{(a;\ c;\ z) : z \in_R \mathbb{Z}_n;\ a_1 \leftarrow g^z.pk^{-c};\ a_2 \leftarrow (\phi_L/\phi_{L'})^z.(\phi_R/\phi'_R)^{-c}\right\}, \end{aligned}$$

with given challenge c. The distributions are the same, and each conversation occurs with a probability of precisely $1/n^2$. □

Bulletproof

In cases where transaction privacy is required, users must prove that the amount of money entering the transaction is greater than the amount leaving. In simpler terms, the sender must prove that the balance he has is more than the amount he wants to send. In a classic payment system, this can be achieved with a simple condition check. However, range proofs are needed in a payment system where transaction amounts and balances are wanted to be kept confidential.

For instance, in Monero [2], ring signatures like Borromean Ring Signatures [30] were initially employed to prove that the processed value is positive and within a certain range. Although these methods provided the desired level of privacy, the drawback was the large size of the associated proofs.

The most significant advancement in this regard is the Bulletproof protocol designed by Bünz et al. [12], which introduces a much more efficient approach in terms of proof size and verification time. Bulletproofs are non-interactive and composable inner-product range proof protocols that enable provers to demonstrate that multiple processed values are within a given range with a succinct combined proof. Essentially, Bulletproofs build upon the inner-product arguments (IPA) introduced by Bootle et al. [10]. Bünz et al. [12] improved this design by halving the size of the underlying vector needed for commitment through a $\log_2 n$ recursive transformation from an n-dimensional vector to a 1-dimensional vector. This significantly reduced the complexity of the original IPA. Additionally, in the Bulletproof protocol, the absence of a need to establish a trusted system at the outset distinguishes it from zero-knowledge proof methods like zk-SNARK [9], emphasizing its importance in terms of decentralization.

The Bulletproof protocol, initially adopted in Monero, has recently been succeeded by the Bulletproof+ [16] protocol following a network update. Bulletproof+ employs a zero-knowledge weighted inner-product argument (zk-WIP) instead of IPA to generate range proofs in a shorter size compared to the short proof sizes achieved by the Bulletproof protocol. When comparing the time taken for the generation and verification of ZKPs in Bulletproof+ to those in the Bulletproof protocol, similar results are obtained. Similarly, the Bulletproof+ protocol allows for the proof of multiple values being within the specified range with a succinct combined proof.

In the system proposed in the thesis, Bulletproof is used in two different ways in a transaction. Its primary purpose is to show that the user has sufficient funds for the transaction. The other one shows that the encrypted value is between 0 and $2^{32} - 1$.

Formally, consider $v \in \mathbb{Z}_p$. Additionally, let V be an element in a group $\mathbb{G}$, representing a Pedersen commitment to the value v using the randomness parameter γ. The objective of the proof system is to convince the verifier that v lies within the range of integers from 0 to $2^n - 1$. In simpler terms, the proof

system aims to establish the following relation:

$$\{(g,\ h \in \mathbb{G},\ V,\ n;\ v,\ \gamma \in \mathbb{Z}_p) : V = h^{\gamma} g^{v} \wedge v \in [0, 2^n - 1]\} \tag{34}$$

Bulletproof are drawn in Figure 6, 7 and 8 . Giving the notation here would be good for the traceability of the protocol. Bold fonts represent a vector. Let $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{n} a_i \cdot b_i$ denotes the inner product. For a vector $\mathbf{g} = (g_1, \ldots, g_n) \in \mathbb{G}^n$ and $\mathbf{a} \in \mathbb{Z}_p^n$, $C = \mathbf{g}^{\mathbf{a}} = \prod_{i=1}^{n} g_i^{a_i} \in \mathbb{G}$.

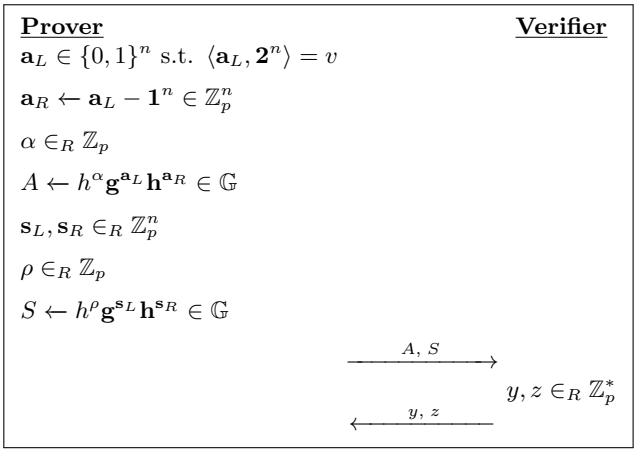


Fig. 6. Bulletproof Part1

After completing the steps in Figure 6, with this setup Prover creates two vector polynomials $l(X), r(X)$ in $\mathbb{Z}_p^n[X]$ and quadratic polynomial $t(X) \in \mathbb{Z}_p[X]$.

$$\begin{aligned} l(X) &= (\mathbf{a}_L - z \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot X && \in \mathbb{Z}_p^n[X] \\ r(X) &= \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + z^2 \cdot \mathbf{2}^n && \in \mathbb{Z}_p^n[X] \\ t(X) &= \langle l(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2 && \in \mathbb{Z}_p[X] \end{aligned}$$

Proposition 4. *The range proof depicted in Figure 6, 7 and 8 has perfect completeness, perfect honest verifier zero-knowledge and computational witness extended emulation.*

Proof. It is shown in Appendix C of [12].

Fiat-Shamir Technique

Fiat-Shamir is a technique used to make an interac- tive protocol non-interactive [8] This technique uses an algorithm that generates a result using a hash function

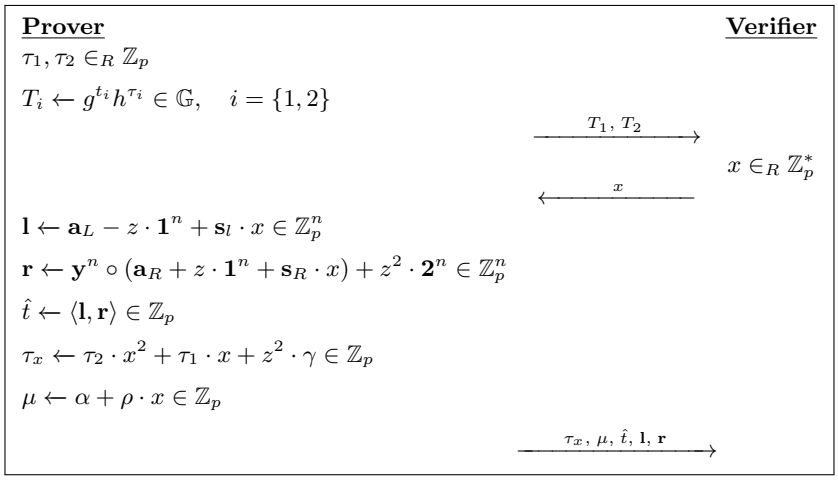


Fig. 7. Bulletproof Part2

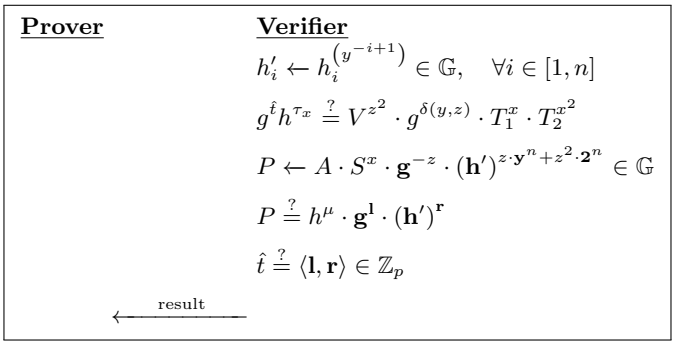


Fig. 8. Bulletproof Part3

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. How to use the Fiat- Shamir Technique in proofs is not shown for simplicity.

Distributed Key Generation

Threshold cryptography involves the implementation of techniques to distribute fundamental cryptographic schemes among multiple parties[17]. In this approach, the ownership or control of cryptographic keys is shared among a specified threshold of participants. Instead of relying on a single entity for key management, threshold cryptography enhances security by requiring collaboration

and consensus among a designated subset of parties to perform cryptographic operations or access sensitive information. This distribution of cryptographic functions reduces the risk associated with a single point of failure, offering a more robust and resilient security framework.

In various scenarios, relying on a single entity for access to valuable assets is often deemed undesirable. For instance, accessing a personal safe within a bank may necessitate the use of two keys: one retained by the safe's owner and another held by a bank employee. Similarly, in numerous cryptographic systems, restricting the possession of a secret key to a single entity is considered undesirable. Instead, the ownership or knowledge of a secret key should be distributed among multiple parties. This approach enhances security and mitigates risks associated with sole ownership, reflecting a principle of shared responsibility in safeguarding sensitive information.

In the proposed system to ensure compliance, a group of authorized institutions, named "regulatory agencies", are responsible for conducting audit procedures and other related tasks. They are able to access user data only by joint decision, through the use of threshold encryption. During the setup phase, regulatory agencies come together to create a private key for ElGamal Encryption. Before giving the definition of Pedersen Distributed Key Generation used [33], the definition of secret sharing are given [38].

Secret sharing schemes serve as the foundation for threshold cryptography. The concept involves dividing a secret into multiple shares in a way that allows the reconstruction of the original secret only when a predetermined number of shares are combined. If an inadequate number of shares is available, it becomes computationally infeasible to reconstruct the secret or any meaningful portion of it. This approach ensures that the security of the secret relies on a collaborative effort, with no single party possessing enough information to compromise the confidentiality of the original secret. The strength of the security lies in the threshold requirement for share reconstruction, providing a robust defense against unauthorized access.

Definition 2. *A secret sharing scheme involving a dealer D and participants $(P_1, ..., P_n)$ typically encompasses two essential protocols.*

- ***Distribution:*** *The dealer D divides a secret s, ensuring that every participant P_i receives a share s_i, $1 \leq i \leq n$.*
- ***Reconstruction:*** *s is recovered by combining shares $s_i i$, $i \in Q$, of any qualified set $Q \subseteq P_1, ..., P_n$.*

Secret sharing involves a dealer distributing a secret among participants, requiring collaboration for reconstruction, while distributed key generation (DKG) enables participants to jointly generate a cryptographic key without the need for a trusted party or dealer. The same distributed key generation used in FROST [27]. is used in the system. The steps of DKG are as follows.

1. Every participant P_i (regulatory agencies in our cases) chooses t random value and uses them as coefficients of polynomial $f_i(x) = \sum_{j=0}^{t-1} a_{ij} x^j$.

2. Each P_i calculates a proof of knowledge for the constant term in the polynomial.
3. Each P_i computes a commitment $\boldsymbol{C}_i = \langle \alpha_{i0}, \ldots, \alpha_{i(t-1)} \rangle$, where $\alpha_{ij} = g^{a_{ij}}$ and broadcasts $\boldsymbol{C}_i$ and the proof.
4. Each P_i, after receiving $\boldsymbol{C}_\ell$ and the proof, verifies the proof.
5. Each P_i securely sends to other participants a secret $(\ell, f_i(\ell))$.
6. Each P_i verifies their shares by calculating: $g^{f_\ell(i)} \stackrel{?}{=} \prod_{k=0}^{t-1} \alpha_{\ell k}^{i^k} \bmod q$, after that calculates private sharing key by computing $sk_i = \sum_{\ell=1}^{n} f_\ell(i)$
7. The group public key is computed

$$pk = \prod \alpha_{j0}$$

Homomorphic Threshold ElGamal Encryption

Threshold encryption is a type of encryption scheme that typically allows a group of users to collaboratively decrypt a message. In this system, a shared private key is often divided, and the pieces of this key can come together to decrypt the message when a specific threshold is reached.

Threshold encryption is useful for enhancing security and ensuring reliability. For instance, in situations requiring the participation of a group of individuals for a critical operation or data access, the shared private key pieces held by these individuals may need to come together. However, if a single individual cannot reach the threshold, they alone will be insufficient to decrypt the message. A variant of Elgamal Encryption [20], Homomorphic Threshold ElGamal Encryption, is used in the proposed system. During the setup phase, regulatory agencies create the public key with distributed key generation and share it in the system. Then the sender encrypts the value he wants to send with this public key. If a suspicious transaction is detected, regulatory agencies can decrypt by consensus and see the value.

Definition 3. *A $(t, n)-$threshold encryption is a scheme involving participants $(P_1, ..., P_n)$ encompasses three essential protocols.*

- ***Distributed Key Generation:*** *A protocol involving participants $(P_1, ..., P_n)$ to create a public key pk. In this process, each party P_i receives a private share x_i (related to the private key x corresponding to pk) and a public verification key h_i. The protocol's execution depends on parameter t.*
- ***Encryption:*** *A protocol that, given a plaintext message m and a public key pk, produces a ciphertext C for m under the public key pk.*
- ***Threshold Decryption:*** *A protocol involving a group of $t+1$ parties $(P_1, ..., P_{t+1})$ which, given a ciphertext C and individual inputs of private shares $(x_1, ..., x_{t+1})$, along with public keys $(pk_1, ..., pk_{t+1})$, jointly produces the plaintext message m.*

Homomorphic encryption is a cryptographic technique that enables certain computations to be performed on encrypted data, producing an encrypted result.

Upon decryption, the outcome matches the result obtained from performing the same operations on the original, unencrypted data. This can be shown in a more abstract way as follows: ($E(x)$ and $E(y)$ represent encryption of x and y respectively.)

$$E(x+y) = E(x)E(y) \tag{35}$$

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

1. **KeyGen.** Private key x is randomly selected $x \xleftarrow{\$} \mathbb{Z}_p^*$ and public key $pk = g^x$ is calculated. (In the system, the KeyGen phase has been replaced with Distributed Key Generation.)
2. **Encryption.** To encrypt the v value, a random r is selected $r \xleftarrow{\$} \mathbb{Z}_p^*$ and c is calculated.
$$(\phi_L,\, \phi_R) = (g^r,\, g^v \cdot pk^r) \tag{36}$$
3. **Decryption.** To decrypt the ciphertext, ϕ_R/ϕ_L^{sk} is calculated.
$$g^v = \phi_R/\phi_L^{sk} \tag{37}$$

Then, the value v is found with brute force. Here, a smarter solution can be used with the Baby-step giant-step algorithm[39].

In the system, balances are kept encrypted in the ledger. When any transaction is sent, this balance is reduced before the plaintext is revealed. On the reciver's side, the receiver's balance increases homomorphically. Let's say the sender's balance is b. This balance is $\phi = (g^r,\ g^b \cdot pk^r)$ in the ledger. When it wants to send the value v to another user, it prepares the ciphertext $\phi' = (g^{r'},\ g^v \cdot pk^{r'})$. Assuming the transaction is verified, the new ciphertext is as follows:

$$\phi'' = (g^{r-r'},\ g^{b-v} \cdot pk^{r-r'}) \tag{38}$$

Threshold Decryption

The following steps are followed to decrypt a text encrypted with a public key created with DKG.

$\prod_{j \neq i} \frac{-x_j}{x_i - x_j}$ is the Lagrange coefficient. We represent it with λ_i. Suppose the ElGamal private key sk is distributed to n parties. That is,

$$sk = \sum sk_i \lambda_i \tag{39}$$

To decrypt a ciphertext, i-party publishes $\phi_L^{sk_i}$, 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.

$$g^b = \phi_R / \prod \phi_L^{sk_i \lambda_i} \tag{40}$$

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