Connection Science

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/ccos20

A blockchain-based transaction system with payment statistics and supervision

Liutao Zhao, Jiawan Zhang & Lin Zhong

To cite this article: Liutao Zhao, Jiawan Zhang & Lin Zhong (2022) A blockchain-based transaction system with payment statistics and supervision, Connection Science, 34:1, 1751-1771, DOI: 10.1080/09540091.2022.2080181

To link to this article: https://doi.org/10.1080/09540091.2022.2080181

Published online: 14 Jun 2022.

CONNECTION SCIENCE
2022, VOL. 34, NO. 1, 1751–1771
https://doi.org/10.1080/09540091.2022.2080181

RESEARCH ARTICLE

OPEN ACCESS Check for updates

A blockchain-based transaction system with payment statistics and supervision

Liutao Zhao[a,b], Jiawan Zhang[a,c] and Lin Zhong[b]

[a]College of Intelligence and Computing, Tianjin University, Tianjin, People's Republic of China; [b]Beijing Computing Center Co., Ltd, Beijing, People's Republic of China; [c]Tianjin Cultural Heritage Conservation And Inheritance Engineering Technology Center and Key Research Center for Surface Monitoring and Analysis of Relics, State Administration of Cultural Heritage, Tianjin, People's Republic of China

ABSTRACT

Due to existing blockchain systems concentrate mainly on privacy protection but lack payment statistics and supervision, we propose a blockchain-based transaction system with payment statistics and supervision. In the system, a payer uses a homomorphic encryption scheme to protect payment amounts. After the transaction is recorded in the blockchain, not only the payee can decrypt the payment amounts and use for future payment, but also the payer can even decrypt it and use for payment statistics. Besides, two supervisors can independently decrypt all users' payment amounts to master the whole economic dynamism and detect illegal transactions. Comparing with existing schemes, our homomorphic scheme only increases a little length of ciphertext, but supports payment amounts decryption by the payer, an additional two receivers. Finally, analyses show that our system is extremely efficient.

ARTICLE HISTORY
Received 13 December 2021
Accepted 17 May 2022

KEYWORDS
Blockchain; homomorphic encryption; privacy protection; payment statistics; supervision

1. Introduction

Blockchain (Nakamoto, 2019) has attracted great attention due to its characteristic of transparency, immutability and distribution, etc. It has a wide range of application prospects. In financial markets, the transparency of blockchain enables any user to conduct relatively fair transactions on the blockchain, which can reduce economic losses caused by the information asymmetry between buyers and sellers. In control and information systems, the immutability of blockchain permanently records data accesses or modifications by any user. Therefore the responsibility of each user is straightforward. In the military, the distributed blockchain can hide the command centre, and any damage to the partial system will not affect the stable operation of the whole systemThe Russian State Tretyakov Art Museum has established "My Triakov" as a fundraising initiative, utilising blockchain technology to allow "the entire globe" to get to be a supporter of the museum's digital building and a collection of the gallery's works. In September 2021, the State Hermitage Museum, one of the world's four largest institutions, will organise an NFTs sale of five world-famous artworks,

CONTACT Jiawan Zhang jwzhang@tju.edu.cn College of Intelligence and Computing, Tianjin University, Tianjin 300350, People's Republic of China

alongside the Louvre in Paris, the British Museums in London and the Museum of art in New York. This greatly improves the extent to which museums, especially digital collections, can make money from work of art (Wang, Li, et al., 2021). Blockchain is typically used to allow actual members of a certain group to share and trade sensitive information. Permissioned blockchains are referred to as such because exterior users can view or engage in private blockchains unless they have been given the authority. When users adopt a commercial blockchain network, they help to maintain the network's decentralised nature by storing a shared ledger and collaborating to reach a consensus on modifications. Besides, blockchain technology has been widely applied to the Internet of Things, the medical, intellectual property, logistics, etc.

The Bitcoin (Nakamoto, 2019) and Ethereum (Wood, 2014) blockchain systems are the most successful application. In financial systems, privacy protection is very crucial. Plaintext transaction has many disadvantages. The difference among Ethereum and Bitcoin would be that Bitcoin is merely a currency, but Ethereum is a digital ledger that is being used by businesses to create new initiatives. Both Bitcoin and Ethereum are based on "blockchain" technologies, but Ethereum's is significantly more reliable. For example, the disclosure of a user's wealth will endanger his life and property safety; disclosure of the company's economic status will lead to malicious competition; leakage of the country's economic strength, will lead to a financial crisis. For museums, digital cultural relics exhibition lacks a complete authorisation verification mechanism, and these digital materials will be arbitrarily spread or even forged (Wang, Chen, et al., 2021; Zhaofeng et al., 2019). Therefore, it is necessary to protect transaction privacy. Monero (Noether, 2015; Noether & Mackenzie, 2016), Zerocoin (Miers et al., 2013) and Zerocash (Ben-Sasson et al., 2014) are typical blockchain systems with good privacy protection. The Monero system was originally built on CryptoNote, which hides the target and source of payment interactions via ring signatures and single-time keys. The strategy is dependent on confidential exchanges which are utilised on Bitcoin's Elements side-chain, but it also enables their own use in ring signatures. Zerocoin is a Bitcoin-based cryptography option that allows for entirely anonymous monetary transactions. This approach is based on normal cryptography principles and therefore does not make any new providing valuable and otherwise alter Bitcoin's security architecture. Zerocash is a derivative of Bitcoin that can be used at an equal scale. As a basis of its enhanced performance and efficiency, Zerocash allows for the complete replacement of standard Bitcoin transactions with untraceable equivalents.

However, privacy protection in financial transactions is not enough for economic development, as privacy protection creates a living space for corruption, illegal financing and illegal transfer abroad. Therefore, it is necessary to supervise each transaction under the premise of privacy protection. All transactions are recorded in the immutable blockchain, allowing regulators to check each transaction for discovering illegal transactions and other financial activities. In addition, the blockchain system can directly reject illegal transfers abroad, which is conducive to improving the stability of finances.

By embedding trapdoors to the Monero, Zerocoin and Zerocash systems, the purpose of supervisory can be fulfilled. For example, traceable ring signatures (Feng et al., 2020; Fujisaki & Suzuki, 2007) enabled the supervisor to trace the true signer. All payers need to count the payment amounts, which is a huge application demand in financial activities. For example, individuals, all companies, need to make periodical financial statistics of payment amounts and use them to plan their future payment activities; The state also needs to count every

financial activity and plan its development direction. However, existing privacy-protected transaction systems, such as Monero, Zerocoin and Zerocash, only enabled the payee to decrypt his denomination, not the payer. As a result, neither individual users nor companies can decrypt payment amounts from the blockchain and conduct financial statistics, but only a plaintext backup of the transaction locally. However, this kind of plaintext backup has a high risk of data theft and tampering.

Therefore, according to existing blockchain systems only concentrate on privacy protection but lack payment statistics and supervision, we propose a transaction system with payment statistics and supervision based on blockchain.

Contributions: We first contribute a transaction system model. Then, we present efficient concrete construction. In the concrete construction, a special homomorphic encryption scheme is the main innovation of this paper. It enabled a payer, a payee and two supervisors to decrypt payment amounts independently. Finally, we prove strictly the security of the homomorphic encryption scheme and compare it with related schemes. The homomorphic encryption architecture reduces the computational cost, allowing it to function with smart appliances with limited processing capacity. It guarantees that information will never ever be relocated or exposed in cleartext. Analyses show that our transaction system and homomorphic encryption scheme are efficient.

2. Related work

Bitcoin (Nakamoto, 2019) is a peer-to-peer electronic cash that enables internet operations would be sent immediately between parties without the need of a bureaucratic commercial bank. By using the proof-of-work consensus mechanism and network timestamps, it prevents double-spending without using a trusted third party. Etherum (Wood, 2014) has demonstrated its strong practicality through a larger quantity of work. Each work can be seen as a simple application on a decentralised, but singleton, compute resource. However, Bitcoin and Ethereum use plaintext to transact, which reveal the private information of each user and lead hackers to conduct statistical analysis attacks on it.

Maxwell (2013) introduced Coinjoin, a numerous input and multiple-output transaction mechanism, to handle the issue of anonymity. However, it requires interaction between the various participants, which increases the probability of information leakage. Besides, each user does not trust each other, and the interaction is extremely difficult. Coinjoin is a form of bitcoin exchange that improves anonymity by eliminating the presumption of common-input ownership. CoinShuffle is a distributed Bitcoin blending system described by Ruffing et al. (2014), which enables payers and payees to conduct Bitcoin transactions in a fully anonymied fashion. CoinShuffle is based on the Dissent accountability anonymised group communication system and has numerous benefits over the Bitcoin blending techniques that came before it. It does not require any trusted third party and it is perfectly compatible with the current Bitcoin blockchain system. However, it is vulnerable to a variety of denial of service attacks as it needs all participants online all the time.

Bissias et al. (2014) developed Xim, a two-party blending system that works with Bitcoin and other virtual payments. This can withstand a variety of assaults, including Sybil, DOS, man-in-the-middle and length of time inferences assaults. It's the first decentralised network to combat several attacks and timing-based inferences assault all at the same time.

It offers a decentralised method for discreetly locating mixing mates through blockchain marketing. Furthermore, it's trying to obfuscate method of blending decreases the consequences of Sybil-based denial-of-service assaults, and it raises attackers expenses directly proportional to the number of participation. CoinParty is a distributed mixing system for Bitcoin suggested by Ziegeldorf et al. (2015), which is built on a mixture of decryption mixnets and threshold cryptosystems. Monero (Noether, 2015; Noether & Mackenzie, 2016) uses ring signature scheme to hide the payer, and uses one-time keys to hide the payee in transactions. Besides, it uses the technique of a commitment scheme or homomorphic encryption scheme to rawhide the amount of a transaction. Zerocoin (Miers et al., 2013)/Zerocash (Ben-Sasson et al., 2014) are full-featured ledger-based cryptocurrencies that provide high privacy protections. It hides the sender, beneficiary and value in an online transaction using zero-knowledge Succinct Non-interactive ARguments of Knowledge (zk-SNARKs) (Ben-Sasson et al., 2014; Groth, 2016).

However, all the above schemes only consider privacy protection. The supervision of the blockchain system includes transaction traceability mechanisms (Koshy et al., 2014 Reid & Harrigan, 2013;), address clustering mechanisms (Meiklejohn et al., 2013; Zhao, 2014), certificate management mechanisms (Androulaki et al., 2018; Moser & Narayanan, 2019; Wust et al., 2018) and trapdoor technologies. We think that trapdoor is a good technology to achieve fine-grained supervision as it can trace each transaction in the system. Therefore, we will embed some trapdoors in our homomorphic encryption scheme to achieve fine-grained supervision. Besides, our homomorphic encryption scheme supports the sender/payer to decrypt his payment amounts, which is a main and special innovation of this paper.

3. Transaction system

As shown in Figure 1, there are four kinds of participants, i.e. a payer, a payee, consensus nodes (or miners) and two independent supervisors, in the blockchain-based transaction system. In generic construction, we will employ a homomorphic encryption scheme, a non-interactive zero-knowledge proof protocol and a digital signature. The main innovation of our transaction system is a homomorphic encryption scheme, which enables the payer, the payee and two supervisors to decrypt the ciphertext payment amounts independently. The transaction system consists of eight procedures, **Init**, **KeyGen**, **Pay**, **Ver**, **PayeeDec**, **PayerDec**, **Supervisor1Dec**, **Supervisor2Dec**, for initialising the system, generating keys, paying, verifying, decrypting by the payee, payer and two supervisors, respectively. Private keys are often used to validate transactions and show that a blockchain address belongs to the owner. You can handle cryptocurrency transactions using a public key. It's a private key that's linked with a cryptography algorithm. While anybody can submit transaction to the public key, you'll need to have the secret key to unlock them and show because you own the bitcoin that was acquired. Ciphertext is information that has been encoded using a data encryption. Moreover, a zero-knowledge proof, also known as a zero-knowledge protocol, is a methodology in cryptography through which one participant can establish to some other entity that a particular statement is accurate without providing any further information other than the premise that the argument is correct.

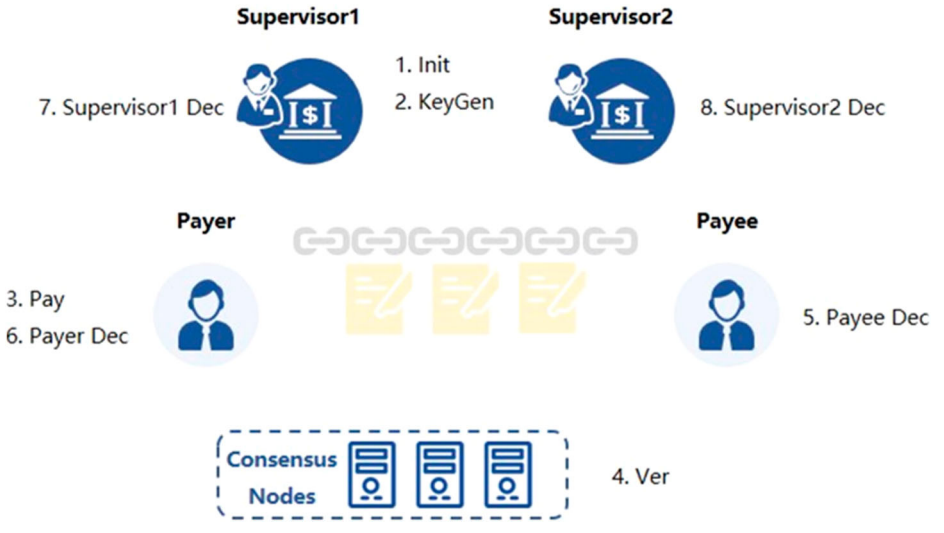

Figure 1. The transaction system architecture.

Formally, for a fixed security parameter, these procedures work as follows:

Init: The *Init* procedure is run by one of the supervisors. It takes as input a security parameter 1^{λ}. It returns the system parameters *SP*.

$$SP \leftarrow Init(1^{\lambda}).$$

KeyGen: The *KeyGen* procedure is run by the payer, the payee and two supervisors. It takes as input the system parameters *SP*. It returns a private key and a public key

$$(sk, PK) \leftarrow KeyGen(SP).$$

Thus the key pairs of the payer, the payee and two supervisors are (sk_1, PK_1), (sk_2, PK_2), (sk_3, PK_3), (sk_4, PK_4), respectively.

Pay: The payment procedure includes three steps, i.e. the homomorphic encryption scheme *HEnc*, the non-interactive zero-knowledge proof *NIZK*, and the digital signature *Sign*. This procedure is run by the payer.

- **HEnc:** It takes as input the system parameters *SP*, payment amounts v_1, his private key sk_1, the public keys PK_2, PK_3, PK_4 of the payee and two supervisors. It outputs a homomorphic ciphertext C_1

$$C_1 \leftarrow HEnc(SP, sk_1, PK_2, PK_3, PK_4, v_1).$$

Note that in the above homomorphic encryption, we embedded the private key sk_1 of the payer and the public keys PK_2, PK_3, PK_4 of the payee and two supervisors, so that all of them can decrypt the homomorphic ciphertext C_1 independently to get the same payment amounts v_1. This procedure does not need a zero-knowledge proof protocol to prove that the payment amounts v_1 decrypted by multiple different participants are the same.

In the same way, it takes as input the system parameters SP, payment amounts v_2, his private key sk_1, the public keys PK_1, PK_3, PK_4 of himself and two supervisors. It outputs a homomorphic ciphertext C_2

$$C_2 \leftarrow HEnc(SP, sk_1, PK_1, PK_3, PK_4, v_2).$$

The payment amount v_2 is the change for the payer. Note that as the change v_2 is the payer pay to himself, so both his private key sk_1 and public key PK_1 are embedded in the ciphertext.

Obviously, one can encrypt multiple payment amounts $v_3, \ldots v_n$ for other payees as follows

$$C_i \leftarrow HEnc(SP, sk_1, PK_i, PK_3, PK_4, v_i),$$
$$i = 3, 4 \ldots, n.$$

$PK_i, i = 3, 4 \ldots, n$, are public keys of other payees.

Similarly, the unspent amounts of the payer read from the blockchain has the following two kinds of expressions

$$C_0 = HEnc(SP, sk_0, PK_1, PK_3, PK_4, v_0),$$
$$C_0' = HEnc(SP, sk_1, PK_1, PK_3, PK_4, v_0').$$

The amount v_0 spent by user 0 whose private key is sk_0, and v_0' is a change that the payer spends to himself. As PK_1 is in both ciphertexts C_0, C_0', the payer can spend v_0, v_0' validly by using his corresponding private key sk_1.

- **NIZK:** The non-interactive zero-knowledge proof *NIZK* includes two protocols, i.e. the Sigma protocol (Damgård, 2000) and the Bulletproofs (Bünz et al., 2018). The use of the customised dedicated hash algorithm and the sigma protocol to conceal the transactions by observing it from the public in the system tends to provide the rigid distance of deterministic public random value to determine the classified information transaction of this Non-Interactive Zero Knowledge (NIZK) confirmation. In the bulletproofs protocol, it was utilised to create a linear logarithmic trustless configuration with a small proofing dimension. Using the ZKP argument, it creates a proof that is monotonically continuous and has a rapid confirmation time period to speed up the process for different proof systems.

The Sigma protocol is used to prove that the sum of the unspent ciphertext amounts $v_0 + v_0'$ is equal to the sum of the ciphertext amounts $v_1 + v_2$ that need to be paid. The Sigma protocol takes as input C_0, C_0', C_1, C_2, and returns proof data,

$$ZK_{Sigma}\{v_0 + v_0' = v_1 + v_2 | C_0, C_0', C_1, C_2\}.$$

This equation is a complete expression, which includes the following two subcases. If $v_0 = 0$ or $v_0{}' = 0$, then the above equation can be reduced as follows

$$ZK_{Sigma}\{v_0{}' = v_1 + v_2 | C_0{}', C_1, C_2\},$$
$$ZK_{Sigma}\{v_0 = v_1 + v_2 | C_0, C_1, C_2\}.$$

It means the payer only intends to spend the unspent amounts v_0 or $v_0{}'$.

The Bulletproofs protocol is used to prove that all ciphertext amounts v_1, v_2 that need to be paid are positive

$$ZK_{Bulletproofs}\{v_1 \geq 0, v_2 \geq 0 | C_1, C_2\}.$$

- **Sign:** It takes as input all the above data *data* and his private key sk_1, and returns a signature σ

$$\sigma \leftarrow Sign(sk_1, data).$$

Let $data = (C_0, C_0{}', C_1, C_2, ZK_{Sigma}, ZK_{Bulletproofs})$.

Ver: The verification procedure includes signature verification Ver_{sign}, Sigma verification Ver_{Sigma}, and Bulletproofs verification $Ver_{Bulletproofs}$. This procedure is run by consensus nodes (or miners) of the blockchain system. Ver_{sign} takes as input the system parameters *SP*, all the above data *data*, the signature σ, the corresponding public key PK_1, and returns a judgment

$$True/False \leftarrow Ver_{sign}(SP, PK_1, data, \sigma).$$

Ver_{Sigma} and $Ver_{Bulletproofs}$ take as input the proof data and returns a judgment respectively

$$True/False \leftarrow Ver_{Sigma}(ZK_{Sigma}),$$
$$True/False \leftarrow Ver_{Bulletproofs}(ZK_{Bulletproofs}).$$

If all output is True and no double-spending, then accept and record in the blockchain using a consensus mechanism, such as Byzantine fault-tolerant (Miller et al., 2016), etc. Deficiency of the Byzantine Tolerance refers to a decentralised channel's potential to access consensus on the same quantity although some participating nodes refuse to reply or answer with inaccurate information.

PayeeDec: The *PayeeDec* procedure is run by the payee. It takes as input the system parameters *SP*, his private key sk_2, the public keys of the payee and two supervisors PK_1, PK_3, PK_4, the homomorphic ciphertext C_1. It outputs an amount v_1

$$v_1 \leftarrow PayeeDec(SP, PK_1, PK_3, PK_4, sk_2, C_1).$$

Similarly, the payer can decrypt his change v_2

$$v_2 \leftarrow PayerDec(SP, PK_1, PK_3, PK_4, sk_1, C_2).$$

PayerDec: The *PayerDec* procedure is run by the payer. It takes as input the system parameters *SP*, his private key sk_1, the public keys of the payer and two supervisors PK_2, PK_3, PK_4,

the homomorphic ciphertext C_1. It outputs an amount v_1

$$v_1 \leftarrow PayerDec(SP, PK_2, PK_3, PK_4, sk_1, C_1).$$

Similarly, the payer can decrypt the other amounts v_2

$$v_2 \leftarrow PayerDec(SP, PK_1, PK_3, PK_4, sk_1, C_2).$$

We believe that the payer can decrypt his payment ciphertext is one of the main contributions of this paper. Payment statistics are one of the most important functions in financial activities, as each individual, our society and our country need to record payment amounts, summarise payment activities and use them for planning future payment activities.

Supervisor1Dec: The *Superviser*1*Dec* procedure is run by the first supervisor. It takes as input the system parameters SP, his private key sk_3, the public key of the other supervisor PK_4, the homomorphic ciphertext C_1. It outputs an amount v_1

$$v_1 \leftarrow Superviser1Dec(SP, PK_4, sk_3, C_1).$$

Similarly, he can decrypt the other amounts v_2

$$v_2 \leftarrow Superviser1Dec(SP, PK_4, sk_3, C_2).$$

Supervisor2Dec: The *Superviser*2*Dec* procedure is run by the second supervisor. It takes as input the system parameters SP, his private key sk_4, the public key of the other supervisor PK_3, the homomorphic ciphertext C_1. It outputs an amount v_1

$$v_1 \leftarrow Superviser2Dec(SP, PK_3, sk_4, C_1).$$

Similarly, he can decrypt the other amounts v_2

$$v_2 \leftarrow Superviser2Dec(SP, PK_3, sk_4, C_2).$$

Therefore, these two supervisors can independently decrypt all users' payments to master the whole economic dynamism and detect illegal transactions.

4. Concrete construction

4.1. Preliminaries

In the concrete construction of the transaction system, we will use the Bulletproofs, and the digital signature scheme in the black-box model, and we omit their introduction. The employment of a black box approach in the cryptanalysis of homomorphic encryption algorithms might be beneficial. Bijective morphisms can indeed be expected to be isotropic instantly, which is a major feature of black box algebra. In computational theory, it's an ideal context for randomised algorithms are used to solve permutations and matrices group problems. We briefly review some related important cryptography concepts including bilinear groups (Boneh & Boyen, 2008) and difficult problems. In recent times, bilinear groups of composite order have been employed to tackle a variety of cryptographic challenges. While the minority choice hypothesis is a valuable tool for creating secure protocols, it poses considerable challenges when it comes to putting them into reality.

4.1.1. Bilinear groups

The payer, the payee and two supervisors need to decrypt the payment amounts in the system, which can be implemented using a bilinear map. The discrete logarithm issue on that category of elliptic curve cryptography over a discrete space can be transported to the discrete logarithm on a narrower sample space, where a sub-exponential index math approach can be utilised. We briefly review the bilinear maps and groups, in the notation of (Boneh & Boyen, 2008):

- Let $\lambda \in \mathbb{Z}^+$ be a security parameter of our encryption system, and $n, 2^{\lambda-1} < n < 2^{\lambda}$ is a large prime. $(\mathbb{G}_1, *)$, $(\mathbb{G}_2, *)$ and $(\mathbb{G}_T, *)$ are three cyclic groups of prime order n.
- G is a generator of $\mathbb{G}_1$ and H is a generator of $\mathbb{G}_2$;
- $\hat{e}$ is a bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, i.e. a map satisfying the following properties:
  − Bilinearity: $\forall G \in \mathbb{G}_1, \forall H \in \mathbb{G}_2, \forall x, y \in \mathbb{Z}_n$,

$$\hat{e}(x \cdot G, y \cdot H) = \hat{e}(G, H)^{xy};$$

  − Non-degeneracy: $\hat{e}(G, H) \neq 1$ and is thus a generator of $\mathbb{G}_T$.

All operations on groups and bilinear maps can be achieved in polynomial-time. Formally, one defines a bilinear group generation algorithm $\mathcal{G}$ that takes as input a security parameter $\lambda \in \mathbb{Z}^+$ and outputs the description of groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We then require the existence of probabilistic polynomial-time algorithms (in λ) for computing the group operation in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and the bilinear map $\hat{e}$. If $\mathbb{G}_1 = \mathbb{G}_2$, we call the bilinear map is a symmetry bilinear map. If $\mathbb{G}_1 \neq \mathbb{G}_2$, we call it asymmetric bilinear maps. In this paper, our scheme rely on the symmetrical bilinear group.

4.1.2. Difficult problems

The complexity of the Bilinear Diffie-Hellman Problem, that is an augmentation of the three challenges outlined below for a multiplicative group $\mathbb{G}_1$, provides the basis for our homomorphic encryption approach. To establish an effective cryptographic algorithm, a variety of Diffie-Hellman-type complexity considerations in bilinear groups were applied. The security of newer Weil-pairing-based crypto algorithms must be established.

Computational Diffie-Hellman (CDH) Problem: Given three group elements $G, a \cdot G, b \cdot G \in \mathbb{G}_1$, where $a, b \in \mathbb{Z}_n$, find an element $H \in \mathbb{G}_1$ such that the following equation holds

$$H = ab \cdot G$$

Bilinear Diffie-Hellman (BDH) Problem: Given four group elements $G, a \cdot G, b \cdot G, c \cdot G \in \mathbb{G}_1$, where $a, b, c \in \mathbb{Z}_n$, find an element $G' \in \mathbb{G}_T$ such that the following equation holds

$$G' = \hat{e}(G, G)^{abc}$$

Decision Bilinear Diffie-Hellman (DBDH) Problem: Given four group elements $G, a \cdot G, b \cdot G, c \cdot G \in \mathbb{G}_1$, where $a, b, c \in \mathbb{Z}_n$, and an element $G' \in \mathbb{G}_T$ decide whether or not the following equation holds

$$G' = \hat{e}(G, G)^{abc}$$

If it holds, then the quintuple $(G, a \cdot G, b \cdot G, c \cdot G, G')$ is a valid DBDH tuple.

Gap Bilinear Diffie-Hellman (GBDH)Problem: Solve a given instance, $(G, a \cdot G, b \cdot G, c \cdot G)$, of the BDH problem with the help of a DBDH oracle that is able to decide whether or not a tuple $(G, a \cdot G, b \cdot G, c \cdot G, G')$ is a valid DBDH tuple. A DBDH group is one in which group elements are difficult to analyse but simple to confirm. It is clearly possible to create a group in which Decision Diffie-Hellman is manifestly simple, but really what evidence do we have indicating Computational Diffie-Hellman is difficult in these kinds of groups.

4.2. Concrete scheme

Formally, a concrete construction of the transaction system is as follows:

Init: Let $\hat{e}$ be symmetric bilinear maps, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. G is a generator of $\mathbb{G}_1$, and $\hat{e}(G, G)$ is a generator of $\mathbb{G}_T$. The prime order of $\mathbb{G}_1, \mathbb{G}_T$ is n. A hash function *hash* maps a data of any length to $\{0, \ldots, n-1\}$, $hash : \{0,1\}^* \to \mathbb{Z}_n$. The system parameters are

$$SP = (\hat{e}, \mathbb{G}_1, \mathbb{G}_T, hash).$$

KeyGen: The private keys and public keys of the payer, the payee and two supervisors are as follows,

$$\begin{aligned}
sk_1 &= (sk_{a_1}, sk_{a_2}) = (a_1, a_2) \in \mathbb{Z}_n^2, \\
PK_1 &= (PK_{a_1}, PK_{a_2}) = (a_1 G, a_2 G) \in \mathbb{G}_1^2; \\
sk_2 &= (sk_{b_1}, sk_{b_2}) = (b_1, b_2) \in \mathbb{Z}_n^2, \\
PK_2 &= (PK_{b_1}, PK_{b_2}) = (b_1 G, b_2 G) \in \mathbb{G}_1^2; \\
sk_3 &= c_1 \in \mathbb{Z}_n, PK_3 = c_1 \cdot G \in \mathbb{G}_1; \\
sk_4 &= d_1 \in \mathbb{Z}_n, PK_4 = d_1 \cdot G \in \mathbb{G}_1.
\end{aligned}$$

Pay: The payment procedure includes three steps, i.e. the homomorphic encryption scheme *HEnc*, the non-interactive zero-knowledge proof *NIZK*, and the digital signature *Sign*.

- **HEnc:** It takes as input the system parameters SP, a random element $x_1 \in \mathbb{Z}_n$, payment amounts v_1, his private key sk_1, the public keys PK_2, PK_3, PK_4 of the payee and two supervisors, and computes as follows:

$$\begin{aligned}
y_1 &:= hash(x_1 || sk_{a_1} \cdot PK_{b_1} || sk_{a_2} \cdot PK_{b_2}) \\
C_1 &:= (C_{1,1}, C_{1,2}, C_{1,3}) \\
&:= (x_1, y_1 \cdot G, \hat{e}(G, G)^{v_1} \cdot \hat{e}(PK_3, PK_4)^{y_1})
\end{aligned}$$

In the same way, it takes as input the system parameters SP, a random element $x_2 \in \mathbb{Z}_n$, payment amounts v_2, his private key sk_1, the public keys PK_1, PK_3, PK_4 of himself and two supervisors, and computes as follows:

$$\begin{aligned}
y_2 &:= hash(x_2 || sk_{a_1} \cdot PK_{a_1} || sk_{a_2} \cdot PK_{a_2}) \\
C_2 &:= (C_{2,1}, C_{2,2}, C_{2,3}) \\
&:= (x_2, y_2 \cdot G, \hat{e}(G, G)^{v_2} \cdot \hat{e}(PK_3, PK_4)^{y_2})
\end{aligned}$$

Obviously, he can encrypt multiple payment amounts $v_i, i \geq 3$ for other payees as follows

$$\begin{aligned} y_i &:= hash(x_i||sk_{a_1} \cdot PK_{i_1}||sk_{a_2} \cdot PK_{i_2}), \\ C_i &:= (C_{i,1}, C_{i,2}, C_{i,3}) \\ &:= (x_i, y_i \cdot G, \hat{e}(G,G)^{v_i} \cdot \hat{e}(PK_3, PK_4)^{y_i}), \end{aligned}$$

where $PK_i = (PK_{i_1}, PK_{i_2})$ are the other payees' public keys.

The unspent amounts of the payer read from the blockchain has the following two expressions

$$\begin{aligned} y_0 &:= hash(x_0||sk_{0_{0,1}} \cdot PK_{a_1}||sk_{0_{0,2}} \cdot PK_{a_2}) \\ C_0 &:= (C_{0,1}, C_{0,2}, C_{0,3}) \\ &:= (x_0, y_0 \cdot G, \hat{e}(G,G)^{v_0} \cdot \hat{e}(PK_3, PK_4)^{y_0}) \\ y_0{}' &:= hash(x_0{}'||sk_{a_1} \cdot PK_{a_1}||sk_{a_2} \cdot PK_{a_2}) \\ C_0 &:= (C_{0,1}{}', C_{0,2}{}', C_{0,3}{}') \\ &:= (x_0{}', y_0{}' \cdot G, \hat{e}(G,G)^{v_0{}'} \cdot \hat{e}(PK_3, PK_4)^{y_0{}'}) \end{aligned}$$

The unspent amounts v_0 spent by user 0 whose private key is $sk_0 = (sk_{0_{0,1}}, sk_{0_{0,2}})$, and the unspent amount $v_0{}'$ is a change that the payer spends to himself.

- **NIZK:** The non-interactive zero-knowledge proof *NIZK* includes two protocols, i.e. the Sigma protocol and the Bulletproofs.

The Sigma protocol is used to prove that the sum of the unspent ciphertext amounts $v_0 + v_0{}'$ is equal to the sum of the ciphertext amounts $v_1 + v_2$ that need to be paid. More specifically, the payer needs to prove that

$$ZK_{Sigma}\{v_0 + v_0{}' = v_1 + v_2 | C_0, C_0{}', C_1, C_2\},$$

so he needs to prove that he knows a witness $\omega = (y_0 + y_0{}') - (y_1 + y_2)$, where

$$\begin{aligned} y_0 &= hash(x_0||sk_{0_{0,1}} \cdot PK_{a_1}||sk_{0_{0,2}} \cdot PK_{a_2}), \\ y_0{}' &= hash(x_0{}'||sk_{a_1} \cdot PK_{a_1}||sk_{a_2} \cdot PK_{a_2}), \\ y_1 &= hash(x_1||sk_{a_1} \cdot PK_{b_1}||sk_{a_2} \cdot PK_{b_2}), \\ y_2 &= hash(x_2||sk_{a_1} \cdot PK_{a_1}||sk_{a_2} \cdot PK_{a_2}). \end{aligned}$$

The value y_0 is equivalent to the following equation due to the Diffie-Hellman key exchange method

$$\begin{aligned} \tilde{y}_0 &= hash(x_0||sk_{a_1} \cdot PK_{0_{0,1}}||sk_{a_2} \cdot PK_{0_{0,2}}) \\ &= hash(x_0||sk_{a_1}sk_{0_{0,1}} \cdot G||sk_{a_2}sk_{0_{0,2}} \cdot G) \\ &= hash(x_0||sk_{0_{0,1}} \cdot PK_{a_1}||sk_{0_{0,2}} \cdot PK_{a_2}) \\ &= y_0 \end{aligned}$$

As the payer knows $\tilde{y}, x_0, x_0', x_1, x_2, sk_{a_1}, sk_{a_2}, PK_{a_1}, PK_{a_2}, PK_{0_{0,1}}, PK_{0_{0,2}}$, so he really knows the witness $\omega = (y_0 + y_0') - (y_1 + y_2)$. Therefore, the payer can prove the following discrete logarithm equation by using the Sigma protocol

$$(C_{0,3} \cdot C_{0,3}')/(C_{1,3} \cdot C_{2,3}) = \hat{e}(PK_3, PK_4)^{\omega}.$$

The Bulletproofs protocol is used to prove that all ciphertext amounts v_1, v_2 that need to be paid are positive.

$$ZK_{Bulletproofs}\{v_1 \geq 0, v_2 \geq 0 | C_1, C_2\}$$

- **Sign:** The signature scheme can use any secure scheme such as ECDSA (Johnson et al., 2001), BLS scheme (Boneh et al., 2001), etc. It takes as input all the above data *data* and his private key sk_1, and returns a signature σ

$$\sigma \leftarrow Sign(sk_1, data)$$

Let $data = (C_0, C_0', C_1, C_2, ZK_{Sigma}, ZK_{Bulletproofs})$.

Ver: The verification procedure includes signature verification Ver_{sign}, Sigma verification Ver_{Sigma} and Bulletproofs verification $Ver_{Bulletproofs}$. They are the same as in the system model and can be used in the black-box model. If all procedures, i.e. signature verification, Sigma verification and Bulletproofs verification, output valid and no double-spending, then accept and record in the blockchain, else reject.

PayeeDec: It takes as input the system parameters *SP*, his private key sk_2, the public keys of the payer and two supervisors PK_1, PK_3, PK_4, the homomorphic ciphertext $C_{1,1}, C_{1,3}$, and computes as follows:

$$y_1' := hash(C_{1,1} || sk_{b_1} \cdot PK_{a_1} || sk_{b_2} \cdot PK_{a_2})$$
$$\hat{e}(G,G)^{v_1'} := C_{1,3}/\hat{e}(PK_3, PK_4)^{y_1'}$$

As the payee knows the payment amounts v_1'', he can verify $\hat{e}(G,G)^{v_1''} = \hat{e}(G,G)^{v_1'}$. If holds, then accept, else reject. Besides, as v_1 is not a big random element in $\mathbb{Z}_n$, he can search a value v_1' such that $\hat{e}(G,G)^{v_1''} = \hat{e}(G,G)^{v_1'}$ to find it. In subsequent description, these two methods will be used frequently and we will omit them.

Similarly, the payer can decrypt his change v_2

$$y_2' := hash(C_{2,1} || sk_{a_1} \cdot PK_{b_1} || sk_{a_2} \cdot PK_{b_2})$$
$$\hat{e}(G,G)^{v_2'} := C_{2,3}/\hat{e}(PK_3, PK_4)^{y_2'}$$

PayerDec: It takes as input the system parameters *SP*, his private key sk_1, the public keys of the payee and two supervisors PK_2, PK_3, PK_4, the homomorphic ciphertext $C_{1,1}, C_{1,3}$, and computes as follows:

$$y_1' := hash(C_{1,1} || sk_{a_1} \cdot PK_{b_1} || sk_{a_2} \cdot PK_{b_2})$$
$$\hat{e}(G,G)^{v_1'} := C_{1,3}/\hat{e}(PK_3, PK_4)^{y_1'}$$

Similarly, the payer can decrypt the other amounts v_2

$$y_2{}' := hash(C_{2,1}||sk_{a_1} \cdot PK_{b_1}||sk_{a_2} \cdot PK_{b_2})$$
$$\hat{e}(G,G)^{v_2{}'} := C_{2,3}/\hat{e}(PK_3, PK_4)^{y_2{}'}$$

Therefore, the payer can decrypt his payment to summarise payment activities and use them for planning future payment activities.

Supervisor1Dec: It takes as input the system parameters SP, his private key sk_3, the public key of the other supervisor PK_4, the homomorphic ciphertext $C_{1,2}, C_{1,3}$, and computes as follows:

$$\hat{e}(G,G)^{v_1{}'} := C_{1,3}/\hat{e}(C_{1,2}, PK_4)^{sk_3}$$

Similarly, he can decrypt the other amounts v_2

$$\hat{e}(G,G)^{v_2{}'} := C_{2,3}/\hat{e}(C_{2,2}, PK_4)^{sk_3}$$

Supervisor2Dec: It takes as input the system parameters SP, his private key sk_4, the public key of the other supervisor PK_3, the homomorphic ciphertext $C_{1,2}, C_{1,3}$, and computes as follows:

$$\hat{e}(G,G)^{v_1{}'} := C_{1,3}/\hat{e}(C_{1,2}, PK_3)^{sk_4}$$

Similarly, he can decrypt the other amounts v_2

$$\hat{e}(G,G)^{v_2{}'} := C_{2,3}/\hat{e}(C_{2,2}, PK_3)^{sk_4}$$

Therefore, these two supervisors can independently decrypt all users' payments to master the whole economic dynamism and detect illegal transactions.

Theorem 1: *In the **PayeeDec** procedure, the payee's and the payer's decryption is consistent.*

The payee's decryption is as follows

$$\begin{aligned} y_1{}' &= hash(C_{1,1}||sk_{b_1} \cdot PK_{a_1}||sk_{b_2} \cdot PK_{a_2}) \\ &= hash(x_1||sk_{b_1}sk_{a_1} \cdot G||sk_{b_2}sk_{a_2} \cdot G) \\ &= hash(x_1||sk_{a_1} \cdot PK_{b_1}||sk_{a_2} \cdot PK_{b_2}) \\ &= y_1 \\ \hat{e}(G,G)^{v_1{}'} &= C_{1,3}/\hat{e}(PK_3, PK_4)^{y_1} \\ &= \hat{e}(G,G)^{v_1} \cdot \hat{e}(PK_3, PK_4)^{y_1}/\hat{e}(PK_3, PK_4)^{y_1} \\ &= \hat{e}(G,G)^{v_1} \end{aligned}$$

The payer's decryption is as follows

$$\begin{aligned} y_2{}' &= hash(C_{2,1}||sk_{a_1} \cdot PK_{b_1}||sk_{a_2} \cdot PK_{b_2}) \\ &= hash(x_2||sk_{a_1} \cdot PK_{b_1}||sk_{a_2} \cdot PK_{b_2}) \\ &= y_2 \end{aligned}$$

$$
\begin{aligned}
\hat{e}(G,G)^{v_2{}'} &= C_{2,3}/\hat{e}(PK_3,PK_4)^{y_2} \\
&= \hat{e}(G,G)^{v_2}\cdot\hat{e}(PK_3,PK_4)^{y_2}/\hat{e}(PK_3,PK_4)^{y_2} \\
&= \hat{e}(G,G)^{v_2} \\
y_2{}' &= hash(C_{2,1}||sk_{a_1}\cdot PK_{b_1}||sk_{a_2}\cdot PK_{b_2}) \\
&= hash(x_2||sk_{a_1}\cdot PK_{b_1}||sk_{a_2}\cdot PK_{b_2}) \\
&= y_2 \\
\hat{e}(G,G)^{v_2{}'} &= C_{2,3}/\hat{e}(PK_3,PK_4)^{y_2} \\
&= \hat{e}(G,G)^{v_2}\cdot\hat{e}(PK_3,PK_4)^{y_2}/\hat{e}(PK_3,PK_4)^{y_2} \\
&= \hat{e}(G,G)^{v_2}
\end{aligned}
$$

Theorem 2: *The payer's decryption is consistent.*

$$
\begin{aligned}
y_1{}' &= hash(C_{1,1}||sk_{a_1}\cdot PK_{b_1}||sk_{a_2}\cdot PK_{b_2}) \\
&= hash(x_1||sk_{a_1}\cdot PK_{b_1}||sk_{a_2}\cdot PK_{b_2}) \\
&= y_1 \\
\hat{e}(G,G)^{v_1{}'} &= C_{1,3}/\hat{e}(PK_3,PK_4)^{y_1} \\
&= \hat{e}(G,G)^{v_1}\cdot\hat{e}(PK_3,PK_4)^{y_1}/\hat{e}(PK_3,PK_4)^{y_1} \\
&= \hat{e}(G,G)^{v_1} \\
y_2{}' &= hash(C_{2,1}||sk_{a_1}\cdot PK_{b_1}||sk_{a_2}\cdot PK_{b_2}) \\
&= hash(x_2||sk_{a_1}\cdot PK_{b_1}||sk_{a_2}\cdot PK_{b_2}) \\
&= y_2 \\
\hat{e}(G,G)^{v_2{}'} &= C_{2,3}/\hat{e}(PK_3,PK_4)^{y_2} \\
&= \hat{e}(G,G)^{v_2}\cdot\hat{e}(PK_3,PK_4)^{y_2}/\hat{e}(PK_3,PK_4)^{y_2} \\
&= \hat{e}(G,G)^{v_2}
\end{aligned}
$$

Theorem 3: *The first supervisor's decryption is consistent.*

$$
\begin{aligned}
\hat{e}(G,G)^{v_1{}'} &= C_{1,3}/\hat{e}(C_{1,2},PK_4)^{sk_3} \\
&= \hat{e}(G,G)^{v_1}\cdot\hat{e}(PK_3,PK_4)^{y_1}/\hat{e}(y_1\cdot G,PK_4)^{sk_3} \\
&= \hat{e}(G,G)^{v_1}\cdot\hat{e}(G,G)^{sk_3sk_4y_1}/\hat{e}(G,G)^{y_1sk_4sk_3} \\
&= \hat{e}(G,G)^{v_1} \\
\hat{e}(G,G)^{v_2{}'} &= C_{2,3}/\hat{e}(C_{2,2},PK_4)^{sk_3} \\
&= \hat{e}(G,G)^{v_2}\cdot\hat{e}(PK_3,PK_4)^{y_2}\cdot\hat{e}(y_2\cdot G,PK_4)^{sk_3} \\
&= \hat{e}(G,G)^{v_2}\cdot\hat{e}(G,G)^{sk_3sk_4y_2}/\hat{e}(G,G)^{y_2sk_4sk_3} \\
&= \hat{e}(G,G)^{v_2}
\end{aligned}
$$

Theorem 4: *The second supervisor's decryption is consistent.*

$$\begin{aligned}
\hat{e}(G,G)^{v_1'} &= C_{1,3}/\hat{e}(C_{1,2},PK_3)^{sk_4} \\
&= \hat{e}(G,G)^{v_1}\cdot\hat{e}(PK_3,PK_4)^{y_1}/\hat{e}(y_1\cdot G,PK_3)^{sk_4} \\
&= \hat{e}(G,G)^{v_1}\cdot\hat{e}(G,G)^{sk_3sk_4y_1}/\hat{e}(G,G)^{y_1sk_4sk_3} \\
&= \hat{e}(G,G)^{v_1} \\
\hat{e}(G,G)^{v_2'} &= C_{2,3}/\hat{e}(C_{2,2},PK_3)^{sk_4} \\
&= \hat{e}(G,G)^{v_2}\cdot\hat{e}(PK_3,PK_4)^{y_2}\cdot\hat{e}(y_2\cdot G,PK_3)^{sk_4} \\
&= \hat{e}(G,G)^{v_2}\cdot\hat{e}(G,G)^{sk_3sk_4y_2}/\hat{e}(G,G)^{y_2sk_4sk_3} \\
&= \hat{e}(G,G)^{v_2}
\end{aligned}$$

5. Security

For the ciphertext $C_1 = (C_{1,1}, C_{1,2}, C_{1,3})$, where

$$\begin{aligned}
C_1 &= (C_{1,1}, C_{1,2}, C_{1,3}) \\
&= (x_1, y_1\cdot G, \hat{e}(G,G)^{v_1}\cdot\hat{e}(PK_3,PK_4)^{y_1}), \\
y_1 &= hash(x_1||sk_{a_1}\cdot PK_{b_1}||sk_{a_2}\cdot PK_{b_2}),
\end{aligned}$$

an adversary can launch an attack from the angle of the payer, the payee and the two supervisors. However, the decryption operations of the payer and the payee are equivalent, and the decryption operations of the two supervisors are also equivalent. Therefore, we need to prove the following two theorems.

Theorem 5: *Suppose the hash function hash is a random oracle. From the angle of the payer or the payee, if the CDH problem is hard, then our homomorphic encryption scheme is provably secure in the classic IND-CPA security model with reduction loss* $L = 1$.

Theorem 6: *Suppose the hash function hash is a random oracle. From the angle of one of the two supervisors, if the DBDH problem is hard, then our homomorphic encryption scheme is provably secure in the classic IND-CPA security model with reduction loss* $L = 2$.

If the adversary attack from the angle of one of the two supervisors, then $C_{1,1}$ *is useless, and the* y_1 *is a random element in* $\mathbb{Z}_n$. *Therefore, the ciphertext can be reduced as follows*

$$C_1 = (C_{1,2}, C_{1,3}) = (y_1\cdot G, \hat{e}(G,G)^{v_1}\cdot\hat{e}(PK_3,PK_4)^{y_1})$$

Therefore, theorem 6 *is a bit easier than theorem* 5, *so we prove theorem 6 first in the classic security model, and prove theorem* 5 *s.*

5.1. Security proof

Suppose there exists an algebra adversary $\mathcal{A}$ who can break the above homomorphic encryption scheme from the angle of one of the two supervisors in the IND-CPA security

model, in time t with non-negligible advantage ε. We can construct a simulator $\mathcal{B}$ to solve the DBDH problem. Given as input a problem instance $(G, a \cdot G, b \cdot G, c \cdot G, Z)$, the simulator $\mathcal{B}$ controls the random oracle *Hash*, runs the algebra adversary $\mathcal{A}$, and computes as follows.

Setup. Let system parameter be $SP = \mathbb{G}_1, G, n, \mathbb{G}_T, \hat{e}$ and *Hash* be the random oracle controlled by the simulator $\mathcal{B}$. The simulator $\mathcal{B}$ sets the public key as

$$PK_3 = b \cdot G, c \cdot G$$

The public key is available from the problem instance.

Challenge. The algebra adversary $\mathcal{A}$ outputs two amounts v_0, v_1 to be challenged. The simulator $\mathcal{B}$ chooses a random coin $\xi \in \{0, 1\}$, a random element $R^* \in \mathbb{G}_T$, and sets the challenge ciphertext CT^* as

$$CT^* = (a \cdot G, \hat{e}(G, G)^{v_\xi} \cdot Z),$$

where $a \cdot G$ and Z is from the problem instance. Let $y_1 = a$. If

$$Z = \hat{e}(b \cdot G, c \cdot G)^{y_1}.$$

Then, the challenge ciphertext CT^* is

$$CT^* = (y_1 \cdot G, \hat{e}(G, G)^{v_1} \cdot \hat{e}(b \cdot G, c \cdot G)^{y_1})$$

Therefore, the challenge ciphertext CT^* is a correct ciphertext from the point of view of the adversary.

Guess. The algebra adversary $\mathcal{A}$ outputs a guess ξ' of ξ. The simulator outputs true if $\xi' = \xi$. Otherwise, false.

This completes the simulation and the solution. The advantage of solving the DBDH problem is

$$\Pr_{Adv} = \left(\frac{1}{2} + \frac{\xi}{2}\right) - \frac{1}{2} = \frac{\xi}{2}.$$

The reduction loss is 2. Let T_s denote the time cost of the simulation. We have $T_s = O(1)$. Therefore, simulator $\mathcal{B}$ will solve the DBDH problem with $(t + T_s, \varepsilon/2)$. This completes the proof of the theorem 6.

As shown in Table 1, Suppose there exists an algebra adversary $\mathcal{A}$ who can break the above encryption scheme in the IND-CPA security model, in time t with non-negligible

Table 1. Theoretical comparison.

Scheme	*sk*	*PK*	*C*	Enc	Dec	Assum	Loss
(Diament et al., 2004) Payer	n	$\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_T\rvert$	$\mathbb{G}_1^e + \mathbb{G}_T^e + \mathbb{G}_T^+ + \hat{e}$	–	BDH	$q_H/2$
(Diament et al., 2004) Payee	n	$\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_T\rvert$	–	$\mathbb{G}_T^e + \mathbb{G}_T^+ + \hat{e}$	BDH	$q_H/2$
(Diament et al., 2004) Supervisor	n	$\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_T\rvert$	–	$\mathbb{G}_T^e + \mathbb{G}_T^+ + \hat{e}$	BDH	$q_H/2$
Payer	$2n$	$2\lvert\mathbb{G}_1\rvert$	$n + \lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_T\rvert$	$3\mathbb{G}_1^e + \mathbb{G}_T^e + \mathbb{G}_T^+ + \hat{e}$	$2\mathbb{G}_1^e + \mathbb{G}_T^e + \mathbb{G}_T^+ + \hat{e}$	CDH	1
Payee	$2n$	$2\lvert\mathbb{G}_1\rvert$	$n + \lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_T\rvert$	–	$2\mathbb{G}_1^e + \mathbb{G}_T^e + \mathbb{G}_T^+ + \hat{e}$	CDH	1
Supervisor1	n	$\lvert\mathbb{G}_1\rvert$	$n + \lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_T\rvert$	–	$\mathbb{G}_T^e + \mathbb{G}_T^+ + \hat{e}$	DBDH	2
Supervisor2	n	$\lvert\mathbb{G}_1\rvert$	$n + \lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_T\rvert$	–	$\mathbb{G}_T^e + \mathbb{G}_T^+ + \hat{e}$	DBDH	2

advantage ε. We can construct a simulator $\mathcal{B}$ to solve the CDH problem. Given as input a problem instance $(G, a \cdot G, b \cdot G)$, the simulator $\mathcal{B}$ controls the random oracle *Hash*, runs the algebra adversary $\mathcal{A}$, and works as follows.

Setup. Let the system parameter be $SP = \mathbb{G}_1, G, n, \mathbb{G}_T, \hat{e}$ and *Hash* be the random oracle controlled by the simulator $\mathcal{B}$. The simulator $\mathcal{B}$ randomly chooses $z_1, z_2 \in_R \mathbb{Z}_p$ and sets the public key as

$$PK = (G_1, G_2) = (a \cdot G, z_1 \cdot G + z_2 a \cdot G)$$

where $\alpha = a$ and $\beta = z_1 + z_2 a$. The public key can be computed from the problem instance $(G, a \cdot G, b \cdot G)$ and the chosen parameters z_1, z_2.

Hash-Query. The simulator $\mathcal{B}$ prepares a Hash-Query list to record all queries and responses. In the beginning, the Hash-Query list is empty. Let the *i*-th hash query be

$$\gamma_i = (x_i || b_i \cdot G_1 || b_i \cdot G_2).$$

If γ_i is a new hash query, then the simulator $\mathcal{B}$ randomly chooses $y_i \in \mathbb{Z}_n$ and sets $Hash(\gamma_i) = y_i$. The simulator $\mathcal{B}$ responds to this query with $Hash(\gamma_i)$ and adds (γ_i, y_i) to the Hash-Query list. If γ_i is already in the Hash-Query list, the simulator $\mathcal{B}$ responds to this query with the existed random value in the Hash-Query list.

Challenge. The algebra adversary $\mathcal{A}$ outputs two amounts v_0, v_1 to be challenged. The simulator $\mathcal{B}$ randomly chooses a random element $x^* \in_R \mathbb{Z}_n$, five group elements $G^*, G_3, G_4 \in \mathbb{G}_1$, and a random element $R^* \in \mathbb{G}_T$. He sets the challenge ciphertext CT^* as

$$CT^* = (x^*, G^*, R^*).$$

The challenge ciphertext can be seen as an encryption of the message $v_\xi \in \{v_0, v_1\}$ using the random coin ξ if

$$\begin{aligned} y^* &= hash(x^* || b \cdot G_1 || b \cdot G_2), \\ G^* &= y^* \cdot G, \\ \hat{e}(G_3, G_4)^{y_1} &= R^* / \hat{e}(G, G)^{v_\xi} \end{aligned}$$

Then, the challenge ciphertext CT^* is

$$\begin{aligned} CT^* &= (C_{1,1}, C_{1,2}, C_{1,3}) \\ &= (x^*, y^* \cdot G, \hat{e}(G, G)^{v_1} \cdot \hat{e}(G_3, G_4)^{y^*}), \\ y^* &= hash(x^* || b \cdot G_1 || b \cdot G_2) \end{aligned}$$

Therefore, if there is no hash query on $\gamma^* = (x^* || b \cdot G_1 || b \cdot G_2)$ to the random oracle *Hash*, then the challenge ciphertext CT^* is a correct ciphertext from the point of view of the adversary.

Guess. The algebra adversary $\mathcal{A}$ outputs a guess or $\perp$. In the above simulation, the challenge hash query is defined as.

$$Q^* = (b \cdot G_1 || b \cdot G_2) = (ba \cdot G, b(z_1 + z_2 a) \cdot G)$$

As $(\gamma_1, y_1), (\gamma_2, y_2), \ldots, (\gamma_{q_H}, y_{q_H})$ are all in the Hash-Query list, where each query $\gamma_i = (x_i || b_i \cdot G_1 || b_i \cdot G_2)$. If x_i does not satisfy this structure, we can delete it. As the following

equation holds

$$z_1 b + z_2 b^* a = b^* z_1 + b^* z_2 a$$

Then, the following equation holds

$$(z_1 b + z_2 b^* a) \cdot G = (b^* z_1 + b^* z_2 a) \cdot G \Rightarrow$$
$$z_1 (b \cdot G) + z_2 (b^* a \cdot G) = b^* (z_1 \cdot G + z_2 a \cdot G).$$

It is equivalent to the following equation

$$z_1 (b \cdot G) + z_2 (b^* G_1) = b^* G_2.$$

Therefore, the simulator $\mathcal{B}$ can find the query $\gamma^* = (x^* || b^* \cdot G_1 || b^* \cdot G_2)$ from the Hash-Query list satisfying the above equation. He uses it as the solution to the CDH problem instance. Therefore, there is no reduction loss in the above security simulation. In other words, if the algebra adversary $\mathcal{A}$ can break the encryption scheme in polynomial-time t with non-negligible probability ε, then we can construct a simulator $\mathcal{B}$ to break the CDH problem instance in polynomial-time $t + T_s$ with the same probability ε, where T_s is the time cost of the simulation. This completes the proof of the theorem 5.

6. Performance and comparison

Our homomorphic encryption scheme is extending from Diament et al.'s (2004) dual receiver public encryption scheme, which has dual receivers. We add an extra receiver in their scheme and achieve the property of homomorphism. Besides, our scheme has a special characteristic that it supports both the payer and the sender to decrypt a message.

The Diament et al.'s (2004) scheme can be converted to a homomorphic encryption scheme easily, and we can make a detailed comparison with it. In this paper, the payer corresponds to the sender, the payee corresponds to the receiver 1, the supervisor 1 corresponds to the receiver 2, and the supervisor 2 corresponds to the receiver 3.

In table 5.1, the second to the fourth columns show the size of the private key, the public key and the ciphertext. The fifth to the sixth columns show the computation complexity of encryption and decryption algorithms. The seventh to the eighth columns show the computational hard problems and reduction loss respectively.

The bilinear maps computational complexity denotes as $\hat{e}$, exponent operation and additive operation on group $\mathbb{G}_1, \mathbb{G}_T$ denotes as $\mathbb{G}_1^e, \mathbb{G}_1^+, \mathbb{G}_T^e, \mathbb{G}_T^+$, respectively. We omitted the computation complexity of the hash function.

For the receiver 3,4 the length of the private key and public key, and decryption complexity is the same. Although the length of the private key and public key of the sender and receiver 1 in our scheme is twice over Diament's scheme, the sender can decrypt it in our scheme. Besides, computational hard problems of our scheme are CDH and DBDH, where the DBDH problem is a bit harder than the BDH problem, and the CDH problem is a standard and is one of the most widely accepted hard problems. Furthermore, the reduction loss of Diament's scheme is $q_H/2$ while the reduction loss of our scheme is only 1 and 2 respectively, which is very tight. More specially, with a fixed security level, the exponent of Diament's scheme should $q_H/4$ bits longer than our scheme. In other words, with a fixed exponent of the elliptic curve group, our scheme is more secure than Diament's scheme for

Table 2. Performance comparison.

Scheme	Enc Time (*ms*)	Dec Time (*ms*)
(Diament et al., 2004) Sender (Payer)	30.61	–
(Diament et al., 2004) Receiver1 (Payee)	–	17.42
(Diament et al., 2004) Receiver2 (Supervisor1)	–	17.43
Payer	57.05	38.33
Payee	–	38.34
Supervisor1	–	24.85
Supervisor2	–	24.83

q_H/4-bits. Fully homomorphic encryption is also too leisure to be practicable. Fully homomorphic encryption is still a relatively new technology for data protection and usefulness. However, it's an intriguing idea, and that we're sure to see quicker variants that may be used in a number of scenarios.

Platform: Centos7.7 with kernel 3.10.0-1062.el7, @2.50 GHz, x86_64, 192GB RAM. We use SHA256 as the concrete construction of the hash function, and select the Type A elliptic curve $y^2 = x^3 + x$ in *jpbc* library. The exponent of the elliptic curve group in our encryption scheme is 160 bits, and the length of an elliptic curve group member is 512 bits. CentOS is a Linux distribution that offers a neighbourhood, open and free computing environment that is operationally consistent including its original source. Software archives contain latest version of the software than those included in the standard distribution.

As is shown in Table 2, the encryption time of the sender(payer) and the decryption time of the receiver 1 and 2 (payee and supervisor) are 30.61, 17.42 and 17.43 ms, respectively. The encryption time of the sender(payer) and the decryption time of the sender (payer), receiver 1,2,3 (payee, supervisor 1 and supervisor 2) is 57.05, 38.33, 38.34 ms, 24.85 and 24.83 ms, respectively. Although the encryption and decryption time of our scheme is a bit longer than Diament's scheme, our scheme supports payment amounts decryption by the sender (payer), an additional receiver and tight security reduction loss.

7. Conclusion

We propose a blockchain-based transaction system with payment statistics and supervision. We use a special homomorphic encryption scheme to protect the privacy of payment while the payer, the payee and two supervisors can decrypt it independently. We believe that the payer can decrypt it and use it for payment statistics is of great importance in financial statistics. Besides, the two supervisors can decrypt it independently so that they can master the whole economic dynamism and detect illegal transactions, which is also important in financial statistics.

We show that if the DBDH issue is difficult, our homomorphic encryption technique is secure and efficient in the conventional IND-CPA security framework with reduction loss $L = 2$ from the perspective of one of the two supervisors. If the CDH problem is difficult, our homomorphic encryption approach is highly robust in the conventional IND-CPA security model without even any reduction loss in the view of payer or payee's perspective. Finally, by comparing with Diament's scheme, our homomorphic scheme only increases a little length of ciphertext, but supports payment amounts decryption by the payer, an additional receiver and tight security reduction loss.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolic, M., Cocco, S. W., & Yellick, J. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. *EuroSys, 30*, 1–30, 15.

Ben-Sasson, E., Chiesa, A., Tromer, E., & Virza, M. (2014). *Succinct non-Interactive zero Knowledge for a von Neumann Architecture*. USENIX Security Symposium (pp. 781–796).

Bissias, D. D., Ozisik, A. P., Levine, B. N., Liberatore, M. (2014). *Sybil-resistant mixing for bitcoin*. WPES (pp. 149–158).

Boneh, D., & Boyen, X. (2008). Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, *21*(2), 149–177. https://doi.org/10.1007/s00145-007-9005-7

Boneh, D., Lynn, B., & Shacham, H. (2001). *Short signatures from the Weil pairing*. ASIACRYPT (pp. 514–532).

Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., & Maxwell, G. (2018). *Bulletproofs: Short proofs for confidential transactions and more*. IEEE Symposium on Security and Privacy (pp. 315–334).

Damgård, I. (2000). *Efficient concurrent zero-knowledge in the auxiliary string model*. EUROCRYPT (pp. 418–430).

Diament, T., Lee, H. K., Keromytis, A. D., & Yung, M. (2004). *The dual receiver cryptosystem and its applications*. CCS (pp. 330–343).

Feng, H., Liu, J., Wu, Q., & Li, Y. (2020). *Traceable ring signatures with post-quantum Security*. CT-RSA (pp. 442–468).

Fujisaki, E., & Suzuki, K. (2007). *Traceable ring signature*. Public Key Cryptography (pp. 181–200).

Groth, J. (2016). *On the size of pairing-based non-interactive arguments*. EUROCRYPT (2, pp. 305–326).

Johnson, D., Menezes, A., & Vanstone, S. (2001). The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, *1*(1), 36–63. https://doi.org/10.1007/s102070100002

Koshy, P., Koshy, D., & McDaniel, P. D. (2014). *An analysis of anonymity in bitcoin using P2P network traffic*. Financial Cryptography (pp. 469–485).

Maxwell G. (2013). *Coinjoin: Bitcoin privacy for the real world*. https://bitcointalk.org/index.php

Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M., & Savage, S. (2013). *A fistful of bitcoins: Characterizing payments among men with no names*. Internet Measurement Conference (pp. 127–140).

Miers, I., Garman, C., Green, M., & Rubin, A. D. (2013). *Zerocoin: Anonymous distributed e-cash from bitcoin*. IEEE Symposium on Security and Privacy (pp. 397–411).

Miller, A., Xia, Y., Croman, K., Shi, E., & Song, D. (2016). *The honey badger of BFT protocols*. CCS (pp. 31–42).

Moser, M., & Narayanan, A. (2019). Effective cryptocurrency regulation through blacklisting. *Preprint*.

Nakamoto, S. (2019). "Bitcoin: A peer-to-peer electronic cash system".

Noether, S. (2015). Ring signature confidential transactions for monero. *IACR Cryptol. EPrint Arch*, *2015*, 1098.

Noether, S., & Mackenzie, A. (2016). Ring confidential transactions. *Ledger, 1*, 1–18. https://doi.org/10.5195/ledger.2016.34

Reid, F., & Harrigan, M. (2011). *An analysis of anonymity in the bitcoin system*. SocialCom/PASSAT (pp. 1318–1326).

Ruffing, T., Moreno-Sanchez, P., & Kate, A. (2014). *CoinShuffle: Practical decentralized coin mixing for bitcoin*. ESORICS (2, pp. 345–364).

Wang, Q., Li, R., Wang, Q., & Chen, S. (2021). "Non-fungible token (NFT): Overview, evaluation, opportunities and challenges," *arXiv preprint arXiv:2105.07447*.

Wang, Y.-C., Chen, C.-L., & Deng, Y.-Y. (2021). Authorization mechanism based on blockchain technology for protecting museum-digital property rights. *Applied Sciences*, *11*(3), 1085. https://doi.org/10.3390/app11031085

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, *151*(2014), 1–32.

Wust, K., Kostiainen, K., Capkun, V., & Capkun, S. (2018). Prcash: Centrally-issued digital currency with privacy and regulation. *IACR Cryptol. EPrint Arch*, *2018*, 412.

Zhao, C. (2014). Graph-based forensic investigation of bitcoin transactions.

Zhaofeng, M., Lingyun, W., Xiaochang, W., Zhen, W., & Weizhe, Z. (2019). Blockchain-enabled decentralized trust management and secure usage control of IoT big data. *IEEE Internet of Things Journal*, *7*(5), 4000–4015. https://doi.org/10.1109/JIOT.2019.2960526

Ziegeldorf, J. H., Grossmann, F., Henze, M., Inden, N., & Wehrle, K. (2015). *CoinParty: Secure multi-party mixing of bitcoins*. CODASPY (pp. 75–86).