Two Efficient Regulatory Confidential Transaction Schemes

Min Yang[1,2], Changtong Xu[1,2], Zhe Xia[3], Li Wang[4], and Qingshu Meng[4]

[1] School of Cyber Science and Engineering, Wuhan University, Wuhan, China
[2] Key Laboratory of Aerospace Information Security and Trust Computing, China
[3] School of Computer Science, Wuhan University of Technology, Wuhan, China
[4] Wuhan Tianyu Information Industry Co.,Ltd, Wuhan, China
qsmeng@126.com

Abstract. Blockchain has been widely used in finance, logistics, copyright and other fields with its outstanding characteristics such as non-centralization, collective maintenance, openness, transparency and non-tamperability. However, as transactions are stored in plaintext in the blockchain for public verification, the anonymity and privacy of users can not be guaranteed and this hampers many financial applications. How to protect the privacy of transactions is worthy further research.
In this paper, we have proposed two regulatory and efficient confidential transaction schemes using homomorphic encrytion and zero-knowledge proof. The first one improves the efficiency of the existing ElGamal based scheme while preserves its privacy. The second one employs the Paillier encryption with homomorphic property and it empowers regulators with greater power to obtain transaction-related specific content. The core of ElGamal based scheme is the Modified ElGamal algorithm, which changes the form of the standard ElGamal algorithm and expands it into four ciphertexts such that (m, r) in the transaction can be decrypted. The Paillier based scheme is mainly to combine Paillier encryption with ElGamal encryption. Contrast to other ElGamal based scheme, the combination makes any token amount can be directly decrypted without calculating a discrete logarithm problem. As any (m, r) in transactions can be decrypted directly, game theory is applied to further reduce transaction size. In our construction, transactions are about 1.1KB.

Keywords: confidential transactions · zero-knowledge proof · regulatory · game theory · modified ElGamal · modified Paillier

Contents

1 Introduction

In most blockchain systems such as Bitcoin [Sat08] and Ethereum [Woo14], the content of a transaction is broadcast in plaintext. After the miner collects and verifies the validity, the transaction is stored on-chain, and each node can access all the on-chain contents. These characteristics also bring the problem of privacy protection [DSPSNAHJ18]. With the much concerning with privacy problem, it is extremely important to protect the privacy of on-chain content. The privacy of transactions can be divided into two aspects: one is anonymity, meaning that the sender and receiver of a transaction are anonymous; the other is confidentiality, meaning that the amount is known only to both parties in the transaction. To achieve confidential transaction, many scholars have carried out relevant research, and many projects, such as Zcash [SCG$^+$14], Monero [NM$^+$16], Zether [BAZB20], have proposed a variety of solutions using cryptography tools. However, enhancing the privacy of transactions brings new challenges to the regulation of transactions [Fin18]. Cryptographic techniques are used in many blockchain applications and academic studies to ensure the privacy of participants, but in some cases the overuse or even abuse of privacy protections can make it difficult to regulate and audit on-chain transactions. So, under the condition of protecting user privacy, new research needs to give regulators greater authority to access transaction information, and find a balance between regulation and privacy to achieve controlled privacy.

This paper proposes two new schemes using homomorphic encryption and zero-knowledge proofs. The first one is an improvement to the existing ElGamal based schemes in efficiency while keeping privacy. And the second one is based on homomorphic Paillier encryption algorithm and empowers regulators greater powers to obtain transaction-related specific content. The core of ElGamal based scheme is the Modified ElGamal algorithm, which changes the standard ElGamal algorithm to be additive homomorphic and expands it into four ciphertexts such that (m, r) in the transaction can be decrypted. The Paillier based scheme is mainly to combine Paillier encryption with ElGamal encryption. Contrast to other Elgamal based scheme, the combination make larger token amounts can be directly decrypted without calculating a discrete logarithm problem. As any (m, r) in transactions can be decrypted directly, game theory is applied to further reduce transaction size. The transactions in both schemes are confidential for nodes, but can be publicly verified by nodes. The Paillier based scheme is a regulator-friendly scheme where the regulator can supervise every on-chain transaction, effectively eliminate the illegal transactions such as money laundering. An efficient, privacy preserving and regulatory transaction system will promote the adoption of blockchain applications.

1.1 Related Work

1.1.1 The Importance of Privacy Protection. In Cryptocurrencies, attackers can analyze a user's trading habits through transaction records stored on-chain. In the application of finance [Pae17], attackers can not only analyze the user's personal trading habits with the help of the on-chain content, but also infer macro trends of the whole market, which is damage to users' privacy, and leaks the core data of financial enterprises in some ways. In energy industry, transaction records may leak energy exchange information, which is very important and sensitive information for a country. In short, for the original system with completely transparent records, analysts can analyze the transaction rules by records, and obtain amount and relationship in the transactions, which makes the user's privacy seriously threatened. Therefore, many privacy protection related researches have emerged that can divide existing solutions into two categories, depending on the use of commitment scheme.

One is to use the Pedersen Commitment [Ped92] scheme, the main problem with such schemes is the commitment opening must be transferred to the receiver off-chain. Maxwell [Max15] first proposed the concept of confidential transactions and apply them to Bitcoin, using Pedersen Commitment and OR-proofs to establish a payment mechanism that hides the amount, and applies range proof to ensure the correctness of the transaction. Mimblewimble/Grin [Poe16, FOS19], [Gri] improves Maxwell's work by reducing signature consumption. Another research direction is anonymity. A lot of work has been done to enhance anonymity through Coinjoin [Max13]. The third direction is to improve privacy and anonymity. Monero [NM$^+$16] uses a similar approach to Maxwell to achieve privacy protection, also based on UTXO model, which enhances the anonymity of transactions using ring signatures [MP15] and StealthAddress. However, the signature size used by Monroe increases linearly as ring members increase. Zcash [SCG$^+$14] offers two trading modes, one is a transparent transaction similar to Bitcoin, the other is confidential transaction using zk-SNARKs [zks] proofs, but zk-SNARKs requires generating a larger Common Reference Strings (CRS) in advance.

The other is to use ElGamal encryption scheme which has been studied more recently. The advantage of this scheme is that the ciphertext part can not only keep the amount confidential, perform homomorphic calculation, but also decrypt the transaction amount. Quisquis [FMMO19] proposed by Fauzi et al. is an anonymous confidential transaction system designed to solve problems that exist in Monroe and Zcash, such as the growing number of UTXO set. Quisquis combines UTXO and account models, using a one-time account and shuffle method to anonymity, while using ElGamal encryption to complete confidentiality. Bünz et al. [BAZB20] proposed Zether, a smart contract on Ethereum. They modified standard ElGamal encryption to be additive homomorphic and used the ElGamal encryption to hide balances and transfer amount, and acquired anonymity using ring signature. Chen et al. [CMTA20] proposed PGC and twisted ElGamal by changing the standard ElGamal algorithm, and the second part of twisted ElGamal is Pedersen Commiment which can directly be used in Bulletproofs [BBB$^+$18] protocol. All the three schemes design accompanying zero-knowledge proofs using Sigma protocol and Bulletproofs, but in different ways to solve the interoperation of ElGamal encryption and Bulletproof. Quisquis introduced ElGamal commitment, and used Sigma protocol to prove consistency of ElGamal commitment and Pedersen commitment are committed to the same amount, then used Bulletproofs to the Pedersen commitment. Zether proposed $\Sigma-$Bullets, which directly combined the Sigma protocol with Bulletproofs. Given an arithmetic circuit, the linear combination of the wires in the circuit is equal to some witness of a Sigma protocol. This enhancement in turn enables proofs on many different encodings such as ElGamal encryptions, Pedersen commitments in different groups or using different generators. PGC modified the standard ElGamal algorithm, which private key is independent of the commitment, so that Bulletproofs could be used directly on the twisted ElGamal algorithm.

1.1.2Regulatory Studies. While trading systems provide privacy protection, transactions should also comply with regulatory requirements. A simple regulatory solution is to have the participant provide private key for the regulator, but this exists a huge security risk and is inconsistent with the privacy policy. Zcash has two features [Zca] that enable the disclosure of shielded transaction information. Both of them need to generate a key which can be provided to a regulator, thereby allowing them to view the details of the transaction. As mentioned in PGC, the range proof and zero-knowledge proof can be used to determine that the regulatory requirements are met. However, the specific amount of the transaction cannot be obtained by the regulator, and the content of regulation is limited, resulting in some audit, statistical and other functions cannot be completed.

1.2 Problems with the existing scheme

1.2.1 Commitment Based Scheme. Many current works for implementing confidential transactions are implemented by homomorphism commitments, such as Pedersen commitment, FO commitment. But additional channel is needed to transfer the opening (m, r) of the commitment, which can be encrypted and stored on-chain or transmitted over private channels. In order to decrypt the on-chain commitment directly, people come up with a solution that combines the ElGamal encryption and the Bulletproofs.

1.2.2 Combine ElGamal Encryption and Bulletproofs. Zether poposed Σ-Bullets which is an extension of Bulletproofs. The ciphertexts of Zether-ElGamal are $(C_1 = g^r, C_2 = pk^r g^m)$, and calculate $g^m = C_2/C_1^{sk}$. C_2 part cannot be directly used for Bulletproofs, need Sigma protocol to prove that both the $pk^r g^m$ and the Bulletproofs are encrypted to the same (m, r). However, this requires the special design and analysis of a more complex Sigma protocol. A similar approach is used in Quisquis.

PGC modified the Zether-ElGamal encryption to directly use Bulletproof in a black-box manner, changing the Zether-ElGamal to Pedersen Commitment $(pk^r, g^r h^m)$. However, this scheme also requires brute-force to calculate

m [Sha71], which is based on discrete logarithms and can be calculated only if the transaction amount m is small (less than 2^{32}).

In addition to the limitation of the transaction amount of Zether, PGC and Quisiquis, new randomness r also need to be selected to re-encrypt sender's balance. There are three disadvantages: (1) that private key is required to prove the equality for re-encryption, which may bring some security risks (2) re-encryption adds extra computation (3) the newly added ciphertexts increase the transaction size.

1.3 Our Contributions: Modified ElGamal

We proposed the Modified ElGamal, new ciphertexts are $C_1 = pk^{r_0}, C_2 = g^{r_0}h^m, C_3 = pk^{r_1}, C_4 = r_0 g^{r_1}$, public key $pk = g^{sk}$, decryption calculation $r_0 = C_4/C_3^{sk^{-1}}, h^m = C_2/C_1^{sk^{-1}}$. However, h^m requires brute-force to compute m, which can be quickly calculated when m is small (less than 2^{32}). And in most cases, the transaction amount m is known to both parties, and the receiver only needs to decrypt h^m with the sk and verify it with the known m. The benefits of this are as following:(1) we can run the bulletproofs on C_2 directly, without a complicated Sigma protocol like Zether;(2) (m, r_0) can be calculated, without additional channel to transmit, and re-encryption is not required for range proof of sender's balance;(3) Achieving the same functionality with fewer on-chain contents. In terms of on-chain data complexity and time complexity for a confidential transaction, our scheme is superior to the existing schemes such as Zether and PGC.

1.4 Our Contributions: Modified Paillier

It is observed that when using ElGamal based scheme to decrypt the transaction amount, a discrete logarithm problem needs to be calculated, which will be much more difficult when the transaction amount is very large. For most trading systems, especially for larger companies, the volume of trade is much larger than 2^{32}. For example, the minimum unit of Ethereum is Wei ($1Ether = 10^{18}wei$). It is difficult to obtain transaction amounts on such a large scale by brute-force attack. On the basis of previous studies, we propose a new scheme, where ($C_1 = pk^{r_0} \bmod n^2, C_2 = k^m h^{r_0} \bmod n^2, C_3 = pk^{r_1} \bmod n^2, C_4 = k^{r_0} h^{r_1} \bmod n^2$) can be decrypted according to the Paillier [Pai99] encryption, and amount m and the randomness r_0 can both be decrypted directly. As (m, r) can be decrypted, the receiver can use them to check if the ciphertexts are right. If the ciphertexts are found illegal, the receiver can submit a ZK-proof to the blockchain and make the transaction invalid. the sender will lose his tokens and cause no harm to the system. By game theory, the sender will not construct illegal ciphertext and it is unnecessary to generate proofs for the legality of ciphertexts. The new solution ensures the security and correctness of the transaction while greatly reducing on-chain data.

In order to give the regulators more power than ordinary users and complete the regulation more effectively, we propose a new method that can compute the private key securely. Under the condition of guaranteeing the privacy of users' transactions, the supervision party can master all the on-chain transactions, and achieve controllable privacy. According to Paillier encryption, the user's private key $sk = L(pk^u \bmod n^2)/L(h^u \bmod n^2)$ can be calculated by the system's private key u, which can be used only if it is authorized by multiple trusted parties.

2 Preliminaries

2.1 Basic Notations.

In this article, λ denotes the security parameter, and a negligible probability is written as $negl(\lambda)$. Let $GroupGen$ be a polynomial time algorithm, input as 1^λ. The output of the $GroupGen$ for the Modified ElGamal scheme is $(p, g, \mathbb{G})$, p is a large prime number, $\mathbb{G}$ is a cyclic group of order p, g is the generator of the group $\mathbb{G}$, $\mathbb{Z}_p$ represents the integer ring of modulus p. The output of the $GroupGen$ for the Modified Paillier scheme is $(k, n, \mathbb{Z}^*_{n^2})$, n is the modulus of the product of two large prime numbers, $\mathbb{Z}^*_{n^2}$ represents the multiplication group of natural numbers less than n^2 which are mutual prime with n^2. Let $x \leftarrow_R \mathbb{Z}_p$ represent a randomness x from $\mathbb{Z}_p$.

2.2 Assumptions

Definition 1 (Decisional Diffie-Hellman Assumption). *Let $\mathbb{G}$ be the group with the order of large prime p, and g be the generator of $\mathbb{G}$, and randomly select $x, y, z \in \mathbb{Z}_p$. Then the following two distributions*
· Random quadruple $R = (g, g^x, g^y, g^z) \in \mathbb{G}^4$
· Quadruple $D = (g, g^x, g^y, g^{xy}) \in \mathbb{G}^4$(called Diffie-Hellman quadruple, short for DH quadruple).
is computationally indistinguishable and is called the DDH assumption.

Specifically, for any adversary $\mathcal{A}$, $\mathcal{A}'s$ advantage in distinguishing R from D is negligible:

$$Adv_{\mathcal{A}}(\lambda) = |Pr[\mathcal{A}(R) = 1] - Pr[\mathcal{A}(D) = 1]| \leqslant negl(\lambda)$$

Definition 2 (Discrete Log Relation). *Given g, a generator of $\mathbb{G}$, and h, a random element in $\mathbb{G}$, $log_g h$ is considered difficult to compute. The specific definition is as follows:*

If for all PPT adversary $\mathcal{A}$, we have

$$Pr[\mathcal{A}(g, h) = x \ s.t. \ g^x = h] \leqslant negl(\lambda)$$

It can be said that the discrete logarithm problem is difficult in $\mathbb{G}$.

2.3 Commitment

The non-interactive commitment scheme is composed of the sender and the receiver, mainly divided into three stages. In the key generation stage, input security parameters λ and output public parameters pp such as the public key and private key. In the commitment stage, input the message m from message space M_{pp}, and randomness r from randomness space R_{pp}, and calculate the commitment $Com = Com(m, r)$. In the opening stage, the sender can send (m, r) to the receiver by encrypted ways or some private secure channel so that the receiver can verify the correctness of the commitment. Formal commitment schemes are defined by the following three algorithms.

Setup$\left(1^{\lambda}\right)$: Input the security parameter λ, and output the public parameter pp, which defined the message space M_{pp}, and the randomness space R_{pp}, and the commitment space C.

Com(m, r) : The sender makes a commitment to the message m and randomness r, calculates $C = Com\left(m, r\right)$, and sends C to the receiver.

Open(C, m, r) : The sender sends (m, r) to the receiver, who verifies that the commitment is correct, outputs accept or reject.

Definition 3 (Homomorphism Commitment). *Homomorphism commitment means that the commitment scheme satisfies homomorphism, that is, for messages $m_1, m_2 \in \mathbb{Z}_p$, randomness $r_1, r_2 \in \mathbb{Z}_p$, which satisfies the following formula:*

$$Com\left(m_1, r_1\right) \otimes Com\left(m_2, r_2\right) = Com\left(m_1 + m_2, r_1 + r_2\right)$$

This means that the commitment scheme satisfies additive homomorphism, where $\otimes$ represents an operator, such as multiplication.

Definition 4 (Hiding Commitment). *A hiding commitment scheme refers to $Com\left(m, r\right)$ do not leak any information related to m, protecting the safety of the sender. Let $\mathcal{A}$ be an adversary against hiding, and the advantage of the adversary is defined as*

$$Adv_{\mathcal{A}}\left(\lambda\right) = Pr\left[\beta' = \beta \middle| \begin{array}{c} pp \leftarrow Setup(1^{\lambda}); m_0, m_1 \leftarrow \mathcal{A}(pp); \\ \beta \leftarrow_R \{0,1\}, r \leftarrow_R R_{pp}, C = Com\left(m_{\beta}, r\right); \\ \beta' \leftarrow \mathcal{A}(c) \end{array}\right] - 1/2$$

If $Adv_{\mathcal{A}}\left(\lambda\right) = 0$ for the adversary with unbounded power, then this commitment satisfies perfect hiding, that is the distribution of $Com\left(m_0, r_0\right)$ is the same as $Com\left(m_1, r_1\right)$; If $Adv_{\mathcal{A}}\left(\lambda\right) = negl\left(\lambda\right)$, this commitment satisfies the statistical hiding that is the distribution of $Com\left(m_0, r_0\right)$ and $Com\left(m_1, r_1\right)$ is statistically indistinguishable; If $Adv_{\mathcal{A}}\left(\lambda\right) = negl\left(\lambda\right)$ for adversary with PPT power, this commitment satisfies computational hiding, that is the distribution of $Com\left(m_0, r_0\right)$ and distribution $Com\left(m_1, r_1\right)$ is computationally indistinguishable.

Definition 5 (Binding Commitment). *A binding commitment scheme refers to a commitment C can not be opened into two different (m, r), protecting the safety of the receiver. $\mathcal{A}'s$ advantage is defined as*

$$Adv_{\mathcal{A}}(\lambda) = Pr\left[\begin{matrix} Com(m_0, r_0) = Com(m_1, r_1) \\ \wedge m_0 \neq m_1 \end{matrix} \middle| \begin{matrix} pp \leftarrow Setup(1^{\lambda}); \\ (C, m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(pp) \end{matrix}\right]$$

If $Adv_{\mathcal{A}}(\lambda) = 0$ for the unbounded adversary, this commitment scheme satisfies perfect binding. If $Adv_{\mathcal{A}}(\lambda) = negl(\lambda)$ for the unbounded adversary, this commitment scheme satisfies statistical binding; If for any PPT adversary, $Adv_{\mathcal{A}}(\lambda) = negl(\lambda)$, this commitment scheme satisfies computational binding.

Pedersen Commitment [Ped92]. In the cyclic group $\mathbb{G}$ of prime order p, and $g, h \in \mathbb{G}$ are randomly selected.

Commitment: For the input message $m \in Z_p$, and randomness $r \in Z_p$ and calculate $C \leftarrow g^m h^r \in \mathbb{G}$.

Opening: Using (m, r) to verify the correctness of commitment C. If $C = g^m h^r$, the receiver accepts the commitment to message m, otherwise rejects. Under the discrete logarithm assumption, Pedersen commitment is perfect hiding and computational binding. Pedersen commitment also satisfies additive homomorphism.

Fujisaki-Okamoto Commitment [FO97]. Suppose sender and receiver do not know the decomposition of n, $g \in Z_n^*, h \in (g)$, the order of g and h is large prime, which makes it infeasible to calculate the discrete logarithm in the generated cyclic group. Sender doesn't know $log_g h$ and $log_h g$, randomly selected from $r \in \{-2^s n + 1, 2^s n - 1\}$, calculate $E(m, r) = g^m h^r \bmod n$, send receiver $E(m, r)$ as a commitment to m. Sender doesn't know the decomposition of n and $log_g h$, it's impossible to find $m_1 \neq m_2$ satisfy $E(m_1, r_1) = E(m_2, r_2)$; receiver is also unable to obtain any information about m from $E(m, r)$, which is statistically secure, and the commitment scheme is referred to as the Fujisaki-Okamoto commitment, or FO commitment.

2.4 Combined Signature and Encryption Schemes

A combined signature and encryption scheme is a combination of a signature scheme and a public key encryption scheme that share a key generation algorithm and hence the same keypair (pk, sk). Paterson et al. [PSST11] revisited this topic and gave a generic construction of combined public key scheme from identity-based encryption. The scheme comprises signature scheme $(Setup, KeyGen, Sign, Verify)$ and PKE scheme $(Setup, KeyGen, Enc, Dec)$. When defining a security game against a component of the scheme, the nature of any oracles depends on the required security of the other components. This means that the PKE component is IND-CPA secure even in the presence of a signing oracle, while the signature component is EUF-CMA secure even in the presence of encryption oracle. The formal security definition of the scheme as following:

IND-CPA security in the presence of a signing oracle. Let $(KeyGen, Sign, Verify, Encrypt, Decrypt)$ be a combined signature and encryption scheme. Indistinguishability of the encryption component under an adaptive chosen plaintext attack in the presence of an additional signing oracle is defined through the following game between a challenger and an adversary $\mathcal{A}$. The advantage of $\mathcal{A}$ can be defined in the following experiment:

$$Adv\left(\lambda\right) = Pr\left[\beta' = \beta \left| \begin{array}{c} pp \leftarrow Setup(\lambda); (pk, sk) \leftarrow keyGen(pp); \\ m_0, m_1 \leftarrow \mathcal{A}^{O_{sign}}(pk); \\ \beta \leftarrow_R \{0,1\}; C \leftarrow Enc(pk, m_\beta); \\ \beta' \leftarrow \mathcal{A}^{O_{sign}}(C) \end{array} \right.\right] - 1/2$$

The signature oracle O_{sign} returns the result of signing the message m using the private key sk. The encryption scheme is IND-CPA secure, if no adversary wins the security game by non-negligible advantage, the encryption component is IND-CPA secure in the presence of a signing oracle.

EUF-CMA security in the presence of a decryption oracle. Let $(KeyGen, Sign, Verify, Encrypt, Decrypt)$ be a combined signature and encryption scheme. Existential unforgeability of the signature component under an adaptive chosen message attack in the presence of an additional decryption oracle is defined through the following game between a challenger and an adversary $\mathcal{A}$. The advantage of $\mathcal{A}$ can be defined in the following experiment:

$$Adv\left(\lambda\right) = Pr\left[\begin{array}{c} Verify(pk, m^*, \sigma^*) = 1 \\ \wedge m^* \notin Q \end{array} \left| \begin{array}{c} pp \leftarrow Setup(\lambda); \\ (pk, sk) \leftarrow keyGen(pp); \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{O}_{sign}(pp, pk) \end{array} \right.\right] - 1/2$$

The set Q represents a request to the signing oracle and returns the signed result $Sign(sk, m)$ when the input is m. The encryption scheme is EUF-CMA secure, if no adversary wins the security game by non-negligible advantage, the signature component is EUF-CMA secure in the presence of a decryption oracle.

2.5 Zero-knowledge Proof

The zero-knowledge proof system consists of two parties, called $Prover(P)$ and $Verifier(V)$, where P knows a secret, and after several rounds of interaction between P and V, V believes that P really has the secret, without revealing any information except that the statement is true. For example, P can convince V that a confidential transaction is valid without revealing the exact amount of the transaction. Zero-knowledge proof can be consist of the following three PPT algorithms $(Setup, P, V)$.

Setup algorithm inputs 1^λ, outputs the public parameter pp used in the proof, such as the common reference string(CRS). Let $R \subseteq X \times W$ be the discriminable NP relation in polynomial time, $w \in W$ is the witness to statement x, and the NP language L dependent on the public parameter pp can be defined as

$$L_{pp} = \{x | \exists w : (x, w) \in R\}$$

P and V are a pair of interactive algorithms that use $tr \leftarrow \langle P(s), V(t)\rangle$ to represent the interaction between P and V, where the input for P is s and the input for V is t. We write $\langle P(s), V(t)\rangle = b$ depending on whether the verifier rejects, b = 0, or accepts, b = 1.

Any zero-knowledge proof should satisfy the following three requirements:

(1) Completeness: If the statement is true, the honest verifier will pass the verification. The verifier always returns TRUE if the prover's input is TRUE. That is, for any $(x, w) \in R$, the following relation holds:

$$\Pr\left[\langle P\left(x, w\right), V\left(x\right)\rangle = 1\right] \geq 1 - negl\left(\lambda\right)$$

(2) Soundness: If the statement is false, the verifier cannot pass with any cheating methods. If the input is wrong, the verifier always returns FALSE, that is, for any $x \notin L$, all dishonest prover P^*, the following relation holds:

$$\Pr\left[\langle P^*\left(x\right), V\left(x\right)\rangle = 1\right] \leq negl\left(\lambda\right)$$

(3) Zero-knowledge: No one else can get any information about the input other than the corresponding statement.

Definition 6 (Computational Witness-Extended Emulation). $(Setup, P, V)$ *has witness-extended emulation [BCC⁺16], if there is an expected polynomial time emulator E for all deterministic polynomial time P, and for all interactive adversaries $\mathcal{A}_1, \mathcal{A}_2$, there exists a negligible function $negl\left(\lambda\right)$ such that:*

$$Pr\left[\begin{array}{c} pp \leftarrow Setup(\lambda); \\ (x, s) \leftarrow \mathcal{A}_1(pp); \\ tr \leftarrow \langle P^*(x, s), V(x)\rangle; \\ \mathcal{A}_2(tr) = 1 \end{array}\right] - Pr\left[\begin{array}{c} pp \leftarrow Setup(\lambda); \\ (x, s) \leftarrow \mathcal{A}_1(pp); \\ (tr, w') \leftarrow E^O(x); \\ (x, w') \in R_{pp}; \\ \mathcal{A}_2(tr) = 1 \end{array}\right] \leq negl(\lambda)$$

Where the $O = \langle P^*\left(x, w\right), V\left(x\right)\rangle$ permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards.

In this definition, s can be interpreted as the state of P^*, including the randomness. So, whenever P^* is able to make a convincing argument in state s, E can extract the witness. This is why we call it an argument of knowledge.

Definition 7 (Public coin). *An argument of knowledge $(Setup, P, V)$ is public coin if all messages sent from V are chosen uniformly at random and independently of the P's messages.*

Definition 8 (Range Proof). *For a commitment scheme $(Setup, Com)$ over message space M_{pp} and randomness space R_{pp}, a zero-knowledge range proof is a argument of knowledge for the following relation:*

$$L = \{C | \exists m \in M_{pp}, r \in R_{pp} \ s.t. \ C = Com\left(m, r\right) \wedge \ m \in [a, b]\}$$

Definition 9 (Sigma Protocol). *Sigma Protocol [Dam02] is used by P to prove that P knows some secrets. The main procedure of the protocol is as follows:*

(1) Commitment: P calculates a commitment c.
(2) Challenge: V sends a random challenge e to P.
(3) Response: After receiving challenge e, P calculates response z and sends it to V.
(4) Verification: V checks the response and outputs to accept or reject. A sigma protocol satisfies standard completeness, special soundness and zero-knowledge.

Definition 10 (Standard completeness). *For any x and the correct (c, e, z) and (c, e', z'), where $e \neq e'$, the witness w can be calculated.*

Definition 11 (Perfect Special Honest Verifier Zero-Knowledge). *A public coin argument of knowledge $(Setup, P, V)$ is a perfect special honest verifier zero-knowledge argument if there exists a PPT simulator S for the interactive adversaries $\mathcal{A}_1, \mathcal{A}_2$ satisfying the following relations.*

$$Pr\left[\begin{array}{l|c} (x,w) \in R; & pp \leftarrow Setup\left(\lambda\right); \\ \mathcal{A}_2\left(tr\right) = 1 & (x,w) \leftarrow \mathcal{A}_1\left(pp\right); \\ & tr \leftarrow \langle P^*\left(x,w\right), V\left(x\right)\rangle \end{array}\right] = Pr\left[\begin{array}{l|c} (x,w) \in R; & pp \leftarrow Setup\left(\lambda\right); \\ \mathcal{A}_2\left(tr\right) = 1 & (x,w) \leftarrow \mathcal{A}_1\left(pp\right); \\ & tr \leftarrow S(x) \end{array}\right]$$

In the definition, the proof system is zero-knowledge if the adversary cannot distinguish between real scheme and simulated scheme.

3 Security model

For the sake of simplicity as Quisquis [FMMO19], we focus solely on the transaction layer of a cryptocurrency and assume network-level or consensus-level attacks are out of scope. Intuitively, The confidential transaction system should provide authenticity, confidentiality and soundness. Correctness requires that the adversary cannot create a transaction, and the transaction can only be generated by the honest sender, that is, the attacker cannot make a transfer from the honest account. For an adversary, the only way to success is to calculate the sk of the honest account. Confidentiality requires that only the sender and receiver can obtain the amount of a confidential transaction, and that the encrypted amount is indistinguishable from m_0 or m_1 with non-negligible advantage. Soundness requires that the sender cannot generate an illegal but verified transaction, which against cheating on his own. The main purpose of the security experiment is to capture the way an adversary can interact with the honest user in the trading system. For example, the adversary can establish a transaction through the honest user or generate a valid transaction by himself.

An adversary can initiate a specific transaction using *transact* queries by an honest user, or inject a malicious transaction. It can also get the private key through *disclose* queries for any account in the system, except for the account in the *challenge* stage. Below we describe oracles adversaries can access.

$O_{register}$: Adversary $\mathcal{A}$ queries this oracle to register an honest account, and challenger $\mathcal{CH}$ puts the result of the query into an initially empty list called T_{honest}. After receiving the query, $\mathcal{CH}$ responds as follows: $\mathcal{CH}$ generates the sequence number i and a keypair (pk_i, sk_i), returns $(i, pk_i, sk_i, balance, C)$ to $\mathcal{A}$, and records it in T_{honest}.

$O_{disclose}(pk)$: $\mathcal{A}$ queries this oracle with an honest public key, if pk_i in T_{honest}, removes it from T_{honest} to $T_{corrupt}$, which records some dishonest account. Then $\mathcal{CH}$ returns $(i, pk_i, sk_i balance, C)$ to $\mathcal{A}$. This kind of oralce captures that the adversary can control an honest account.

$O_{verify}(tx)$: $\mathcal{A}$ queries this oracle with a transaction tx. If it is a valid transaction, $\mathcal{CH}$ returns 1; otherwise, $\mathcal{CH}$ returns 0.

$O_{inject}(pk_s, pk_r, v)$: $\mathcal{A}$ queries this oracle with parameter (pk_s, pk_r, v) to generate a confidential transaction, where $pk_s \in T_{corrupt}$. If $VerifyTX(tx) = 1$, $\mathcal{CH}$ updates the state of relevant account. This means that $\mathcal{A}$ can generate a transaction itself (possibly a malicious transaction).

4 Our Construction

4.1 Confidential Transaction System

***Setup*$(\mathbf{1}^{\boldsymbol{\lambda}})$** : Input a security parameter λ to generate relevant parameters for encryption and zero-knowledge proof.

***CreateAddress*$(\mathbf{1}^{\boldsymbol{\lambda}})$** : Input a security parameter λ, and execute PKE.KeyGen(pp) to get a keypair(pk, sk), then generate the account according to the encryption scheme designed in this paper, calculate $C_0 = Enc(PK, m_0, r_0)$ as the initial balance of the account, where $m_0 = 0$. It then outputs (pk, sk) and uses the public key as the address for subsequent transactions.

$\boldsymbol{Transact(sk_s, pk_s, pk_r, m)}$: On input sender's keypair (pk_s, sk_s) and receiver's address pk_r, suppose the sender transfer m to the receiver. And $E_s^* = (pk_s^{r^*}, g^{r^*} h^{m^*})$ represent the sender's current balance, the specific transaction process is as follows:

Sender: Sender first checks whether $m \in [0, 2^n - 1]$ and $m^* \in [0, 2^n - 1]$, and encrypts m with pk_s and pk_r respectively to get $E_s = (C_1 = pk_s^{r_0}, C_2 = g^{r_0} h^m), E_r = (C_1 = pk_r^{r_0}, C_2 = g^{r_0} h^m, C_3 = pk_r^{r_1}, C_4 = r_0 g^{r_1})$, the ciphertext of the transaction amount has five parts. The ciphertext of the balance consists of two parts: $E_s' = (pk_s^{r^* - r_0}, g^{r^* - r_0} h^{m^* - m}) = (pk_s^{r'}, g^{r'} h^{m'})$, and r_0 can be calculated. Since the r_0 of each transaction can be solved, it is considered that the randomness in the sender's balance is known and r' can be calculated. The sender is also required to use zero-knowledge proofs to prove (1) range proofs for the transaction amount m, the transaction amount is within the specified range, and get π_1 (2) range proofs for sender's current balance, which is a positive value, and get π_2 (3) the randomness in E_{r1} and E_{r2} have the same r_0, and the corresponding π_3 is generated. More formally, a user proves the following statement:

$$S_{range1} = \{(pk_s, E_s) : \exists r_0, r_1, m \ s.t. \ E_s = Enc(pk_s, m, r_0, r_1) \wedge m \in [0, 2^n - 1]\}$$

$$S_{range2} = \{(pk_s, E_s') : \exists r', m' \ s.t. \ E_s' = Enc(pk_s, m', r') \wedge m' \in [0, 2^n - 1]\}$$

$$S_{equal} = \{(pk_r, E_{r1}, E_{r2}) : \exists r_0, m \ s.t. \ E_{r1} = pk_r^{r_0} \wedge E_{r2} = g^{r_0} h^m\}$$

Here, only part of the ciphertext of the receiver is proved to be valid, because (1) E_{r1} and E_{r2} with the same randomness can compute the correct transaction amount m (2) the correctness of r_0 can be ensured by receiver's verification, without increase on-chain content (3) adversary gets no benefit from constructing $pk_s^{r_0}$ and does not change the balance in the commitment. Then run the signature algorithm to the transaction with the sender's private key. And the final transaction is $tx = (pk_s, pk_r, E_s, E_s', E_r, \pi_1, \pi_2, \pi_3)$ and corresponding signature Sig. There is no need for the sender to prove the ciphertext is correct, that is, the ciphertext of E_s and E_r is encrypted with the same (m, r_0) with the public keys of both parties. Instead, it is the receiver to verify the correctness of ciphertext. If it is a malicious transaction, the receiver call the smart contract to punish the sender, eliminating the sender's idea of evil from the sources.

VerifyTX(tx, sig) : Verify the validity of Sig with the sender's public key, verify $E_s' = E_s^*/E_s$ and π_1, π_2, π_3. If all the verifications pass, miners confirm that the transaction is valid and record it on the blockchain via consensus protocol.

ConfirmTX(tx) : After the receiver obtains the on-chain transaction information, verify $E_s' = E_s^*/E_s$ and validity of π_1, π_2. Then decrypt $E_r = (C_1 = pk_r^{r_0}, C_2 = g^{r_0} h^m, C_3 = pk_r^{r_1}, C_4 = r_0 g^{r_1})$ to get $\overline{r}_0 = C_4/C_3^{sk^{-1}}, h^{\overline{m}} = C_2/C_1^{sk^{-1}}$, receiver check if $g^{\overline{r}_0} h^{\overline{m}} = g^{r_0} h^m$. If the verification pass, tx is a valid transaction, and the reveiver update corresponding balance and randomness. If not, then this is a malicious transaction, indicating that the sender changes the randomness r_0 in C_4 so that the receiver cannot solve the correct randomness, but the receiver can calculate $pk^{\overline{r}_0}$ and compare with on-chain content to determine whether it is a malicious transaction. During the challenge stage, the receiver can prove that the transaction is malicious with proof of fraud, and the honest receiver will execute the $reportTX(TX)$. If the receiver does not report the transaction until the end of the challenge stage, the transaction is considered valid.

Report(tx) : When the receiver finds that is a malicious transaction, the receiver reports the transaction to the smart contract. Record the wrong $\overline{r}_0$ on the blockchain and prove that $\overline{r}_0$ is actually calculated by the on-chain ciphertext. More formally, a user proves the following statement:

$$S_{enc} = \left\{(sk_r, \overline{r}_0) : \exists \overline{r}_0, \ s.t. \ C_4 = E_{r3}^{sk_r^{-1}} \overline{r}_0\right\}$$

where E_{r3} represents the third ciphertext of E_r, and generates a zero-knowledge proof π_4. After the smart contract verification, it is confirmed that this is a malicious transaction, and then it performs a homomorphism calculation on the receiver's account $E_s^* = E_s' \cdot E_s$, returns to the state before the malicious transaction is completed, and destroys the token corresponding to this transaction. Because normal user only needs to input transaction amount m when performing confidential transactions, the reason for the above malicious transaction is that the attacker changed the randomness in $r_0 g^{r_1}$ to make it different from the randomness in $g^{r_0} h^m$, it can be considered that this kind of transaction must be maliciously constructed by the sender, so the smart contract can destroy the token in the transaction to punish the malicious sender.

***ReadBalance*$(\boldsymbol{E}, \boldsymbol{sk})$** Taking the sender as an example, input the private key sk_s of the sender and the corresponding ciphertext E_s to obtain the balance $m = Dec\left(E_s, sk_s\right)$ of the sender.

Above all, the attack cannot succeed in this process, from the perspective of Game Theory, an adversary will not execute an attack that is unprofitable or even at a loss, and does not effect on the honest receiver, so we can assume that malicious transactions won't appear and the system can operate safely.

4.2 Security Proof

Theorem 1. *The confidential transaction system satisfies correctness if there is no PPT adversary to win the following game with non-negligible advantage.*

Proof of correctness.

Game 0. A real experiment for correctness. The interaction between adversary $\mathcal{A}$ and Challenger $\mathcal{CH}$ is as follows.

1. Setup: $\mathcal{CH}$ generates the system, sends the public key and other public parameters to $\mathcal{A}$.

2. Training: $\mathcal{A}$ queries the following oracles $O_{register}, O_{disclose}\left(pk\right),$ $O_{verify}\left(tx\right), O_{transact}\left(pk_s, pk_r, v\right), O_{inject}\left(pk_s, pk_r, v\right)$ adaptively, and $\mathcal{CH}$ answers these queries with corresponding results.

3. Challenge: If the adversary generates a legitimate transaction through an honest user, then the adversary succeeds, otherwise fails.

Game 1. Game 1 is the same as Game 0, except that the extractor runs every time an adversary creates a malicious transaction. If an adversary generates a transaction through $O_{transact}\left(pk_s, pk_r, m\right)$, the extractor can extract the witness $w = \left(sk_s, balance, m, r\right)$.

Game 2. Game 2 is the same as Game 1, except that $\mathcal{CH}$ randomly selects an honest user that the adversary wants to forge at the beginning, such as pk_j from T_{honest}. If the adversary obtains the private key of pk_j in the training stage or $pk_s \neq pk_j$ in the challenge stage, $\mathcal{CH}$ terminates and starts Game 2 again. Obviously, Game 2 executes a round in polynomial time, let W be the event that $\mathcal{CH}$ does not terminate, the probability of $Pr[W] \geq \frac{1}{Q_{honest}}$. Q_{honest} is the number of the honest set.

Game 3. Game 3 is the same as Game 2, except that the real zero-knowledge proof system is replaced with the simulator and generates a simulated π. When the adversary accesses the oracle $O_{transact}\left(pk_s, pk_r, m\right)$, the oracle runs $tx \leftarrow Transact(sk_s, pk_s, pk_r, m)$, but the zero-knowledge proof parameters such as CRS are replaced by simulated parameters.

The above experiments show that the system is zero-knowledge and the adversary cannot obtain additional information from the interaction. If $\mathcal{A}$ succeeds it means that $\mathcal{A}$ controls an honest account to execute a transaction, $w = (sk_s, balance, m, r)$ can be obtained from the extractor, indicating that $\mathcal{A}$ calculates the sender's private key sk_s from public parameter, which is impossible according to Theorem 6.

Theorem 2. *The confidential transaction system satisfies confidentiality, if there is no PPT adversary to win the following game with non-negligible advantage.*

If the adversary can tell $E = (C_1 = pk^{r_0}, C_2 = g^{r_0} h^{m_\beta}, C_3 = pk^{r_1}, C_4 = r_0 g^{r_1})$ is encrypted to m_0 or m_1. The adversary can only distinguish from the evidence π of zero knowledge proof, or according to the final ciphertext discrimination. The difference between E_{m_0} and E_{m_1} is the randomness and the encrypted message m_β, and we conclude that the adversary cannot distinguish the ciphertext based on hiding property of commitment.

Proof of confidentiality.

Game 0. A real experiment, the interaction between adversary $\mathcal{A}$ and Challenger $\mathcal{CH}$ is as follows.

1. Setup: $\mathcal{CH}$ generates the system, sends the public key and other public parameters to $\mathcal{A}$.

2. Training Stage 1: $\mathcal{A}$ queries the following oracles $O_{register}, O_{disclose}\left(pk\right)$, $O_{verify}\left(tx\right), O_{transact}\left(pk_s, pk_r, v\right), O_{inject}\left(pk_s, pk_r, v\right)$ adaptively, and $\mathcal{CH}$ answers these queries with corresponding results.

3. Challenge: The adversary selects pk_s, pk_r, and two transaction amounts m_0, m_1, where $pk_s, pk_r \in T_{honest}$. Both m_0, m_1 can form a legal transaction issued by pk_s. $\mathcal{CH}$ selects random bits β, runs $tx \leftarrow Transact(sk_s,$
$pk_s, pk_r, m_\beta)$, and sends tx to $\mathcal{A}$.

4. Training Stage 2: $\mathcal{A}$ queries the following oracles $O_{register}, O_{disclose}\left(pk\right)$, $O_{verify}\left(tx\right), O_{transact}\left(pk_s, pk_r, v\right), O_{inject}\left(pk_s, pk_r, v\right)$ adaptively, and $\mathcal{CH}$ answers these queries as stage 1. But at this time $\mathcal{A}$ is denied to use pk_s and pk_r to query the oracle $O_{disclose}\left(pk\right)$, and pk_s to query the oracle $O_{transact}\left(pk_s, pk_r, v\right)$.

5. Guess: $\mathcal{A}$ outputs β' and wins if $\beta = \beta'$.

Game 1. Game 1 is the same as game 0, except that the real zero-knowledge proof system is replaced with the simulator and generates a simulated π. Based on the property of NIZK, we can conclude that Game 0 and Game 1 are indistinguishable.

Game 2. Game 2 is the same as game 1, except that changing the encryption of m_0 in Game 1 to the encryption of m_1, Game 1 and Game 2 are indistinguishable because of the hiding property of the commitment.

Game 3. Game 3 is the same as game 2, except that simulator is replaced with the real zero-knowledge proof system. Game 2 and Game 3 are indistinguishable because of the property of NIZK. So we have:

$$|Pr(G_3) - Pr(G_0)| < negl(\lambda)$$

Theorem 3. *The confidential transaction system satisfies soundness if there is no PPT adversary to win the following game with non-negligible advantage.*

Soundness requires that the sender cannot generate an illegal but verified transaction and cannot cheat on his own. A successful attack by an adversary means that the transferred amount is greater than the account balance, and the transaction is valid, indicating that the adversary has constructed another pair of opening (m', r') that can also open the commitment. The binding property of commitment shows that the adversary cannot success.

The specific proof process is similar to the correctness proof, omitted here.

4.3 Regulation of Transactions

We use zero-knowledge proof to regulate the legality of transactions, mainly proving the following two aspects: the total amount of transactions within a period of time is in a certain range, and a transaction can be opened in accordance with the requirements of the regulatory.

For each transaction E_i needs to prove relationship as blow:

$$S_{sum} = \{(pk, E_i, MAX) : \exists sk\ s.t.\ sum = \Sigma_{i=1}^{n} E_i \wedge Dec(sum, sk) < MAX\}$$

$E_i = (C_1 = pk_i^{r_0}, C_2 = g^{r_0}h^{m_i}, C_3 = pk_i^{r_1}C_4 = r_0 g^{r_1})$, according to additive homomorphic of Modified ElGamal, we can calculate the sum of these values, m_i and r_i satisfy $sum_m = \Sigma_{i=1}^{n} m_i, sum_r = \Sigma_{i=1}^{n} r_i$, and prove that the sum of values in a given range.

If the user opens a particular transaction, the relation to prove can be expressed as:

$$S_{open} = \{(pk, E, m) : \exists sk\ s.t.\ pk^r = (C_2/h^m)^{sk} \wedge pk = g^{sk}\}$$

That is, the private key is used to prove that the amount m corresponding to this transaction is indeed encrypted in the ciphertext.

5 Instantiation of the Conifdential Transaction System

In this section, we instantiate our transaction system by instantiating the newly proposed Modified ElGamal encryption and Schnorr signature and then designing a zero-knowledge proof scheme with bulletproofs.

5.1 Instantiation of Signature and Encryption Part

5.1.1 Modified ElGamal

· $Setup(1^\lambda)$: run $(p, g, \mathbb{G}) \leftarrow GroupGen\left(1^\lambda\right)$, select $h \leftarrow_R \mathbb{G}^*$, set$(p, g, h, \mathbb{G})$ as public parameter pp and $m, r \in \mathbb{Z}_p$.

· $KeyGen(pp)$: select $sk \leftarrow_R \mathbb{Z}_p$ and calculate $pk = g^{sk}$.

· $Enc(pk, m, r)$: calculate $C_1 = pk^{r_0}, C_2 = g^{r_0}h^m, C_3 = pk^{r_1}, C_4 = r_0 g^{r_1}$, output $E = (C_1, C_2, C_3, C_4)$.

· $Dec(sk, C)$: according to $E = (C_1, C_2, C_3, C_4)$, calculate $h^m = C_2/C_1^{sk^{-1}}$, $r_0 = C_4/C_3^{sk^{-1}}$, m can be calculated from h^m.

In general, the transaction amount m is known to both parties of the transaction, so user can take known m into calculation. If user wants to quickly calculate m from h^m, then m needs to be small enough(less than 2^{32}), and most transactions are less than 2^{32}, so user can uses the algorithm of fast discrete logarithm to compute m efficiently.

Obviously, the new algorithm satisfies correctness and homomorphism, and at the same time, it satisfies IND-CPA security based on DDH assumption in the standard model. The specific proof is given in Appendix.

Kurosawa et al. [Kur02] first proved that in the standard ElGamal encryption, randomness can be reused in the single-plaintext multi-receiver setting, that is, use pk_s and pk_r to encrypt the same (m, r). Zether and PGC also use Kurosawa's result to make their zero-knowledge component more efficient. Our Modified ElGamal encryption scheme is also secure when reusing randomness. This technique not only reduces the size of the transaction, but also makes related zero-knowledge proof more efficient. The specific safety certification is as follows:

Theorem 4. *Modified ElGamal encryption scheme that reuses randomness is IND-CPA secure based on the DDH assumption.*

Game 0. In the real IND-CPA security experiment, the interaction between challenger $\mathcal{CH}$ and adversary $\mathcal{A}$ is as blow. Let S_i be the probability that $\mathcal{A}$ wins in Game i.

1. Setup. $\mathcal{CH}$ generate system and related parameters, sends public keys $pk_0 = g^{sk_0}, pk_1 = g^{sk_1}$ to $\mathcal{A}$.

2. Training. $\mathcal{A}$ generates messages to obtain encrypted ciphertext (bounded polynomial times).

3. Challenges. $\mathcal{A}$ outputs two messages of equal length m_0 and m_1. $\mathcal{CH}$ selects random bit β and randomness r_0, r_1, calculate $X_0 = pk_0^{r_0}, X_1 = pk_1^{r_0}, Y = g^{r_0}h^{m_\beta}, Z_0 = pk_0^{r_1}, Z_1 = pk_1^{r_1}, U = r_0 g^{r_1}$, and send X_0, X_1, Y, Z_0, Z_1, U to $\mathcal{A}$

4. Guess. $\mathcal{A}$ outputs β', and wins if $\beta' = \beta$.

The adversary's advantage in Game 0 can be defined as below.

$$Adv_{\mathcal{A}}(\lambda) = \Pr[S_0] - 1/2$$

Game 1. Same as Game 0, except that $\mathcal{CH}$ picks a random bit β and randomness r_0, r_1, s_0, s_1, compute $X_0 = pk_0^{r_0}, X_1 = pk_1^{r_0}, Y = g^{s_0} h^{m_\beta}, Z_0 = pk_0^{r_1}, Z_1 = pk_1^{r_1}, U = r_0 g^{s_1}$ and send X_0, X_1, Y, Z_0, Z_1, U to $\mathcal{A}$.

In Game 1, ciphertext distribution is independent of β, so $\mathcal{A}$ has no message about β, $\Pr[S_1] = 1/2$. Random quad $(g, g^{s_0}, g^{sk_{0,1}}, g^{sk_{0,1} \cdot r_0}), (g, g^{s_1}, g^{sk_{0,1}}, g^{sk_{0,1} \cdot r_1})$ can be expressed as follows. In the quad (g, g^a, g^b, g^c), assume that $c = c'b, a = c''+c'$, ciphertext can be expressed as $(g^{c_0' b_0}, g^{c_0' b_1}, g^{c_0'}(g^{c_0''} h^m), g^{c_1' b_0}, g^{c_1' b_1}, g^{c_1'}(g^{c_1''} a_0))$, and $(g, g^{c_0' + c_0''}, g^{b_{0,1}}, g^{c_0' b_{0,1}}), (g, g^{c_1' + c_1''}, g^{b_{0,1}}, g^{c_1' b_{0,1}})$ constitute a random quad.

Next, it is proved that the difference between $Pr[S_0]$ and $Pr[S_1]$ is negligible. We construct adversary $\mathcal{B}$ with the same advantage as $\mathcal{A}$ to attack DDH assumption. Given a quad $\left(g, g^a, g^b, g^c\right)$, $\mathcal{B}$ determines whether it is a random quad or a DH quad. $\mathcal{B}$ is constructed as follows.

1. Setup. $\mathcal{B}$ generates system and related parameters, treats g^{b_0}, g^{b_1} as the public keys pk_0 and pk_1, b_0 and b_1 are corresponding private keys, which is unknown to $\mathcal{B}$. Then $\mathcal{B}$ sends $pk_0 = g^{b_0}, pk_1 = g^{b_1}$ to $\mathcal{A}$.

2. Training. $\mathcal{A}$ generates messages to obtain encrypted ciphertext (bounded polynomial times).

3. Challenges. $\mathcal{A}$ outputs two messages of equal length m_0, m_1 and sends them to $\mathcal{B}$. $\mathcal{B}$ selects random bits β, calculate $X_0 = g^{c_0}, X_1 = g^{c_1}, Y = g^{a_0} h^{m_\beta}, Z_0 = g^{c_0'}, Z_1 = g^{c_1'}, U = a_0 g^{a_1}$, sends X_0, X_1, Y, Z_0, Z_1, U to $\mathcal{A}$.

4. Guess. $\mathcal{A}$ outputs β', and wins if $\beta' = \beta$.

If the quad $\left(g, g^a, g^b, g^c\right)$ is a DH quad, that is, $(g, g^{r_0}, g^{sk_{0,1}}, g^{sk_{0,1} \cdot r_0})$, $(g, g^{r_1}, g^{sk_{0,1}}, g^{sk_{0,1} \cdot r_1})$ consist of a DH quad, then $\mathcal{B}$ is the same view as Game 0, where $c = ab$. If $\left(g, g^a, g^b, g^c\right)$ is a random quad, that is, $(g, g^{s_0}, g^{sk_{0,1}}, g^{sk_{0,1} \cdot r_0})$, $(g, g^{s_1}, g^{sk_{0,1}}, g^{sk_{0,1} \cdot r_1})$ consist of a random quad, then $\mathcal{B}$ is the same view as Game 1. Therefore, if $\mathcal{A}$ can distinguish between $\mathcal{B}$ representing Game 0 and Game 1 with non-negligible advantage, then $\mathcal{B}$ can break the DDH assumption with the same advantage.

5.1.2 Signature scheme In the signature part, Schnorr signature [Sch91] that satisfies EUF-CMA security [PS00] was selected for two reasons: Schnorr signature is the same as Modified ElGamal algorithm in the key generation process, and signature procedure and the encryption procedure are unrelated to each other. In addition, Schnorr signature is efficient and multi-signature scheme that can be constructed easily. At present, multi-signature scheme is widely used in blockchain [BDN18].

5.2 Zero-knowledge Proof

5.2.1 Range Proof for Transaction Amount. The transaction generated by the sender $E_s = (C_1 = pk_s^{r_0}, C_2 = g^{r_0} h^m, C_3 = pk_s^{r_1}, C_4 = r_0 g^{r_1})$, we can directly use Bulletproofs for $C_2 = g^{r_0} h^m$ with logarithmic proof size and output the evidence π_1. You can refer to the original Bullteproofs [BBB$^+$18] for more details.

5.2.2 Range Proof for Sender's Balance According to additive homomorphism of Modified ElGamal, we can calculate the balance of sender by $E'_s = E^*_s / E_s = g^{r^*-r} h^{m^*-m} = g^{r'} h^{m'}$. Where r^* can be considered as already known, because r_0 of each received transaction can be calculated from $E = (pk^{r_0}, g^{r_0} h^m, pk^{r_1},$
$r_0 g^{r_1})$, and the randomness of self-initiated transaction is also known, so r', m' is computable. So we can directly use Bulletproofs for $g^{r'} h^{m'}$, and get the evidence π_2.

5.2.3 Aggregating Logarithmic Proofs The two range proofs can also be combined. Both (m, r) of the two transactions are known. Using the method of aggregate range proofs, multiple range proofs can only increase the proof size at logarithmic level. The relationship can be written as

$$S_{range} = \{(C_{s2}, C'_{s2}, m, r_0, m', r') : C_{s2} = g^{r_0} h^m \wedge C'_{s2} = g^{r'} h^{m'} \wedge m \in [0, 2^n - 1] \wedge m' \in [0, 2^n - 1]\}$$

Where (m, r_0) is the transaction amount and the corresponding randomness.

5.2.4 Validity of E_{r1} and E_{r2}. Here we need to prove that E_{r1} and E_{r2} use the same randomness r_0, and the relationship to be proved is

$$S_{equal} = \{(pk_r, E_{r1}, E_{r2}) : \exists r_0, m \ s.t. \ E_{r1} = pk_r^{r_0} \wedge E_{r2} = g^{r_0} h^m\}$$

We construct a non-interactive Sigma protocol using the Fiat-Shamir heuristic [FS86]:
(1) P selects randomness a, b, and calculates $A_1 = pk_r^a, A_2 = g^a h^b$
(2) P computes random challenge $e = Hash(E_{r1}, E_{r2}, A_1, A_2)$
(3) P calculates $s_1 = a + er_0, s_2 = b + em$, and sends A_1, A_2, s_1, s_2 to V
(4) V calculates

$$pk_r^{s_1} = A_1 (E_{r_1})^e$$

$$g^{s_1} h^{s_2} = A_2 (E_{r_2})^e$$

If the two equality verifications are both true, then output the evidence π_3.

Security Proof of Sigma protocol

Completeness: From the above process, the correctness is clear if the P and V are executed as specified in the protocol.

Special Soundness: For certain (A_1, A_2), suppose there are two different accepting transcripts $(e, s = (s_1, s_2))$ and $(e', s' = (s'_1, s'_2))$, $e \neq e'$, then r_0, m can be extracted by the following method. We have $s_1 = a + er_0, s'_1 = a + e'r_0$, from which we can get $r_0 = (s_1 - s'_1) / (e - e')$. And we can extract m with the same method.

Special Honest-Verifier Zero-Knowledge: Assuming there is a polynomial-time simulator S, where the simulator picks a random challenge e and response s_1, s_2, and computes $A_1 = pk_r^{s_1}(E_{r1})^{-e}, A_2 = g^{s_1}h^{s_2}(E_{r2})^{-e}$, it is clear that $(A_1, A_2, E_{r1}, E_{r2}, s_1, s_2)$ is a valid transcript, and for any probabilistic polynomial-time verifier, these parameters are computationally indistinguishable from the parameters in the real protocol.

5.2.5 Prove that $\overline{r}_0$ is calculated by on-chain ciphertext. When a receiver finds a malicious transaction, receiver can use the private key as witness to prove the wrong randomness are indeed solved by the on-chain ciphertext, and send it to smart contract, the relationship to prove is

$$S_{enc} = \left\{ (sk_r, \overline{r}_0) : C_4 = E_{r3}^{sk_r^{-1}} \overline{r}_0 \right\}$$

A non-interactive Sigma protocol is as below:
(1) P selects a randomness a, and calculates $A_1 = E_{r3}^a, A_2 = pk_r^a$
(2) P computes random challenge $e = Hash(E_{r3}, C_4, A_1, A_2)$
(3) P calculates $s = a + e \cdot sk^{-1}$, and sends A_1, A_2, s to V
(4) V calculates

$$E_{r3}^s = A_1(C_4/\overline{r}_0)^e$$

$$pk_r^s = A_2 \cdot g^e$$

If the two equality verifications are both true, then output the evidence π_4. After the smart contract verified and confirmed that this is a malicious transaction, it can be homomorphically calculated again to subtract the transaction amount and at the same time punish the malicious sender.

Security Proof of Sigma protocol

Completeness: From the above process, the correctness is clear if the P and V are executed as specified in the protocol.

Special Soundness: For certain A_1, suppose there are two different accepting transcripts (e, s) and (e', s'), $e \neq e'$, then sk can be extracted by the following method. We have $s = a + e \cdot sk, s' = a + e' \cdot sk$, which can get $sk = (s - s') \,/\, (e - e')$.

Special Honest-Verifier Zero-Knowledge: Assuming there is a polynomial-time simulator S, where the simulator picks a random challenge e and response z, and computes $A_1 = E_{r_3}^s(g^{r_0})^{-e}$, it is clear that (A_1, E_{r_3}, s) is a valid transcript , and for any probabilistic polynomial-time verifier , these parameters are computationally indistinguishable from the parameters in the real protocol.

6 Scheme Based On Modified Paillier

6.1 Handling Large Transaction Amounts

The feasibility of the above scheme is based on the fact that the transaction amount m is small (less than 2^{32}), because when m is relatively large, the difficulty of calculating m from h^m will increase greatly, affecting the immediacy of a transaction. Taking Ethereum as an example, the smallest transaction unit is $wei(1Ether = 10^{18}wei)$. If large transactions generated with this precision are calculated by brute-force enumeration, it will be slower. So we need a method that can quickly and directly decrypt the transaction amount m. From the perspective of regulation, the regulator needs to know the specific amount and destination of each transaction. The regulator needs to access the total amount of transactions in an account over a period to monitor criminal acts such as money laundering. But the method based on zero-knowledge proof unable to get the specific amount, Zcash chooses a new keypair which they can provide to a third party. It works but increases the complexity of transaction system and on-chain content, which also brings additional troubles to regulation and audit of transactions.

To solve the above problems, we propose a confidential transaction system based on Paillier [Pai99] encryption, which can efficiently calculate the transaction amount m even the transaction amount is relatively large (greater than 2^{32}). Also, the transaction process does not require the regulator to participate. The regulator can operate independently and only participate in the transaction when regulation is needed. So that the transaction is regulated while privacy is protected and controllable privacy is realized.

6.2 Audit and Regulation of Transactions

With the rapid development of cryptocurrencies, different suggestions have been put forward on how to regulate them. A basic idea is that the system should verify the legitimacy of participating entities, that is, users who have completed authentication can proceed with subsequent transactions. Narula et al. [NVV18] and Tian et al. [TCD$^+$19] proposed an alternative approach to digital currency regulation that would require changing the structure of ledger. At present, some confidential transaction schemes provide too strong anonymity and privacy, which might be abused in some cases. For example, Pedersen commitment are used to hide transaction amount, and regulators cannot obtain the specific transaction information of users in the blockchain network. If users engage in transactions with high frequency and large amount, such as money laundering, the regulator will not get any relevant information, which will lead to some illegal behaviors that are difficult to restrain. Therefore, it is an important challenge to realize controllable privacy and give the regulator higher authority while protecting users' transaction privacy.

According to our investigation, the US Securities and Exchange Commission (SEC), the US Federal Bureau of Investigation (FBI), the US Financial

Consumer Protection Bureau (CFPB) and other law enforcement agencies have taken regulatory actions against financial activities on the blockchain, involving anti-money laundering, tax evasion and other issues. Especially in the rapid development of Decentralized Finance (Defi) in recent years, the need to strengthen regulation is more urgent. In order to design a practical and efficient regulation scheme, we choose the idea of verifying the legitimacy of the transaction participant, and expect to realize the regulatory confidential transaction system at the minimum cost. Our proposal satisfies the following requirements:

(1) Every user in the system is under regulation, that is, regulation is not an option for users.

(2) The activities of the regulator and the transactions between users are independent of each other, that is, the implementation of regulation and audit does not require users to be online, and users do not have to go through the regulator when conducting transactions.

(3) Make the minimum change to the existing user account structure. As far as users are concerned, there is no difference between the new regulatory scheme and the existing scheme.

The following are detailed introductions from cryptographic algorithms to the construction of the regulated confidential transaction system.

6.3 Confidential Transaction System Based on Modified Paillier

6.3.1 Modified Paillier

· *Setup* : Generate two large prime p, q, where $p = 2p' + 1, q = 2q' + 1, p \neq q$ and p', q' are primes. Set $\lambda' = p'q'$, now the order of Z_n^* is $\psi(n) = 4\lambda'$. Z_n^* is consisting of Z_p^* and Z_q^*, Z_n^* can be calculated by Chinese Remainder Theorem(CRT). Then randomly select $g_{p'} \in \mathbb{G}_{p'}, g_{q'} \in \mathbb{G}_{q'}$, and $\mathbb{G}_{p'}, \mathbb{G}_{q'}$ are the subgroup of Z_p^* and Z_q^* with order p' and q'. We can compute the generator $g_1 \in Z_{n^2}^*$ of cyclic group $\mathbb{G}_{p'q'}$ by Chinese Remainder Theorem, $g_1 = g_{p'} \bmod p, g_1 = g_{q'} \bmod q$, the order of $\mathbb{G}_{p'q'}$ is $p'q'$, and g_1 satisfy $gcd(L(g_1^u \bmod n^2), n) == 1$ simultaneously. Compute $u = lcm(p-1, q-1)$ and $k = g_1^u \bmod n^2$, select randomness $r \in Z_{n^2}^*$, compute $h = g_1^r \bmod n^2$, the public key is $pk = h^{sk}$. Now the public key of the homomorphic algorithm is pk, the private key is sk, the system parameter (k, h, n) is public, and u is the system private key.

· $Enc(m, r_0, r_1)$: For message m, $m \in Z_n$, select random number $r_0, r_1 < n$, and calculate $C_1 = pk^{r_0} \bmod n^2, C_2 = k^m h^{r_0} \bmod n^2, C_3 = pk^{r_1} \bmod n^2, C_4 = k^{r_0} h^{r_1} \bmod n^2$. The ciphertext is (C_1, C_2, C_3, C_4), and C_2, C_4 are in the form of FO commitment.

· $Dec(C_1, C_2, C_3, C_4, sk)$: Compute $C_m = C_2/C_1^{sk^{-1}} = k^m \bmod n^2$, $m = L(C_m \bmod n^2)/L(k \bmod n^2)$ to recover m, and compute $C_{r_0} = C_4/C_3^{sk^{-1}} = k^{r_0} \bmod n^2, r_0 = L(C_{r_0} \bmod n^2)/L(k \bmod n^2)$ to recover r_0

Obviously, the new algorithm satisfies correctness and homomorphism, and at the same time, it satisfies IND-CPA security based on DDH assumption in the standard model. The specific proof is given in Appendix B.

6.3.2 Combine FO Commitment and Bulletproofs

Unlike the Modified ElGamal is the form of the Pedersen commitment, which can directly use Bulletproofs. The ciphertext obtained by Modified Paillier encryption is the form of FO commitment, which requires the extra proof that the Pedersen commitment contains the same (m, r) with FO commitment, and then use Bulletproofs to Pedersen commitment. This requires a new Sigma protocol that differs from above and similar to Zether. The relations need to prove as below:

(1) Transaction amount m is non-negative and within the correct range (less than 2^{64})

(2) The sender's balance is non-negative

Sender does not prove the correctness of the ciphertext, but receiver verify. If the receiver receives a malicious transaction, the transaction can be reported. According to the idea of Game Theory, the sender will actively eliminate evil thoughts. So all the sender needs to do is recording the ciphertext of the transaction $(C_1 = pk^{r_0} \bmod n^2, C_2 = k^m h^{r_0} \bmod n^2, C_3 = pk^{r_1} \bmod n^2, C_4 = k^{r_0} h^{r_1} \bmod n^2)$ and aggregate range proofs evidence on the blockchain, greatly reducing the data amount.

6.4 Construction of Transaction System

The transaction system is similar to the scheme using Modified Elgamal. The main difference lies in the way of dealing with malicious transactions, because the scheme based on Modified Paillier can accurately calculate the transaction amount m, while the algorithm based on Modified Elgamal have to calculate m from h^m by brute-force enumeration, and the wrong h^m may not be able to calculate m. Therefore, if the amount calculated is different from the commitment amount, the malicious transaction can be reported by the calculation evidence. The specific transaction process is as follows.

***Setup*$(\mathbf{1}^{\boldsymbol{\lambda}})$** : Input a security parameter λ to generate relevant parameters for encryption and zero-knowledge proof.

***CreateAddress*$(\mathbf{1}^{\boldsymbol{\lambda}})$** : Input a security parameter λ, and execute PKE.KeyGen(pp) to get a keypair(pk, sk). then generate the account according to encryption designed in this paper, calculate $C_0 = Enc(PK, m_0, r_0)$ as the initial balance of the account, where $m_0 = 0$. It then outputs (pk, sk) and uses public key as the address for subsequent transactions.

$\boldsymbol{Transact(sk_s, pk_s, pk_r, m)}$:On input sender's keypair (pk_s, sk_s) and receiver's address pk_r, suppose sender transfer m to receiver. And $E_s^* = (pk_s^{r^*}, k^{m^*} h^{r^*})$ represent sender's current balance, the specific transaction process is as follows:

Sender:Sender first checks whether $m \in [0, 2^n - 1]$ and $m^* \in [0, 2^n - 1]$, and encrypts m with pk_s and pk_r respectively to get $E_s = (C_1 = pk_s^{r_0}, C_2 =$

$k^m h^{r_0}), E_r = (C_1 = pk_r^{r_0}, C_2 = k^m h^{r_0}, C_3 = pk_r^{r_1}, C_4 = k^{r_0} h^{r_1})$, the ciphertext of the transaction has five parts. The ciphertext of the balance consists of two parts: $E'_s = (pk_s^{r^*-r_0}, k^{m^*-m} h^{r^*-r_0} = k^{m'} h^{r'})$, and r_0 can be calculated. Since the r_0 of each transaction can be solved, it is considered that the randomness in the sender's balance is known and r' can be calculated. The sender is also required to use zero-knowledge proof to prove (1) range proofs for the transaction amount m, the transaction amount is within the specified range, and get π_1 (2) range proofs for sender's current balance, which is a positive value, and get π_2. More formally, a user proves the following statement:

$$S_{range1} = \{(pk_s, E_s) : \exists r_0, r_1, m \ s.t. \ E_s = Enc(pk_s, m, r_0, r_1) \wedge m \in [0, 2^n - 1]\}$$

$$S_{range2} = \{(pk_s, E'_s) : \exists r', m' \ s.t. \ E'_s = Enc(pk_s, m', r') \wedge m' \in [0, 2^n - 1]\}$$

Then run signature algorithm to the transaction with sender's private key. And the final transaction is $tx = (pk_s, pk_r, E_s, E'_s, E_r, \pi_1, \pi_2)$ and corresponding signature Sig. There is no need for the sender to prove the ciphertext is correct, that is, the ciphertext of E_s and E_r is encrypted with the same (m, r_0) with the public keys of both parties. Instead, it is the receiver to verify the correctness of ciphertext. If it is a malicious transaction, the receiver calls the smart contract to punish the sender, which make the sender give up the will to construct illegal transaction.

$\boldsymbol{VerifyTX(tx, sig)}$:Verify the validity of Sig with the sender's public key, verify $E'_s = E^*_s/E_s$ and π_1, π_2. If all the verifications pass, miners confirm that transaction is valid and record it on the blockchain via consensus protocol.

$\boldsymbol{ConfirmTX(tx)}$:After the receiver obtains the on-chain transaction information, verify $E'_s = E^*_s/E_s$ and validity of π_1, π_2. Then decrypt $E_r = (C_1 = pk_r^{r_0}, C_2 = k^m h^{r_0}, C_3 = pk_r^{r_1}, C_4 = k^{r_0} h^{r_1})$ to get $k_0^{\overline{r}} = C_4/C_3^{sk^{-1}}, k^{\overline{m}} = C_2/C_1^{sk^{-1}}$, receiver checks if $k^{\overline{m}} h^{\overline{r}_0} = k^m h^{r_0}$. If the verification pass, tx is a valid transaction, and the reveiver updates corresponding balance and randomness. If not, then this is a malicious transaction, indicating that the sender changes the randomness r_0 in C_1 so that the receiver cannot solve the correct transaction amount, or the sender changes the randomness r_0 in C_4 so that the receiver cannot solve the correct randomness. The receiver also need to determine whether $pk^{\overline{r}_0}$ is the same with the on-chain ciphertext. During the challenge stage, the receiver can prove that the transaction is malicious with proof of fraud, and the honest receiver will execute the $reportTX(TX)$. If the receiver does not report the transaction until the end of the challenge stage, the transaction is considered valid.

When the sender and the receiver are both malicious users, that the receiver does not report after receiving malicious transactions (under normal circumstances, the receiver program will automatically call $ReportTX(tx)$ after calculating the malicious transaction). However, the updated balance of the receiver is the true amount corresponding to the on-chain commitment, not the wrong amount $\overline{m}$(which may be greater than m), so the receiver cannot obtain the amount greater than m.

Report(tx) When the receiver calculates that the transaction is malicious, he reports the transaction to the smart contract. Record the wrong $\overline{m}, \overline{r}_0$ on the blockchain and prove that $\overline{m}, \overline{r}_0$ is actually calculate from the on-chain ciphertext. More formally, the receiver proves the following statement:

$$S_{enc} = \left\{ \left(sk_r, k^{\overline{m}}, k_0^{\overline{r}} \right) : \exists k^{\overline{m}}, k_0^{\overline{r}} \ s.t. \ C_2 = E_{r1}^{sk_r^{-1}} k^{\overline{m}} \wedge C_4 = E_{r3}^{sk_r^{-1}} k_0^{\overline{r}} \right\}$$

and generate a zero-knowledge proof π_3. After the smart contract verification, it is confirmed that this is a malicious transaction, and then it performs a homomorphism calculation on the receiver's account $E_s^* = E_s' \cdot E_s$, returns to the state before the malicious transaction is completed, and destroys the token corresponding to this transaction. Because normal user only needs input transaction amount m when performing confidential transactions. the reason for the above malicious transaction is that the attacker changed the randomness in C_1 or C_4 to make it different from the randomness in $k^m h^{r_0}$, it can be considered that this kind of transaction must be maliciously constructed by the sender, so the smart contract can destroy the token in the transaction to punish the malicious sender.

ReadBalance(E, sk) Taking the sender as an example, input the private key sk_s of the sender and the corresponding ciphertext E_s to obtain the balance $m = Dec\left(E_s, sk_s\right)$ of the sender user.

Above all, attack cannot succeed in this process, from the perspective of Game Theory, an adversary will not execute an attack that is unprofitable or even at a loss, and has no effect on the honest receiver, so we can assume that malicious transactions won't appear and the system can operate safely.

6.5 Construction of Regulatory System

The transaction procedure is the same as the scheme of Modified ElGamal. The regulator can calculate user's private key sk through the system private key u when regulation and audit are needed. There are two advantages of this method: (1) It is not necessary to save the user's private key, but to calculate user's private key when it is necessary to regulate or audit a user. (2) There is no interaction between the regulator and the user, and the operation of regulation and audit can be completed independently. Compared with the scheme of encrypting the user's private key with the public key of the regulator, this scheme does not need to save user's private key in the database, does not need to transfer the private key, and saves the trouble of keeping user's private key. The system private key can be saved with multiple signatures to ensure that regulators can not do evil at will. The algorithms involved in regulation are as follows:

$Setup\left(\mathbf{1}^{\lambda}\right)$: Input a security parameter λ to generate relevant parameters used by the system, including system private key u, system parameter (k, h, n), etc

$GetSysSk\left(msg, sig_1, sig_2, sig_3\right)$: When the message using the system private key is received, the system will verify the validity of the message, which requires 3 regulators' signature (to simplify, the regulator set to 3, that is, this is a 3/3 multi-signature scheme), after passing the verification, return the system private key u and change the state of the system private key to TRUE.

$GetUserSk\left(pk, u\right)$: First, determine the user to be regulated or audited and calculate the corresponding private key according to the public key. The algorithm is $sk = L\left(pk^{u} \bmod n^2\right)/L\left(h^{u} \bmod n^2\right)$. And put the message that the user's private key obtained by the regulator on the blockchain, and then the private key can be used to verify the validity of each transaction. The state of the system private key is changed to FALSE after use.

$GetAmount(Tid, pk, tx, sk)$: After calculating the private key of the user, regulator can obtain m_i of a specific transaction according to Tid of a transaction, or sum of the transaction amount within a certain period. And then record relevant information.

$AuditTx\left(pk, m, sum\right)$: Audit the information obtained and the total transactions of the user during a certain period. Use relevant audit tools such as range proof etc. If the audit result is TRUE, the user is honest; FALSE indicates that the user committed some illegal acts.

6.6 Zero knowledge Proof

6.6.1 Aggregating Logarithmic Proofs. According to additive homomorphic of Modified Paillier, sender's new balance is $C_2' = C_2^*/C_2 = k^{m^*-m}h^{r^*-r} = k^{m'}h^{r'}$. Because the $k^{m'}h^{r'}$ is FO commitment, we need to prove that the $k^{m'}h^{r'}$ contains the same (m', r') as Pedersen commitment, and then use Bulletproofs for the Pedersen commitment. Moreover, (m', r') is computable, and (m, r) is the amount and randomness of the transaction, so the aggregate range proof can directly use $(m', r'), (m, r)$ as witness. The relationship to be proved consists of two parts, (1)using Bulletproofs to prove m' and m is non-negative and within the correct range (2)proving that the balance m, m' in FO commitment are equal to m, m' in Bulletproofs, and we generalize the protocol by simply requiring that the prover proves that $\hat{t} = \sum_{i=1}^{m} v_i \cdot z^i + \delta(y, z) + Open(T)$. The relationship in (2) can be written as:

$$\{(C_2, C_2') : \exists\ m, r_0, m', r_0'\ s.t. C_2 = k^m h^{r_0} \wedge C_2' = k^{m'} h^{r_0'} \wedge$$

$$\mathrm{g}_1^{\hat{t}-\delta(y,z)-m\cdot z^2-m'\cdot z^3} h_1^{\tau-r_0 z^2-r_0' z^3} = T_{1,2}\},\ T_{1,2} = T_1^x T_2^{x^2}$$

A non-interactive Sigma protocol is as below:

(1) P selects a random number a, b, and calculates $A_1 = k^a h^b \bmod n^2, A_2 = g_1^{-a} h_1^{-b} \bmod p$.

(2) P computes random challenge $e = Hash(C_s', C', A_1, A_2)$.

(3) P calculates $s_1 = a + e(mz^2 + m'z^3),\ s_2 = b + e(r_0z^2 + r_0'z^3)$, and sends A_1, A_2, s_1, s_2 to V.

(4) V calculates

$$k^{s_1}h^{s_2} = A_1(C_2)^{ez^2}\,(C_2')^{ez^3}$$

$$g_1^{\left(\hat{t}-\ \delta(y,z)\right)e-s_1}h_1^{\tau e-s_2} = A_2T_{1,2}^{e}$$

Security Proof of Sigma protocol

Completeness: From the above process, the correctness is clear if the P and V are executed as specified in the protocol.

Special Soundness: For certain (A_1, A_2), suppose there are two different accepting transcripts $(e, s = (s_1, s_2))$ and $(e', s' = (s_1', s_2'))$, $e \neq e'$, then m can be extracted by the following method. We have $s_1 = a + e(mz^2 + m'z^3), s_1' = a + e'(mz^2 + m'z^3)$, which can imply $mz^2 + m'z^3 = (s_1 - s_1')/(e - e')$. In order to extract m' and m we need to rewind the whole Sigma protocol twice, and use the same extraction procedure for the Sigma protocol we get the extracted m, m'. Now we form the equations $M_1 = mz_1^2 + m'z_1^3, M_2 = mz_2^2 + m'z_2^3$, and then extract m' and m. And we can extract r_0, r_0' in the same way.

Special Honest-Verifier Zero-Knowledge: Assuming there is a polynomial-time simulator S, where the simulator picks a random challenge e and response (s_1, s_2), and computes $A_1 = k^{s_1}h^{s_2} \cdot C_2^{-ez^2}\,(C_2')^{-ez^3}$, $A_2 = g_1^{\left(\hat{t}-\ \delta(y,z)\right)e-s_1}h_1^{\tau e-s_2} \cdot T_{1,2}^{-e}$, it is clear that A_1, A_2, e, s_1, s_2 is a validate transcript, and for any probabilistic polynomial-time verifier, these parameters are computationally indistinguishable from the parameters in the real protocol.

6.6.2 Prove that $(\overline{m}, \overline{r}_0)$ is calculated by on-chain ciphertext.

When a receiver finds a malicious transaction, the receiver can use the private key as witness to prove the wrong transaction amount and randomness are indeed solved by the on-chain ciphertext, and send it to smart contract, the relationship to prove is

$$S_{enc} = \left\{\left(sk_r, k^{\overline{m}}, k_0^{\overline{r}}\right) : C_2 = E_{r1}^{sk_r^{-1}}\,k^{\overline{m}} \wedge C_4 = E_{r3}^{sk_r^{-1}}\,k_0^{\overline{r}}\right\}$$

A non-interactive Sigma protocol is as below:

(1) P selects a random number a, and calculates $A_1 = E_{r1}^{a}, A_2 = E_{r3}^{a}, A_3 = pk^a$

(2) P computes random challenge $e = Hash(E_{r1}, C_2, E_{r3}, C_4, A_1, A_2, A_3)$

(3) P calculates $s = a + e \cdot sk^{-1}$, and sends A_1, A_2, A_3, s to V

(4) V calculates

$$E_{r1}^{s} = A_1(C_2/k^{\overline{m}})^{e}$$

$$E_{r3}^{s} = A_2(C_4/k^{\overline{r}_0})^e$$

$$pk_r^s = A_3 \cdot h^e$$

If the two equality verifications are both true, then output the evidence π_4. After the smart contract verified and confirmed, it can be homomorphically calculated again to subtract the transaction amount, and at the same time punish the malicious sender.

Security Proof of Sigma protocol

Completeness: From the above process, the correctness is clear if the P and V are executed as specified in the protocol.

Special Soundness: For certain (A_1, A_2), suppose there are two different accepting transcripts (e, s) and (e', s'), $e \neq e'$, then sk can be extracted by the following method. We have $s = a + e \cdot sk, s' = a + e' \cdot sk$, which can imply $sk = (s - s') \,/\, (e - e')$.

Special Honest-Verifier Zero-Knowledge: Assuming there is a polynomial-time simulator S, where the simulator picks a random challenge e and response s, and computes $A_1 = E_{r1}^{s}(h^{r_0})^{-e}, A_2 = E_{r3}^{s}(h^{r_1})^{-e}$, it is clear that (A_1, A_2, e, s) is a validate transcript , and for any probabilistic polynomial-time verifier , these parameters are computationally indistinguishable from the parameters in the real protocol.

6.7 Security Analysis

The on-chain content has passed the range proof and legality verification, so it can be considered that the transaction data obtained by the regulator from the blockchain is correct. If the ciphertext with error exists on the blockchain, that is, there is an wrong but verified transaction. The correctness and soundness of the transaction system ensures that the probability of such a transaction is negligible. And the ciphertext is the format of FO commitment, which has global homomorphism. The regulator can first analyze the total transaction amount of an address in a period, and if there is a problem, analyze the specific transaction amount. According to the correctness of the transaction system and the correctness of homomorphic encryption, it can be inferred that the scheme is auditable and meets audit reliability.

6.8 Application Scenario Analysis

The above confidential transaction scheme based on Modified ElGamal can be directly used in most public blockchain systems with advantages of encryption and decryption speed. Although the calculation speed is fast, in terms of its regulability, it is not as effective as the scheme based on Modified Paillier. The Modified Paillier scheme is more suitable for systems with large transaction amount or some consortium blockchain systems. It gives full play to the advantages of high trading efficiency of consortium blockchain, hands the system private key to several trusted parties of consortium blockchain, and prevents the abuse of system private key by multi-signature, and more effective in regulation. Users can choose the application scheme according to their own needs.

7 Performance

7.1 Benchmark of the Scheme

We have given the prototype implementation of the scheme. We have implemented a standalone cryptocurrency in C++. To evaluate the specific performance of our project in communication and computational costs. Recall that the confidential transaction we designed consists of the following aspects: (1) the transaction information *tx* (2) the sender's signature *sig* of the transaction (3) and the evidence of the aggregate Bulletproofs. For the modified ElGamal, we implemented the code based on OpenSSL and GMP, and selected elliptic curve prime256v1 [ope] at 128-bit security level, in which each element in $\mathbb{G}$ requires 33Byte (32Byte for the x-coordinate and 1 bit for the sign), and each element in $\mathbb{Z}_p$ requires 32Byte. For the Modified Paillier, we implemented the code based on OpenSSL with 1536-bit security level, in which each element in $\mathbb{G}_{p'q'}$ requires 384Byte, and each element in $\mathbb{Z}_n$ requires 192Byte. We implemented it on the AMD Ryzen 3700X 3.59GHz CPU, and the specific results are as follows.

7.2 Communication and Computational Costs

Scheme Based on Modified ElGamal. The size of a confidential transaction($\mathcal{O}$) is $(2log_2(2l)+16)\,\mathbb{G}+8\mathbb{Z}_p$, where $l=32$. It includes 9 elements in $\mathbb{G}$ for transaction information, 1 element in $\mathbb{G}$ for digital signature, aggregation range proof $(2log_2(2l)+4)\,\mathbb{G}$ and 5 elements in $\mathbb{Z}_p$, and $2\mathbb{G}+3\mathbb{Z}_p$ elements for validity proof.

Scheme Based on Modified Paillier. The size of a confidential transaction($\mathcal{O}$) is $(2log_2(2l)+5)\,\mathbb{G}+11\mathbb{G}_{p'q'}+5\mathbb{Z}_p+2\mathbb{Z}_n$, where $l=64$. It includes 9 elements in $\mathbb{G}_{p'q'}$ for transaction information, 1 element in $\mathbb{G}_{p'q'}$ for digital signature, aggregation range proof need $(2log_2(2l)+5)\,\mathbb{G}+\mathbb{G}_{p'q'}+5\mathbb{Z}_p+2\mathbb{Z}_n$ elements.

Table 1. The computation and communication complexity of the transaction

	transaction size		transaction cost(ms)	
	big-$\mathcal{O}$	byte	generation	verify
Modified ElGamal	$(2log_2(2l)+16)\,\mathbb{G}+8\mathbb{Z}_p$	1180	230.7	60.2
Modified Paillier	$(2log_2(2l)+5)\,\mathbb{G}+11\mathbb{G}_{p'q'}+5\mathbb{Z}_p+2\mathbb{Z}_n$	5329	607.3	341.2
PGC	$(2log_2(2l)+20)\,\mathbb{G}+10\mathbb{Z}_p$	1376	40	14

Table 2. The computation and communication complexity of reporting

Report	big-$\mathcal{O}$	byte	time cost(ms)
Modified ElGamal	$2\mathbb{G} + 2\mathbb{Z}_p$	130	0.5
Modified Paillier	$5\mathbb{G}_{p'q'} + \mathbb{Z}_n$	2112	202.3

References

BAZB20. Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security*, pages 423–443, Cham, 2020. Springer International Publishing.

BBB$^+$18. 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, 2018.

BCC$^+$16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 327–357, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

BDN18. Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 435–464, Cham, 2018. Springer International Publishing.

CMTA20. Yu Chen, Xuecheng Ma, Cong Tang, and Man Ho Au. Pgc: Decentralized confidential payment system with auditability. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve Schneider, editors, *Computer Security – ESORICS 2020*, pages 591–610, Cham, 2020. Springer International Publishing.

Dam02. Ivan Damgård. On σ-protocols. *Lecture Notes, University of Aarhus, Department for Computer Science*, 2002.

DSPSNAHJ18. Sergi Delgado-Segura, Cristina Pérez-Sola, Guillermo Navarro-Arribas, and Jordi Herrera-Joancomartí. Analysis of the bitcoin utxo set. In *International Conference on Financial Cryptography and Data Security*, pages 78–91. Springer, 2018.

Fin18. Michèle Finck. *Blockchain regulation and governance in Europe*. Cambridge University Press, 2018.

FMMO19. Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 649–678, Cham, 2019. Springer International Publishing.

FO97. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, pages 16–30, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

FOS19. Georg Fuchsbauer, Michele Orrù, and Yannick Seurin. Aggregate cash systems: A cryptographic investigation of mimblewimble. In Yuval

Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 657–689, Cham, 2019. Springer International Publishing.

FS86. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

Gri. Grin. `https://grin-tech.org/`.

Kur02. Kaoru Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In *International Workshop on Public Key Cryptography*, pages 48–63. Springer, 2002.

Max13. Gregory Maxwell. Coinjoin. In Post on Bitcoin forum, 2013.

Max15. Gregory Maxwell. Confidential transactions. `https://www.weusecoins.com/confidential-transactions/`, 2015.

MP15. Gregory Maxwell and Andrew Poelstra. Borromean ring signatures. *Accessed: Jun*, 8:2019, 2015.

NM$^+$16. Shen Noether, Adam Mackenzie, et al. Ring confidential transactions. *Ledger*, 1:1–18, 2016.

NVV18. Neha Narula, Willy Vasquez, and Madars Virza. zkledger: Privacy-preserving auditing for distributed ledgers. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 65–80, Renton, WA, April 2018. USENIX Association.

ope. openssl. `https://github.com/openssl/openssl/tree/4e6647506331fc3b3ef5b23e5dbe188279ddd575`.

Pae17. Philipp Paech. The governance of blockchain financial networks. *The Modern Law Review*, 80(6):1073–1110, 2017.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

Ped92. Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

Poe16. Andrew Poelstra. Mimblewimble. `https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf`, 2016.

PS00. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13:361–396, 2000.

PSST11. Kenneth G Paterson, Jacob CN Schuldt, Martijn Stam, and Susan Thomson. On the joint security of encryption and signature, revisited. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 161–178. Springer, 2011.

Sat08. Nakamoto Satoshi. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008.

SCG$^+$14. Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

Sch91. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

Sha71. Daniel Shanks. Class number, a theory of factorization, and genera. In *Proc. of Symp. Math. Soc., 1971*, volume 20, pages 41–440, 1971.

TCD+19. Haibo Tian, Xiaofeng Chen, Yong Ding, Xiaoyan Zhu, and Fangguo Zhang. Afcoin: A framework for digital fiat currency of central banks based on account model. In Fuchun Guo, Xinyi Huang, and Moti Yung, editors, *Information Security and Cryptology*, pages 70–85, Cham, 2019. Springer International Publishing.

Woo14. Gavin Wood. Ethereum: A secure decentralized transaction ledger. `https://ethereum.github.io/yellowpaper/paper.pdf`, 2014.

Zca. Zcash. `https://z.cash/wp-content/uploads/2019/09/Zcash-Regulatory-Brief-201909.pdf`.

zks. zksnarks. `https://z.cash/technology/zksnarks/`.

A Appendix

A.1 Security Proof of Modified ElGamal

Theorem 5. *The modified ElGamal encryption scheme is IND-CPA secure based on the DDH assumption.*

Game 0. In the real IND-CPA security experiment, the interaction between challenger $\mathcal{CH}$ and adversary $\mathcal{A}$ is as blow. Let S_i be the probability that $\mathcal{A}$ wins in Game i.

1. Setup. $\mathcal{CH}$ generate system and related parameters, sends public keys $pk = g^{sk}$ to $\mathcal{A}$.

2. Training. $\mathcal{A}$ generates messages to obtain encrypted ciphertext (bounded polynomial times).

3. Challenges. $\mathcal{A}$ outputs two messages of equal length m_0 and m_1. $\mathcal{CH}$ selects random bit β and randomness r_0, r_1, calculate $C_1 = pk^{r_0}, C_2 = g^{r_0} h^{m_\beta}, C_3 = pk^{r_1}, C_4 = r_0 g^{r_1}$, and send C_1, C_2, C_3, C_4 to $\mathcal{A}$. Now DH quad $\left(g, g^a, g^b, g^{ab}\right)$ corresponding to $(g, g^{r_0}, g^{sk}, g^{r_0 \cdot sk}), (g, g^{r_1}, g^{sk}, g^{r_1 \cdot sk})$.

4. Guess. $\mathcal{A}$ outputs β', and wins if $\beta' = \beta$.

The adversary's advantage in Game 0 can be defined as below.

$$Adv_{\mathcal{A}}(\lambda) = \Pr[S_0] - 1/2$$

Game 1. The same as Game 0, except that $\mathcal{CH}$ picks a random bit β and randomness r_0, r_1, s_0, s_1, compute $C_1 = pk^{r_0}, C_2 = g^{s_0} h^{m_\beta}, C_3 = pk^{r_1}, C_4 = r_0 g^{s_1}$ and send C_1, C_2, C_3, C_4 to $\mathcal{A}$.

In Game 1, ciphertext distribution is independent of β, so $\mathcal{A}$ has no message about β, $\Pr[S_1] = 1/2$. $(g, g^{s_0}, g^{sk}, g^{sk \cdot r_0}), (g, g^{s_1}, g^{sk}, g^{sk \cdot r_1})$ constitute a random quad. Assume that $c = c'b, a = c' + c''$, ciphertext can be represented as $(g^{c'_0 b}, g^{c'_0}(g^{c''_0} h^m), g^{c'_1 b}, g^{c'_1}(g^{c''_1} a_0))$ and $(g, g^{c'_0 + c''_0}, g^b, g^{c'_0 b}), (g, g^{c'_1 + c''_1}, g^b, g^{c'_1 b})$ constitute a random quad.

Next, it is proved that the difference between $Pr[S_0]$ and $Pr[S_1]$ is negligible. We construct adversary $\mathcal{B}$ with the same advantage as $\mathcal{A}$ to attack DDH assumption. Given a quad $\left(g, g^a, g^b, g^c\right)$, $\mathcal{B}$ determines whether it is a random quad or a DH quad. $\mathcal{B}$ is constructed as follows.

1. Setup. $\mathcal{B}$ generates system and related parameters, treats g^b as the public keys pk, b is the corresponding private key, which is unknown to $\mathcal{B}$. Then $\mathcal{B}$ sends $pk = g^b$ to $\mathcal{A}$.

2. Training. $\mathcal{A}$ generates messages to obtain encrypted ciphertext (bounded polynomial times).

3. Challenges. $\mathcal{A}$ outputs two messages of equal length m_0, m_1 and sends them to $\mathcal{B}$. $\mathcal{B}$ selects random bits β, calculate $C_1 = g^{a_0}, C_2 = g^{c_0} h^{m_\beta}, C_3 = g^{a_1}, C_4 = a_0 g^{c_1}$, sends C_1, C_2, C_3, C_4 to $\mathcal{A}$.

4. Guess. $\mathcal{A}$ outputs β', and wins if $\beta' = \beta$.

If the quad $\left(g, g^a, g^b, g^c\right)$ is a DH quad, that is, $(g, g^{r_0}, g^{sk}, g^{r_0 \cdot sk})$, $(g, g^{r_1}, g^{sk}, g^{r_1 \cdot sk})$ consist of a DH quad, then $\mathcal{B}$ is the same view as Game 0, where $a = bc$. If $\left(g, g^a, g^b, g^c\right)$ is a random quad, that is, $(g, g^{c'_0 + c''_0}, g^b, g^{c'_0 b})$, $(g, g^{c'_1 + c''_1}, g^b, g^{c'_1 b})$ consist of a random quad, then $\mathcal{B}$ is the same view as Game 1. Therefore, if $\mathcal{A}$ can distinguish between $\mathcal{B}$ representing Game 0 and Game 1 with non-negligible advantage, then $\mathcal{B}$ can break the DDH assumption with the same advantage.

A.2 Private key cannot be obtained from public parameters.

Theorem 6. *It is known that $pk = g^{sk} \bmod p$, the public key is pk, g, p, and the private key is sk. Computing the private key sk from the public key pk, g, p belongs to the discrete logarithm problem on the group, where g is the generator of the group, and pk is the element on the group. It is difficult to calculate $\log_g pk$. So it can be concluded that sk cannot be obtained from the public keys pk, g, p.*

B Appendix

B.1 Security Proof of Modified Paillier

Theorem 7. *The modified Paillier encryption scheme is IND-CPA secure based on the DDH assumption.*

Game 0. In the real IND-CPA security experiment, the interaction between challenger $\mathcal{CH}$ and adversary $\mathcal{A}$ is as blow. Let S_i be the probability that $\mathcal{A}$ wins in Game i.

1. Setup. $\mathcal{CH}$ generates system and related parameters, sends public keys $pk = g^{sk}$ to $\mathcal{A}$.

2. Training. $\mathcal{A}$ generates messages to obtain encrypted ciphertext (bounded polynomial times).

3. Challenges. $\mathcal{A}$ outputs two messages of equal length m_0 and m_1. $\mathcal{CH}$ selects random bit β and randomness r_0, r_1, calculates $C_1 = pk^{r_0} \bmod n^2, C_2 = h^{r_0} k^{m_\beta} \bmod n^2, C_3 = pk^{r_1} \bmod n^2, C_4 = h^{r_1} k^{r_0} \bmod n^2$, and sends C_1, C_2, C_3, C_4 to $\mathcal{A}$

4. Guess. $\mathcal{A}$ outputs β', and wins if $\beta' = \beta$.

The adversary's advantage in Game 0 can be defined as below.

$$Adv_{\mathcal{A}}(\lambda) = \Pr[S_0] - 1/2$$

Game 1. The same as Game 0, except that CH picks a random bit β and randomness r_0, r_1, s_0, s_1, compute $C_1 = pk^{r_0} \bmod n^2, C_2 = h^{s_0} k^{m_\beta} \bmod n^2, C_3 = pk^{r_1} \bmod n^2, C_4 = h^{s_1} k^{r_0} \bmod n^2$ and send C_1, C_2, C_3, C_4 to $\mathcal{A}$.

In Game 1, ciphertext distribution is independent of β, so $\mathcal{A}$ has no message about β, $\Pr[S_1] = 1/2$. $(h, h^{s_0}, h^{sk}, h^{sk \cdot r_0}), (h, h^{s_1}, h^{sk}, h^{sk \cdot r_1})$ constitute a random quad. Assume that $c = c'b, a = c' + c''$, ciphertext can be represented as $(h^{c_0' b}, h^{c_0'}(h^{c_0''} k^m), h^{c_1' b}, h^{c_1'}(h^{c_1''} k^{a_0}))$ and $(h, h^{c_0' + c_0''}, h^b, h^{c_0' b}), (h, h^{c_1' + c_1''}, h^b, h^{c_1' b})$ constitute a random quad.

Next, it is proved that the difference between $Pr[S_0]$ and $Pr[S_1]$ is negligible. We construct adversary $\mathcal{B}$ with the same advantage as $\mathcal{A}$ to attack DDH assumption. Given a quad $\left(h, h^a, h^b, h^c\right)$, $\mathcal{B}$ determines whether it is a random quad or a DH quad. $\mathcal{B}$ is constructed as follows.

1. Setup. $\mathcal{B}$ generates system and related parameters, treats h^b as the public keys pk, b is the corresponding private key, which is unknown to $\mathcal{B}$. Then $\mathcal{B}$ sends $pk = h^b$ to $\mathcal{A}$.

2. Training. $\mathcal{A}$ generates messages to obtain encrypted ciphertext (bounded polynomial times).

3. Challenges. $\mathcal{A}$ outputs two messages of equal length m_0, m_1 and sends them to $\mathcal{B}$. $\mathcal{B}$ selects random bits β, calculate $C_1 = pk^{r_0} = h^{a_0} = h^{b \cdot c_0}, C_2 = h^{c_0} k^{m_\beta}, C_3 = pk^{r_1} = h^{a_1} = h^{b \cdot c_1}, C_4 = h^{c_1} k^{c_0}$, sends C_1, C_2, C_3, C_4 to $\mathcal{A}$.

4. Guess. $\mathcal{A}$ outputs β', and wins if $\beta' = \beta$.

If the quad $\left(h, h^a, h^b, h^c\right)$ is a DH quad, that is, $(h, h^{r_0}, h^{sk}, h^{sk \cdot r_0})$, $(h, h^{r_1}, h^{sk}, h^{sk \cdot r_1})$ consist of a DH quad, then $\mathcal{B}$ is the same view as Game 0, where $c = a \cdot b$. If $\left(h, h^a, h^b, h^c\right)$ is a random quad, that is, $(h, h^{s_0}, h^{sk}, h^{sk \cdot r_0})$, $(h, h^{s_1}, h^{sk}, h^{sk \cdot r_1})$ consist of a random quad, then $\mathcal{B}$ is the same view as Game 1. Therefore, if $\mathcal{A}$ can distinguish between $\mathcal{B}$ representing Game 0 and Game 1 with non-negligible advantage, then $\mathcal{B}$ can break the DDH assumption with the same advantage.

B.2 Private key cannot be obtained from public parameters

Theorem 8. *For $N = p_1^{v_1} \ldots p_m^{v_m} . \lambda(N) = lcm(p_1^{v_1 - 1}(p_1 - 1), \ldots p_m^{v_m - 1}(p_m - 1))$ L is a multiple of $\lambda(N)$. So there is a polynomial time algorithm, when input (N, L), decomposes N with non-negligible probability.*

Set $pk = h^{sk} \bmod n^2$, user's public key is h, n^2, pk, and private key is sk.

To obtained the private key sk from the public key h, n^2, pk, which is a $class[n]$ problem of $pk = 1$, we can say the problem is still unsolvable. The reason for this is that, assuming that from the public key h, n^2, pk gives sk such that $pk = h^{sk} \bmod n^2$ is true, we can also compute x such that $h = pk^x \bmod n^2$. So $sk \cdot x = 1 \bmod \lambda(n^2)$, that $\lambda(n^2) | (sk \cdot x - 1)$, so based on Theorem 7, we can decomposes n^2, which solves the problem of large number decomposition. So the public key h, n^2, pk cannot be used [BAZB20] to obtain [BBB+18] the private key sk.