PGC: Pretty Good Confidential Transaction System with Accountability

Yu Chen * Xuecheng Ma † Cong Tang ‡

Abstract

Due to the public visible nature of blockchain, the seminal cryptocurrencies such as Bitcoin and Ethereum do not provide sufficient level of privacy, i.e., the addresses of sender and receiver and the transfer amount are all stored in plaintexts on the blockchain. As the privacy concerns grow, several newly emerged cryptocurrencies such as Monero and Zcash provide strong privacy guarantees (including anonymity and confidentiality) by leveraging advanced cryptographic techniques.

Despite strong privacy is appealing, it might be overkilled or even could be abused in some cases. In decentralized transaction systems, anonymity seems go against accountability, which is a crucial property for scenarios that require compliance, auditing or dispute resolution mechanism.

In this work, we trade anonymity for accountability. We present a general framework of confidential transaction with accountability from integrated signature and encryption scheme and non-interactive zero-knowledge proof. We then instantiate our framework, yielding a simple and efficient cryptocurrency called PGC, without trusted setup. The core of PGC is a new public-key encryption scheme that we introduce, twisted ElGamal, which is not only as secure as standard exponential ElGamal, but also quite friendly to Sigma protocols and range proofs. This enables us to devise the accompanying zero-knowledge proofs for transaction well-formedness in a modular fashion. Moreover, the keypair of PGC inherited from twisted ElGamal is largely compatible with Bitcoin and Ethereum, thus PGC could be used as a drop-in to provide confidential enforcements with accountability for Bitcoin/Ethereum-like cryptocurrencies.

*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences. School of Cyber Security, University of Chinese Academy of Sciences. Email: yuchen.prc@gmail.com

†State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences. School of Cyber Security, University of Chinese Academy of Sciences. Email: maxuecheng@iie.ac.cn

‡Peking University. Email: lemuria.tc@gmail.com

Contents

1 Introduction

Unlike traditional centralized transaction systems, modern cryptocurrencies such as Bitcoin [Nak08] and Ethereum [Woo14] establish a decentralized peer-to-peer transaction system by maintaining an append-only ledger, known as blockchain. The ledger is globally distributed and synchronized by consensus mechanisms. A crucial property of blockchain-based cryptocurrencies is public verifiability, that is, anyone can verify the validity of all transactions on the ledger. Bitcoin attains this property by simply exposing all details public: the addresses of sender and receiver as well as the transfer amounts. According to [BBB+18], privacy for transactions consists of two aspects: (1) anonymity, hiding the identities of sender and receiver in a transaction and (2) confidentiality, hiding the transfer amount. While Bitcoin-like and Ethereum-like cryptocurrencies provide some weak anonymity through unlinkability of account addresses to real world identities, they lack confidentiality, which is of paramount importance.

1.1 Motivation

Strong privacy is a double-edged sword. While confidentiality is arguably the primary concern of privacy in the context of cryptocurrencies, anonymity might be abused or even prohibitive for applications that require accountability. This is because anonymity somewhat provides plausible deniability [FMMO19]. Consider an employer pay salaries to employees via cryptocurrencies with strong privacy (e.g., Monero [Noe15]). If the employer did not receive the right salary, how can he certify this and demand justice. Vice versa, how can the employer justify that he indeed paid the right salary to the right employee to resolve dispute. The crux is that anonymity allows users to deny their involvements in a given transaction. Similar scenarios are ubiquitous in e-commercial environment. This is not a problem in centralized system in which there exists a trusted third party, such as central bank, Amazon or Alipay, being able to audit every transactions. But, it could cause far-reaching damage to businesses and economics in decentralized systems.

1.2 Our Contributions

Based on the above discussion, we view confidentiality, anonymity, and accountability as *impossible trinity* in security of cryptocurrencies. In this work, we still focus on confidentiality, but trade anonymity for accountability. We summarize our contributions as below.

A framework of CTx with accountability. We address accountability in confidential transaction (CTx) system for the first time. Roughly, accountability stipulates that given a transaction, any involved party is able to disclose its memo information (sender, receiver and transaction value), and make the correctness of opening *publicly auditable*. Particularly, the opening is required to be local, i.e., would not compromise the confidentiality of rest transactions.

We propose a general framework of CTx along with a formal security model. For the sake of simplicity, we take an account-based approach and focus only on the *transaction layer* of cryptocurrencies, and treat the network/consensus-level protocols as black-box and ignore attacks against them. Our framework is built from two building blocks, integrated signature and encryption (ISE) schemes and non-interactive zero-knowledge (NIZK) proofs. ISE plays an important role in our framework. First, it ensures that each account can safely use a single keypair for both encryption and signing. This greatly simplifies the overall design from both conceptual and practical aspects. Second, the encryption component of ISE guarantees that the resulting CTx system is *self-contained*, i.e., as soon as a transaction is on chain, it takes effect immediately — receiver's balance increases with the amount that sender's balance decreases, and the receiver can spend his coin at his command. This is in contrast to those cryptocurrencies requiring *out-of-band* transfer, which thus are not self-contained. NIZK not only makes the validity of confidential transaction be publicly verifiable, but also automatically provides accountability in a zero-knowledge flavor.

PGC: an efficient instantiation. We instantiate our CTx framework by carefully choosing and devising the ISE scheme and NIZK proofs, yielding a simple yet efficient accountable CTx system, which we refer to as PGC (Pretty Good Confidentiality). Notably, PGC does not require trusted setup, and its security is based solely on the widely-used discrete logarithm assumption. PGC also enjoys reasonably

small transaction size and fast transaction generation/verification. To demonstrate the efficiency of PGC, we implement it and present performance benchmarks in Section 7.

1.3 Technical Overview

We give a technical overview of the general CTx framework and its instantiation.

1.3.1 General framework of confidential transaction system

We begin with the choice of cryptographic building blocks.

PKE vs. Commitment. To acquire confidentiality, a common approach is to commit the balance and transfer amount using a global homomorphic commitment scheme (e.g., the Pedersen commitment [Ped91]), then derive the witness or signing key from the openings to prove well-formedness of transaction and authorize transfer. The seminal CTx systems [Max, Poe] exactly follow this approach.

Nevertheless, commitment-based approaches suffer from several drawbacks. First, the resulting CTx systems are not *self-contained*. Due to lack of decryption capability, senders are required to honestly transmit the openings of outgoing commitments (includes randomness and amount) to receivers in an *out-of-band* manner. This will render the system much complicated, as it must be assured that the out-of-band transfer is correct and secure. Second, users must be stateful. Receivers have to keep track of the randomness and amount of each incoming commitment. Otherwise, failure to open a single incoming commitment will render an account totally unusable, due to either incapable of creating the NIZK proofs (lack of witness), or generating the signature (lack of signing key). This will incur extra security burden to the design of wallet (the openings must be kept in a safe and reliable way). Last but not the least, as pin-pointed by Bünz et al. [BBB+18], since Pedersen commitment is only computational binding based on discrete logarithm assumption, an adversary is able to open a given commitment to an arbitrary value when quantum computers are available, which is a serious issue in the context of cryptocurrencies.

Observe that homomorphic PKE can be viewed as a computationally hiding and perfectly binding commitment, while the secret key serves as a natural trapdoor to recover message. This suggests us to overcome the aforementioned drawbacks by equipping each user with a PKE keypair rather than making all users share a global commitment.

Integrated Signature and Encryption vs. SIG+PKE. Intuitively, to secure CTx system, we need to use a PKE scheme to provide confidentiality, and use a signature scheme to provide authenticity. If we follow the principle of *key separation*, i.e., use different keypairs for encryption and signing operations respectively, the overall design would be complicated. To see this, note that now each account will be associated with two keypairs. Choose which public key as address and how to link the two public keys together turn out to be very tricky.

A better solution is to adopt *key reuse* strategy, i.e., use the same keypair for both encryption and signing. This will greatly simplify the design of overall system, which is critical in cryptocurrency applications. However, reusing keypairs is not without its problems. As pointed out in [PSST11], the two uses will interact with one another badly, in such a way as to undermine the security of one or both of the primitives, e.g., the case of textbook RSA encryption and signature. In this work, we propose to use integrated signature and encryption (ISE) scheme with joint security, wherein a single keypair is used for both signature and encryption components in a secure manner (cf. Section 2.5 for definition), to replace the naive combination of signature and encryption. To the best of our knowledge, this is the first time that ISE is used in CTx system to ensure provable security. We remark that the existing proposal Zether [BAZB19] essentially adopts the key reuse strategy, employing a signature scheme and a encryption scheme with same keypair. Nevertheless, they do not explicitly identify and prove the joint security.

CTx from ISE and NIZK. We put forward a general framework of blockchain-based CTx in account-based model from ISE and NIZK. In our CTx framework, user creates an account by generating a keypair of ISE, in which public key is used as account address and secret key is used to control the account. The state of an account consists of a serial number (a counter that increments with every outgoing transaction) and an encrypted balance (encryption of plaintext balance under individual public key).

State changes is triggered by transactions from one account to another. The blockchain tracks the state of every account.

Let $\tilde{C}_1$ and $\tilde{C}_2$ be the encrypted balances of two accounts controlled by Alice and Bob respectively. Suppose Alice is going to transfer v coins to Bob. She constructs a confidential transaction via the following steps. First, she encrypts v under her public key pk_1 and Bob's public key pk_2 respectively to obtain C_1 and C_2. Then, she produces an NIZK proof π_{valid} for the well-formedness of this transaction: (i) C_1 and C_2 are two encryptions of the same positive transfer amount under pk_1 and pk_2; (ii) her left balance is still positive. Finally, she signs the serial number sn together with $(pk_1, C_1, pk_2, C_2, \pi_{\text{valid}})$ under her secret key, yielding a signature σ. The transaction is of the form $(\mathsf{sn}, pk_1, C_1, pk_2, C_2, \pi_{\text{valid}}, \sigma)$. In this way, validity of a transaction can be publicly verifiable by checking the signature and NIZK proof. If the transaction is valid, it will be recorded on the blockchain. Accordingly, Alice's balance (resp. Bob's balance) will be updated as $\tilde{C}_1 = \tilde{C}_1 - C_1$ (resp. $\tilde{C}_2 = \tilde{C}_2 + C_2$), and Alice's serial number increments. Such balance update operation implicitly requires that the underlying PKE scheme satisfies additive homomorphism.

To attain accountability, a trivial solution is to make the user involved in the transaction reveal his secret key. However, this will expose all the transactions related to this user in clear. Note that a ciphertext encrypting a given message can be expressed as an $\mathcal{NP}$ relation with secret key as witness. Accountability can thus be easily achieved by leveraging the power of zero-knowledge proofs. Given a transaction on the blockchain, say, $(\mathsf{sn}, pk_1, C_1, pk_2, C_2, \pi_{\text{valid}}, \sigma)$, either the sender or the receiver is able to justify the transaction by simply providing a zero-knowledge proof for C_1 encrypts v under pk_1 or C_2 encrypts v under pk_2, where v is the claimed transfer amount.

In summary, the signature component is used to provide authenticity (proving solvency of an account), the encryption component is used to hide the balance and transferred amount, while zero-knowledge proofs is used to prove the validity and correctness of transactions in a privacy-preserving manner.

1.3.2 PGC: a secure and efficient instantiation

A secure and efficient instantiation of the above CTx framework turns out to be the most technical part. Before proceeding, it is instructive to list the desirable features of PGC in mind:

1. self-contained — once a transaction is on-chain, it takes effect immediately. This implies that there is no out-of-band transfer and users are stateless.
2. transparent setup — system does not require a trusted setup.
3. simple and efficient — only employ simple and lightweight cryptographic schemes based on well-studied assumptions.

While feature (1) could be met by employing homomorphic PKE to hide balance and transfer amount, to attain features (2) and (3), we plan to devise lightweight ZKP that admits transparent setup, rather than resorting to general-purpose zk-SNARKs, which are heavyweight or require trusted setup.

We begin with the instantiation of ISE. A common choice is to choose Schnorr signature [Sch91] as the signature component and exponential ElGamal PKE [CGS97] as the encryption component.[1] This choice brings us at least three benefits: (i) Schnorr signature and ElGamal PKE share the same keypair (i.e., $pk = g^{sk} \in \mathbb{G}, sk \in \mathbb{Z}_p$), and the keypair is largely compatible to Bitcoin and Ethereum. (ii) The signing operation of Schnorr's signature is irrelevant to the decryption operation of ElGamal PKE, which indicates that they are jointly secure. (iii) ElGamal PKE is additively homomorphic. Efficient Sigma protocols can thus be employed to prove linear relations on algebraically encoded-values. For instance, proving plaintext equality: C_1 and C_2 encrypt the same message under pk_1 and pk_2 respectively.

Obstacle of working with Bulletproof. Besides using Sigma protocols to prove linear relations over encrypted values, we also need range proofs to prove the encrypted values lie in the right interval. In more details, we have to prove the value v encrypted in C_1 and value encrypted in $\tilde{C}_1 - C_1$ (the current balance subtracts v) lie in the right range. State-of-the-art range proof is Bulletproof [BBB+18], which enjoys efficient proof generation/verification, logarithmic proof size, and transparent setup. As per the desirable features of PGC, Bulletproof is a promising choice. Recall that Bulletproof only accepts statements of the form of Pedersen commitment $g^r h^v$. To guarantee soundness, the DL relation between

[1] In the remainder of this paper, we simply refer to the exponential ElGamal PKE as ElGamal PKE for ease of exposition.

commitment key (g, h) must be unknown to the prover. Note that an ElGamal ciphertext C of v under pk is of the form $(g^r, pk^r g^v)$. in which the second part ciphertext can be viewed as a commitment of v under commitment key (pk, g). To prove v lies in the right range, it seems that we can simply run the Bulletproof on $pk^r g^v$. However, this approach does not make sense since the prover owns an obvious trapdoor, say sk, of such commitment.

There are two approaches to get around this obstacle. The first approach is to commit v with randomness r under commitment key (g, h), then prove (v, r) in $g^v h^r$ is consistent with that in the ciphertext $(g^r, pk^r g^v)$. The shortcoming of this approach is that it brings extra overhead of proof size as well as proof generation/verification. The second approach is due to Bünz et al. [BAZB19] used in Zether. They extended Bulletproof to Σ-Bullets, which enables the interoperability between Sigma protocols and Bulletproof. Though Σ-Bullets is flexible, it requires customized design and establishes the security from scratch.

Twisted ElGamal. We are motivated to directly use Bulletproof in a black-box manner, without dissecting Bulletproof or introducing Pedersen commitment as bridge. Our idea is to twist standard ElGamal, yielding the twisted ElGamal. We sketch our twisted ElGamal as below. The setup algorithm picks two random generators (g, h) of $\mathbb{G}$ as global parameters, while the key generation algorithm is same as that of standard ElGamal. To encrypt a message $m \in \mathbb{Z}_p$ under pk, the encryption algorithm picks $r \xleftarrow{R} \mathbb{Z}_p$, then computes ciphertext as $C = (X = pk^r, Y = g^r h^m)$. The crucial difference to standard ElGamal is that the roles of key encapsulation and session key are switched and the message m is lifted on a new generator h. Our twisted ElGamal retains additive homomorphism, and is as efficient and secure as the standard ElGamal (as we will rigorously prove later). More importantly, it is zero-knowledge friendly. Note that the second part of twisted ElGamal ciphertext (even encrypted under different public keys) can be viewed as Pedersen commitment under the same commitment key (g, h), whose discrete logarithm is unknown to all users. This allows us to directly employ Bulletproof to generate range proofs for encrypted values in a black-box manner, and these proofs can be easily aggregated. We distill two distinguished cases as below.

Prover knows the randomness. In our CTx framework, to create a transaction, sender encrypts transfer amount under his public key, then proves the encrypted amount lies in the right range. This case generalizes the scenarios in which the prover is the producer of ciphertexts. Concretely, consider the twisted ElGamal ciphertext $C = (X = pk^r, Y = g^r h^v)$, to prove m lies in the right range, the prover first executes a Sigma protocol on C to prove the knowledge of (r, v), then invokes the Bulletproof on Y to prove v lies in the right range. The proof of knowledge property follows from the output of Sigma protocol's extractor equals that of the Bulletproof's extractor with overwhelming probability based on the hardness of DLP related to (g, h). See Section 4.2.2 for the details.

Prover knows the secret key. In our CTx framework, sender also needs to prove the amount encrypted by $\tilde{C}_1 - C_1$ lies in the right range. Here, $\tilde{C}_1$ is the encryption of her balance, which is the accumulation of all previous incoming and outgoing transactions. Note that the randomness beneath $\tilde{C}_1 - C_1$ is generally unknown to the sender, even assuming homomorphism on randomness.[2] This case generalizes the scenarios where the prover is the recipient of ciphertexts. Due to the lack of witness, one cannot fulfill the range proof as above. We resolve this problem by developing *ciphertext refreshing* approach. The prover knows sk and thus can decrypt $\tilde{C}_1 - C_1$, say v. Afterwards, it generates a new ciphertext C_1^* of v under fresh randomness r^*, and proves $\tilde{C}_1 - C_1$ and C^* encrypt the same message under his public key. As we will see later, this can be efficiently done by a Sigma protocol using sk as witness. Finally, it proves that v encrypted by C_1^* lies in the right range by composing a Sigma protocol and a Bulletproof, via the same way as the first case. See Section 4.2.3 for the details.

The above range proofs constitute two specialized "proof gadgets" for proving encrypted values lie in the right range. The final well-formedness proof can thus be constructed by linking them with efficient Sigma protocol for plaintext equality. Such modular design helps to reduce the footprint of overall cryptographic code, and have the potential to admit parallel proof generation/verification. We highlight the two "proof gadgets" are interesting on their own right, which may find applications in other domains as well, e.g., privacy-preserving machine learning [TMZC17].

[2] Most encryption schemes are not randomness recovering.

Achieving accountability. Given a transaction $(\mathsf{sn}, pk_1, C_1, pk_2, C_2, \pi_{\text{valid}}, \sigma)$, where $C_i = (pk_i^{r_i}, g^{r_i}h^v)_{i=1,2}$. The user with pk_i is able to justify it by simply producing a zero-knowledge proof for C_i is indeed an encryption of v under pk_i. Looking ahead, this relation can be efficiently proved by the classic Sigma protocol for discrete logarithm equality. See Protocol 4.2.3 for the details.

1.4 Related Work

The seminal Bitcoin and Ethereum do not provide strong privacy. In the past years several research lines have been dedicated to provide privacy-enhancement for them. Here we provide a succinct overview of each line.

The first research line aims to provide confidentiality. Maxwell [Max] initials the study of *confidential transaction*, He proposes a CTx scheme by employing Pedersen commitment to hide transfer amount and range proofs to prove the well-formedness of transaction. Mimblewimble/Grin [Poe, Gri] further improve Maxwell's construction by reducing the cost of signature.

The second research line aims to add anonymity. A large body of works attain anonymity via utilizing mixing mechanisms. For instance, Coinjoin [Max13], CoinShuffle [RMK14], TumbleBit [HAB+17], Dash [Das], and Mixcoin [BNM+14] are for Bitcoin, while Möbius [MM] is for Ethereum.

The third research line aims to attain both confidentiality and anonymity. Monero [Noe15] achieves confidentiality via similar technique used in Maxwell's CTx scheme, and further acquires anonymity by employing linkable ring signature and stealth address. Zcash [ZCa] achieves strong privacy by equipping two types of addresses and leveraging on key-privacy PKE and zk-SNARK.

Despite great progress in privacy-enhancement, the aforementioned schemes are not without their limitations. In terms of reliability, some of them require out-of-band transfer. In terms of efficiency, some of them suffer from slow transaction generation. In terms of security, some of them are not proven secure based on well-studied assumption, or rely on trusted setup.

Concurrent and independent work. Fauzi et al. [FMMO19] put forward a new design of cryptocurrency anonymous called Quisquis in the UTXO model. They employ updatable public keys to achieve anonymity, use a slight variant of standard ElGamal encryption to achieve confidentiality. Bünz et al. [BAZB19] propose a confidential payment system called Zether, which is compatible with Ethereum-like smart contract platforms. They use standard ElGamal to hide the balance and transfer amount, and use signature derived from NIZK to authenticate the transaction. They also sketch how to acquire anonymity for Zether using Monero technique. Both Quisquis and Zether design accompanying zero-knowledge proofs from Sigma protocols and Bulletproof, but take different approaches to tackle the incompatibility between ElGamal encryption and Bulletproof. Quisquis introduces ElGamal commitment to bridge the ElGamal encryption, uses Sigma protocol to prove the correctness of bridging, and then invoke the Bulletproof on the ElGamal commitment. Zether develops a custom ZKP called Σ-Bullets, which is a delicate integration of Bulletproof and Sigma protocol. Given an arithmetic circuit, a Σ-Bullets ensures that a public linear combination of the circuit's wires is equal to some witness of a Sigma protocol. This enhancement in turn enables proofs on algebraically-encoded values such as ElGamal encryptions or Pedersen commitments in different groups or using different commitment keys.

We highlight the following crucial differences of our work to Quisquis and Zether: (i) We focus on confidentiality, and trade anonymity for accountability. (ii) We use jointly secure ISE, rather than ad-hoc combination of signature and encryption, to build CTx framework in a provably secure way. (iii) As to instantiation, PGC employs our newly introduced twisted ElGamal rather than standard ElGamal to hide balance and transfer amount. The nice structure of twisted ElGamal enables PGC to generate range proofs for encrypted values by directly executing Bulletproof in a black-box manner, without any extra bridging cost as Quisquis. The final well-formedness proof of PGC is obtained by assembling small "proof gadgets" together in a simple and modular fashion, which would be flexible and reusable. This is opposed to Zether's approach, in which the well-formedness proof is produced by a Σ-Bullets as a whole, yet building case-tailored Σ-Bullets requires to dissect Bulletproof and skillfully design its interface to Sigma protocol.

Though we describe our CTx framework in account-based model, the mechanisms used in our work are insensitive of account type and network/consensus-level protocols. We believe they can be used as a drop-in enhancement to provide confidentiality and accountability for existing Bitcoin/Ethereum like cryptocurrencies, and may also benefit the design of Quisquis and Zether by simple adaption.

2 Preliminaries

Basic Notations. Throughout the paper, we denote the security parameter by $\lambda \in \mathbb{N}$. A function is negligible in λ, written $\mathsf{negl}(\lambda)$, if it vanishes faster than the inverse of any polynomial in λ. Let $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be two distribution ensembles indexed by λ. We say that X and Y are statistically indistinguishable, written $X \approx_s Y$, if the statistical distance between X_λ and Y_λ is negligible in λ. We say that X and Y are computationally indistinguishable, written $X \approx_c Y$, if the advantage of any PPT algorithm in distinguishing X_λ and Y_λ is $\mathsf{negl}(\lambda)$. A probabilistic polynomial time (PPT) algorithm is a randomized algorithm that runs in time $\mathsf{poly}(\lambda)$. If $\mathcal{A}$ is a randomized algorithm, we write $z \leftarrow \mathcal{A}(x_1, \ldots, x_n; r)$ to denote that $\mathcal{A}$ outputs z on inputs $(x_1, \ldots, x_n)$ and randomness r. For notational clarity we usually omit r and write $z \leftarrow \mathcal{A}(x_1, \ldots, x_n)$. For a positive integer n, we use $[n]$ to denote the set of numbers $\{1, \ldots, n\}$. For a set X, we use $|X|$ to denote its size and use $x \xleftarrow{R} X$ to denote sampling x uniformly at random from X. We use U_X to denote the uniform distribution over X.

Below, we review the cryptographic assumptions and primitives that will be used in this work.

2.1 Cryptographic Assumptions

Let $\mathsf{GroupGen}$ be a PPT algorithm that on input a security parameter 1^λ, outputs description of a cyclic group $\mathbb{G}$ of prime order $p = \Theta(2^\lambda)$, and a random generator g for $\mathbb{G}$. In what follows, we describe the discrete-logarithm based assumptions related to $(\mathbb{G}, g, p) \leftarrow \mathsf{GroupGen}(1^\lambda)$.

Definition 2.1 (Discrete Logarithm Assumption)**.** The discrete logarithm assumption holds if for any PPT adversary, we have:

$$\Pr[\mathcal{A}(g, h) = a \text{ s.t. } g^a = h] \leq \mathsf{negl}(\lambda),$$

where the probability is over the randomness of $\mathsf{GroupGen}(1^\lambda)$, $\mathcal{A}$'s random tape, and the random choice of $h \xleftarrow{R} \mathbb{G}$.

Definition 2.2 (Decisional Diffie-Hellman Assumption)**.** The DDH assumption holds if for any PPT adversary, we have:

$$\left|\Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^c) = 1]\right| \leq \mathsf{negl}(\lambda),$$

where the probability is over the randomness of $\mathsf{GroupGen}(1^\lambda)$, $\mathcal{A}$'s random tape, and the random choices of $a, b, c \xleftarrow{R} \mathbb{Z}_p$.

Definition 2.3 (Divisible Decisional Diffie-Hellman Assumption)**.** The divisible DDH assumption holds if for any PPT adversary, we have:

$$\left|\Pr[\mathcal{A}(g, g^a, g^b, g^{a/b}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^c) = 1]\right| \leq \mathsf{negl}(\lambda),$$

where the probability is over the randomness of $\mathsf{GroupGen}(1^\lambda)$, $\mathcal{A}$'s random tape, and the random choices of $a, b, c \xleftarrow{R} \mathbb{Z}_p$.

As proved in [BDZ03], the divisible DDH assumption is equivalent to the standard DDH assumption.

2.2 Commitments

A non-interactive commitment scheme is a two-party protocol between a sender and a receiver with two stages. At the committing stage, the sender commits to some value m by sending a commitment to the receiver. At the opening stage, the sender can open the commitment by providing m and some auxiliary information, by which the receiver can verify that the value it received is indeed the value committed by the sender during the committing stage. Formally, a commitment scheme consists of three polynomial time algorithms as below:

- $\mathsf{Setup}(1^\lambda)$: on input security parameter 1^λ, output public parameters pp. We assume that pp includes the descriptions of message space M, randomness space R. pp will be used as implicit input of the following two algorithms.

- $\mathsf{Com}(m;r)$: the sender commits a message m by choosing uniform random coins r, and computing $c \leftarrow \mathsf{Com}(m;r)$, then sends c to receiver.
- $\mathsf{Open}(c,m,r)$: the sender can later decommit c by sending m,r to the receiver; the receiver outputs $\mathsf{Com}(m;r) \stackrel{?}{=} c$.

Correctness. For all $pp \leftarrow \mathsf{Setup}(1^\lambda)$, all $m \in M$ and all $r \in R$, we have $\mathsf{Open}(\mathsf{Com}(m;r),m,r) = 1$. For security, we require hiding and binding.

Hiding. A commitment $\mathsf{Com}(m;r)$ should not reveal anything about its committed value of m. Let $\mathcal{A}$ be an adversary against hiding, we define its advantage via the following experiment:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\beta' = \beta : \begin{array}{l} pp \leftarrow \mathsf{Setup}(\lambda); \\ (m_0,m_1) \leftarrow \mathcal{A}_1(pp); \\ \beta \stackrel{\text{R}}{\leftarrow} \{0,1\}, r \stackrel{\text{R}}{\leftarrow} R, c \leftarrow \mathsf{Com}(m_\beta;r); \\ \beta' \leftarrow \mathcal{A}_2(c); \end{array}\right] - \frac{1}{2}.$$

A commitment scheme is perfectly hiding if $\mathsf{Adv}_{\mathcal{A}}(\lambda) = 0$ even for unbounded adversary, is statistical hiding (resp. computational hiding) if $\mathsf{Adv}_{\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ w.r.t. unbounded adversary (resp. PPT adversary).

Binding. A commitment c cannot be opened to two different messages. Let $\mathcal{A}$ be an adversary against binding, we define its advantage via the following experiment:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c} m_0 \neq m_1 \wedge \\ c = \mathsf{Com}(m_0;r_0) = \mathsf{Com}(m_1;r_1) \end{array} : \begin{array}{l} pp \leftarrow \mathsf{Setup}(\lambda); \\ (c,m_0,r_0,m_1,r_1) \leftarrow \mathcal{A}(pp); \end{array}\right].$$

A commitment scheme is perfectly binding if $\mathsf{Adv}_{\mathcal{A}}(\lambda) = 0$ even for unbounded adversary (a.k.a. $\forall m_0 \neq m_1$, their commitment values are disjoint.), statistical binding (resp. computational binding) if $\mathsf{Adv}_{\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ w.r.t. unbounded adversary (resp. PPT adversary).

Pedersen Commitment. Below we recall the Pedersen commitment [Ped91]:

- $\mathsf{Setup}(1^\lambda)$: run $(\mathbb{G},p,g) \leftarrow \mathsf{GroupGen}(1^\lambda)$, pick $h \stackrel{\text{R}}{\leftarrow} \mathbb{G}^*$, then output $pp = (\mathbb{G},p,g,h)$. Here, $M = R = \mathbb{Z}_p$, $C = \mathbb{G}$.
- $\mathsf{Com}(m;r)$: on input message $m \in \mathbb{Z}_p$ and randomness $r \stackrel{\text{R}}{\leftarrow} \mathbb{Z}_p$, output $c \leftarrow g^m h^r$.
- $\mathsf{Open}(c,m,r)$: output "1" if $c = g^m h^r$ and "0" otherwise.

The Pedersen commitment is perfectly hiding and computational binding under the discrete logarithm assumption.

2.3 Public-Key Encryption

A PKE scheme consists of four polynomial time algorithms as follows.

- $\mathsf{Setup}(1^\lambda)$: on input a security parameter 1^λ, output public parameters pp.
- $\mathsf{KeyGen}(pp)$: on input pp, output a public key pk and a secret key sk.
- $\mathsf{Enc}(pk,m)$: on input a public key pk and a plaintext m, output a ciphertext c. When emphasizing the randomness r used for encryption, we denote this by $c \leftarrow \mathsf{Enc}(pk,m;r)$.
- $\mathsf{Dec}(sk,c)$: on input a secret key sk and a ciphertext c, output a plaintext m or a distinguished symbol $\perp$ indicating that c is invalid.

Correctness. For all $pp \leftarrow \mathsf{Setup}(1^\lambda)$, all $(pk,sk) \leftarrow \mathsf{KeyGen}(pp)$ and all $m \in M$ (here M is the message space), we have $\mathsf{Dec}(sk,\mathsf{Enc}(pk,m)) = m$.

2.4 Signature

A signature scheme consists of four polynomial time algorithms as follows.

- $\mathsf{Setup}(1^\lambda)$: on input a security parameter 1^λ, output public parameters pp.
- $\mathsf{KeyGen}(pp)$: on input pp, output a public key vk and a secret key sk.
- $\mathsf{Sign}(sk, m)$: on input sk and a message m, output a signature σ.
- $\mathsf{Verify}(pk, m, \sigma)$: on input pk, a message m, and a purported signature σ, output a bit b indicating acceptance or rejection.

Correctness. For all $pp \leftarrow \mathsf{Setup}(1^\lambda)$, all $(vk, sk) \leftarrow \mathsf{KeyGen}(pp)$ and all $m \in M$ (here M is the message space), it holds that $\mathsf{Verify}(pk, m, \mathsf{Sign}(sk, m)) = 1$.

2.5 Integrated Signature and Encryption Scheme

Haber and Pinkas [HP01] introduced the concept of combined public key scheme, which is a combination of a signature scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ and encryption scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$: the existing Sign, Verify, Enc, Dec algorithms are preserved, while the two sets of Setup, KeyGen algorithms are combined into a single one. In this work, we use a special case of combined public key scheme which we refer to as integrated signature and encryption scheme (ISE) [PSST11]. ISE requires the Setup and KeyGen algorithms of the underlying signature and encryption component are identical or are subroutines of one another. In this way, the two sets of Setup and KeyGen algorithms can be completely merged to produce a single keypair.

As discussed in [PSST11], ISE uses the same keypair for both signing and encryption, thus the two operations may interact with one another badly, in such a way to undermine the security of one or both of the components. For this reason, joint security must be considered when using ISE, which captures possible dangerous interactions.

In the context of our CTx framework, the joint security of ISE stipulates that the PKE component is IND-CPA secure in the single-plaintext/2-recipient setting even in the presence of a signing oracle, while the signature component is EUF-CMA secure even in the presence of two encryption oracles. Note that in the public key setting the adversary can always perfectly simulate encryption oracles itself, thus standard EUF-CMA security implies that the signature component is secure in the joint sense and we only need to enhance the IND-CPA security for the PKE component. Let ISE $= (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Enc}, \mathsf{Dec})$. We formally define the case-tailored joint security model used in this work.

Definition 2.4 (Joint Security for ISE)**.** We say an ISE is jointly secure (case-tailored for CTx setting) if its encryption and signature components satisfy the following security notions respectively.

IND-CPA security (1-plaintext/2-recipient) in the presence of a signing oracle. Let $\mathcal{A}$ be an adversary against the PKE component and define its advantage in the following experiment:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\beta = \beta' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk_i, sk_i) \leftarrow \mathsf{KeyGen}(pp) \text{ for } i = 1, 2; \\ (state, m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sign}}}(pp, pk_1, pk_2); \\ \beta \xleftarrow{\mathrm{R}} \{0, 1\}; \\ C_i \leftarrow \mathsf{Enc}(pk_i, m_\beta) \text{ for } i = 1, 2; \\ \beta' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{sign}}}(state, C_1, C_2); \end{array}\right] - \frac{1}{2}.$$

Here, $\mathcal{O}_{\mathsf{sign}}$ provides unlimited access to signing oracle with respect to sk_1 and sk_2. More precisely, $\mathcal{O}_{\mathsf{sign}}$ returns $\mathsf{Sign}(sk_i, m)$ on input $i \in \{1, 2\}$ and $m \in M$.

EUF-CMA security. The security requirement for the signature component is exactly the standard EUF-CMA security. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against the signature component and define its advantage in the following experiment:

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

The set $\mathcal{Q}$ records queries to $\mathcal{O}_{\mathsf{sign}}$, which returns $\mathsf{Sign}(sk, m)$ on input m. The signature component is EUF-CMA if no PPT adversary $\mathcal{A}$ has non-negligible advantage in the above security experiment.

2.6 Zero-Knowledge Protocols

We first recall the definition of interactive proof systems.

Definition 2.5 (Interactive Proof System)**.** An interactive proof system is a two party protocol in which a prover can convince a verifier that some statement is true without revealing any knowledge about why it holds. Formally, it consists of three PPT algorithms (Setup, P, V) as below.

- The Setup algorithm on input 1^λ, outputs public parameter pp. Let $\mathsf{R}_{pp} \subseteq X \times W$ be an $\mathcal{NP}$ relation indexed by pp. We say $w \in W$ is a witness for a statement x iff $(x, w) \in \mathsf{R}_{pp}$. R_{pp} naturally defines a family of public-parameter-dependent $\mathcal{NP}$ languages:

  $$L_{pp} = \{x \mid \exists w \in W \text{ s.t. } (x, w) \in \mathsf{R}_{pp}\}$$

  From now on, we will drop the subscript pp occasionally when the context is clear.

- P and V are a pair of interactive algorithms, which both take pp as implicit input and the statement x as common input. We use the notation $tr \leftarrow \langle P(x), V(y)\rangle$ to denote the transcript of an execution between P and V, where P has input x and V has input y. We write $\langle P(x), V(y)\rangle = b$ depending on whether V accepts, $b = 1$, or rejects, $b = 0$. When the context is clear, we will also slightly abuse the notation of $\langle P(x), V(y)\rangle$ to denote V's view (View_V) in the interaction, which consists of V's input tape, random tape and the incoming messages sent by P.

An interactive proof system satisfies the following two properties:

Completeness. For any $(x, w) \in \mathsf{R}_{pp}$ where $pp \leftarrow \mathsf{Setup}(1^\lambda)$, we have:

$$\Pr[\langle P(x, w), V(x)\rangle = 1] = 1$$

Statistical soundness. For any $x \notin L_{pp}$ where $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any cheating prover P^*, we have:

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

If we restrict P^* to be a PPT cheating prover in the above definition, we obtain an interactive argument system.

Public coin. An interactive proof/argument system is public-coin if all messages sent from V are chosen uniformly at random and independent of P's message.

We then recall several zero-knowledge extensions of interactive zero knowledge proof systems that will be used in this work.

Definition 2.6 (Zero-Knowledge Argument of Knowledge)**.** We say an interactive proof system is a zero-knowledge argument of knowledge if it satisfies standard completeness, argument of knowledge and zero knowledge.

Following [BCC+16], we use computational witness-extended emulation to define argument of knowledge. Intuitively, this definition says that whenever a malicious prover produces an accepting argument with some probability, there exists an emulator producing a similar argument with roughly the same probability together with a witness.

Computational witness-extended emulation. For all deterministic polynomial time P^* there exists an expected PPT emulator E such that for all PPT interactive adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function $\mu(\lambda)$ such that:

$$\Pr\left[\begin{array}{l} pp \leftarrow \mathsf{Setup}(\lambda); \\ (x, s) \leftarrow \mathcal{A}_1(pp); \\ tr \leftarrow \langle P^*(x, s), V(x)\rangle; \\ \mathcal{A}_2(tr) = 1 \end{array}\right] - \Pr\left[\begin{array}{l} pp \leftarrow \mathsf{Setup}(\lambda); \\ (x, s) \leftarrow \mathcal{A}_1(pp); \\ (tr, w) \leftarrow E^{\mathcal{O}}(x); \\ \mathcal{A}_2(tr) = 1 \wedge \\ tr \text{ accepts} \Rightarrow (x, w) \in \mathsf{R}_{pp} \end{array}\right] \leq \mu(\lambda)$$

where the oracle $\mathcal{O} = \langle P^*(x, s), V(x)\rangle$ permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards. In the definition, s is interpreted as the state of P, including the randomness and auxiliary input. According to the definition, whenever P^* makes a convincing argument in state s, E can extract a witness. This is why it defines argument of knowledge.

Computational zero-knowledge. For any malicious PPT V^* there exists an expected PPT simulator $\mathcal{S}$ such that for any $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and $(x, w) \in \mathsf{R}_{pp}$, we have:

$$\langle P(x, w), V^*(x) \rangle \approx_c \mathcal{S}(x)$$

Computational zero-knowledge can be strengthened to statistical (resp. perfect) zero-knowledge by requiring the views are statistically indistinguishable (resp. identical).

Definition 2.7 (Sigma Protocol (Σ-protocol) [Dam])**.** An interactive proof system is called a Sigma protocol if it follows the following communication pattern (3-round public-coin):

1. (Commit) P sends a first message a to V;
2. (Challenge) V sends a random challenge e to P;
3. (Response) P replies with a second message z.

and satisfies standard completeness and the variants of soundness and zero-knowledge as below:

Special soundness. For any x and any pair of accepting transcripts (a, e, z), (a, e', z') with $e \neq e'$, there exists a PPT extractor outputs a witness w for x.

Special honest-verifier zero-knowledge (SHVZK). There exists a PPT simulator $\mathcal{S}$ such that for any $(x, w) \in \mathsf{R}_{pp}$ and randomness e, we have:

$$\langle P(x, w), V(x, e) \rangle \equiv \mathcal{S}(x, e)$$

Definition 2.8 (Range Proof)**.** For a commitment scheme over total ordering message space M and randomness space R, a range proof is a zero-knowledge argument of knowledge for the following language:

$$L = \{c \mid \exists m \in M, r \in R \text{ s.t. } c = \mathsf{Com}(m; r) \wedge x \in [a, b]\}$$

Interaction is typically expensive and sometime is even impossible, thus removing interaction is of particular interest in practice. However, non-interactive zero-knowledge (NIZK) proofs for non-trivial languages are impossible in the plain model. We need to consider NIZK in the common reference string (CRS) model, wherein a string of a special structure is generated after setup phase, and made available to everyone to prove/verify statement.

Definition 2.9 (Non-Interactive Zero-Knowledge Proof [BFM88, FLS90])**.** An NIZK proof system in the CRS model consists of four PPT algorithms $(\mathsf{Setup}, \mathsf{CRSGen}, P, V)$:

- $\mathsf{Setup}(1^\lambda)$: same as that of ordinary zero-knowledge proof system, which on input 1^λ, output public parameters pp.
- $\mathsf{CRSGen}(pp)$: on input pp, output a common reference string crs.
- $P(crs, x, w)$: on input crs and a statement-witness pair $(x, w) \in \mathsf{R}_{pp}$, output a proof π. We also denote this process by $\pi \leftarrow \mathsf{Prove}(crs, x, w)$.
- $V(crs, x, \pi)$: on input crs, a statement x, and a proof π, outputs "0" if rejects and "1" if accepts. We also denote this process by $b \leftarrow \mathsf{Verify}(crs, x, \pi)$.

An NIZK proof system in the CRS model satisfies the following requirements:

Completeness. For any $(x, w) \in \mathsf{R}_{pp}$ where $pp \leftarrow \mathsf{Setup}(1^\lambda)$,

$$\Pr\left[V(crs, x, \pi) = 1 : \begin{array}{l} crs \leftarrow \mathsf{CRSGen}(pp); \\ \pi \leftarrow P(crs, x, w); \end{array}\right] = 1.$$

Statistical soundness. For any cheating prover P^*,

$$\Pr = \left[\begin{array}{c} x \notin L \wedge \\ V(crs, x, \pi) = 1 \end{array} : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ crs \leftarrow \mathsf{CRSGen}(pp); \\ (x, \pi) \leftarrow P^*(crs); \end{array}\right] \leq \mathsf{negl}(\lambda).$$

This definition is in the adaptive sense in that P^* may choose x after seeing crs. Statistical soundness can be relaxed to computational soundness by restricting P^* to be a PPT algorithm. In this case, a proof system degenerates to an argument system.

Computational zero-knowledge. For any $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that:

$$\Pr\left[\begin{array}{c} crs \leftarrow \mathsf{CRSGen}(pp); \\ (x,w) \leftarrow \mathcal{A}_1(crs); \\ \pi \leftarrow P(crs,x,w); \\ \mathcal{A}_2(crs,x,\pi) = 1 \end{array}\right] - \Pr\left[\begin{array}{c} (crs,\tau) \leftarrow \mathcal{S}_1(pp); \\ (x,w) \leftarrow \mathcal{A}_1(crs); \\ \pi \leftarrow \mathcal{S}_2(crs,x,\tau); \\ \mathcal{A}_2(crs,x,\pi) = 1 \end{array}\right] \leq \mathsf{negl}(\lambda).$$

This definition is also in the adaptive sense in that $\mathcal{A}_1$ may adaptively choose x after seeing crs. Computational zero-knowledge can be strengthened to statistical zero-knowledge by requiring the above holds even for unbounded $\mathcal{A}$.

Interestingly, in the random oracle model NIZK is possible without relying on common reference string. The celebrated Fiat-Shamir transform [FS86] shows how to compile a Sigma protocol into an NIZK by modeling cryptographic hash function as random oracle. It not only removes interaction, but also strengthens honest-verifier zero-knowledge to full zero-knowledge (against malicious verifiers). Fiat-Shamir transform actually applies to any public-coin SHVZK argument of knowledge. Formally, we have the following theorem.

Theorem 2.1 (Fiat-Shamir Transform [BR93, FKMV12])**.** *Let* (Setup, P, V) *be a* $(2k+1)$*-move public-coin SHVZK argument of knowledge,* x *be the statement,* a_i *be prover's* i*th round message and* e_i *be verifier's* i*th round challenge,* H *is a hash function whose range equal to verifier's challenge space. By setting* $e_i = \mathsf{H}(a_1, \ldots, a_i)$ *in* (Setup, P, V)*, we obtain* $(\mathsf{Setup}, P^{\mathsf{H}}, V^{\mathsf{H}})$*, which is an NIZK assuming* H *is a random oracle.*[3]

General Forking Lemma. We recall the general forking lemma of [BCC+16, BBB+18] that will be used in our proofs.

Suppose that we have a $(2k+1)$-move public-coin argument with k challenges, $e_1, \ldots, e_k$ in sequence. Let $n_i \geq 1$ for $i \in [k]$. Consider $\Pi_{i=1}^k n_i$ accepting transcripts whose challenges satisfying the following tree format. The tree has depth k and $\Pi_{i=1}^k n_i$ leaves. The root of the tree is labeled with the statement. Each node of depth $i < k$ has exactly n_i children, each labeled with a distinct value of the ith challenge e_i. This can be referred to as an $(n_1, \ldots, n_k)$-tree of accepting transcripts, which is a natural generalization of special soundness for Sigma protocols where $k = 1$ and $n = 2$. For the simplicity in the following lemma, we assume that the challenge space is $\mathbb{Z}_p$ and $|p| = \lambda$.

Theorem 2.2 (Forking Lemma [BCC+16, BBB+18])**.** *Let* (Setup, P, V) *be a* $(2k+1)$*-move, public-coin interactive protocol. Let* E *be a PPT witness extraction algorithm that succeeds with probability* $1 - \mu(\lambda)$ *for some negligible function* $\mu(\lambda)$ *in extracting a witness from an* $(n_1, \ldots, n_k)$*-tree of accepting transcripts. If* $\Pi_{i=1}^k n_i$ *is bounded by a polynomial in* λ*, then* (Setup, P, V) *has witness-extended emulation.*

3 Confidential Transaction System with Accountability

3.1 A General Framework of Accountable CTx from ISE and NIZK

Let ISE $= (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Enc}, \mathsf{Dec})$ be an ISE scheme whose PKE component satisfies the following two properties: (i) additively homomorphic on message space $\mathbb{Z}_p$; (2) each ciphertext uniquely determines its randomness and plaintext.[4] Fix the public parameters, the KeyGen algorithm of ISE naturally induces an $\mathcal{NP}$ relation $\mathsf{R}_{\mathrm{key}} \subseteq PK \times SK$, i.e., $\mathsf{R}_{\mathrm{key}} = \{(pk, sk) : \exists r \text{ s.t. } (pk, sk) = \mathsf{KeyGen}(pp; r)\}$. Let NIZK $= (\mathsf{Setup}, \mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$.[5] be an NIZK proof system. We present a general accountable CTx framework from ISE and NIZK as below.

- $\mathsf{Setup}(1^\lambda)$. This algorithm is used to setup the CTx system. It takes as input the security parameter 1^λ, runs $pp_{\mathrm{ise}} \leftarrow \mathrm{ISE.Setup}(1^\lambda)$, $pp_{\mathrm{nizk}} \leftarrow \mathrm{NIZK.Setup}(1^\lambda)$, $crs \leftarrow \mathrm{NIZK.CRSGen}(pp_{\mathrm{nizk}})$, sets $pp =$

[3] To get a unifying syntax of NIZK, one can also interpret the description of H as common reference string.

[4] This property can be viewed as an enhancement of standard correctness.

[5] We describe our general framework using NIZK in the CRS model. The construction and security proof carries out naturally if using NIZK in the random oracle model instead.

$(pp_{\text{ise}}, pp_{\text{nizk}}, crs)$ as global public parameters that will be implicitly used by all user's algorithms. Let $\mathcal{V}$ be the interval $\{0, \ldots, 2^\ell - 1\}$, where $2^\ell - 1$ is an upper bound on the maximum possible number of coins in the system (e.g., set $\ell = 64$). For simplicity, we also assume $\mathcal{V}$ is the legal range of transfer amount.

- $\mathsf{CreateAcct}(\tilde{m}, \mathsf{sn})$. This algorithm is used to create an account. It takes as input an initial balance $\tilde{m} \in \mathbb{Z}_p$ and serial number $\mathsf{sn} \in \{0,1\}^n$ (e.g., $n = 256$), runs $\text{ISE.}\mathsf{KeyGen}(pp_{\text{ise}})$ to generate a keypair (pk, sk), sets $\tilde{C} \leftarrow \text{ISE.}\mathsf{Enc}(pk, \tilde{m}; r)$ as the initial encrypted balance, sets sn as the initial serial number,[6] and outputs pk as the public key (acts as the pseudonym for a user and serves as the account address) and sk as the secret key.
- $\mathsf{RevealBalance}(sk, \tilde{C})$. This algorithm is used to reveal the balance of an account. It takes as input the secret key sk and the current encrypted balance $\tilde{C}$, outputs $\tilde{m} \leftarrow \text{ISE.}\mathsf{Dec}(sk, \tilde{C})$.
- $\mathsf{CreateCTx}(sk_1, pk_1, v, pk_2)$. This algorithm is used to transfer v coins from account pk_1 to account pk_2. On input sender's keypair (pk_1, sk_1), the transfer amount v, and receiver's public key pk_2, the algorithm first checks if $(\tilde{m}_1 - v), v \in \mathcal{V}$ (here $\tilde{m}_1$ is the current balance of pk_1), returns $\perp$ if not. Otherwise, it creates a confidential transaction ctx via the following steps:

  1. compute $C_1 \leftarrow \text{ISE.}\mathsf{Enc}(pk_1, v; r_1)$, $C_2 \leftarrow \text{ISE.}\mathsf{Enc}(pk_2, v; r_2)$, set the memo information of the transaction as $\mathsf{memo} = (\tilde{C}_1, pk_1, C_1, pk_2, C_2)$[7];
  2. run NIZK.Prove with witness (sk_1, r_1, r_2, v) to generate a proof π_{valid} for $\mathsf{memo} = (\tilde{C}_1, pk_1, C_1, pk_2, C_2) \in L_{\text{valid}}$, where L_{valid} is defined as

$$\{(\tilde{C}_1, pk_1, C_1, pk_2, C_2) \mid \exists sk_1, r_1, r_2, v \text{ s.t. } C_i = \text{ISE.}\mathsf{Enc}(pk_i, v; r_i)_{i=1,2} \\ \wedge v \in \mathcal{V} \ \wedge (pk_1, sk_1) \in \mathsf{R}_{\text{key}} \wedge \text{ISE.}\mathsf{Dec}(sk_1, \tilde{C}_1 - C_1) \in \mathcal{V}\}$$

     By the unique binding property of ISE's PKE component[8], L_{valid} can be decomposed to $L_{\text{equal}} \wedge L_{\text{right}} \wedge L_{\text{enough}}$ as specified below.

$$L_{\text{equal}} = \{(pk_1, C_1, pk_2, C_2) \mid \exists r_1, r_2, v \text{ s.t. } C_i = \text{ISE.}\mathsf{Enc}(pk_i, v; r_i)_{i=1,2}\}$$
$$L_{\text{right}} = \{(pk_1, C_1) \mid \exists r_1, v \text{ s.t. } C_1 = \text{ISE.}\mathsf{Enc}(pk_1, v; r_1) \wedge v \in \mathcal{V}\}$$
$$L_{\text{enough}} = \{(pk_1, \tilde{C}_1, C_1) \mid \exists sk_1 \text{ s.t. } (pk_1, sk_1) \in \mathsf{R}_{\text{key}} \wedge \text{ISE.}\mathsf{Dec}(sk_1, \tilde{C}_1 - C_1) \in \mathcal{V}\}$$

     L_{equal} captures the amount sent by pk_1 equals to that received by pk_2, L_{right} captures the amount sent by pk_1 lies in the right range, and L_{enough} captures the balance of pk_1 is enough for making the transfer.
  3. run $\sigma \leftarrow \text{ISE.}\mathsf{Sign}(sk_1, (\mathsf{sn}, \mathsf{memo}, \pi_{\text{valid}}))$, where sn is the current serial number of pk_1; the entire confidential transaction $\mathsf{ctx} = (\mathsf{sn}, \mathsf{memo}, \pi_{\text{valid}}, \sigma)$.
- $\mathsf{VerifyCTx}(\mathsf{ctx})$. This algorithm is used to check if a confidential transaction ctx is valid. It takes as input $\mathsf{ctx} = (\mathsf{sn}, \mathsf{memo}, \pi_{\text{valid}}, \sigma)$, parses $\mathsf{memo} = (\tilde{C}_1, pk_1, C_1, pk_2, C_2)$, then checks its validity via the following steps:

  1. check if sn is a fresh serial number of pk_1 (this can be done by inspecting the blockchain);
  2. check if $\text{ISE.}\mathsf{Verify}(pk_1, (\mathsf{sn}, \mathsf{memo}, \pi_{\text{valid}}), \sigma) = 1$;
  3. check if $\text{NIZK.}\mathsf{Verify}(crs, \mathsf{memo}, \pi_{\text{valid}}) = 1$.

  If all the above tests pass, outputs "1". Miners confirm that ctx is valid and records it on the blockchain by engaging some consensus protocol; sender updates his balance as $\tilde{C}_1 = \tilde{C}_1 - C_1$ and increments the serial number, and receiver updates his balance as $\tilde{C}_2 = \tilde{C}_2 + C_2$. Else, outputs "0" and miners discard ctx.
- $\mathsf{JustifyCTx}(sk_i, \mathsf{ctx}, v)$. This algorithm is used to justify the value hidden in ctx is indeed v in a zero-knowledge manner. Here, we assume ctx is a valid confidential transaction recorded on the blockchain. On input sk_i, v and $\mathsf{ctx} = (\mathsf{sn}, \mathsf{memo}, \pi_{\text{valid}}, \sigma)$, where $\mathsf{memo} = (\tilde{C}_1, pk_1, C_1, pk_2, C_2)$,

[6] By default, $\tilde{m}$ and sn should be zero, r should be a fixed and publicly known randomness, say the zero string 0^λ. This settlement guarantees that the initial account state is publicly auditable. Here, we do not make it as an enforcement for flexibility.

[7] Since $\tilde{C}_1$ is public state of account pk_1, it can be removed from memo to shrink the overall confidential transaction size. We include it here only for ease of exposition.

[8] Each valid ciphertext uniquely determines the randomness and plaintext.

sk_i is a secret key for pk_i, the algorithm runs NIZK.Prove with witness sk_i to generate a proof π_{correct} for the statement $(C_i, pk_i, v) \in L_{\text{correct}}$, where $i = \{1, 2\}$. Here, the language L_{correct} is defined as:

$$\{(C, pk, v) \mid \exists sk \text{ s.t. } (pk, sk) \in \mathsf{R}_{\text{key}} \wedge v = \text{ISE.}\mathsf{Dec}(sk, C)\}$$

Anyone can check if the transaction value in ctx is indeed the claimed value v by checking if the correctness proof is correct, i.e., NIZK.$\mathsf{Verify}(crs, C_i, pk_i, v, \pi_{\text{correct}})$.

3.2 Security Model for CTx Framework

Following the treatment of 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, a CTx system should provide *authenticity*, *confidentiality* and *soundness*. Authenticity requires that the sender can only be the owner of an account, nobody else (who does not know the secret key) is able to make a transfer from this account. Confidentiality requires that other than the sender and receiver, no one can learn the value hidden in a confidential transaction. While the former two notions address security against outsider adversary (who does not know the secret keys of sender and receiver), soundness addresses security against insider adversary (e.g., the sender himself). It requires that an adversary is unable to generate an accepting proof π_{valid} for any ill-formed transaction $\mathsf{memo} \notin L_{\text{valid}}$, even it knows the secret key of sender.

We then formalize the above intuitions into a game-based security model for CTx system. Let $\mathcal{A}$ be an adversary attacking a CTx system, whose attack behaviors are modeled as adversarial queries to oracles implemented by a challenger $\mathcal{CH}$. $\mathcal{CH}$ keeps track of all honest and corrupt registered accounts and valid transactions by maintaining three lists T_{honest}, T_{corrupt} and T_{ctx}, which are initially empty. Below we formally describe the oracles provided to the adversary.

- $\mathcal{O}_{\mathsf{regH}}$: $\mathcal{A}$ queries this oracle for registering an honest account. $\mathcal{CH}$ responds as below: picks a random balance $\tilde{m}$ and serial number sn, runs $\mathsf{CreateAcct}(\tilde{m}, \mathsf{sn})$ to obtain (pk, sk) and $\tilde{C} \leftarrow \text{ISE.}\mathsf{Enc}(pk, \tilde{m})$. $\mathcal{CH}$ records $(pk, sk, \tilde{C}, \tilde{m}, \mathsf{sn})$ in T_{honest}, then returns $(pk, \tilde{C}, \mathsf{sn})$ to $\mathcal{A}$. This type of oracle captures $\mathcal{A}$ can observe the public information of honest accounts in the system. Hereafter, let Q_{honest} be the maximum number of honest account registration queries that $\mathcal{A}$ makes.
- $\mathcal{O}_{\mathsf{regC}}$: $\mathcal{A}$ queries this oracle with a public key pk, an initial encrypted balance $\tilde{C}$ as well as an initial serial number sn. $\mathcal{CH}$ records $(pk, \perp, \tilde{C}, \perp, \mathsf{sn})$ in T_{corrupt}. We stress that pk and $\tilde{C}$ submitted by $\mathcal{A}$ may not be honestly generated. This type of oracle captures $\mathcal{A}$ can fully control some accounts in the system.
- $\mathcal{O}_{\mathsf{extH}}$: $\mathcal{A}$ queries this oracle with a public key pk that was registered as an honest account, a.k.a. in T_{honest}. $\mathcal{CH}$ returns the associated sk to $\mathcal{A}$ and moves this entry to T_{corrupt}. This oracle captures $\mathcal{A}$ can corrupt some honest accounts.
- $\mathcal{O}_{\mathsf{trans}}$: $\mathcal{A}$ queries this oracle with (pk_s, pk_r, v) to conduct a confidential transaction, subject to the restriction that $pk_s \in T_{\text{honest}}$. This restriction is natural because for $pk_s \in T_{\text{corrupt}}$, $\mathcal{A}$ can generate the confidential transaction itself. $\mathcal{CH}$ handles such queries as below. If v does not constitute a valid transaction originated from pk_s with balance $\tilde{m}_s$, i.e., $v \notin \mathcal{V}$ or $(\tilde{m}_s - v) \notin \mathcal{V}$, $\mathcal{CH}$ returns $\perp$. Else, $\mathcal{CH}$ runs $\mathsf{ctx} \leftarrow \mathsf{CreateCTx}(sk_s, pk_s, pk_r, v)$, updates the state of associated accounts, records ctx on the ledger T_{ctx}, then sends ctx to $\mathcal{A}$. This oracle captures $\mathcal{A}$ can direct honest accounts to make specific transactions as its will.
- $\mathcal{O}_{\mathsf{inject}}$: $\mathcal{A}$ submits a confidential transaction ctx subject to the restriction that $pk_s \in T_{\text{corrupt}}$. If $\mathsf{VerifyCTx}(\mathsf{ctx}) = 1$, $\mathcal{CH}$ updates the associated account state and records ctx in T_{ctx}. Otherwise, $\mathcal{CH}$ ignores. This oracle captures $\mathcal{A}$ can generate (possibly malicious) transactions itself.

Authenticity. We define authenticity via the following security experiment between $\mathcal{A}$ and $\mathcal{CH}$.

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c} pk_s^* \in T_{\text{honest}} \wedge \\ \text{ISE.}\mathsf{Verify}(pk_s^*, (\mathsf{sn}^*, \mathsf{memo}^*, \pi_{\text{valid}}^*), \sigma^*) = 1 \\ \wedge\ (\mathsf{sn}^*, \mathsf{memo}^*, \pi_{\text{valid}}^*) \notin T_{\text{ctx}}(pk_s^*) \end{array} : \begin{array}{c} pp \leftarrow \mathsf{Setup}(\lambda); \\ \mathsf{ctx}^* \leftarrow \mathcal{A}(pp); \end{array}\right].$$

Here $\mathsf{ctx}^* = (\mathsf{sn}^*, \mathsf{memo}^* = (pk_s^*, C_s^*, pk_r^*, C_r^*), \pi_{\text{valid}}^*, \sigma^*)$ is a confidential transaction from pk_s^*, $T_{\text{ctx}}(pk_s^*)$ denotes the set of all the confidential transactions originated from pk_s^* in T_{ctx}.

Remark 3.1. The above definition of authenticity is essentially much stronger than its intuition. It stipulates no PPT adversary can generate a valid signature for a fresh $(\mathsf{sn}^*, \mathsf{memo}^*, \pi^*_{\text{valid}})$. In our CTx system, unauthorized transfers against pk_s likely diverge from sender's original intention only when the outsider adversary (without the knowledge sk_s) forges a signature for fresh $(\mathsf{sn}^*, \mathsf{memo}^*)$, because the two pieces encode the core information (serial number, sender and receiver's addresses, and transfer amount) of a transaction.

With this observation, we can weaken the definition of authenticity by only requiring unforgeability for $(\mathsf{sn}, \mathsf{memo})$. Hence, it suffices to sign only $(\mathsf{sn}, \mathsf{memo})$ rather than the entire $(\mathsf{sn}, \mathsf{memo}, \pi_{\text{valid}})$. As we will see in 6, this weakening leads a more efficient instantiation of our CTx framework.

Confidentiality. We define confidentiality via the following security experiment between $\mathcal{A}$ and $\mathcal{CH}$.

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\beta = \beta' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(\lambda); \\ (state, pk^*_s, pk^*_r, v_0, v_1) \leftarrow \mathcal{A}_1(pp); \\ \beta \xleftarrow{\text{R}} \{0,1\}; \\ \mathsf{ctx}^* \leftarrow \mathsf{CreateCTx}(sk^*_s, pk^*_s, pk^*_r, v_\beta); \\ \beta' \leftarrow \mathcal{A}_2(state, \mathsf{ctx}^*); \end{array}\right] - \frac{1}{2}.$$

To prevent trivial attacks, $\mathcal{A}$ is subject to the following restrictions: (i) pk^*_s, pk^*_r chosen by $\mathcal{A}_1$ are required to be honest accounts, and $\mathcal{A}_2$ is not allowed to make corrupt queries to either pk^*_s or pk^*_r; (ii) let $\tilde{m}_s$ be the balance of pk^*_s, both v_0 and v_1 must constitute a valid transaction originated from pk^*_s; (iii) let v_{sum} be accumulation of the transfer amounts in $\mathcal{O}_{\text{trans}}$ queries related to pk^*_s after receiving ctx^*, we require that both $\tilde{m}_s - v_0 - v_{\text{sum}}$ and $\tilde{m}_s - v_1 - v_{\text{sum}}$ lie in $\mathcal{V}$. Restriction (i) prevents trivial attack by decryption, and restrictions (ii), (iii) prevent inferring β by testing whether overdraft happens.

Soundness. We define soundness via the following security experiment between $\mathcal{A}$ and $\mathcal{CH}$.

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c} \mathsf{memo} \notin L_{\text{valid}} \wedge \\ \text{NIZK.Verify}(\mathsf{memo}^*, \pi^*_{\text{valid}}) = 1 \end{array} : \begin{array}{l} pp \leftarrow \mathsf{Setup}(\lambda); \\ (\mathsf{memo}^*, \pi^*_{\text{valid}}) \leftarrow \mathcal{A}(pp); \end{array}\right].$$

3.3 Proof for General CTx Framework

For the security of our CTx framework, we have the following theorem.

Theorem 3.1. *Assuming the security of ISE and NIZK, our CTx framework is secure.*

Proof. We prove this theorem via the following three lemmas.

Lemma 3.2. *Assuming the security of ISE's signature component and the adaptive zero-knowledge property of NIZK, our CTx framework satisfies authenticity.*

Proof. We proceed via a sequence of games.

Game 0. The real experiment for authenticity. $\mathcal{CH}$ interacts with $\mathcal{A}$ as below.

1. Setup: $\mathcal{CH}$ runs $pp_{\text{ise}} \leftarrow \text{ISE.}\mathsf{Setup}(1^\lambda)$, $pp_{\text{nizk}} \leftarrow \text{NIZK.}\mathsf{Setup}(1^\lambda)$, and $crs \leftarrow \text{NIZK.}\mathsf{CRSGen}(pp_{\text{nizk}})$, then sends $pp = (pp_{\text{ise}}, pp_{\text{nizk}}, crs)$ to $\mathcal{A}$.
2. Queries: Throughout the experiment, $\mathcal{A}$ can adaptively query $\mathcal{O}_{\mathsf{regH}}$, $\mathcal{O}_{\mathsf{regC}}$, $\mathcal{O}_{\mathsf{extH}}$, $\mathcal{O}_{\mathsf{trans}}$ and $\mathcal{O}_{\mathsf{inject}}$, and $\mathcal{CH}$ answers these queries as described above.
3. Forge: $\mathcal{A}$ outputs $\mathsf{ctx}^* = (\mathsf{sn}^*, \mathsf{memo}^* = (\tilde{C}_s, pk^*_s, C^*_s, pk^*_r, C^*_r), \pi^*_{\text{valid}}, \sigma^*)$, and wins if $pk^*_s \in T_{\text{honest}} \wedge \text{ISE.}\mathsf{Verify}(pk^*_s, (\mathsf{sn}^*, \mathsf{memo}^*, \pi^*_{\text{valid}}), \sigma^*) = 1 \wedge (\mathsf{sn}^*, \mathsf{memo}^*, \pi^*_{\text{valid}}) \notin T_{\text{ctx}}(pk^*_s)$.

Game 1. Game 1 is same as Game 0 except $\mathcal{CH}$ makes a random guess for the index of target pk^*_s at the beginning, i.e., randomly picks an index $j \in [Q_{\text{honest}}]$. If $\mathcal{A}$ makes an extraction query of pk_j in the learning stage, or $\mathcal{A}$ picks $pk^*_s \neq pk_j$ in the challenge stage, $\mathcal{CH}$ aborts. Let W be the event that $\mathcal{CH}$ does not abort. It is easy to see that $\Pr[W] \geq 1/Q_{\text{honest}}$.

Conditioned on $\mathcal{CH}$ does not abort, $\mathcal{A}$'s view in Game 0 is identical to that in Game 1. Thereby, we have:

$$\Pr[S_1] \geq \Pr[S_0] \cdot \frac{1}{Q_{\text{honest}}}$$

Game 2. Same as Game 1 except that $\mathcal{CH}$ generates the zero-knowledge proof via running $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ in the simulation mode. More precisely. In the Setup stage, $\mathcal{CH}$ runs $(crs, \tau) \leftarrow \mathcal{S}_1(pp_{\text{nizk}})$. When handling transaction queries, $\mathcal{CH}$ runs $\mathcal{S}_2(crs, \tau, \mathsf{memo})$ to generate π_{valid} for $\mathsf{memo} \in L_{\text{valid}}$. By a direct reduction to the adaptive zero-knowledge property of the underlying NIZK, we have:

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

We now argue that no PPT adversary has non-negligible advantage in Game 2.

Claim 3.3. *Assuming the EUF-CMA security of ISE's signature component,* $\Pr[S_2] \leq \mathsf{negl}(\lambda)$ *for all PPT adversary* $\mathcal{A}$*.*

Proof. Suppose there exists a PPT $\mathcal{A}$ has non-negligible advantage in Game 2, we can build an adversary $\mathcal{B}$ breaks the EUF-CMA security of ISE with the same advantage. Given the challenge (pp_{ise}, pk^*), $\mathcal{B}$ simulates Game 2 as follows:

1. Setup: $\mathcal{B}$ runs $pp_{\text{nizk}} \leftarrow \text{NIZK.}\mathsf{Setup}(1^\lambda)$, $\mathcal{S}_1.\mathsf{Setup}(pp_{\text{nizk}}) \rightarrow (crs, \tau)$, sends $pp = (pp_{\text{ise}}, pp_{\text{nizk}}, crs)$ to $\mathcal{A}$. $\mathcal{B}$ randomly picks an index $j \in [Q_{\text{honest}}]$
2. Queries: Throughout the experiment, $\mathcal{A}$ can query the following types of oracles adaptively. $\mathcal{B}$ answers these queries by maintaining two lists T_{honest} and T_{corrupt}, which both are initially empty.
   - $\mathcal{O}_{\mathsf{regH}}$: On the i-th query, $\mathcal{B}$ picks a random balance $\tilde{m}$ and serial sn and proceeds as below:
     - If $i \neq j$, $\mathcal{B}$ runs $\mathsf{CreateAcct}(\tilde{m}, \mathsf{sn})$ to obtain (pk, sk) and the initial encrypted balance $\tilde{C} \leftarrow \text{ISE.}\mathsf{Enc}(pk, \tilde{m})$, then records $(pk, sk, \tilde{C}, \tilde{m}, \mathsf{sn})$ in T_{honest}.
     - If $i = j$, $\mathcal{B}$ sets $pk = pk^*$, $\mathcal{B}$ computes $\tilde{C} \leftarrow \text{ISE.}\mathsf{Enc}(pk, \tilde{m})$, then records $(pk, \perp, \tilde{C}, \tilde{m}, \mathsf{sn})$ in T_{honest}.

     Finally, $\mathcal{B}$ returns $(pk, \tilde{C}, \mathsf{sn})$ to $\mathcal{A}$.
   - $\mathcal{O}_{\mathsf{regC}}$: $\mathcal{A}$ makes this query with a public key pk, a serial number sn and an initial encrypted balance $\tilde{C}$. $\mathcal{B}$ records $(pk, \perp, \tilde{C}, \perp, \mathsf{sn})$ in T_{corrupt}.
   - $\mathcal{O}_{\mathsf{extH}}$: $\mathcal{A}$ makes a query with pk in T_{honest}. If $pk = pk_j$, $\mathcal{B}$ aborts. Else, $\mathcal{B}$ returns sk to $\mathcal{A}$, then moves the corresponding entry to T_{corrupt}.
   - $\mathcal{O}_{\mathsf{trans}}$: $\mathcal{A}$ makes a transaction query (pk_s, pk_r, v) subject to the restriction that $pk_s \in T_{\text{honest}}$. Let sn be the serial number, $\tilde{C}_s$ be the encrypted balance of pk_s. $\mathcal{B}$ proceeds as below:
     (a) compute $C_s \leftarrow \text{PKE.}\mathsf{Enc}(pk_s, v)$, $C_r \leftarrow \text{PKE.}\mathsf{Enc}(pk_r, v)$;
     (b) set $\mathsf{memo} = (\tilde{C}_s, pk_r, C_r, pk_s, C_s)$, compute $\pi_{\text{valid}} \leftarrow \mathcal{S}_2(crs, \tau, \mathsf{memo})$;
     (c) if $pk_s \neq pk_j$, generate a signature σ for $(\mathsf{sn}, \mathsf{memo}, \pi_{\text{valid}})$ with sk_s; else, generate such signature σ by querying the signing oracle.

     $\mathcal{B}$ updates the states of associated accounts, then returns ctx to $\mathcal{A}$.
   - $\mathcal{O}_{\mathsf{inject}}$: $\mathcal{A}$ submits a confidential transaction ctx subject to the restriction that $pk_s \in T_{\text{corrupt}}$. If $\mathsf{VerifyCTx}(\mathsf{ctx}) = 1$, $\mathcal{B}$ updates the associated account state. Otherwise, $\mathcal{B}$ ignores.
3. Forge: $\mathcal{A}$ outputs $\mathsf{ctx}^* = (\mathsf{sn}^*, \mathsf{memo}^*, \pi^*_{\text{valid}}, \sigma^*)$ as the forge transaction. If $pk^*_s \neq pk_j$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ forwards $(\mathsf{sn}^*, \mathsf{memo}^*, \pi^*_{\text{valid}}, \sigma^*)$ to its own challenger.

It is easy to see that $\mathcal{B}$'s simulation for Game 2 is perfect. The claim immediately follows. □

This proves Lemma 3.2. □

Lemma 3.4. *Assuming the security of ISE's PKE component and the zero-knowledge property of NIZK, our CTx framework satisfies confidentiality.*

Proof. We proceed via a sequence of games.

Game 0. The real experiment for confidentiality. $\mathcal{CH}$ interacts with $\mathcal{A}$ as below.

1. Setup: $\mathcal{CH}$ runs $pp_{\text{ise}} \leftarrow \text{ISE.}\mathsf{Setup}(1^\lambda)$, $pp_{\text{nizk}} \leftarrow \text{NIZK.}\mathsf{Setup}(1^\lambda)$, and $crs \leftarrow \text{NIZK.}\mathsf{CRSGen}(pp_{\text{nizk}})$, then sends $pp = (pp_{\text{ise}}, pp_{\text{nizk}}, crs)$ to $\mathcal{A}$.
2. Pre-challenge queries: Throughout the experiment, $\mathcal{A}$ can adaptively query $\mathcal{O}_{\mathsf{regH}}$, $\mathcal{O}_{\mathsf{regC}}$, $\mathcal{O}_{\mathsf{extH}}$, $\mathcal{O}_{\mathsf{trans}}$ and $\mathcal{O}_{\mathsf{inject}}$, and $\mathcal{CH}$ answers these queries as described above.

3. Challenge: $\mathcal{A}$ picks sender pk_s^*, receiver pk_r^* and two transfer values v_0, v_1 as the challenge, subject to the restriction that $pk_s^*, pk_r^* \in T_{\text{honest}}$ and both v_0 and v_1 constitute valid transactions from pk_s^*. $\mathcal{CH}$ picks a random bit β, computes $\mathsf{ctx}^* \leftarrow \mathsf{CreateCTx}(sk_s^*, pk_s^*, pk_r^*, v_\beta)$ and sends ctx^* to $\mathcal{A}$.
4. Post-challenge queries: After receiving the challenge, $\mathcal{A}$ can continue query $\mathcal{O}_{\mathsf{regH}}$, $\mathcal{O}_{\mathsf{regC}}$, $\mathcal{O}_{\mathsf{extH}}$, $\mathcal{O}_{\mathsf{trans}}$ and $\mathcal{O}_{\mathsf{inject}}$, $\mathcal{CH}$ responds the same way as in pre-challenge stage, but with the following exceptions: (i) reject $\mathcal{A}$'s $\mathcal{O}_{\mathsf{extH}}$ query with pk_s^* or pk_r^*; (ii) reject $\mathcal{A}$'s $\mathcal{O}_{\mathsf{trans}}$ query with pk_s^* if it will make either $(\tilde{m}_s^* - v_0 - v_{\text{sum}})$ or $(\tilde{m}_s^* - v_1 - v_{\text{sum}})$ fall outside of $\mathcal{V}$. Here, v_{sum} is the accumulation of transfer amounts related to pk_s^* in post-challenge $\mathcal{O}_{\mathsf{trans}}$ queries.
5. Guess: $\mathcal{A}$ outputs a guess β' for β and wins if $\beta' = \beta$.

Game 1. Same as Game 0 except $\mathcal{CH}$ makes a random guess for index of target accounts (pk_s^*, pk_r^*) at the beginning, i.e., randomly picks two distinct indices $j, k \in [Q_{\text{honest}}]$. If $\mathcal{A}$ makes an extraction query to pk_j or pk_k in pre-challenge queries, or $\mathcal{A}$ picks $(pk_s^*, pk_r^*) \neq (pk_j, pk_k)$ in the challenge stage, $\mathcal{CH}$ aborts. Let W be the event that $\mathcal{CH}$ does not abort. It is easy to see that $\Pr[W] \geq 1/Q_{\text{honest}}(Q_{\text{honest}} - 1)$.

Conditioned on $\mathcal{CH}$ does not abort, $\mathcal{A}$'s view in Game 0 is identical to that in Game 1. Therefore, we have:

$$\Pr[S_1] \geq \Pr[S_0] \cdot \frac{1}{Q_{\text{honest}}(Q_{\text{honest}} - 1)}$$

Game 2. Same as Game 1 except that $\mathcal{CH}$ runs $(crs, \tau) \leftarrow \mathcal{S}_1(pp_{\text{nizk}})$ in the Setup stage, then generates the zero-knowledge proof in the simulation mode. More precisely, when handling transaction queries and challenge queries, $\mathcal{CH}$ runs $\mathcal{S}_2(crs, \tau, \mathsf{memo})$ to generate π_{valid} for $\mathsf{memo} \in L_{\text{valid}}$. By a direct reduction to the adaptive zero-knowledge property of the underlying NIZK, we have:

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

We now argue that no PPT adversary has non-negligible advantage in Game 2.

Claim 3.5. *Assuming the IND-CPA security (1-plaintext, 2-recipient) of the ISE's encryption component,* $\Pr[S_2] \leq \mathsf{negl}(\lambda)$ *for all PPT adversary* $\mathcal{A}$.

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ has non-negligible advantage in Game 2, we can build an adversary $\mathcal{B}$ breaks the IND-CPA security (1-plaintext, 2-recipient) of ISE's encryption component with the same advantage. Given the challenge $(pp_{\text{ise}}, pk_a, pk_b)$, $\mathcal{B}$ simulates Game 2 as follows:

1. Setup: $\mathcal{B}$ runs $pp_{\text{nizk}} \leftarrow \text{NIZK.}\mathsf{Setup}(1^\lambda)$, $\mathcal{S}_1(pp_{\text{nizk}}) \rightarrow (crs, \tau)$, sends $pp = (pp_{\text{ise}}, pp_{\text{nizk}}, crs)$ to $\mathcal{A}$. $\mathcal{B}$ randomly picks two indices $j, k \in [Q_{\text{honest}}]$.
2. Pre-challenge queries: Throughout the experiment, $\mathcal{A}$ can query $\mathcal{O}_{\mathsf{regH}}$, $\mathcal{O}_{\mathsf{regC}}$, $\mathcal{O}_{\mathsf{extH}}$, $\mathcal{O}_{\mathsf{trans}}$ and $\mathcal{O}_{\mathsf{inject}}$ adaptively. $\mathcal{B}$ answers these queries by maintaining two lists T_{honest} and T_{corrupt}, which both are initially empty.
   - $\mathcal{O}_{\mathsf{regH}}$: On the i-th query, $\mathcal{B}$ randomly picks an initial balance $\tilde{m}$ and serial number sn and proceeds as below:
     - If $i \neq j$ and k, $\mathcal{B}$ runs $\mathsf{CreateAcct}(\tilde{m}, \mathsf{sn})$ to obtain (pk, sk) and encrypted balance $\tilde{C} \leftarrow \text{ISE.}\mathsf{Enc}(pk, \tilde{m})$, then records $(pk, sk, \tilde{C}, \tilde{m}, \mathsf{sn})$ in T_{honest}.
     - If $i = j$ or k, $\mathcal{B}$ sets $pk = pk_a$ or pk_b, $\mathcal{B}$ computes $\tilde{C} \leftarrow \text{ISE.}\mathsf{Enc}(pk, \tilde{m})$, then records $(pk, \perp, \tilde{C}, \tilde{m}, \mathsf{sn})$ in T_{honest}.

     Finally, $\mathcal{B}$ returns $(pk, \tilde{C}, \mathsf{sn})$ to $\mathcal{A}$.
   - $\mathcal{O}_{\mathsf{regC}}$: $\mathcal{A}$ makes this query with a public key pk, a serial number sn and an initial encrypted balance $\tilde{C}$. $\mathcal{B}$ records $(pk, \perp, \tilde{C}, \perp, sn)$ in T_{corrupt}.
   - $\mathcal{O}_{\mathsf{extH}}$: $\mathcal{A}$ makes this query with pk in T_{honest}. If $pk = pk_j$ or pk_k, $\mathcal{B}$ aborts. Else, $\mathcal{B}$ returns the associated sk to $\mathcal{A}$, then moves the corresponding entry to T_{corrupt}.
   - $\mathcal{O}_{\mathsf{trans}}$: $\mathcal{A}$ makes a transaction query (pk_s, pk_r, v) subject to the restriction that $pk_s \in T_{\text{honest}}$. Let sn and $\tilde{C}_s$ be the serial number and encrypted balance of pk_s, $\mathcal{B}$ proceeds as below:
     (a) compute $C_s \leftarrow \text{ISE.}\mathsf{Enc}(pk_s, v)$, $C_r \leftarrow \text{ISE.}\mathsf{Enc}(pk_r, v)$;
     (b) set $\mathsf{memo} = (pk_r, C_r, pk_s, C_s)$, compute $\pi_{\text{valid}} \leftarrow \mathcal{S}_2(crs, \tau, \mathsf{memo})$;
     (c) if $pk_s \neq \{pk_j, pk_k\}$, generate a signature σ for $(\mathsf{sn}, \mathsf{memo}, \pi_{\text{valid}})$ with sk_s; else, generate such signature σ by querying the signing oracle of the underlying ISE.

$\mathcal{B}$ updates the associated account state, then returns ctx to $\mathcal{A}$.

- $\mathcal{O}_{\mathsf{inject}}$: $\mathcal{A}$ submits a confidential transaction ctx subject to the restriction that $pk_s \in T_{\text{corrupt}}$. If $\mathsf{VerifyCTx}(\mathsf{ctx}) = 1$, $\mathcal{B}$ updates the associated account state. Otherwise, $\mathcal{B}$ ignores.

3. Challenge: $\mathcal{A}$ picks sender pk_s^*, receiver pk_r^* and two transfer values v_0, v_1 as the challenge, subject to the restriction that $pk_s^*, pk_r^* \in T_{\text{honest}}$ and both v_0 and v_1 constitute a valid transaction from pk_s^*. If $(pk_s^*, pk_r^*) \neq (pk_j, pk_k)$, $\mathcal{B}$ aborts. Else, $\mathcal{B}$ submits (v_0, v_1) to its own challenger and receives back $C^* = (C_s^*, C_r^*)$, which is an encryption of v_β under (pk_s^*, pk_r^*). Let sn^* and $\tilde{C}_s^*$ be the serial number and encrypted balance of pk_s^*, $\mathcal{B}$ then prepares ctx^* as follows:
   (a) set $\mathsf{memo}^* = (\tilde{C}_s^*, pk_s^*, C_s^*, pk_r^*, C_r^*)$.
   (b) generate $\pi_{\text{valid}}^* \leftarrow \mathcal{S}_2(crs, \tau, \mathsf{memo}^*)$.
   (c) query the signing oracle of ISE to obtain a signature σ^* of $(\mathsf{sn}^*, \mathsf{memo}^*, \pi_{\text{valid}}^*)$ under sk_s^*.

   $\mathcal{B}$ updates the states of accounts pk_s^* and pk_r^*, then returns the challenge $\mathsf{ctx}^* = (\mathsf{sn}^*, \mathsf{memo}^*, \pi_{\text{valid}}^*, \sigma^*)$ to $\mathcal{A}$.
4. Post-challenge queries: After receiving the challenge, $\mathcal{A}$ can continue query to $\mathcal{O}_{\mathsf{regH}}$, $\mathcal{O}_{\mathsf{regC}}$, $\mathcal{O}_{\mathsf{extH}}$, $\mathcal{O}_{\mathsf{trans}}$ and $\mathcal{O}_{\mathsf{inject}}$, $\mathcal{B}$ responds the same way as in pre-challenge stage, but with the following exceptions: (i) reject $\mathcal{A}$'s $\mathcal{O}_{\mathsf{extH}}$ query with pk_s^* or pk_r^*; (ii) reject $\mathcal{A}$'s $\mathcal{O}_{\mathsf{trans}}$ query with pk_s^* if it will make either $(\tilde{m}_s^* - v_0 - v_{\text{sum}})$ or $(\tilde{m}_s^* - v_1 - v_{\text{sum}})$ fall outside of V. Here, v_{sum} is the accumulation of transfer amounts related to pk_s^* in post-challenge $\mathcal{O}_{\mathsf{trans}}$ queries.
5. Guess: Finally, $\mathcal{A}$ outputs its guess β' for β. $\mathcal{B}$ forwards $\mathcal{A}$'s guess to its own challenger.

It is easy to see that $\mathcal{B}$'s simulation for Game 2 is perfect. The claim immediately follows. □

This proves Lemma 3.4. □

Lemma 3.6. *Assuming the soundness of NIZK, our CTx framework satisfies soundness.*

Proof. The reduction is straightforward. We omit the details here. □

Putting Lemma 3.2, 3.4 and 3.6 together, we proves the theorem. □

4 PGC: an Efficient Instantiation

We now present an efficient realization of our CTx framework. We first instantiate ISE from our newly introduced twisted ElGamal PKE and Schnorr signature, then devise NIZK proofs from custom Sigma protocols and Bulletproof.

4.1 Instantiating ISE

We first describe twisted ElGamal encryption and recall Schnorr signature, which share the same setup and key generation algorithms. We then formally prove their integration constitutes ISE that satisfies expected joint security.

Twisted ElGamal. We propose twisted ElGamal encryption as the PKE component. The underlying reason has been elaborated in the introduction part. Formally, twisted ElGamal consists of four algorithms as below:

- $\mathsf{Setup}(1^\lambda)$: run $(\mathbb{G}, g, p) \leftarrow \mathsf{GroupGen}(1^\lambda)$, pick $h \xleftarrow{\text{R}} \mathbb{G}^*$, set $pp = (\mathbb{G}, g, h, p)$ as global public parameters. The randomness and message spaces are $\mathbb{Z}_p$.
- $\mathsf{KeyGen}(pp)$: on input pp, choose $sk \xleftarrow{\text{R}} \mathbb{Z}_p$, set $pk = g^{sk}$.
- $\mathsf{Enc}(pk, m; r)$: compute $X = pk^r$, $Y = g^r h^m$, output $C = (X, Y)$.
- $\mathsf{Dec}(sk, C)$: parse $C = (X, Y)$, compute $h^m = Y / X^{sk^{-1}}$, recover m from h^m.

Correctness and additively homomorphism are obvious. The standard IND-CPA security can be proved in the standard model based on the divisible DDH assumption. We provide the proof in Appendix A for the sake of completeness.

In this work, we require twisted ElGamal is also secure in the single plaintext, two recipient setting. A trivial solution is simply concatenating independently encryptions for two recipients, a.k.a., $\mathsf{Enc}(pk_1, m_1; r_1)||\mathsf{Enc}(pk_2, m_2; r_2)$. Security of this trivial solution is implied by the results independently proved by Bellare et al. [BBM00] and Baudron et al. [BPS00]. Kurosawa [Kur02] first showed that for standard ElGamal PKE, randomness can be reused in the single-plaintext, multi-recipient setting. Zether [BAZB19] used the same idea to make their zero-knowledge component more efficient. Thanks to the random self-reducibility of the divisible DDH problem, our twisted ElGamal PKE is also secure in the single-plaintext, multi-recipient setting even after implementing the randomness reusing trick. This not only shortens the overall transaction size, but also improves the efficiency of the associated zero-knowledge proofs. The security proof is summarized in the following theorem.

Theorem 4.1. *Twisted ElGamal is IND-CPA secure in the single-plaintext, two-recipient setting based on the divisible DDH assumption.*

Proof. We proceed via two games. Let S_i be the probability that $\mathcal{A}$ wins in Game i.

Game 0. The real IND-CPA security experiment in the single-plaintext, two-recipient setting. Challenger $\mathcal{CH}$ interacts with $\mathcal{A}$ as below:

1. Setup: $\mathcal{CH}$ runs $pp \leftarrow \mathsf{Setup}(1^\lambda)$, then runs $\mathsf{KeyGen}(pp)$ twice independently to obtain $(pk_1 = g^{sk_1}, sk_1)$ and $(pk_2 = g^{sk_2}, sk_2)$, then sends (pp, pk_1, pk_2) to $\mathcal{A}$.
2. Challenge: $\mathcal{A}$ submits m_0, m_1 to $\mathcal{CH}$ as the target messages. $\mathcal{CH}$ picks a random bit β and fresh randomness r, computes $X_1 = pk_1^r$, $X_2 = pk_2^r$, $Y = g^r h^{m_\beta}$, sends $C = (X_1, X_2, Y)$ to $\mathcal{A}$.
3. Guess: $\mathcal{A}$ outputs its guess β' for β and wins if $\beta' = \beta$.

According to the definition of Game 0, we have:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr[S_0] - 1/2.$$

Game 1. Same as Game 0 except $\mathcal{CH}$ generates the challenge ciphertext in a different way:

2. Challenge: $\mathcal{A}$ submits m_0, m_1 to $\mathcal{CH}$ as the target messages. $\mathcal{CH}$ picks a random bit β and two independent randomness r and s, computes $X_1 = pk_1^r$, $X_2 = pk_2^r$, $Y = g^s h^{m_\beta}$, sends $C = (X_1, X_2, Y)$ to $\mathcal{A}$.

In Game 1, the distribution of C is independent of β, thus the following holds even for unbounded $\mathcal{A}$:

$$\Pr[S_1] = 1/2$$

It remains to prove $\Pr[S_1]$ and $\Pr[S_0]$ are negligibly close. We prove this by showing if not so, we can build an adversary $\mathcal{B}$ breaks the divisible DDH assumption with the same advantage. Let the public parameter be $(\mathbb{G}, g, p)$, given the divisible DDH challenge instance (g, g^a, g^b, g^c), $\mathcal{B}$ is asked to decide if it is a divisible DDH tuple or a random tuple. To do so, $\mathcal{B}$ interacts with $\mathcal{A}$ by simulating $\mathcal{A}$'s challenger in the following IND-CPA experiment.

1. Setup: $\mathcal{B}$ picks $t \xleftarrow{R} \mathbb{Z}_p$, picks $h \xleftarrow{R} \mathbb{G}^*$, sends $pp = (\mathbb{G}, g, h, p)$ and $pk_1 = g^b$ and $pk_2 = g^{bt}$ to $\mathcal{A}$. Here, b and bt serve as sk_1 and sk_2, which are unknown to $\mathcal{B}$.
2. Challenge: $\mathcal{A}$ submits m_0, m_1 to $\mathcal{B}$. $\mathcal{B}$ picks a random bit β and sets $X_1 = g^a$, $X_2 = g^{at}$, $Y = g^c h^{m_\beta}$, sends $C = (X_1, X_2, Y)$ to $\mathcal{A}$.
3. Guess: $\mathcal{A}$ outputs a guess β' for β. $\mathcal{B}$ outputs "1" if $\beta' = \beta$ and "0" otherwise.

If (g, g^a, g^b, g^c) is a divisible DDH tuple, $\mathcal{B}$ simulates Game 0 perfectly (with randomness $c = a/b$). Else, $\mathcal{B}$ simulates Game 1 perfectly (with two independent randomness a/b and c). Thereby, we have $\mathsf{Adv}_{\mathcal{B}} = |\Pr[S_0] - \Pr[S_1]|$, which is negligible in λ by the divisible DDH assumption.

Putting all the above together, Theorem 4.1 follows. □

Schnorr Signature. We choose Schnorr signature [Sch91] as the signature component. The choice is out of the following reasons. First, the setup and key generation algorithms of Schnorr signature are almost identical to those of twisted ElGamal. Second, the signing procedure of Schnorr signature

is irrelevant to the decryption procedure of twisted ElGamal. This suggests that we are able to safely implement key reuse strategy to build ISE, as we will rigorously prove later.

Last but not the least, Schnorr signature is efficient and can be easily adapted to multi-signature scheme, which is particularly useful in cryptocurrencies setting [BN06], e.g. shrinking the size of the ledger [BDN18].

For the sake of completeness, we recall the Schnorr signature and sketch its security proof as below.

- $\mathsf{Setup}(1^\lambda)$: run $(\mathbb{G}, g, p) \leftarrow \mathsf{GroupGen}(1^\lambda)$, pick a cryptographic hash function $\mathsf{H} : M \times \mathbb{G} \to \mathbb{Z}_p$, where M denotes the message space. Finally, output $pp = (\mathbb{G}, g, p, \mathsf{H})$ as global public parameters.
- $\mathsf{KeyGen}(pp)$: on input pp, choose $sk \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, set $pk = g^{sk}$.
- $\mathsf{Sign}(sk, m)$: pick $r \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, set $A = g^r$, compute $e = \mathsf{H}(m, A)$, $z = r + sk \cdot e \bmod p$, output $\sigma = (A, z)$.
- $\mathsf{Verify}(pk, \sigma, m)$: parse $\sigma = (A, z)$, compute $e = \mathsf{H}(m, A)$, output "1" if $g^z = A \cdot pk^e$ and "0" otherwise.

Theorem 4.2 ([PS00]). *Assume* H *is a random oracle, Schnorr signature is EUF-CMA secure based on the discrete logarithm assumption.*

We sketch the security proof due to Pointcheval and Stern [PS00] as follows. On a high level, the reduction $\mathcal{R}$ (from DLP to the EUF-CMA security of Schnorr signature) works as follows. Given the DLP challenge instance (g, g^a), $\mathcal{R}$ embeds g^a into the public key (i.e., sets $pk = g^a$), then simulates the signing oracle by programming the random oracle H without knowledge of a and uses the oracle-replay attack to obtain two different forgeries that share the signing randomness ($A = g^r$) from the forger. This enables $\mathcal{R}$ to solve for a.

ISE from Schnorr signature and twisted ElGamal encryption. By merging the Setup and KeyGen algorithms of twisted ElGamal encryption and Schnorr signature, we obtain the ISE scheme, whose joint security is captured by the following theorem.

Theorem 4.3. *The obtained ISE scheme is jointly secure if the twisted ElGamal is IND-CPA secure (single-plaintext, two-recipient) and the Schnorr signature is EUF-CMA secure.*

Proof. As we analyzed before, in the context of our CTx construction we only require the PKE component of ISE satisfying passive security, thus the security of the signature component is implied by its standalone EUF-CMA security. In what follows, we prove the security for the PKE component, namely IND-CPA security (single plaintext, two recipient) in the presence of a signing oracle for Schnorr signature. We proceed via a sequence of games.

Game 0. The real security experiment for ISE's PKE component (cf. Definition 2.4). Challenger $\mathcal{CH}$ interacts with $\mathcal{A}$ as below:

1. Setup: $\mathcal{CH}$ runs $pp \leftarrow \mathsf{Setup}(1^\lambda)$, runs $\mathsf{KeyGen}(pp)$ twice independently to obtain $(pk_1 = g^{sk_1}, sk_1)$ and $(pk_2 = g^{sk_2}, sk_2)$, then sends (pp, pk_1, pk_2) to $\mathcal{A}$.
2. Queries: Throughout the experiment, $\mathcal{A}$ can make hash queries and signing queries. $\mathcal{CH}$ emulates random oracle by maintaining a list T_{hash} using standard lazy method. T_{hash} is initially empty, which stores triples $(\cdot, \cdot, \cdot) \in M \times \mathbb{G} \times \mathbb{Z}_p$, an entry (m, A, e) indicates that $e := \mathsf{H}(m, A)$. Concretely, $\mathcal{CH}$ responds $\mathcal{A}$'s queries as below:
   - Hash queries: On query (m, A), if there is an entry (m, A, e) in T_{hash}, $\mathcal{CH}$ returns e. Else, $\mathcal{CH}$ picks $e \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and inserts (m, A, e) in T_{hash}, then returns e.
   - Signing queries for sk_1 or sk_2: $\mathcal{CH}$ responds with corresponding secret key. On query (b, m) where $b \in \{0, 1\}$ indicating which secret key is used, $\mathcal{CH}$ picks a fresh randomness $r \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, sets $A = g^r$, obtains $e = \mathsf{H}(m, A)$ by accessing the hash oracle, then computes $z = r + sk_b \cdot e \bmod p$, returns $\sigma = (A, z)$.
3. Challenge: $\mathcal{A}$ submits m_0, m_1 to $\mathcal{CH}$ as the target messages. $\mathcal{CH}$ picks a random bit β and fresh randomness r, computes $X_1 = pk_1^r$, $X_2 = pk_2^r$, $Y = g^r h^{m_\beta}$, sends $C = (X_1, X_2, Y)$ to $\mathcal{A}$.
4. Guess: $\mathcal{A}$ outputs its guess β' for β and wins if $\beta' = \beta$.

According to the definition of Game 0, we have:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr[S_0] - 1/2.$$

Game 1. The same as Game 0 except that $\mathcal{CH}$ simulates signing oracle by programming random oracle H, rather than using the real secret keys. To do so, $\mathcal{CH}$ emulates random oracle by maintaining a list T^*_{hash}, which is initially empty. T^*_{hash} stores triples $(\cdot,\cdot,\cdot,\cdot,\cdot) \in M \times \mathbb{G} \times \mathbb{Z}_p \times \mathbb{Z}_p \times \{0,1\}$, an entry $(m, A, e, *, *)$ indicates that $e := \mathsf{H}(m, A)$, where $*$ is the wildcard.

2. Queries: $\mathcal{CH}$ handles hash queries and signing queries as below.
   - Hash queries: On query (m, A), if there is an entry $(m, A, e, *, *)$ in T^*_{hash}, $\mathcal{CH}$ returns e. Else, $\mathcal{CH}$ picks $e \xleftarrow{\text{R}} \mathbb{Z}_p$, inserts $(m, A, e, \perp, \perp)$ in T^*_{hash}, then returns e.
   - Signing queries for sk_1 or sk_2: $\mathcal{CH}$ responds with corresponding secret key. On query (b, m), if there is an entry (m, A, e, z, b) (with the same value of b and m) in the T_{hash} list, $\mathcal{CH}$ simply returns $\sigma = (A, z)$. Otherwise, $\mathcal{CH}$ picks $z, e \xleftarrow{\text{R}} \mathbb{Z}_p$, sets $A = g^z/pk_b^e$, if there is no entry indexed by (m, A) in T_{hash}, then inserts (m, A, e, z, b) in T_{hash} and returns $\sigma = (A, z)$. Else, $\mathcal{CH}$ aborts to avoid possible inconsistent programming.

Denote the event that $\mathcal{CH}$ aborts in Game 1 by E. Conditioned on E does not occur, $\mathcal{A}$'s view in Game 0 and Game 1 are identical. This follows from the fact that $\mathcal{CH}$ perfectly mimic the hash oracle and signing oracle. Let Q_{hash} and Q_{sign} be the maximum number of hash queries and signing queries that $\mathcal{A}$ makes during security experiment. By the union bound, we conclude that $\Pr[E] \leq (Q_{\text{hash}}Q_{\text{sign}})/p$, which is negligible in λ. In summary, we have:

$$|\Pr[S_1] - \Pr[S_0]| \leq \Pr[E] \leq \mathsf{negl}(\lambda)$$

It remains to bound $\Pr[S_1]$. We have the following claim.

Claim 4.4. *Assume the IND-CPA security (single-message, two-recipient) of twisted ElGamal PKE, $\Pr[S_1]$ is negligible in λ for any PPT adversary $\mathcal{A}$.*

Proof. We prove this claim by showing that if there exists a PPT adversary $\mathcal{A}$ has non-negligible advantage in Game 1, we can build a PPT adversary $\mathcal{B}$ that breaks the IND-CPA security (single-message, two-recipient) of twisted ElGamal PKE with the same advantage. Note that according to the definition of Game 1, $\mathcal{CH}$ can simulate the signing oracles without using the secret keys. Thereby, given (pp, pk_1, pk_2), $\mathcal{B}$ can perfectly simulate Game 1 by forwarding the encryption challenge to its own challenger. This proves the claim. □

Putting all the above together, Theorem 4.3 immediately follows. □

4.2 Instantiating NIZK

Now, we design efficient NIZK for L_{valid} and L_{correct} respectively.

As stated in Section 3.1, L_{valid} can be decomposed as $L_{\text{equal}} \wedge L_{\text{right}} \wedge L_{\text{enough}}$. Let Π_{equal}, Π_{right}, Π_{enough} be NIZK proof systems for L_{equal}, L_{right}, L_{enough} respectively, and let $\Pi_{\text{valid}} := \Pi_{\text{equal}} \circ \Pi_{\text{right}} \circ \Pi_{\text{enough}}$, where $\circ$ denotes sequential composition.[9] By the property of NIZK for conjunctive statements [Gol06], Π_{valid} is an NIZK proof system for L_{valid}. Now, the task breaks down to design Π_{equal}, Π_{right}, Π_{enough} and Π_{correct} separately. We describe them as below.

4.2.1 NIZK for L_{equal}

According to our CTx framework and the definition of twisted ElGamal, L_{equal} can be written as:

$$\{(pk_1, X_1, Y_1, pk_2, X_2, Y_2) \mid \exists r_1, r_2, v \text{ s.t. } X_i = pk_i^{r_i} \wedge Y_i = g^{r_i}h^v \text{ for } i = 1, 2\}.$$

As analyzed before, for twisted ElGamal randomness can be safely reused in the single-ciphertext multi-recipient setting. L_{equal} can thus be simplified to:

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

[9] In the non-interactive setting, there is no distinction between sequential and parallel composition.

Sigma protocol for L_{equal}. To obtain an NIZK for L_{equal}, we first design a Sigma protocol $\Sigma_{\text{equal}} = (\mathsf{Setup}, P, V)$ for L_{equal}. The Setup algorithm is same as that of twisted ElGamal. On statement $(pk_1, pk_2, X_1, X_2, Y)$, P and V interact as below:

1. P picks $a, b \xleftarrow{R} \mathbb{Z}_p$, sends $A_1 = pk_1^a$, $A_2 = pk_2^a$, $B = g^a h^b$ to V.
2. V picks $e \xleftarrow{R} \mathbb{Z}_p$ and sends it to P as the challenge.
3. P computes $z_1 = a + er$, $z_2 = b + ev$ using witness $w = (r, v)$, then sends (z_1, z_2) to V. V accepts iff the following three equations hold simultaneously:

$$pk_1^{z_1} = A_1 X_1^e \quad (1)$$
$$pk_2^{z_1} = A_2 X_2^e \quad (2)$$
$$g^{z_1} h^{z_2} = B Y^e \quad (3)$$

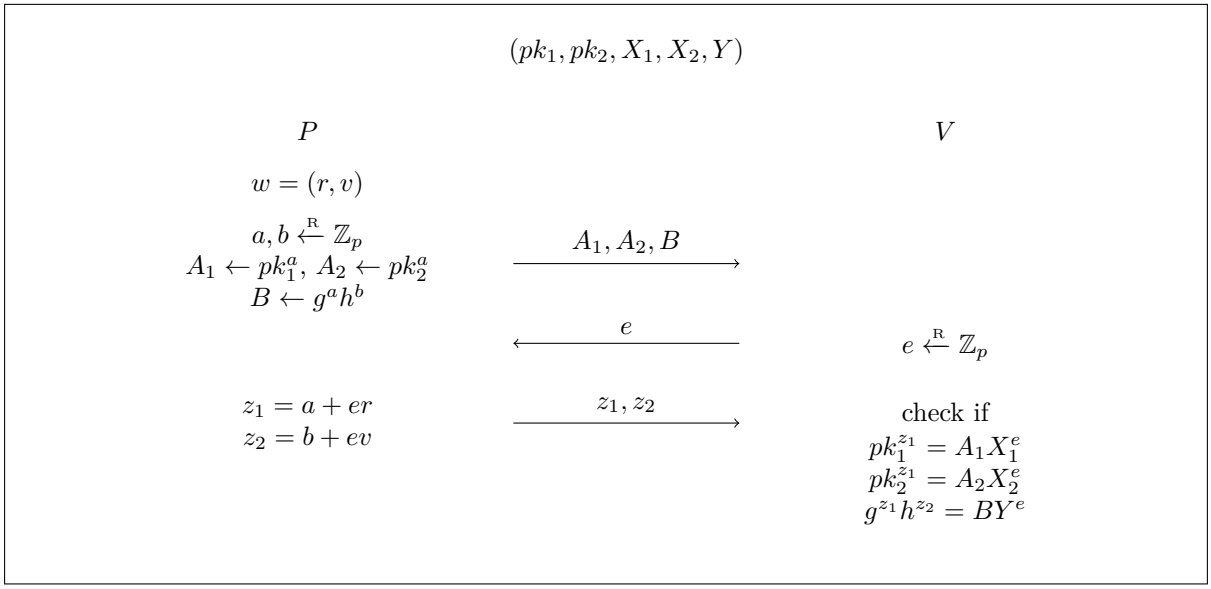

Figure 1: Σ_{equal} for L_{equal}: proof of knowledge of two twisted ElGamal ciphertexts encrypting the same value under different public keys

Lemma 4.5. *Σ_{equal} is a public-coin SHVZK proof of knowledge for L_{equal}.*

Proof. We prove that all three properties required for Sigma protocol are met.

Perfect completeness is obvious from simple calculation.

To show special soundness, fix the initial message (A_1, A_2, B), suppose there are two accepting transcripts $(e, z = (z_1, z_2))$ and $(e', z' = (z_1', z_2'))$ with $e \neq e'$, the witness can be extracted as below. From either (1) or (2), we have $z_1 = a + er$ and $z_1' = a + e'r$, which implies $r = (z_1 - z_1')/(e - e')$. Further from (3), we have $z_2 = b + ev$ and $z_2' = b + e'v$, which imply $v = (z_2 - z_2')/(e - e')$.

To show special HVZK, for a fixed challenge e, the simulator $\mathcal{S}$ works as below: picks $z_1, z_2 \xleftarrow{R} \mathbb{Z}_p$, computes $A_1 = pk_1^{z_1}/X_1^e$, $A_2 = pk_2^{z_1}/X_2^e$, $B = g^{z_1} h^{z_2}/Y^e$. Clearly, $(A_1, A_2, B, e, z_1, z_2)$ is an accepting transcript, and it is distributed as in the real protocol.

This proves Lemma 4.5. □

Applying Fiat-Shamir transform to Σ_{equal}, we obtain Π_{equal}, which is actually an NIZKPoK for L_{equal}.

4.2.2 NIZK for L_{right}

According to our CTx framework and the definition of twisted ElGamal, we need to prove $(pk_1, X_1, Y) \in L_{\text{right}}$, where L_{right} is defined as:

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

For ease of analysis, we additionally define L_{enc} and L_{range} as below:

$$L_{\text{enc}} = \{(pk, X, Y) \mid \exists r, v \text{ s.t. } X = pk^r \wedge Y = g^r h^v\}$$
$$L_{\text{range}} = \{Y \mid \exists r, v \text{ s.t. } Y = g^r h^v \wedge v \in \mathcal{V}\}$$

It is straightforward to verify that $L_{\text{right}} \subset L_{\text{enc}} \wedge L_{\text{range}}$. Observing that each instance $(pk, X, Y) \in L_{\text{right}}$ has a unique witness, while the last component Y can be viewed as a Pedersen commitment of value v under commitment key (g, h), whose discrete logarithm $\log_g h$ is unknown to any users. To prove $(pk, X, Y) \in L_{\text{right}}$, we first prove $(pk, X, Y) \in L_{\text{enc}}$ with witness (r, v) via a Sigma protocol $\Sigma_{\text{enc}} = (\mathsf{Setup}, P_1, V_1)$, then prove $Y \in L_{\text{range}}$ with witness (r, v) via a Bulletproof $\Lambda_{\text{bullet}} = (\mathsf{Setup}, P_2, V_2)$.

Sigma protocol for L_{enc}. We begin with a Sigma protocol $\Sigma_{\text{enc}} = (\mathsf{Setup}, P, V)$ for L_{enc}. The Setup algorithm is same as that of twisted ElGamal. On statement $x = (pk, X, Y)$, P and V interact as below:

1. P picks $a, b \xleftarrow{R} \mathbb{Z}_p$, sends $A = pk^a$ and $B = g^a h^b$ to V.
2. V picks $e \xleftarrow{R} \mathbb{Z}_p$ and sends it to P as the challenge.
3. P computes $z_1 = a + er$, $z_2 = b + ev$ using witness $w = (r, v)$, then sends (z_1, z_2) to V. V accepts iff the following two equations hold simultaneously:

$$pk^{z_1} = AX^e \tag{4}$$
$$g^{z_1} h^{z_2} = BY^e \tag{5}$$

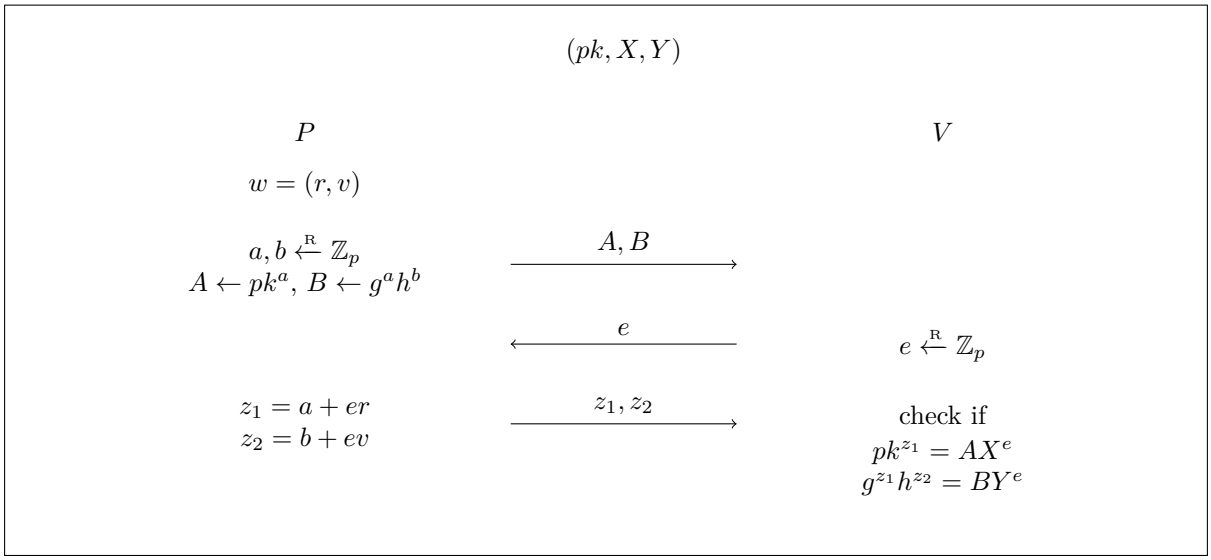

Figure 2: Σ_{enc} for L_{enc}: proof of knowledge of twisted ElGamal ciphertext

Lemma 4.6. *Σ_{enc} is a public-coin SHVZK proof of knowledge for L_{enc}.*

Proof. We prove that all three properties required for Sigma protocol are met.

Perfect completeness is obvious from simple calculation.

To show special soundness, fix the initial message (A, B), suppose there are two accepting transcripts $(e, z = (z_1, z_2))$ and $(e', z' = (z'_1, z'_2))$ with $e \neq e'$, we can extract the witness as below. From (4), we have $z_1 = a + er$ and $z'_1 = a + e'r$, which implies $r = (z_1 - z'_1)/(e - e')$. Further from (5), we have $z_2 = b + ev$ and $z'_2 = b + e'v$, which imply $v = (z_2 - z'_2)/(e - e')$.

To show special HVZK, for a fixed challenge e, the simulator $\mathcal{S}$ works as below: picks $z_1, z_2 \xleftarrow{R} \mathbb{Z}_p$, computes $A = pk^{z_1}/X^e$, $B = g^{z_1} h^{z_2}/Y^e$. Clearly, (A, B, e, z_1, z_2) is an accepting transcript, and it is distributed as in the real protocol.

This proves Lemma 4.6. □

Bulletproofs for L_{range}. We employ the logarithmic size Bulletproof $\Lambda_{\text{bullet}} = (\mathsf{Setup}, P, V)$ to prove L_{range}. To avoid repetition, we refer to [BBB+18, Section 4.2] for the details of the interaction between P and V.

Lemma 4.7. *[BBB+18, Theorem 3] Assuming the hardness of discrete logarithm problem, Λ_{bullet} is a public-coin SHVZK argument of knowledge for L_{range}.*

Sequential composition. Let $\Gamma_{\text{right}} = \Sigma_{\text{enc}} \circ \Lambda_{\text{bullet}}$ be the sequential composition of Σ_{enc} and Λ_{bullet}. The Setup algorithm of Γ_{right} is a merge of that of Σ_{enc} and Λ_{bullet}. For range $\mathcal{V} = [0, 2^\ell - 1]$, it first generates a group $\mathbb{G}$ of prime order p together with two random generators g and h, then picks independent generators $\mathbf{g}, \mathbf{h} \in \mathbb{G}^\ell$. Let $P_1 = \Sigma_{\text{enc}}.P$, $V_1 = \Sigma_{\text{enc}}.V$, $P_2 = \Lambda_{\text{bullet}}.P$, $V_2 = \Lambda_{\text{bullet}}.V$. We have $\Gamma_{\text{right}}.P = (P_1, P_2)$, $\Gamma_{\text{right}}.V = (V_1, V_2)$.

Lemma 4.8. *Assuming the discrete logarithm assumption, $\Gamma_{\text{right}} = (\mathsf{Setup}, P, V)$ is a public-coin SHVZK argument of knowledge for L_{right}.*

Proof. The completeness and zero-knowledge properties of Γ_{right} follow from that of Σ_{enc} and Λ_{bullet}.

In order to prove argument of knowledge, it suffices to construct a PPT witness extraction algorithm E. Let E_1 and E_2 be the witness extraction algorithm for Σ_{enc} and Λ_{bullet} respectively. Note that Γ_{right} is a sequential composition of Σ_{enc} and Λ_{bullet}, thus an $(n_0, n_1, \ldots, n_k)$-tree of accepting transcripts for Γ_{right} is a composition of an n_0-subtree of accepting transcripts for Σ_{enc} (where $n_0 = 2$) and an $(n_1, \ldots, n_k)$-subtree of accepting transcripts for Λ_{bullet} (see [BBB+18] for the concrete values of $n_1, \ldots, n_k$).

E first runs E_1 on n_0-subtree of accepting transcripts to extract $w = (r, v)$, then runs E_2 on $(n_1, \ldots, n_k)$-subtree of accepting transcripts to extract $w' = (r', v')$. Finally, E outputs w if $w = w'$ and aborts otherwise. Note that $\Pi_{i=0}^{k} n_i$ is still bounded by a polynomial of λ.

According to Lemma 4.6, E_1 outputs a witness of statement $(pk, X, Y) \in L_{\text{enc}}$ with probability 1. According to Lemma 4.7, E_2 outputs a witness of statement $Y \in L_{\text{range}}$ with overwhelming probability assuming the hardness of DLP. We argue that E_2's output $w' = (r', v')$ equals E_1's output $w = (r, v)$ with overwhelming probability, since otherwise the relation $g^v h^r = Y = g^{v'} h^{r'}$ immediately yields a solution $\log_g h$ to the DLP instance (g, h). We thus conclude that E outputs a witness for $x = (pk, X, Y) \in L_{\text{right}}$ with overwhelming probability. Thus, Γ_{right} satisfies computational witness-extended emulation.

Putting all the above together, Lemma 4.8 immediately follows. □

Applying Fiat-Shamir transform to Γ_{right}, we obtain Π_{right}, which is an NIZKAoK for L_{right}.

4.2.3 NIZK for L_{enough}

According to our CTx framework, we need to prove $(pk_1, \tilde{C}_1, C_1) \in L_{\text{enough}}$, where L_{enough} is defined as:

$$\{(pk, \tilde{C}, C) \mid \exists sk \text{ s.t. } (pk, sk) \in \mathsf{R}_{\text{key}} \wedge \text{ISE.}\mathsf{Dec}(sk, \tilde{C} - C) \in \mathcal{V}\}.$$

In the above, $\tilde{C} = (\tilde{X} = pk^{\tilde{r}}, \tilde{Y} = g^{\tilde{r}} h^{\tilde{m}})$ is the encryption of current balance $\tilde{m}$ of account pk under randomness $\tilde{r}$, $C = (X = pk^r, Y = g^r h^v)$ encrypts the transfer amount v under randomness r. Let $C' = (X' = pk^{r'}, Y' = g^{r'} h^{m'}) = \tilde{C} - C$, L_{enough} can be rewritten as:

$$\{(pk, C') \mid \exists r', m' \text{ s.t. } C' = \text{ISE.}\mathsf{Enc}(pk, m'; r') \wedge m' \in \mathcal{V}\}.$$

By additive homomorphism of twisted ElGamal, we have $r' = \tilde{r} - r$, $m' = \tilde{m} - v$. It seems that we can prove L_{enough} using the same protocol for L_{right}. However, while the sender (playing the role of prover) learns $\tilde{m}$ (by decrypting $\tilde{C}$ with sk), v and r, it generally does not know the randomness $\tilde{r}$. This is because $\tilde{C}$ is the sum of all the incoming and outgoing transactions of pk, whereas the randomness behind incoming transactions is unknown. The consequence is that r' (the first part of witness) is unknown, which renders us unable to directly invoke the Bulletproof on instance Y'.

Our trick is encrypting the value $m' = (\tilde{m} - v)$ under a fresh randomness r^* to obtain a new ciphertext $C^* = (X^*, Y^*)$, where $X^* = pk^{r^*}$, $Y^* = g^{r^*} h^{m'}$. C^* could be viewed as a refreshment of C'. In a nutshell, we express L_{enough} as $L_{\text{equal}} \wedge L_{\text{right}}$, a.k.a. we have:

$$(pk, C') \in L_{\text{enough}} \iff (pk, C', C^*) \in L_{\text{equal}} \wedge (pk, C^*) \in L_{\text{right}}$$

To prove C' and C^* encrypting the same value under public key pk, we cannot simply use a Sigma protocol like Protocol 4.2.1, in which the prover uses the message and randomness as witness, as the randomness r' behind C' is typically unknown. Luckily, we are able to prove this more efficiently by using the secret key as witness. Generally, L_{equal} can be written as:

$$\{(pk, C_1, C_2) \mid \exists sk \text{ s.t. } (pk, sk) \in \mathsf{R}_{\text{key}} \wedge \text{ISE.}\mathsf{Dec}(sk, C_1) = \text{ISE.}\mathsf{Dec}(sk, C_2)\}$$

When instantiated with twisted ElGamal, $C_1 = (X_1 = pk^{r_1}, Y_1 = g^{r_1}h^m)$ and $C_2 = (X_2 = pk^{r_2}, Y_2 = g^{r_2}h^m)$ are ciphertexts under the same public key pk, then proving membership of L_{equal} is equivalent to proving $\log_{Y_1/Y_2} X_1/X_2$ equals $\log_g pk$. This can be efficiently done by utilizing the Sigma protocol $\Sigma_{\text{ddh}} = (\mathsf{Setup}, P, V)$ for discrete logarithm equality due to Chaum and Pedersen [CP92]. For the sake of completeness, we describe it as below.

The Setup algorithm generates a cyclic group $\mathbb{G}$ of prime order p. Define the language L_{ddh} as below:

$$L_{\text{ddh}} = \{(g_1, h_1, g_2, h_2) \mid \exists w \in \mathbb{Z}_p \text{ s.t. } \log_{g_1} h_1 = w = \log_{g_2} h_2\}$$

It is easy to see that $\log_{g_1} h_1 = \log_{g_2} h_2$ iff (g_1, h_1, g_2, h_2) constitutes a DDH tuple. On statement $x = (g_1, h_1, g_2, h_2)$, P interacts with V as below:

1. P picks $a \xleftarrow{\text{R}} \mathbb{Z}_p$, sends $A_1 = g_1^a$, $A_2 = g_2^a$ to V;
2. V picks $e \xleftarrow{\text{R}} \mathbb{Z}_p$ and sends it to P as the challenge;
3. P computes $z = a + we$ with witness w and sends it to V. V accepts iff:

$$g_1^z = A_1 h_1^e \wedge g_2^z = A_2 h_2^e$$

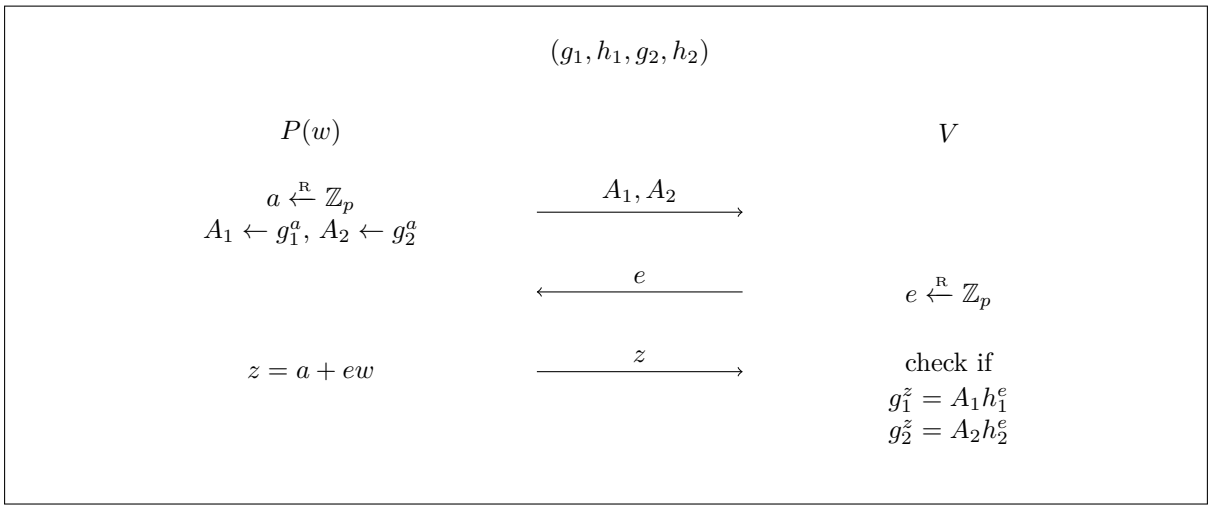

Figure 3: Σ_{ddh} for L_{ddh}: proof of knowledge of discrete logarithm equality/DDH tuple

Lemma 4.9 ([CP92])**.** Σ_{ddh} *is a public-coin SHVZK proof of knowledge for* L_{ddh}.

Applying Fiat-Shamir transform to Σ_{ddh}, we obtain an NIZKPoK Π_{ddh} for L_{ddh}. We then prove $(pk, C^* = (X^*, Y^*)) \in L_{\text{right}}$ using the NIZKPoK Π_{right} as we described before. Let $\Pi_{\text{enough}} = \Pi_{\text{ddh}} \circ \Pi_{\text{right}}$, we conclude that Π_{enough} is an NIZKPoK for L_{enough} by the properties of AND-proofs.

Putting all the sub-protocols described above, we obtain $\Pi_{\text{valid}} = \Pi_{\text{equal}} \circ \Pi_{\text{right}} \circ \Pi_{\text{enough}}$.

Theorem 4.10. Π_{valid} *is an NIZKAoK for* $L_{\text{valid}} = L_{\text{equal}} \wedge L_{\text{right}} \wedge L_{\text{enough}}$.

Proof. The proof of this theorem follows from the properties of AND-proofs. □

4.2.4 NIZK for L_{correct}

According to our CTx framework and the definition of twisted ElGamal, L_{correct} can be written as:

$$\{(pk, C = (X, Y), v) \mid \exists sk \text{ s.t. } X = (Y/h^v)^{sk} \wedge pk = g^{sk}\}$$

The above language is equivalent to $(Y/g^v, X, g, pk) \in L_{\text{ddh}}$, which in turn can be efficiently proved via Π_{ddh} for discrete logarithm equality, as already described in Protocol 4.2.3.

5 Further Discussions

5.1 Transparent Setup

Benefit from our dedicate design, the global parameters of PGC are exactly the public parameters of Bulletproofs, a.k.a. simply random generators $(g, h, \mathbf{g}, \mathbf{h})$ without special structure. We can thus use hash function to dismiss trusted setup. We refer to [BBB+18] for more details on this point.

5.2 Necessity of Composing Sigma protocol with Bulletproof

We prove the membership of L_{right} by sequentially composing a Sigma protocol for L_{enc} and a Bulletproof for L_{range}. One may wonder if we can only use the Bulletproof for L_{range} to fulfill this task. Actually, that will be problematic. Consider a malicious prover P^* generates ciphertext dishonestly as below: picks random r, r' from $\mathbb{Z}_q$ and $v' \in \mathcal{V}$, sets $X = pk^r$, $Y = g^{r'} h^{v'}$. Using r' and v' as witness, it can generate a valid Bulletproof for $Y \in L_{\text{range}}$. However, the real message v encrypted by (X, Y) fall outside the range $\mathcal{V}$ with overwhelming probability, and thus $(pk, X, Y) \notin L_{\text{enough}}$. This compromises proof of knowledge. It is also instructive to examine where the security proof will break. Let (pk, X, Y) be the statement and (r, v) be the corresponding randomness-message pair. To prove witness-extended emulation via general forking lemma, we hope to argue that the output of Bulletproof's extractor (r', v') equals (r, v) overwhelmingly based on the collision resistance of hash function $\mathsf{H}(x, y) = g^x h^y$, which in turn is implied the hardness of DLP related to (g, h). However, the forking lemma is required to hold for any fixed statement and thus there is no way to embed the collision instance.

5.3 Ciphertext Refreshing vs. Key Switching

In order to safely use Bulletproof to prove the membership of L_{enough}, we refresh $C' = (pk^{r'}, g^{r'} h^{m'})$ to $C^* = (pk^{r^*}, g^{r^*} h^{m'})$, then directly invoking the Bulletproof on the second component of C^* with witness (r^*, m'). To ensure the soundness of the overall protocol, we also need another protocol to prove that C' and C^* encrypt the same message. One may attempt to use "key switching" trick to avoid the overhead of refreshing. Note that $pk = g^{sk}$, thus $g^{r'} h^{m'}$ can be rewritten as $(pk^{r'})^{sk^{-1}} h^{m'}$. It seems that now we are able to invoke Bulletproof on $(pk^{r'})^{sk^{-1}} h^{m'}$ by interpreting $(pk^{r'}, h)$ as commitment key of Pedersen commitment and using (sk^{-1}, m') as the witness. The soundness of the protocol is based on the intuition that the prover does not know the DL relation between $(pk^{r'}, h)$. Is it possible for the prover to rigorously prove this intuition? It is well-known that one can generate a proof of knowledge of discrete logarithm using the Schnorr's protocol. But, how can the prover give a proof of no-knowledge? Interestingly, the prover can prove that he indeed does not know $\log_h pk^{r'}$ by proving his knowledge of $\log_g pk^{r'}$. This makes sense because otherwise he can solve the DL relation between (g, h), which contradicts to the assumption. Inevitably, an honest prover falls into a deadlocked setting — he needs to know r' to create a proof of knowledge of $\log_g pk^{r'}$, which is exactly the obstacle he want to overcome at the beginning.

6 Optimizations

6.1 Faster Decryption for Twisted ElGamal

As in standard exponential ElGamal, to ensure additive homomorphism on message space $\mathbb{Z}_p$, message in twisted ElGamal is also encoded to the exponent. The makes decryption inefficient even with knowledge

of secret key. This seems problematic because the exact balance is required to be known when generating the range proof for enough balance. Similar problem also occurs in other confidential transaction systems [FMMO19, BAZB19].

Fortunately, this is not an issue. First, each user can learn its current balance by keeping track of the incoming and outgoing transfers. While the outgoing transfer amounts are always known to the senders themselves by default, the incoming transfer amounts are also known to the recipients in most times. This makes sense because senders typically communicate the transfer amount to their recipients off-chain in real-world applications. Now, let us consider the worst case, i.e., the recipients do not know the exact value of incoming transfer amounts. Suppose the transfer amount is restricted to the range $[0, 2^\ell - 1]$ by protocol specification. [FMMO19, BAZB19] suggested that the recipients can figure out m from g^m by brute-force enumeration with time complexity $O(2^\ell)$. Here, we recommend employing dedicated algorithms to compute a discrete logarithm in an interval more efficiently. For instance, Pollard's kangaroo method [Pol78] and Galbraith-Ruprai algorithm [GR10] run in time $O(2^{\ell/2})$ with constant memory cost, Shanks's algorithm [Sha71] runs in time $O(2^{\ell/2})$ using a table of size $O(2^{\ell/2})$, and Bernstein-Lange algorithm [BL12] runs in time $O(2^{\ell/3})$ using a table of size $O(2^{\ell/3})$. In this work, we implement the Shanks's algorithm for simplicity and trade more space for time. When $\ell = 32$, the average decryption time is less than 0.85ms by using a table of roughly 264MB. This demonstrates that Shanks's algorithm is more preferable in practice.

If larger range are needed, we can adopt the approach due to Zether [BAZB19] to represent 64-bit transfer amount by a concatenation of two 32-bit values (we remark that extra zero-knowledge proofs are needed to ensure that the values are encrypted in a correct form). The time complexity of decryption only grows linearly in the size of bit-length rather than exponentially.

6.2 More Efficient Assembly of NIZK

Towards a modular design of NIZK for L_{valid}, we break this task to designing NIZK proofs for L_{equal}, L_{right} and L_{enough} separately. Particularly, to build NIZK for L_{right} (asserting a twisted ElGamal ciphertext encrypts a message in expected range), we compose a Sigma protocol Σ_{enc} for L_{enc} and a Bulletproof Λ_{bullet} for L_{range} sequentially, then apply the Fiat-Shamir transform. The resulting Π_{right} for L_{right} can be used as a sub-protocol when proving membership in L_{enough}.

When comes to instantiation, we can assemble the sub-protocols in a more efficient manner. More precisely, let $L_{\text{legal}} = L_{\text{equal}} \wedge L_{\text{right}}$, which asserts that two twisted ElGamal ciphertexts encrypt the same message and the message lies in the right range. To prove membership in L_{legal}, one can compose Σ_{equal} and Λ_{bullet} sequentially, then obtain an NIZK Π_{legal} for L_{legal} by applying Fiat-Shamir transform to $\Sigma_{\text{equal}} \circ \Lambda_{\text{bullet}}$. Recall that the naive construction of Π_{valid} is $\Pi_{\text{equal}} \circ \Pi_{\text{right}} \circ \Pi_{\text{enough}}$. By replacing $\Pi_{\text{equal}} \circ \Pi_{\text{right}}$ with Π_{legal}, we manage to save the cost of an NIZK for L_{enc} compared to the naive construction.

Note that when producing proofs for L_{legal} and L_{enough}, we invoke Bulletproof twice independently. Benefit from the nice feature of twisted ElGamal, the two range statements can be generated and verified in batches, leading to the range proof part only growing by two group elements, rather than doubling.

6.3 Eliminate Explicit Signature

At a high level, our CTx system utilizes signature to provide authenticity and uses NIZK to provide soundness. The celebrated Fiat-Shamir transform [FS86] squashes an interactive public-coin proof into a non-interactive one by setting the verifier's challenges as the hash of the prover's history messages. Particularly, the Fiat-Shamir transform can convert a three round public-coin proof of knowledge into a signature scheme [AABN02]. To generate a signature of message m, the signer runs the PoK itself (with sk as the witness) by setting the challenge e as $\mathsf{H}(I, m)$, where H is a hash function modeled as a random oracle, I is the prover's first round message. The signature is the proof, i.e., the entire transcript.

Based on this connection between signature and ZKPs, Bünz et al. [BAZB19] suggested that instead of employing a separate signature scheme one can leverage ZKPs to provide signature functionality. However, their construction signs the entire transaction including the NIZK proof. Thus, the signing operation and signature are still explicit.

In our PGC instantiation, when building NIZK for L_{enough} we use Σ_{ddh} for L_{ddh} as a sub-protocol, which is exactly a proof of knowledge of sender's secret key sk_1. As we highlighted in the security model

of CTx, it suffices to sign only the core information, say, $(\mathsf{sn}, \mathsf{memo})$. Benefit from this relaxation, the signature component can be subsumed into the NIZK proof. More precisely, by appending $(\mathsf{sn}, \mathsf{memo})$ to the prover's first round message in Σ_{ddh}, the obtained zero-knowledge proof π_{ddh} for L_{ddh} also acts as sender's signature of $(\mathsf{sn}, \mathsf{memo})$. In this way, PGC obtains the signature functionality for free.[10]

7 Performance

We give a prototype implementation of PGC in C++, and collect the benchmarks on a MacBook Pro with an Intel i7-7500U CPU (2.5GHz using a single thread) and 8GB of RAM. For demo purpose only, we do not explore any optimizations.[11] The experimental results are described in the table below. (We do not compare with Zether [BAZB19] because: (i) Zether does not specify the signature and well-formedness proof π_{transfer} in details, thus we are unable to figure out its transaction size in asymptotic sense; (ii) Zether does not report implementation as a standalone cryptocurrency, so it is difficult to determine its concrete running times for their scheme.)

	ctx size		ctx cost (ms)	
	big-$\mathcal{O}$	bytes	generation	verify
PGC	$(2\log(\ell)+20)\lvert\mathbb{G}\rvert + 10\lvert\mathbb{Z}_p\rvert$	1310	156.79	77.6

Table 1: The computation and communication complexity of PGC. Here we set the maximum transaction value as 2^ℓ, where $\ell = 32$. We choose elliptic curve secp256k1 which has 128 bit security. The elliptic points are expressed in their compressed form. Each $\mathbb{G}$ element can be stored as 33 bytes, and $\mathbb{Z}_p$ element can be stored as 32 bytes.

Acknowledgments

We thank Benny Pinkas and Jonathan Bootle for clarifications on Sigma protocols and Bulletproofs in the early stages of this research. We thank Yi Deng and Shunli Ma for helpful comments. We particularly thank Shuai Han for many enlightening discussions.

References

[AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 418–433. Springer, 2002.

[BAZB19] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. Cryptology ePrint Archive, Report 2019/191, 2019. https://eprint.iacr.org/2019/191.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy, SP 2018*, pages 315–334. IEEE Computer Society, 2018.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2000.

[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 *Advances in Cryptology - EUROCRYPT 2016*, volume 9666 of *Lecture Notes in Computer Science*, pages 327–357. Springer, 2016.

[10] A subtlety here is that our approach does not enable us to sign the whole information of $(\mathsf{sn}, \mathsf{memo}, \pi_{\mathrm{valid}})$ implicitly, since there seems no way to append π_{valid} (containing π_{ddh}) to the first round message of Σ_{ddh} before generating it. Nevertheless, as we discussed before, signing $(\mathsf{sn}, \mathsf{memo})$ is sufficient for authenticity.

[11] We expect at least 5× speedup after applying multi-exponentiation and batch verification.

[BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In *Advances in Cryptology - ASIACRYPT 2018*, volume 11273 of *Lecture Notes in Computer Science*, pages 435–464. Springer, 2018.

[BDZ03] Feng Bao, Robert H. Deng, and Huafei Zhu. Variations of diffie-hellman problem. In *Information and Communications Security, 5th International Conference, ICICS 2003*, volume 2836 of *Lecture Notes in Computer Science*, pages 301–312. Springer, 2003.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC 1988*, pages 103–112. ACM, 1988.

[BL12] Daniel J. Bernstein and Tanja Lange. Computing small discrete logarithms faster. In *Progress in Cryptology - INDOCRYPT 2012*, volume 7668 of *Lecture Notes in Computer Science*, pages 317–338. Springer, 2012.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006*, pages 390–399. ACM, 2006.

[BNM+14] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *Financial Cryptography and Data Security - 18th International Conference, FC 2014*, volume 8437 of *Lecture Notes in Computer Science*, pages 486–504. Springer, 2014.

[BPS00] Olivier Baudron, David Pointcheval, and Jacques Stern. Extended notions of security for multicast public key cryptosystems. In *Automata, Languages and Programming, 27th International Colloquium, ICALP 2000*, volume 1853 of *Lecture Notes in Computer Science*, pages 499–511. Springer, 2000.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology - EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 1997.

[CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Advances in Cryptology - CRYPTO 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992.

[Dam] Ivan Damgård. On sigma protocols. http://www.cs.au.dk/~ivan/Sigma.pdf.

[Das] Dash. https://www.dash.org.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. 7668:60–79, 2012.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, STOC 1990*, pages 308–317. IEEE Computer Society, 1990.

[FMMO19] Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. ASIACRYPT, 2019. https://eprint.iacr.org/2018/990.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194, 1986.

[Gol06] Oded Goldreich. *Foundations of Cryptography: Volume 1*. Cambridge University Press, New York, NY, USA, 2006.

[GR10] Steven D. Galbraith and Raminder S. Ruprai. Using equivalence classes to accelerate solving the discrete logarithm problem in a short interval. In *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 368–383. Springer, 2010.

[Gri] Grin. https://grin-tech.org/.

[HAB+17] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017*. The Internet Society, 2017.

[HP01] Stuart Haber and Benny Pinkas. Securely combining public-key cryptosystems. In *Proceedings of the 8th ACM Conference on Computer and Communications Security, CCS 2001*, pages 215–224. ACM, 2001.

[Kur02] Kaoru Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems,*

PKC 2002, volume 2274 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 2002.

[Max] Greg Maxwell. Confidential transactions.

[Max13] Gregory Maxwell. Coinjoin: Bitcoin privacy for the real world, 2013. https://people.xiph.org/~greg/confidential_values.txt.

[MM] Sarah Meiklejohn and Rebekah Mercer. Möbius: Trustless tumbling for transaction privacy. *PoPETs*, 2018(2):105–121.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. https://bitcoin.org/bitcoin.pdf.

[Noe15] Shen Noether. Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098, 2015. https://eprint.iacr.org/2015/1098.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[Poe] Andrew Poelstra. Mimblewimble. https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf.

[Pol78] John M. Pollard. Monte carlo methods for index computation (mod p). *Mathematics of Computation*, 32, 1978.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000.

[PSST11] Kenneth G. Paterson, Jacob C. N. Schuldt, Martijn Stam, and Susan Thomson. On the joint security of encryption and signature, revisited. In *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 161–178. Springer, 2011.

[RMK14] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security*, volume 8713 of *Lecture Notes in Computer Science*, pages 345–364. Springer, 2014.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[Sha71] D. Shanks. Class number, a theory of factorization and genera. *Proc. Symposium in Pure Mathematics*, 20, 1971.

[TMZC17] Raymond K. H. Tai, Jack P. K. Ma, Yongjun Zhao, and Sherman S. M. Chow. Privacy-preserving decision trees evaluation via linear functions. In *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security*, volume 10493 of *Lecture Notes in Computer Science*, pages 494–512. Springer, 2017.

[Woo14] Gavin Wood. Ethereum: A secure decentralized transaction ledger. http://gavwood.com/paper.pdf, 2014. https://www.ethereum.org/.

[ZCa] Zcash: Privacy-protecting digital currency. https://z.cash/.

A Missing Proofs

Theorem A.1. *Twisted ElGamal is IND-CPA secure based on the divisible DDH assumption.*

Proof. We proceed via two games. Let S_i be the probability that $\mathcal{A}$ wins in Game i.

Game 0. The real IND-CPA security experiment. Challenger $\mathcal{CH}$ interacts with $\mathcal{A}$ as below:

1. Setup: $\mathcal{CH}$ sends $pk = g^{sk}$ to $\mathcal{A}$ as the public key.
2. Challenge: $\mathcal{A}$ submits m_0, m_1 to $\mathcal{CH}$ as the target messages. $\mathcal{CH}$ picks a random bit β and randomness r, computes $X = pk^r$, $Y = g^{m_\beta} h^r$, sends $C = (X, Y)$ to $\mathcal{A}$.
3. Guess: $\mathcal{A}$ outputs its guess β' for β and wins if $\beta' = \beta$.

According to the definition of Game 0, we have:

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

Game 1. Same as Game 0 except $\mathcal{CH}$ generates the challenge ciphertext in a different way.

3. Challenge: $\mathcal{A}$ submits m_0, m_1 to $\mathcal{CH}$ as the target messages. $\mathcal{CH}$ picks a random bit β and two independent randomness r and s, computes $X = pk^r$, $Y = g^{m_\beta} h^s$, sends $C = (X, Y)$ to $\mathcal{A}$.

In Game 1, the distribution of C is independent of β, thus we have:

$$\Pr[S_1] = 1/2$$

It remains to prove $\Pr[S_1]$ and $\Pr[S_0]$ are negligibly close. We prove this by showing if not so, one can build an adversary $\mathcal{B}$ breaks the divisible DDH assumption with the same advantage. Given (g, g^a, g^b, g^c), $\mathcal{B}$ is asked to decide if it is a divisible DDH tuple or a random tuple. To do so, $\mathcal{B}$ interacts with $\mathcal{A}$ by simulating $\mathcal{A}$'s challenger in the following IND-CPA experiment.

1. Setup: $\mathcal{B}$ sets $pp = (\mathbb{G}, g, h, p)$ as public parameters, sets g^b as the public key, where b is interpreted as the corresponding secret key $b \in \mathbb{Z}_p$ and unknown to $\mathcal{B}$. $\mathcal{B}$ sends pp and $pk = g^b$ to $\mathcal{A}$.
2. Challenge: $\mathcal{A}$ submits m_0, m_1 to $\mathcal{B}$. $\mathcal{B}$ picks a random bit β and sets $X = g^a$, $Y = g^c h^{m_\beta}$, sends $C = (X, Y)$ to $\mathcal{A}$.
3. Guess: $\mathcal{A}$ outputs a guess β'. $\mathcal{B}$ outputs "1" if $\beta' = \beta$ and "0" otherwise.

If (g, g^a, g^b, g^c) is a divisible DDH tuple, $\mathcal{B}$ simulates Game 0 perfectly (with randomness $c = a/b$). Else, $\mathcal{B}$ simulates Game 1 perfectly (with two independent randomness a/b and c). Thereby, we have $\mathsf{Adv}_{\mathcal{B}} = |\Pr[S_0] - \Pr[S_1]|$, which is negligible in λ by the divisible DDH assumption.

Putting all the above together, Theorem A.1 follows. □

Remark A.1. We stress that twisted ElGamal does not require trusted setup. It remains secure even the adversary knows the discrete relation between (g, h).