PGC: Pretty Good Confidential Transaction System with Accountability

Yu Chen * Xuecheng Ma †

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 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 cryptographic technique.

Despite strong privacy is promising, it might be overkilled or even could be abused in some cases. In particular, anonymity seems contradict to accountability, which is a crucial property for scenarios requiring disputes resolving mechanism, such as e-commerce.

To address the above issues, we introduce accountability to blockchain-based confidential transaction system for the first time. We first formalize a general framework of confidential transaction system with accountability from digital signature, homomorphic public-key encryption and non-interactive zero-knowledge arguments, then present a surprisingly simple and efficient realization called PGC. To avoid using general-purpose zero-knowledge proofs (such as zk-SNARK and zk-STARK), we twist the ElGamal encryption as the underlying homomorphic PKE and develop ciphertext-refreshing approach. This not only enables us to prove transaction validity/correctness by using efficient Σ protocols and zero-knowledge range proofs, but also makes PGC largely compatible with Bitcoin and Ethereum, which could be used as a drop-in to provide confidential enforcements with accountability.

Keywords: cryptocurrencies, confidential transaction, accountability, twisted ElGamal

*Institute of Information Engineering, Chinese Academy of Sciences. Email: yuchen.prc@gmail.com
†Institute of Information Engineering, Chinese Academy of Sciences. Email: maxuecheng@iie.ac.cn

Contents

1 Introduction

Unlike traditional bank systems and e-cash schemes, modern cryptocurrencies such as Bitcoin [Nak08] and Ethereum [Woo14] establish a decentralized peer-to-peer electronic cash system by maintaining an append-only ledger, known as blockchain. The ledger is global distributed and synchronized by consensus mechanisms. An important property of blockchain 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 amount. 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 both Bitcoin and Ethereum provide some weak anonymity through the unlinkability of account addresses to real world identities, it lacks confidentiality, which is a serious limitation in privacy-preserving applications.

1.1 Motivation

Strong privacy is a double-edged sword. While confidentiality is arguably the primary concern of privacy for blockchain-based cryptocurrencies, anonymity might be abused or even prohibitive for applications that require accountability, since it somehow provides plausible deniability. Suppose the employer pay salaries to employees via cryptocurrencies with strong privacy (e.g., Zcash [ZCa] or Monero [Noe15]). If the employer did not receive the right salary, how can he justify 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.

1.2 Our Contributions

To address the aforementioned limitation, in this work we still concentrate on confidentiality, but trade anonymity for accountability. We present PGC (Pretty Good Confidentiality), a simple and efficient confidential transaction system with accountability. For the sake of simplicity and portability, we take an account-based approach and focus only on the simplistic *transaction layer* of cryptocurrencies. The network-level and consensus-level protocols/attacks are put aside for the time being.

We first have to make a choice over commitment and encryption, which will be used as a core primitive to attain confidentiality for PGC.

Encryption vs. Commitment. Maxwell [Max13] first introduced the notion of *confidential transaction* in the context of UTXO model, with the purpose to provide privacy-enhancement for Bitcoin. In CT, the input and output transfer amounts are hidden in Pedersen commitments [Ped91]. To spend a UTXO a user first generates a zero-knowledge proof for validity of transaction (the sum of the committed inputs is greater than the sum of committed outputs, and all outputs are positive), then provides a signature to ensure the transaction to be approved. Recently, an improved CT called Mimblewimble [Poe] was proposed. The improvement is based on a clever observation that for a valid confidential transaction the difference between outputs and inputs value must be 0, thus the difference between input and output Pedersen commitments and the difference between corresponding randomness form an ECDSA key pair. The sender can thus sign the transaction simply with the difference of randomness. This greatly simplifies the structure of confidential transactions.

Nevertheless, commitment-based approach suffers from several drawbacks. On the first place, users must be stateful. They have to keep track of the randomness and the value of each incoming transaction. Otherwise, they are unable to spend it anymore, due to either unable to

generate the zero-knowledge proof (lack of witness) or to generate the signature (lack of signing key). This problem not only renders the overall protocol more complicated (the openings of these commitments must be transferred to Bob through a separate secure channel), but also incurs extra security burden to the design of wallet (the openings must kept in safety). As indicated by Bünz et al. [BAZB19], failure to recover the randomness of a single commitment could make an UTXO isolated and render an account totally unusable. On the second place, they have to record the randomness and value of all incoming and outgoing transactions to achieve accountability. 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.

One may wonder if we can at least solve the first issue by using a computational hiding and perfectly binding commitment, say ElGamal commitment, instead of using Pedersen commitment. We note that this patch does not help. The reason is that all users in CT share the same commitment instance (this means trapdoor is not available), and thus each user still has to be stateful. Actually, the above disadvantages seem inherent for all commitment-based confidential transaction systems [Max13, Poe].

Observing that PKE can be viewed as a computationally hiding and perfectly binding commitment in which secret key serves as a natural trapdoor to recover message, we can simply address the aforementioned issues by equipping each user with a PKE instance rather than making all users share the same global commitment.

1.3 Overview of PGC

Now, we are ready to give a brief technical overview of PGC, which is built from digital signature, public-key encryption and non-interactive zero-knowledge arguments. In PGC, each account is equipped with a key pair, which could be used for both encryption and signature. The public key is used as the address of the account. The balance of each account is kept as an encryption of the real value under individual public key. Let C_A and C_B be the encrypted balances of Alice and Bob respectively. Now, suppose Alice wants to transfer v coins to Bob. She constructs the transaction via the following steps: (1) encrypt v under her public key pk_A and Bob's public key pk_B respectively to obtain C_{out} and C_{in}; (2) prove the validity of this transaction in a zero-knowledge way: (i) C_{out} and C_{in} are two encryptions of the same message under pk_A and pk_B; (ii) her current balance deducting the amount encrypted in C_{out} is still positive; (3) sign $(C_{\text{out}}, C_{\text{in}})$ with resulting zero-knowledge proof π_{valid} using her secret key. The final transaction is roughly of the form $(pk_A, C_{\text{out}}, pk_B, C_{\text{in}}, \pi_{\text{valid}}, \sigma)$. The validity of this transaction can be publicly verifiable by checking the signature and zero-knowledge proof. If the transaction is valid, it will be posted on the blockchain. Accordingly, Alice's balance (resp. Bob's balance) will be updated as $C_A = C_A - C_{\text{out}}$ (resp. $C_B = C_B + C_{\text{in}}$). Such balance update operation implicitly requires that the underlying PKE scheme satisfies additive homomorphism. In addition, whenever there is a dispute over some transaction recorded on the blockchain, either the sender or the receiver is able to reveal the exact amount of the transaction by simply providing a zero-knowledge proof, without exposing their secret keys.

In summary, signature is used to prove ownership of an account, PKE is used to hide the balance and transferred amount, while zero-knowledge argument is used to prove the validity and correctness of transactions. Though the high-level idea is pretty simple, an efficient instantiation of this framework turns out to be non-trivial. Before identifying the potential technical obstacles and introducing our approaches, it is instructive to list our design disciplines of PGC in mind in advance:

- system does not rely on a trusted setup
- users are stateless, i.e., there is no need to maintain a state – information on demand can always be computed from data on the blockchain
- only uses simple and efficient cryptographic schemes based on well-studied assumptions, desirably compatible with Bitcoin and Ethereum

1.3.1 Public-Key Cryptosystems

To be compatible with Bitcoin and Ethereum, we choose ECDSA as the signature scheme. Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p. The secret key sk is a random element in $\mathbb{Z}_p$, and the public key $pk = g^{sk}$. With the aim to use this key pair for both signature and encryption in mind, ElGamal encryption with message encoded in the exponent [CGS97] is arguably the most simple and nature candidate for PKE. This choices of ECDSA and ElGamal PKE bring us at least three benefits:

- The key pair is identical to that used in Bitcoin and similar to that used in Ethereum[1]. This makes our PGC compatible to Bitcoin and Ethereum to a large extent.
- ElGamal encryption is additively homomorphic. This enables balance update operation. Moreover, users could be stateless – the account state (i.e., balance) can be computed from the data on the blockchain with corresponding secret key on demand. There is no need to keep track of the randomness and messages beneath transactions.
- ElGamal encryption is zero-knowledge protocols friendly. Its algebra structure allows us to prove C_{out} and C_{in} are two encryptions of the same message under two public keys via an efficient Σ-protocol. See Protocol B.1 for the details.

1.3.2 Working with Efficient Range Proof

Besides proving C_{out} and C_{in} encrypt the same message, we still have to prove the value v encrypted in C_{out} and value encrypted in $C_A - C_{\text{out}}$ (the current balance deducts v) lie in the right range. The specific tool for this task is zero-knowledge range proof. For both efficiency and security concern, we are going to use Bulletproof [BBB+18], which is efficient and free of trusted setup. Its security is based on discrete logarithm assumption. This makes our whole PGC system based on solely the DDH assumption. Next, we identify the obstacles of making these two range proofs, and how we overcome them.

The first range proof. It turns out that the standard ElGamal encryption cannot work with Bulletproof, because Bulletproof only accepts statements of the form if Pedersen commitment. One may notice that the second component of the ElGamal encryption could be viewed as a Pedersen commitment with public parameter (pk, g), thus it seems that we can feed the Bulletproof with the second component. But, this does not make sense due to the corresponding sk is an obvious trapdoor of such commitment, with which a malicious prover can open the second component to arbitrary value (not necessarily equal the one fixed by the ciphertext) and thus compromises the desired argument of knowledge.

Our idea is *twisting ElGamal* to ensure the second component free of obvious trapdoor. In addition to g, we pick another random generator h. Now, the global public parameter is

[1] Ethereum also uses the discrete logarithm key pair, but uses the hash of public key rather than public key itself as the account address.

$(\mathbb{G}, g, h, p)$. We modify the standard ElGamal encryption as follows. The key pair is still of the form $(pk = g^{sk} \in \mathbb{G}, sk \in \mathbb{Z}_p)$. The encryption of v is tweaked as $(X = pk^r, Y = g^r h^v)$. Decryption can be done by computing $Y/X^{sk^{-1}}$. Now, the second component can be viewed as a Pedersen commitment for v under randomness r, where the public parameter is (g, h) and trapdoor is unknown to all users. Thus, we can safely invoke Bulletproof to generate a range proof of v hidden in statement $g^r h^v$. Luckily, the twisted ElGamal encryption is IND-CPA secure under the divisible DDH assumption, which is equivalent of the standard DDH assumption. Besides, it retains the nice algebra structure and thus still admits efficient Σ-protocol for plaintexts equality.

The second range proof. Another technical obstacle emerges when dealing with the second range proof. In Bulletproof, the prover has to use the opening of the commitment as the witness to generate the proof. This is fine when dealing with the first range proof for v encrypted by C_{out} lies in the right range, because the prover knows the corresponding randomness and value. But, it seems difficult to make a range proof for the value encrypted by $C_A - C_{\text{out}}$ via Bulletproof. The problem stems from the stateless feature of PGC – users do not have to keep track of any states including the randomness under all coming and going transactions, thus the randomness beneath $C_A - C_{\text{out}}$ is typically unknown even assuming homomorphism on randomness.[2]

We resolve this problem by developing *ciphertext refreshing* approach. Note that Alice (the prover) knows sk and thus can decrypt $C_A - C_{\text{out}}$, say v'.[3] She can thus generate a new ciphertext C' of v' under fresh randomness r', and proves $C_A - C_{\text{out}}$ and C' are two encryptions of the same message under the same public key. Then, she can invoke Bulletproof on the second component of C' with witness (r', v'). Note that $C_A - C_{\text{out}}$ and C' are encrypted under the same public key, thereby the plaintext equality argument can be reduced to discrete logarithm equality argument, which in turn can be efficiently proved by a Σ-protocol with witness sk. See Protocol B.2 for the details.

1.3.3 Achieving Accountability

In our confidential transaction system the transferred amount is encrypted, and thus it seems difficult to achieve accountability. A naive solution is making the user involved in the transaction in dispute give out his secret key. However, this will expose all the transactions related to this user in clear. Our approach again leverages the power of zero-knowledge protocols. Let $C_i = (pk_i^r, h^r g^v)$ be the ciphertext in the disputed transaction. The user with pk_i can account C_i by simply publishing a zero-knowledge proof for C_i is indeed an encryption of v under pk_i. Looking ahead, this argument can proved by the Σ-protocol for discrete logarithm equality. See Protocol B.2 for the details.

1.4 Related Work

Maxwell [Max13] first introduced confidential transactions (CT), in which every transaction amount involved is hidden from public view using a commitment. To enable public validation of the blockchain, a zero-knowledge proof of validity (range proof) is appended in every transaction. Mimblewimble [Poe] further improves CT by reducing the cost of signature. To further achieve anonymity, Monero [Noe15] employs linkable ring signature and stealth address, ZCash based on Zerocash [BCG+14] utilizes shielded addresses and general purpose succinct zero-knowledge proofs. Ma et al. [MDH+17] proposed a confidential transaction scheme from

[2] Most encryption schemes are not randomness recovering, e.g., the ElGamal encryption.

[3] In fact, as we will see shortly, ciphertext refreshing could be done efficiently by partial decryption-then-randomization.

the linear encryption [BBS04] and the range proof based on Boneh-Boyen signature [CCS08]. We note that the zero-knowledge proofs used in the aforementioned CT proposals either require a trusted setup (e.g. zk-SNARK [BCG+13, GGPR13] and range proof [CCS08]) or only been asymptotically efficient but practically large (e.g. zk-STARK [BBHR18]).

Concurrent work. Fauzi et al. [FMMO18] proposed a new design for anonymous cryptocurrencies called Quisquis. They mainly employed updatable public keys to achieve anonymity, and used similar cryptographic mechanism to achieve confidentiality.

Bünz et al. [BAZB19] proposed a fully-decentralized, confidential payment system called Zether, which is compatible with Ethereum and other smart contract platforms. In more details, they chosen ElGamal encryption to instantiate the underlying PKE, and developed a custom ZKP named Σ-Bullets to instantiate the ZKPoK.

Compared to their work, we focus solely on building a blockchain-based confidential transaction system, and address the accountability for the first time. Our system is insensitive of account type and network/consensus-level protocols, and thus can be used as a drop-in enhancement to provide confidentiality to many existing cryptocurrencies, such as Bitcoin and Ethereum. From technical aspect, we use the twisted ElGamal as the underlying PKE scheme rather than the standard ElGamal adopted in [FMMO18, BAZB19], and develop ciphertext refreshing approach. This design enables us not only to prove plaintext equality and account transfer amount via efficient Sigma protocol, but also to generate range proofs by directly invoking Bulletproof in a black-box manner. We believe our technique might also benefit the design of Quisquis and Zether by simple adaption.

2 Preliminaries

2.1 Basic Notations

For a set X, we use $x \xleftarrow{R} X$ to denote the operation of sampling x uniformly at random from X, and use $|X|$ to denote its size. We use U_X to denote the uniform distribution over X. For a positive integer d, we use $[d]$ to denote the set $\{1, \ldots, d\}$. We denote $\lambda \in \mathbb{N}$ as the security parameter. We say that a quantity is negligible, written $\mathsf{negl}(\lambda)$, if it vanishes faster than the inverse of any polynomial in λ. 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 indicate that $\mathcal{A}$ outputs z on inputs $(x_1, \ldots, x_n)$ and random coins r. For notational clarity we usually omit r and write $z \leftarrow \mathcal{A}(x_1, \ldots, x_n)$.

Let $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ denote two ensembles of random variables 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 $\mathsf{negl}(\lambda)$.

2.2 Cryptographic Assumptions

Let $\mathsf{GroupGen}$ be a PPT algorithm that takes as input a security parameter 1^λ and outputs a tuple $(\mathbb{G}, g, p)$, where $\mathbb{G}$ is a group of λ-bit prime order p, g is a random generator of $\mathbb{G}$. In what follows, we review the discrete-logarithm based assumptions w.r.t. $\mathsf{GroupGen}(1^\lambda) \to (\mathbb{G}, g, p)$.

Definition 2.1 (Discrete Logarithm Assumption)**.** The discrete logarithm assumption states that for any PPT adversary it holds that:

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

The probability is defined over the random coins of $\mathsf{GroupGen}(1^\lambda)$, $\mathcal{A}$, and the random choice of $h \xleftarrow{R} \mathbb{G}$.

Definition 2.2 (Decisional Diffie-Hellman Assumption)**.** The DDH assumption states that for any PPT adversary it holds that:

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

The probability is defined over the random coins of $\mathsf{GroupGen}(1^\lambda)$, $\mathcal{A}$, 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 states that for any PPT adversary it holds that:

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

The probability is defined over the random coins of $\mathsf{GroupGen}(1^\lambda)$, $\mathcal{A}$, and the random choices of $a, b, c \xleftarrow{R} \mathbb{Z}_p$.

According to [BDZ03], the divisible DDH assumption is equivalent to the standard DDH assumption.

2.3 Zero-Knowledge Proofs

We recall the notions of zero-knowledge argument of knowledge, sigma protocol and zero-knowledge range proofs.

Let $\mathsf{R} \subset X \times W$ be a polynomial-time decidable relation over instance set X and witness set W, and let L be the corresponding language, i.e., $L = \{x \mid \exists w : (x, w) \in \mathsf{R}\}$. Let P and V be two interactive algorithms. We use the notation $\langle P, V\rangle$ denote a two-party protocol, and $\langle P, V\rangle(x)$ denote the final output of V. When the context is clear, we will also slightly abuse the notation of $\langle P, V\rangle(x)$ to denote V's view in the interaction, which includes the common input x, its randomness r and P's messages.

Definition 2.4 (Zero-Knowledge Argument of Knowledge)**.** $\langle P, V\rangle$ is a ZKPoK for language L if the following requirements hold:

- **Completeness.** $\forall x \in L$, we have:

  $$\Pr[\langle P, V\rangle(x) = 1] = 1 - c(\lambda)$$

  Here, $c(\lambda)$ is the completeness error. When $c(\lambda) = 0$, we obtain perfect completeness.

- **Argument of Knowledge.** For any malicious PPT P^* and any $x \in L$, if $\Pr[\langle P^*, V\rangle(x)] = p_x(\lambda)$, then there exists an expected PPT extractor Ext and a polynomial $q(\lambda)$ such that:

  $$\Pr[\mathsf{Ext}^{P^*}(x) \in \mathsf{R}] \geq \frac{q(\lambda)}{p_x(\lambda) - \kappa(\lambda)}$$

  Here, $\kappa(\lambda)$ is the knowledge error.

- **Zero Knowledge.** For any malicious PPT V^* there is an expected PPT simulator Sim such that for all $x \in L$, we have:

  $$\langle P, V^*\rangle(x) \approx_c \mathsf{Sim}(x)$$

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

Definition 2.5 (Sigma-protocols (Σ-protocol))**.** $\langle P, V \rangle$ is a Σ-protocol for language L 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 element e to P;
3. (Response) P replies with a second message z.

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

- **Special soundness.** For any x and any pair of accepting transcripts (a, e, z), (a, e', z') for x with $e \neq e'$, there exists a PPT extractor outputs a witness w for x.
- **Special honest-verifier ZK.** There exists a PPT simulator Sim such that for any $x \in L$ and randomness e, we have:

$$\langle P, V(e) \rangle(x) \equiv \mathsf{Sim}(x, e)$$

A public-coin SHVZK argument of knowledge can be crushed into a non-interactive argument of knowledge by applying the Fiat-Shamir transform [FS86], in which V's random challenge e is set as the hash value of P's first round message a.

Definition 2.6 (Zero-Knowledge Range Proof)**.** For a commitment scheme with message space M and randomness space R, a zero-knowledge range proof is a SHVZK argument of knowledge for the language:

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

3 General Framework of Accountable CT

We formalize the general framework of confidential transaction system with accountability from digital signature, homomorphic PKE and NIZK.

- $\mathsf{Setup}(1^\lambda)$. This algorithm is used to setup the confidential transaction system. It takes as input the security parameter 1^λ, runs $\mathrm{SIG}.\mathsf{Setup}(1^\lambda)$, $\mathrm{PKE}.\mathsf{Setup}(1^\lambda)$ and $\mathrm{NIZK}.\mathsf{Setup}(1^\lambda)$ to generate the global public parameter pp that will be used by all users algorithm.
- $\mathsf{CreateAcct}(pp)$. This algorithm is used to create an account. It takes as input pp, runs $\mathrm{PKE}.\mathsf{KeyGen}(pp)$ to generate a key pair (pk, sk), sets $C^* \leftarrow \mathrm{PKE}.\mathsf{Enc}(pk, 0; r_{\mathrm{dummy}})$[4] as the encryption of initial balance, and initializes a nonce nonce. The public key pk is used as the address of account, the secret key sk is kept by the user itself.[5]
- $\mathsf{RevealAcct}(sk, C^*)$. This algorithm is used to reveal the balance of an account. It takes as input sk and the current encrypted balance C^*, outputs $m \leftarrow \mathrm{PKE}.\mathsf{Dec}(sk, C^*)$.

[4]Here, r_{dummy} could be any fixed and publicly known randomness, say, the zero string 0^λ. This treatment admits the legality of the initial balance value is publicly auditable.

[5]Here, we adopt the key reuse strategy for simplicity and better efficiency, i.e., PKE and SIG share the same key generation algorithm and one key pair is used for both encryption and signature.

- $\mathsf{CreateCTx}(\mathsf{nonce}_1, C_1^*, pk_1, sk_1, v, pk_2)$. This algorithm is used to transfer v coins from account pk_1 to account pk_2. It takes as input the current nonce nonce_1, encrypted balance C_1^* and the key pair (pk_1, sk_1) of the origin account, the coin value v, the public key pk_2 of the destination account, builds the transaction tx via the following steps:

  1. computes $C_1 \leftarrow \text{PKE.}\mathsf{Enc}(pk_1, v; r_1)$, $C_2 \leftarrow \text{PKE.}\mathsf{Enc}(pk_2, v; r_2)$, sets the meta information of the transaction as $\mathsf{meta} = (\mathsf{nonce}_1, C_1^*, pk_1, C_1, pk_2, C_2)$;
  2. runs NIZK.Prove with witness (sk_1, r, s) to generate a proof π_{valid} for the statement $(C_1^*, pk_1, C_1, pk_2, C_2) \in L_{\text{valid}}$, where L_{valid} can be decomposed to $L_1 \wedge L_2 \wedge L_3$:
     - $L_1 = \{(pk_1, C_1, pk_2, C_2) \mid \exists r_1, r_2, v \text{ s.t. } C_i = \text{PKE.}\mathsf{Enc}(pk_i, v; r_i) \text{ for } i = 1, 2\}$
       L_1 ensures that the value sent by pk_1 equal to that received by pk_2.
     - $L_2 = \{(pk_1, C_1) \mid \exists r_1, v \text{ s.t. } C_1 = \text{PKE.}\mathsf{Enc}(pk_1, v; r_1) \wedge v \in [0, 2^\ell]\}$
       L_2 ensures that the value sent by pk_1 lies in a right range.
     - $L_3 = \{(pk_1, C_1^*, C_1) \mid \exists sk_1 \text{ s.t. } \text{PKE.}\mathsf{Dec}(sk_1, C_1^* - C_1) \in [0, 2^\ell]\}$
       L_3 ensures that the balance of pk_1 is enough for the transfer.
  3. runs SIG.$\mathsf{Sign}(sk_1, (\mathsf{meta}, \pi_{\text{valid}})) \rightarrow \sigma$, outputs the confidential transaction as $\mathsf{ctx} = (\mathsf{meta}, \pi_{\text{valid}}, \sigma)$.

- $\mathsf{VerifyCTx}(\mathsf{ctx})$. This algorithm is used to check the validity of ctx. It takes as input $\mathsf{ctx} = (\mathsf{meta}, \pi_{\text{valid}}, \sigma)$, parses $\mathsf{meta} = (\mathsf{nonce}_1, C_1^*, pk_1, C_1, pk_2, C_2)$, then checks its validity via the following steps:

  1. check if SIG.$\mathsf{Verify}(pk_1, \mathsf{meta}) = 1$;
  2. check if NIZK.$\mathsf{Verify}(\mathsf{meta}, \pi_{\text{valid}}) = 1$.

  If both checks pass output "1", else output "0". Particularly, if the confidential transaction is valid, the system appends ctx on the blockchain, and updates the balance of pk_1 as $C_1^* = C_1^* - C_1$ and the balance of pk_2 as $C_2^* = C_2^* + C_2$.

- $\mathsf{RevealCTx}(sk_i, \mathsf{ctx}, v)$. This algorithm is used to testify the value hidden in ctx is indeed v. It takes as input sk_i, v and $\mathsf{ctx} = (\mathsf{meta}, \pi_{\text{valid}}, \sigma)$, where ctx is a valid confidential transaction recorded on the blockchain, $\mathsf{meta} = (\mathsf{nonce}_1, C_1^*, pk_1, C_1, pk_2, C_2)$, and sk_i is the secret key corresponding to pk_i (the index i could be either 1 or 2), then runs NIZK.Prove with witness sk_i to generate a proof π_{correct} for statement $(C_i, v) \in L_{\text{correct}}$. Here, the language L_{correct} is defined as $\{(C_i, v) \mid \exists sk_i \text{ s.t. } v = \text{PKE.}\mathsf{Dec}(sk_i, C_i)\}$. Anyone can check if the value hidden in ctx is indeed the claimed v by running NIZK.$\mathsf{Verify}(C_i, v, \pi_{\text{correct}})$.

Theorem 3.1. *Our PGC is sound based on the EUF-CMA security of signature and the argument of knowledge of NIZKPoK, and is confidential based on the IND-CPA security of PKE and the zero-knowledge property of NIZKPoK.*

4 PGC: an Efficient Instantiation

We now present an efficient realization of the above framework. This is the most technical part of this work. For digital signature, we choose ECDSA, which is also adopted by Bitcoin and Etherum. For PKE, we twist the classical ElGamal encryption. The underlying reason has been elaborated in the introduction part. Next, we present our twisted ElGamal encryption and prove its security in the standard model.

4.1 Instantiating the PKE

We first present our twisted ElGamal PKE as follows.

- $\mathsf{Setup}(1^\lambda)$: run $\mathsf{GroupGen}(1^\lambda) \to (\mathbb{G}, g, p)$, pick $h \xleftarrow{\mathrm{R}} \mathbb{G}^*$, set $pp = (\mathbb{G}, g, h, p)$ as global public parameters. The randomness space R is $\mathbb{Z}_p$ and the message space M is $[0, 2^\ell - 1]$.
- $\mathsf{Gen}(pp)$: on input pp, choose $sk \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, 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 $g^m = Y/X^{sk^{-1}}$, then recover m from h^m.

Remark 4.1. Note that the message is encoded in the exponent, thus the decryption algorithm will need to enumerate over the possible 2^ℓ values, which works only for small values of ℓ, i.e., $\ell = O(\log \lambda)$. Similar situation also occurs in other confidential transaction systems such as [MDH+17, FMMO18, BAZB19], which is not regarded as an issue. First, users typically know their balances and transfer amounts (or at least a good estimate). Thus, brute-force enumeration can be largely avoid in most times. Second, large range can be expressed by several smaller ranges. Third, the average and amortized case complexity of search routine in the decryption algorithm could be reduced to $O(1)$ by maintaining a hash table of size $O(2^\ell)$. In this work, we set ℓ to be 32.

Homomorphism. The above twisted ElGamal satisfies both randomness and message additive homomorphism, i.e., for any pk, (m_1, r_1) and (m_2, r_2), we have $\mathsf{Enc}(pk, m_1; r_1) + \mathsf{Enc}(pk, m_2; r_2) = \mathsf{Enc}(pk, m_1 + m_2; r_1 + r_2)$. Here, we slightly abuse the notation "+" to denote component-wise operation over ciphertext space $\mathbb{G}^2$. Looking ahead, additive homomorphism on message suffices for our usage: $\mathsf{Enc}(pk, m_1) + \mathsf{Enc}(pk, m_2)$ is an encryption of $(m_1 + m_2)$ under some appropriate randomness.

Correctness is obvious. For security, we have the following theorem.

Theorem 4.1. *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 a randomness r, computes $X = pk^r$, $Y = g^r h^{m_\beta}$, 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 $X = pk^r$, $Y = g^s h^{m_\beta}$, 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, we 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: $\mathcal{B}$ sends g^b to $\mathcal{A}$ as the public key, where b is interpreted as the corresponding secret key $b \in \mathbb{Z}_p$ and 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 $X = g^a$, $Y = g^c h^{m_\beta}$, 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, the theorem follows. □

4.2 Instantiating the NIZKPoK

The languages L_{valid} and L_{correct} are fixed after choosing the twisted ElGamal encryption as the PKE scheme. Now, we are ready to design efficient ZKPoK protocol for them.

We begin with $L_{\text{valid}} = L_1 \wedge L_2 \wedge L_3$. Our strategy is designing Σ-protocol for L_1, then using existing range proof in a black-box manner to prove $L_2 \wedge L_3$.

ZKPoK for L_1. According to definition of the twisted ElGamal, L_1 is defined as:

$$\{(pk_1, (X_1, Y_1), pk_2, (X_2, Y_2)) \mid \exists r_1, r_2, v \text{ s.t. } X_1 = pk_1^{r_1} \wedge Y_1 = g^{r_1} h^v \wedge X_2 = pk_2^{r_2} \wedge Y_2 = g^{r_2} h^v\}.$$

Given the common input $(pk_1, X_1, Y_1, pk_2, X_2, Y_2)$, P and V interact in Σ_1 as described below:

1. P picks $a_1, a_2, b \xleftarrow{\text{R}} \mathbb{Z}_p$, sends $A_1 = pk_1^{a_1}$, $A_2 = pk_2^{a_2}$, $B_1 = g^{a_1} h^b$, $B_2 = g^{a_2} h^b$ to V.
2. V picks $e \xleftarrow{\text{R}} \mathbb{Z}_p$ and sends it to P as the challenge.
3. P computes $z_1 = a_1 + er_1$, $z_2 = a_2 + er_2$, $z_3 = b + ev$ using witness $w = (r_1, r_2, v)$, then sends (z_1, z_2, z_3) to V. V accepts the proof iff the following four equations hold simultaneously:

$$pk_1^{z_1} = A_1 X_1^e \quad (1)$$
$$pk_2^{z_2} = A_2 X_2^e \quad (2)$$
$$g^{z_1} h^{z_3} = B_1 Y_1^e \quad (3)$$
$$g^{z_2} h^{z_3} = B_2 Y_2^e \quad (4)$$

Theorem 4.2. *Σ_1 is a public-coin SHVZK argument of knowledge for L_1.*

Proof. Perfect completeness is obvious from simple calculation.

To show special soundness, fix the initial message (A_1, A_2, B_1, B_2), suppose there are two accepted transcripts $(e, z = (z_1, z_2, z_3))$ and $(e', z' = (z'_1, z'_2, z'_3))$ with $e = e'$, we can extract the witness as below. From equation (1), we have $z_1 = a_1 + er_1$ and $z'_1 = a_1 + e'r_1$, which imply $r = (z_1 - z'_1)/(e - e')$. From equation (2), we have $z_2 = a_2 + er_2$ and $z'_2 = a_2 + e'r_2$, which imply $s = (z_2 - z'_2)/(e - e')$. From equation (1) and (3), we have $z_3 = b + ev$ and $z'_3 = b + e'v$, which imply $m = (z_3 - z'_3)/(e - e')$.

To show special HVZK, for a fixed challenge e, the simulator Sim works as below: picks $z_1, z_2, z_3 \xleftarrow{R} \mathbb{Z}_p$, computes $A_1 = pk_1^{z_1}/X_1^e$, $A_2 = pk_2^{z_2}/X_2^e$, $B_1 = g^{z_1}h^{z_3}/Y_1^e$, $B_2 = g^{z_2}h^{z_3}/Y_2^e$.

This finishes the proof. □

ZKPoK for L_2. According to definition of the twisted ElGamal, L_2 is defined as:

$$\{(pk_1, X_1, Y_1) \mid \exists r_1, v \text{ s.t. } X_1 = pk_1^{r_1} \wedge Y_1 = g^{r_1}h^v \wedge v \in [0, 2^\ell]\}.$$

Observe that the statement $\exists r_1, v$ s.t. $X_1 = pk_1^{r_1} \wedge Y_1 = g^{r_1}h^v$ has already been proved in L_1, it suffice to prove $\exists r_1, v$ s.t. $Y_1 = g^{r_1}h^v \wedge v \in [0, 2^\ell]$. Note that the instance Y_1 can be viewed as a Pedersen commitment of value v under public parameters (g, h), where the discrete logarithm of $\log_g h$ is unknown to any users. Thereby, we can directly invoke Bulletproof [BBB+18] to prove this fact by using (r_1, v) as witness.

ZKPoK for L_3. According to the definition of twisted ElGamal, L_3 is defined as:

$$\{(pk_1, C_1^*, C_1) \mid \exists sk_1 \text{ s.t. } \mathsf{PKE.Dec}(sk_1, C_1^* - C_1) \in [0, 2^\ell - 1]\}.$$

Let the current encrypted balance $C_1^* = (X_1^* = pk_1^{r^*}, Y_1 = g^{r^*}h^m)$, which encrypts the current balance m of pk_1 under some randomness r^*. Here, the value m can be revealed by decrypting C using sk_1, but the randomness r^* is unlikely known to the account owner. This is because C_1^* is the sum of all the incoming and outgoing transactions, in which the randomness underlying incoming transactions is unknown. Let $\tilde{C}_1 = C_1^* - C_1$, which is of the form $(\tilde{X}_1 = pk_1^{\tilde{r}}, \tilde{Y}_1 = g^{\tilde{r}}h^{m-v})$. By the additive homomorphism of twisted ElGamal, proving L_3 is equivalent to show that the value $m - v$ encrypted in $\tilde{C}$ lie in the claimed range.

As we discussed above, the randomness r^* underlying C_1^* is unknown, which in turn implies that $\tilde{r}$ is unknown, even exploiting the randomness homomorphism of twisted ElGamal.[6] The consequence is that we can not directly invoke the Bulletproof with instance $\tilde{Y}_1$, since we do not know the associated witness. Our trick is encrypting the value $(m - v)$ under new fresh randomness r' to obtain a new ciphertext $C'_1 = (X'_1, Y'_1)$, where $X'_1 = pk_1^{r'}$, $Y'_1 = g^{r'}h^{m-v}$.[7] Now, we can first prove $\tilde{C}$ and C' are the encryption of the same value, then invoke the Bulletproof on instance $Y'_1 = g^{r'}h^{m-v}$ and witness $(r', m-v)$. It remains to show how to prove $(\tilde{C}, C') \in L_{\text{equal}}$, where L_{equal} is defined as:

$$\{(\tilde{C}, C') \mid \exists sk_1 \text{ s.t. } \mathsf{PKE.Dec}(sk_1, \tilde{C}) = \mathsf{PKE.Dec}(sk_1, C')\}$$

Note that $\tilde{C} = (pk_1^{\tilde{r}}, g^{\tilde{r}}h^{m-v})$ and $C' = (pk_1^{r'}, g^{r'}h^{m-v})$ are encrypted under the same public key pk_1, then proving membership of L_{equal} is equivalent to prove the discrete logarithm of $\log_{\tilde{Y}/Y'} \tilde{X}/X'$ equals $\log_g pk_1$ with witness sk_1. This can be efficiently done by using the Σ-protocol for discrete logarithm equality [CGS97]. For completeness, we describe this protocol in Appendix B.2.

[6] By the randomness homomorphism, we have $\tilde{r} = r^* - r_1$.

[7] Actually, the ciphertext $\tilde{C}_1$ could be efficiently refreshed by firstly using sk_1 to compute h^{m-v} and then masking with newly chosen fresh randomness. There is no need to recover $m-v$, which might be time-consuming.

Let Σ be the protocol obtained by composing Σ_1 for L_1, Bulletproof for L_2, and $\Sigma_3 \circ$ Bulletproof for L_3 in a parallel manner. We have the following lemma.

Lemma 4.3. *Σ is a public-coin zero-knowledge argument of knowledge for $L_1 \wedge L_2 \wedge L_3$.*

Proof. The proof of this lemma follows from the properties of AND-proofs. Σ can be made non-interactive by applying the Fiat-Shamir transform. □

ZKPoK for L_{correct}. According to the definition of twisted ElGamal, L_{correct} can be written as:

$$\{(pk_i, C_i, v) \mid \exists sk_i \text{ s.t. } X_i = (Y_i/h^v)^{sk_i} \wedge pk_i = g^{sk_i}\}$$

Again, this can be efficiently proved by the Σ-protocol for discrete logarithm equality. See Protocol B.2 for the details.

5 Future Works

We defer the performance analysis as the future work.

Acknowledgments

We thank Benny Pinkas for the helpful discussions on Sigma Protocol, and thank Jonathan Bootle for the enlightening discussions on Bulletproof. We thank Yi Deng, Shunli Ma and Cong Tang for many insightful comments and discussions on this work.

References

[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.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. 2018. http://eprint.iacr.org/2018/046.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55, 2004.

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for C: verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology - CRYPTO 2013*, volume 8043 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014*, pages 459–474. IEEE Computer Society, 2014.

[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.

[CCS08] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In *Advances in Cryptology - ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 234–252. Springer, 2008.

[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.

[FMMO18] Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. Cryptology ePrint Archive, Report 2018/990, 2018. 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.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology - EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 626–645. Springer, 2013.

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

[MDH+17] Shunli Ma, Yi Deng, Debiao He, Jiang Zhang, and Xiang Xie. An efficient NIZK scheme for privacy-preserving transactions over account-model blockchain. 2017. https://eprint.iacr.org/2017/1239.

[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.

[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 Basic Cryptographic Schemes

A.1 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, generates public parameter 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 $\mathsf{Com}(m; r) \stackrel{?}{=} c$.

For correctness, we require that for all $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any $m \in M$ and any $r \in R$, 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:

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

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. We recall the celebrated Pedersen commitment as follows:

- $\mathsf{Setup}(1^\lambda)$: on input 1^λ, runs $\mathsf{GroupGen}(1^\lambda)$ to obtain $(\mathbb{G}, p, g)$, picks $h \xleftarrow{\mathrm{R}} \mathbb{G}^*$, outputs $pp = (\mathbb{G}, p, g, h)$. Here, $M = R = \mathbb{Z}_p$, $C = \mathbb{G}$.
- $\mathsf{Com}(m; r)$: on input message $m \in \mathbb{Z}_p$ and randomness $r \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, outputs $c \leftarrow g^r h^m$.
- $\mathsf{Open}(c, m, r)$: outputs "1" if $c = g^r h^m$ and "0" otherwise.

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

A.2 Signatures

A signature scheme with message space M and signature space Σ consists of three 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 verification key vk and a signing key sk.
- $\mathsf{Sign}(sk, m)$: on input sk and a message $m \in M$, output a signature $\sigma \in \Sigma$.
- $\mathsf{Verify}(vk, m, \sigma)$: on input vk, a message m, and a purported signature σ, output 1 indicating acceptance or 0 indicating rejection.

For correctness, we require that for all $pp \leftarrow \mathsf{Setup}(1^\lambda)$, all $(vk, sk) \leftarrow \mathsf{KeyGen}(\lambda)$ and all $m \in M$, we have $\mathsf{Verify}(vk, m, \mathsf{Sign}(sk, m)) = 1$. For security, the exitential unforgeability against chosen message attack (EUF-CMA) is defined as below:

EUF-CMA. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against signature and define its advantage in the following experiment:

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

Here $\mathcal{O}_{\mathsf{sign}}(\cdot)$ is a signing oracle that on input m outputs $\sigma \leftarrow \mathsf{Sign}(sk, m)$. The set $\mathcal{Q}$ records queries to $\mathcal{O}_{\mathsf{sign}}(\cdot)$. A signature is EUF-CMA if no PPT adversary $\mathcal{A}$ has non-negligible advantage in the above security experiment.

B Protocols

B.1 Proving Plaintext Equality

B.2 Proving Discrete Logarithm Equality

Let $\mathbb{G}$ be a cyclic group of prime order p, g_1 and g_2 be two generators of $\mathbb{G}$, h_1 and h_2 be two elements of $\mathbb{G}$. Define the language $L^{\log}_{\text{equal}}$ as below:

$$L^{\log}_{\text{equal}} = \{(g_1, h_1, g_2, h_2) \mid \exists w \in \mathbb{Z}_p \text{ s.t. } \log_{g_1} h_1 = w = \log_{g_2} h_2\}$$

Given the common input (g_1, h_1, g_2, h_2), P interacts with V in the Σ-protocol as below using witness w:

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

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

Theorem B.1. *Σ_2 is a public-coin SHVZK argument of knowledge for $L^{\log}_{\text{equal}}$.*

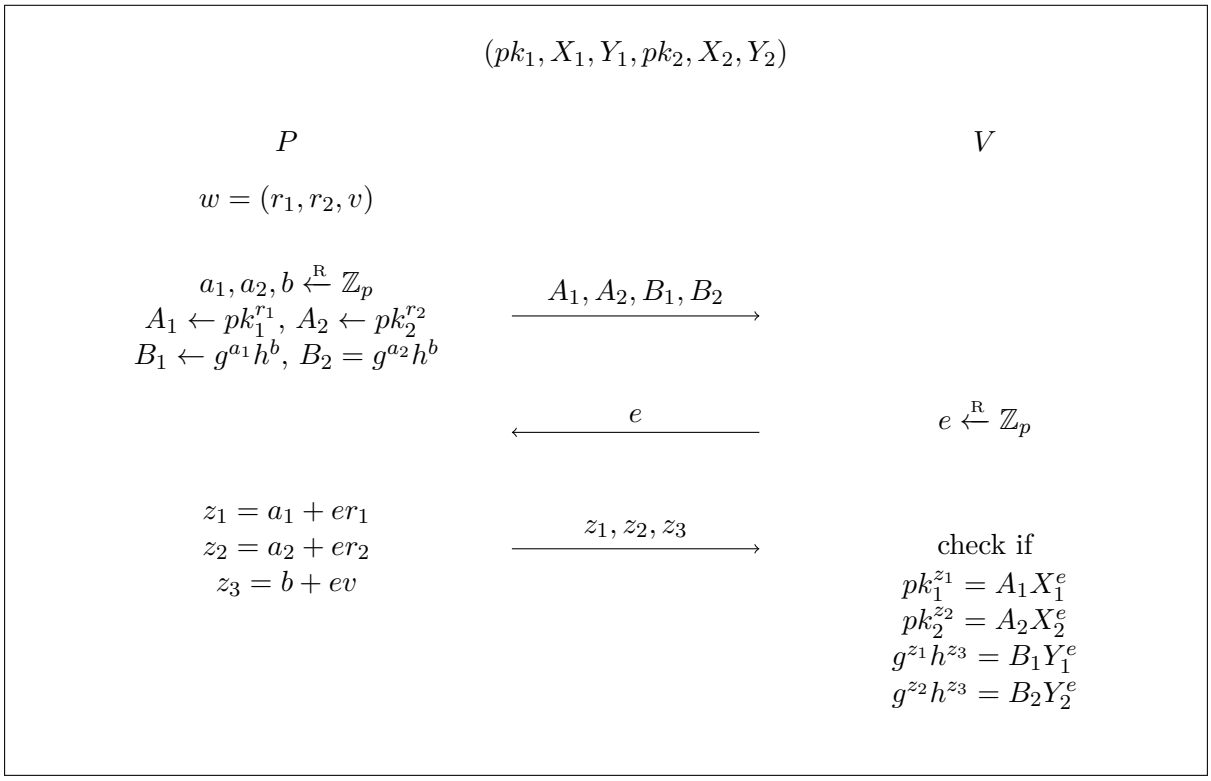

Figure 1: Σ_1 – ZKPoK for L_1: two ciphertexts encrypt the same value

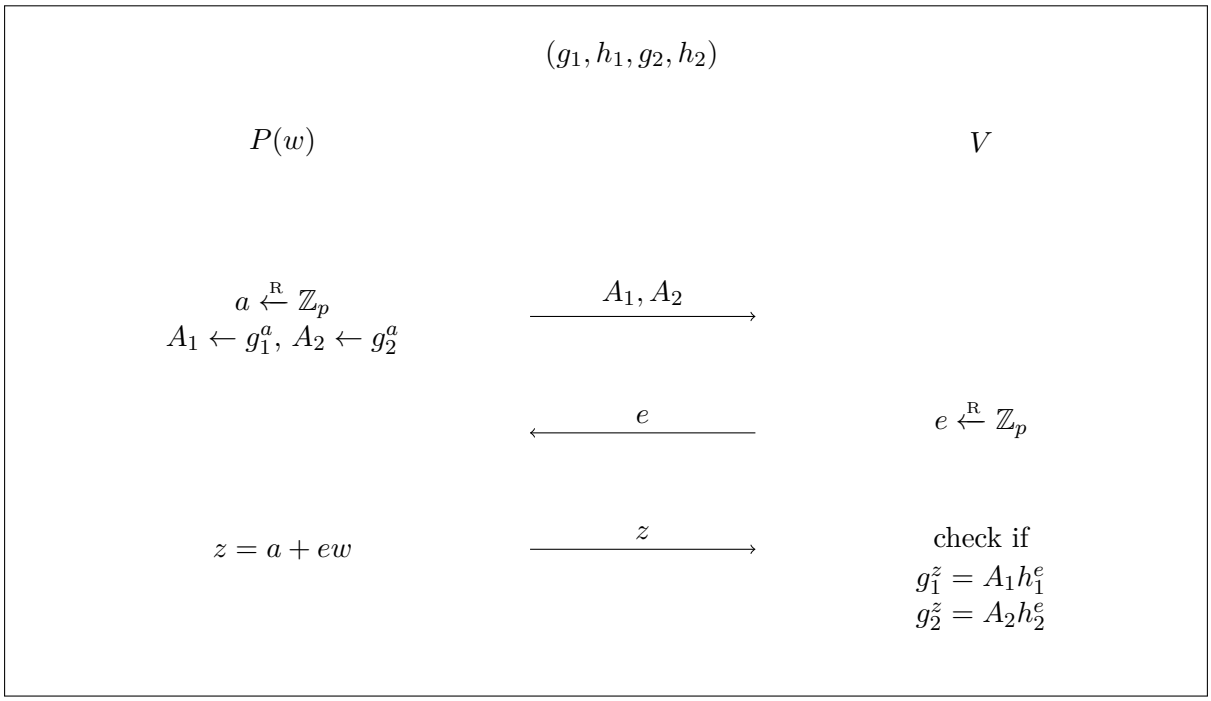

Figure 2: Σ_2 – ZKPoK for $L_{\text{equal}}^{\log}$: discrete logarithm equality