ASOZ: a decentralized payment system with privacy-preserving and auditing on public blockchain

Tianjian Liu[1][0009−0003−9475−1170], Dawei Zhang[1][0000−0001−5942−8245], Chang Chen[1], and Wei Wang[1](✉)

[1] Beijing Jiaotong University
tianjian.liu@bjtu.edu.cn
[2] dwzhang@bjtu.edu.cn
[3] chang.chen@bjtu.edu.cn
[4] wangwei1@bjtu.edu.cn

Abstract. Decentralized payment systems have gradually gotten more attention in recent years. Removing the trusted third party used for accounting ledgers, fundamentally empowers users to control their assets. As privacy concerns grow, some cryptocurrencies are proposed to preserve the privacy of users. However, those cryptocurrencies cause illegal transactions such as money laundering, fraudulent trading and so on. So it is necessary to design an auditing scheme. To solve this problem, many privacy-preserving and audit scheme was proposed. But there exists no scheme that effectively solves the issue of privacy-preserving and auditing on both user identity and transaction content.
In this paper, we propose a design for a decentralized payment system with privacy-preserving and auditing. We use cryptographic accumulators based on Merkle trees for accounting and use a combination of Twist ElGamal, NIZK (Non-Interactive Zero-Knowledge), Bulletproofs, and zk-SNARKs for privacy-preserving and auditing.

Keywords: blockchain · cryptocurrencies · decentralized payment system· privacy-preserving· auditiable· proof of no-knowledge

1 Introduction

With the development of blockchain, decentralized digital payment systems are gradually becoming the future direction of digital payment systems. Compared to centralized payment systems, decentralized digital payment systems are built on public ledgers and digital transaction schemes based on consensus algorithms. Therefore, they are more trusted by the public and offer better prospects for development.

Bitcoin[21] is the first decentralized digital payment system. It records plaintext transaction information on the public ledger, the public ledger can be easily analyzed, tracked, and monitored. So many cryptocurrencies and protocols are proposed to solve those problems, such as ZCash[4], Monero[19,20], Coinjoins[2], and so on. Essentially, those cryptocurrencies and protocols follow the concept of coin mixing. Coin mixing refers to using cryptology method or trust mechanism, to verify the correctness of a transaction without leaking the information of a transaction in a transaction list. Among them, the zerocash protocol of ZCash has the strongest mixing ability, we call it global mixing. Using membership proof of Merkle tree[23], zerocash scheme mixing sender address into a whole Merkle tree. This means if the depth of the Merkle tree is 32, then the transaction list size is 2^{32}.

However, the privacy-preserving method above leads to some problems. Since the transaction information is not visible, those cryptocurrencies lead to illegal transactions. according to the report of Chainalysis[16], the cryptocurrency received by mixers is \$7.8 billion in 2022, 24% of which came from illicit addresses. In the year 2021, this percentage amounted to a mere 10%. So it is imperative to introduce audit mechanisms within the frameworks of privacy-preserving schemes. To solve those problems, many privacy-preserving and audit scheme was proposed [12,17,7,25,18]. [12] proposed a basic improvement method to add audit function based on zero cash; [7] proposed a transaction content auditing scheme based on Twist ElGamal, but it does not support audit for identities; [25] use Pedersen Commitment and ElGamal to preserve transaction content and identities, but it lacks a design for recording transactions and relies on a conventional blockchain structure for recording; [17] builds on the design of transaction values and identities and uses cryptographic accumulators as a way to record transactions but employs a significant amount of zk-SNARKs, resulting in lower computational efficiency. Its sender address is sent in plaintext, the privacy-preserving ability is limited; [18] designs transaction records using a tabular structure based on the design of transaction values and identities, but this makes the scheme unsuitable for handling a large number of users, only effective in scenarios involving a small number of institutions within the defined consortium chain.

From above, we can see that there are some challenges in decentralized digital payment systems currently, including how to define audit capabilities, how to introduce effective audit mechanisms based on privacy-preserving schemes, how to balance the audit capabilities and scheme efficiency.

In this paper, we propose a design for a decentralized payment system with privacy-preserving and auditing. We use cryptographic accumulators based on Merkle trees for accounting and use a combination of Twist ElGamal, NIZK (Non-Interactive Zero-Knowledge), Bulletproofs, and zk-SNARKs for privacy-preserving and auditing. We summarize our contributions as follows:

– We design a decentralized transaction privacy-preserving and auditing scheme called ASOZ for scenarios with large-scale users, supporting privacy-preserving and auditing of both identities and transaction content. We formalize the system model and security properties of ASOZ, including correctness, auditability, soundness, and privacy. We give the principles in design audit schemes, including offline audit, Out-of-band cost, full transaction audit, and minimal information disclosure. We instantiate our scheme following the principles,

and prove that the scheme achieves all the security requirements in the formalized security model.
- We utilize a global mixing scheme for identity privacy-preserving, offering the strongest hiding capabilities among all cryptocurrency schemes.
- We propose a sigma-protocol-friendly construction, in comparison to typical zk-SNARK solutions, our construction has lower computational and storage cost.

The rest of this paper is organized as follows:

In section 2, we introduce the basic concepts and notations used in this paper. In section 3, we give an overview of our scheme. In section 4, we describe the details of our scheme. In section 5, we give security proof. In section 6, we analyze the performance of our scheme. In section 7, we give further discussion.

2 Preliminaries

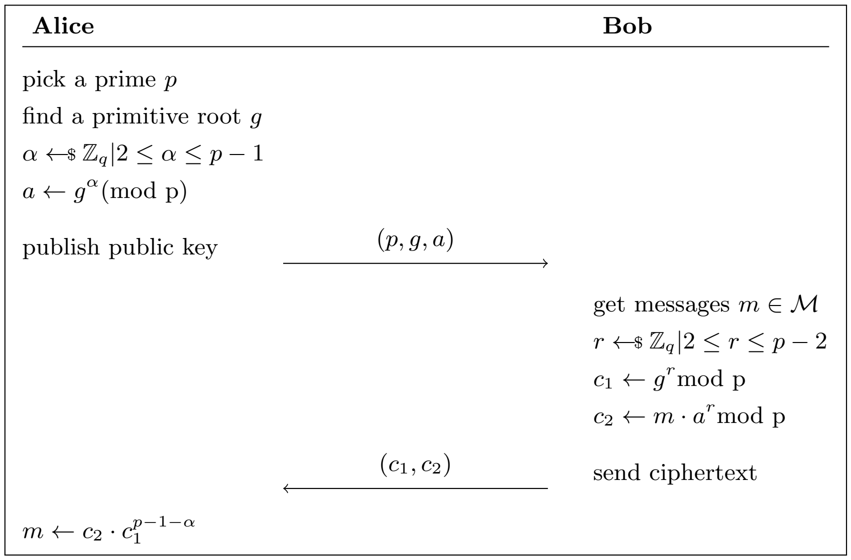
2.1 Notaion

Pedersen Commitment Below we recall Pedersen Commitment[22]:

$$comm(v, r) = g^v h^r \tag{1}$$

Where v is the value we want to hide, r is a random number, g and h are two generators of a cyclic group G with $s = |G|$ elements and prime order p, $\mathbb{Z}_p = \{0, 1, \ldots, s-1\}$, $v \in \mathbb{Z}_p$, and $r \in \mathbb{Z}_p$. The Pedersen commitment is perfectly hiding and computational binding under the discrete logarithm assumption.

ElGamal ElGamal encryption is a public-key cryptosystem, below we recall ElGamal encryption[9,10]:



Alice
Bob
pick a prime p
find a primitive root g
$\alpha \xleftarrow{\$} \mathbb{Z}_q | 2 \le \alpha \le p-1$
$a \leftarrow g^{\alpha} (\text{mod p})$
publish public key
(p, g, a)
get messages $m \in \mathcal{M}$
$r \xleftarrow{\$} \mathbb{Z}_q | 2 \le r \le p-2$
$c_1 \leftarrow g^r \text{mod p}$
$c_2 \leftarrow m \cdot a^r \text{mod p}$
(c_1, c_2)
send ciphertext
$m \leftarrow c_2 \cdot c_1^{p-1-\alpha}$

ElGamal encryption algorithm is based on the security of Diffie-Hellman distribution scheme. This algorithm is IND-CPA secure. As for its friendliness to Sigma protocol, we use ElGamal encryption algorithm to encrypt the audit information in our scheme.

Twist ElGamal Below we recall Twist ElGamal encryption[7]:

Alice		**Bob**
pick a prime p		
find a primitive root g, h		
$\alpha \leftarrow\$ \mathbb{Z}_q \vert 2 \leq \alpha \leq p-1$		
$a \leftarrow g^{\alpha}$(mod p)		
publish public key	$\xrightarrow{(p, g, h, a)}$	
		get value $v \in \mathbb{Z}_q$
		$r \leftarrow\$ \mathbb{Z}_q$
		$c_1 \leftarrow a^{r}$mod p
		$c_2 \leftarrow g^{r} \cdot h^{v}$mod p
	$\xleftarrow{(c_1, c_2)}$	send ciphertext
$v \leftarrow \log_h \frac{{c_1}^{-\alpha}}{c_2}$		

In the above algorithm, the size of v is much smaller than p so that we can calculate the discrete logarithm of $({c_1}^{-\alpha})/c_2$. This algorithm supports homomorphic, so we directly use this algorithm to calculate transaction value in a black box manner. This algorithm is IND-CPA secure based on the divisible DDH assumption, which satisfies the requirements of our scheme.

Sigma Protocol Sigma protocol [24] is an interactive proof protocol, it contains three parts: commitment, challenge, and response. And it is easy to convert to non-interactive proof by random oracle model. In general, Sigma protocol is more efficient than zk-SNARK.

Fiat-Shamir Heuristic The Fiat-Shamir heuristic [11] is a technique to convert an interactive protocol to a non-interactive proof in the random oracle model. This technique assumes pseudorandom function (PRF) as a random oracle, and uses it to replace the random challenge from verifier.

Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) zk-SNARK[22] is an efficient variant of zero-knowledge proof of knowledge. It can prove an arithmetic circuit without revealing any information about witness

Bulletproof Bulletproof [5] is a non-interactive zero-knowledge proof protocol. It can prove inner-product argument and range proof based on Pedersen Commitment. It has a small proof size and fast speed, so it is widely used in many privacy coin such as ZCash [8], and Monero.

3 Solution Overview

3.1 Design Goal

Our scheme is shown in figure 1. In privacy-preserving scenarios, how to ensure that transaction information is not visible to others, but that others can verify the correctness of the transaction is a key issue. In this part, our scheme is similar to the zerocash protocol, the owner of the coin uses the public key signature to represent the ownership of the coin, and uses private key signature to get nullifier to indicate that the coin has been spent. The purpose is that the spending of the coin is stored in SNList, while the ownership of the coin is stored in CMList, this can achieve the unlinkability of the transaction. In our scheme, Auditor can open the transaction form the trapdoor and link the transaction. Afterwards, auditor can sanction the illegal transactions. To trace the flow of illegal transactions, Auditor needs to open the transaction from the trapdoor and link the transaction. Afterwards, auditor can sanction the illegal transactions. In this process, a key issue is ensuring the reliability of regulation, meaning that regulatory entities can accurately reconstruct transaction data.

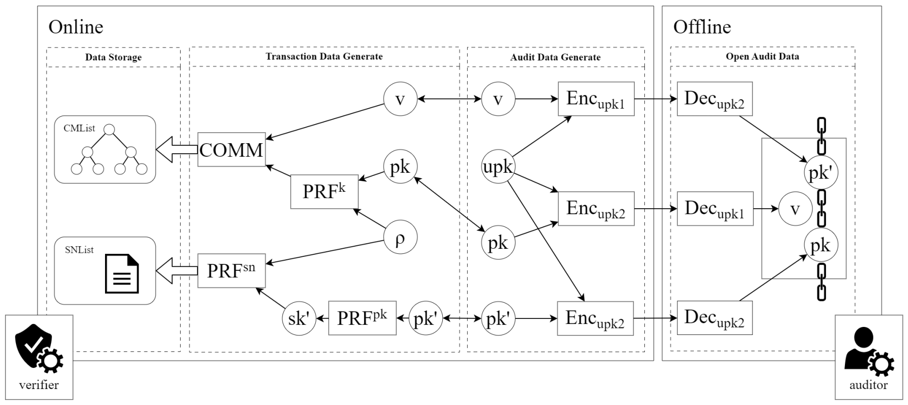


Fig. 1. scheme design

3.2 Base Character

Below we introduce the base character in our scheme.

- **Sender**: The initiator of the transaction.
- **Receiver**: The recipient of the transaction.

- **Verifier**: The leader of updating public state in consensus algorithm, Generally verify a large number of transactions according to the protocol rules, then record the transactions in public ledger. In our scheme, the verification process does not require the involvement of auditors, and can be implemented as smart contracts.
- **Auditor**: In order to audit money laundering, illegal transactions, and other behaviors. We need a character to open transactions to get the identities and values. Such as the tax bureau and so on.

3.3 Scheme Definition

Below we propose the definition of the scheme. Firstly, we provide a rough definition of the symbols in table 1, with more precise definitions to be provided in scheme design in chapter 4. Then we will formalize the notions of correctness, soundness, and privacy via security experiments, and capture the threat with oracles.

Our scheme is composed of the following algorithms:

- $Setup : pp \leftarrow 1^\lambda$
- $CreateAddress : (sk, pk) \leftarrow U \times pp$
- $CreateAuditKey : (usk, upk) \leftarrow A \times pp$
- $CreateTran : (\pi_t, cm, sn, \rho_n) \leftarrow U_s \times (pk_r, \rho_o, sk_s, v, r)$
- $CreateAudit : (\pi_a, s) \leftarrow U_s \times (cm, sn, upk)$
- $VerifyTran : 0/1 \leftarrow V \times (\pi_t, cm, sn)$
- $VerifyAudit : 0/1 \leftarrow V \times (\pi_a, s)$
- $AuditTran : (pk_s, pk_r, v) \leftarrow A \times (usk, cm, sn, s)$
- $SendSecret : (sm) \leftarrow U_s \times (epk_r, \rho_n, v)$

3.4 Security Model

Let $\mathcal{A}$ be an adversary attacking our system. Formally, we capture attack behaviors as adversarial queries to oracles implemented by a challenger $\mathcal{CH}$. We list the oracles available to the adversary as below.

- $\mathcal{O}_P$: $\mathcal{A}$ queries this oracle to get a public key pk. The $\mathcal{CH}$ invoke $CreateAddress$ to get (sk, pk), then move sk, pk to the keys list T, return pk to $\mathcal{A}$. This oracle describes that $\mathcal{A}$ can get honest public keys.
- $\mathcal{O}_S$: $\mathcal{A}$ queries this oracle with a public key pk. $\mathcal{CH}$ first check if pk appears in corrupt list $T_{corrupt}$, if not, return a secret key sk, then $\mathcal{CH}$ move pk and sk to the corrupt list $T_{corrupt}$; if so, return $\perp$. This oracle describe that $\mathcal{A}$ can get a set of leaked public-secret key pairs.
- $\mathcal{O}_V$: $\mathcal{A}$ queries this oracle with $(pp, pk_r, \rho, sk_s, v, r)$, $\mathcal{CH}$ first check if pk_r or sk_s appears in $T_{corrupt}$, if so, return $\perp$; if not, the $\mathcal{CH}$ invoke $CreateTran$ to get (π_t, cm, sn), then $\mathcal{CH}$ invoke $CreateTran$ to get (π_a, s), finally $\mathcal{CH}$ return $(\pi_t, \pi_a, cm, sn, s)$ to $\mathcal{A}$. The oracle describe that $\mathcal{A}$ can use an honest public key to create a valid transaction.

– $\mathcal{O}_{RT}$: $\mathcal{A}$ queries this oracle with (cm, s) or (sn, s), $\mathcal{CH}$ first get (ρ, pk_r, v) or (ρ, sk_s) that generate cm or sn. Then $\mathcal{CH}$ check if pk_r or sk_s appears in $T_{corrupt}$, if not, return (ρ, pk_r, v) or (ρ, sk_s) above; if so, return $\perp$. This oracle describes a chosen plaintext attack (CPA) on transaction.

Table 1. Symbols define

Symbol	**Meaning**
U	is the set of users. In this set, we define U_s as sender, and U_r as receiver in 3.2.
V	is the set of verifier in 3.2.
A	is the set of auditor in 3.2.
ρ	is the secret identity of coin. To ensure security, U_s needs to randomly update its value in a transaction. So we define ρ_o as the identifier of the old coins and ρ_n as the identifier of the coins to be poured. We want to point out that the effect of ρ is similar to the superposition of (ρ, r) in Zerocash[4].
v	is the transaction value, we define $\{v_{in,i}\}$ as input value from sneder, and $\{v_{out,i}\}$ as output value to receivers.
c	is the challenge parameter attached to the proof.
cm	is the commitment of the ownership.
sn	is the nullifier of the coin.
π_t	is the zero-knowledge proof (ZKP) of transaction information.
r	is the additional parameter used by sender U_s to calculate commitment cm.
(sk, pk)	is the signature key of users. And pk is the address of users. (sk_r, pk_r) refer in particular to the key pair of receiver, and(sk_s, pk_s) refer in particular to the key pair of sender.
(usk, upk)	is the audit key pair.
s	is the additional parameter used by auditor A to decrypt transaction information. s is encrypted by upk.
π_a	is the ZKP of audit information.
(esk, epk)	is the secret transmission key pair of users.
sm	is the ciphertext encrypted by epk.

Then, we give the security model of the description above[6]:

correctness Transactions that follow the rule of $CreateTran$ can pass $VerifyTran$.

$$Adv_{\mathcal{A}}^{C} = 1 - \Pr\left[\beta = 1 : \begin{array}{r}(\pi_t, cm, sn, \rho_n) \leftarrow CreateTran(pk_r, \rho_o, sk_s, v, r)\\ \beta \leftarrow VerifyTran(\pi_t, cm, sn)\end{array}\right] \quad (2)$$

auditability If transactions can pass $VerifyAudit$ it can be opened by $AuditTran$ correctly.
We propose the following security experiment to describe the second half of the attribute above.

$$Adv_{\mathcal{A}}^{A} = 1 - \Pr\left[\begin{array}{ll} pk_s = pk_s^* & (\pi_t, cm, sn, \rho_n) \leftarrow CreateTran(pk_r, \rho_o, sk_s, v, r) \\ pk_r = pk_r^* : & 1 \leftarrow VerifyAudit(\pi_t, cm, sn) \\ v = v^* & (pk_s^*, pk_r^*, v^*) \leftarrow AuditTran(usk, cm, sn, s) \end{array}\right] \quad (3)$$

soundness Transactions that do not follow the rule of $CreateTran$ cannot pass $VerifyTran$.

$$Adv_{\mathcal{A}}^{C} = 1 - \Pr\left[\beta = 0 : \begin{array}{r} (\pi_t^*, cm^*, sn^*) \leftarrow\$ \mathbb{Z}_p \\ \beta \leftarrow VerifyTran(\pi_t^*, cm^*, sn^*) \end{array}\right] \quad (4)$$

privacy verifier cannot get transaction information from public data. Including transaction value and transaction identity.

Experiment 0. This experiment describe the advantage that adversary can get v, pk_r, pk_s from public data.

1. generation phase: on input a security parameter λ, run $pp \leftarrow Setup(1^\lambda)$
2. pre query phase: $\mathcal{A}$ adaptively makes queries to $\mathcal{O}_P, \mathcal{O}_S, \mathcal{O}_V, \mathcal{O}_{RT}$ finitely.
3. query phase: $\mathcal{CH}$ invoke $CreateAddress$ and get (pk_r, sk_s), $\mathcal{CH}$ random generate ρ and generate v from a smaller range, then invoke $CreateTran$, $CreateAudit$ to get $(\pi_t, \pi_a, cm, sn, s)$ and send to $\mathcal{A}$.
4. guess phase: $\mathcal{A}$ output $pk_r{}', pk_s{}', v'$

We define the advantage of $\mathcal{A}$ to guess v, pk_s, pk_r:
$Adv_{\mathcal{A}}^{v}(\lambda) = \Pr[v = v']$.
$Adv_{\mathcal{A}}^{s}(\lambda) = \Pr\left[pk_s = pk_s{}'\right]$.
$Adv_{\mathcal{A}}^{r}(\lambda) = \Pr\left[pk_r = pk_r{}'\right]$.
Then we define the advantage of $\mathcal{A}$:
$Adv_{\mathcal{A}}^{P}(\lambda) = \max\{Adv_{\mathcal{A}}^{v}(\lambda), Adv_{\mathcal{A}}^{s}(\lambda), Adv_{\mathcal{A}}^{r}(\lambda)\}$.

In addition, we propose other principles for the design of audit scheme, and we think that these attributes help improve system efficiency and protect the privacy of legitimate users. In our plan, we will follow below principles:

Offline audit Scheme allows offline auditing, the auditor does not need to track and maintain the entire public state.

Out-of-band Cost Audit scheme should use the data structure provided by the privacy-preserve scheme whenever possible, with the aim of minimizing additional storage, computation, and interaction costs.

Full transaction audit In an auditing scheme, the auditing party has the ability to open all transactions and obtain information about all participants and value, we refer to this auditing scheme as a full transaction audit scheme. Under a full transaction audit scheme, the auditing party can get the flow of all funds and the individuals involved.

Minimal information disclosure When the auditor opens private data for the purpose of auditing, he should strive to minimize unnecessary disclosures of privacy.

Security Strength Unchanged The introduction of audit measures will not result in a diminution of the privacy-preserving strength in the original privacy-preserving schemes.

3.5 How to Preserve Privacy

Our solution here is based on the membership proof of Merkle tree[23]. In our plan, we use Merkle tree to record the commitment of the coin, and we call the Merkle tree as CMList. We record the value (v, ρ, pk) as a commitment on the leaf node of CMList.

$$\begin{aligned} k &= PRF^{k}(pk, \rho) = g^{pk}h^{\rho} \\ cm &= PRF^{cm}(v, k) = g^{k}h^{v} \end{aligned} \tag{5}$$

Where:

– v is the value of the coin.
– ρ provides the right to use the coin, and introduce randomness into commitment.
– pk provides ownership of the coin.

When the user uses the coin, he gives the membership proof of (v, ρ, pk) below a root of CMList in ledger, membership proofs do not disclose the position of the commitment in the cmlist, thereby preserving the privacy of the sender's identity. When he spends the coin, he uses a new ρ generated randomly and receiver's public key to generate a new commitment. The secret parameter ρ is sent to receiver in a security way, which we do not consider in this paper.

We can see only users who know ρ with a correct pk can spend the coin. Now, we only need a PRF function of commitment, the specific form of the PRF we will explain in audit part in 3.6.

When a user spends a coin, he also needs to generate a nullifier for the coin and record it on a publicly maintained list SNList.

$$\begin{aligned} sn &= PRF^{sn}(\rho, sk) = g^{sk}h^{\rho} \\ pk &= g^{sk} \end{aligned} \tag{6}$$

The nullifier is unidirectional binding with CMList using sk, user uses zk-SNARK to prove this binding. This can avoid the double spending problem by searching for the same record on the SNList. The function PRF^{sn} need to be collision-resistant, once the resistant exists, the spending of later coin will fail. We instantiate PRF^{sn} as the form of Pedersen commitment, because of its random uniform distribution, the probability of collision is negligible.

We would like to emphasize here that, Although both CMList and SNList are member lists that require public maintenance to ensure their immutability. However, the CMList needs to prove that the coin is in the Merkle tree structure in the commitment list when the specific content of the coin is not visible; SNList is just a search list, which can be a structure such as B+ tree, Bloom Filter, etc. The specific form of the PRF we will explain in the audit part in 3.6.

Compared to the tabular ledger data structure of FabZK[18], our solution only needs to record the commitment of the coin and the information of the coin spent during each transaction, rather than record transactions generated by every part, so the storage cost is linear correlation with the number of users, making it more suitable for large-scale transaction scenarios compared to tabular ledger data structures.

3.6 How to Audit

According to the prior definition, our auditing refers to the process that the verifier leave a Trapdoor in the encrypted transaction for the auditor before the transaction is confirmed. The auditor can open the encrypted transaction and get the transaction value and identities. In our scheme, the audit does not need to interact online, the auditor can open the encrypted data on blockchain alone. This follows the Offline Audit principle.

To describe our scheme, let us review our PRF^{sn}, PRF^{cm}, Twist ElGamal and ElGamal ciphertext:

$$\begin{aligned} PRF^{cm}(v,k) &= g^k h^v \\ PRF^{sn}(\rho, sk) &= g^{sk} h^{\rho} \\ pk &= g^{sk} \\ TwistElGamal(m) &= (a^r, g^r h^m) \\ ElGamal(m) &= (g^r, ma^r) \end{aligned} \tag{7}$$

We can see that the form of PRF^{cm} and PRF^{sn} is similar to the form of $TwistElGamal(m)[1]$, $ElGamal(m)[1]$. In the equations above, the (v, pk, sk) is the parameter need to be preserved and (v, pk) is the parameter needs to be audited, so if we give the:

$$\begin{aligned} TwistElGamal(v) &= (upk^k, g^k h^v) \\ ElGamal(pk) &= (g^r, pk \cdot h^{\rho}) \\ upk &= g^{usk} \end{aligned} \tag{8}$$

Then we can audit v in CMList and pk in SNList, only left to prove the consistency of the parameters involved in the calculation. We give the Sigma protocol of those proofs later in 4. Unfortunately, we cannot give a pure Sigma protocol to prove the consistency of pk in CMList, but we combine the Sigma protocol with zk-SNARK to prove it.

Besides, by using ElGamal and TwistElGamal to encrypt transaction information, our introducing of audit is IND-CPA secure, which follows the Security Strength Unchanged principle. From another perspective, the form of PRF^{cm} is friendly to bulletproof, through which we can prove the sender has solvency by proving assets are not negative after payment.

4 Scheme Design

4.1 Transaction Scheme

In order to explain our scheme, we take 2 input-1 output transaction as an example. Actually, the scheme can be extended to m input-n output transactions.

Sender has two coin record: $coin_1 = (\rho_1, v_1)$ and $coin_2 = (\rho_2, v_2)$(ρ is the secret identifier of coin, v is the value of coin), and sender wants to merge these two coins and send them to receiver.

1. Sender calculate commitment $cm_1^{old} = g^{PRF^k(pk_s,\rho_1)}h^{v_1}$ $cm_2^{old} = g^{PRF^k(pk_s,\rho_2)}h^{v_2}$
2. Sender calculate $cm' = cm_1^{old} \cdot cm_2^{old}$
3. Sender uses zk-SNARK to make membership proof: π_1, to prove commitment cm_1^{old} and cm_2^{old} in the Merkle tree of CMList and the correctness of cm'.
4. Sender calculate $sn_1 = PRF^{sn}(\rho_1, sk_s)$ and $sn_2 = PRF^{sn}(\rho_2, sk_s)$
5. Sender generates proof, to prove nullifier is generated by his private key: $\pi_2 = PoK\{(sk_s, pk_s, \rho_1, \rho_2) : pk_s = PRF^{pk}(sk_s) \bigwedge sn_1 = PRF^{sn}(\rho_1, sk_s) \bigwedge sn_2 = PRF^{sn}(\rho_2, sk_s)\}$. We use zk-SNARK to prove π_2.
6. Sender randomly generate ρ_3 and calculate new commitment of coin: $cm^{new} = g^{PRF^k(pk_r,\rho_3)}h^{v_3}$
7. Sender calculate $r_{equation}$, make it satisfy $PRF^k(pk_s, \rho_1) + PRF^k(pk_s, \rho_2) - PRF^k(pk_r, \rho_3) + r_{equation} = 0$
8. Sender use bulletproof to generate range proof $\pi_3 = PoK\{(v_3, cm^{new}, r) : cm^{new} = g^r h^{v_3} \bigwedge v_3 \in [0, 2^n]\}$
9. Sender encrypt transaction value with audit key: $X = upk^{PRF^k(pk_r,\rho_3)}, Y = cm^{new}$. We define $C_{content} = Enc_{upk1}(v_3) = (X, Y)$
10. Sender generates ZKP, prove that the values encrypted by audit key are correct $\pi_4 = PoK\{(r_2) : X = upk^{r_2} \bigwedge Y = g^{r_2}h^{v_3}\}$, where $r_2 = PRF^k(pk_r, \rho_3)$
    Sigma protocol of π_4(P for prover and V for verifier):
    (a) P randomly generates a,b. send $A = upk^a, B = g^a h^b$ to V
    (b) V randomly choose e, send e to P as challenge
    (c) P calculate $z_1 = a + er_2, z_2 = b + ev_3$, send z_1, z_2 to V, V check the following equation:
    $$upk^{z_1} = AX^e \tag{9}$$
    $$g^{z_1}h^{z_2} = BY^e \tag{10}$$
    We use Fiat-Shamir Transform to convert the above Sigma protocol into a NIZK. We want to point out that this proof can be easily extended to m input-n output transactions. And the proof of security properties can be seen in [7].
11. Sender uses audit key encrypt identity information: $C_{sender} = Enc_{upk2}(pk_s)$, $C_{receiver} = Enc_{upk2}(pk_r)$
12. Sender generates ZKP, prove that the identities of sender encrypted with audit key are correctly $\pi_5 = PoK\{(pk_s, sk_s) : C_{sender} = Enc_{upk2}(pk_s) \bigwedge pk_s = PRF^{pk}(sk_s) \bigwedge sn_1 = PRF^{sn}(\rho_1, sk_s) \bigwedge sn_2 = PRF^{sn}(\rho_2, sk_s)\}$
    Then we expand PRF and Enc function, π_5 can be written as:
    $$\begin{aligned} &\pi_5 = \pi_7 \circ \pi_8, C_{sender} = (X_1, Y_1) \\ &\pi_7 = PoK\{(r_3) : X_1 = g^{r_3} \bigwedge Y_1 = upk^{r_3}g^{sk_s}\} \\ &\pi_8 = PoK\{(sk_s) : Y_1 = upk^{r_3}g^{sk_s} \bigwedge sn_1 = g^{sk_s}h^{\rho_1} \bigwedge sn_2 = g^{sk_s}h^{\rho_2}\} \end{aligned} \tag{11}$$

Where $Enc_{upk2}(pk_s) = (X_1, Y_1)$, and r_3 is random generated. The proof of π_7 is similar with Sigma protocol of π_4, so we do not prove them in there. The proof of π_8 is below:

Sigma protocol of π_8:

(a) P random generate a, b_1, b_2, b_3. send $A = g^a, B_1 = upk^{b_1}, B_2 = h^{b_2}, B_3 = h^{b_3}$ to V

(b) V random generate e, send e to P as challenge

(c) P calculate $z = a + e \cdot sk_s, z_1 = b_1 + e \cdot r_3, z_2 = b_2 + e \cdot \rho_1, z_3 = b_3 + e \cdot \rho_2$, send z, z_1, z_2, z_3 to V, V check the following equation:

$$g^z upk^{z_1} = A(Y_1)^e B_1 \tag{12}$$

$$g^z h^{z_2} = A(sn_1)^e B_2 \tag{13}$$

$$g^z h^{z_3} = A(sn_2)^e B_3 \tag{14}$$

We use Fiat-Shamir Transform to convert the above Sigma protocol into a NIZK. We want to point out that this proof can be easily extended to m input-n output transactions. We will give the proof of security properties later in A.1.

13. Sender generates ZKP, prove that the identities of receiver encrypted with audit key are correct: $\pi_6 = PoK\{(pk_r) : C_{receiver} = Enc_{upk2}(pk_r) \bigwedge cm^{new} = g^{PRF^k(pk_r,\rho_3)} h^{v_3}\}$.
Then we expand PRF and Enc function, π_6 can be written as:

$$\begin{aligned} \pi_6 &= \pi_9 \circ \pi_{10}, C_{receiver} = (X_2, Y_2) \\ \pi_9 &= PoK\{(r_4) : X_2 = g^{r_4} \bigwedge Y_2 = upk^{r_4} pk_r\} \\ \pi_{10} &= PoK\{(pk_r) : Y_2 = upk^{r_4} pk_r \bigwedge cm^{new} = g^{g^{pk_r} h^{\rho_3}} h^{v_3}\} \end{aligned} \tag{15}$$

Where $Enc_{upk2}(pk_r) = (X_2, Y_2)$, and r_4 is random generated. We use zk-SNARK to prove π_9, π_{10}.
14. Sender send $\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6, cm', sn_1, sn_2, r_{equation}, C_{content}, C_{sender}, C_{receiver}$ to verifier.
15. Sender use public key of receiver to encrypt ρ_3, v_3 and send to receiver.

4.2 Transaction Verify Scheme

1. Verifier verify $\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6$.
2. Verifier verify $cm' \cdot g^{r_{equation}} \cdot cm^{-new} = 0, (cm^{new} = C_{content}[1])$
3. Verifier verify if sn_1, sn_2 already in SNList, if not, then record nullified sn_1, sn_2 into SNList.
4. Verifier record cm^{new} in Merkle tree node of CMList.
5. Verifier send $C_{content}, C_{receiver}, C_{sender}$ to auditor.

4.3 Transaction Audit Scheme

1. Auditor get (X, Y) from $C_{content}$, calculate $v_3 = \log_h(\frac{Y}{X^{usk^{-1}}})$, get the transaction value v_3.
2. Auditor calculate $Dec(C_{sender}), Dec(C_{receiver})$ to get public key of sender and receiver.

5 Security

Theorem 4.1. Assuming the security of ZKP, the IND-CPA security of ElGamal, and Twisted ElGamal, the above scheme is secure.

5.1 Correctness

The correctness of the ZKP is included in the assumption. And other parts of the scheme are easy to verify by simple calculations.

5.2 Auditability

The auditability of the scheme is based on the public-key encryption scheme of ElGamal protocol and Twist ElGamal protocol. The encryption scheme is described in 2.1 and 2.1.

5.3 Privacy

Lemma 4.2. Assume the adaptive zero-knowledge property of NIZK and the IND-CPA security of ElGamal and Twisted ElGamal, the above scheme satisfies the privacy property.

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

Game 0. This game corresponds to the Game 0 in Experiment 0.

Game 1. Game 1 is the same as Game 0 except in the challenge phase, the difference is that zero-knowledge proof $\pi_1, \pi_2, \pi_3, \pi_4$ is generated by simulator simulation. If the advantage of $\mathcal{A}$ winning in Game 0 and Game 1 has difference, then we can build a scheme to break the zero-knowledge property. So we have:

$$|\Pr[S_1] - \Pr[S_0]| \leq negl(\lambda) \tag{16}$$

Similar to Game 0, we use$\Pr[S_1^v], \Pr[S_1^s], \Pr[S_1^r]$ to describe the advantage of $\mathcal{A}$ to guess v, pk_s, pk_r. We know that $\Pr[S_1] = \max\{\Pr[S_1^v], \Pr[S_1^s], \Pr[S_1^r]\}$

Game 2. Game 2 is a part of Game 1. In Game 2, $\mathcal{CH}$ makes a random guess for the index of target pk, i.e., randomly picks an index $j \in (T - T_{corrupt})$. If $\mathcal{A}$ makes an extraction query of pk_j in the pre query stage, or picks $pk \neq pk_j$ in the query stage, then $\mathcal{CH}$ abort. Let W be the event that $\mathcal{CH}$ does not abort. We can get $\Pr[W] = 1/(T - T_{corrupt})$. $\mathcal{A}$'s view in Game 0 is identical to that in Game 1. Then we have:

$$\begin{aligned} \Pr[S_2^s] &= \Pr[S_1^s] \cdot \Pr[W] \\ \Pr[S_2^r] &= \Pr[S_1^r] \cdot \Pr[W] \end{aligned} \tag{17}$$

Game 3. Game 3 is a part of Game 1. The difference is that Game 3 random generate r_1 to replace $cm = g^{g^{pk} h^{\rho}} h^v$.

We know that the ρ are random generated in $cm = g^{g^{pk} h^{\rho}} h^v$, so cm is random distributed as r_1,so cm is indistinguishable with r_1, we have:

$$|\Pr[S_3] - \Pr[S_1^v]| \leq negl(\lambda) \tag{18}$$

Game 4. Game 4 is a part of Game 2. The difference is that Game 4 random generate r_2 to replace $cm = g^{g^{pk_r}h^{\rho}}h^v$. Similar to Game 3, r_2 is indistinguishable with cm because of the randomness of ρ. So we have:

$$|\Pr[S_4] - \Pr[S_2^r]| \leq negl(\lambda) \tag{19}$$

Game 5. Game 5 is a part of Game 2. The difference is that Game 5 random generate r_3 to replace $sn = pk_s h^{\rho}$. Similar to Game 3 and 4, r_3 is indistinguishable with sn because of the randomness of ρ. So we have:

$$|\Pr[S_5] - \Pr[S_2^s]| \leq negl(\lambda) \tag{20}$$

Game 6. Game 6 is the sum of Game 3,4,5. Considering the message sent to Verifier. In $SumPara = (\pi_1, \pi_2, \pi_3, \pi_4, cm', sn_1, sn_2, r_{equation}, C_{content}, C_{sender}, C_{receiver})$, we have already use simulation value to replace ZKP π, and we use random value to replace commitment and nullifier cm, sn, then $r_{equation}$ is randomly generated. Now we remain the privacy of $C_{content}, C_{sender}, C_{receiver}$ to be proven.

Lemma 4.3. If encryption scheme ElGamal and Twisted ElGamal are IND-CPA secure, then we can get $\Pr[S_6] \leq negl(\lambda)$.

Proof. If $\mathcal{A}$ wins in Game 6 in a negligible advantage, then we can build a simulator $\mathcal{B}$ to break the IND-CPA security of ElGamal or Twisted ElGamal scheme. The simulator simulation runs Game 6 as follows.

1. generation phase: $\mathcal{B}$ run $pp \leftarrow Setup(1^{\lambda})$
2. pre query phase: $\mathcal{A}$ adaptively makes queries to below oracles finitely:
   $\mathcal{O}_P$: $\mathcal{B}$ invoke $CreateAddress$ to get (sk, pk), then move sk, pk to the keys list T, then return undisclosed pk to $\mathcal{A}$.
   $\mathcal{O}_S$: $\mathcal{B}$ move sk, pk to $T_{corrupt}$ and return sk to $\mathcal{A}$.
   $\mathcal{O}_V$: $\mathcal{B}$ check if pk in $T_{corrupt}$, then invoke $CreateTran$ and return $SumPara$ to $\mathcal{A}$.
   $\mathcal{O}_{RT}$ $\mathcal{B}$ get (cm, s) or (sn, s) and return correspond (ρ, pk_r, v) or (ρ, sk_s) to $\mathcal{A}$.
3. query phase: $\mathcal{B}$ invoke $CreateAddress$ and get (pk_r, sk_s), then random generate ρ and generate v, finally invoke $CreateTran, CreateAudit$ to get $SumPara$ and send to $\mathcal{A}$.
4. guess phase: $\mathcal{A}$ output ${pk_r}', {pk_s}', v'$, $\mathcal{B}$ send pk_r', pk_s' to ElGamal encryption scheme, and send v' to Twist ElGamal encryption scheme.

We know that $\mathcal{A}$'s observation in Game 6 is equivalence distribute with $\mathcal{B}$, so the probability of $\mathcal{A}$ succeeding in Game 6 is the same as the probability of $\mathcal{A}$ succeeding in $\mathcal{B}$. Game 6 and simulation of $\mathcal{B}$ are PPT algorithms, so the advantage of $\mathcal{A}$ win in Game 6 is the same as $\mathcal{B}$ win in ElGamal encryption scheme or Twist ElGamal encryption scheme. This prove Lemma 4.3.

Now we can get:

$$\Pr[S_6] \leq negl(\lambda) \tag{21}$$

To sum up, we prove Lemma 4.2.

5.4 Soundness

The soundness of cm, sn are based on the pseudorandom function.

Lemma 4.4. If PRF is collision-resistant, the scheme satisfies soundness.

In our scheme, all the data be recorded in blockchain is cm, sn, so if $\mathcal{A}$ do not follow the rule of $CreateTran$ want to pass $VerifyTran$, he must generate $cm^*, sn^*(cm^* \neq cm, sn^* \neq sn)$ satisfy equation 4. Consider the soundness of cm:

if the $\mathcal{A}$ can forge cm, sn, then we can build a simulator $\mathcal{B}$ to break the collision-resistant of PRF:

1. generation phase: $\mathcal{B}$ run $pp \leftarrow Setup(1^\lambda)$
2. query phase: $\mathcal{B}$ invoke $CreateTran$, calculate $cm = g^{PRF^k(pk,\rho)}h^v$ and send cm to A
3. guess phase: $\mathcal{A}$ send cm^* to $\mathcal{B}$.

In the security proof in privacy, we prove that $\mathcal{A}$ can not forge $cm^* = cm$, so $cm^* \neq cm$, we know that the Pedersen Hash function is collision-resistant for fixed-length input, so we get $PRF^k(pk^*, \rho^*) = PRF^k(pk, \rho)$. So if $\mathcal{A}$ can win the game in 4 in a non-negligible probability, $\mathcal{B}$ can break the collision-resistant of PRF.

The soundness proof of sn is similar to the cm above. Then we prove Lemma 4.4.

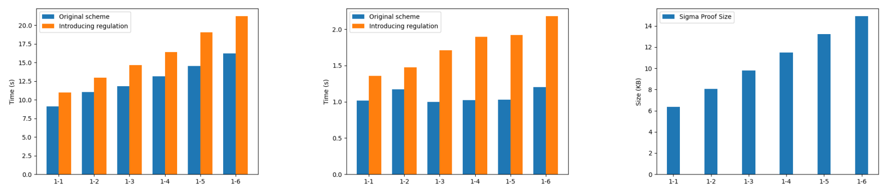
6 Performance

We now give a prototype implementation of ASOZ in JavaScript and circom mainly based on circomlibjs[15], and collect the benchmarks on Intel i7-12700H CPU (2.30GHz) and 16GB of RAM. The source code of ASOZ is publicly available at Github [1].

Our scheme is implemented on Jubjub curve [8], and the space for generating our random numbers is 253 bits. We employ the Poseidon [13] as the construction function for the Merkle tree. We choose Groth16 [14] as our zk-SNARK's proving system.

The proof generation time, verification time, and transmission cost of our scheme are illustrated in Figures 2. After surveying the ZCash explorer[3], we noted that cryptocurrency transactions commonly manifest as either 1 input to n outputs or n inputs to 1 output, with the former being more prevalent. The most common scenario is the 1 input-1 output transaction. Based on these observations. We test the performance of 1-1 to 1-6 transactions. In 1-1 transaction, the auditing functionality introduces 21% increment in proof generation time and 33% rise in verification time compared to the original scheme. The transmission cost on Sigma protocol is 6.5 KB. The results indicate that the auditing cost introduced by our scheme falls within an acceptable range, which follows the Out-of-band Cost principle.

Besides, we would like to emphasize that, due to our implementation in JavaScript, which is relatively inefficient, there is considerable room for improvement in the computational speed of the Sigma protocol.



(a) Comparison of Proof Generating Time (b) Comparison of Verify Time (c) Transmission Cost

Fig. 2. Comparison of Execution Cost Before and After Audit Introduction

7 Further Discussion

How to balance privacy and auditability remains a big challenge to decentralized payment systems. In this paper, we give our solution by using an auditing agency. The agency can open the transaction and trace illegal transactions. Furthermore, we propose security properties and design principles to refine our solution. Our solution remains open to further exploration and research.

Is there a better way to construct commitment and nullifier? In our scheme, we design a sigma-protocol-friendly construction to reduce out-of-band cost, but we still use zk-SNARK in some parts, leading to significant computational cost. We think that apart from the membership proof in the Merkle tree, there is no necessity to employ the relatively inefficient zk-SNARK in the remaining portions.

Using key agreement and key derivation. In our scheme, user and audit reuse the public keys, which may lead to a decrease in security strength. Maybe we can consider referencing Sapling and Orchard versions of ZCash, incorporating key agreement and key derivation mechanisms to avoid the reuse of keys.

Rethinking audit capabilities. In our scheme, we define a trusted auditing agency, which follows the Full transaction audit principle, but this design undermines the decentralized idea to some extent. One reasonable enhancement is to decentralize auditor authority through the use of secret sharing mechanisms.

References

1. https://github.com/AwakeLithiumFlower/ASOZ
2. Coinjoins - learn about bitcoin collaborative transactions, https://www.coinjoins.org/
3. nighthawk apps: Zcash explorer - search the zcash blockchain, https://zcashblockexplorer.com/
4. Ben Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474 (2014). https://doi.org/10.1109/SP.2014.36

5. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 315–334 (2018). https://doi.org/10.1109/SP.2018.00020
6. Chatzigiannis, P., Baldimtsi, F., Chalkias, K.: Sok: Auditability and accountability in distributed payment systems. In: Applied Cryptography and Network Security: 19th International Conference, ACNS 2021, Kamakura, Japan, June 21–24, 2021, Proceedings, Part II. p. 311–337. Springer-Verlag, Berlin, Heidelberg (2021). https://doi.org/10.1007/978-3-030-78375-4$_1$3, https://doi.org/10.1007/978-3-030-78375-4_13
7. Chen, Y., Ma, X., Tang, C., Au, M.H.: Pgc: Decentralized confidential payment system with auditability. In: Chen, L., Li, N., Liang, K., Schneider, S. (eds.) Computer Security – ESORICS 2020. pp. 591–610. Springer International Publishing, Cham (2020)
8. Daira Hopwood, Sean Bowe, T.H.N.W.: Zcash protocol specification, version 2022.3.8 [nu5] (2022), https://zips.z.cash/protocol/canopy.pdf
9. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6), 644–654 (1976). https://doi.org/10.1109/TIT.1976.1055638
10. Elgamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4), 469–472 (1985). https://doi.org/10.1109/TIT.1985.1057074
11. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology — CRYPTO' 86. pp. 186–194. Springer Berlin Heidelberg, Berlin, Heidelberg (1987)
12. Garman, C., Green, M., Miers, I.: Accountable privacy for decentralized anonymous payments. In: Grossklags, J., Preneel, B. (eds.) Financial Cryptography and Data Security. pp. 81–98. Springer Berlin Heidelberg, Berlin, Heidelberg (2017)
13. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A new hash function for Zero-Knowledge proof systems. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 519–535. USENIX Association (Aug 2021), https://www.usenix.org/conference/usenixsecurity21/presentation/grassi
14. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016. pp. 305–326. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
15. iden3: iden3, https://github.com/iden3
16. Inc, C.: The chainalysis 2023 crypto crime report, https://go.chainalysis.com/2023-crypto-crime-report.html
17. Jeong, G., Lee, N., Kim, J., Oh, H.: Azeroth: Auditable zero-knowledge transactions in smart contracts. IEEE Access **11**, 56463–56480 (2023). https://doi.org/10.1109/ACCESS.2023.3279408
18. Kang, H., Dai, T., Jean-Louis, N., Tao, S., Gu, X.: Fabzk: Supporting privacy-preserving, auditable smart contracts in hyperledger fabric. In: 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). pp. 543–555 (2019). https://doi.org/10.1109/DSN.2019.00061
19. Lab, M.R.: Monero research lab (mrl) — monero - secure, private, untraceable, https://www.getmonero.org/resources/research-lab/
20. Li, Y., Yang, G., Susilo, W., Yu, Y., Au, M.H., Liu, D.: Traceable monero: Anonymous cryptocurrency with enhanced accountability. IEEE Trans-

actions on Dependable and Secure Computing **18**(2), 679–691 (2021). https://doi.org/10.1109/TDSC.2019.2910058
21. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Decentralized business review (2008)
22. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) Advances in Cryptology — CRYPTO '91. pp. 129–140. Springer Berlin Heidelberg, Berlin, Heidelberg (1992)
23. Sander, T., Ta-Shma, A.: Auditable, anonymous electronic cash. In: Wiener, M. (ed.) Advances in Cryptology — CRYPTO' 99. pp. 555–572. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
24. Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptol. **4**(3), 161–174 (jan 1991). https://doi.org/10.1007/BF00196725, `https://doi.org/10.1007/BF00196725`
25. Wüst, K., Kostiainen, K., Capkun, V., Capkun, S.: Prcash: Fast, private and regulated transactions for digital currencies. Cryptology ePrint Archive, Paper 2018/412 (2018), `https://eprint.iacr.org/2018/412`

A Missing Proofs and Protocols

A.1 The Proof of Sigma Protocol

Sigma protocol of π_8 is an example of 2 input-1 output transaction. In order to give a general proof, we define the following problem in n input-m output transaction:

$$\pi_{11} = PoK\{(sk_s) : Y_1 = upk^{r_3} g^{sk_s} \bigwedge sn_1 = g^{sk_s} h^{\rho_1} \bigwedge \ldots \bigwedge sn_n = g^{sk_s} h^{\rho_n}\} \tag{22}$$

The Sigma protocol of π_{11} is similar to π_8.

Sigma protocol of π_{11}:

1. P random generate $a, a_2, b_1, \ldots, b_n$. send $A = g^a, A_2 = upk^{a_2}, B_i = h^{b_i}$ to V
2. V random generate e, send e to P as challenge
3. P calculate $y = a + e \cdot sk_s, y_2 = a_2 + e \cdot r_3, z_i = b_i + e \cdot \rho_i (i = 1, \ldots, n)$, send $y, y_2, z_1, \ldots, z_n$ to V, V check the following equations:

$$\begin{aligned} g^y upk^{y_2} &= A(Y_1)^e A_2 \\ g^y h^{z_i} &= A(sn_i)^e B_i \end{aligned} \tag{23}$$

Then we give the security proof of the protocol above:

Prefect Completeness This is obvious from simple calculation.

Special Soundness Fix the initial message $(A, A', B_1, \ldots, B_n)$, suppose there are two accepting transcripts (e, y, y_2, z_i) and (e', y', y_2', z_i'). We have $sk_s = (y - y')/(e - e'), r_3 = (y_2 - y_2')/(e - e'), \rho_i = (z_i - z_i')/(e - e')$. So if P can answer with probability greater than $1/(2^t)$ then P is consider to know sk_s, r_3, rho_i

Special HVZK For a fixed challenge e, the simulator $\mathcal{S}$ works as below: picks y, y_2, z_i randomly, we can calculate equation23, let A to be equal to k ,then $A_2 = g^y upk^{y_2}/(k \cdot Y_1^e), B_i = g^y h^{z_i}/(k \cdot sn_i^e)$. Which means that an accepting transcript $(A, A_2, B_i, e, y, y_2, zi)$ is distributed exactly like a real execution where V sends e.