MANTA: Privacy Preserving Decentralized Exchange

Shumo Chu[1,2], Qiudong Xia[3,2], and Zhenfei Zhang[4]

[1] UC Santa Barbara
[2] Manta Network
shumo@manta.network
[3] University of Science and Technology of China
xqd95@mail.ustc.edu.cn
[4] zhenfei.zhang@hotmail.com

Abstract

Cryptocurrencies and decentralized ledger technology has been widely adopted over the last decades. However, there isn't yet a decentralized exchange that protects users' privacy from end to end. In this paper, we construct the first ledger-based decentralized token exchange with strong privacy guarantees. We propose the first *Decentralized Anonymous eXchange* scheme (DAX scheme) based on automated market maker (AMM) and zkSNARK and present a formal definition of its security and privacy properties.

1 Introduction

In an ideal world, cryptocurrencies bring a decentralized token economy while protecting users' privacy. The widely used cryptocurrencies on permissionless consensus protocols, such as Bitcoin [11], Ethereum [14], and Polkadot [1], are pseudo-anonymous: although there are no real-world identities explicitly attached to public keys, the transaction history is public. Users' identities can still be revealed by link analysis on transaction history [10, 15, 13, 7]. Privacy-preserving cryptocurrencies such as Zcash [2] provide a decentralized and anonymous payment (DAP) scheme. Zcash uses zero-knowledge Succinct Non-interactive ARguments of Knowledge (zkSNARKs) and a consensus protocol similar to Bitcoin. However, having DAPs is not enough. To ensure a privacy preserving token economy, we need to ensure that the token exchange process, an integral part of token economy, will not leak the users' privacy as well.

In this paper, we propose a *Decentralized Anonymous eXchange* (DAX) scheme, which formally captures the functionality and security guarantees of a decentralized exchange based on automated market maker (AMM). We provide a construction of this primitive and prove its security under standard cryptographic assumptions. This construction leverages recent advancements on zero-knowledge proofs for verifiable computation, especially, the zkSNARKs

[6, 12, 3, 9]. Specifically, this construction consists of a mint mechanism that converts base coins to be exchanged to private coins, and a decentralized exchange mechanism that trades private coins anonymously.

We plan to build Manta, an instantiation of this DAX scheme, on Parity Substrate, a virtual machine that interoperates with DOT (Polkadots' native token), and many parachain assets on Polkadot, including stable coins.

We note that Manta is a framework that is independent of the underlying zero-knowledge prove system, i.e., the framework may be optimized for verifier efficiency via Groth16 [9], which best suits blockchain uses cases with a sub 5 seconds block generation time; and may also be instantiated for other proving systems that aim for none trusted setups or quantum resistance. Therefore, we do not report any concrete performance parameters at the current stage, and leave concrete implementations to future work. We do, however, note that when instantiated with Groth16 and BLS curves, we will achieve a constant proof size of 196 bytes and a sub 10 ms verification time.

In the following part of this paper, we first summarize the technology that underlies building blocks of our overall scheme: zero-knowledge proofs and zkSNARKs in section 2. We present our mint mechanism for converting base coins to private coins in section 3 and our decentralized exchange scheme for private coins in section 4, respectively, and anaylze the security in section 5. Finally, section 6 concludes this paper.

2 Background: Zero-knowledge Proofs and zkSNARKs

In cryptography, a zero-knowledge proof protocol is a proof system, in which one party, a.k.a Alice (the prover), proves to another party, a.k.a Bob (the verifier), that she knows a value x, without conveying any information apart from the fact she knows x. By the seminal work from Goldwasser, Micali, and Rackoff [8], we know that this notion of knowledge can be generalized to any NP statement.

In recent years, emerging from a pure theoretical concept, zero-knowledge proof systems have become practical, thanks to many great work in this space [9, 6, 3, 12]. The major cryptographic primitive used in this paper is a typical kind of Succinct Non-interactive ARgument of Knowledge (SNARK): publicly-verifiable preprocessing zero-knowledge SNARK, or zkSNARK for short. We informally define zkSNARK as follows.

For a finite field $\mathbb{F}$, an $\mathbb{F}$-arithmetic circuit, where the inputs, outputs and intermediate values are all in $\mathbb{F}$. We consider circuits that have an input $x \in \mathcal{F}^n$ and an auxiliary input $a \in \mathcal{F}^h$, namely a *witness*. We define arithmetic circuit satisfiability as follows:

Definition 1. *The arithmetic circuit satisfiability problem of an $\mathcal{F}$-arithmetic circuit C: $\mathcal{F}^n \times \mathcal{F}^h \to \mathcal{F}^l$ is captured by the relation $\mathcal{R}_C = \{(x, a) \in \mathcal{F}^n \times \mathcal{F}^h : C(x, a) = 0^l\}$, where the language $\mathcal{L}_C = \{x \in \mathcal{F}^n : \exists a \in \mathcal{F}^h$ s.t. $C(x, a) = 0^l\}$.*

Given a field $\mathcal{F}$, a zkSNARK for $\mathcal{F}-$arithmetic circuit satisfiability is defined by a triple of a polynomial-time algorithms (`KeyGen`, `Prove`, `Verify`):

1. $\texttt{KeyGen}(1^\lambda, C) \rightarrow (\texttt{pk}, \texttt{vk})$. Taken a security parameter λ (e.g. 128 bits) and an $\mathcal{F}$-arithmetic circuit C, KeyGen probabilistically samples a *proving key* pk and a *verification key* vk. Both keys are published as public parameters and can be used for any number of times, to prove/verify the memberships in $\mathcal{L}_C$.
2. $\texttt{Prove}(\texttt{pk}, x, a) \rightarrow \pi$. Taken a proving key pk and any $(x, a) \in \mathcal{R}_C$ as input, the *prover* Prove outputs a non-interactive proof π for the statement $x \in \mathcal{L}_C$.
3. $\texttt{Verify}(\texttt{vk}, x, \pi) \rightarrow b$. Taken a verification key vk, x , and a proof π as input, the *verifier* Verify outputs 1 if it is convinced that $x \in \mathcal{L}_C$, and outputs 0 otherwise.

A zkSNARK satisfies the following properties:

Completeness. For every security parameter λ, any $\mathcal{F}-$arithmetic circuit C, and any $(x, a) \in \mathcal{R}_C$, an honest prover must convince the verifier. Namely, output 1 with probability $1 - \text{negl}(\lambda)$ with the following: $(\texttt{pk}, \texttt{vk}) \leftarrow \texttt{KeyGen}(1^\lambda, C)$, $\pi \leftarrow \texttt{Prove}(\texttt{pk}, x, a)$, $b \leftarrow \texttt{Verify}(\texttt{vk}, x, \pi)$.

Soundness. If the verifier accepts a proof from a bounded prover, then the prover must know the secret input corresponding to the witness of the given instance [5].

Succinctness. An honestly-generated proof π has $O_\lambda(1)$ bits and $\texttt{Verify}(\texttt{vk}, x, \pi)$ runs in time $O_\lambda(|x|)$ (O_λ hides a fixed polynomial factor in λ).

Zero knowledge. An honestly-generated proof is perfect zero knowledge: there exists a $\text{poly}(\lambda)-$size simulator Sim, who has no access to the secret inputs, can generate a simulated proof, such that all stateful $\text{poly}(\lambda)-$bounded distinguishers $\mathcal{D}$ cannot distinguish this proof from an honest proof.

3 Manta$_{\texttt{DAP}}$: Decentralized Anonymous Payment

In this section, we present Manta$_{\texttt{DAP}}$, a decentralized anonymous payment (DAP). While this construction is similar to Zcash, the difference is that the base coins can be either DOT, Polkadot's native currency or a parachain asset.This DAP scheme supports both minting and forfeit, therefore, allows for bidirectional transfers between private coins and base coins.

In addition to zkSNARK, we use the following cryptographic primitives:

- COMM, a non-interactive commitment scheme that is both hiding and binding. For example, given a random seed r and a message m, the commitment is $c := \texttt{COMM}_r(m)$. c can be *opened* by revealing r and m, which verifies the commitment.
- pseudorandom functions. More specifically, we use three labeled pseudorandom functions that may be instantiated from a same core function. For a seed x, we derive $\texttt{PRF}_x^{addr}(\cdot)$, $\texttt{PRF}_x^{sn}(\cdot)$, and $\texttt{PRF}_x^{pk}(\cdot)$, which will be used to generate payment addresses and serial numbers.

[5] The formal definition of soundness is based on the concept of extractor, which can be found in [4].

Addresses

A user u generates an address key pair $(a_{\mathtt{pk}}, a_{\mathtt{sk}})$. The coins of u can be only spent with the knowledge of $a_{\mathtt{sk}}$. To generate a key pair, u randomly samples a secret from the domain $a_{\mathtt{sk}} \xleftarrow{\$} 1^{\lambda}$, and sets $a_{\mathtt{pk}} := \mathtt{PRF}^{addr}_{a_{\mathtt{sk}}}(0)$. A user could generate and use any number of key pairs.

Mint private coins

To mint a private coin with value v, a user u needs to initiate a coin minting transaction tx_{mint} with the deposit of a base coin with value v[6]. To mint a new coin, a user generates and submits tx_{mint} to the ledger as the following:

1. u samples a random number $\rho \xleftarrow{\$} 1^{\lambda}$, which is a secret value that determines the coins serial number $\mathtt{sn} := \mathtt{PRF}^{sn}_{a_{\mathtt{sk}}}(\rho)$. Note that this $\mathtt{sn}$ is not included in tx_{mint}.
2. u commits to the triple $(a_{\mathtt{pk}}, v, \rho)$ in two phases: (a) sample a random r, and compute $k := \mathtt{COMM}_r(a_{\mathtt{pk}}||\rho)$; then (b) sample a random s, and compute $\mathtt{cm} := \mathtt{COMM}_s(v||k)$.
3. u thus mints a private coin $c := (a_{pk}, v, \rho, r, s, \mathtt{cm})$ and a mint transaction $tx_{mint} := (v, k, s, \mathtt{cm})$.
4. the ledger adds $\mathtt{cm}$ to the merkle tree that represents the ledger state rt.

This design allows anyone to verify $\mathtt{cm}$ in tx_{mint} with value v but doesn't disclose the address of the owner $(a_{\mathtt{pk}})$ or the serial number. Therefore, tx_{mint} is accepted by the ledger if the user deposits a base coin of value v.

Transfer private coins

Private coins can be transferred and spent using the `transfer` operation, which takes a set of input private coins to be consumed, and transfers their total value into a set of new output coins: the total value of output coins equals the total value of the input coins.

For example, suppose a user u, with address key pair $(a^{\mathtt{old}}_{\mathtt{pk}}, a^{\mathtt{old}}_{\mathtt{sk}})$, tries to transfer or spend his old coin $c^{\mathtt{old}} = (a^{\mathtt{old}}_{\mathtt{pk}}, v^{\mathtt{old}}, \rho^{\mathtt{old}}, r^{\mathtt{old}}, s^{\mathtt{old}}, \mathtt{cm}^{\mathtt{old}})$ and produce a new coin $c^{\mathtt{new}}$ that targets at address $a^{\mathtt{new}}_{\mathtt{pk}'}$ [7]. To create a `transfer` transaction, u samples a trapdoor $\rho^{\mathtt{new}}$ and compute $k^{\mathtt{new}} := \mathtt{COMM}_{r^{\mathtt{new}}}(a_{\mathtt{pk}'}||\rho^{\mathtt{new}})$ with a random $r^{\mathtt{new}}$, and then compute $\mathtt{cm}^{\mathtt{new}} := \mathtt{COMM}_{s^{\mathtt{new}}}(k^{\mathtt{new}}||s^{\mathtt{new}})$ with a random $s^{\mathtt{new}}$. This creates a new coin $c^{\mathtt{new}} := (a^{\mathtt{new}}, v^{\mathtt{new}}, \rho^{\mathtt{new}}, r^{\mathtt{new}}, s^{\mathtt{new}}, \mathtt{cm}^{\mathtt{new}})$. The user u also produces a zkSNARK proof $\pi_{\mathtt{transfer}}$ for the following $\mathtt{NP}$ statement, which is called `transfer`:

---

[6] Here we assume a 1 : 1 exchange ratio. A minor transaction fee e.g. 0.1% could be charge when minting private coins.

[7] Note that the corresponding $a^{\mathtt{new}}_{\mathtt{sk}'}$ is not present in `transfer`. This implies that the address could belong to u, or some other user.

Definition 2 (NP Statement of transfer). *Given an accumulator acc that represents the ledger state, serial number* $\mathtt{sn}^{old}$*, and coin commitment* $\mathtt{cm}^{new}$*, "I" know coins* c^{old}*,* c^{new}*, and secret key* a_{sk}^{old} *such that:*

- *Both* c^{old} *and* c^{new} *are* well formed*: for example,* $k^{old} = \mathtt{COMM}_{r^{old}}(a_{pk}^{old}||\rho^{old})$ *and* $\mathtt{cm}^{old} = \mathtt{COMM}_{s^{old}}(v^{old}||k^{old})$*.*
- *The address and the secret key derive the public key:* $a_{pk}^{old} = \mathtt{PRF}^{addr}_{a^{sk}_{old}}(0)$*.*
- *The old coin's commitment appears is a member of acc (*$cm^{old} \in acc$*).*
- *The old coin and the new coin have the same value:* $v^{new} = v^{old}$

The user u sends a transfer transaction $tx_{\mathtt{transfer}} := (acc, \mathtt{sn}^{\mathrm{old}}, \mathtt{cm}^{\mathrm{new}}, \pi_{\mathtt{transfer}})$ to the ledger. The ledger checks the validity of the transaction: the transaction is valid only if $\mathtt{sn}^{\mathrm{old}}$ has never been used in a previous transaction in the ledger (otherwise it is a double spend), and the zkSNARK verifier verifies the validity of $\pi_{\mathtt{transfer}}$.

We make two remarks. First, the proof does not specify which $\mathtt{cm}^{\mathrm{old}}$ the coin is transferred from. Instead, it proves the existences of such a coin. This breaks the link between old and new commitments, and is a key requirement for anonymity. Second, a `transfer` does not update the state of the accumulator. In deployment, there will be a block proposer, who collects a set of `transfer`s, validates them, and updates the accumulator in batch . Block validators therefore need to check both the validity of the set of `transfer`s, as well as the fact that the ledger updates is a correct result of applying those `transfer`s.

Stop double spending

Manta prevents double spending by binding the serial numbers with commitments and enforces that each `transfer` transaction has a unique serial number. Concretely, the ledger (a.k.a the consensus protocol) maintains two lists, represented by two accumulators, namely, acc_{all} and acc_{spend}. acc_{all} contains all commitments that have ever appeared; while acc_{spend} contains `sn`s for all spend `transfer`s. When generating a new `transfer`, the sender needs to prove that (the commitment of) the coin it is about to spend is in acc_{all} and (the `sn` of) the coin is not in acc_{spend}. When the `transfer` is accepted, $\mathtt{sn}^{old}$ will be released and added to acc_{spend}.

Note that this creates a link between a new commitment $\mathtt{cm}^{new}$ and an $\mathtt{sn}^{old}$ since they both appear in a same `transfer`. This, however, do not break the anonymity, since our commitment scheme is hiding.

Claim public coins from private coins.

This construction so far allows us to transfer the value from a private coin to another with potentially a new address. A natural question is: how can a user claim public coins from private coins? We make two simple modifications to the transfer operations so that we can split or merge private coins, and transfer them back to public coins. The first modification is to allow a `transfer` operation with multiple input coins and output coins. This enables splitting and merging private coins. The second modification is to add an optional public output in the output coins. As a result, the last invariant in the NP Statement of `transfer` (Definition 2) becomes "$v_1^{\texttt{new}} + v_2^{\texttt{new}} + v_{\texttt{pub}} = v_1^{\texttt{old}} + v_2^{\texttt{old}}$"[8].

With the modification of public output, we also need to guarantee that `transfer` operation is non-malleable. During the `transfer` operation, the user u also needs to sign the entire transaction:

1. samples a key pair $(\texttt{pk}_{\texttt{sig}}, \texttt{sk}_{\texttt{sig}})$ for a one-time signature scheme.
2. computes $h_{\texttt{Sig}} := \texttt{CRH}(\texttt{pk}_{\texttt{sig}})$.
3. computes the hash for two input coins, $h_1 := \texttt{PRF}^{\texttt{pk}}_{a^{\texttt{old}}_{\texttt{sk},1}}$, $h_2 := \texttt{PRF}^{\texttt{pk}}_{a^{\texttt{old}}_{\texttt{sk},2}}$.
4. add $h_{\texttt{sig}}$, h_1, and h_2 into the NP statement of `transfer` and prove that h_1 and h_2 are computed correctly.
5. use $\texttt{sk}_{\texttt{sig}}$ to sign the entire `transfer` operation. This will produce a signature σ. The transaction $tx_{\texttt{transfer}}$ should include both σ and $\texttt{pk}_{\texttt{sig}}$.

4 MANTA$_{\texttt{DAX}}$: Decentralized Anonymous Exchange

In this section, we describe MANTA$_{\texttt{DAX}}$, a *Decentralized Anonymous eXchange* (DAX) scheme that extends the DAP scheme (section 3) to support AMM [9] style swap. In principle, our idea could extend to other kind of decentralized exchange as well, we choose AMM for its elegant simplicity.

Ledger State of MANTA$_{\texttt{DAX}}$

Without loss of generality, we assume that this MANTA$_{\texttt{DAX}}$ scheme can exchange two kinds of private coins p`ACoin`, and p`BCoin`, each of which is constructed using the DAP scheme in section 3 (It is trivial to extend this scheme to multiple private coins).

To support the exchange of private coins, we extend each ledger to support a special coin, namely, $\texttt{DEXCoin}_i$ ($i \in \{A, B\}$). For example, $\texttt{DEXCoin}_A$ and is supported by the ledger of p`ACoin` (referred to as $L_{\texttt{A}}$) and $\texttt{DEXCoin}_B$ is supported by the ledger of p`BCoin` (referred to as $L_{\texttt{B}}$). A $\texttt{DEXCoin}_i$ is a tuple that consists of a serial number `sn` and a value v, i.e, $c_{\texttt{DEX}} := (\texttt{sn}, v)$. The $\texttt{DEXCoin}_i$ can only be spent by the decentralized exchange, a.k.a the ledger, denote by $L_{\texttt{DEX}}$, which is controlled by the consensus.

The ledger state of MANTA can be defined as a triple $\mathcal{S} = (L_{\texttt{A}}, L_{\texttt{B}}, L_{\texttt{DEX}})$:

- $L_{\texttt{A}}$: the ledger state of p`ACoin`, which is committed in an accumulator $acc_{\texttt{A}}$.
- $L_{\texttt{B}}$: the ledger state of p`BCoin`, which is committed in an accumulator $acc_{\texttt{B}}$.

[8] For simplicity, we ignore the transaction fee in this statement.

[9] AMM can be viewed a trading pair that always maintain an invariant on the balances of the assets. Please refer [5] for a detailed explaination of AMM.

- $L_{\texttt{DEX}}$: the ledger state of the exchange pair of p`ACoin` and p`BCoin`. It follows automated market maker scheme. Here, we present a simplified design first, using the "$x \times y = k$" market maker scheme [5]. $L_{\texttt{DEX}}$ consists of the following fields:
  - $\texttt{cm}_{\texttt{A}}$: a commitment of an p`ACoin`, $\texttt{cm}_{\texttt{A}}$ must be a valid $\texttt{DEXCoin}_A$ in $L_{\texttt{A}}$.
  - $\texttt{cm}_{\texttt{B}}$: a commitment of a p`BCoin`, $\texttt{cm}_{\texttt{B}}$ must be a valid $\texttt{DEXCoin}_B$ in $L_{\texttt{B}}$.
  - $v_{\texttt{A}}$: the value of $\texttt{cm}_{\texttt{A}}$.
  - $v_{\texttt{B}}$: the value of $\texttt{cm}_{\texttt{B}}$.

  As in the normal automated market maker setting, $L_{\texttt{DEX}}$ needs to maintain the invariant $v_{\texttt{A}} \times v_{\texttt{B}} = l$, where l is a constant.

The $\text{Manta}_{\texttt{DAX}}$ Protocol

A user can exchange an p`ACoin` for a p`BCoin` or vice versa, using an exchange transaction. Due to the symmetry, we only focus on exchanging a `ACoin` for a `BCoin` in this paper.

$\text{Manta}_{\texttt{DAX}}$.Prove: For example, if a user u, with address key pair $(a^{\text{old}}_{\text{pk}}, a^{\text{old}}_{\text{sk}})$, tries to exchange his old p`ACoin` $c^{\text{old}} = (\text{“pACoin”}, a^{\text{old}}_{\text{pk}}, v^{\text{old}}, \rho^{\text{old}}, r^{\text{old}}, s^{\text{old}}, \texttt{cm}^{\text{old}})$ and for a new p`BCoin`, c^{new} that targeted at address $a^{\text{new}}_{\text{pk}'}$ (the address could either belong to u or an other user). To create a transfer transaction, u samples a trapdoor ρ^{new} and compute $k^{\text{new}} := \texttt{COMM}_{r^{\text{new}}}(a_{\text{pk}'}||\rho^{\text{new}})$ with a random r^{new}, and then computes $\texttt{cm}^{\text{new}} := \texttt{COMM}_{s^{\text{new}}}(k^{v^{\text{new}}}||s^{\text{new}})$ with a random s^{new}. This creates a new coin $c^{\text{new}} := (\text{“pBCoin”}, a^{\text{new}}_{\text{pk}'}, v^{\text{new}}, \rho^{\text{new}}, r^{\text{new}}, s^{\text{new}}, \texttt{cm}^{\text{new}})$. The user u also produces a zero-knowledge proof $\pi_{\texttt{exchange}}$ for the following `NP` statement:

Definition 3 (`NP` Statement of $\pi_{\texttt{exchange}}$). *Given an accumulator of ledger A acc_A, a serial number $\texttt{sn}^{old}$, and a coin commitment $\texttt{cm}^{new}$, "I" know coins c^{old}, c^{new}, and a secret key a^{old}_{sk} such that:*

- *Both c^{old} and c^{new} are* well formed*: for example, $k^{old} = \texttt{COMM}_{r^{old}}(a^{old}_{pk}||\rho^{old})$ and $\texttt{cm}^{old} = \texttt{COMM}_{s^{old}}(v^{old}||k^{old})$.*
- *The address secret key matches the public key: $a^{old}_{pk} = \texttt{PRF}^{addr}_{a^{sk}_{old}}(0)$.*
- *$\texttt{cm}^{old} \in acc_A$.*
- *This trading pair didn't use all liquidity: $v_A > v_{old} \wedge v_B > v_{new}$*
- *After exchange, the invariant of L_{DEX} remains valid: $(v_A + v^{old}) \times (v_B - v^{new}) = v_A \times v_B$*

This proof is generated by $\text{Manta}_{\texttt{DAX}}$.`Prove`, more specifically $(\pi_{\texttt{exchange}}, \texttt{c}^{\text{new}}) := \text{Manta}_{\texttt{DAX}}.\texttt{Prove}(pk, acc_A, \texttt{sn}^{\text{old}}, a^{\text{old}}_{\text{sk}}, \texttt{c}^{\text{old}})$. The user u sends an exchange transaction $tx_{\texttt{exchange}} := (acc_A, \texttt{sn}^{\text{old}}, \texttt{cm}^{\text{new}}, \pi_{\texttt{exchange}})$ to the ledger.

$\text{Manta}_{\texttt{DAX}}$.Verify: The ledger checks the validity of the transaction: the transaction is valid only if $\texttt{sn}^{\text{old}}$ has never been used in a previous transaction in the $L_{\texttt{A}}$ (otherwise it is a double spend), and the zkSNARK verifier verifies the validity of $\pi_{\texttt{exchange}}$. In addition to adding cm^{old} and cm^{new} to $L_{\texttt{A}}$ and $L_{\texttt{B}}$ respectively, the ledger also mints two new $\texttt{DEXCoins}$: $\texttt{DEXCoin}_A$ with value $v_{\texttt{A}} + v^{\texttt{old}}$ and $\texttt{DEXCoin}_B$ with $v_{\texttt{B}} - v^{\texttt{new}}$, and new serial numbers to replace the old ones in $L_{\texttt{A}}$ and $L_{\texttt{B}}$.

5 Security

In this section, we formally state the security properties of $\textsc{Manta}_{\texttt{DAX}}$. Note that the ledger state of our DAX scheme is a triple: $\mathcal{S} = (L_A, L_B, L_{\texttt{DEX}})$. We assume that the zkSNARK uses a proving key pk and verification key vk during the trusted setup.

- **Completeness.** For any ledger state $\mathcal{S}$, a valid exchange transaction $tx_{\texttt{exchange}} = (acc_{\texttt{A}}, \texttt{sn}^{\texttt{old}}, \texttt{cm}^{\texttt{new}}, \pi_{\texttt{exchange}})$, that is :
  - $\texttt{sn}^{\texttt{old}}$ has never been used in a previous transaction in the $L_{\texttt{A}}$.
  - $(\pi_{\texttt{exchange}}, \_) := \textsc{Manta}_{\texttt{DAX}}.\texttt{Prove}(pk, acc_A, \texttt{sn}^{\texttt{old}}, a^{\texttt{old}}_{\texttt{sk}}, \texttt{c}^{\texttt{old}})$

  It holds that:

$$Pr[\textsc{Manta}.\texttt{Verify}(vk, tx_{exchange}, \mathcal{S}) = 1] = 1$$

- **Soundness.** For any PPT adversary $\mathcal{A}$, the following probability is negligible in λ (C is the zkSNARK circuit of $\textsc{Manta}_{\texttt{DAX}}$):

$$Pr\left[\begin{array}{l}(vk, pk) \leftarrow Gen(1^\lambda, C),\\ (c^{new'}, \pi'_{exchange}) \leftarrow \mathcal{A}(1^\lambda, vk, pk)\\ \text{where } c^{new'} = (\text{“pBCoin”}, a^{new'}_{pk'}, v^{new'}, \rho^{new'}, r^{new'}, s^{new'}),\\ k^{new} \leftarrow \mathsf{COMM}_{r^{new}}(a^{new'}_{pk'} \| \rho^{new'}),\\ cm^{new} \leftarrow \mathsf{COMM}_{s^{new}}(v^{new'} \| k^{new'}),\\ \mathsf{let} \quad tx'_{cxchange} = (acc_{\mathtt{A}}, sn^{\mathtt{old}}, cm^{\mathtt{new}}, \pi_{\mathtt{exchange}})\\ \mathsf{then} \quad 1 \leftarrow \textsc{Manta}_{\mathtt{DAX}}.\mathsf{Verify}(vk, tx'_{exchange}, \mathcal{S}),\\ tx'_{exchange} \neq tx_{exchange} \quad \text{for} \quad sn^{new'} \neq sn^{new}\end{array}\right]$$

- **Zero Knowledge.** An honestly-generated proof is perfect zero knowledge. For security parameter λ and (vk, pk), PPT distinguisher $\mathcal{D}$, there exists a PPT simulator Sim such that the following probabilities are indistinguishable (at most differ by $negl(\lambda)$):
  - The probability that $\mathcal{D}(cm, tx_{exhange}) = 1$ on an honest transaction ($\textsc{Manta}_{\texttt{DAX}}.\mathsf{Gen}$ is the procedure for trusted setup):

$$\Pr\left[\mathcal{D}(cm, tx_{exchange}) = 1 \;\middle|\; \begin{array}{l}(vk, pk) \leftarrow \textsc{Manta}_{\mathtt{DAX}}.\mathsf{Gen}(1^\lambda, C),\\ (sn^{old}, \rho^{new}) \leftarrow \mathcal{D}(vk, pk),\\ (cm, tx_{exchange}) \leftarrow\\ \quad \textsc{Manta}_{\mathtt{DAX}}.\mathsf{Prove}(pk, acc_A, \mathtt{sn}^{\mathtt{old}}, a^{\mathtt{old}}_{\mathtt{sk}}, \mathtt{c}^{\mathtt{old}}).\end{array}\right]$$

- The probability that $\mathcal{D}(cm, tx'_{exhange}) = 1$ on a simulated transaction:

$$pr\left[\mathcal{D}(cm, tx_{exhange}) = 1 \middle| \begin{array}{l}(vk, pk) \leftarrow \mathsf{Sim}(1^\lambda, C),\\ (sn^{old}, \rho^{new}) \leftarrow \mathcal{D}(vk, pk),\\ (cm, tx'_{exchange}) \leftarrow \\ \quad \mathsf{Sim}(pk, acc_A, \mathtt{sn}^{\mathtt{old}}, a^{\mathtt{old}}_{\mathtt{sk}}, \mathtt{c}^{\mathtt{old}}).\end{array}\right]$$

6 Conclusion

In this paper, we present Manta, a decentralized anonymous exchange scheme that ensures users' privacy while exchanging private coins. The core idea of Manta is to leverage zkSNARKs to private zero knowledge proof for an automated market maker scheme. We formally define the security and privacy properties of our scheme.

References

1. Polkadot: Decentralized web 3.0 blockchain interoperability platform. https://polkadot.network/.
2. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society, 2014.
3. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for C: verifying program executions succinctly and in zero knowledge. In *CRYPTO (2)*, volume 8043 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013.
4. Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, volume 7785 of *Lecture Notes in Computer Science*, pages 315–333. Springer, 2013.
5. Vitalik Buterin. Improving front running resistance of x*y=k market makers. https://ethresear.ch/t/improving-front-running-resistance-of-x-y-k-market-makers/1281, 2018.
6. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 626–645, 2013.
7. Steven Goldfeder, Harry A. Kalodner, Dillon Reisman, and Arvind Narayanan. When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies. *Proc. Priv. Enhancing Technol.*, 2018(4):179–199, 2018.
8. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *STOC*, pages 291–304. ACM, 1985.
9. Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2016.
10. George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn. An empirical analysis of anonymity in zcash. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 463–477. USENIX Association, 2018.

11. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf.
12. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society, 2013.
13. Florian Tramèr, Dan Boneh, and Kenny Paterson. Remote side-channel attacks on anonymous transactions. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2739–2756. USENIX Association, August 2020.
14. Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger.
15. Haaroon Yousaf, George Kappos, and Sarah Meiklejohn. Tracing transactions across cryptocurrency ledgers. In Nadia Heninger and Patrick Traynor, editors, *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pages 837–850. USENIX Association, 2019.