MANTA: Privacy Preserving Decentralized Exchange

Shumo Chu

UC Santa Barbara and Manta Network
`contact@manta.network`
`www.manta.network`

version 1.0, November 16, 2020

Abstract

Cryptocurrencies and decentralized ledger technology has been widely adopted over the last decades. However, there isn't yet a decentralized exchange that can protect users' privacy from end to end. In this paper, we construct the first ledger-based decentralized token exchange with strong privacy guarantees. We first propose a *Decentralized Anonymous eXchange* scheme (DAX scheme) based on automated market maker (AMM) and zk-SNARK. We prove that this scheme is secure and preserves users' privacy.

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 [9], Ethereum [11], 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. 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 (zk-SNARKs) and a consensus protocol similar to Bitcoin. However, having DAPs are 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 AMM. We provide a construction of this primitive and prove its security under specific cryptographic assumptions. This construction leverages recent advancements on zero-knowledge proofs for verifiable computation, especially zk-SNARKs [6,10,3,8]. Specifically, this construction consists of a mint mechanism that convert 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 ADEX scheme on Parity Substrate, a virtual machine that can interoperate with Polkadots' native token DOT, and many others tokens that issued on Polkadot, including many stable coins.

In the following part of this paper, we first introduce the key technology our scheme build upon: zero knowledge proofs and zk-SNARKs 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.

2 Background: Zero Knowledge Proofs and zk-SNARKs

In cryptography, a zero knowledge proof protocol is a proof system, in which one party, a.k.a Alice (the prover), can prove to another party, a.k.a Bob (the verifier), that she knows the value x without conveying any information apart from the fact she knows x. By the seminal work from Goldwasser, Micali, and Rackoff [7], we know that this notion of knowledge can be generalized to any NP statement.

In recent years, from pure theoretical concept, zero-knowledge proof systems have become practical, thanks to many great works in this space [8,6,3,10]. The major cryptographic primitive used in this paper is a special kind of Succinct Non-interactive ARgument of Knowledge (SNARK): publicly-verifiable preprocessing zero-knowledge SNARK, or zk-SNARK for short. We informally define zk-SNARK as follows.

For a finite field $\mathbb{F}$, a $\mathbb{F}-$arithmetic circuit takes input that are element in $\mathbb{F}$, and outputs elements in $\mathbb{F}$. We consider circuits that have an input $x \in \mathbb{F}^n$ and an auxiliary input $a \in \mathbb{F}^h$, namely a *witness*. We define arithmetic circuit satisfiability as follows:

Definition 1. *The arithmetic circuit satisfiability problem of an $\mathbb{F}$-arithmetic circuit C: $\mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^l$ is captured by the relation $\mathcal{R}_C = \{(x, a) \in \mathbb{F}^n \times \mathbb{F}^h : C(x, a) = 0^l\}$, 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 zk-SNARK for $\mathcal{F}-$arithmetic circuit satisfiability is defined by a triple of a polynomial-time algorithms (`KeyGen`, `Prove`, `Verify`):

1. $\texttt{KeyGen}(1^\lambda, C) \to (\texttt{pk}, \texttt{vk})$. 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 any number of times, to prove/verify the memberships in $\mathcal{L}_C$.
2. $\texttt{Prove}(\texttt{pk}, x, a) \to \pi$. 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. $\texttt{Verify}(\texttt{vk}, x, \pi) \to b$. Taken a verification key `vk`, x , and a proof π as input, the *verifier* `Verify` outputs 1 if it convinced that $x \in \mathcal{L}_C$, and outputs 0 otherwise.

A zk-SNARK satisfies the following properties:

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

Soundness. If the verifier accepts a proof from bounded prover, then the prover has the knowledge of witness of the given instance [1].

Succinctness. An honestly-generated proof π has $O_\lambda(1)$ bits and $\mathtt{Verify}(\mathtt{vk}, x, \pi)$ 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 is a $\mathrm{poly}(\lambda)-$size simulator $\mathtt{Sim}$ such that for all stateful $\mathrm{poly}(\lambda)-$size distinguisher $\mathcal{D}$ cannot distinguish the honest proof and the simulated proof.

3 From Base Coins to Private Coins

In this section, we present a decentralized anonymous payment (DAP) scheme that is similar to ZCash. This DAP scheme supports mint private coins from base coins, transfer private coins to either private or or base coins.

In addition to zk-SNARK, we use the following cryptographic primitives:

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

Addresses

A user u generates an address key pair (a_{pk}, a_{sk}). The coins of u can be only spent with the knowledge of a_{sk}. To generate a key pair, u random sample a secret from the domain $a_{sk} \overset{\$}{\leftarrow} 1^\lambda$, and set $a_{pk} := \mathtt{PRF}^{addr}_{a_{sk}}(0)$. 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[2]. To mint a new coin, a user generates and submits tx_{mint} to the ledger as the following:

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

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

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

This design allows anyone to verify $\texttt{cm}$ in tx_{mint} with value v but doesn't disclose the address of the owner ($a_{\texttt{pk}}$) 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, if a user u, with address key pair $(a^{\text{old}}_{\text{pk}}, a^{\text{old}}_{\text{sk}})$, try to transfer the spend his old coin $c^{\text{old}} = (a^{\text{old}}_{\text{pk}}, v^{\text{old}}, \rho^{\text{old}}, r^{\text{old}}, s^{\text{old}}, \texttt{cm}^{\text{old}})$ and produce a new coin c^{new} that targeted at address $a^{\text{new}}_{\text{pk}'}$ (the address could belong u or some 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 compute $\texttt{cm}^{\text{new}} := \texttt{COMM}_{s^{\text{new}}}(k^{v^{\text{new}}}||s^{\text{new}})$ with a random s^{new}. This creates a new coin $c^{\text{new}} := (a^{\text{new}}, v^{\text{new}}, \rho^{\text{new}}, r^{\text{new}}, s^{\text{new}}, \texttt{cm}^{\text{new}})$. The user u also produces a zk-SNARK proof $\pi_{\texttt{transfer}}$ for the following $\texttt{NP}$ statement, which call $\texttt{transfer}$:

Definition 2 ($\texttt{NP}$ Statement of $\texttt{transfer}$). *Given the Merkle-tree root rt, serial number $\texttt{sn}^{old}$, and coin commitment $\texttt{cm}^{new}$, "I" know coins c^{old}, c^{new}, and 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: $a^{old}_{pk} = \texttt{PRF}^{addr}_{a^{sk}_{old}}(0)$.*
- *The old coin's commitment appears as a leaf of a merkle tree with root rt.*
- *The old coin and the new coin have the same value: $v^{new} = v^{old}$*

The user u sends a transfer transaction $tx_{\texttt{transfer}} := (rt, \texttt{sn}^{\text{old}}, \texttt{cm}^{\text{new}}, \pi_{\texttt{transfer}})$ to the ledger. 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 ledger (otherwise it is a double spend), and the zk-SNARK verifier verifies the validity of $\pi_{\texttt{transfer}}$.

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 it can support split, merge private coins and transfer to a public coin. The first modification is to allow a `transfer` operation has multiple input coins and output coins. The 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) changes to "$v_1^{\mathtt{new}} + v_2^{\mathtt{new}} + v_{\mathtt{pub}} = v_1^{\mathtt{old}} + v_2^{\mathtt{old}}$".

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 need to sign the entire transaction:

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

4 Anonymous Decentralized Exchange for Private Coins

In this section, we describe Manta, which extends the DAP scheme in section 3 to a *Decentralized Anonymous eXchange* (DAX) scheme.

Ledger State of Manta

Without the loss of generality, we assume that this Manta scheme can exchange two kinds of private coins `ACoin`, and `BCoin`, which are constructed using the DAP scheme in section 3 (It is trivial to extend this scheme to more private coins). To support exchanging private coins, we let $L_{\mathtt{A}}$, and $L_{\mathtt{B}}$ support a special kind of coin, `DEXCoin`, that can only be spent by $L_{\mathtt{DEX}}$. However, a `DEXCoin` in $L_{\mathtt{A}}$ is still a valid `ACoin`, and `DEXCoin` in $L_{\mathtt{B}}$ is still a valid `BCoin`. A `DEXCoin` is a tuple that consists of a serial number `sn` and a value v, $c_{\mathtt{DEX}} := (\mathtt{sn}, v)$. $L_{\mathtt{DEX}}$.

The ledger state of Manta can be defined as a triple $(L_{\mathtt{A}}, L_{\mathtt{B}}, L_{\mathtt{DEX}})$:

- $L_{\mathtt{A}}$: the ledger state of `ACoin`, which is committed in a merkle root $rt_{\mathtt{A}}$.
- $L_{\mathtt{B}}$: the ledger state of `BCoin`, which is committed in a merkle root $rt_{\mathtt{B}}$.
- $L_{\mathtt{DEX}}$: the ledger state of the exchange pair of `ACoin` and `BCoin`. It follows automated market maker scheme. Here, we present a simplified design first, using the "$x \times y = k$" market maker scheme [5]. $L_{\mathtt{DEX}}$ consists of the following fields:

- $\mathtt{cm_A}$: a commitment of an `ACoin`, $\mathtt{cm_A}$ must be a valid `DEXCoin` in $L_{\mathtt{A}}$.
- $\mathtt{cm_B}$: a commitment of a `BCoin`, $\mathtt{cm_B}$ must be a valid `DEXCoin` in $L_{\mathtt{B}}$.
- $v_{\mathtt{A}}$: the value of $\mathtt{cm_A}$.
- $v_{\mathtt{B}}$: the value of $\mathtt{cm_B}$.

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

Exchange Private Coins

A user can exchange a `ACoin` for a `BCoin` or `BCoin` for `ACoin` using an exchange transaction. Since they are symmetric, we only focus on exchanging a `ACoin` for a `BCoin` in this paper.

For example, if a user u, with address key pair $(a^{\mathsf{old}}_{\mathsf{pk}}, a^{\mathsf{old}}_{\mathsf{sk}})$, tries to exchange his old `ACoin` $c^{\mathsf{old}} = (\text{“ACoin”}, a^{\mathsf{old}}_{\mathsf{pk}}, v^{\mathsf{old}}, \rho^{\mathsf{old}}, r^{\mathsf{old}}, s^{\mathsf{old}}, \mathtt{cm}^{\mathsf{old}})$ and for a new `BCoin`, c^{new} that targeted at address $a^{\mathsf{new}}_{\mathsf{pk'}}$ (the address could belong u or some other user). To create a transfer transaction, u samples a trapdoor ρ^{new} and compute $k^{\mathsf{new}} := \mathtt{COMM}_{r^{\mathsf{new}}}(a_{\mathsf{pk'}} || \rho^{\mathsf{new}})$ with a random r^{new}, and then computes $\mathtt{cm}^{\mathsf{new}} := \mathtt{COMM}_{s^{\mathsf{new}}}(k^{v^{\mathsf{new}}} || s^{\mathsf{new}})$ with a random s^{new}. This creates a new coin $c^{\mathsf{new}} := (\text{“BCoin”}, a^{\mathsf{new}}_{\mathsf{pk'}}, v^{\mathsf{new}}, \rho^{\mathsf{new}}, r^{\mathsf{new}}, s^{\mathsf{new}}, \mathtt{cm}^{\mathsf{new}})$. The user u also produces a zk-SNARK proof π_{exchange} for the following `NP` statement, which call `exchange`:

Definition 3 (`NP` Statement of `exchange`). *Given the Merkle-tree root $rt_{\mathtt{A}}$, serial number $\mathtt{sn}^{old}$, and coin commitment $\mathtt{cm}^{new}$, “I” know coins c^{old}, c^{new}, and secret key a^{old}_{sk} such that:*

- *Both c^{old} and c^{new} are* well formed*: for example, $k^{old} = \mathtt{COMM}_{r^{old}}(a^{old}_{pk} || \rho^{old})$ and $\mathtt{cm}^{old} = \mathtt{COMM}_{s^{old}}(v^{old} || k^{old})$.*
- *The address secret key matches the public key: $a^{old}_{pk} = \mathtt{PRF}^{addr}_{a^{sk}_{old}}(0)$.*
- *The old coin's commitment appears as a leaf of a merkle tree with root $rt_{\mathtt{A}}$, $L_{\mathtt{A}}$'s merkle root.*
- *This trading pair didn't use all liquidity: $v_{\mathtt{A}} > v_{old} \land v_{\mathtt{B}} > v_{new}$*
- *After exchange, the invariant of $L_{\mathtt{DEX}}$ remains valid: $(v_{\mathtt{A}} + v^{old}) \times (v_{\mathtt{B}} - v^{new}) = v_{\mathtt{A}} \times v_{\mathtt{B}}$*

The user u sends an exchange transaction $tx_{\mathsf{exchange}} := (rt_{\mathtt{A}}, \mathtt{sn}^{\mathsf{old}}, \mathtt{cm}^{\mathsf{new}}, \pi_{\mathsf{exchange}})$ to the ledger. The ledger checks the validity of the transaction: the transaction is valid only if $\mathtt{sn}^{\mathsf{old}}$ has never been used in a previous transaction in the $L_{\mathtt{A}}$ (otherwise it is a double spend), and the zk-SNARK verifier verifies the validity of π_{exchange}. In addition to adding cm^{old} and cm^{new} to $L_{\mathtt{A}}$ and $L_{\mathtt{B}}$ respectively, the ledger also mint two new `DEXCoin`s with value $v_{\mathtt{A}} + v^{\mathsf{old}}$ and $v_{\mathtt{B}} - v^{\mathsf{new}}$ and new serial numbers to replace the old ones in $L_{\mathtt{A}}$ and $L_{\mathtt{B}}$.

5 Conclusion

In this paper, we present a Manta, a decentralized anonymous exchange scheme that ensures users' privacy while exchanging private coins. The core idea of Manta is to leverage zk-SNARKs to private zero knowledge proof for an automated market maker scheme. We show that our scheme is secure and preserves users' privacy.

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. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *STOC*, pages 291–304. ACM, 1985.
8. 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.
9. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf.
10. 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.
11. Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger.