MantaPay Protocol Specification
v0.4.0

Shumo Chu and Brandon H. Gomes

October 30, 2021

Abstract

MantaPay is an implementation of a *decentralized anonymous payment* scheme based on the MANTA$_{\text{DAP}}$ protocol outlined in the original MANTA whitepaper.

Contents

1 Introduction

2 Notation

3 Concepts

3.1 Assets

The Asset is the fundamental currency object in the MantaPay protocol. An asset $a : \mathsf{Asset}$ is a tuple

$$a = (a.\mathsf{id}, a.\mathsf{value}) : \mathsf{AssetId} \times \mathsf{AssetValue}$$

The MantaPay protocol is a *decentralized anonymous payment* scheme which facilitiates the private ownership and private transfer of Asset objects. The AssetId field encodes the type of currency being used, and the AssetValue encodes how many units of that currency are being used, in the standard base unit of that currency.

Whenever an Asset is being used in a public setting, we simply refer to it as an Asset, but when the AssetId and/or AssetValue of a particular Asset is meant to be hidden from public view, we refer to the Asset as either, *secret*, *private*, *hidden*, or *shielded*.

Assets form the basic units of *transactions* which consume Assets on input, transform them, and return Assets on output. To preserve the economic value stored in Assets, the sum of the input AssetValues must balance the sum of the output AssetValues, and all assets in a single transaction must have the same AssetId[1].

3.2 Addresses

In order for participants in the MantaPay protocol to send and receive Assets, they must create secret and public *addresses* according to an *address scheme*. For MantaPay, the address scheme consists of a *spending key* sk, a *viewing key* vk, and a *public key* pk. The keys have the following uses/properties:

- Access to a public key pk represents the ability to send Assets to the owner of the associated sk.
- Access to a viewing key vk represents the ability to reveal shielded Asset information for Assets belonging to the owner of the associated sk.
- Access to a spending key sk represents the ability to spend Assets that were received under the associated public key pk.

See § 4.1.3 and § 4.2 for more information on how these keys are constructed and used for spending, viewing, and receiving.

3.3 Ledger

Ensuring that Assets maintain their economic value is not only dependent on transactions preserving inputs and outputs, but also that Assets are not *double-spent*. The *double-spending problem* can be solved by using a public ledger[2] that keeps track of the flow of Assets from one participant to the other. Unfortunately, using a public ledger alone does not allow participants to remain anonymous, so MantaPay extends the public ledger by adding a special account called the ShieldedAssetPool. The ShieldedAssetPool is responsible for keeping track of the Assets which have been anonymized by the protocol.

$$\begin{tikzcd}
\mathsf{ShieldedAssetPool} & & \\
& \mathsf{Transfer} \arrow[ul, leftrightarrow] \arrow[dl, leftrightarrow] \arrow[r, "\mathsf{spend}"] & \mathsf{Void} \\
\mathsf{PublicLedger} & &
\end{tikzcd}$$

Figure 1: Lifecycle of an Asset.

Assets can be in one of three states, public (tracked by the PublicLedger), allocated (spendable subset of the ShieldedAssetPool), or spent (voided Assets). By way of the § 4.3 Transfer Protocol, Assets can be sent to and from the PublicLedger and the ShieldedAssetPool.

The ShieldedAssetPool is made up of four parts:

[1] It is beyond the scope of this paper to discuss transactions with inputs and outputs that feature different AssetIds, like those that would be featured in a *decentralized anonymous exchange*.

1. ShieldedAssetPool Balance: The MantaPay ledger contains a collection of Assets which represent the combined economic value of the ShieldedAssetPool and the PublicLedger. The ShieldedAssetPool Balance is the subset of this total value that has been anonymized by the MantaPay protocol.
2. § 3.3.1 UTXO Set: A collection of claims to subsets of the ShieldedAssetPool, each owned by participants of the MantaPay protocol.
3. § 3.3.2 Encrypted Notes: For each UTXO there is a matching encrypted note which contains information necessary to spend the Asset, which is commited in the UTXO, but can only be decrypted by the recipient of the Asset, specifically, the correct viewing key vk. See § 3.2 for more.
4. § 3.3.3 VoidNumber Set: A collection of commitments keeping track of those UTXOs which have participated in exactly one instance of the Transfer Protocol.

An Asset is in the public state if it belongs to the PublicLedger. An Asset is in the allocated state if a UTXO for the Asset is a member of the UTXO Set, but its matching VoidNumber is **not** in the VoidNumber Set. An Asset is in the spent state if it was allocated in the past, but its matching VoidNumber is now in the VoidNumber Set.

The operation of the different parts of the ShieldedAssetPool is elaborated in the following subsections.

3.3.1 UTXO Set

3.3.2 Encrypted Notes

3.3.3 VoidNumber Set

4 Abstract Protocol

4.1 Abstract Cryptographic Schemes

4.1.1 Commitments

A *commitment scheme* COM is defined by the following schema:

$$\begin{aligned}\mathsf{Trapdoor} &: \mathsf{Type}\\ \mathsf{Input} &: \mathsf{Type}\\ \mathsf{Output} &: \mathsf{Type}\\ \mathsf{TrapdoorDistribution} &: \mathcal{D}(\mathsf{Trapdoor})\\ \mathsf{commit} &: \mathsf{Trapdoor} \times \mathsf{Input} \to \mathsf{Output}\end{aligned}$$

with the properties:

- Binding: It is infeasible to find an x, y : Input and r, s : Trapdoor such that $x \neq y$ and $\mathsf{commit}(r, x) = \mathsf{commit}(s, y)$.
- Hiding: For all x, y : Input, the distributions $\{\mathsf{commit}(r, x) \mid r \sim \mathsf{TrapdoorDistribution}\}$ and $\{\mathsf{commit}(r, y) \mid r \sim \mathsf{TrapdoorDistribution}\}$ are *computationally indistinguishable.*

Notation: For convenience we refer to $\mathsf{COM.commit}(r, x)$ by $\mathsf{COM}_r(x)$.

4.1.2 Hash Functions

A *hash function* CRH is defined by the following schema:

$$\begin{aligned}\mathsf{Input} &: \mathsf{Type}\\ \mathsf{Output} &: \mathsf{Type}\\ \mathsf{hash} &: \mathsf{Input} \to \mathsf{Output}\end{aligned}$$

with the properties:

- Pre-Image Resistance: For a given y : Output, it is infeasible to find x : Input such that $\mathsf{hash}(x) = y$.
- Collision Resistance: It is infeasible to find an x_1, x_2 : Input such that $x_1 \neq x_2$ and $\mathsf{hash}(x_1) = \mathsf{hash}(x_2)$.

Notation: For convenience we refer to $\mathsf{CRH.hash}(x)$ by $\mathsf{CRH}(x)$.

[2] A public (or private) ledger is not enough to solve the *double-spending problem.* A *consensus mechanism* is also required to ensure that all participants agree on the current state of the ledger. The *consensus mechanism* that secures the MantaPay ledger is beyond the scope of this paper.

4.1.3 Encryption

A *symmetric encryption echeme* Symm is defined by the following schema:

$$\begin{aligned}
\mathsf{Key} &: \mathsf{Type} \\
\mathsf{Plaintext} &: \mathsf{Type} \\
\mathsf{Ciphertext} &: \mathsf{Type} \\
\mathsf{encrypt} &: \mathsf{Key} \times \mathsf{Plaintext} \to \mathsf{Ciphertext} \\
\mathsf{decrypt} &: \mathsf{Key} \times \mathsf{Ciphertext} \to \mathsf{Option}(\mathsf{Plaintext})
\end{aligned}$$

with the properties:

- Valid Decryption: $\mathsf{decrypt}(k, \mathsf{encrypt}(k, p)) = \mathsf{Some}(p)$
- **TODO**: hiding? one-time encryption security?

4.1.4 Zero-Knowledge Proving Systems

4.2 Addresses and Key Components

4.3 Transfer Protocol

4.3.1 Senders

4.3.2 Receivers

4.3.3 Transfers

4.3.4 TransferPosts

5 Concrete Protocol

5.1 Conventions

5.2 Constants

5.3 Concrete Cryptographic Schemes

5.3.1 Commitments

5.3.2 Hash Functions

5.3.3 Encryption

5.3.4 Zero-Knowledge Proving Systems

5.3.4.1 Groth16

5.3.4.2 PLONK

6 Differences from MANTA$_{\mathtt{DAP}}$

6.1 Reusable Addresses

6.2 Transfer Circuit Unification

7 Acknowledgements

8 References