

# MantaPay Protocol Specification

## v1.0.0

Shumo Chu, Boyuan Feng, Brandon H. Gomes, Francisco Hernández Iglesias, and Todd Norton \*

August 28, 2022

### Abstract

MantaPay is an implementation of a *decentralized anonymous payment* scheme based on the MANTADAP protocol outlined in the original [MANTA whitepaper](#).

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Notation</b>	<b>2</b>
<b>3</b>	<b>Concepts</b>	<b>2</b>
3.1	zkAssets . . . . .	2
3.2	UTXOs . . . . .	3
3.3	Nullifiers . . . . .	3
3.4	zkAddresses . . . . .	3
3.5	Notes . . . . .	4
3.6	ShieldedPool . . . . .	4
<b>4</b>	<b>Abstract Protocol</b>	<b>4</b>
4.1	Abstract Cryptographic Schemes . . . . .	4
4.2	Addresses and Key Components . . . . .	4
4.3	Transfer Protocol . . . . .	4
4.4	Batched Transactions . . . . .	4
<b>5</b>	<b>Concrete Protocol</b>	<b>4</b>
5.1	Poseidon Permutation . . . . .	4
5.2	Elliptic Curve Cryptography . . . . .	4
5.3	Concrete Cryptographic Schemes . . . . .	4
<b>6</b>	<b>Acknowledgements</b>	<b>4</b>
<b>7</b>	<b>References</b>	<b>4</b>

---

\*ordered alphabetically

# 1 Introduction

MantaPay aims to solve the long-standing privacy problems facing cryptocurrencies. At its heart, it uses various cryptographic constructions including NIZK (non-interactive zero knowledge proof) systems to ensure user privacy from *first principles* and to build the foundational layer for programmable private money. The MantaPay protocol provides the following features:

1. Dynamic Multi-Asset Shielded Pool: A shielded pool for every kind of asset with dynamic anonymity set resizing
2. Verifiable Viewing Keys: Opt-in transaction transparency with audit correctness assurance
3. Programmable zkAssets: New Transparent UTXO model allowing programmability layers to be built on top of the shielded pool
4. Delegated Proof Generation: Decoupling the spending access from the proof generation access gives hardware wallets native support for zkAssets

## 2 Notation

The following notation is used throughout this specification:

- `Type` is the type of types<sup>1</sup>.
- If  $x : T$  then  $x$  is a value and  $T$  is a type, denoted  $T : \text{Type}$ , and we say that  $x$  *has type*  $T$ .
- `Bool` is the type of booleans with values `True` and `False`.
- For any types  $A : \text{Type}$  and  $B : \text{Type}$  we denote the *type of functions* from  $A$  to  $B$  as  $A \rightarrow B : \text{Type}$ .
- For any types  $A : \text{Type}$  and  $B : \text{Type}$  we denote the *product type* over  $A$  and  $B$  as  $A \times B : \text{Type}$  with constructor  $(-, -) : A \rightarrow (B \rightarrow A \times B)$ . Depending on context, we may omit the constructor and inline the pair into another constructor/destructor. For example, if  $f : A \times B \rightarrow C$  we can denote  $f((a, b))$  as  $f(a, b)$  to reduce the number of parentheses.
- For any type  $T : \text{Type}$ , we define  $\text{Option}(T) : \text{Type}$  as the inductive type with constructors:

$$\begin{aligned} \text{None} &: \text{Option}(T) \\ \text{Some} &: T \rightarrow \text{Option}(T) \end{aligned}$$

- We denote the *type of finite sets* over a type  $T : \text{Type}$  as  $\text{FinSet}(T) : \text{Type}$ . The membership predicate for a value  $x : T$  in a finite set  $S : \text{FinSet}(T)$  is denoted  $x \in S$ .
- We denote the *type of finite ordered sets* over a type  $T : \text{Type}$  as  $\text{List}(T) : \text{Type}$ . This can either be defined by an inductive type or as a  $\text{FinSet}(T)$  with a fixed ordering. We denote the constructor for a list as  $[\dots]$  for an arbitrary set of elements.
- We denote the *type of distributions* over a type  $T : \text{Type}$  as  $\mathfrak{D}(T) : \text{Type}$ . A value  $x$  sampled from  $\mathfrak{D}(T)$  is denoted  $x \sim \mathfrak{D}(T)$  and the fact that the value  $x$  belongs to the range of  $\mathfrak{D}(T)$  is denoted  $x \in \mathfrak{D}(T)$ . So namely,  $y \in \{x \mid x \sim \mathfrak{D}(T)\} \leftrightarrow y \in \mathfrak{D}(T)$ .
- Depending on the context, the notation  $|\cdot|$  denotes either the absolute value of a quantity, the length of a list, the number of characters in a string, or the cardinality of a set.

## 3 Concepts

### 3.1 zkAssets

The `zkAsset` is the fundamental currency object in the MantaPay protocol. An asset  $a : \text{zkAsset}$  is a tuple

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

where the `AssetId` encodes the type of currency stored in  $a$  and the `AssetValue` encodes how many units of that currency are stored in  $a$ . MantaPay is a *decentralized anonymous payment* protocol which facilitates the private ownership and private transfer of `zkAssets`.

---

<sup>1</sup>By *type of types*, we mean the type of *first-level* types in some family of type universes. Discussion of the type theory necessary to make these notions rigorous is beyond the scope of this paper.

zkAssets are the basic building-blocks of *transactions* which consume a set of input zkAssets and produce a set of transformed output zkAssets. To preserve the economic value stored in zkAssets, 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<sup>2</sup>. This is called a *balanced transfer*: no value is created or destroyed in the process. The MantaPay protocol uses a distributed algorithm called *Transfer* to perform balanced transfers and ensure that they are valid.

### 3.2 UTXOs

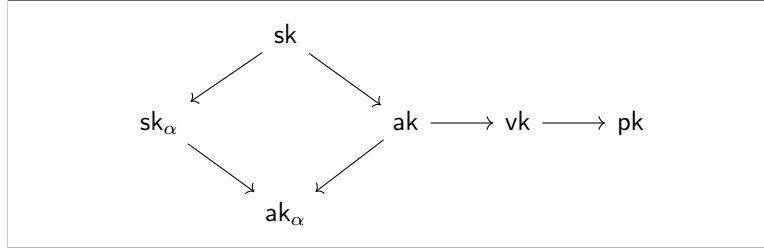
But zkAssets are not private on their own. A UTXO is a container for a zkAsset that hides its value and its owner and is the main object that MantaPay uses to transfer the spending power of zkAssets between different protocol participants. A UTXO is a cryptographic commitment along with some associated data that represents a spendable subset of an account stored in the protocol. In the MantaPay protocol, UTXOs come in two flavors, *opaque* and *transparent*. The *opaque* UTXOs are completely private and they do not reveal the owner or underlying asset contained in them, whereas *transparent* UTXOs reveal the underlying asset but not the owner. The *opaque* UTXO is used for the private transfer of zkAssets and the *transparent* UTXO is used to give programability to zkAssets whenever the MantaPay protocol lives in the same environment as other smart contracts by allowing contracts to control the AssetId and AssetValue stored in the *transparent* UTXO.

### 3.3 Nullifiers

One of the important ways that privacy is preserved for zkAssets across many transactions is that the exact transaction where a UTXO is spent is not known to the public. Instead, only the owner of the zkAsset, or anyone with the appropriate viewing key, can know this information. The Nullifier is another cryptographic commitment that takes the place of the UTXO when it is spent and it is cryptographically hard for any particular UTXO to be derived from its Nullifier.

### 3.4 zkAddresses

In order for MantaPay participants to receive zkAssets via the *Transfer* protocol, they create *zk-addresses* which they use as identifiers to represent them on the ledger.



**Figure 1:** Key Schedule for MantaPay.

MantaPay uses four kinds of keys all derived from a base secret, spending key  $sk$ , which give the following kinds of privileged access in the protocol:

- **zkAddress (send):** Access to the zk-address  $pk$  gives the user the right to send zkAssets to the owner of the associated  $sk$ .
- **Viewing Key (view):** Access to the viewing key  $vk$  gives the user the right to view all transactions for the owner of the associated  $sk$ .
- **Proof Authorization Key (prove):** Proof authorization key  $ak$  gives the user the right to build the *Transfer* proof on behalf of the owner of  $sk$ . This key is used when delegating proof generation to a semi-trusted entity while still protecting the spending rights associated to the  $sk$ , for example, if a hardware wallet holds  $sk$  it can ask a more capable computer to produce the *Transfer* proof for it without sending the spending rights off of the hardware wallet.
- **Spending Key (spend):** Access to the spending key  $sk$  gives total control over the assets owned by this secret, including spending, proof generation, and viewing.

<sup>2</sup>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*.

Participants in MantaPay are represented by their zk-addresses, but they are not unique representations, since one participant may have access to more than one secret key. See § 4.2 for more information on how these keys are constructed and used for spending, proving, viewing, and receiving.

### 3.5 Notes

The encrypted Note is the primary means of communication in the MantaPay protocol. For a zkAddress owner to know that they have received a zkAsset and can now spend it they decrypt Notes with their viewing key to discover how much of an asset they have received and what information they need to spend it. The Note is also used to keep track of the balances of an entire account over its transaction history.

There are two kinds of Notes in the MantaPay protocol, *incoming* Notes and *outgoing* Notes. The IncomingNote is attached to every new UTXO and contains the same zkAsset as the UTXO and also a secret randomizer used to hide the UTXO commitment. The OutgoingNote is attached to every new Nullifier and contains the same zkAsset as the UTXO that the Nullifier is marking. When performing accounting over a zkAddress to measure how much of a particular AssetId that address controls, the AssetValue stored in the IncomingNotes should be *added* to the running total whereas the AssetValue stored in the OutgoingNotes should be *subtracted* from the running total as they represent inflows and outflows respectively.

### 3.6 ShieldedPool

## 4 Abstract Protocol

### 4.1 Abstract Cryptographic Schemes

In the following section, we outline the formal specifications for all of the *cryptographic schemes* used in the MantaPay protocol.

### 4.2 Addresses and Key Components

### 4.3 Transfer Protocol

### 4.4 Batched Transactions

## 5 Concrete Protocol

We define the instantiation of the abstract protocol in this section, but first some preliminary notes.

### 5.1 Poseidon Permutation

### 5.2 Elliptic Curve Cryptography

### 5.3 Concrete Cryptographic Schemes

## 6 Acknowledgements

We would like to thank Luke Pearson and Toghrul Maharramov for our insightful discussions on reusable shielded addresses.

## 7 References

### References