Midnight Transaction Kernel Specification

draft

Amira Bouguera, Thomas Kerber

June 9, 2022

This document specifies the semantics of creating and validating Midnight transactions, outside of the sandboxed realm of each smart contract. Notably, this includes native currencies, their transfer, and interactions between multiple contracts, and contracts and currency.

Contents

1 Introduction

This document leans heavily on research in ZSwap [], Zexe [], and to a lesser extent, Kachina [].

State of the art

2 Notation

Definitions of notation used throughout the document.

3 Concepts

The principal abstractions needed to understand the protocol

3.1 Zcash sapling and Orchard

Diversifier keys

Both Sapling and Orchard have three step in their key derivation mechanisms:

- Spending keys extension
- Child key derivation
- Diversifier key derivation

The first step prevents the child keys to be completely dependent on the parent key ans that's why they are extended. In sapling, three keys are derived: Spend authorizing key, proof authorizing key, and outgoing viewing key. In Orchard, the proof authorizing key is removed and the spending key sk is used to derive a Spend authorizing key. The second step derives full viewing keys from which payment addresses can be generated. Sapling and Orchard do not derive public keys directly, as this would prevent the use of diversified addresses. Viewing keys enable the separation of spending and viewing permissions for Zcash shielded addresses. By creating a separate viewing key, Zcash users can share visibility of the transactions sent and received from their shielded address without compromising their private spend key. As with Sapling, we define a mechanism

for deterministically deriving a sequence of diversifiers, without leaking how many diversified addresses have already been generated for an account. For the last step, in order to deterministically derive a sequence of diversifiers and prevent the diversifier leaking how many diversified addresses have already been generated for an account, the sequence of diversifiers is pseudorandom and uncorrelated to that of any other account. Sapling derives a diversifier key as part of the extended spending key whereas in Orchard it's derived directly from the full viewing key and provides the capability to determine the position of a diversifier within the sequence, which matches the capabilities of a Sapling extended full viewing key but simplifies the key structure.

3.2 Zexe Swaps

3.3 Transactions and Treestates

3.4 Notes

3.5 Spend Transfers and Output Transfers

3.6 Transaction merge

3.7 Commitment schemes

3.7.1 Pedersen commitment

3.7.2 Homomorphic commitment

3.8 Binding signature

3.9 Hash Functions (Poseiden)

3.10 Curves

3.11 Nullifier Sets

3.12 Zero-Knowledge Proving Systems(Plonk, Halo 2, Groth16)

4 Protocol

4.1 Structure of Coins

A basic coin consists of a controlling **secret key**, a random **nonce**, a **value**, and a **color**. Coins may additionally have a **spend predicate**, and a **locking contract**.

Proposed data types:

- **secret key** – a 256-bit bitstring
- **nonce** – a 256-bit bitstring
- **value** – a 64-bit unsigned integer
- **color** – a 256-bit bitstring

- **spend predicate** – a hash of the circuit IR, 0 representing no predicate
- **locking contract** – a contract address, 0 representing no lock

Coins the twin cryptographic projections of **notes** and **nullifiers**, both of which commit to all of the coin's contents. They are domain-separated, and **notes** commit to a (diversified) public key derived from the secret key.

4.2 Notes

A note n represents that a value v is spendable by the recipient who holds the spending key corresponding to a given address. Each note is associated with a note commitment where every commitment represents a leaf in a tree of note commitments. The goal of using note commitments is to preserve privacy of the receipient's address and the value they are receiving as only the commitment to the above values is disclosed publicly when a note is spent. The commitment is then used by a zk-SNARK proof to check that it exists on the blockchain in the commitment tree.
A note is also associated to a unique nullifier which requires the associated private spending key in order to be computed. It is infeasible to correlate the note commitment or note position with the corresponding nullifier without knowledge of at least this key.

4.2.1 Note generation

4.2.2 Note components

We define the note components similar to ZCash Sapling as a tuple (d, pk_d, v, rcm) where:

- $d : \mathbb{B}^{[l_d]}$ is the diversifier of the receipient address where $\mathbb{B}^{[l_d]}$ means the type of byte values, i.e. 0 .. 255.
- $pk_d : KA.pps$ is the diversified transmission key used to encrypt the note. KA is a key agreement scheme and $KA.p$ is a type of public keys which includes a subset $KA.pps$ (i.epps is a short for PublicPrimeSubgroup) such that

$$KA.pps \subseteq KA.p$$

- $v : \{0, ..., MAX\}$ is an integer that represents the value of the note
- $rcm : n_{com}.tr$: is a random commitment trapdoor (see commitment section).

Each note has a type n_{ty} that is computed as follow:

$$n_{ty} := \mathbb{B}^{[l_d]} \times n_{com}.tr \times KA.pps$$

4.2.3 Note commitment

Each time a note is spent, the note commitment gets publicly revealed. The note spender only proves that some commitment for that note had been revealed, without revealing which makes a spent note and the transaction that created it unlinkable. Notes commit to a diversifier key in the following way:

4.3 Nullifiers

The role of nullifiers is to prevent notes double spending. This is guaranteed by proving in a deterministic way the correctness of the nullifier marking the input as spent and thus cannot be spent again. S_{nf} is the set of used nullifiers

4.4 Structure of Transactions

A transaction consists of n outputs, m spends, a balancing signature, binding randomness, a public transcript, and a set of proofs of valid smart contracts transitions. We call spends and outputs inside a transaction Spend descriptions and Output descriptions, they represent a piece of data included in a transaction that describe Spend transfers and Output transfers, respectively. Each Spend is authorized by a signature, called the spend authorization signature.

4.4.1 Zswap Spends and outputs

Spends A spend transfer is a transfer creating value of the issued currency, i.e. increasing the circulating supply.

Outputs

5 Appendix