Sybil Resistant Airdrops

1 Introduction

Notebook provides users with a set of fragmented identities that have the following properties:

1. Users can prove their humanity
2. It is impossible to link these identities together (unless they have the user's secret key)
3. It is impossible to link these to the user's real-world identity
4. Credentials can be aggregated across identities
5. Each human only receives a single set of fragmented identities

To our knowledge, no other identity solution offers these properties. Furthermore, this set of properties is perfectly suited to prevent Sybil attacks during Airdops. The proof of humanity check ensures that each user can only enter one wallet into the airdrop whilst our protocol ensures that they remain completely anonymous.

2 Creating a Notebook

Notebook has an initial onboarding phase which acts as a proof of humanity check. The users firsts generate a secret sk and a random nonce r. A user then submits an ID and a short video of their face, which is used to create a face mask. A hash of the face mask is stored to ensure that if the same user attempts to create a second Notebook, a collision would accur the user would be denied. The user uploads a leaf, L, to be signed and provides a ZK-Proof to the signing party that they know sk, r such that $L = H(sk||r)$. This check ensures that the user is acting according to the protocol definition. If the check passes, $H(sk||r)$ is signed by the audited third party using the ECDSA key sk_N. We denote this signature $Signature$. Next, the user must add this identity to the $Sybil$ Merkle Tree stored on the smart contract $\mathcal{C}$. First, we must check that the user hasn't already added this identity to the $Sybil$ Merkle Tree. Then we verify that $Signature$ is a valid ECDSA signature of $H(sk||r)$ with public key pk_N. So, we first check that L is not in the created map. Then, the user

will provide an Update Merkle proof to the next available leaf of the *Sybil* tree using the following SNARK.

$$SNARK(H(sk||r), \pi, MerkleRoot, j)$$

The proof will check that π is a valid update proof for $H(sk||r)$ being added to index j. If the proof is valid, the smart contract updates the stored $MerkleRoot$ and increments j.

3 Sybil Resistant Login

In many cases, a protocol may want a user to prove ownership over a Notebook. For example, they may want to check that the user is not a bot, or that they have not already registered an account.

The user will present a proof that they know sk, r such that $H(sk||r)$ is a leaf of the *Sybil* Merkle tree. To do this, they will use an Opening proof. The user also needs to prove that they haven't previously logged in through another sub-identity. To do this, they will provide $Null = H(sk||ProtocolAddress)$. The protocol will store the value in a map such that if the user created another account, there would be a collision.

$$ZK - SNARK(sk, r, L = H(sk||r), \pi, Null, ProtocolAddress)$$

Here we have that sk, r, L, π are private inputs to the proof. The proof works in the following way:

1. Check that $L = H(sk||r)$
2. Check that π is a valid opening for $H(sk||r)$ in the Sybil tree.
3. Check that $Null = H(sk||ProtocolAddress)$

The protocol then checks that $Null$ hasn't already been used, and if it hasn't, it stores $Null$ and lets the user log in.