Proof-of-Uniqueness: A Decentralized Approach to Privacy-Preserving Identity Verification based on zk-SNARKs

Excel @FIT 2026

Author: Adam Vožda
Supervisor: doc. Ing. Ivan Homoliak Ph.D.

Motivation: Sybil resistance

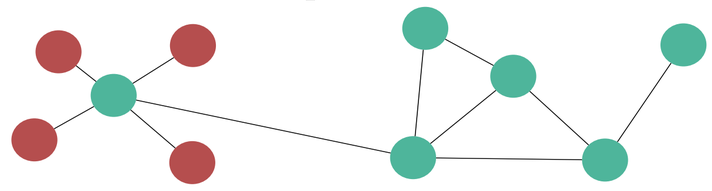

Figure 1: Sybil attack

- Permissionless applications allow unlimited participation.
- This enables Sybil entities (one person with multiple accounts).
- Manual whitelisting is not flexible or robust.
- They require privacy-sensitive identity verification.
- Example application is Proof-of-Social-Capital.

W3C Verifiable Credentials

1


```
{
  "issuer": "...",
  "validUntil": "...",
  "credentialSubject": {
    "id": "...",
    "name": "Jan Novak",
    "dateOfBirth": "...",
    ...
    "holderPublicKey": "..."
  },
  "proof": {...}
}
```

Figure 2: W3C Verifiable credentials issuane

- Every person's identity is represented by W3C verifiable credentials issued by trusted credential issuers (e.g. EUDI wallet).

Auth zk-Snark

2

- Checks VC validity
- Creates HashID and blinded query
- Verifies holder´s signature

Outputs: { Blinded HashID, requestID }

Figure 3: OPRF authorization zk-SNARK

- To authenticate, user makes a zk-SNARK using his VC, request ID, and his signature over the request.
- The circuit verifies the issuer signature on the VC and checks that user really owns the key inside it.
- This prevents attackers from computing the OPRF from leaked user credentials

$$h_{id} = \text{Poseidon}(name, dob, \ldots, id)$$

$$B_q = \beta \cdot h_{id}$$

Where β represent blinding factor randomly chosen by user

TACEO´S MPC Network

3

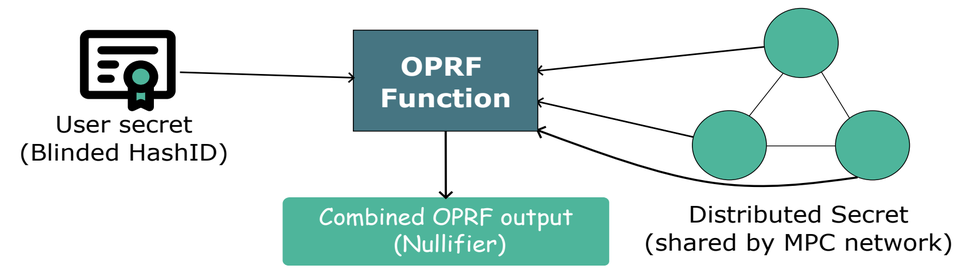

Figure 4: Diagram showcasing TACEO:OPRF

- If user credentials leak, attacker could compute . Exposing user´s privacy. TACEO:OPRF solves this h_{id} problem.
- Nullifier is computed deterministically from user secret and MPC network secret. Network secret is distributed across nodes so nobody control everything.
- Nodes compute OPRF result without ever seeing user data. They provide zero-knowledge proofs that they did not cheat.

$$R_i = k_i \cdot B_q$$

$$R = \sum_{i=1}^{t} \lambda_i \cdot R_i$$

Enroll zk-Snark

4

- Validates blinded queary creation
- Validates OPRF

Outputs: { nullifier, issuer, expirationDate, OPRF Publ key }

Figure 5: Enrollment zk-SNARK for Identity Registry

This circuit is required to prove to the smart contract that:

1. The blinded query was correctly formed by using VC fields
2. The proofs from MPC nodes are valid.
3. OPRF response was unblinded correctly

$$Y = \beta^{-1} \cdot R$$

$$R \stackrel{?}{=} \beta \cdot K \cdot h_{id}$$

Where K represents TACEO´S OPRF key

Identity Registry Smart Contract

5

- Verifies zk-SNARK
- Checks deduplication
- Checks if issuer is trusted

Stores: nullifier => { issuer, expirationDate }

Figure 6: Identity registry smart contract enrollment flow

- Smart contract is the public storage place for all the user nullifiers
- It needs to save issuer and expiration date so it can purge invalid records in the future.

zk-SNARKs metrics

Circuit	Size	Prove time	Verification time
Auth zk-SNARK	2.1kB	~10s	~7s
Enroll zk-SNARK	2.1kB	~18.5s	~14s

Onchain cost

On-chain operation	Measured cost (gas)
Enroll zk-SNARK verification	~5.4M
Storage write	~211k

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

BlockSec@FIT