RARIMO: A USER-OWNED SOCIAL PROTOCOL

WHITE PAPER, V3

April 4, 2024

ABSTRACT

Rarimo is a privacy-first social protocol that seamlessly combines various identity standards and allows the formation of a private yet verifiable history of their use and relations. This paper aims to describe the different components of the Rarimo protocol, which includes: 1. the core layer responsible for propagating identity states across networks; 2. protocols for creating identity artifacts such as Verifiable Credentials and self-issued credentials based on existing identity documents; 3. the mechanism of identity management by the owner and creating a private graph of identity actions and relations.

1 Introduction

Bitcoin empowered individuals with a decentralized currency. Ethereum enabled the decentralization of financial applications. The next frontier in the web3 realm is developing a reliable social protocol, which must meet several criteria:

1. **Control**. Users should have exclusive authority over managing their identity and all connected attributes. Optionally, certain processes (e.g., identity recovery) can involve trusted parties for additional protection, but only at the user's request.
2. **Privacy**. Sensitive data must remain inaccessible to third parties. The user alone should decide what information to share, when, and with whom. While some protocols may require verifying the user's uniqueness, this should not compromise the privacy of personal information.
3. **Compatibility**. The digital identity solution should support various industry standards, such as W3C DID credentials, soulbound tokens (SBTs), ENS records, etc. Each of these identity formats offers unique advantages; therefore, the social protocol must allow interoperability among them. Additionally, it should support both off-chain and on-chain verification methods.

2 Problem Statement

The current landscape of digital identity management, particularly through the use of NFTs and SBT tokens, faces significant challenges in maintaining ownership privacy and ensuring untraceability for its users. These tokens, while useful in representing ownership and attributes, fall short in providing comprehensive control and privacy guarantees. Other identity solutions (Verified credentials, proofs) offer a means to create accounts and associate attributes privately, yet they struggle with mapping inter-account relationships, a key aspect of social interactions.

We believe the social protocol must allow for private control of identity. At the same time, we hold that the most significant aspect of identity is its relationships with other identities, including bidirectional attestations, shared communion, and participation, among others. However, all of these should be also hidden by default and disclosed only upon request.

Different standards and their verification methods create big challenges in making digital social interactions that can scale easily and work seamlessly. This is because digital identities are complex, involving different types of digital attributes, each with its own way of being managed and structured.

- **Account addresses**, which denote asset ownership and balance.
- **NFT and SBT tokens** signifying memberships and attributes.
- **SSI states and Verifiable Credentials** representing statements.
- **Web services** which bridge web3 applications with the traditional web.
- **Document-derived artifacts** representing legal documents to be integrated and utilized within web3.

From these separately evolving building blocks, three primary challenges emerge, complicating the landscape:

- **Identity fragmentation and lack of interoperability:** This fragmentation makes it difficult for dApps from accurately verifying users from different ecosystems and obstructs the unification of different identity components across different chains.
- **Privacy concerns**: Maintaining a reasonable level of anonymity or managing sensitive data on-chain proves difficult. Currently, only protocols incorporating zero-knowledge proofs effectively address these privacy issues.
- **Complex identity verification algorithms**: The challenge of building a universal identity query that covers all potential scenarios complicates identity verification. Although verification methods are robust, a standardized API is essential for developing applications that can navigate this complexity.

3 Preliminaries

3.1 Hashing Functions

Let $\{0, 1\}$ be a binary element and $\mathbb{F}_p$ be a finite field of prime order p. We introduce two hashing functions: H_n and H_{zk}. H_n is a regular cryptographic one-way function $\mathsf{H}_n : \{0, 1\}^* \to \{0, 1\}^n$ that sends the string of an arbitrary length $\{0, 1\}^*$ to the string of fixed size n, while $\mathsf{H}_{\text{zk}} : \{0, 1\}^* \to \mathbb{F}_p^m$ is the zk-friendly hash function, which works natively with the field $\mathbb{F}_p$. Usually, m is set to 1 to simplify the arithmetic circuit.

Formally, the hashing function is a deterministic one-way 2nd-preimage-resistant function. The primary, simply explained reason for using a hashing function is the following: based on $y = \mathsf{H}(x)$, it is impossible to deduce x, meaning y can be a public parameter (preimage resistance). Also, having a random message m_1, it is computationally impossible to find m_2 such that $\mathsf{H}(m_1) = \mathsf{H}(m_2)$. These two properties would come in handy in the subsequent sections.

3.2 Binary Tree

Let $\mathcal{T}_H$ be a binary tree instance with the height H and elements $T_{i,j} \in \{0, 1\}^n$, where $i \in \{0, \ldots, H\}$ and $j \in \{0, \ldots, 2^{H-i-1} - 1\}$. $\mathsf{Path}(T_{0,j}, T_{\text{root}})$ is a Merkle path for the leaf $T_{0,j}$ for the tree $\mathcal{T}_H$ with the root $T_{H,0}$. We will use the notation $\mathsf{Root}(L_1, \ldots, L_m)$ as a function of calculating Merkle Root for a particular set of leaves $(L_1, \ldots, L_m)$.

3.3 Zero-knowledge Proof

Further, since the protocol uses zero-knowledge proofs, we formally define it. *Zero-knowledge proof (ZKP)* is a two-party protocol running between a prover and a verifier for proving a statement without revealing information behind the statement. Each zero-knowledge protocol satisfies three properties: completeness, soundness, and zero-knowledge.

The zero-knowledge protocol consists of the following functions:

- $\mathsf{Setup}(1^\lambda)$ – given security parameter $\lambda \in \mathbb{N}$, gives the public parameters $\langle \mathsf{pp}, \mathsf{vp} \rangle$ for prover and verifier, respectively.

- $\mathsf{Prove}(\mathsf{pp}, \mathbf{w}, \mathbf{x})$ – given prover parameters pp, private info (witness) $\mathbf{w}$, and public statement $\mathbf{x}$, generates a short proof π.
- $\mathsf{Verify}(\mathsf{vp}, \mathbf{x}, \pi)$ – given public parameters vp, verifies whether for the proof π exists some witness $\mathbf{w}$ such that $\langle \mathbf{w}, \mathbf{x} \rangle \in \mathcal{R}$ where $\mathcal{R}$ is a relation.

When defining the zero-knowledge proof, we will use the notation below. Here, **pub_signals** stands for public signals, **priv_signals** for private signals and **circuit_logic** is where the main verification logic is specified.

$$\left| \begin{array}{c} \textbf{pub\_signals:} \\ \mathsf{signal}_1, \mathsf{signal}_2, \dots, \mathsf{signal}_n \\ \textbf{priv\_signals:} \\ \mathsf{signal}_{n+1}, \mathsf{signal}_{n+2}, \dots, \mathsf{signal}_m \\ \textbf{circuit\_logic:} \\ \mathsf{cond}_1 \wedge \mathsf{cond}_2 \wedge \dots \wedge \mathsf{cond}_k \end{array} \right| \tag{1}$$

Additionally, we further use the notation $a \leftarrow b$, which reads as "variable a gets the value of b", and $a \stackrel{!}{=} b$ as "we require a to be equal to b".

3.4 Certificates and Signatures

Let $\mathsf{Cert}(\mathsf{pk})$ be a valid certificate that belongs to the passport issuer with the public key pk. We further use $\mathcal{C}$ to denote the set of all valid certificates. When writing $\mathsf{pk} \in \mathcal{C}$, we mean that "the certificate for public key pk is valid".

We also use the signature scheme, consisting of the following functions:

- $\mathsf{Sign}(m, \mathsf{sk})$ – sign message $m \in \mathcal{M}$ using the secret key sk and output the signature σ.
- $\mathsf{SigVerify}(\sigma, \mathsf{pk}, m)$ – verifies that signing m with a secret key, corresponding to pk, produces the signature σ. Outputs accept if the signature is correct, and reject otherwise.

4 Rarimo Core

The **Rarimo Core** is a decentralized blockchain-based system designed for timestamping, storing, and updating identity states and social relations that other networks and protocols can use.

Rarimo Core is maintained by a set of validators that achieve consensus using the BFT-based algorithm [Buc16]. It assumes that for consensus reaching (with fault tolerance), the maximum amount of faulty validators f can be $f = \left\lfloor \frac{n-1}{3} \right\rfloor$, where n is the total number of nodes.

Rarimo uses a delegated Proof-of-Stake mechanism for validator selection. Any user can propose themselves as a validator; if other users trust the delegate, they can stake their tokens for them to become a validator. The delegates with the top staked balances are selected as validators.

Additionally, Rarimo Core allows the use and propagation of identity states over connected networks using protocol oracles (selected, governed, and rewarded by the Rarimo DAO). It also increases the decentralization of fetching identity data from external networks (if identity events did not originate on the Rarimo chain).

4.1 Roles and Actors

The Rarimo protocol supports several roles, including the identity owner, issuer, verifier, validator, and signer.

- The **Identity owner** is the user who collects identity artifacts such as ownership and membership statements. The mechanism for controlling credentials depends on the type of credential, whether it is a W3C credential, self-issued credential, SBT, or another type.
- The **Issuer** is the party that can create a digital claim with a statement about the identity owner. For certain types of identity artifacts, such as document-based identities, a

Field	Purpose
Block header	The structure that includes all metadata needed for block validation
Data (transaction list)	The list of transactions that are confirmed in this block
Evidence	The threshold ECDSA[GG20] and EdDSA[MBS23] signatures of the current validators' quorum

Table 1: Rarimo block structure.

decentralized issuer may be used as a smart contract that verifies proofs and updates their state accordingly. Sometimes, we will use the "identity provider" term for the same role.

- The **Verifier** is the off-chain or on-chain application that verifies the proof's correctness (connected to an identity statement). For that, the verifier retrieves states from the Rarimo protocol.
- The **Validator** is responsible for maintaining the protocol and reaching a consensus with other validators regarding the acceptance of new blocks and transactions.
- Finally, the **Signer** is a participant who can bring data to or from Rarimo Core and co-sign this action with a threshold digital signature.

Furthermore, we will introduce two key roles: the **Initiator**, who initiates a transaction that triggers a process (such as sending a cross-chain message), and the **Prover**, who provides evidence of the validity of an action (for example, verifying that Rarimo validators have endorsed a block with a specific state).

4.2 Quorum-driven State of Rarimo Core

As mentioned, Rarimo Core uses blockchain technology to maintain a history of transactions associated with the protocol when sending and confirming transactions. These mechanisms are essential for retrieving the complete history and origin of the unique object that navigates through web3. The structure of blocks within the blockchain is represented as follows (Table 1).

Therefore, when a transaction needs to be added to the blockchain, the proposer (the validator who initially forms the block and shares it with all other validators) incorporates this transaction into the block (along with other transactions), signs it, and then broadcasts it across the network. The validator who places their signature on the block verifies that all included transactions are correct. Following a consensus round (where all other validators agree to add the transaction), the block is incorporated into the blockchain, and the system's state is updated.

Using the provided block structure, the prover can create a witness about including the particular transaction into the block (through providing the Merkle path).

4.3 Cross-chain Messaging Protocol

The main flow of cross-chain message transfer happens in 4 steps (Figure 1):

1. The user creates an action and sends it to the Rarimo Contract. An alternative case is when some Oracle services track specific events on defined contracts and react to them, sending appropriate messages to the Rarimo Core.
2. Validators add a transaction to the Rarimo Core (by consensus reaching) and generate a time-stamped witness of its existence.
3. Prover receives the witness from the Rarimo Core and provides it to the Rarimo Contract on the destination chain. The prover means any relayer who can call the appropriate smart contract method.
4. The recipient (dApp) updates their state depending on the received message.

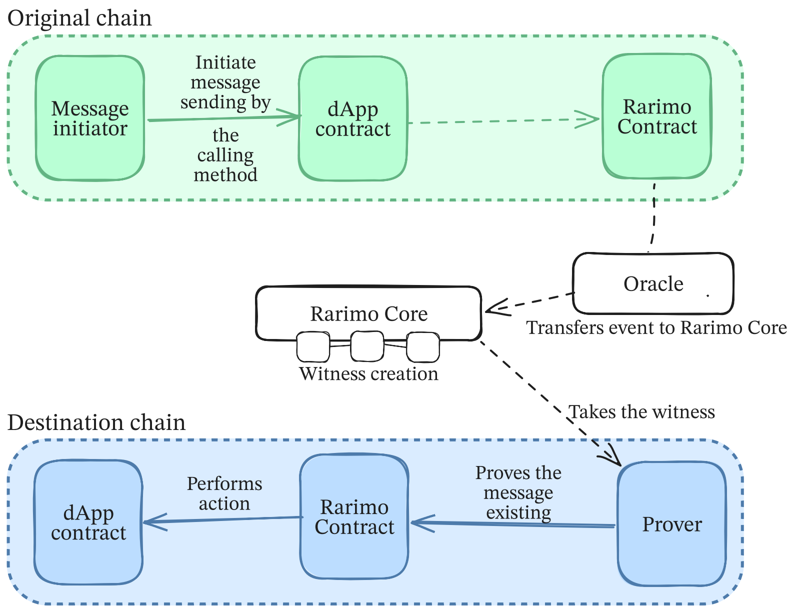


Figure 1: Cross-chain messaging flow.

Algorithm 1 Client $\mathcal{C}\ell$ performs a cross-chain transaction $\mathsf{tx}_{A\to B}$ to transfer the message from system A to system B. SC_A is a bridge smart contract in system A, SC_B is a smart contract in B. $\mathcal{P}$ is a block proposer, and $\mathcal{V}$ is the set of current validators of the Rarimo Core (CORE). tx_{msg} is a transaction that includes the corresponding message. $\mathsf{tx}_{w\to B}$ – a transaction that provides needed witness w for SC_B initiating.

```
Step 0. Crosschain transaction.
Let $\mathsf{acc}_A$ – account of the user in system $A$, $\mathsf{acc}_B$ – account of the receiver in system $B$.
Step 1. Message sending
1. $\mathcal{C}\ell$ sends the $\mathsf{tx}_{\text{msg}}: \mathsf{acc}_A \to \mathsf{SC}_A$ to initiate a witness transferring.
2. $\mathcal{P}$ parses all $\mathsf{SC}_A$ events for receiving a set of transactions $\boldsymbol{L}_{\mathsf{tx}} = \{\mathsf{tx}^{(i)}_{\text{msg}}\}$ from $\mathsf{acc}_X$ to $\mathsf{SC}_A$. $\mathcal{P}$ verifies each $\mathsf{tx}_{\text{msg}} \in \boldsymbol{L}_{\mathsf{tx}}$ and forms a list of valid transactions $\boldsymbol{L}_{\mathsf{valid}} \subset \boldsymbol{L}_{\mathsf{tx}}$.
3. For each $\mathsf{tx}_{\text{msg}} \in \boldsymbol{L}_{\mathsf{valid}}$:
  3.1. $\mathcal{P}$ calculates $\mathsf{data\_hash} \leftarrow \mathsf{Root}(\boldsymbol{L}_{\mathsf{valid}})$.
  3.2. $\mathcal{P}$ forms a block header $\langle \mathsf{prev\_hash}, \mathsf{Cert}(\mathcal{V}), \mathsf{data\_hash}\rangle$
  3.3. $\mathcal{P}$ forms a block $\langle h, \boldsymbol{L}_{\mathsf{valid}}, \sigma_{\mathcal{P}}\rangle$ with header $h$ and signature $\sigma_{\mathcal{P}}$.
4. $\mathcal{P}$ propagates the block with a set of validators $\mathcal{V}$.
5. $\mathcal{V}$ performs validation of the received block and adds the corresponding evidence of its correctness $\mathsf{Cert}(\mathcal{V})$.
6. So, the final block has the structure $\langle h, \boldsymbol{L}_{\mathsf{valid}}, \mathsf{Cert}(\mathcal{V})\rangle$ with a header $h$.
Step 2. Message receiving
1. $\mathcal{C}\ell$ parses $\mathsf{CORE}$ for receiving the block with $\mathsf{tx}_{\text{msg}} : \mathsf{acc}_A \to \mathsf{SC}_A$. This transaction can be parsed by a separate service and returned to $\mathcal{C}\ell$.
2. $\mathcal{C}\ell$ received the block $\langle h, \boldsymbol{L}_{\mathsf{valid}}, \mathsf{Cert}(\mathcal{V})\rangle$ and takes a header, only required transaction and evidence.
3. For $\mathsf{tx}_{\text{msg}}$, $\mathcal{C}\ell$ generates the Merkle Branch $(\mathsf{auth}_0, \mathsf{auth}_1, \ldots, \mathsf{auth}_{n-1})$ – witness that particular $\mathsf{tx}_{\text{msg}}$ included to the block
4. $\mathcal{C}\ell$ forms $\mathsf{tx}_{w\to B}$ and sends it to $\mathsf{SC}_B$
5. $\mathsf{SC}_B$ performs the following checks:
  5.1 $\mathsf{tx}_{\text{msg}} \notin \boldsymbol{L}_{\mathsf{valid}}$ stored by the $\mathsf{SC}_B$
  5.2 $\mathsf{Cert}(\mathcal{V})$ is valid for block.
  5.3 $\mathsf{tx}_{\text{msg}} \in h$ via verification $(\mathsf{auth}_0, \mathsf{auth}_1, \ldots, \mathsf{auth}_{n-1})$
6. $\mathsf{SC}_B$ sends the transaction $\mathsf{tx}_{\text{msg}} : \mathsf{acc}_{\mathsf{SC}_B} \to \mathsf{acc}_B$ to transfer the final message.
```

5 Social Protocol

Rarimo supports EVM-compatible smart contracts that allow contributors to build sub-protocols using a cross-chain messaging layer. The primary sub-protocol built on top is the zero-knowledge identity, based on the Iden3 standard[Ide] and passport-derived profiles.

The Iden3 protocol is suitable for receiving identity statements in a Verifiable Credential format. Rarimo allows identity providers to publish only state hashes into the Rarimo blockchain. After propagating to connected networks, DApps can use the mentioned states to verify various identity statements. This flow presumes the creation of the identity claims of the user by the identity provider and pushing its state in the Rarimo Core or connected blockchains. Periodically or upon request, identity states are broadcast across connected networks, making them accessible to end-users directly on the requested chains.

Rarimo's cross-chain messaging and hub architecture allow the synchronization of these states between all connected chains quickly and with lower fees. This enables identities published on Rarimo to be used on any chain.

Passport-derived profiles enable users to set up an identity solely with their government-issued documents without needing a third-party issuer. Using this functionality, users can verify the authenticity of their documents without disclosing their personal information, thereby creating a profile linked to the specified data. Once the identity profile is established, users can demonstrate the validity of certain information within their documents and ensure the uniqueness of the identity for the application in use.

Passport-derived profiles are also presumed to build the tree, the state of which can be broadcast over the Rarimo network. These tree states allow for the organization of deterministic proof of user uniqueness with additional eligibility proofs. From the cryptographic perspective, passport-derived profiles are compatible with the Iden3 protocol (duplicate keys and signatures can be used). Users can create passport-derived profiles and connect additional verifiable credentials or attributes. Additionally, identity providers can track passport revocation events and automatically revoke/reissue Verifiable Credentials.

5.1 Verifiable Credentials Infrastructure

As mentioned, identity providers can create Verifiable Credentials for a user's identity. Rarimo Core or any connected network could be used as a genesis chain. According to the Iden3 protocol, the state is represented as Figure 2.

The flow of issuing the Verifiable Credentials and using it cross-chain can be seen in Algorigm 2.

This concept shows the universality of digital identity use. As highlighted at the outset of this whitepaper, the type of identity artifacts extends beyond the VCs, NFTs/SBTs, or reputational systems. The Rarimo method allows to work with each component through a unified process, ensuring that integration is easy and seamless.

5.2 Passport-derived Profiles

5.2.1 Creating a Profile in the Rarimo Ecosystem

To receive the profile connected to the passport, the user performs the following actions:

1. The user creates the request to Rarimo to create an account (identity profile).
   (a) The user generates the keypair $\langle \mathsf{pk}, \mathsf{sk} \rangle$ for identity management and calculates the blinder $\beta \leftarrow \mathsf{H}_{\mathsf{zk}}(\mathsf{sk})$.
   (b) The user signs pk with the passport signature $\sigma_{\text{pass}} \leftarrow \mathsf{Sign}(\mathsf{pk}, \mathsf{sk}_{\mathsf{pass}})$ by using passport active authentication mechanism and providing $\mathsf{H}_{\mathsf{zk}}(\mathsf{pk})[: 64]$ as a $challenge$.
   (c) The user generates proof π that $\mathsf{pk}_{\mathsf{pass}}$ is a public key belonging to the passport (signed by some key from the ICAO list, without revealing who exactly is the issuer authority of the passport) and that identity management pk is signed by the passport $\mathsf{sk}_{\mathsf{pass}}$. The circuit then calculates $d_{\text{commit}} \leftarrow \mathsf{H}_{\mathsf{zk}}(d_1 \parallel \beta)$.

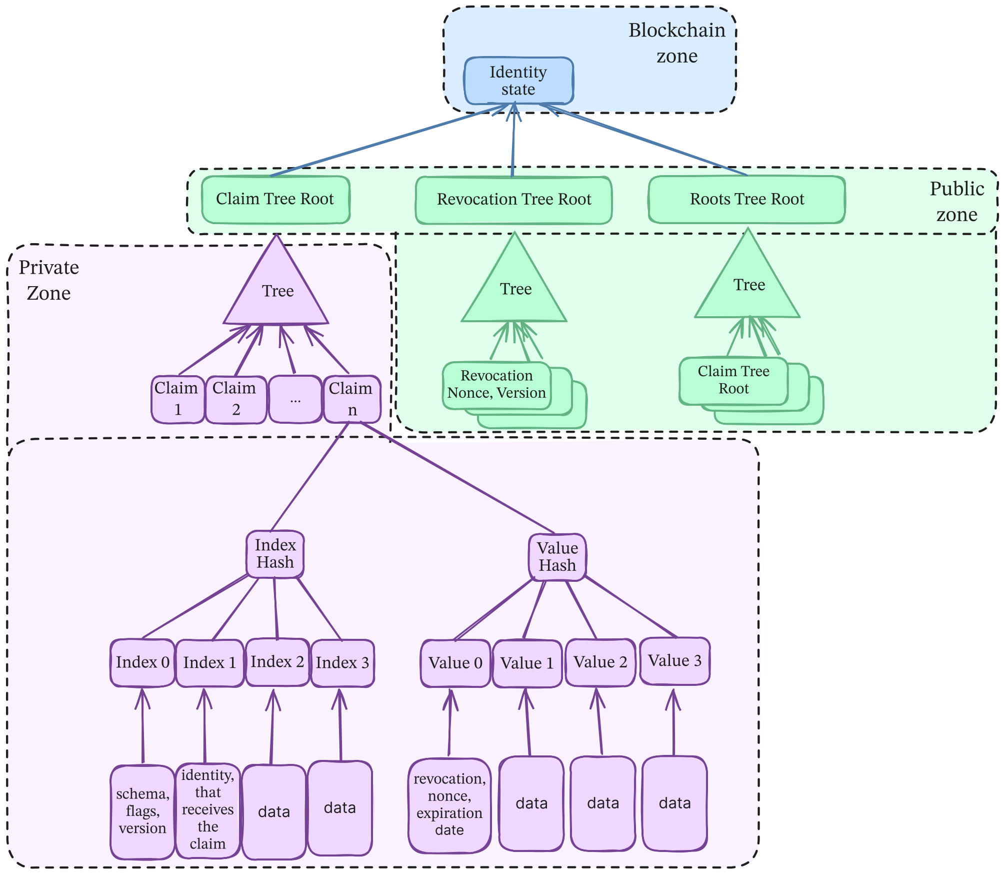


Figure 2: Iden3 architecture.

(d) The user submits a profile creation transaction with the following data:

i. pk

ii. $\mathsf{pk}_{\text{pass}}$

iii. σ_{pass}

iv. $\pi \leftarrow \mathsf{Prove}(\mathsf{pp}, \mathsf{pk}_{\text{pass}} \sim \text{passport} \wedge d_1 \sim \text{passport} \wedge \mathsf{pk}_{\text{iss}} \in \mathcal{C})$

2. This way, a specific passport is linked with identity keys $(\mathsf{pk} :: \mathsf{pk}_{\text{pass}})$ at the Rarimo level (tree leaf in a Sparse Merkle Tree $\mathcal{T}_{\text{ID}}$) (Figure 3).

(a) Leaf position: $\mathsf{ID}_{\text{pos}} \leftarrow \mathsf{H}_{\text{zk}}(\mathsf{pk} \parallel \mathsf{pk}_{\text{pass}})$

(b) Leaf value: $v \leftarrow \mathsf{H}_{\text{zk}}(\mathsf{ID}_{\text{pos}}, d_{\text{commit}})$

The user can try to create several profiles using different pk for the same passport (and, consequently, the same $\mathsf{pk}_{\text{pass}}$). In this case, Rarimo will reject the request until the current identity key is revoked, which can only be done by the passport owner.

5.2.2 Profile Revocation

The passport owner can revoke the current pk associated with the passport.

1. The user creates a request to revoke the pk key:

(a) The user signs the current pk using the passport $\sigma'_{\text{pass}} \leftarrow \mathsf{Sign}(\mathsf{pk}, \mathsf{sk}_{\text{pass}})$ (note that $\sigma'_{\text{pass}} \neq \sigma_{\text{pass}}$).

2. The user submits a profile key revocation transaction with the following data:

Algorithm 2 Issuance of the VC and usage it cross-chain

```
Step 0. VC issuance
1. The identity provider $\mathcal{P}$ issues identity owner $\mathcal{O}$ a verifiable credential VC (claim) on the operational chain and updates the identity state $S_{\mathcal{P}}$. The identity provider can add an arbitrary number of other identities $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_n$ to the state without updating its cross-chain after each new identity (it allows reduced costs and state transitions allow the verification of the correctness of the new state).
2. If $\mathcal{O}$ urgently requires her identity on the external chain, she can call Rarimo Core and update appropriate $\mathcal{T}$ with a state $S$. Otherwise, they can wait for the Identity provider or Oracles to do that.
Step 1. Updating the state in Rarimo Core
1. The block proposer $\mathcal{P}$ adds this transaction (with the updated state $S$) to the block $B$, signs it with $\sigma_{\mathcal{P}}$, and transfers it to other validators. If more than $\frac{2}{3}$ of current validators have signed the block (if the $\mathsf{Cert}(\mathcal{V})$ is valid), it is added to the blockchain and is irreversible.
2. Confirmation of the block by the $\mathsf{Cert}(\mathcal{V})$ automatically initiates the creation of the witness needed by the message client.
3. The Prover listens to events in the Rarimo Core and, in the case of needed block confirmation, receives the needed witness. The witness is a data structure that includes:
    3.1 $\mathsf{tx}'_{\text{state}}$
    3.2 The header $h$ of the block in which the transaction was added
    3.3 Evidence $\mathsf{Cert}(\mathcal{V})$ of the block in which the transaction was added
    3.4 $\mathsf{Path}(\mathsf{tx}'_{\text{state}}, \mathsf{Root}(\mathcal{T}))$
Step 2. Verification
1. The Prover provides the witness $w$ to the Rarimo Contract, and the contract performs the following verification:
    1.1 $w \notin \mathcal{W}_{\text{previous}}$, where $\mathcal{W}_{\text{previous}}$ is the list of previously used witnesses.
    1.2 $\mathsf{Cert}(\mathcal{V})$ is valid.
    1.3 $\mathsf{Root}(\mathcal{T})$ is in block $B$.
    1.4 $S \to S'$ is a valid state transition.
2. Then the user can prove her claim ownership directly on the destination chain.
```

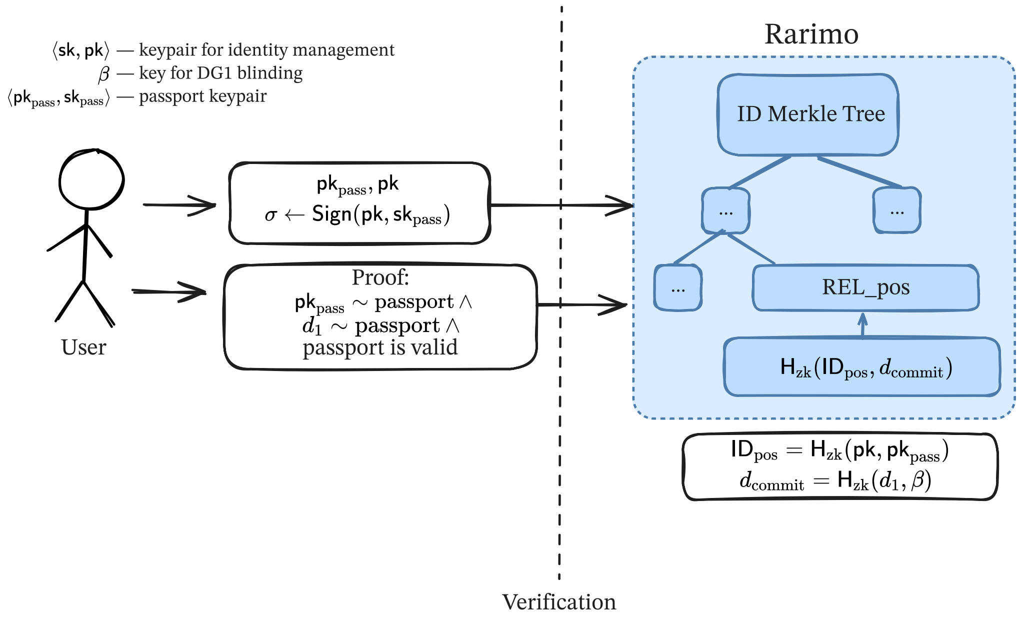

$\langle \mathsf{sk}, \mathsf{pk} \rangle$ — keypair for identity management
β — key for DG1 blinding
$\langle \mathsf{pk}_{\text{pass}}, \mathsf{sk}_{\text{pass}} \rangle$ — passport keypair
User
$\mathsf{pk}_{\text{pass}}, \mathsf{pk}$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{pk}, \mathsf{sk}_{\text{pass}})$
Proof:
$\mathsf{pk}_{\text{pass}} \sim$ passport $\wedge$
$d_1 \sim$ passport $\wedge$
passport is valid
Verification
Rarimo
ID Merkle Tree
...
...
...
REL_pos
$\mathsf{H}_{\text{zk}}(\mathsf{ID}_{\text{pos}}, d_{\text{commit}})$
$\mathsf{ID}_{\text{pos}} = \mathsf{H}_{\text{zk}}(\mathsf{pk}, \mathsf{pk}_{\text{pass}})$
$d_{\text{commit}} = \mathsf{H}_{\text{zk}}(d_1, \beta)$

Figure 3: The process of creating an identity profile.

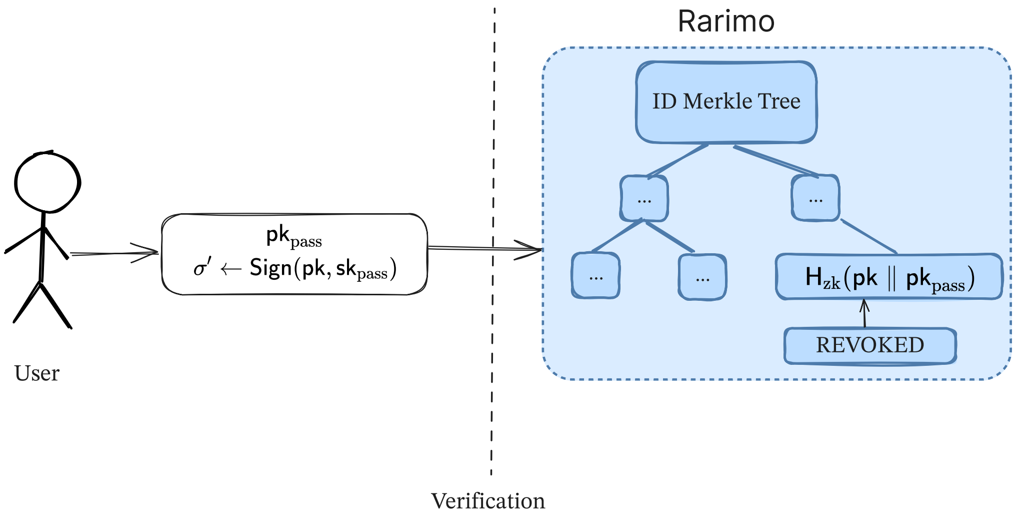

Rarimo
ID Merkle Tree
...
...
...
...
$\mathsf{H_{zk}}(\mathsf{pk} \parallel \mathsf{pk_{pass}})$
REVOKED
$\mathsf{pk_{pass}}$
$\sigma' \leftarrow \mathsf{Sign}(\mathsf{pk}, \mathsf{sk_{pass}})$
User
Verification

Figure 4: The process of revocation of the profile.

(a) $\mathsf{pk_{pass}}$
(b) pk
(c) σ'_{pass}

3. Identity contract calculates the tree position as $\mathsf{ID_{pos}} \leftarrow \mathsf{H_{zk}}(\mathsf{pk} \parallel \mathsf{pk_{pass}})$ and verifies the signature $\mathsf{SigVerify}(\sigma'_{\text{pass}}, \mathsf{pk_{pass}}, \mathsf{pk}) \overset{!}{=} \mathsf{accept}$
4. If the signature is correct, the $\mathcal{T}_{\text{ID}}$ is updated with $\mathsf{ID_{pos}} = \mathsf{Revoked}$ (see Figure 4).

The user can customize the profile revocation logic. For example, the user can set a revoking event with another public key from a predefined set $\{\mathsf{pk}_1, \mathsf{pk}_2 ... \mathsf{pk}_n\}$ signs the revocation request (or threshold quorum of them).

The user can also turn off key revocation. Then, it can only be revoked by creating a new profile with the same keys.

5.2.3 Reissuance

The user can redefine a profile with a new identity key (separately or with the revocation procedure). For this:

1. The user generates a new key pair $\langle \mathsf{pk}', \mathsf{sk}' \rangle$
2. The user signs pk' using the passport $\sigma_{\text{pass}} \leftarrow \mathsf{Sign}(\mathsf{pk}', \mathsf{sk_{pass}})$
3. The user submits a profile update transaction with the following data:
   (a) pk'
   (b) $\mathsf{pk_{pass}}$
   (c) σ_{pass}
4. The profile is updated at the Rarimo layer $(\mathsf{pk}' :: \mathsf{pk_{pass}})$
5. The $\mathcal{T}_{\text{ID}}$ is updated (Figure 5):
   (a) Set $\mathsf{Revoked}$ constant to the $\mathsf{H_{zk}}(\mathsf{pk} \parallel \mathsf{pk_{pass}})$ tree position.
   (b) Set $\mathsf{H_{zk}}(\mathsf{ID_{pos}} \parallel d'_{\text{commit}})$ to the $\mathsf{ID}'_{\text{pos}} = \mathsf{H_{zk}}(\mathsf{pk}' \parallel \mathsf{pk_{pass}})$ tree position.

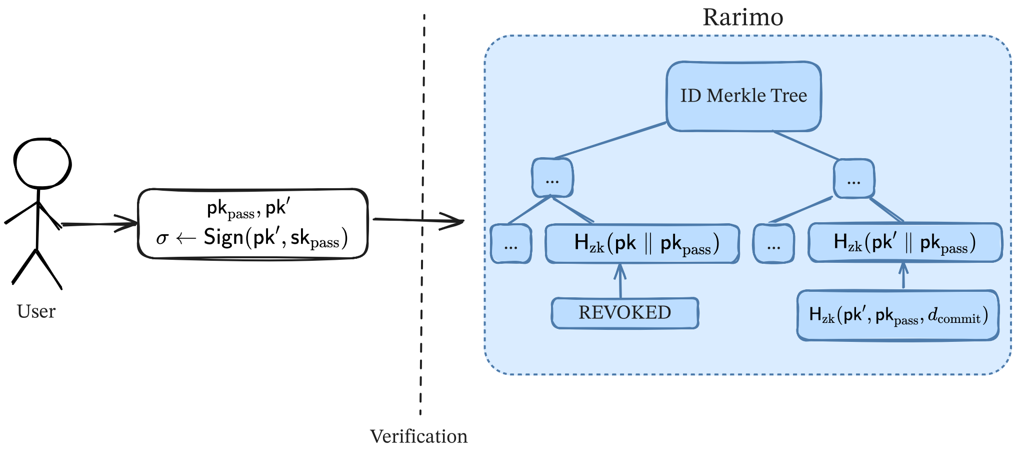

Rarimo
ID Merkle Tree
...
$H_{zk}(pk \parallel pk_{pass})$
$H_{zk}(pk' \parallel pk_{pass})$
REVOKED
$H_{zk}(pk', pk_{pass}, d_{commit})$
pk_{pass}, pk'
$\sigma \leftarrow Sign(pk', sk_{pass})$
User
Verification

Figure 5: The process of updating an identity profile.

5.2.4 Auth Proof with Selected Data Disclosure

At a specific time, the user can prove that he has a registered profile and data set connected to the corresponding passport. For example, there is an event E with the identifier ID_E. Additionally to the proof of the existence in the $\mathcal{T}_{\text{ID}}$ the user wants to prove:

1. Their citizenship is included in the list of allowed.
2. Expiration date of their passport is within some time bounds $(\tau^-_{\text{exp}}, \tau^+_{\text{exp}})$.
3. The date of their birth is within some time bounds $(\tau^-_{\text{birth}}, \tau^+_{\text{birth}})$.
4. Their sex is equal to F value (data is disclosed)

The user generates the following proof (Figure 6). Below, ν denotes the *nullifier*:

$$\left|\begin{array}{c}
\textbf{pub\_signals:}\\
\nu \leftarrow \mathsf{H}(\mathsf{sk} \parallel \mathsf{ID}_E)\\
\mathsf{Root}(\mathcal{T}_{\text{ID}}), \mathsf{ID}_E, \mathsf{Sel}, (\tau^-_{\text{exp}}, \tau^+_{\text{exp}}), (\tau^-_{\text{birth}}, \tau^+_{\text{birth}}), M_{\text{cit}}, F^*_{\text{pass}}, \mathsf{AA}_{\text{flag}}\\
\textbf{priv\_signals:}\\
\mathsf{sk}, \mathsf{pk}_{\text{pass}}, d_1\\
\sigma_{\text{pass}} \leftarrow \mathsf{Sign}(\mathsf{ID}_E, \mathsf{sk}_{\text{pass}})\\
\textbf{circuit\_logic:}\\
\mathsf{pk} \in \mathcal{T}_{\text{ID}}:\\
\mathsf{ID}_{\text{pos}} \overset{!}{=} \mathsf{H}_{\text{zk}}(\mathsf{pk} \parallel \mathsf{pk}_{\text{pass}}) \wedge\\
\mathsf{ID}_{\text{pos}}.\mathsf{value} \overset{!}{=} \mathsf{H}_{\text{zk}}(\mathsf{H}_{\text{zk}}(\mathsf{pk} \parallel \mathsf{pk}_{\text{pass}}) \parallel \mathsf{H}_{\text{zk}}(d_1 \parallel \beta))\\
\mathsf{H}_{\text{zk}}(\mathsf{sk} \parallel \mathsf{ID}_E) \overset{!}{=} \nu \wedge\\
\mathsf{Select}(\mathsf{Selector}, d_1) \overset{!}{=} F^*_{\text{pass}} \wedge\\
(d_1.\mathsf{Cit} \wedge M_{\text{cit}}) \overset{!}{=} 0 \wedge\\
d_1.\mathsf{exp} \in (\tau^-_{\text{exp}}, \tau^+_{\text{exp}}) \wedge d_1.\mathsf{birth} \in (\tau^-_{\text{birth}}, \tau^+_{\text{birth}})\\
\textbf{if } \mathsf{AA}_{\text{flag}} = 0, \text{ require } \mathsf{SigVerify}(\sigma_{\text{pass}}, \mathsf{ID}_E, \mathsf{pk}_{\text{pass}}) \overset{!}{=} \mathsf{accept}
\end{array}\right| \quad (2)$$

To ensure the user's uniqueness, it is necessary to fix $\mathsf{Root}(\mathcal{T}_{\text{ID}})$. To generate proof, a request should be made to the historical state of the Identity Smart Contract. It is important to note that this contract cannot be frozen for a specific event, as it provides a global state and should always be available for state updates. If the user wants to take another action using the same identity key, the nullifier will be matched, and the transaction will be reversed.

Selector Sel is needed to disclose personal information from a passport selectively. Each private data signal is multiplied by a corresponding Sel bit. If the bit is 0, data is blinded by multiplying it with 0. If the bit is 1, data is sent to the output signal without modifications.

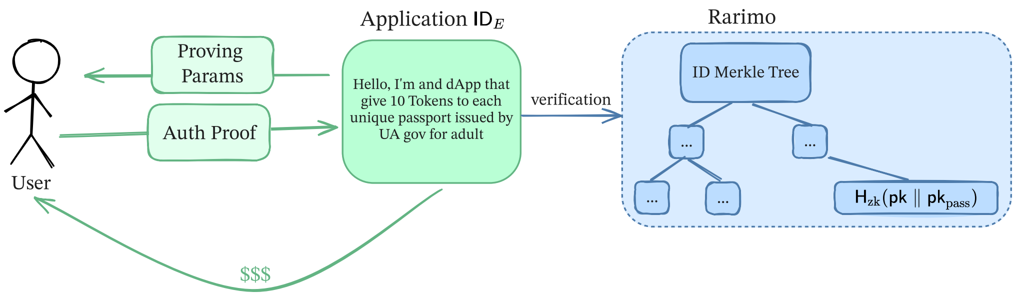
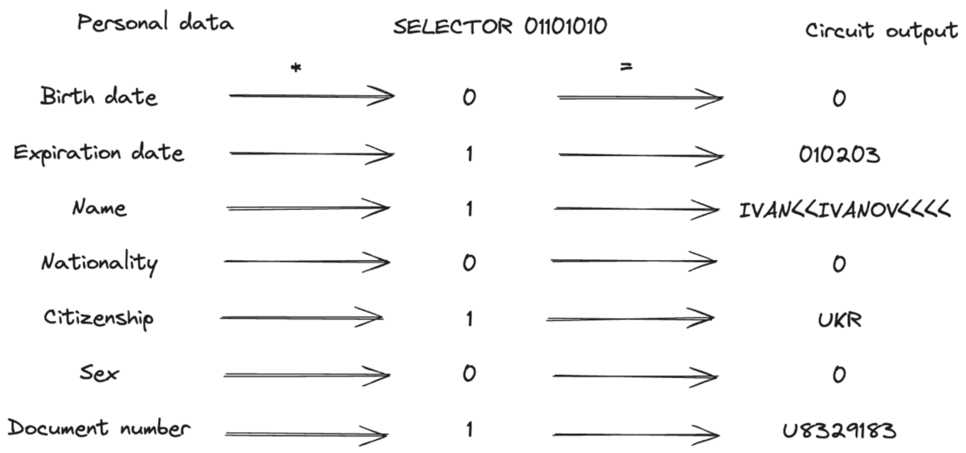

Application ID$_E$
Rarimo
Proving Params
Auth Proof
User
Hello, I'm and dApp that give 10 Tokens to each unique passport issued by UA gov for adult
verification
ID Merkle Tree
...
$H_{zk}(pk \| pk_{pass})$
$$$

Figure 6: The process of proving the identity ownership.



Personal data	SELECTOR 01101010	Circuit output
Birth date	0	0
Expiration date	1	010203
Name	1	IVAN<<IVANOV<<<<
Nationality	0	0
Citizenship	1	UKR
Sex	0	0
Document number	1	U8329183

Figure 7: An example of selector functioning.

By applying this algorithm, we can use the same circuit for any revealed and unrevealed personal data set.

6 Private Social Graph

The mentioned architecture allows the creation of several types of actions privately but with the ability to prove them in the future (even recursively). Examples of such actions are the following:

- Prove the credential and attestation ownership at a particular time (and continuously).
- Prove that some data in the identifier passed the verification defined by a particular dApp or verifier.
- Give/receive attestation to another identity owner (of the whole identity or some parameter).
- Prove that a particular user initiated some actions.
- Prove that specific actions were initiated by the set of identities (group of people), and all/some/at least one satisfies defined requirements.
- Prove that some claims and/or passports are connected to one identity.
- Prove that the user is(not) included in specific lists or groups.
- Prove that the sent/received attestation came to/from the identity or service that satisfies some criteria without revealing the identity owner.

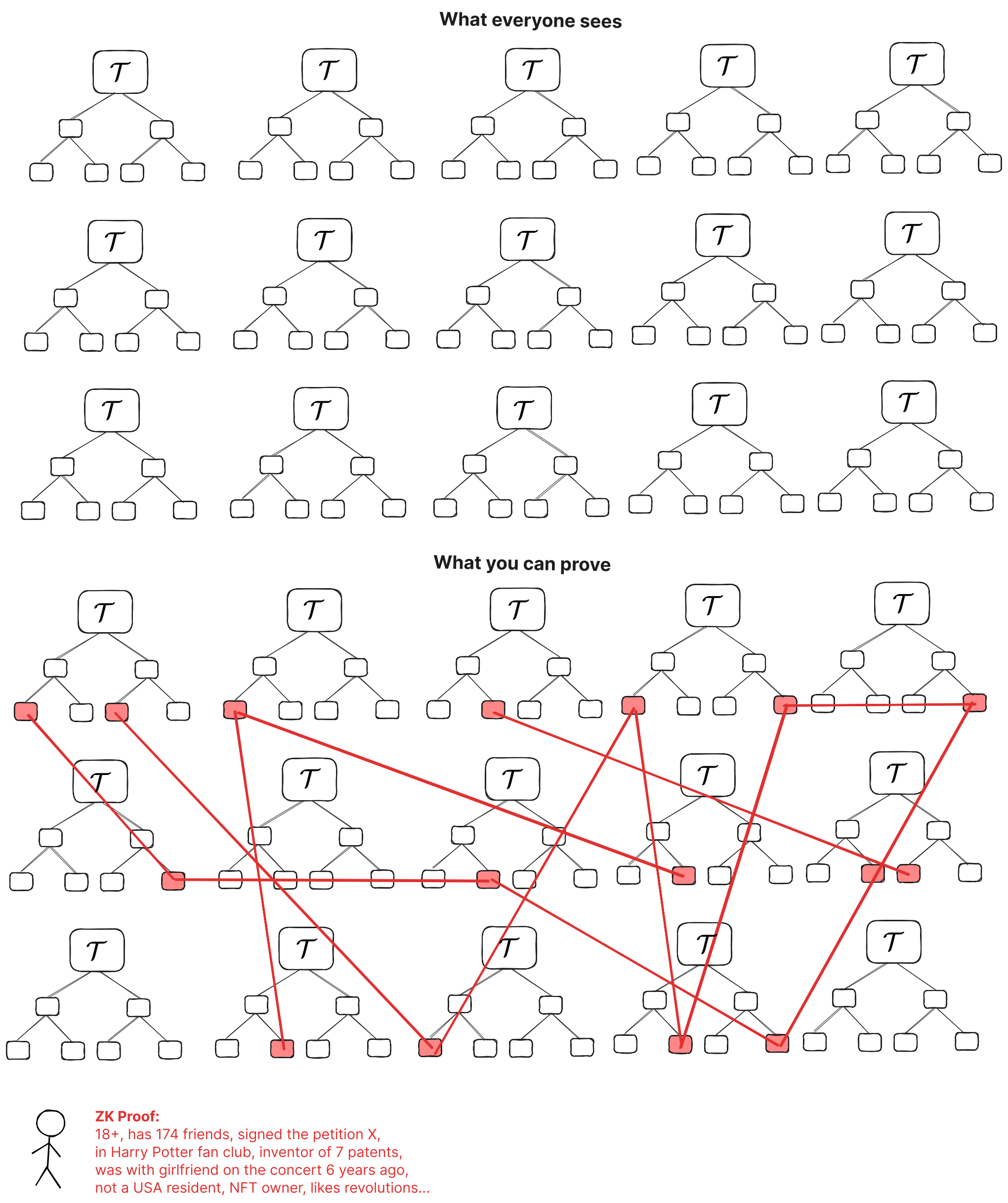

What everyone sees
What you can prove
ZK Proof:
18+, has 174 friends, signed the petition X,
in Harry Potter fan club, inventor of 7 patents,
was with girlfriend on the concert 6 years ago,
not a USA resident, NFT owner, likes revolutions...

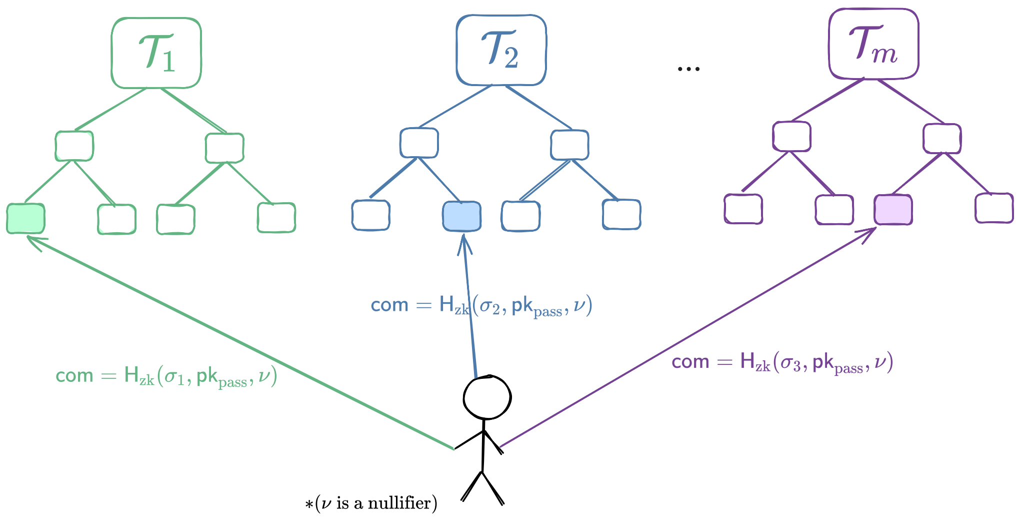
Figure 8: Proofs over the private metagraph.

All these actions are invisible to the public until the owner decides to prove them.

6.1 Identity Profile Liveness

Here is an example that allows users to prove the ownership of a particular passport over time. Imagine a set of Trees $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \dots, \mathcal{T}_m$, each indicating the authentication tree in the particular time range (each tree represents a particular month, for example). Each Tree has an identifier $\mathsf{ID}_1, \mathsf{ID}_2, \dots, \mathsf{ID}_m$, representing a large 256-bit number. Trees are self-manageable, meaning users can update them with their identity artifacts (no trusted party is required).

By default, each $\mathcal{T}_i, i \in \{1, \dots, m\}$ is zero, meaning all its leaves are equal to 0. Then, it will be updated with specific commitments representing the ownership of passports.



$\mathcal{T}_1$
$\mathcal{T}_2$
...
$\mathcal{T}_m$
$\mathsf{com} = \mathsf{H}_{\mathrm{zk}}(\sigma_2, \mathsf{pk}_{\mathrm{pass}}, \nu)$
$\mathsf{com} = \mathsf{H}_{\mathrm{zk}}(\sigma_1, \mathsf{pk}_{\mathrm{pass}}, \nu)$
$\mathsf{com} = \mathsf{H}_{\mathrm{zk}}(\sigma_3, \mathsf{pk}_{\mathrm{pass}}, \nu)$
$*(\nu$ is a nullifier$)$

Figure 9: Collecting signature samples in time trees.

When the user wants to update the tree, they must:

1. Sign the ID with their passport key $\sigma_{\text{pass}} \leftarrow \mathsf{Sign}(\mathsf{ID}, \mathsf{sk}_{\text{pass}})$.
2. Create the commitment $\mathsf{com} \leftarrow \mathsf{H}(\sigma_{\text{pass}}, \mathsf{pk}_{\text{pass}}, \nu)$.
3. Create a proof:

$$\left|\begin{array}{c}\textbf{pub\_signals:}\\ \mathsf{com}, \mathsf{ID}\\ \textbf{priv\_signals:}\\ \sigma_{\text{pass}}, \mathsf{pk}_{\text{pass}}, \nu, \sigma_{\text{iss}}, \mathcal{C}, S_{\text{OD}}\\ \textbf{circuit\_logic:}\\ \mathsf{H}(\sigma_{\text{pass}}, \mathsf{pk}_{\text{pass}}, \nu) \overset{!}{=} \mathsf{com}\wedge\\ \mathsf{SigVer}(\sigma_{\text{pass}}, \mathsf{ID}, \mathsf{pk}_{\text{pass}}) \overset{!}{=} 0\wedge\\ \mathsf{pk}_{\text{iss}} \in \mathcal{C} \wedge \mathsf{pk}_{\text{pass}} \in S_{\text{OD}}\wedge\\ \mathsf{SigVerify}(\sigma_{\text{iss}}, S_{\text{OD}}, \mathsf{pk}_{\text{iss}}) \overset{!}{=} 0\end{array}\right| \tag{3}$$

4. Send the transaction with an update on the tree with com.

In such a way, the user leaves the proofs bonded to the same passport public key. These proofs are designed to prove that passports use active authentication constantly within the corresponding time frame.

If the user needs to prove they have used the AA method in the sequence of proofs, they can create the following proof:

$$\left|\begin{array}{c}\textbf{pub\_signals:}\\ \{\mathcal{T}_i\}_{i=1}^{k}\\ \textbf{priv\_signals:}\\ \mathsf{pk}_{\text{pass}}, \{\mathsf{com}_i\}_{i=1}^{k}, \{\mathsf{Path}(\mathsf{com}_i, \mathcal{T}_{\text{root}}^{(i)})\}_{i=1}^{k}, \{\sigma_i\}_{i=1}^{k}, \nu\\ \textbf{circuit\_logic:}\\ \forall i : \mathsf{com}_i \in \mathcal{T}_i \wedge \mathsf{SigVerify}(\sigma_i, \mathsf{ID}_i, \mathsf{pk}_{\text{pass}}) \overset{!}{=} \mathsf{accept}\end{array}\right| \tag{4}$$

This approach doesn't allow the creation of millions of new backdated passports and attaching them to profiles with a high reputation level (only owners of passports confirmed over time should be eligible or should have more reputation power than fresh passports). The verifier application can set the time threshold of the reputation-calculating approach that decreases the impact of freshly printed documents on the final use case.

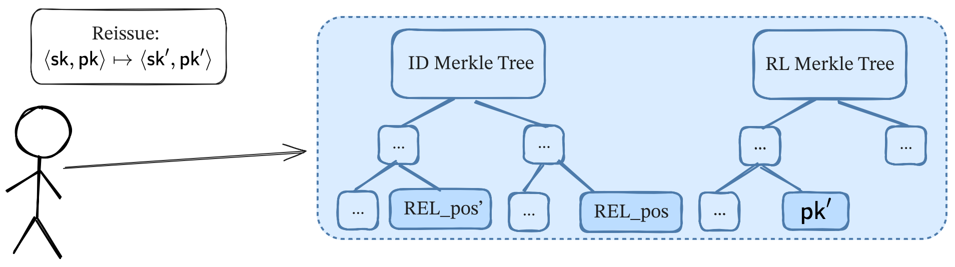

Reissue:
$\langle \mathsf{sk}, \mathsf{pk} \rangle \mapsto \langle \mathsf{sk}', \mathsf{pk}' \rangle$
ID Merkle Tree
RL Merkle Tree
...
REL_pos'
REL_pos
pk'

Figure 10: The process of building rate limiting tree.

6.2 Attestations

The same commitments and verification methods could be applied to other social artifacts. Suppose the user wants to prove they participated in several events with another person. In that case, they can put such attestations in event trees and prove the connection with a particular identity in the future. Each credential, action, and property can be proved the same way.

6.3 Rate Limiting Trees

Rarimo social protocol allows users to generate anonymous proofs based on their identity profile (and connected attributes). The problem arises when some protocols want to ensure users can interact with it only once.

The naive solution would be publishing a commitment to the unique user's credentials (for instance, the public key of the passport), but this would expose users to a dictionary attack. An attacker with a set of known credential data could trace the user's and protocol interaction.

We propose another way to solve this problem with a compact security gadget. The procedure of creating the identity profile allows the tracking of new identity public keys and adding them to a Rate-Limiting Sparse Merkle Tree $\mathcal{T}_{\text{RL}}$. The organization that requires limitations for user operation can build this tree off-chain (only for time bounds it's interested in).

The rate-limiting tree functioning is pretty simple. Based on time (block height), the service that requires the tree fetches all events connected with updating identity profiles and builds the Tree with the new pk (new public keys connected to identity).

To operate with the application, the user should generate a non-inclusion proof (that they have not reissued the identity). Users cannot generate appropriate proofs if they change their identity after tracking the starting point.

$$\left|\begin{array}{c} \textbf{pub\_signals}: \\ \mathcal{T}_{\text{RL}} \\ \textbf{priv\_signals}: \\ \mathsf{pk} \\ \textbf{circuit\_logic}: \\ \mathsf{pk} \notin \mathcal{T}_{\text{RL}} \end{array}\right| \tag{5}$$

This proof is an extension to action proof (it could be used with joining to initiator's pk within proof construction).

6.4 Challenges

- Creating user-oriented queries for working with the private graph (the user can recursively prove some relations or actions hidden in the graph)
- Creating queries that can calculate some results based on the graph but without revealing particular events.

7 Utilities of the Token

The Rarimo protocol has its internal utility token – RMO, the main utilities of which are as follows: (1) payment of transaction fees, (2) governance within the DAO of the Rarimo protocol, (3) selection of validators of the Rarimo Core, and (4) rewards to oracles and signers for transferring data cross-chain.

7.1 Fees

The primary utility of the RMO token is to pay transaction fees within the Rarimo protocol. Transaction fees cover operations involving the transfer of identity states, tokens, and messages across connected networks.

7.2 Governance

RMO tokens will be used as a governance tool for the DAO of the Rarimo protocol.

Any RMO token holder can participate in Rarimo's governance process. From time to time, it is expected that the community of the Rarimo protocol will adopt and implement a governance policy, acceptable for the Rarimo DAO, but, in general, it is suggested that the proposal's creator must block a number of the RMO tokens, prescribed in the governance policy, to submit a proposal. The blocked tokens will be burned if the other participants consider the proposal fraudulent. After that, a voting round will begin, the duration of which may be approved in the governance policy. If a certain amount of all voters support the proposal (the exact amount should be reflected in the governance policy), it will be deemed accepted, and certain requirements concerning the quorum, which may be prescribed in the governance policy, must be met to update the protocol. The proposal's creator will receive a reward, the amount of which will be determined by the protocol. If the proposal is not accepted and users do not consider it fraudulent, the blocked tokens will be returned to the user's account.

7.3 Selecting Validators of the Rarimo Core

The RMO token is designed to ensure Rarimo security. The smart contract mandates a minimum amount of RMO to be sent along with the transaction to be valid. Once a node has staked a certain amount of RMO, other token holders can also stake their tokens for that node. The nodes with the most tokens staked become the guarantors of the protocol, resulting in an inclusive process. Users can report it through an on-chain transaction if a validator violates the protocol. Other users can then report the violation. The validator's stake will be transferred to the community pool if the violation is approved. Two types of violations result in up to 100% stake penalty: signing two or more blocks of the same height and attempting to add conflicting transactions within one block. This ensures that bad actors will be penalized and that validators will act together to ensure the protocol's security.

7.4 Rewards for Signers for Fetching and Confirming External Data

Another primary utility of the RMO token is to reward data transfer from connected networks to the Rarimo core. In this context, the degree of decentralization within the Oracle layer is significantly important. By distinguishing this role from the validator node, technically and economically, it enables participants to choose accounting systems that align with their existing infrastructure and assumptions for indexing.

References

[Buc16] Ethan Buchman. Tendermint: Byzantine Fault Tolerance in the Age of Blockchains. https://knowen-production.s3.amazonaws.com/uploads/attachment/file/1814/Buchman_Ethan_201606_Msater%2Bthesis.pdf, 2016. [Online; accessed 2-Apr-2024].

[GG20] Rosario Gennaro and Steven Goldfeder. One Round Threshold ECDSA with Identifiable Abort. https://eprint.iacr.org/2020/540, 2020. [Online; accessed 2-Apr-2024].

[Ide] Iden3. Iden3 Protocol Specifications. https://docs.iden3.io/protocol/spec/. [Online; accessed 2-Apr-2024].

[MBS23] Alessio Meneghetti Michele Battagliola, Riccardo Longo and Massimiliano Sala. A Provably-Unforgeable Threshold EdDSA with an Offline Recovery Party. https://arxiv.org/abs/2009.01631, 2023. [Online; accessed 2-Apr-2024].

Disclaimer

The content of this document is for informational purposes only, you should not construe any such information or other material as legal, tax, investment, financial, or other advice. Nothing contained in this document constitutes a solicitation, recommendation, endorsement, or offer of any person to buy or sell any financial instruments in any jurisdiction.

All content of this document is information of a general nature and does not address the circumstances of any particular individual or entity. Nothing in the content of this document constitutes professional or financial advice or both, nor does any information in the content of this document constitute a comprehensive or complete statement of the matters discussed or the law relating thereto. You alone assume the sole responsibility of evaluating the merits and risks associated with the access or use, or both, of the content of this document or the described in this documents project (the "Project"), or both, before making any decisions based on the information contained in the content of this document.

When interacting with this document or the Project, or both, in any way, you acknowledge and agree that:

- you understand and accept that blockchain networks, the smart contract system concept, the underlying software application and software platform (such as the Ethereum blockchain) is still in an early development stage and unproven, why there is no warranty that the process for creating the Project, will be uninterrupted or error-free and why there is an inherent risk that the software could contain weaknesses, vulnerabilities or bugs;
- you understand and accept that the blockchain technology allows new forms of interaction and that it is possible that certain jurisdictions will apply existing regulations on, or introduce new regulations addressing, blockchain technology based applications, which may be contrary to the current setup of the blockchain network and smart contract system and which may, inter alia, result in substantial modifications of the smart contract system or the Project, or both, including its termination;
- you understand and accept that the creation of any tokens and the development of the Project may be abandoned for a number of reasons, including lack of interest from the public, lack of funding, lack of commercial success or prospects (e.g. caused by competing projects);
- you understand and accept that the Project may give rise to other, alternative to such projects;
- you understand and accept that the blockchain network, smart contract system concept, the underlying software application and software platform (e.g. the Ethereum blockchain) may be exposed to attacks by hackers or other persons, impacting the ability to develop the Project;
- you have a deep understanding of the functionality, usage, storage, transmission mechanisms and intricacies associated with digital assets, and blockchain-based software systems;
- you understand and accept that the token of the Project (RMO) described herein is a cryptographic utility token. The purpose of such a token is to enable usage and facilitate the participation and interaction with the Project, as described in this document, for governance, and as service fee in relation to relevant activities on the protocol associated with the Project, as well as other usage as described herein;
- you understand and accept that ownership of the RMO token carries no rights, whether express or implied, other than as described in this document, if and to the extent the Project has been successfully completed and launched;
- you understand and accept that by receiving of the RMO token herein no form of partnership, joint venture or any similar relationship between you and any person is created;
- you understand and accept that except as otherwise expressly set forth herein, the receipt of the RMO token: (i) does not provide you with rights of any form with respect to any one or more of the following: any person or its revenues or assets, including, without limitation, any voting, distribution, redemption, liquidation, proprietary (including all forms of their intellectual property rights) or other financial or legal rights; (ii) is not a loan to any person; and (iii) does not provide you with any ownership, equity, or other interest in any person; and
- you understand and accept that: (i) the RMO token does not represent or confer on you any ownership right, shareholding, participation, right, title, or interest of any form with respect to any person, or any of its revenues or assets, including without limitation any right to receive future revenue, dividends, shares, ownership right or stake, share or security, any voting, distribution, redemption, liquidation, proprietary (including all forms of intellectual property), or other financial or legal rights or equivalent rights, or intellectual property rights or any other form of participation in or relating to the Project; (ii) is not intended to be a representation of currency or money (whether fiat or virtual or any form of electronic money), security, commodity, or any other kind of financial instrument or investment; (iii) is not intended to represent any rights under a contract for differences or under any other contract the purpose or pretended purpose of which is to secure a profit or avoid a loss; (iv) is not a loan and is not intended to represent a debt owed by any person, and there shall be no expectation of profit or interest income arising in connection therewith; (v) is not any form of financial derivative; (vi) is not any form of commercial paper or negotiable instrument, and (vii) is not any commodity or asset that any person is obliged to redeem or purchase, or both.