Citadel Protocol Specification

Dusk Network

March 27, 2023

Contents

1 General Overview

1.1 What is Citadel

A Self-Sovereign Identity (SSI) protocol serves the purpose of allowing users of a given service to manage their identities in a fully transparent manner. In other words, every user can know which information about them is shared with other parties, and accept or deny any request for personal information.

`Citadel` is a SSI protocol build on top of Dusk Network. Users of a service can get a *license*, which represents their *right* to use such a service. In particular, `Citadel` allows for the following properties:

- **Proof of Ownership:** a user of a service is able to prove ownership of a license that allows them to use such a service.
- **Proof of Validity:** users can prove ownership of a valid license, that has not been revoked.
- **Unlinkability:** no one can link any activity with other activities done in the network.
- **Decentralized Nullification:** when a user spends a license, everyone in the network learns that this happened, so it cannot be spent again.
- **Attribute Blinding:** the user is capable of deciding which information they want to leak, blinding any other sensitive information and providing only the desired one.

1.2 Document Organization

In Section 2 we define all the object types and entities involved in the protocol. In Section 3 we roll out the protocol with full details.

2 Definitions

2.1 The Roles involved

- **User:** An entity that interacts with the wallet to request licenses and prove ownership of those.
- **Service Provider:** An entity offering an off-chain service that receives requests for licenses, and upon acceptance, issues them. It also provides the service upon verification that a service request is correct.

2.2 The Elements involved

- **Request:** A request includes the encryption of a stealth address belonging to the user, where the license has to be sent to. The structure is as follows:

Element	Type	Info.
nonce	-	It is a randomness needed to compute enc.
enc	-	It is a ciphertext of size 3.
(lpk, R)	-	It is a stealth address of the SP.

- **License:** A license is an asset that represents the right of a user to use a given service. The structure is as follows:

Element	Type	Info.
pos	-	It is the position of the license into a Merkle tree of licenses.
nonce	-	It is a randomness needed to compute enc.
enc	-	It is a ciphertext of size 4.
(lpk, R)	-	It is a stealth address of the user.

- **LicenseProverParameters:** A prover needs some auxiliary parameters to compute the proof that nullifies a license when desired to be spent. The structure is as follows:

Element	Type	Info.
lpk'	-	The license public key prime.
$\mathsf{sig}_{\mathsf{lic}}$	-	The signature of the license.
com_0^{hash}	-	A hash commitment of the public key of the SP.
com_1	-	A Pedersen Commitment of the attributes.
com_2	-	A Pedersen Commitment of the c value.
$\mathsf{tx\_hash}$	-	The hash of the transaction calling the nullifying contract.
$\mathsf{sig}_{\mathsf{tx}}$	-	The signature of the transaction calling the nullifying contract.
$\mathsf{nullifier}_{\mathsf{lic}}$	-	The nullifier of the license.
$\mathsf{merkle\_proof}$	-	Membership proof of the license in the Merkle tree of licenses.

- **Session Cookie:** A session cookie is a secret value known only by the user and the SP. It contains a set of openings to a given set of commitments. The structure is as follows:

Element	Type	Info.
$\mathsf{pk}_{\mathsf{SP}}$	-	The public key of the SP.
attr	-	The attributes of the user.
c	-	The challenge value.
$(\mathsf{s}_0, \mathsf{s}_1, \mathsf{s}_2)$	-	The randomness used to compute the commitments.

3 Protocol Workflow

The workflow is depicted in Figure 1, and described with full details as follows.

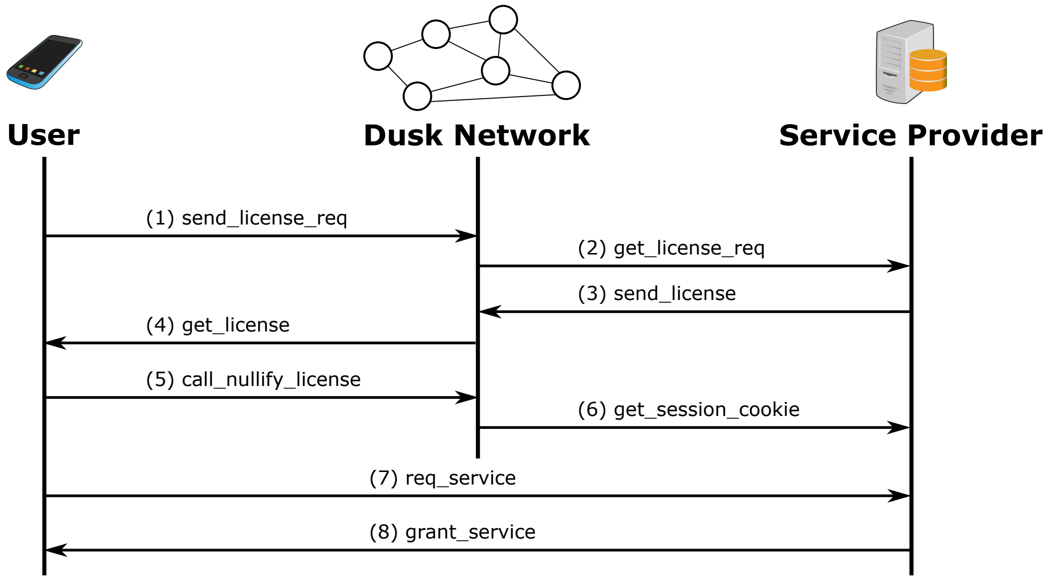

Figure 1: Overview of the protocol messages exchanged between the user, the Dusk Network, and the SP.

1. (**user**) $\mathsf{send\_license\_req}$: TBD.
2. (**SP**) $\mathsf{get\_license\_req}$: Continuously check the network for incoming license requests. Upon receiving the payment from a user, define a set of attributes attr representing the license, and compute a digital signature as follows:

$$\mathsf{sig}_{\mathsf{lic}} = \mathsf{sign\_single\_key}_{\mathsf{sk}_{\mathsf{SP}}}(\mathsf{lpk}, \mathsf{attr})$$

3. (**SP**) $\mathsf{send\_license}$: Compute $\mathsf{enc} = \mathsf{Enc}_{\mathsf{k}_{\mathsf{user}}}((\mathsf{sig}_{\mathsf{lic}}, \mathsf{attr}); \mathsf{nonce})$, set all the parameters of the license struct, and send it to the user.
4. (**user**) $\mathsf{get\_license}$: Receive the license.
5. (**user**) $\mathsf{call\_nullify\_license}$: When desiring to use the license, nullify it by executing a call to the license contract. The following steps are performed:

- The user sets a session cookie sc where $\mathsf{enc} = \mathsf{Enc_k}((\mathsf{s_0}, \mathsf{s_1}, \mathsf{s_2}); \mathsf{nonce})$ and sets the SP as the receiver.
- The user issues the transaction that includes the session cookie described in the previous step, by calling the license contract. In this case, the $\mathsf{tx\_proof}$ is computed as done in the standard Phoenix model to pay for the gas, but into the same circuit, the gadget depicted in Figure 2 is appended.
- The network validators will execute the smart contract, which verifies the proof. Upon success, the session cookie will be forwarded, and the license nullifier $\mathsf{nullifier_{lic}}$ will be added to a shared list of nullifiers.

6. (**SP**) $\mathsf{get\_session\_cookie}$: Receive the session cookie sc.
7. (**user**) $\mathsf{req\_service}$: Request the service to the SP, establishing communication using a secure channel, and providing the tuple $(\mathsf{tx\_hash}, \mathsf{pk_{SP}}, \mathsf{attr}, c, \mathsf{sc})$.
8. (**SP**) $\mathsf{grant\_service}$: Grant or deny the service upon verification of the following steps:
   - Check whether or not the values $(\mathsf{attr}, \mathsf{pk_{SP}}, c)$ are correct.
   - Check whether or not the openings $((\mathsf{pk_{SP}}, \mathsf{s_0}), (\mathsf{attr}, \mathsf{s_1}), (c, \mathsf{s_2}))$ match the commitments $\mathsf{com}_0^{hash}, \mathsf{com}_1, \mathsf{com}_2$ found in the transaction $\mathsf{tx\_hash}$.

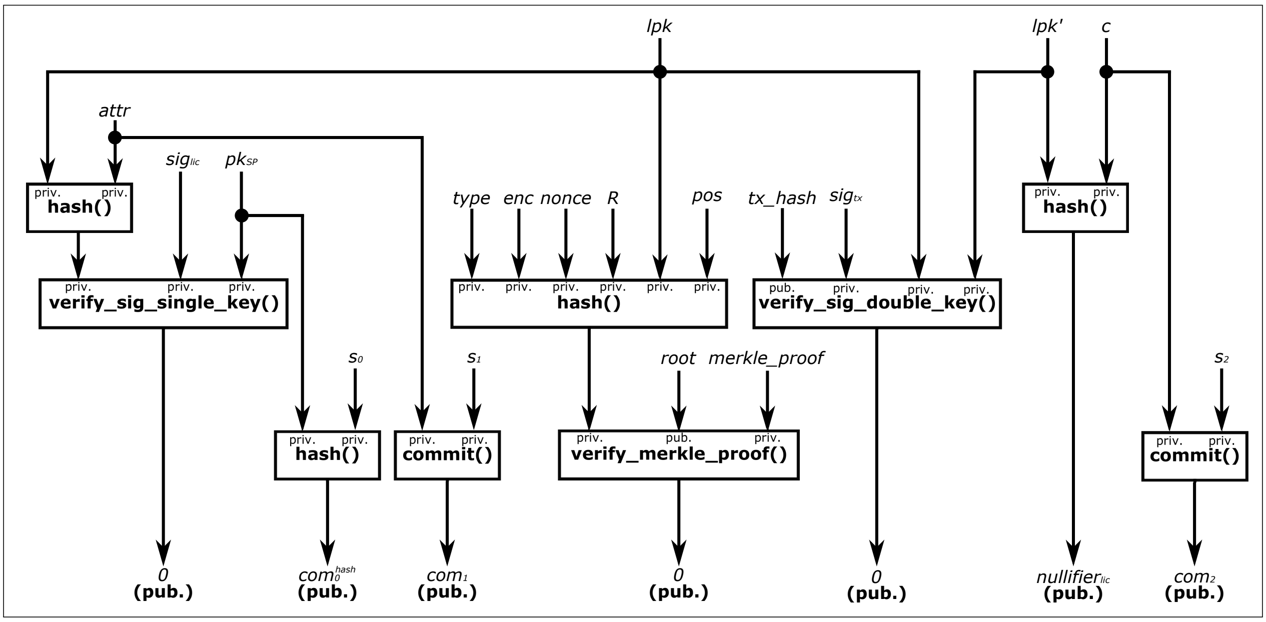

Figure 2: Arithmetic circuit for proving a license's ownership.

Furthermore, the SP might request the user to nullify the license they are using (i.e. this is a single-use license, like entering a concert). This is done through the computation of $\mathsf{nullifier_{lic}}$. The deployment of this part of the circuit has two different possibilities:

- If we set $c = 0$ (or directly remove this input from the circuit), the license will be able to be used only once.
- If the SP requests the user to set a custom value for c (e.g. the date of an event), the license will be able to be reused only under certain conditions.