

Citadel Protocol Specification

Dusk Network

March 23, 2023

Contents

1	General Overview	2
1.1	What is Citadel	2
1.2	Document Organization	2
2	Definitions	2
2.1	The Roles involved	2
2.2	The Elements involved	2
3	Protocol Workflow	3

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- **Proof of Validity:** users can prove ownership of a valid license, that has not been revoked.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:** TBD.
- **License:** A license is an asset that represents the right of a user to use a given service. The structure is as follows:

$$\text{lic} = \{\text{type}, \text{pos}, \text{nonce}, \text{enc}, (\text{lpk}, R)\}.$$

where

Element	Type	Info.
type	-	Can be transparent (0) or obfuscated (1).
pos	-	It is the position of the license into a Merkle tree of licenses.
nonce	-	It is a randomness needed to compute <i>enc</i> .
enc	-	It is a ciphertext of size 4.
(lpk, R)	-	It is the public key of the license belonging to the user.

- **Session Cookie:** A session cookie is a secret value (thus, always obfuscated) 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:

$$\text{sc} = \{\text{pos}, \text{nonce}, \text{enc}, (\text{lpk}_{\text{SP}}, R_{\text{SP}})\}.$$

where

Element	Type	Info.
pos	-	It is the position of the license into a Merkle tree of session cookies.
nonce	-	It is a randomness needed to compute <i>enc</i> .
enc	-	It is a ciphertext of size 3.
(lpk _{sc} , R _{sc})	-	It is the license public key of the session cookie.

3 Protocol Workflow

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

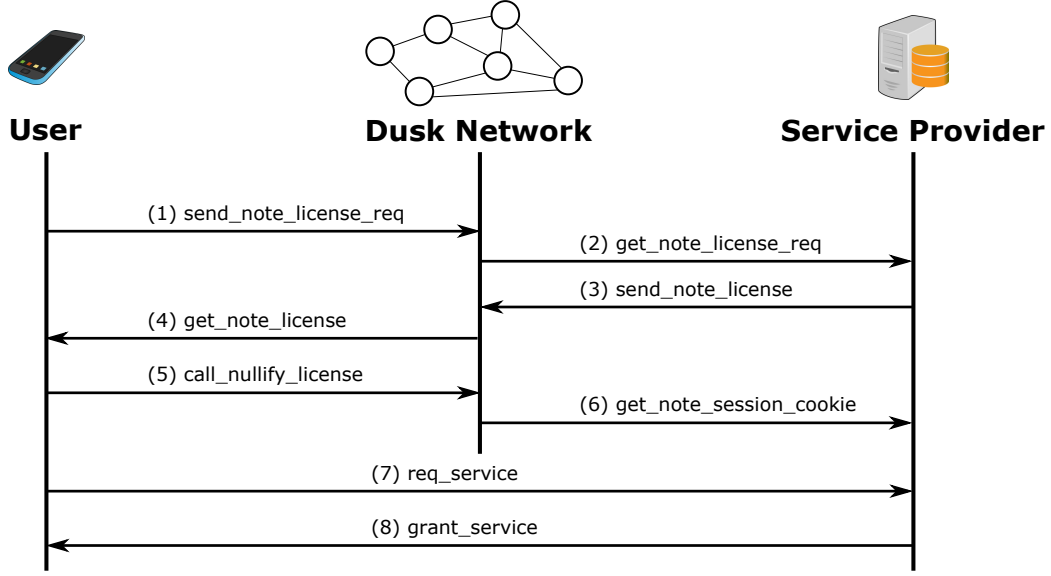


Figure 1: Overview of the protocol messages exchanged between the user, the Dusk Network, and the SP.

1. **(user)** send_license_req : TBD.
2. **(SP)** 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:

$$\text{sig}_{\text{lic}} = \text{sign_single_key}_{\text{sk}_{\text{SP}}}(\text{lpk}, \text{attr})$$

3. **(SP)** send_license : Compute $\text{enc} = \text{Enc}_{\text{k}_{\text{user}}}((\text{sig}_{\text{lic}}, \text{attr}); \text{nonce})$, set all the parameters of the license struct, and send it to the user.
4. **(user)** get_license : Receive the license.
5. **(user)** 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 $\text{enc} = \text{Enc}_{\text{k}}((s_0, s_1, s_2); \text{nonce})$ 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 *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 $\text{nullifier}_{\text{lic}}$ will be added to a shared list of nullifiers.

6. **(SP)** get_session_cookie : Receive the session cookie *sc*.
7. **(user)** req_service : Request the service to the SP, establishing communication using a secure channel, and providing the tuple $(\text{tx_hash}, \text{pk}_{\text{SP}}, \text{attr}, c, \text{sc})$.
8. **(SP)** grant_service : Grant or deny the service upon verification of the following steps:
 - Check whether or not the values $(\text{attr}, \text{pk}_{\text{SP}}, c)$ are correct.
 - Check whether or not the openings $((\text{pk}_{\text{SP}}, s_0), (\text{attr}, s_1), (c, s_2))$ match the commitments $\text{com}_0^{\text{hash}}, \text{com}_1, \text{com}_2$ found in the transaction *tx_hash*.

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 $\text{nullifier}_{\text{lic}}$. The deployment of this part of the circuit has two different possibilities:

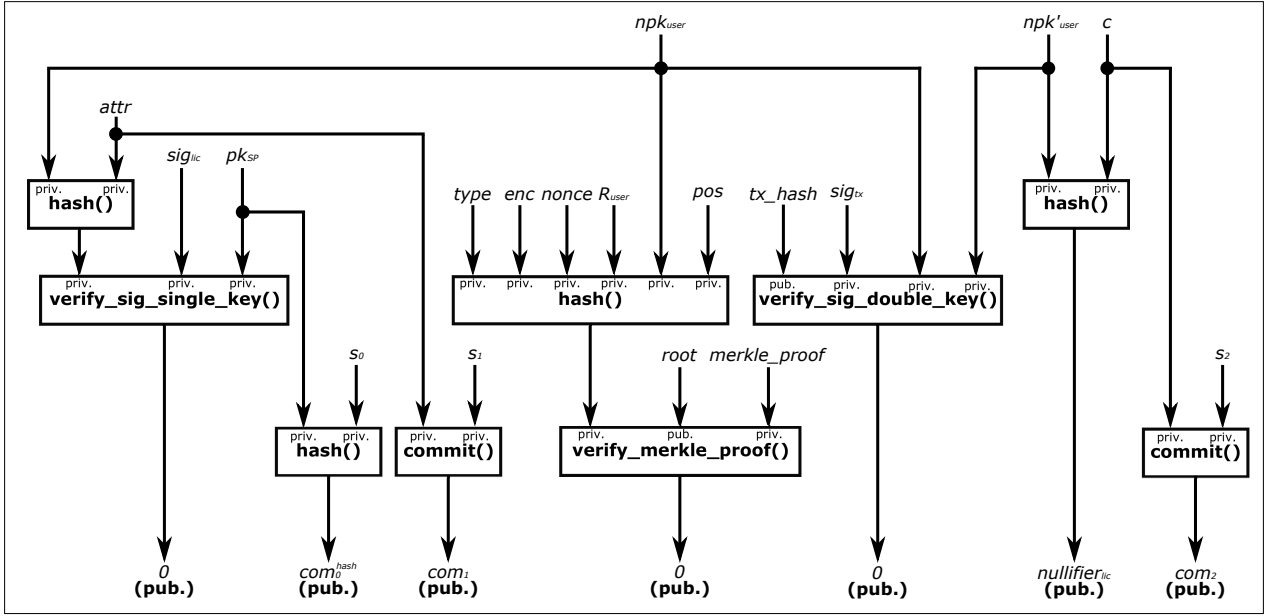


Figure 2: Arithmetic circuit for proving a license's ownership.

- 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.