Cardano authentication system based on data signature

Cardano Solutions.

2022/07/02

Abstract

Cardano is an open-source blockchain, based on the UTXO model, which internal cryptocurrency is ADA. Moreover, Cardano is a smart contract platform and therefore serves as a backend for all kind of decentralized applications. Users of these interact directly with the blockchain by signing and submitting or observing transactions, which acts as inputs and outputs for smart-contracts (self-executing scripts). Though, some applications necessitate to either act as an intermediate between the blockchain and the user, or to exchange information with him/her. An example of this could be a Metaverse where user position must be be stored, update and read as the game revolves on a central server. Such applications, qualified as semi-decentralized, require the ability to authenticate a user with proof of wallet ownership. This article aims to propose an authentication protocol relying on Cardano message signing standard: CIP-8 [1].

1 Introduction

1.1 Cardano wallets

A Cardano wallet is a set of private and public keys generated using Ed25519 cryptographic elliptic curve from a seed (in practice, mnemonic words) [2], as described in BIP32-Ed25519 [3]. Using differrents paths, 2^{32} addresses can then be derived from those root keys following CIP-1852 [5], see Figure 1.

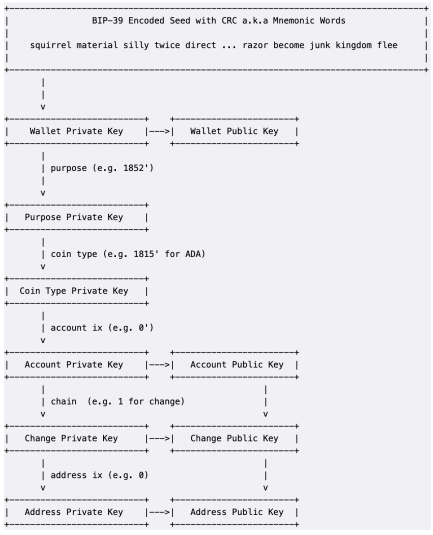

Figure 1: Wallets Key Hierarchy, with example derivation path 1852/1815/0/1/0 (source: [4]).

1.2 Message signing

Message signing on Cardano works exactly the same way as transaction signature and is described in CIP-8 [1]. Message is first serialized into Concise Binary Object Representation (CBOR) [6], a compact standard for representation of data structure, which can then be signed by any address using

CBOR Object Signing and Encryption (COSE) [7], exactly the same way a JSON can be signed using JSON Object Signing and Encryption (JOSE) [8]. Figure 2 shows the structure of a CBOR signed object.

```
[
      bstr,               ; protected header
      { * label => any }, ; unprotected header
      bstr / nil,         ; message to sign
      [                   ; signature array
        [                     ; first signature
          bstr                ; protected
          { * label => any }, ; unprotected
          bstr                ; signature
        ]
      ]
    ]
```

Figure 2: Basic message signature structure using COSE standard (source: [1]).

A standard way of setting CBOR claims that need to be shared by two parties and validated (like expiration time and issuer) on a message which will be signed using COSE is CBOR Web Token (CWT) (equivalent of JSON web token (CWT) for CBOR) [9]. A CWT can either be encrypted, MACed or signed. For authentication puposes the signed case is the one of interest here. A signed CWT is composed of: COSE header (prepended by COSE CBOR tag), COSE signed data payload (with both header claims and user data).

```
(
[
  / protected / << {
    / alg / 1: -7 / ECDSA 256 /
  } >>,
  / unprotected / {
    / kid / 4: h'4173796d6d6574726963454344534413
                23536' / 'AsymmetricECDSA256' /
  },
  / payload / << {
    / iss / 1: "coap://as.example.com",
    / sub / 2: "erikw",
    / aud / 3: "coap://light.example.com",
    / exp / 4: 1444064944,
    / nbf / 5: 1443944944,
    / iat / 6: 1443944944,
    / cti / 7: h'0b71'
  } >>,
  / signature / h'5427c1ff28d23fbad1f29c4c7c6a555e601d6fa29f
                  9179bc3d7438bacaca5acd08c8d4d4f96131680c42
                  9a01f85951ecee743a52b9b63632c57209120e1c9e
                  30'
]
)
```

Figure 3: Example of a signed CWT (source: [9]).

2 Authentication methods

2.1 Usual authentication protocol

Cardano based application usually implement naive ways of authenticating their users. Most just ask for enabling read access to the extension wallet public key like jpg.store, mostly because they are fully decentralized application. There is therefore no need to send authenticated requests to a centralized server.

Otherwise, some application ask for the user to sign a trivial message with private key, like cnft.io, for instance: "Log in to cnft.io". A malicious individual could social engineer someone into signing such a message and logging into his account at any time, as there is no expiration of the signed message, nor validation that it was originally emitted by the server.

Another possible authentication could rely on CWT directly containing all the user data and directly be used to authenticate the user. But this method would not allow for the server to revoke user sessions, and user data would have to be constant and light.

2.2 Proposed authentication protocol

To solve those issues, we propose an authentication protocol based on a handshake using a CWT. User is asked to sign a specific CWT issued by the server with an expiration date. Protocol is pre-

sented in Figure 4.

Connexion request: First the user declares his intention to authenticate to the server by fetching the data to sign (steps 1, 2).

Handshake token generation: Upon receiving request, server generates a random handshake token string, with an expiration timestamp 5 minutes in the future and stores both those informations a database (step 3) along with user's public address. He can then forge a message containing those informations and sends its CBOR back to the user (step 4, 5).

Message signing: User deserializes said message from CBOR and signs it with his private key (step 6). He then serialize signed message and sends its CBOR back to the server (step 7, 8).

Message verification: Serverside, signed message integrity must be checked after CBOR deserialization. With user address, handshake token can be retrieved from database (setp 9, 10). Then occures a check wether handshake token in message matches the stored handshake token, and if token in database is not expired (step 11). Also it is checked that the address of signer is the same as the one sent by user. Handshake token can finally be deleted from database (step 13).

User authentication: If all conditions above are met, a session token is generated, stored along with user address in a database (step 12), and returned to the user (step 14). It is then stored in a cookie on the user side, and can be used to authenticate the user for any kind of requests.

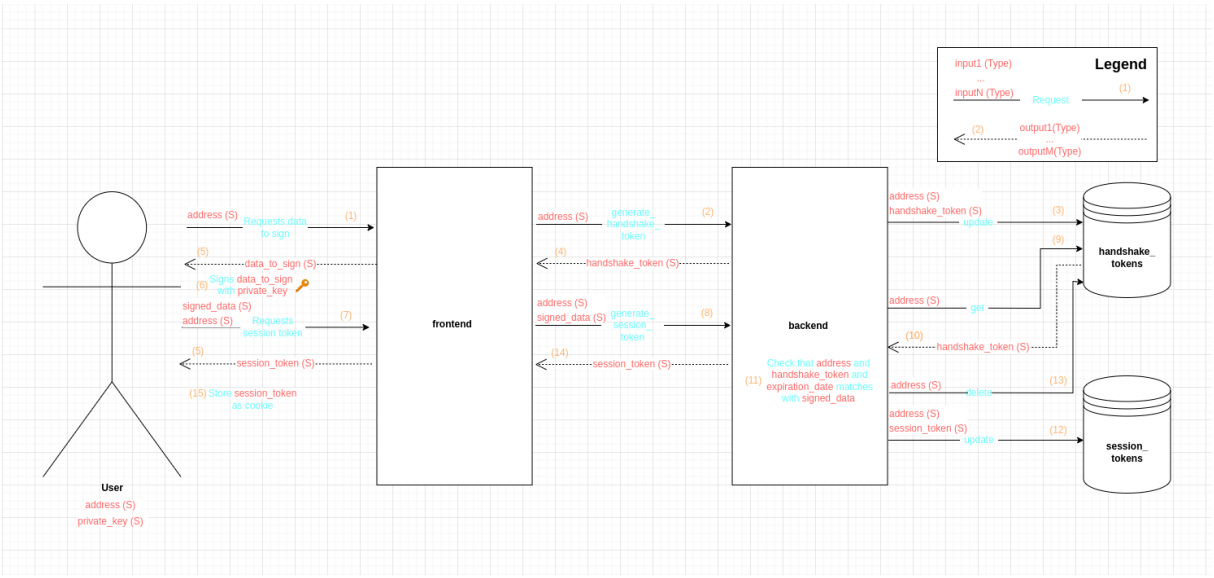

Figure 4: Data signature based authentication handshake protocol.

3 Conclusion

Proposed protocol allows authentication of any user to a semi-decentralized application. Though it is quite demanding on resources both from authentication server and database, as session token is requested to "session_tokens" table every time user authentication is needed.

For use cases where user data payload is not too heavy and there is no need for revoking user sessions, using just a CWT should be preferred as it lightens the infrastructure. But for a usecase like a metaverse, where access needs to be regulated at any time, in a secured way, and players' data is dense and consistently changing, this protocol is perfectly suited, as a CWT would imply the user signing data with his private key every time he changes position for instance.

References

[1] Matthias Benkort, Duncan Coutts, Sebastien Guillemot, Sebastien Guillemot, Frederic Johnson, Robert Phair
CIP 8: Message Signing,
https://cips.cardano.org/cips/cip8/

[2] Adrestia team, IOHK
Cardano Wallet documentation - HD Wallets (2020-10-21),
https://input-output-hk.github.io/adrestia/cardano-wallet/concepts/hierarchical-deterministic-wallets

[3] Dmitry Khovratovich: University of Luxembourg, Jason Law: Evernym, Inc.
Hierarchical Deterministic Keys over a Non-linear Keyspace,
https://input-output-hk.github.io/adrestia/static/Ed25519_BIP.pdf

[4] Adrestia team, IOHK
Cardano Wallet documentation - Address Derivation (2020-10-21),
https://input-output-hk.github.io/adrestia/cardano-wallet/concepts/address-derivation

[5] Matthias Benkort, Duncan Coutts, Sebastien Guillemot, Sebastien Guillemot, Frederic Johnson, Robert Phair
Cardano Improvement Proposal,
https://cips.cardano.org/

[6] C. Bormann, P. Hoffman
Concise Binary Object Representation (CBOR) [RFC 7049*]* (2021-10),
https://datatracker.ietf.org/doc/html/rfc7049

[7] J. Schaad
CBOR Object Signing and Encryption (COSE) [RFC 8152*]* (2017-07),
https://datatracker.ietf.org/doc/html/rfc8152

[8] M. Miller
JSON Object Signing and Encryption (JOSE) [RFC 7520*]* (2015-05),
https://datatracker.ietf.org/doc/html/rfc8152

[9] M. Jones, E. Wahlstroem, S. Erdtman, H. Tschofenig
CBOR Web Token (CWT) [RFC 8392*]* (2020-10-21),
https://datatracker.ietf.org/doc/html/rfc8392