CSE 598 – Engineering Blockchain Applications – Project 2 Part 2

Title: **Dash Wallet Mnemonic Transaction**

Student Name: Abhishek Patil

ASU ID: 1216126522

Abstract:

This report introduces the concept of Wallets and Mnemonic codes and is primarily focused on describing an example code to use mnemonics. It lists few important terms relating to this project and then provides the goal of the project. After that, it describes the steps taken to reach the solution and explains the code for the same. Later, it discusses issues faced, cites some of the related works and ends with a conclusion briefing on the lessons learned.

Keywords:

Dash, Blockchain, Wallet, Mnemonic code

Introduction:

We are all familiar or have heard about blockchain technologies and their use in cryptocurrency transactions. Whenever a transaction happens there are interactions with the blockchain network. Such interactions require public and private keys to make it successful.

It is not possible for a person to remember multiple of these keys and thus an application called wallet was created. It stores and manages keys in a secure way so that they can be used to sign messages or transactions. Additionally, as described in BIP-0039 [1], mnemonics were created to have some human readable, writable, or speak-able representation of wallet seeds that create the keys.

In project 2 part 2, we see how to create a simple transaction in Dash blockchain network that will only need the sender's mnemonic and the receiver's address to send to.

Terminology:

Wallet: an application that stores and manages keys which are needed to interact with a blockchain. Wallet helps to keep these keys secure and easy access without any human error.

Mnemonic codes: a mnemonic is a learning technique that aids information retention or retrieval for humans [2]. Mnemonic codes are human representation for the wallet seeds.

Identities: these are lower-level constructs that make the foundation for interactions with Dash Platform. In simple words, they are public keys recorded on platform chain to sign documents [3].

Dash Platform Name Service (DPNS): this is a service used to register names for identities on Dash Platform. This enables to easily establish trust when users interact with other users and applications.

Goal Description:

The goal of this project is to use the Dash library to send funds of 1 Dash from a given sender's mnemonic to a given receiver's address.

Given sender mnemonic: 'tip rabbit soon wage judge weekend thunder employ reject dutch today artist'

Given receiver address: 'yP8A3cbdxRtLRduy5mXDsBnJtMzHWs6ZXr'

Description of proposed solution:
At first, I followed the tutorial for building with the Dash Platform at https://dashplatform.readme.io/docs/tutorials-introduction. Some parts of the tutorial do not

form the actual solution. Here, I am describing the tutorial steps I took before describing the solution because it sets up the foundation which is crucial. The steps are as follows:

1. Connect to Dash test network - I acquired the Dash library in javascript and then created a client using *Dash.Client()*. Next, I used *client.getDAPIClient().core.getBestBlockHas()* to connect and get the next block hash in the blockchain.

2. Create a Wallet – I created the client by passing some client options. The client options indicate *'testnet' network*, and a *wallet* with *mnemonic* field set to *NULL* and *offlineMode* field set to *true*. On the *client* object, *getWalletAccount()* is called to get a new wallet. It gives new one as mnemonic was set to *NULL*. The corresponding new mnemonic is obtained using *client.wallet.exportWallet()* and unused address is returned by *getUnusedAddress()* method of the wallet *account*. My details on the new wallet were:

Mnemonic: inmate attack vapor term tray bitter forward weekend grief winner tray voyage
Unused address: yXk58AjoQC48wxdvhDqfwPeU97fuJBTSU4

3. Fund a Wallet – I funded the wallet by transferring some random amount of Dash using this website http://faucet.testnet.networks.dash.org/.

4. Register an Identity – By passing the newly created mnemonic in client options, I created a new client and registered new identity using *platform.identities.register()* on it.

5. Register a Name – I also created a name for my identity. First, I got the *identity* for the client with the mnemonic shown above using *platform.identities.get()*. Then, I used *platform.names.register()* which took three arguments: '*<name>.dash*' (without the angular brackets), a dictionary with *dashUniqueIdentityId* set to a value received from *identity.getId()*, and the *identity* returned from *get()* function.

After doing these basic steps, I felt I had enough foundation for building up the solution of this project. The 'Send Funds' section of the tutorial provides a good example code for transferring funds from a given mnemonic code to an address.

As learned previously, I acquired the Dash library and set the *clientOpts* to have *'testnet'* network, and *wallet* with *mnemonic* provided to us as 'tip rabbit soon wage judge weekend thunder employ reject dutch today artist'. Also, because of the simplicity of the project we can set to skip the synchronization starting from start of this year, which would be at 415000 block height.

Once the client is created on the *clientOpts* using *Dash.Client()*, we get the wallet *account* for it using *getWalletAccount()* method of the *client*. This *account* is accessed in the *Promise.then()* function. The *account* has the *createTransaction()* method which takes a dictionary containing two fields: *'recipient'* set to the given value in project instructions

'yP8A3cbdxRtLRduy5mXDsBnJtMzHWs6ZXr' and *'satoshis'* set to 100000000 (which equals 1 Dash). The created transaction is returned into the *trxn* variable.

Finally, the *account* variable's *broadcastTransaction()* method is used to broadcast the specific transaction in the *trxn* variable. This method returns with a response stored in the *brdcst_rspns* variable (inside *Promise.then()* function). The *brdcst_rspns* is printed in the console. The order followed in the code uses the nested *Promise.then()* due to the asynchronous flow.

Any errors associated to both getting the wallet and broadcasting the transaction are separately and clearly printed as *'Getting Wallet Error:'* and *'Broadcast Transaction Error:'* respectively. Also, using the *finally* block ensures the client is disconnected whether an error occurred or not.

Additionally, any other asynchronous errors associated with the client's wallet are handled at the end. For this, *client.on()* method is used that prints both the *error* and its *context*.

After running my code, I received the transaction ID which was broadcast and a screenshot of it on Dashevo [4] is shown in figure 1.

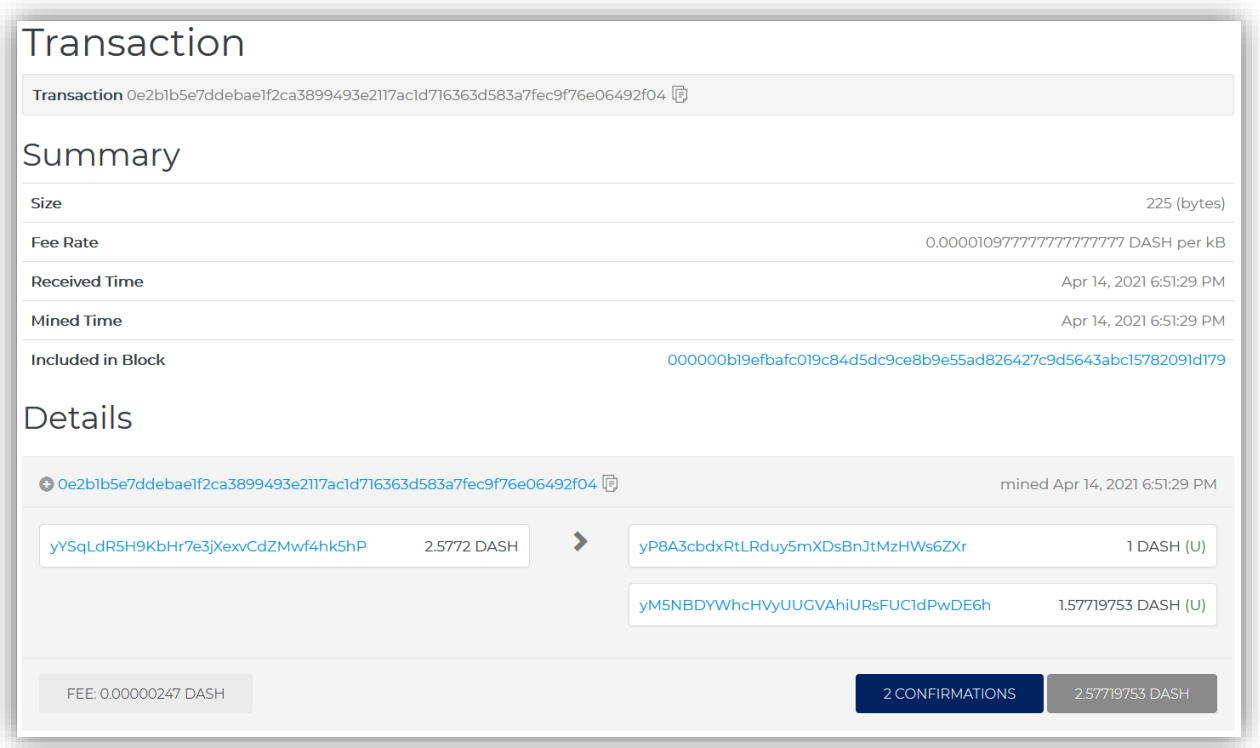

Figure 1. My Dash Wallet Mnemonic transaction:
'0e2b1b5e7ddebae1f2ca3899493e2117ac1d716363d583a7fec9f76e06492f04' on Dashevo

Issues Faced and Methods used to resolve them:

This project was mostly straightforward because of a very good tutorial for it. However, I did have some issues/concerns that I would like to discuss. Firstly, the changes of different Node versions throughout this course is time consuming. Getting Node to work properly can be tough and thus, I just uninstall it entirely and reinstall required version. This does take up a lot of time and can get frustrating. For this part of project, Node version 12+ was required to run.

Secondly, learning and understanding new things is naturally tough and can cause issues on the way. In this project too, I had some difficulties in making sense of the code in the tutorial. The tutorial is well explained but due to low experience in javascript it was a little rough. I spent some time trying to understand how the client and a specific wallet were linked to each other through the *clientOps* parameter. Also, the asynchronous function calls made the flow slightly confusing at first. Over the duration of the tutorial, I was able to build up my foundations by executing the example code multiple times and reviewing the outputs.

Related Works (if any):

[1]. Bitcoin, "bitcoin/bips," GitHub, 12-Feb-2021. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki. [Accessed: 11-Apr-2021].

[2]. "Mnemonic," Wikipedia, 08-Mar-2021. [Online]. Available: https://en.wikipedia.org/wiki/Mnemonic. [Accessed: 11-Apr-2021].

[3]. "Identity," Dash Platform. [Online]. Available: https://dashplatform.readme.io/docs/explanation-identity. [Accessed: 10-Apr-2021].

[4]. "Dash Insight," Home | Insight. [Online]. Available: https://testnet-insight.dashevo.org/insight/. [Accessed: 10-Apr-2021].

Conclusions:

In conclusion, this project has provided an insight on transactions that use wallets and mnemonic codes. Having a human representation of wallet seeds and being able to understand how it is coded is crucial in the blockchain industry. Additionally, it has enhanced my general javascript skills especially in assessing asynchronous code and debugging them properly. All these tools are very important for my overall development as a computer science student.

Bibliography: none