Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017] 10/09/2021

01) Mobile Wallet

I have chosen the Dash Wallet which uses an altcoin called Dash. Version 7.02 is the first stable release. The mainnet version is there on F-Droid and Google Playstore but the testnet version has to be side loaded from the developer's GitHub [repo link](#) .

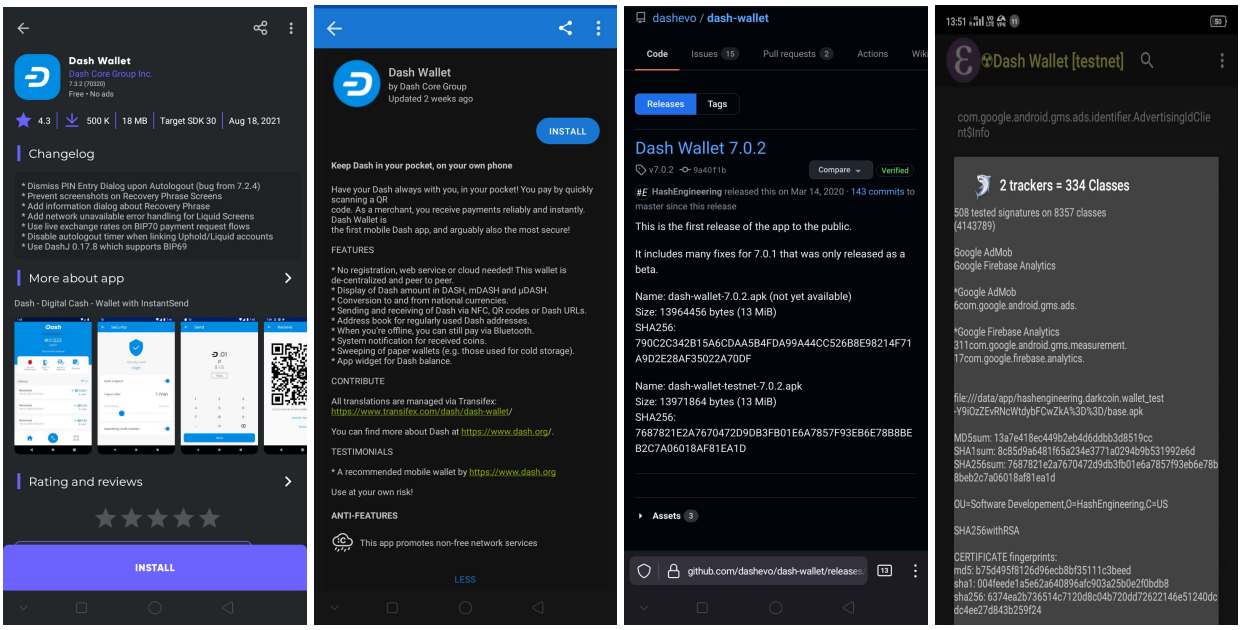


Figure 1 Dash Wallet as on F-Droid, Google Play Store, GitHub repo and Dash Wallet[testnet] app details

The home screen of the Dash Wallet [testnet] allows us to either create a new wallet or restore existing using recovery phrase.

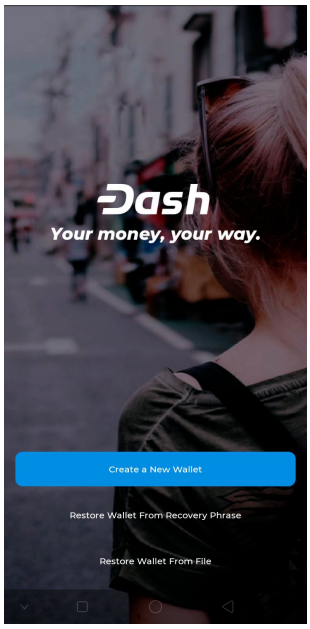


Figure 2 Greeter page of Dash wallet

Creating a New Wallet will require the user to set a 4 digit PIN(automatically locks the wallet after 8 incorrect tries, after which we have to restore using the mnemonic phrase) for protection and an optional mnemonic phrase for wallet recovery.

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017] 10/09/2021

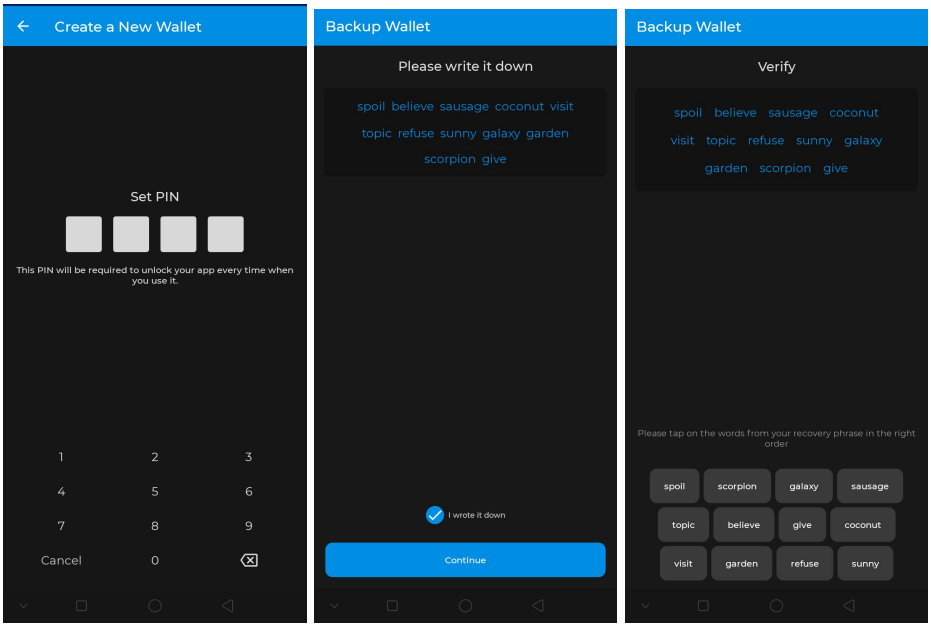


Figure 3 Setting up PIN, creating and verifying mnemonic phrase

Before doing transactions, the wallet needs to be synced.Interestingly the oldest block available with the application is only 2 hours old, as during the syncing process; it downloads the ledger upto a certain time, verifies it is correct and then deletes it. The process is repeated until the transactions are only a few hours old.The Network monitor used for viewing the blocks and the connected nodes is in Tools —> Network monitor

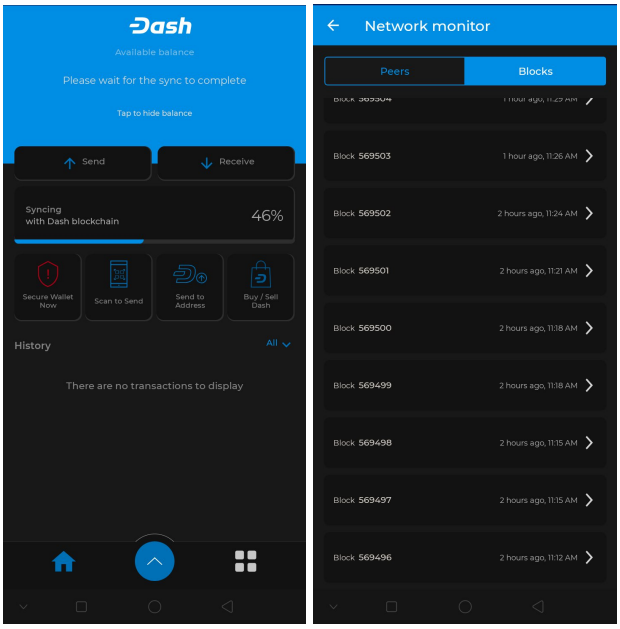


Figure 4 Syncing up the wallet

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017] 10/09/2021

I requested some test Dash coins (tDash) from a faucet and verified the same with a explorer.

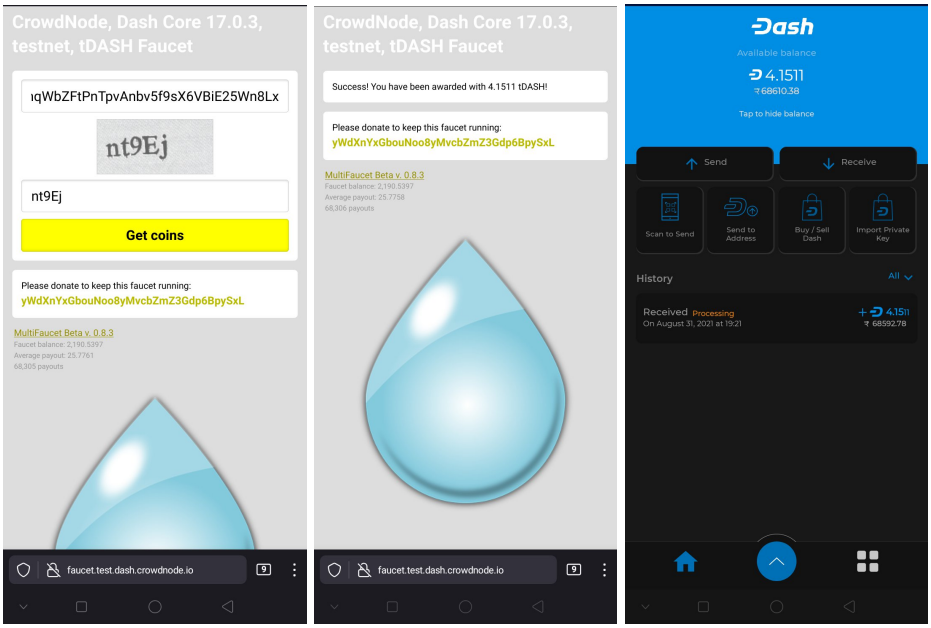


Figure 5 Requesting test coins from faucet

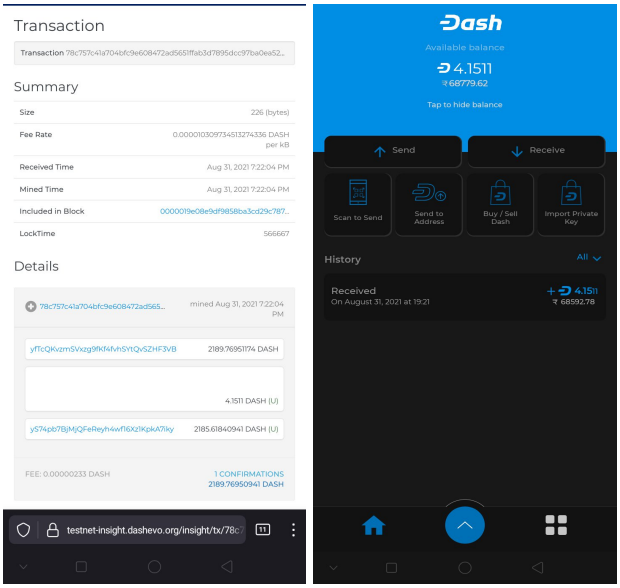


Figure 6 Coins received from faucet and verified it on explorer

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017] 10/09/2021

02) Web Wallet

I have chosen the Brave “crypto wallet” which is an integrated component of Brave Web Browser.After downloading the browser in the address bar type “brave://wallet”, this allows to create/restore wallets or connect with hardware based wallets.More information can be found here.

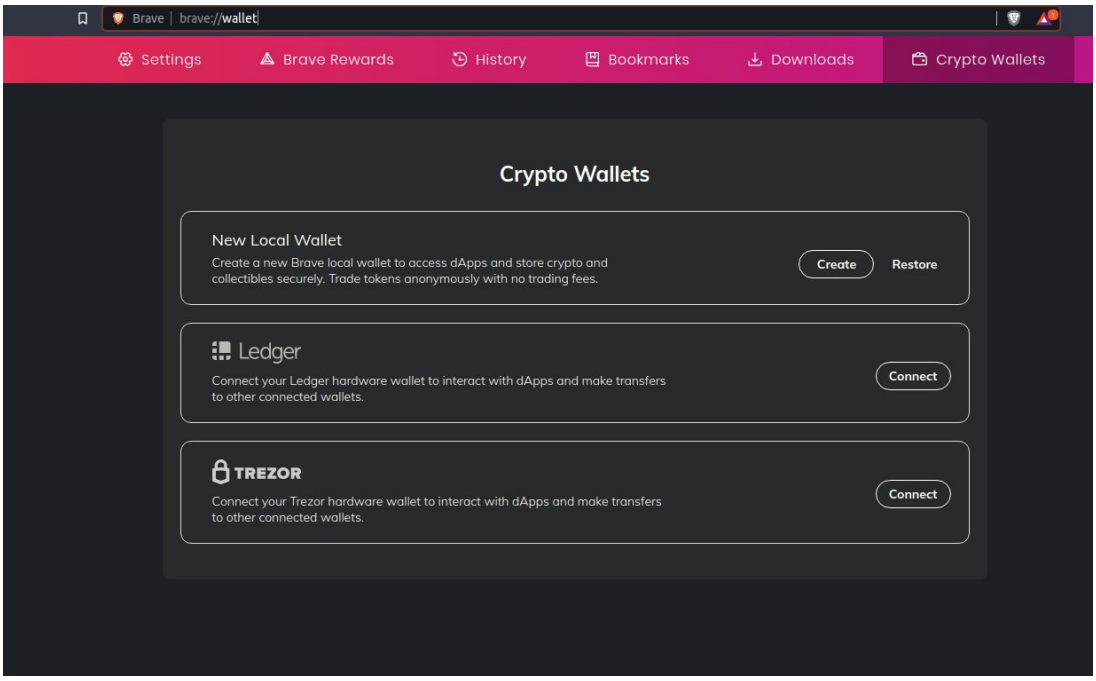


Figure 7 Brave wallet greeter page

Hitting on “Create” will allow use to create a new local wallet.Using this you can trade BAT (Basic Attention Token of Brave browser) and other Ethereum assets and collectibles. You can also connect to decentralized applications and interact with smart tokens.

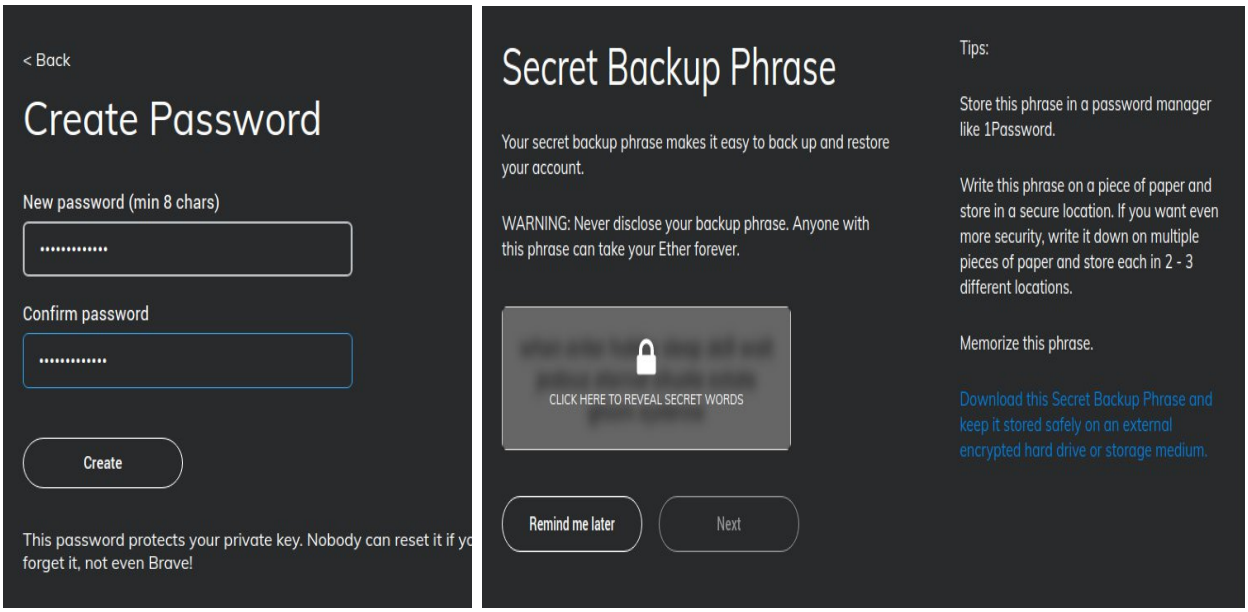


Figure 8 Setting up wallet

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017] 10/09/2021

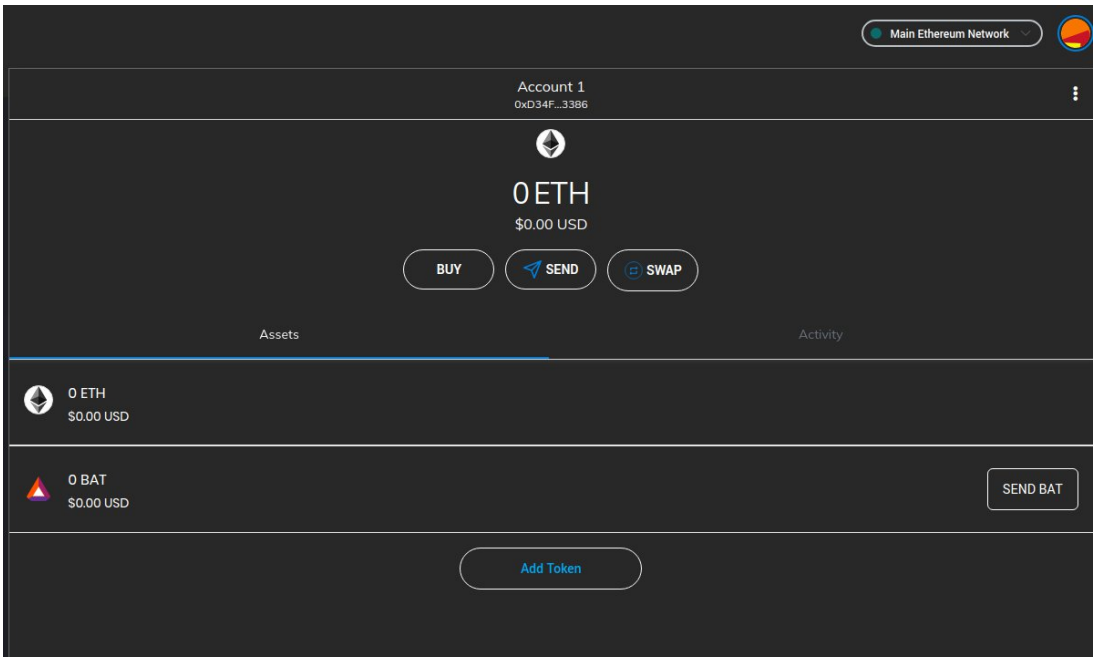


Figure 9 The Brave Crypto Wallet

Let us get some testnet currency for experimenting with this wallet.First create a node that runs the Binance Smart Chain (BSC) Network. We can use a “speedy node” provided by moralis.io

Select BSC network and copy the testnet link.

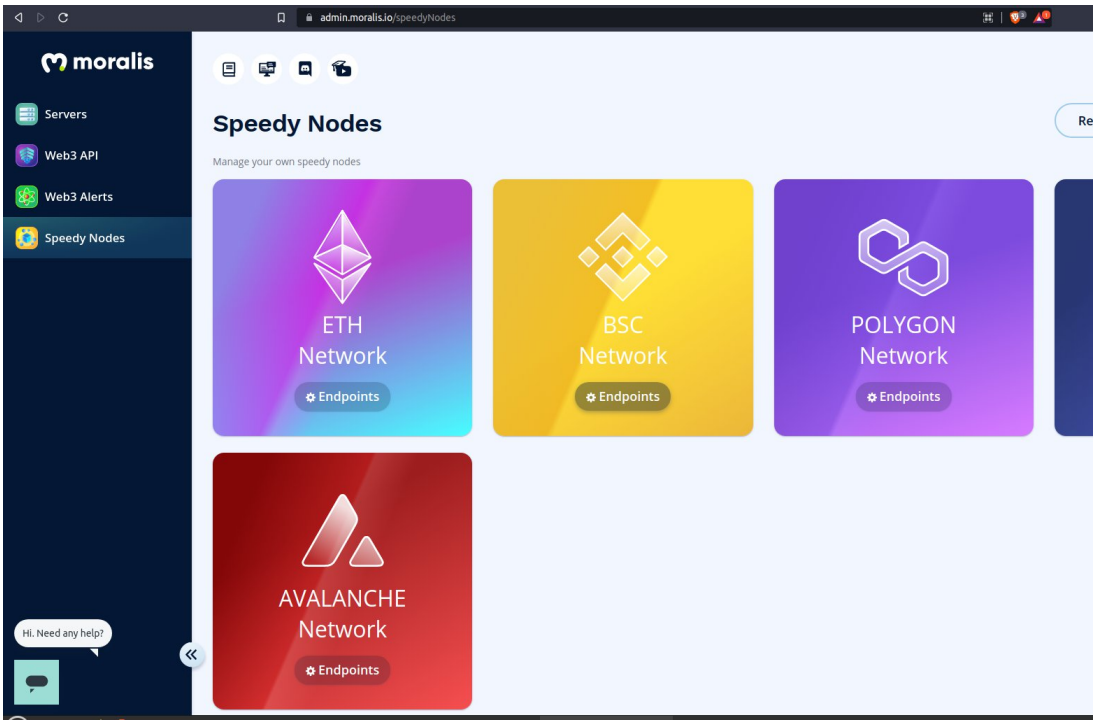


Figure 10 moralis.io node offerings

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017] 10/09/2021

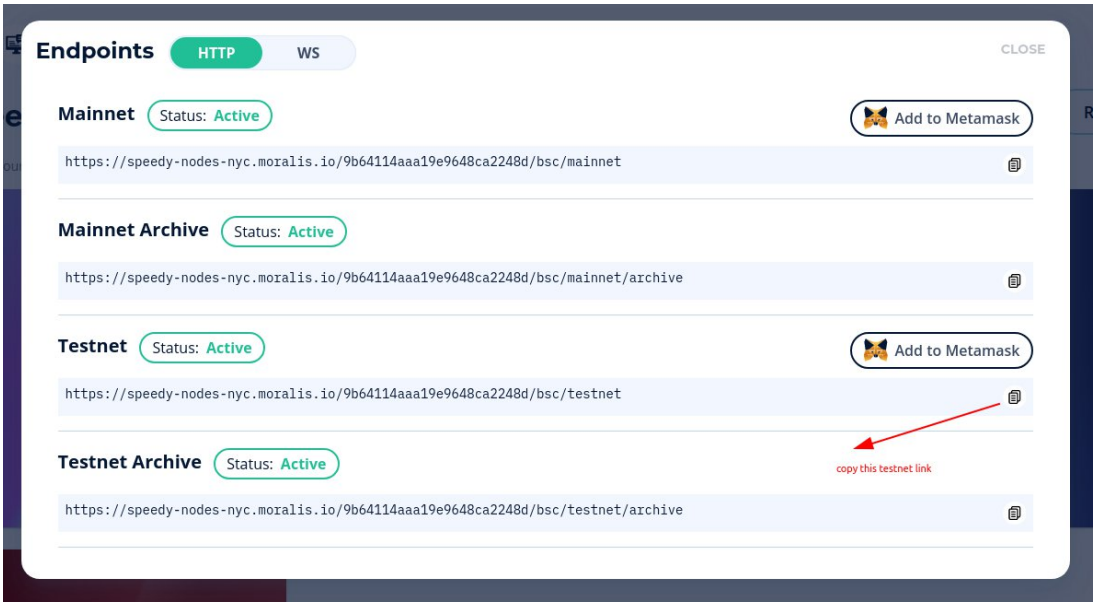


Figure 11 All endpoints of BSC chain available on moralis.io

In Settings —> Networks of Brave Crypto wallet, add the custom RPC with the RPC url being what we copied earlier.

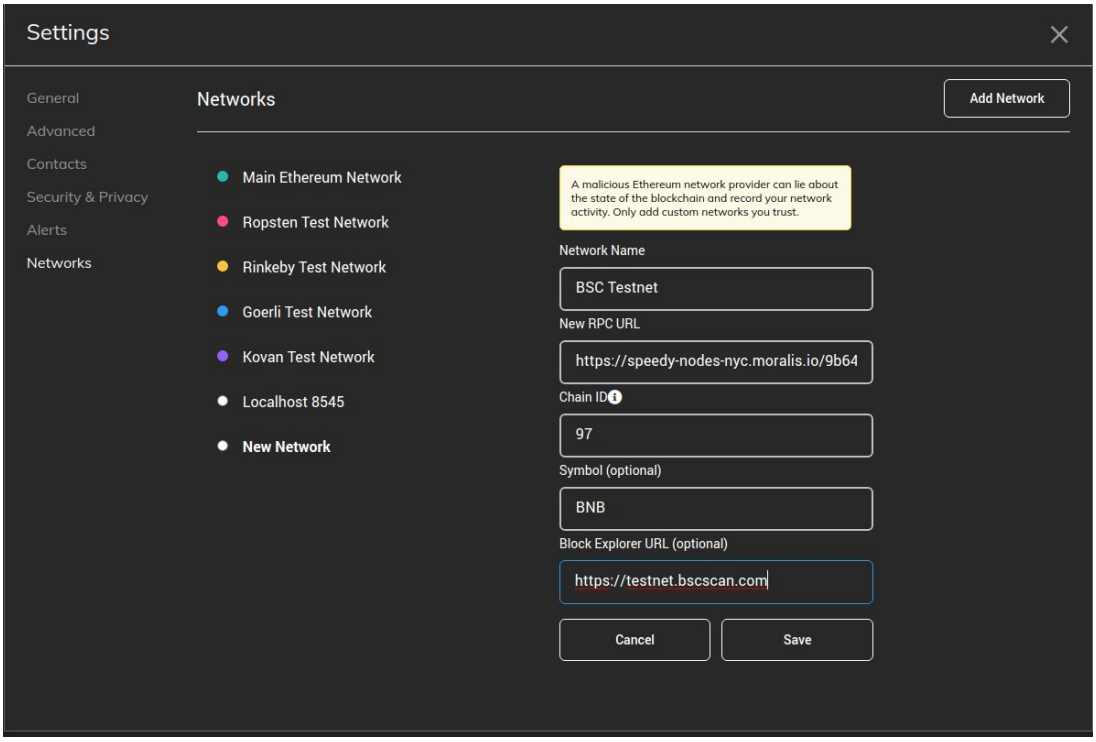


Figure 12 Adding custom RPC

We will be sending some test BNB from my Binance Chain Wallet to Brave Wallet.

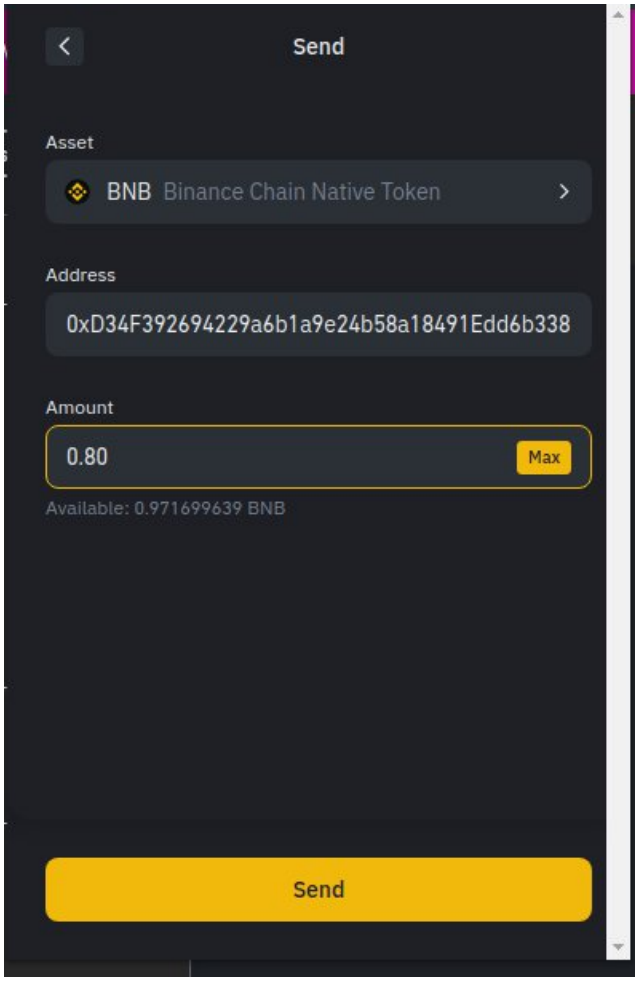


Figure 13 Sending transaction from Binance Chain wallet to Brave wallet

The transaction details are seen on the explorer here.

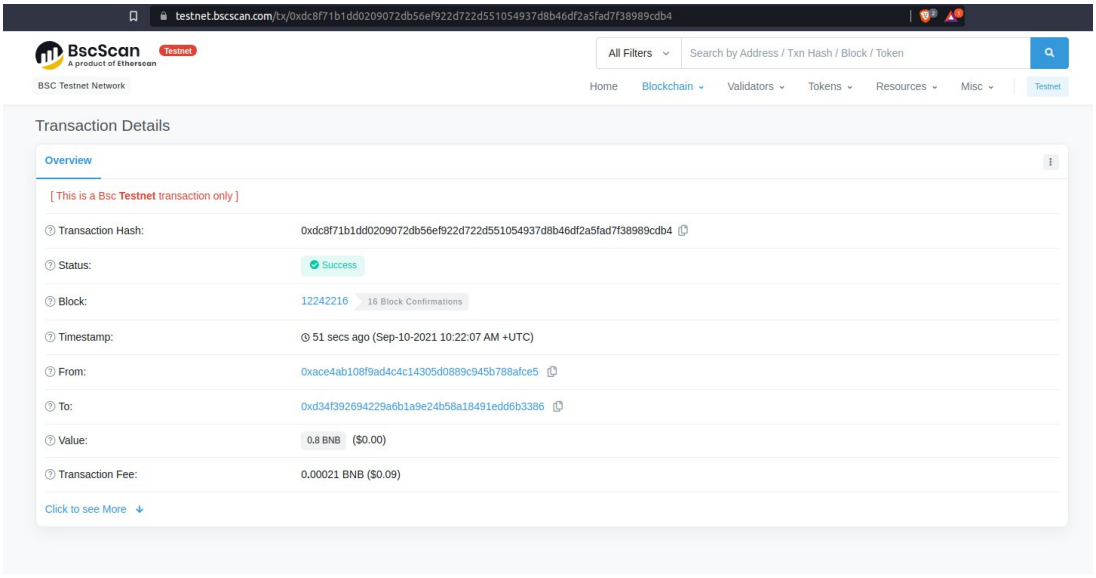


Figure 14 Transaction details

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017] 10/09/2021

Let us now create a token using smart contracts and deploy it on the BSC testnet.

For this we need a compatible IDE , we can use the remix IDE. We can use the smart contract template available here to make it easier for us to implement the token. The language used is solidity.

```
pragma solidity ^0.6.2;
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.3.0/contracts/token/ERC20/ERC20.sol";

contract Token is ERC20 {

    constructor () public ERC20("GokulCoin", "GCN"){
    _mint(msg.sender, 1000000 * (10  ** uint256(decimals())));
}
}
```

Figure 15 Code for deploying the non-native token

Create and compile(more information here about compiling) a new .sol file under contracts and write your smart contract. The above code basically imports a ERC20 contract template and creates a new token called GokulCoin, and gives who executes this 1 million GokulCoin.

When successfully compiled, you can deploy it to the BSC testnet, Brave Wallet automatically detects w3.0 apps like remix IDE so we can connect to the BSC testnet simply by hitting Next in the popup.

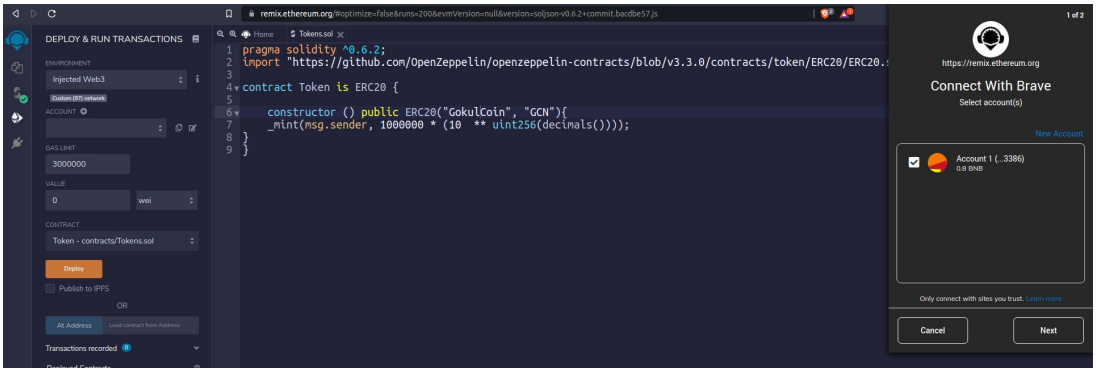


Figure 16 Deploying the token on BSC testnet

Now we can confirm the transaction in a similar manner.

Figure 17 Conforming the deployment

The details about this token can be found in the testnet explorer here.

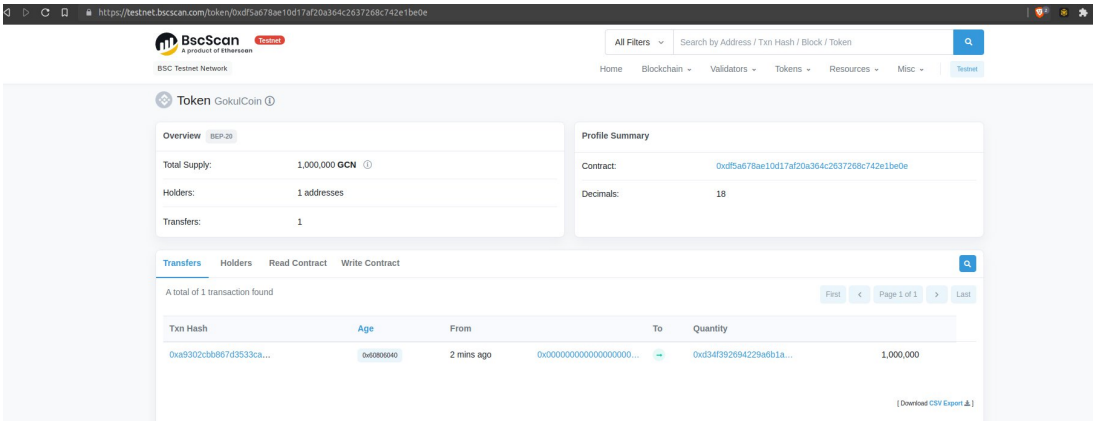


Figure 18 Details about GokulCoin

Copy the transaction ID (seen as Contract in the profile summary section above). Click on Add token section in the brave wallet (refer figure) and paste the ID in custom token and click Add Token

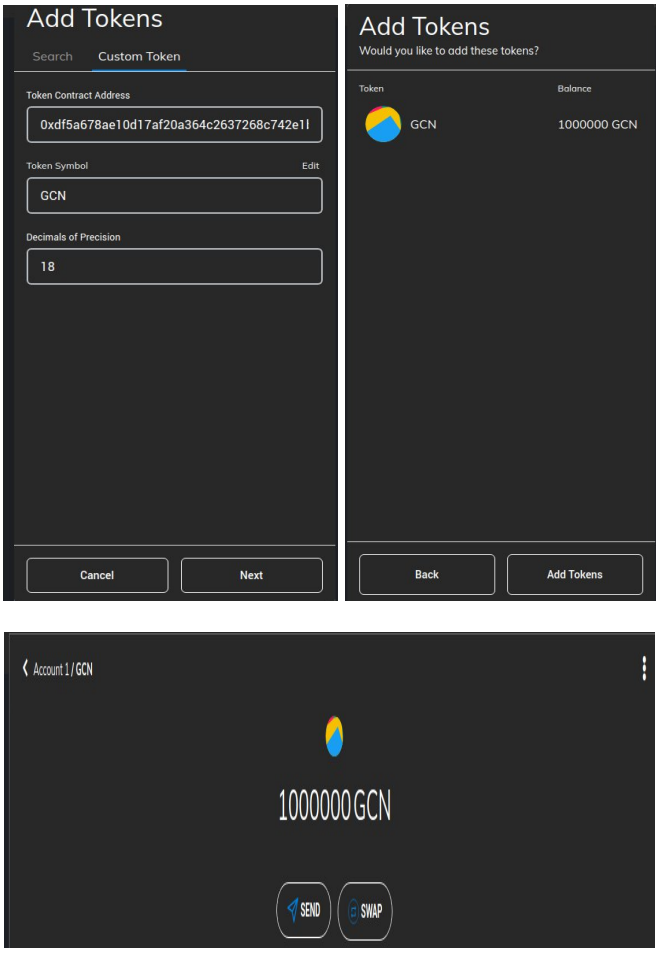


Figure 19 Adding GokulCoin on brave wallet

We can send this to any node on BSC testnet.

For example we can send back some GCN to the Binance wallet account which helped us with .8 test BNB.

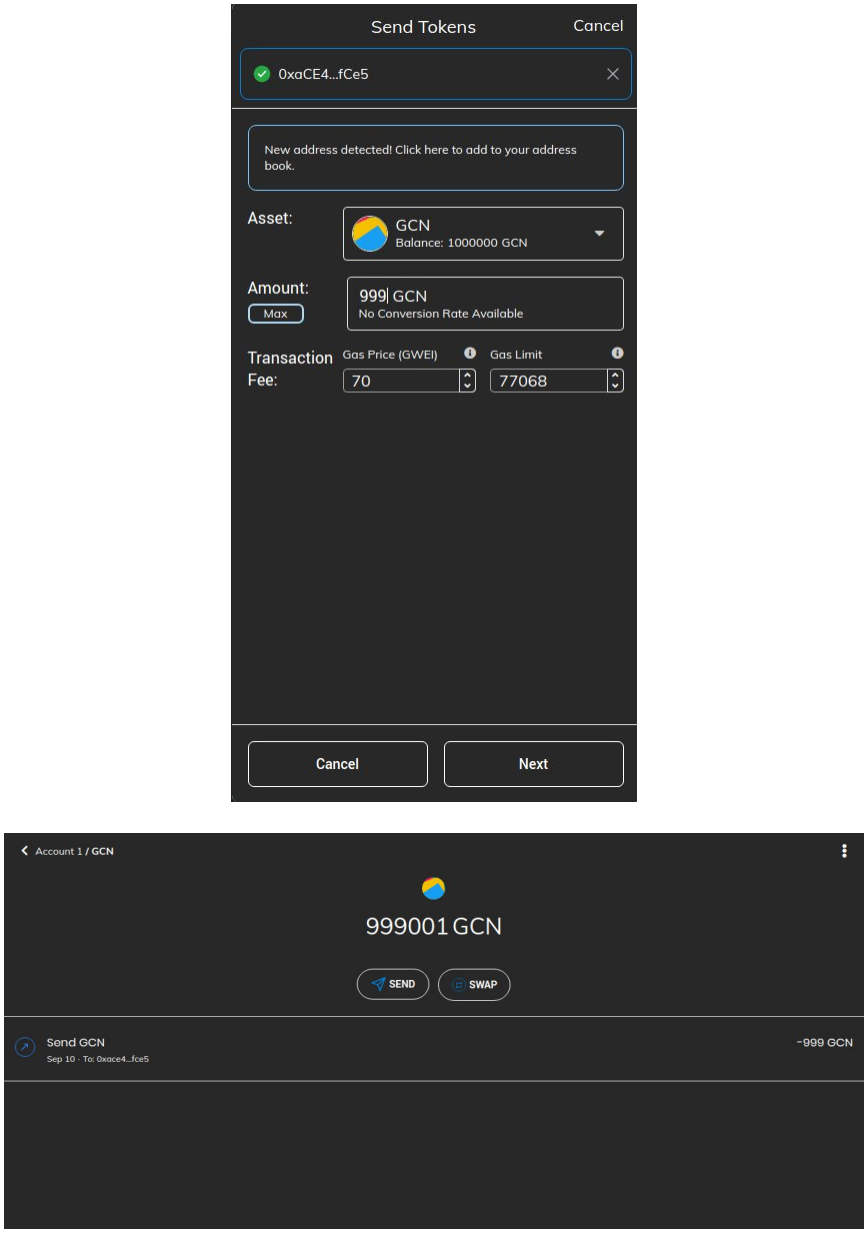


Figure 20 Sending GokulCoin to other wallets

The transaction status maybe viewed here.

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017] 10/09/2021

03) Desktop Wallet

I have chosen the Electrum wallet which is used for trading bitcoins. It aims to simplify the bitcoin transaction process by caching block information on its Electrum's servers. It connects to these high performance servers which does all the time consuming processes, and the final result is pushed to the wallet client.The client verifies the information send by servers using the "Simple Payment Verification", using which a simple node needs to download only the block headers and not the whole block chain to verify a transaction.

The application has many optional features, some of them being

- cold wallet
- integrating hardware wallets
  - Official documentation on the same is provided for the Debian based GNU/Linux.
  - The following wallets are currently supported
    - Trezor
    - Ledger
    - KeepKey
    - Digital Bitbox
    - BitBox02
- Variety of plugins, some of them being
  - Two factor authentication
    - Adds two factor authentication to the wallet
  - Email
    - Send and receive payment request keys with email account keys
  - Virtual Keyboard
    - Type your password in with integrated virtual keyboard
- The official full list is found here.

The developers have provided an Appimage which makes running the application on GNU/Linux very easy.

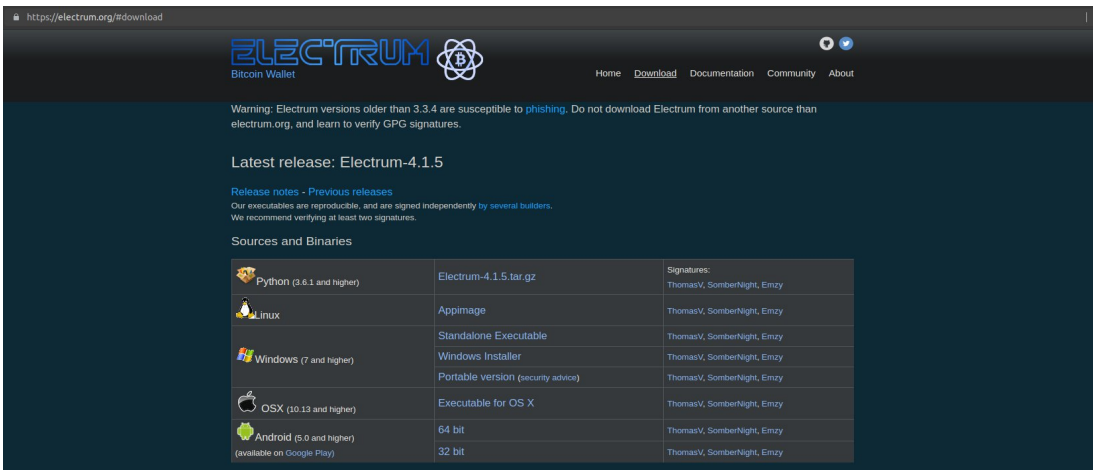


Figure 21 Download page of Electrum wallet

Once downloaded, We can either invoke the executable or supply the -h flag to display some additional options.

```
gokul@linuxbox:firefox-downloads$ file electrum.appimage
electrum.appimage: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so
.2, for GNU/Linux 2.6.18, stripped
gokul@linuxbox:firefox-downloads$ ./electrum.appimage -h
usage: electrum [-h] [-v VERBOSITY] [-V VERBOSITY_SHORTCUTS]
                [-D ELECTRUM_PATH] [-P] [--testnet] [--regtest] [--simnet]
                [--signet] [-o] [-w WALLET_PATH] [--forgetconfig]
                <command> ...

positional arguments:
  <command>
    gui                 Run GUI (default)
    daemon              Run Daemon
    add_lightning_request
    add_peer
    add_request         Create a payment request, using the first unused
                        address of the wallet
    addtransaction      Add a transaction to the wallet history
    broadcast           Broadcast a transaction to the network
    changegaplimit      Change the gap limit of the wallet
    clear_invoices      Remove all invoices
    clear_ln_blacklist
    clear_requests      Remove all payment requests
    close_channel
    close_wallet        Close wallet
    commands            List of commands
    convert_xkey        Convert xtype of a master key
    create              Create a new wallet
    createmultisig      Create multisig address
    createnewaddress    Create a new receiving address, beyond the gap limit
                        of the wallet
    decode_invoice
    decrypt             Decrypt a message encrypted with a public key
    deserialize         Deserialize a serialized transaction
    dumpgraph
    dumpprivkeys        Deprecated
    enable_htlc_settle
    encrypt             Encrypt a message with a public key
    export_channel_backup
```

Figure 22 Appimage and its additional options

We are only interested in working on testnet which can be accessed by specifying the flag --testnet.

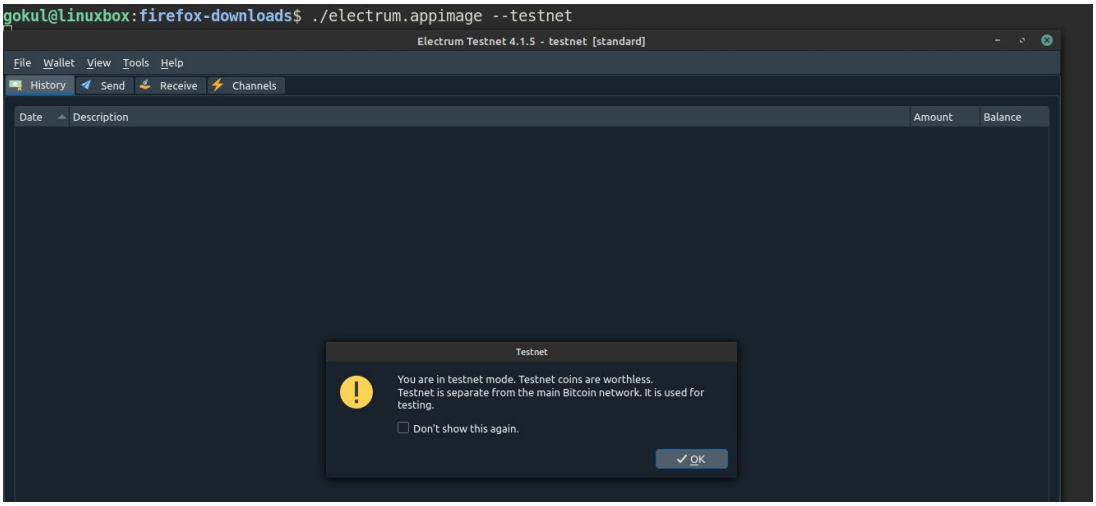
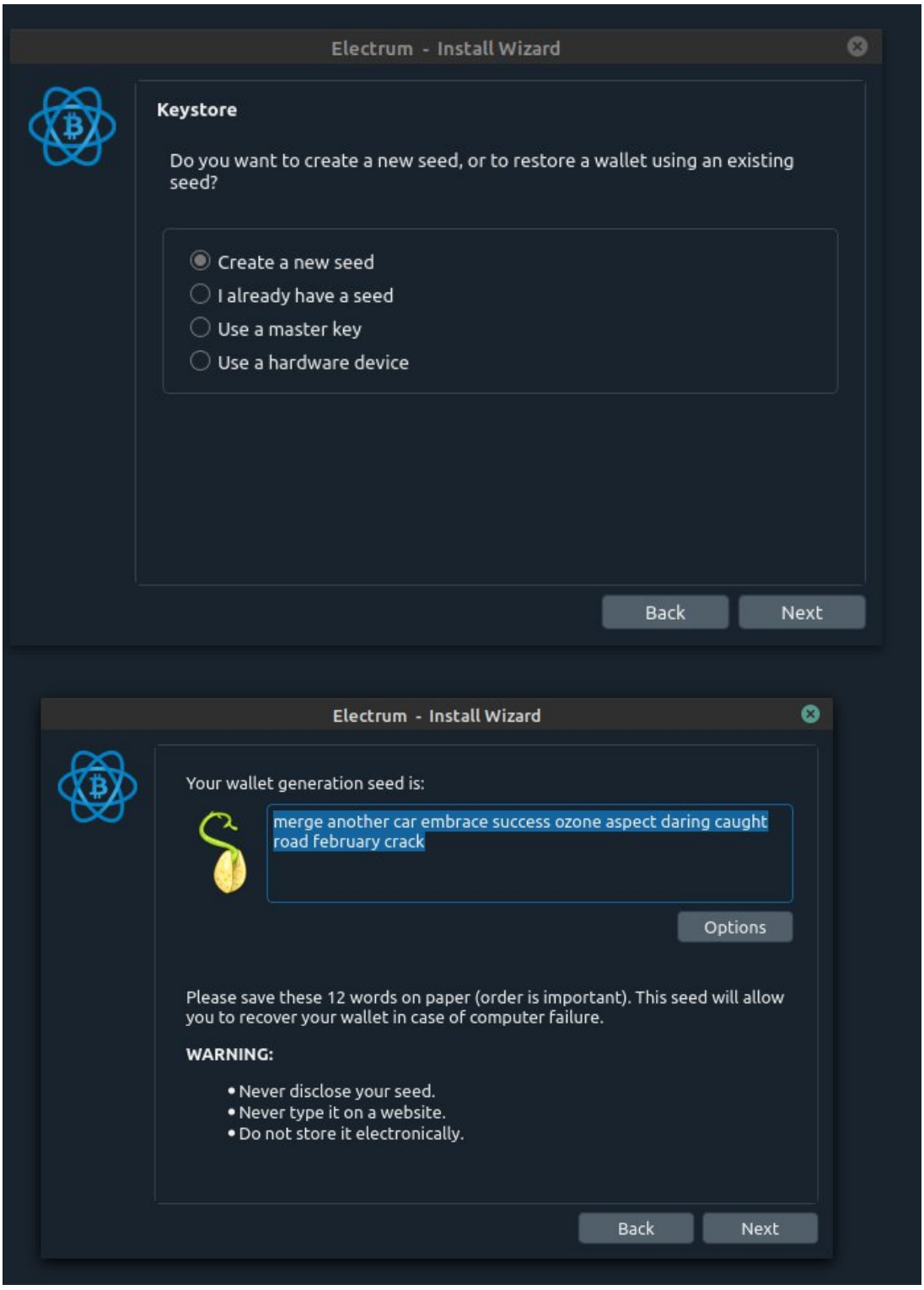


Figure 23 Testnet wallet

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017] 10/09/2021

Let us try to create a unencrypted standard wallet



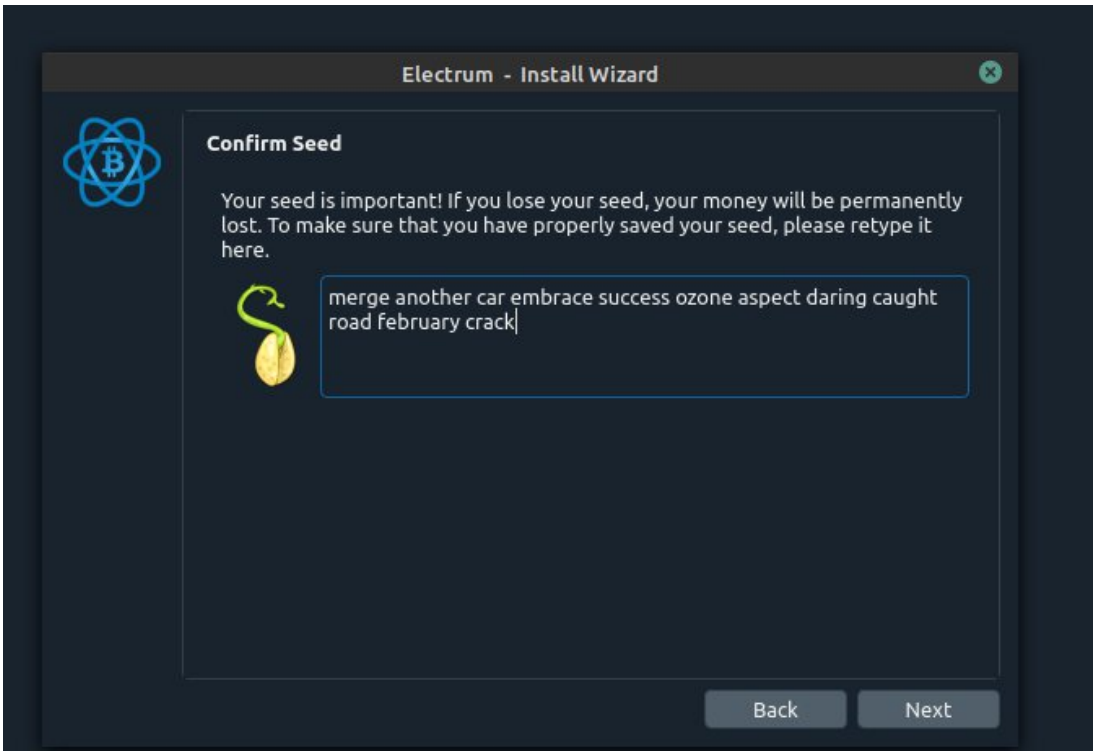


Figure 24 Procedure for creating a standard wallet till conforming seed

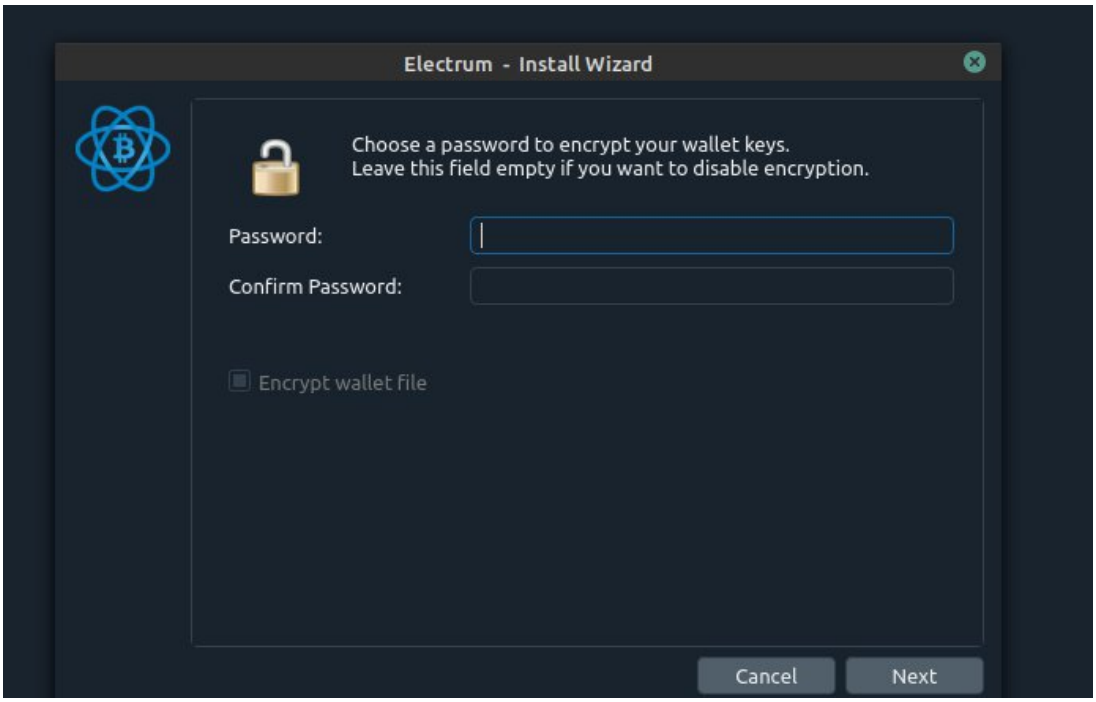


Figure 25 Password field is left empty to see if private and public keys are obtainable

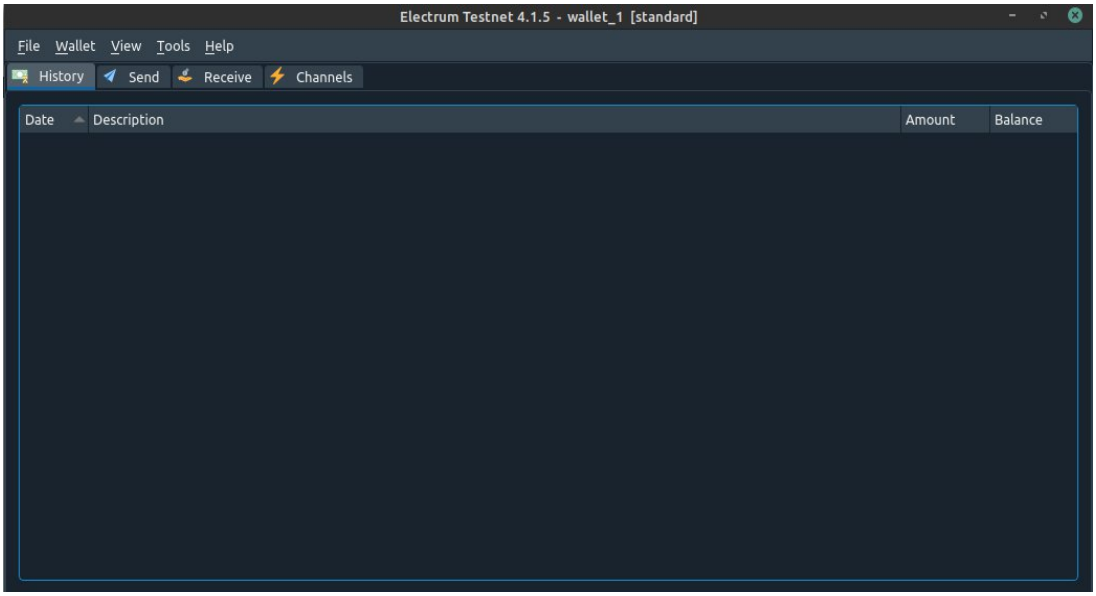


Figure 26 Wallet_1 is created

All the wallet information is stored in a hidden directory in the user's home directory.

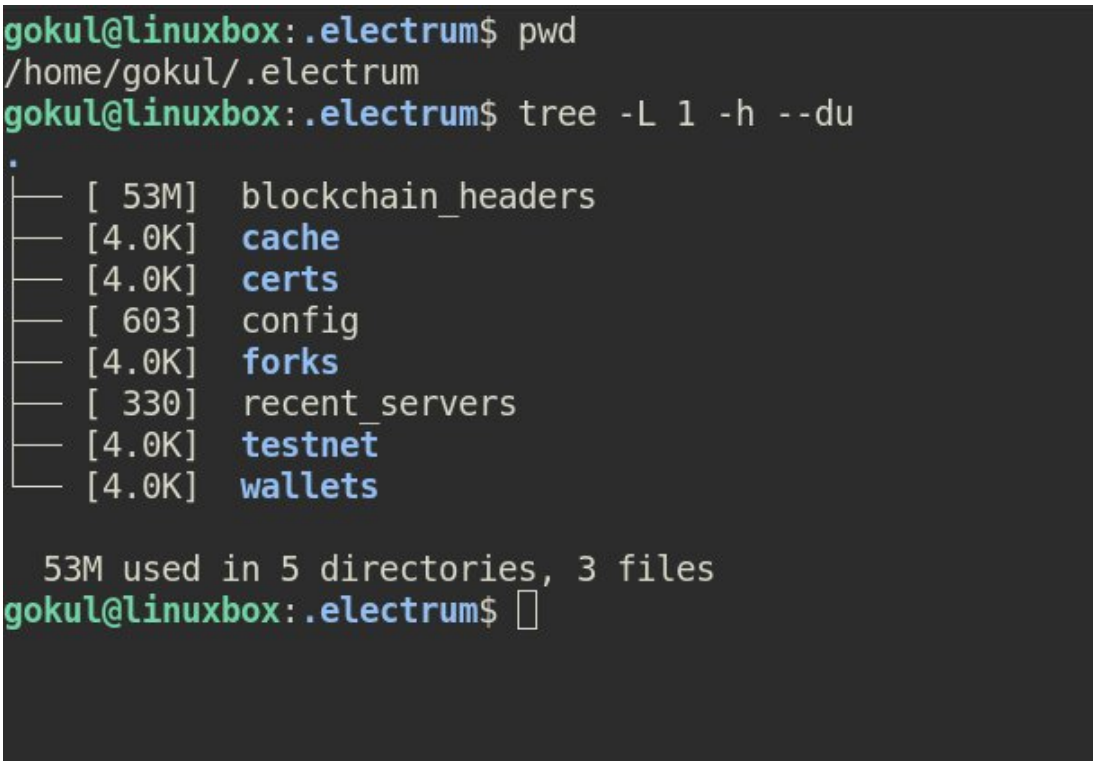


Figure 27 Location and file structure of the wallet

The blockchain_headers of testnet (testnet/blockchain_headers) are viewable with a hex viewer.

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017] 10/09/2021

Figure 28 Blockchain headers file

The testnet wallet (found in the wallets directory) we created has the following public and private key pair. This information is visible as we did not choose to encrypt the wallet during the wallet creation step.

Figure 29 Private and public keys are visible in plain text

Now we that we have the wallet, let us request some money from a bitcoin faucet.

We need to create a receiving address in the wallet first and then we request the money.

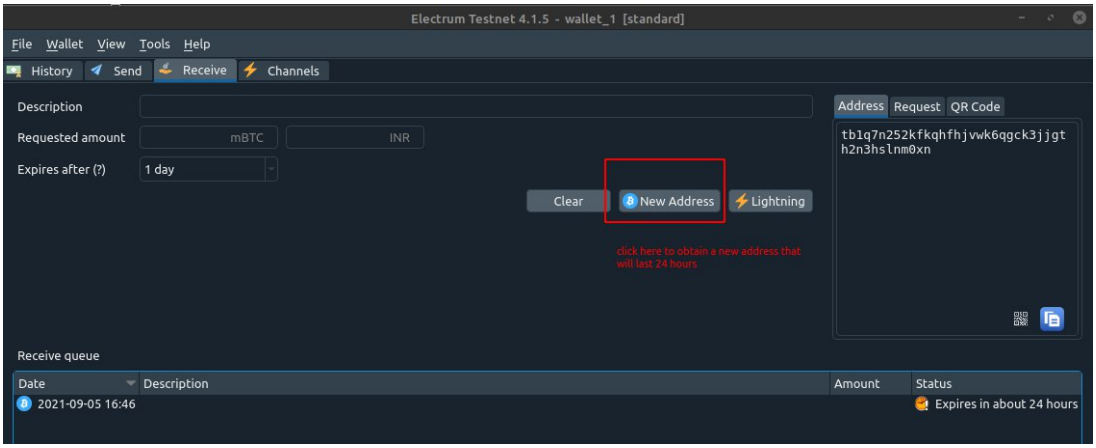


Figure 30 Creating a new address for getting bitcoin

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017] 10/09/2021

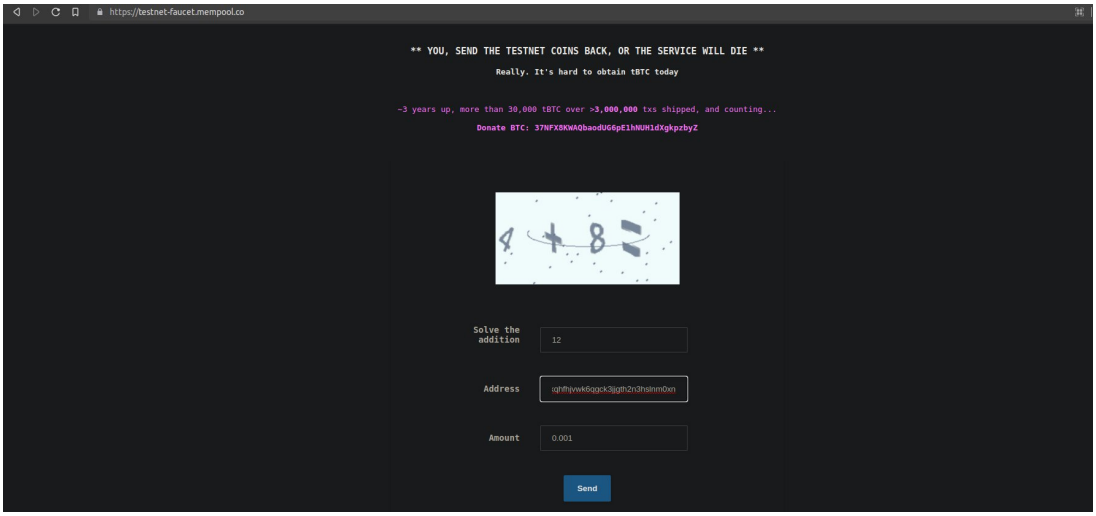


Figure 31 Requesting test coins from faucet

Figure 32 Transaction unconfirmed as seen on the wallet

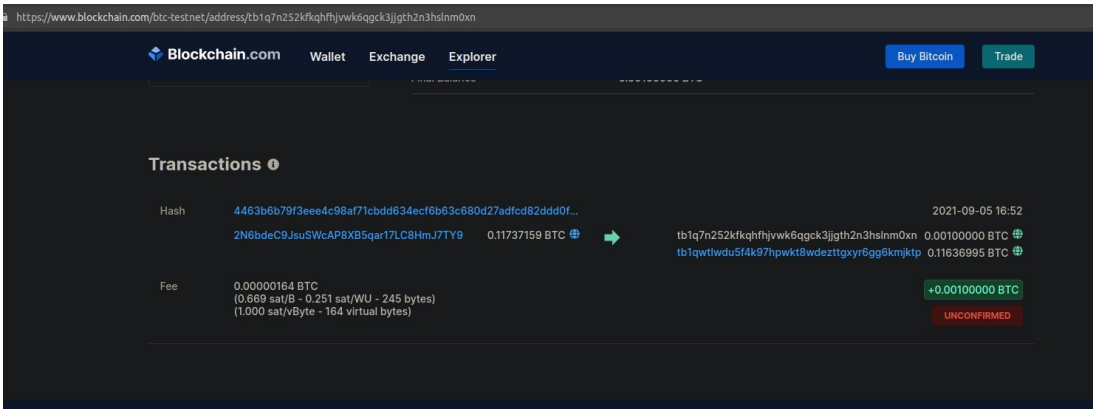


Figure 33 Transaction unconfirmed as seen on the explorer

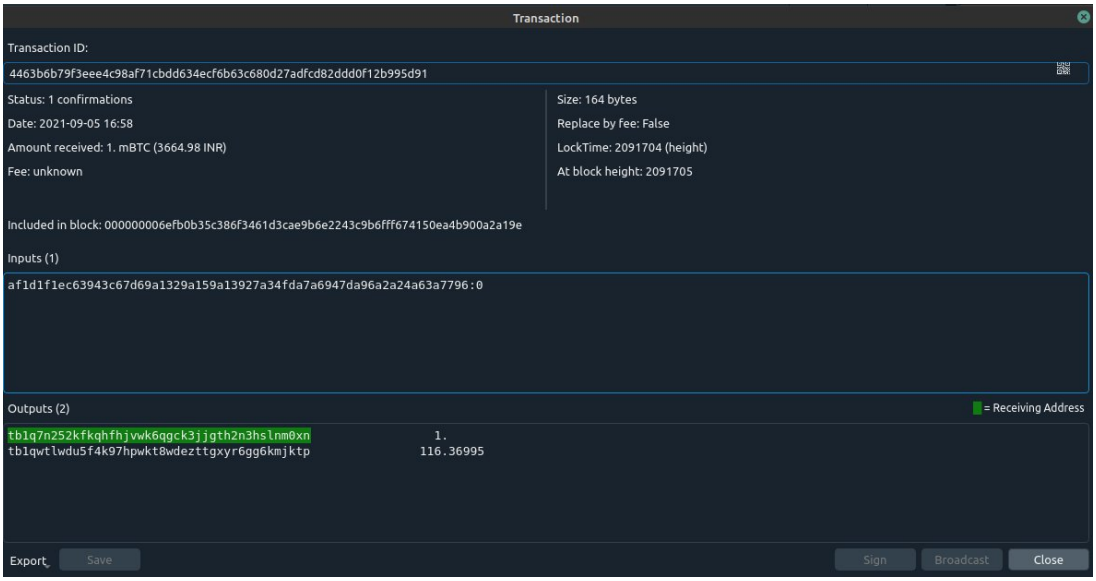


Figure 34 After 10 minutes or so , the transaction is confirmed

Let us now try to create an encrypted cold wallet

First step is to transfer the appimage to a offline machine(A VM is used for demo purpose). I'm using a oneshot server which does the transfer to a virtual machine on host only network.

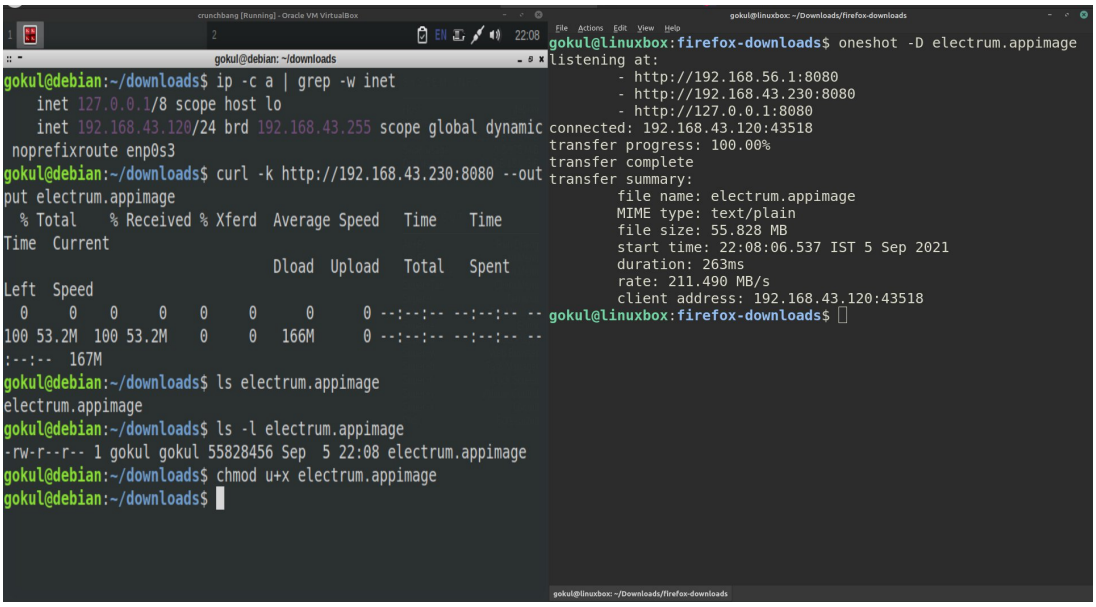


Figure 35 Transferring the Appimage to virutal machine

On the virtual machine create a standard encrypted wallet.

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017] 10/09/2021

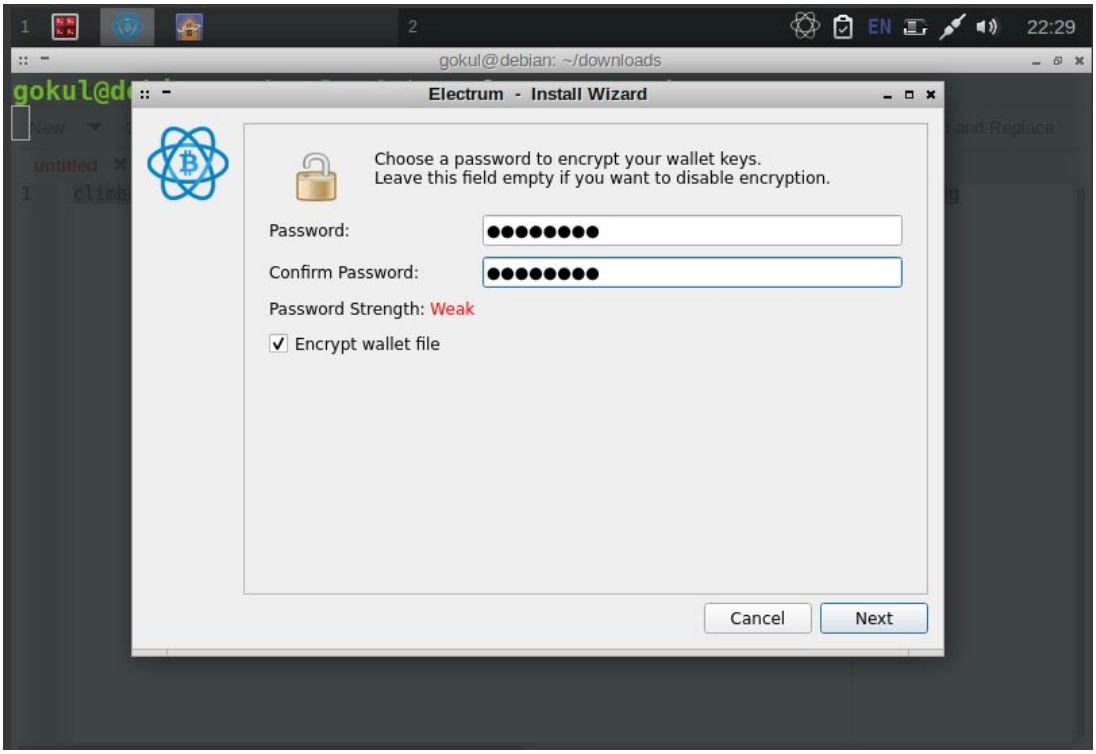


Figure 36 Encrypting the wallet

The wallet file is unreadable as excepted.

Figure 37 File encrypted with AES-256-CBC

We need to get the Master Public Key(MPK) to be transferred from the virtual machine to the host where the online version of the wallet is running. Click on Wallet —> Information to get a QR code which can be used to do the same.

Figure 38 QR code of MPK

Create a standard wallet on the online machine but in the option where we previously gave generate new seed, we give “Use a master key”

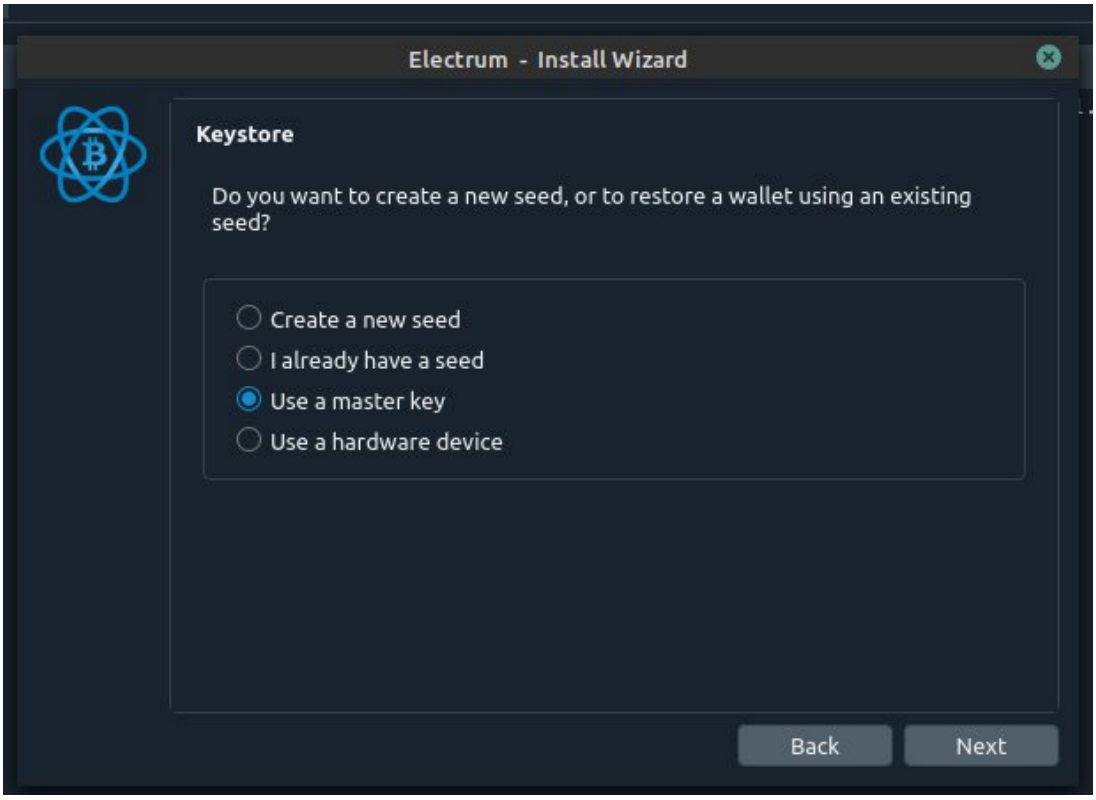


Figure 39 Use a master key for creating a "watch only" wallet version of the offline wallet

And the key from the offline wallet is pasted in the dialog box as shown

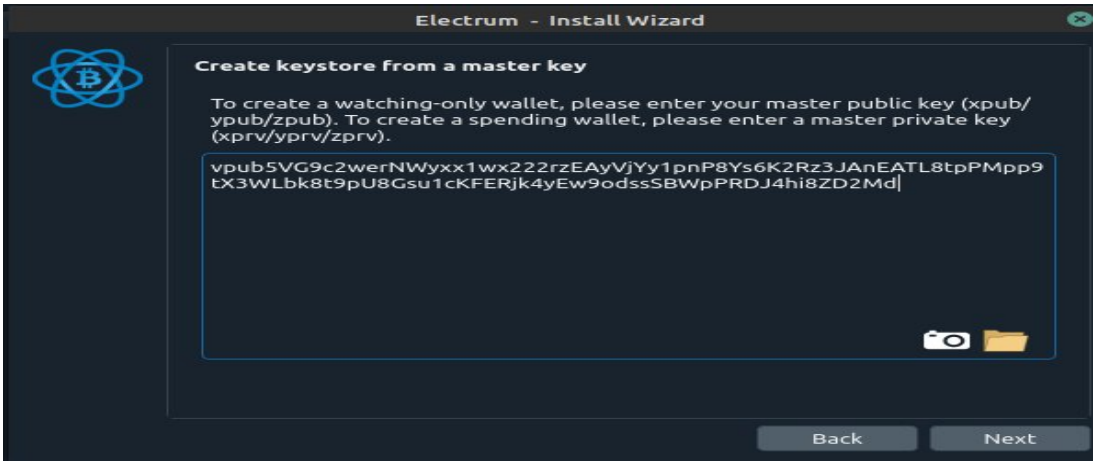


Figure 40 Copy the MPK of offline wallet

Now we have a watching-only-wallet which is online and has only the public key which it uses to monitor the blockchain and the main wallet which is offline which has the private key. We can create an unsigned transaction, transfer that transaction detail to the offline wallet (via for example USB) and sign it and then transfer it back to the online wallet and broadcast it to the network. This albeit cumbersome method of doing transactions greatly improves the secrecy of the private key.

First enable preview transaction in online wallet (Preferences —> Transactions —> Advanced Preview)

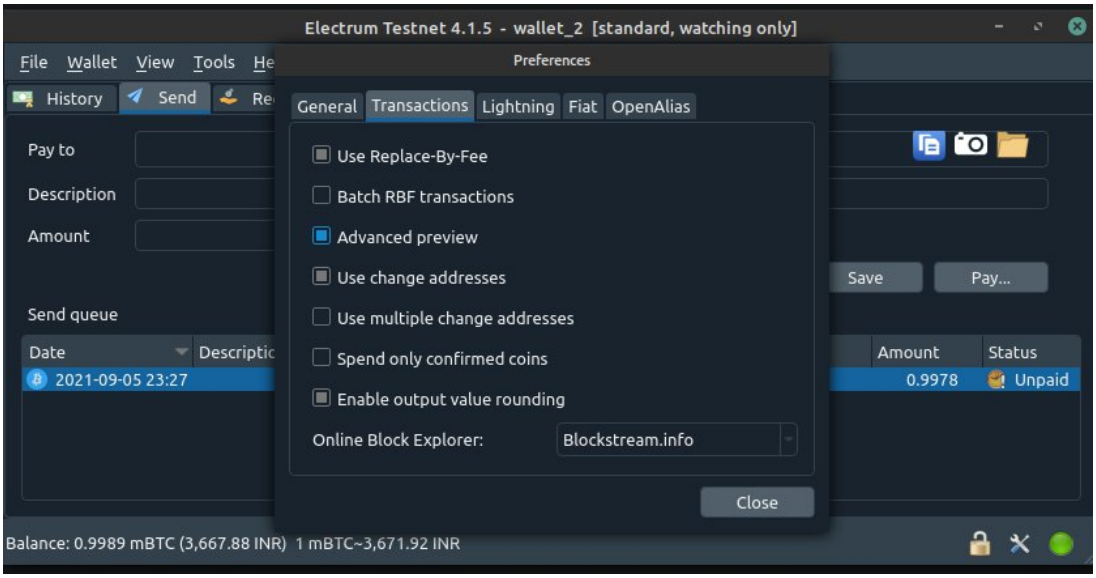


Figure 41 The setting to enable previews is not mentioned in the documentation, but needs to be enabled

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017] 10/09/2021

Now Do the create, save and finalize a transaction and export it to a file.

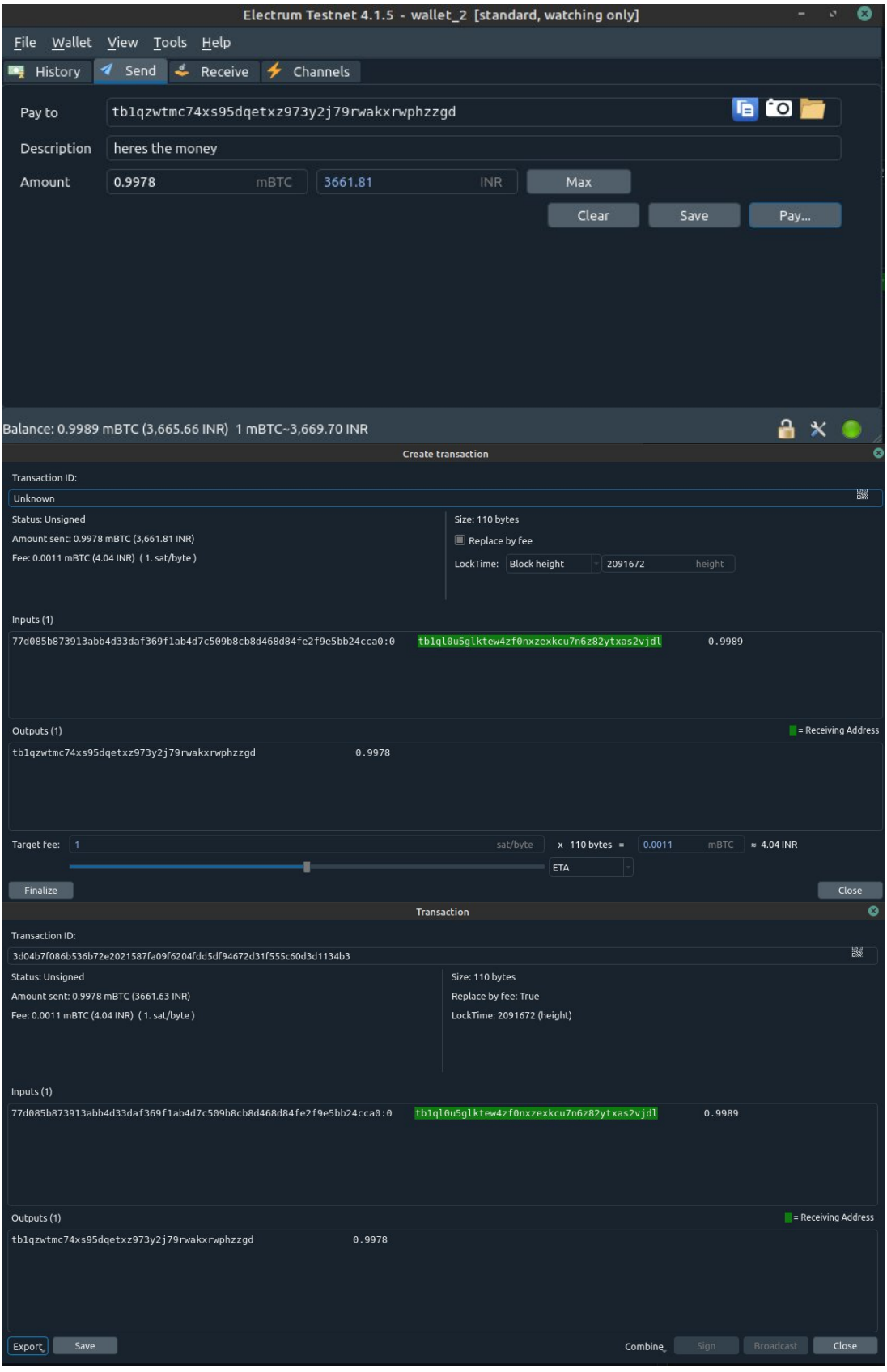


Figure 42 Click on Save, Finalize and Export (export to a psbt file)

Transfer the partial transaction to the offline machine. We will be using the oneshot server again.

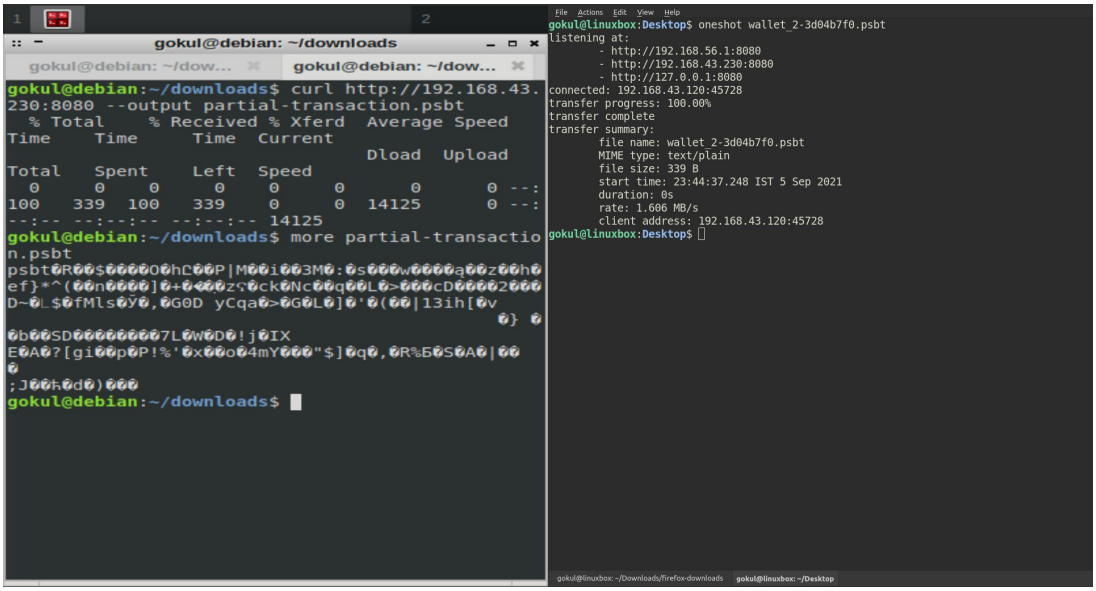


Figure 43 Partial transaction is copied to the offline machine

Now we load the this to offline wallet and sign it and sent it back to online wallet.Click on Tools —> Transactions —> Load Transactions —> From file in the menu and load the psbt file.

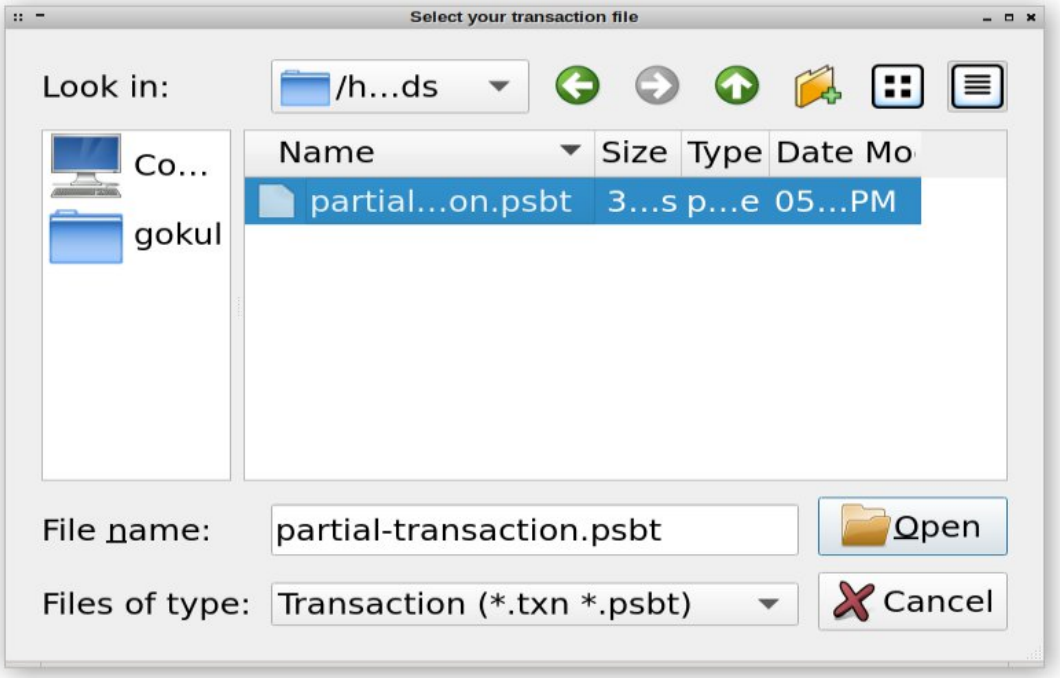


Figure 44 psbt file loaded on to offline wallet

Click on sign and then on the export option and have the saved .txn file send to the online wallet, from there hit on the broadcast option to send the transaction to blockchain.

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017] 10/09/2021

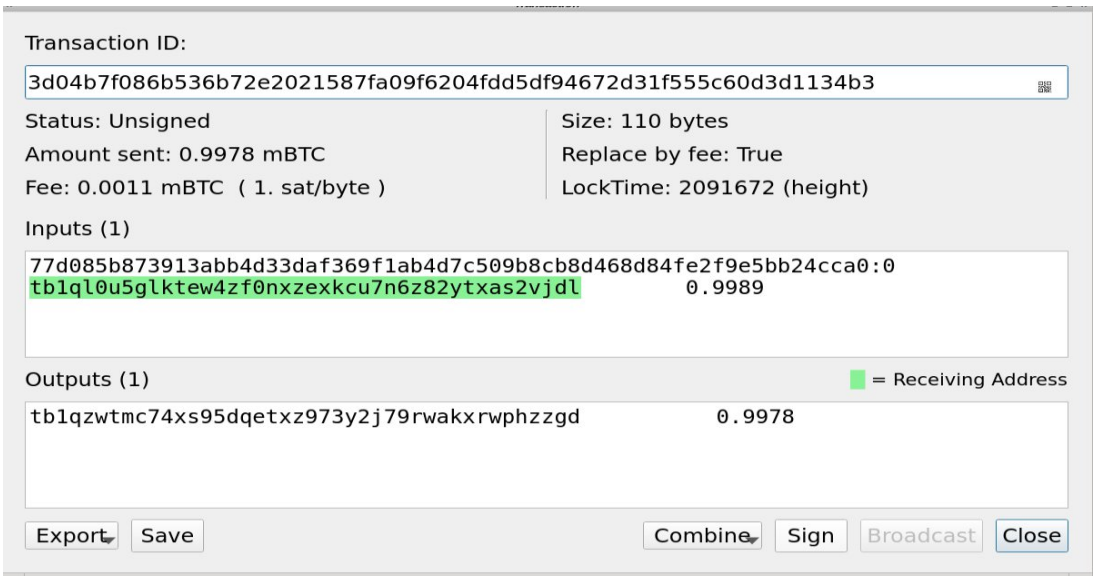


Figure 45 Sign and Export the transaction and have that file send to online wallet for broadcasting

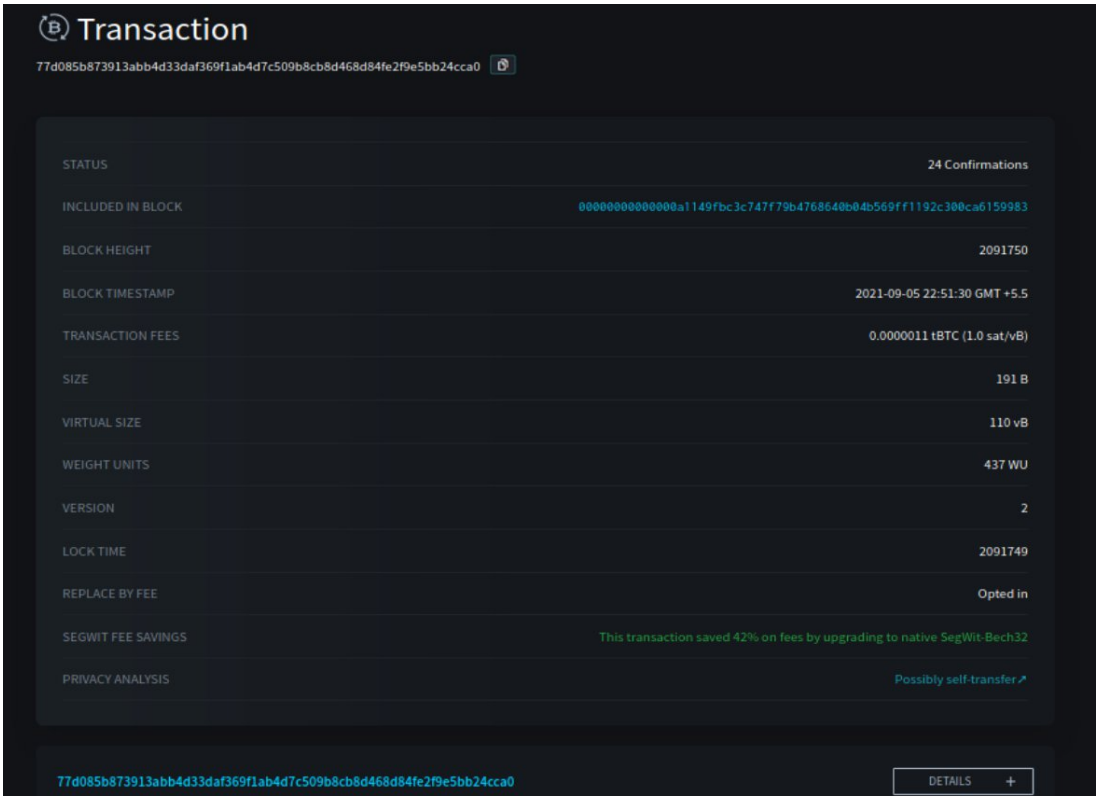


Figure 46 Transaction as seem on the explorer