

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](#).

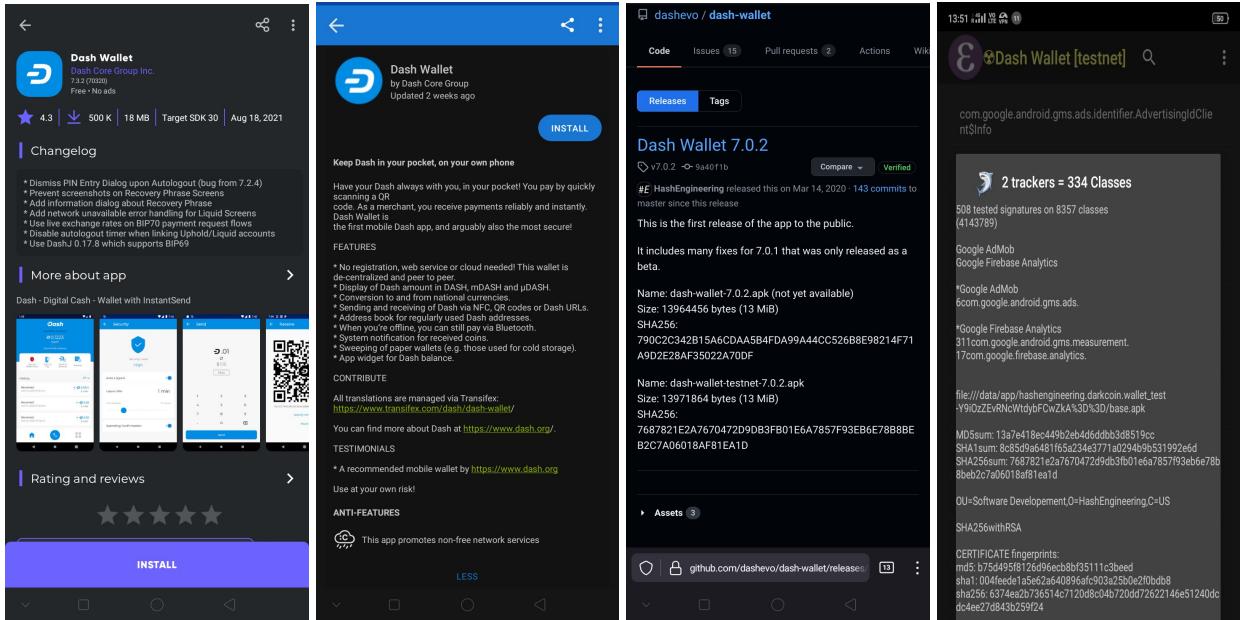


Figure 1 Dash Wallet as on F-Droid, Google Play Store, GitHub repo and Dash Wallet[testnet] app details

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

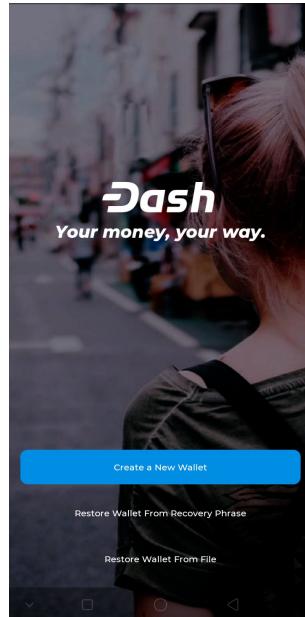


Figure 2 Greeter page of Dash wallet

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

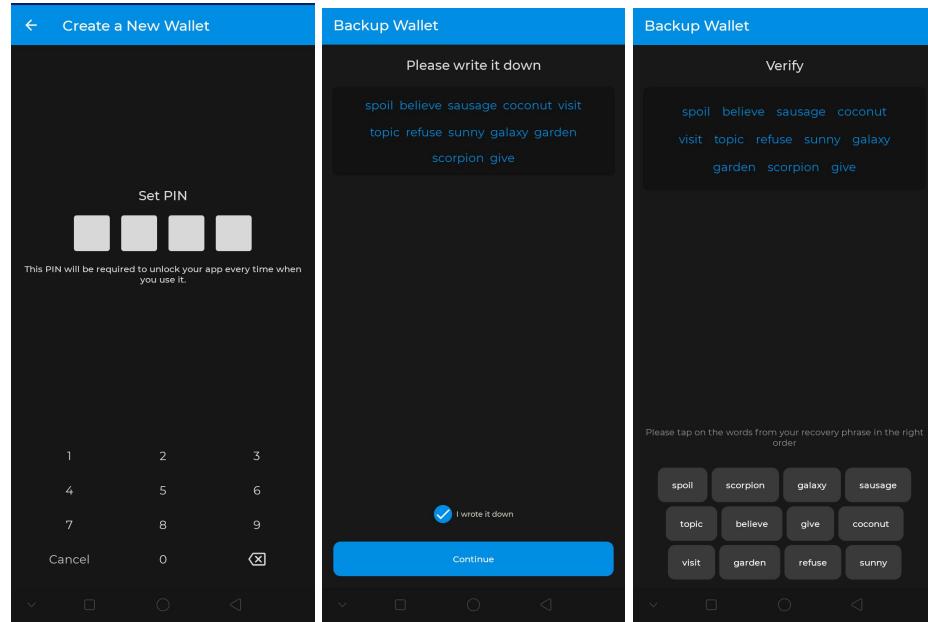


Figure 3 Setting up PIN, creating and verifying mnemonic phrase

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

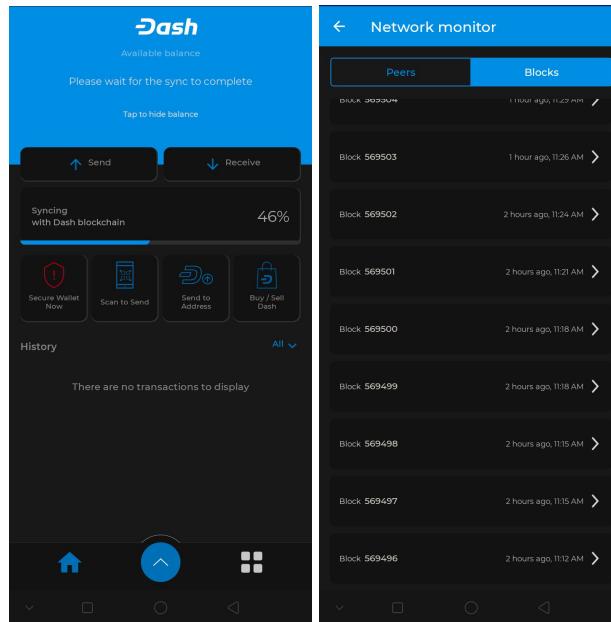


Figure 4 Syncing up the wallet

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](#).

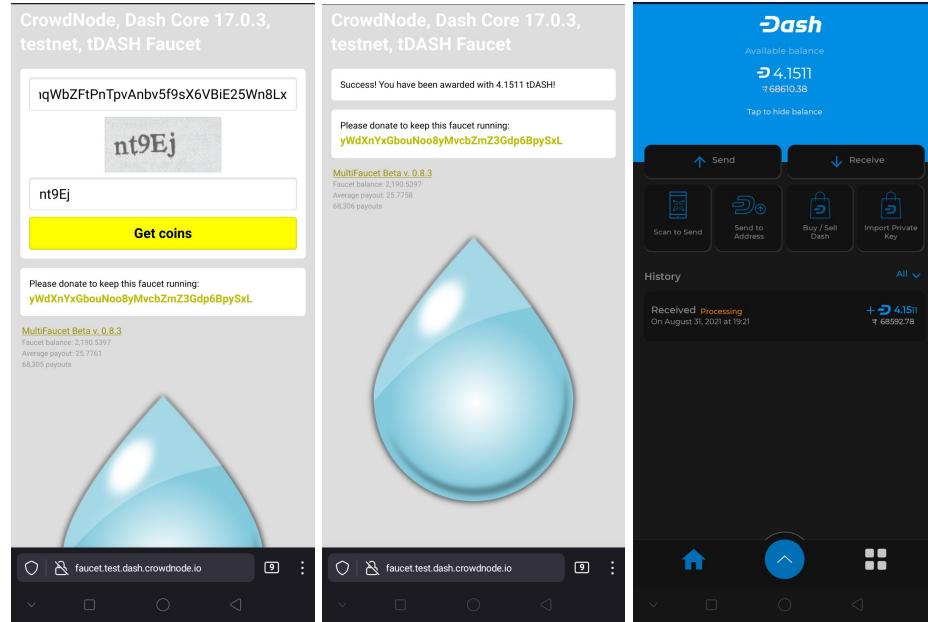


Figure 5 Requesting test coins from faucet

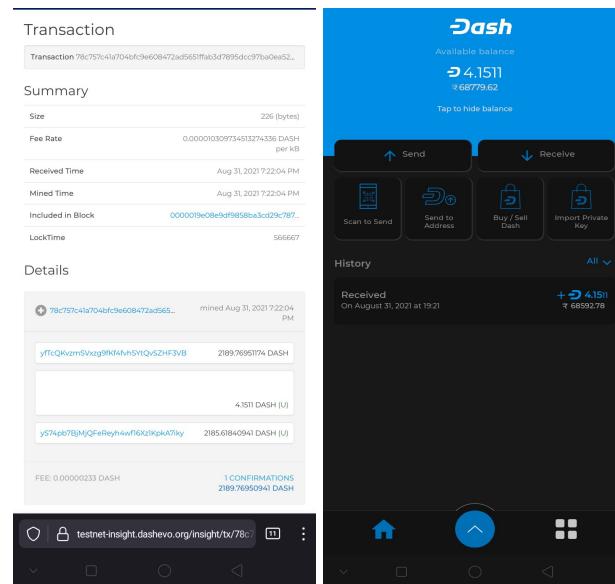


Figure 6 Coins received from faucet and verified it on explorer

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](#).

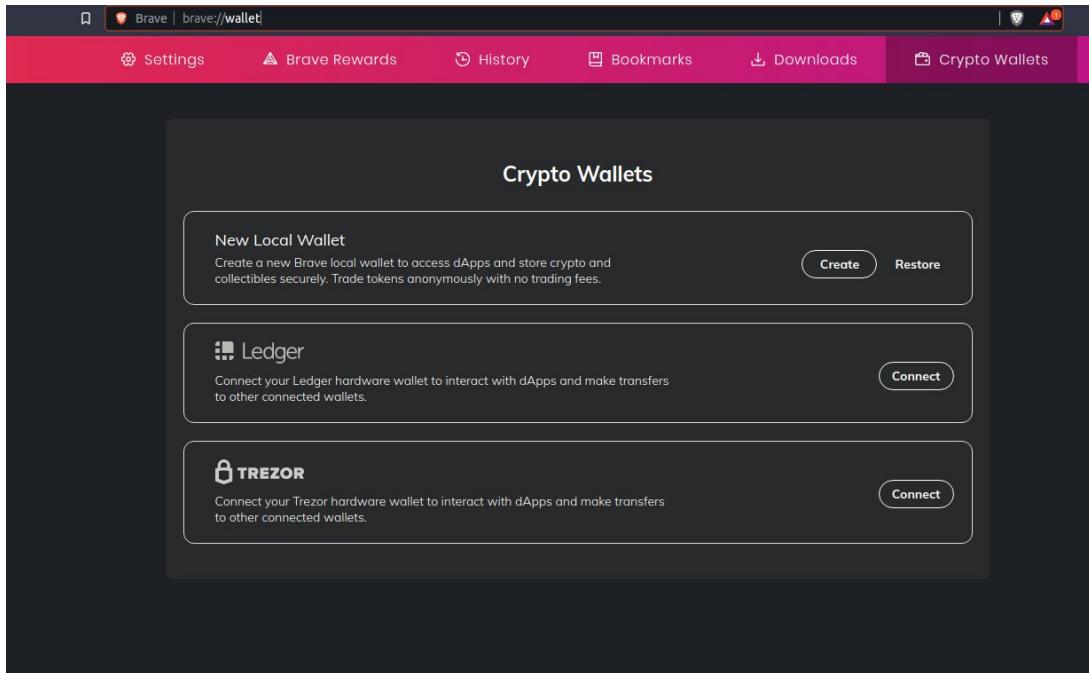


Figure 7 Brave wallet greeter page

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.

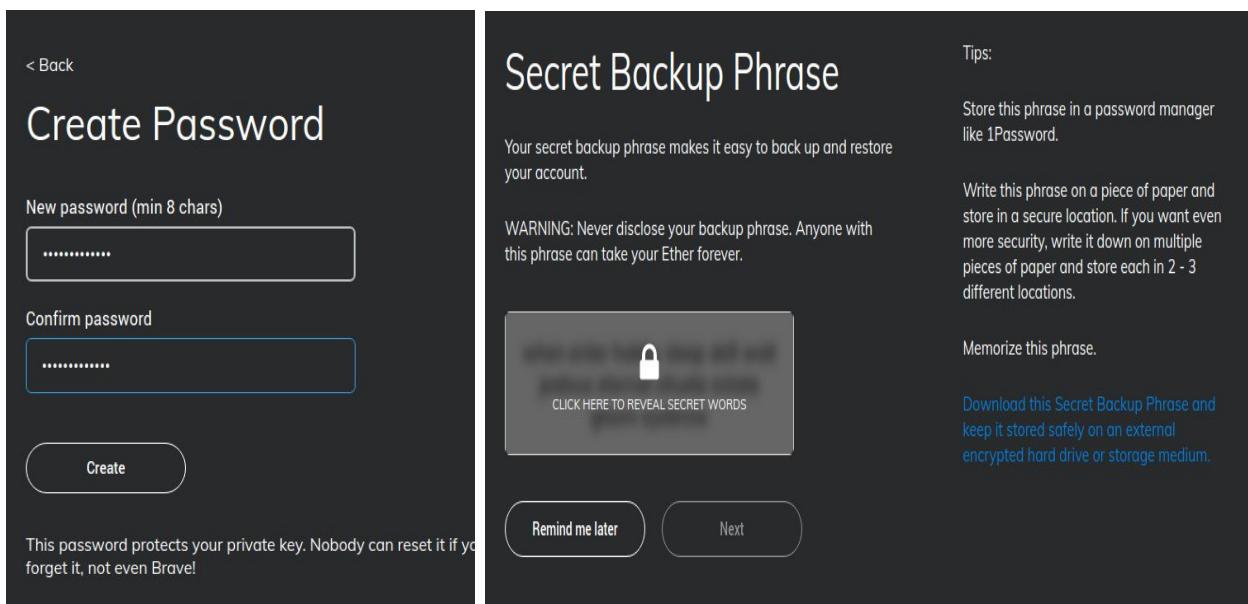


Figure 8 Setting up wallet

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

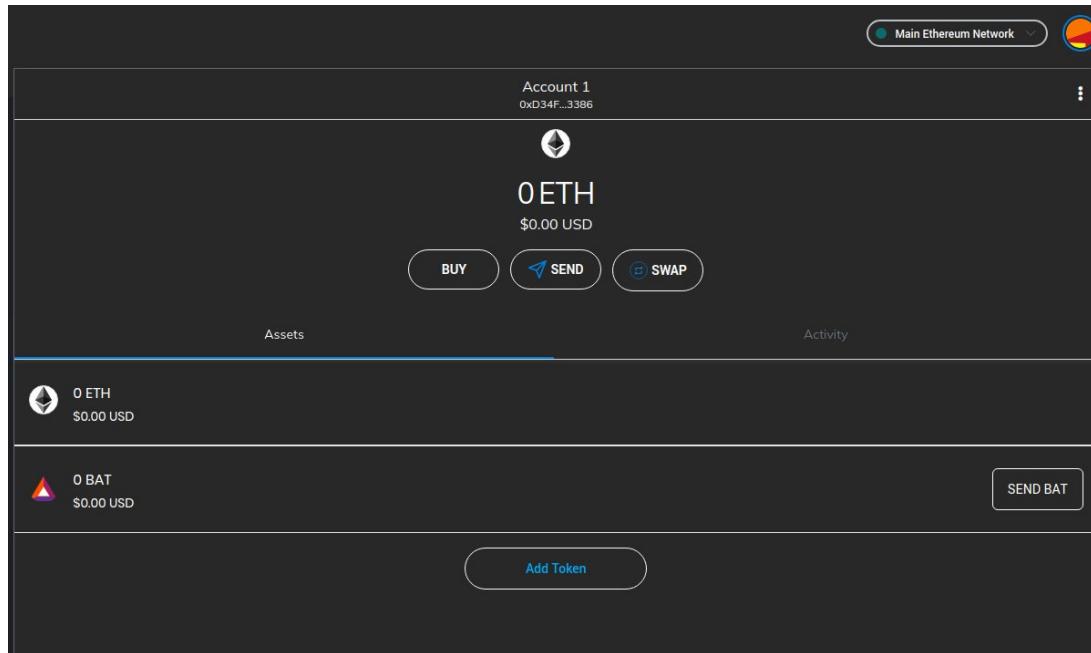


Figure 9 The Brave Crypto Wallet

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.

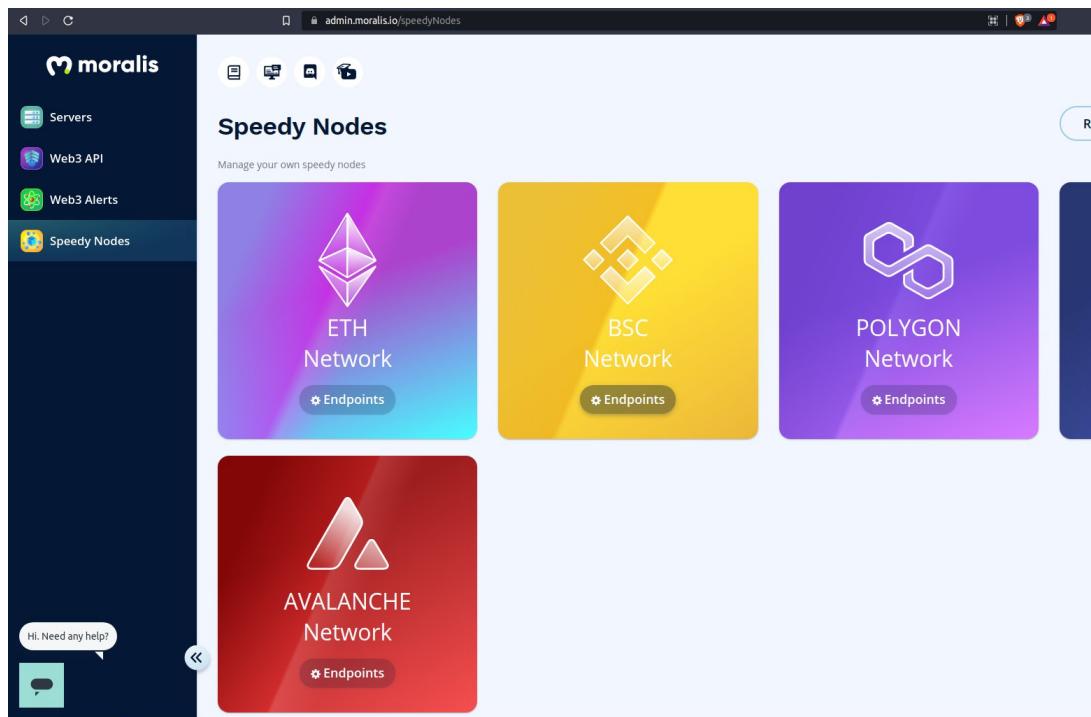


Figure 10 moralis.io node offerings

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

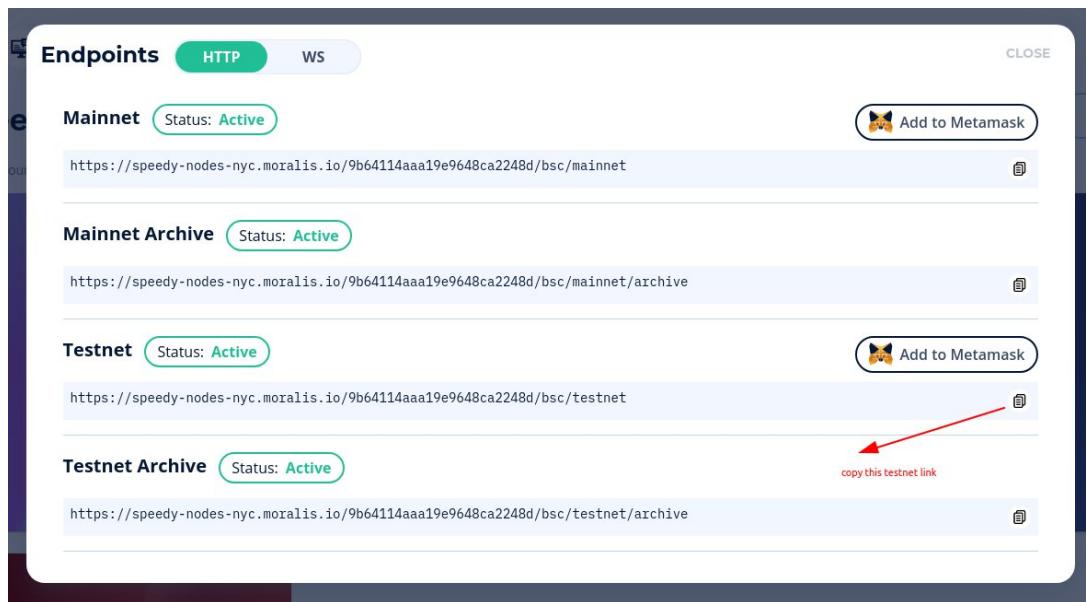


Figure 11 All endpoints of BSC chain available on moralis.io

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

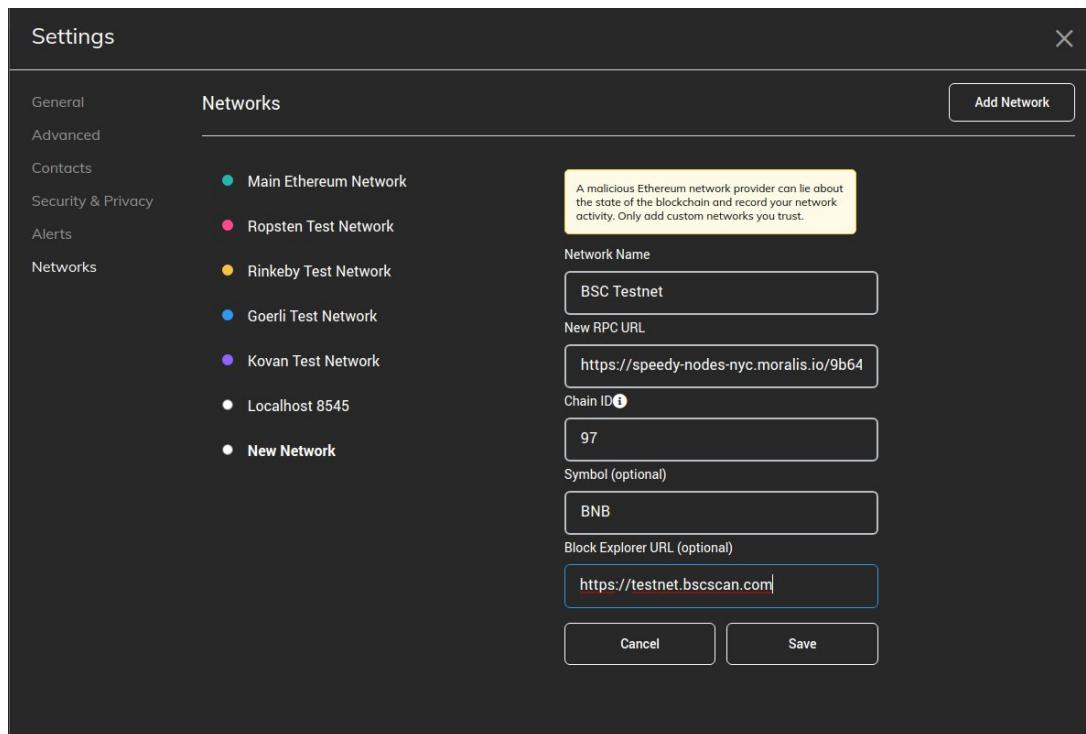


Figure 12 Adding custom RPC

We will be sending some test BNB from my [Binance Chain Wallet](#) to Brave Wallet.

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

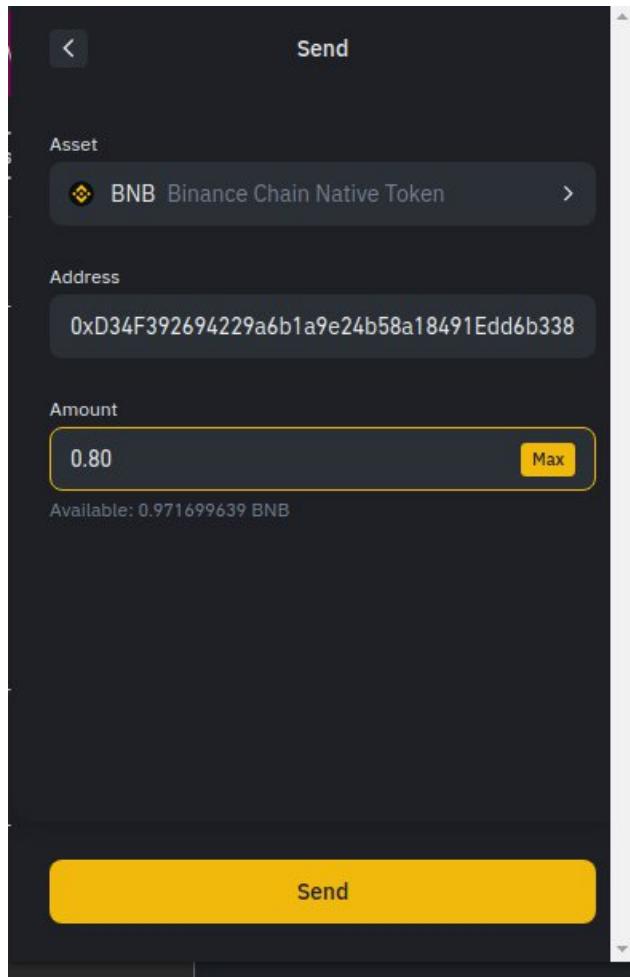


Figure 13 Sending transaction from Binance Chain wallet to Brave wallet

The transaction details are seen on the [explorer](#) here.

A screenshot of the BscScan Testnet transaction details page. The URL in the address bar is "testnet.bscscan.com/tx/0xdc8f71b1dd0209072db56ef922d722d551054937d8b46df2a5fad7f38989cdb4". The page header includes the BscScan logo, a search bar, and navigation links for Home, Blockchain, Validators, Tokens, Resources, and Misc. The main content area is titled "Transaction Details" and shows the following information:

- Overview: [This is a Bsc Testnet transaction only]
- Transaction Hash: 0xdc8f71b1dd0209072db56ef922d722d551054937d8b46df2a5fad7f38989cdb4
- Status: Success
- Block: 12242216 (16 Block Confirmations)
- Timestamp: 51 secs ago (Sep-10-2021 10:22:07 AM +UTC)
- From: 0xace4ab108f9ad4c4c14305d0889c945b788afce5
- To: 0xd34f392694229a6b1a9e24b58a18491edd6b3386
- Value: 0.8 BNB (\$0.00)
- Transaction Fee: 0.00021 BNB (\$0.09)

A "Click to see More" button is at the bottom.

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.

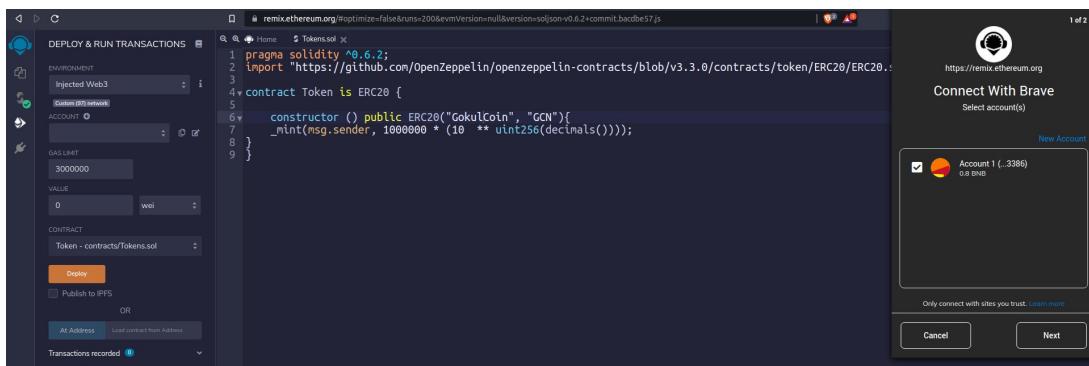


Figure 16 Deploying the token on BSC testnet

Now we can confirm the transaction in a similar manner.

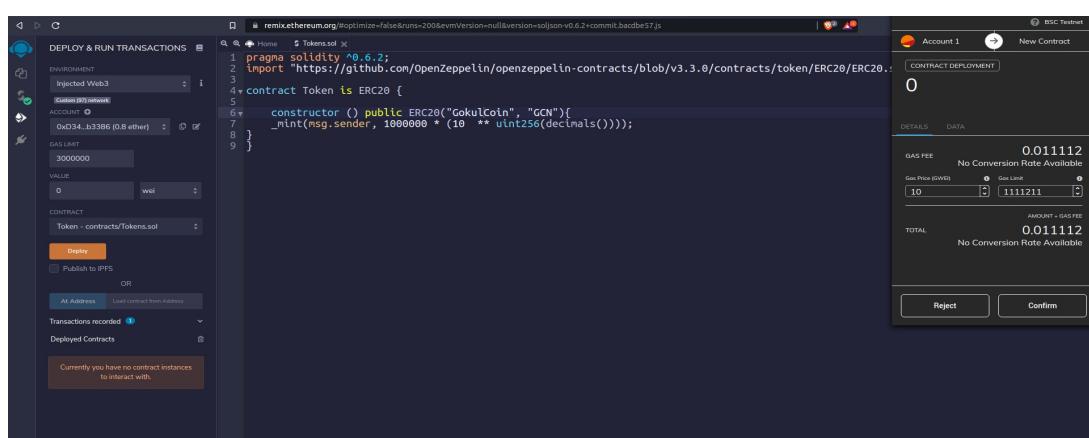


Figure 17 Conforming the deployment

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

The details about this token can be found in the testnet explorer [here](#).

The screenshot shows the BscScan testnet interface for a token named 'GokulCoin'. The 'Overview' section indicates a total supply of 1,000,000 GCN, held by 1 address, with 1 transfer recorded. The 'Profile Summary' section shows the contract address as 0xdf5a678ae10d17af20a364c2637268c742e1be0e and decimals set to 18. A transaction history table lists a single transaction from 0x60806040 to 0xd34f392694229a691a... with a quantity of 1,000,000.

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

The screenshots show the process of adding the token to the Brave wallet. Step 1: 'Add Tokens' screen with 'Custom Token' selected, showing fields for Token Contract Address (0xdf5a678ae10d17af20a364c2637268c742e1), Token Symbol (GCN), and Decimals of Precision (18). Step 2: Confirmation screen asking 'Would you like to add these tokens?' showing the token details: Token GCN and Balance 1000000 GCN. Step 3: Wallet interface showing the added token 'GCN' with a balance of 1000000GCN. Buttons for 'SEND' and 'SWAP' are visible at the bottom.

Figure 19 Adding GokulCoin on brave wallet

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

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.

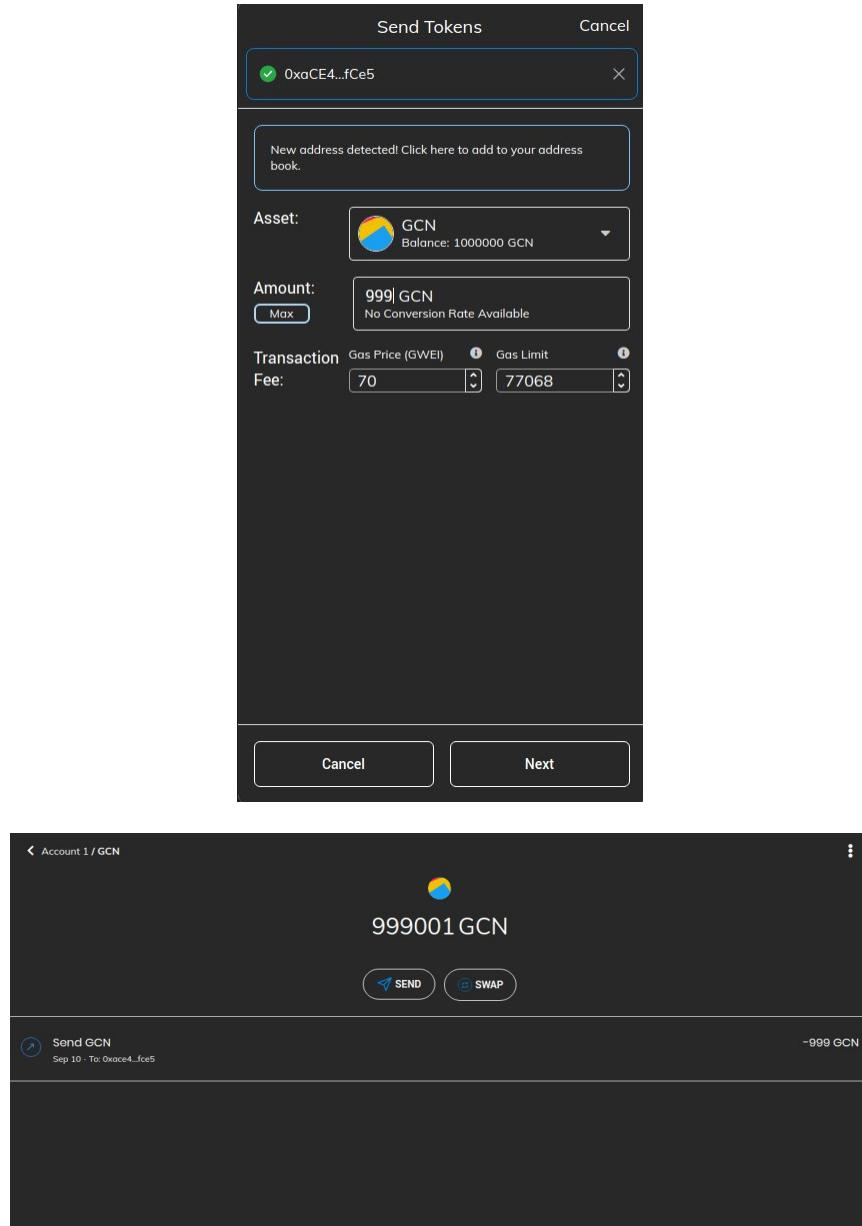


Figure 20 Sending GokulCoin to other wallets

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.

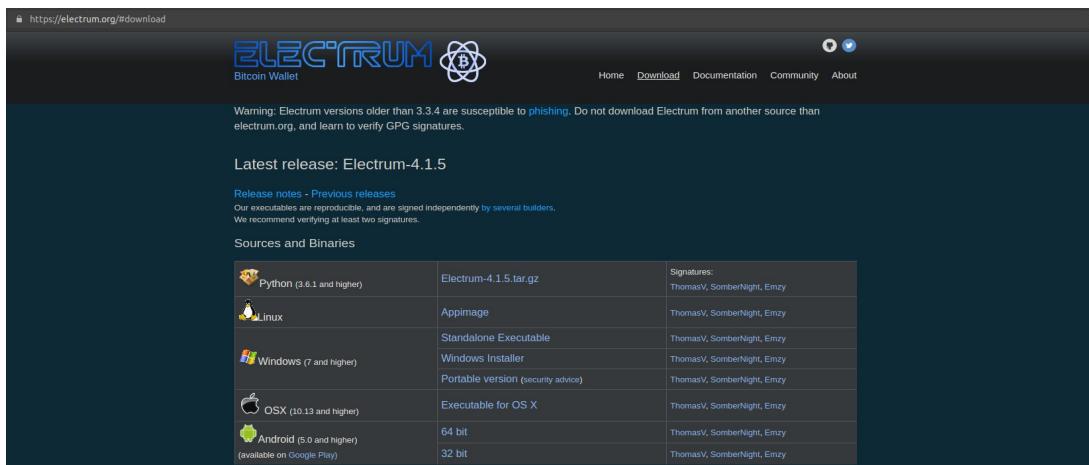


Figure 21 Download page of Electrum wallet

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

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

```
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] [-t 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    Deprecated
  dumpprivkeys
  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.

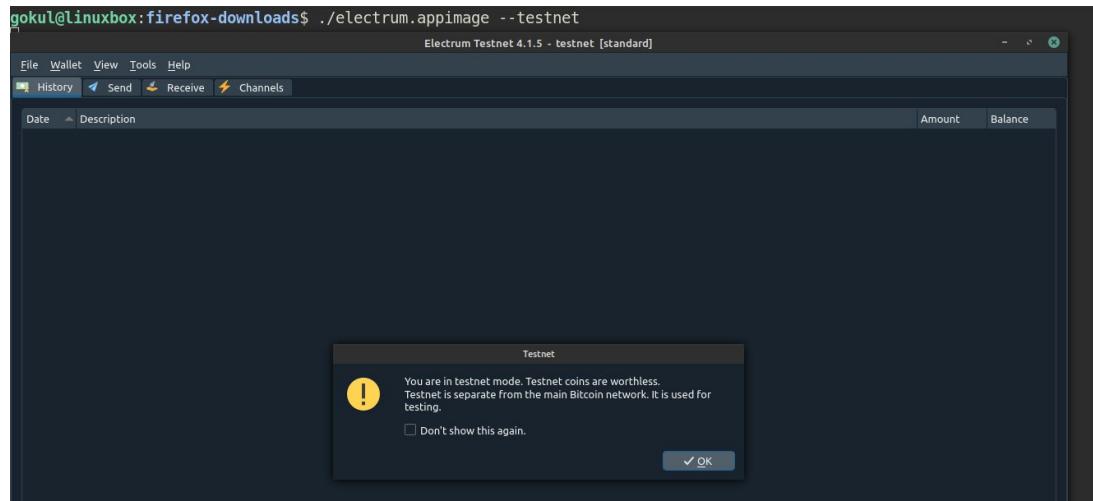


Figure 23 Testnet wallet

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

Let us try to create a unencrypted standard wallet

The image consists of two vertically stacked screenshots of the Electrum - Install Wizard. Both screenshots show the 'Keystore' step of the wizard.

Screenshot 1 (Top): This screenshot shows the initial question: "Do you want to create a new seed, or to restore a wallet using an existing seed?". Below this is a radio button group with four options:

- Create a new seed
- I already have a seed
- Use a master key
- Use a hardware device

At the bottom right are 'Back' and 'Next' buttons.

Screenshot 2 (Bottom): This screenshot shows the wallet generation seed. It displays the 12-word seed phrase: "merge another car embrace success ozone aspect daring caught road february crack". To the left of the seed is a small graphic of a stylized plant growing from a seed. At the bottom right is an 'Options' button. Below the seed, there is a note: "Please save these 12 words on paper (order is important). This seed will allow you to recover your wallet in case of computer failure." Underneath this note is a bold 'WARNING:' section with three bullet points:

- Never disclose your seed.
- Never type it on a website.
- Do not store it electronically.

At the bottom right are 'Back' and 'Next' buttons.

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

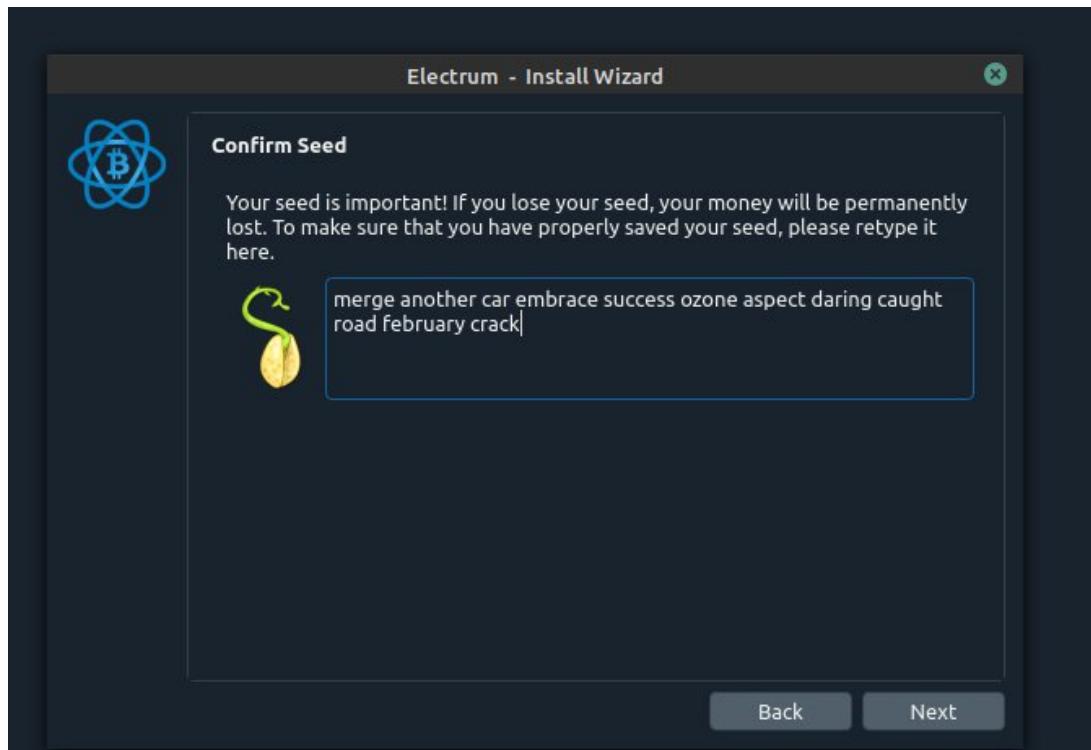


Figure 24 Procedure for creating a standard wallet till conforming seed

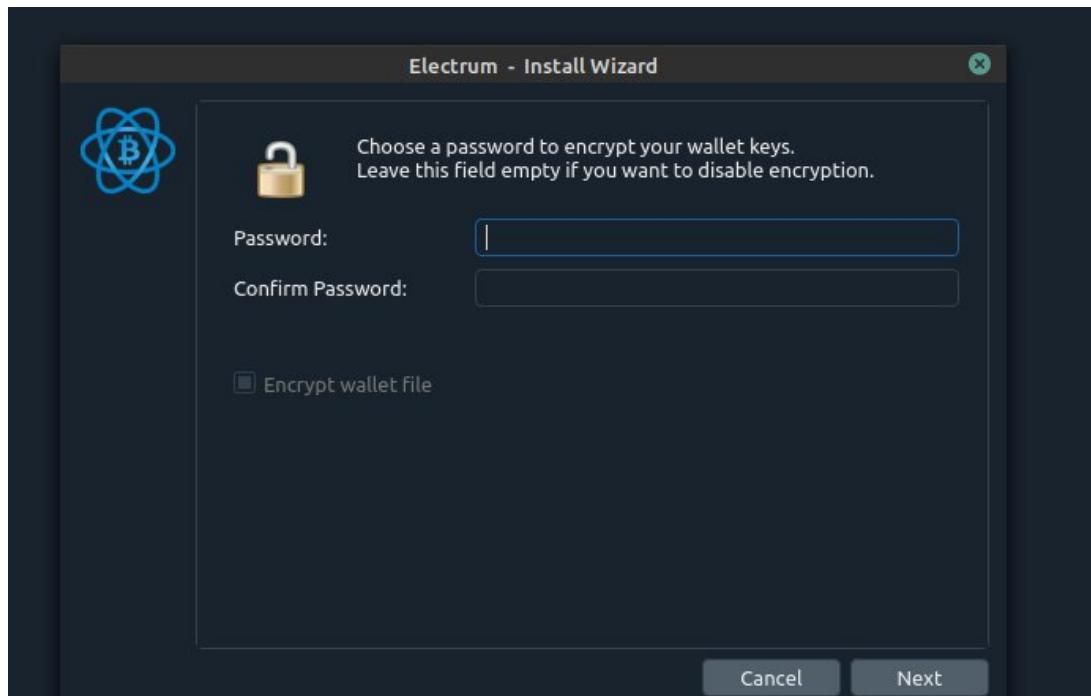


Figure 25 Password field is left empty to see if private and public keys are obtainable

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

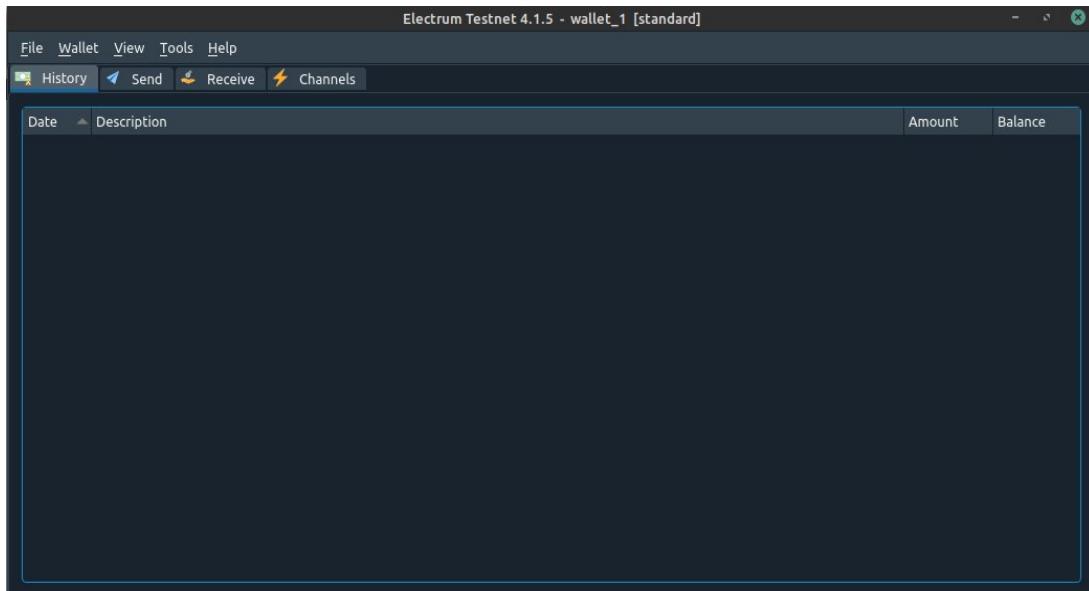


Figure 26 Wallet_1 is created

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

```
gokul@linuxbox:~/.electrum$ pwd
/home/gokul/.electrum
gokul@linuxbox:~/.electrum$ tree -L 1 -h --du
.
├── [ 53M]  blockchain_headers
├── [4.0K]   cache
├── [4.0K]   certs
├── [ 603]   config
├── [4.0K]   forks
├── [ 330]   recent_servers
├── [4.0K]   testnet
└── [4.0K]   wallets

      53M used in 5 directories, 3 files
gokul@linuxbox:~/.electrum$ 
```

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



A screenshot of a hex editor window titled "blockchain_headers - GHex". The window displays a large block of binary data representing the blockchain headers. The data consists of a series of bytes, each represented by two hexadecimal digits. The text area is filled with these digits, showing patterns like '00000004 DA 93 51 FB ...'. There are several tabs at the top of the editor, and the main pane shows the raw byte sequence.

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.



A terminal window on a Linux box showing the contents of a wallet directory. It lists several files: .key, .key.pub, .key.signed, .key.unlocked, .key.unlocked.pub, and .key.unlocked.signed. The .key file contains the private key, and the .key.pub file contains the public key. The .key.signed file contains a signed version of the private key, and the .key.unlocked file contains the unlocked private key. The .key.unlocked.pub and .key.unlocked.signed files contain the corresponding public keys and signed versions.

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.

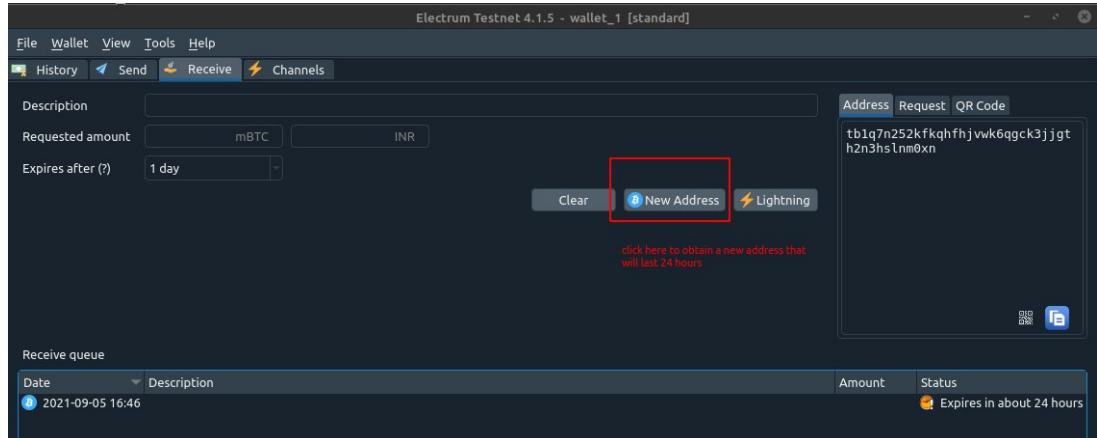


Figure 30 Creating a new address for getting bitcoin

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

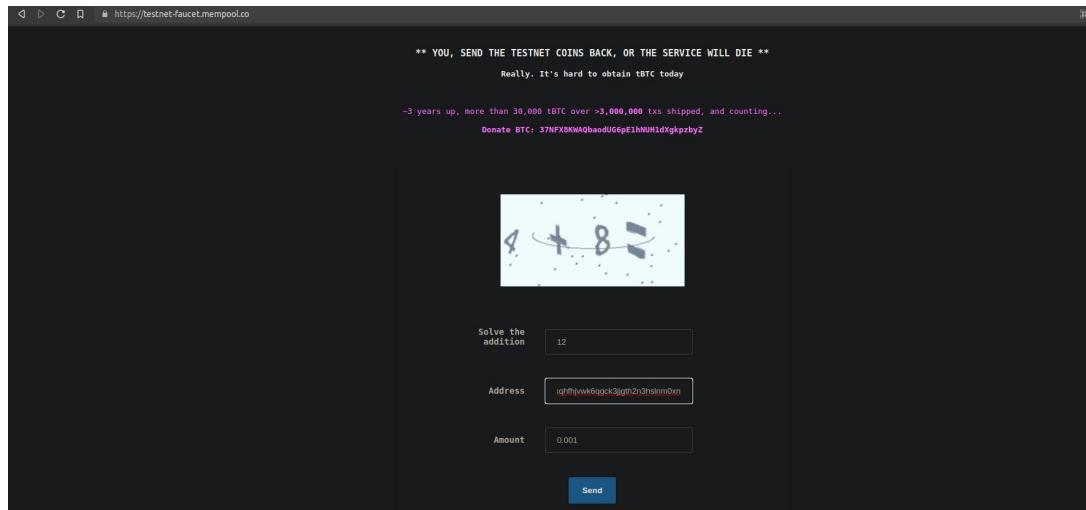


Figure 31 Requesting test coins from faucet



Figure 32 Transaction unconfirmed as seen on the wallet

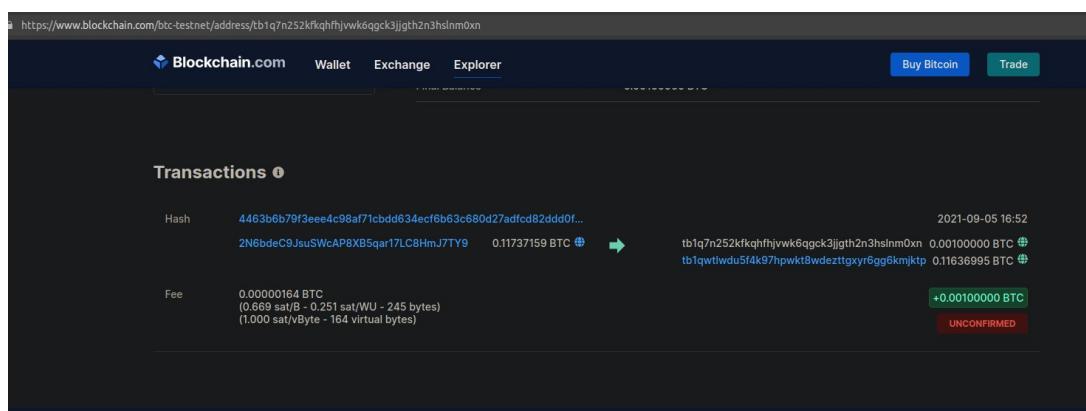


Figure 33 Transaction unconfirmed as seen on the explorer

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

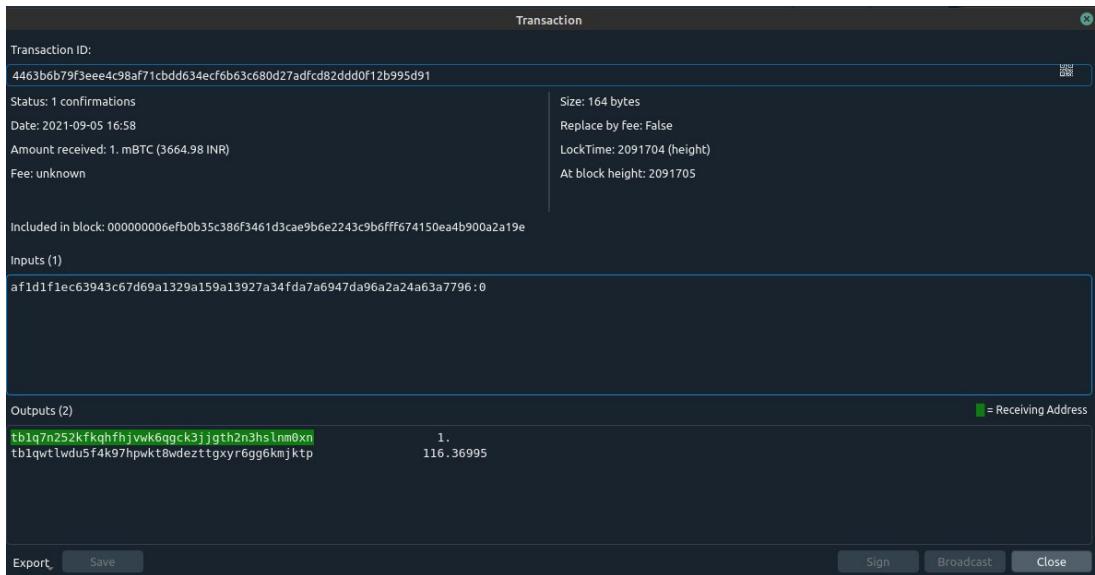


Figure 34 After 10 minutes or so , the transaction is confirmed

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.

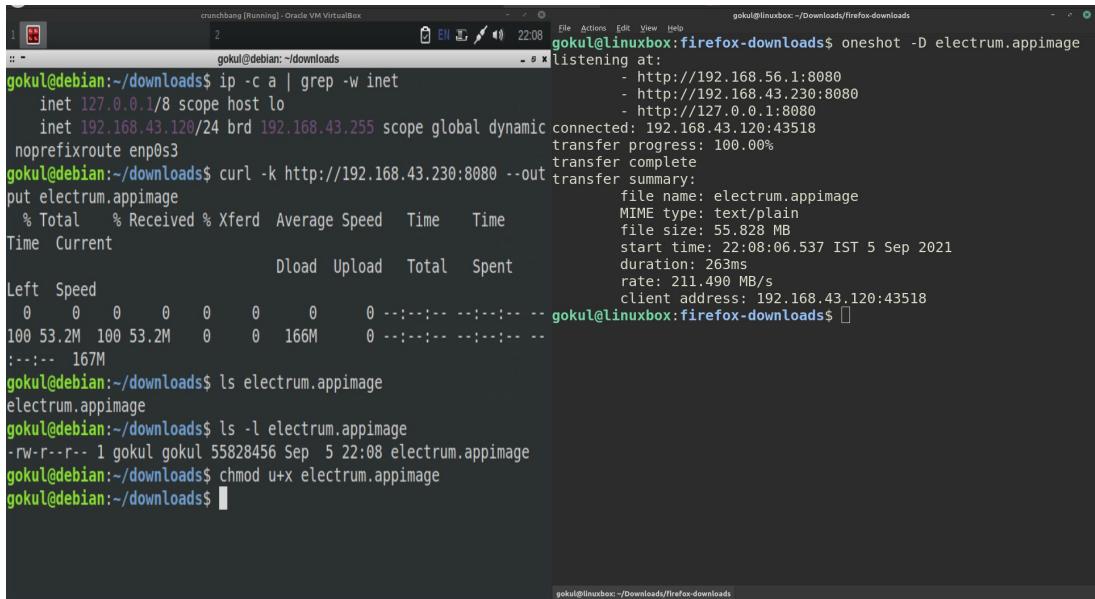


Figure 35 Transferring the Appimage to virutal machine

On the virtual machine create a standard encrypted wallet.

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

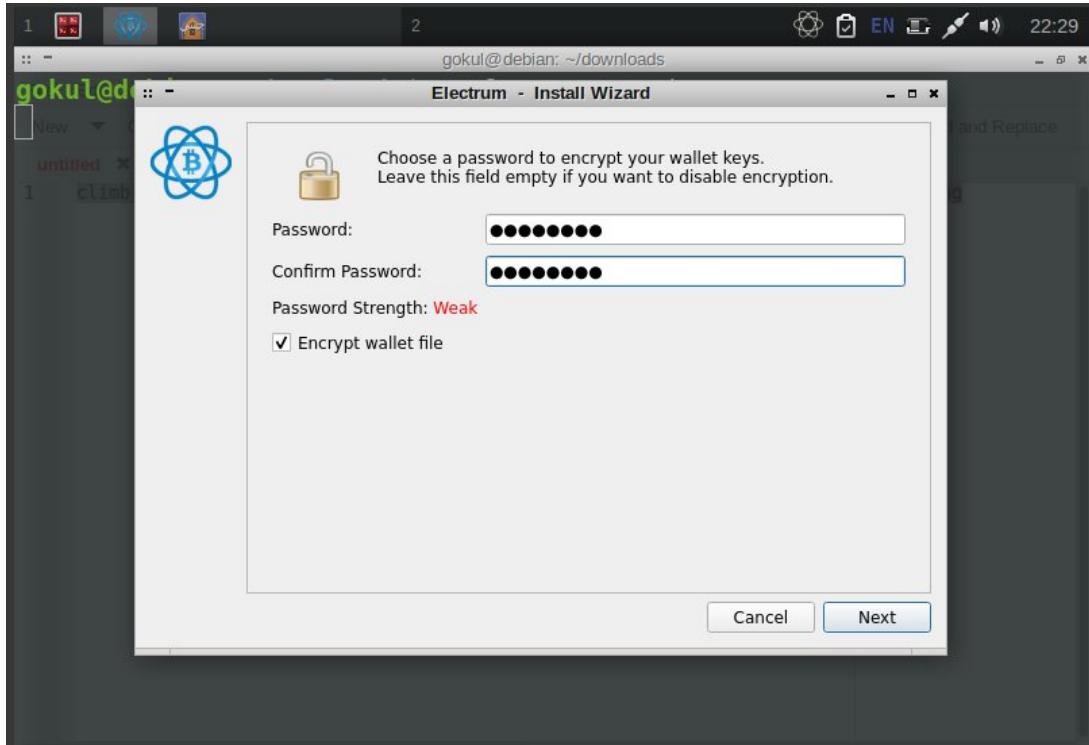


Figure 36 Encrypting the wallet

The wallet file is unreadable as expected.

```
gokul@debian:~/downloads$ cd ~/electrum/testnet/wallets/
gokul@debian:~/electrum/testnet/wallets$ more cold_wallet_1
OkLMF0Jsql83CPGK0ob7kSc/NONTdYL0+jTis/WoN6F1Bqe4pmB8qg/T3BPeRmAgJV0abgPZ++EBho60M0JRVw5n1pXyq/KlDnepefS9vVV0qiIxduQpNet7/4FGDcuICH/dkM
HhwH5A3u30pMaG3p/+Jv5/81E0$9MDm0o0Aehg7TphcxzETlqa0aYfeaxrgoAp9pZjRtUq0mEfVnpXadoMMQ9niY-84lgX6chFu+a06SNVn13S0i0iJ0HJftnFsL/tnz+
BT000DXJpZPVYjvoTRtCRX3AnqqWbf206GeWslj9M0dLfkAhKZL1WF0kNPUEAs9q0/Z6bW15lct7Pj5jzfTLzZ6f8JZzIXx03YtJs009ry4W0ohi1wfNdV10v9AYzuKZeHN4
h+VLwnhD+I0raL7x3+Udm1TVbW9hZ1lt+68EPNV+0mjmW/9V4h+Lup1kr+uKHMBCXvaGN9xvpV0mEd9bS6WMyvwjFmLVgnSi4cb80A50kPjKh65ogLaKnpIgjkEg-k2muw1
iiqYRFuBA5DowaI3XEEFy40lw9VAZUEWSMtFUh87c/Q8K3GxhnbZLuakU/R0TSDTSLUHMioPo4oqElPwFFoA4r8RfvuaIB607isoTMlobBa06DXYkv70nCxmrK/pLwDX9btP3G
TetIdYedal2/6h904k2x5+EVTEY0+4crq3ofqla/RPd6t97s78m045/Q0c05t/9j1Onah40Ycsr0tMAUegf0nfncs85QRA2HAKsas+DjchAoFCIfm8/Txf3mhMpX/hmLb
040kb5bp1u9YDp001e0OpHcCxbc8Xhx1l0VnVwToV91z76bGd514p1Izy2k6Xyq/1dBoN1SHDszvqpxe+EfobMctupl854GbIBHw+N4wLRDpumWzlc4svcqMaFo1XixG
b2PZndrJaaufw0DZXzEcK5MD0kpShw80cwFJAgZjmh1WgojNqUsa2z+5xL0b4X/UF0TJJ1Clqf8/K7w5ashs9LzhWbqMRjzzFmxKCZ4tE/MdT/0D7+pPFNt+uTyhJu8e5zt
Vp8QPQoeEZJrcifQij4209Rj03LFdbFDC2+yRMBO+pa6WJz10b1FMX+5U/IU4EuV3c+bJ3BEBBfGY0095bk8CJPd4GWfnKXpdzx1TwIC9opNCJVgdwW4SiRPmlCDyv05zVxn
g76VLDDdmw57ldtyg2LeVebrXe7oatE2z1+P2g$+YLyfmpY38XD05qr1mdnq/pOcooQ/HLEiM0sePQ3DXE0jbraXPfx0oMMFmDjYqzHnK1eHf38jYkdxChp0BmwCk8oW
dycyxA8ZAlOMcyXIRtQuwPhA9yH4LZ80PVb/UPZEbmge9pgpnXh53+tu2uo0/sVcpjxrtLKE43in+uCV4IXobchYMKZVbSy41Ufj00LMUfxJdxJx61FKxNHVqipQcgcoiyde
a6zrwp+R5xbTpA4Puxo15GCRskw01IyqHcRw09jfVUGwx5V7nra8hLYFxKnPSSza6B0b9mo5dLrKnyatrsNpN1280TrbaAcupjxFrK6vp0mUkkrxr59JNmDIEi504lc
x6PLRpZexFw6bIW2Ggd1X0ko7+T0emxBi28EKbBqauf+XPNPLm/U9zotB0b9f3/NikunjEBEN47o1QwCty799JFNR/bjxQwp10G7fr002nDExr0x2dMjJnPumvBkPwTA953Fn
bvJ+9ccVwSIAicTF6opW6p7b0kW4uyHmk0kQJclV8q59ccoeXh3qPkfp33MdfLAczt/xh+TpjLny1tIEW10untZ2hUooco0XSw05D0/4dZ/U6NE37RU3q80snASPN8XYw2hmz
6FRafgnlTfxjesH0GfI7hLeLY8CpsFhvC2xp1dC7bMv5dnoARz0A0jfpC3R77Rv8qNtH2hYfRnAIWC/uWtdGuKyv+E4NxrsBuJnvccPQhznLx0M9edlgx6Edgauj6xu
IF7MTx5zmBukAeoNplgwcrIkhQ1U0p+cbtFdP0R0zwjT81tuhblaxJ4CwA0bmY4V65fhdcd9LSaYedhK1511cL95eCSeM0McUR16etHloWYQccC19CFig3cokJ54u3o/Tfz
r2avW73qTLVo3arhhBgfnpBk3cWvky1wBx0oN1c40hykjU56pxoS0dFaa1tQ0Eq0gmt54p2trCZBqmrlXwRLE+kYm7s4MuttRDH0jx5fxtZ6mgqyBwqlrxsQKQAr4Un
/gZJW3F0c80SDPY0L6GRIXHMKL0mCkL1aVZLHVagFyhZ5pmPR5wNw8Ck5yMsWzLdZwLbe-zhdD0vZQsZ7tV5R3w+Crz16PVULGS0l0051Piq2k3nBNLjC1KKbbYXbgSG0
g5Wj1zL4xZ0FZzj/KUwyt1jCmNB1xaCb1y0f6d59oWrv1F4adqgG0n1X0sPAFc3KhyR3VN1clfHgNgMD+-+pe0PKFVnq9z07Mx0c251rN0MN0d/ZQm3wP61Yc5bj01j
evq2
gokul@debian:~/electrum/testnet/wallets$ file cold_wallet_1
cold_wallet_1: ASCII text, with very long lines, with no line terminators
gokul@debian:~/electrum/testnet/wallets$
```

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.

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

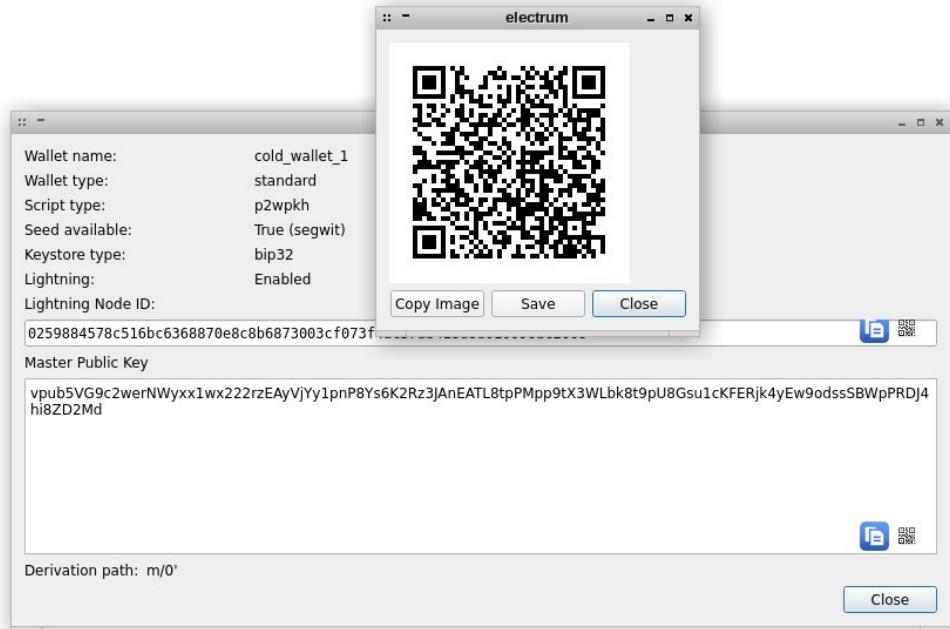


Figure 38 QR code of MPK

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”

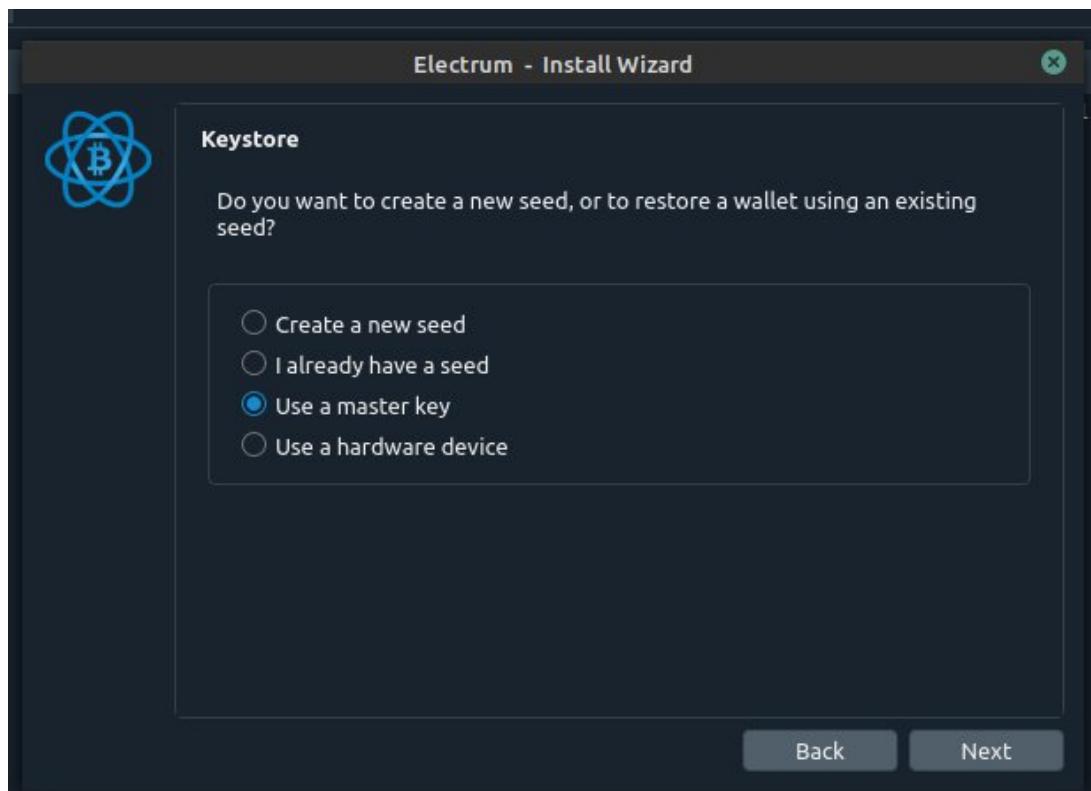


Figure 39 Use a master key for creating a "watch only" wallet version of the offline wallet

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

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

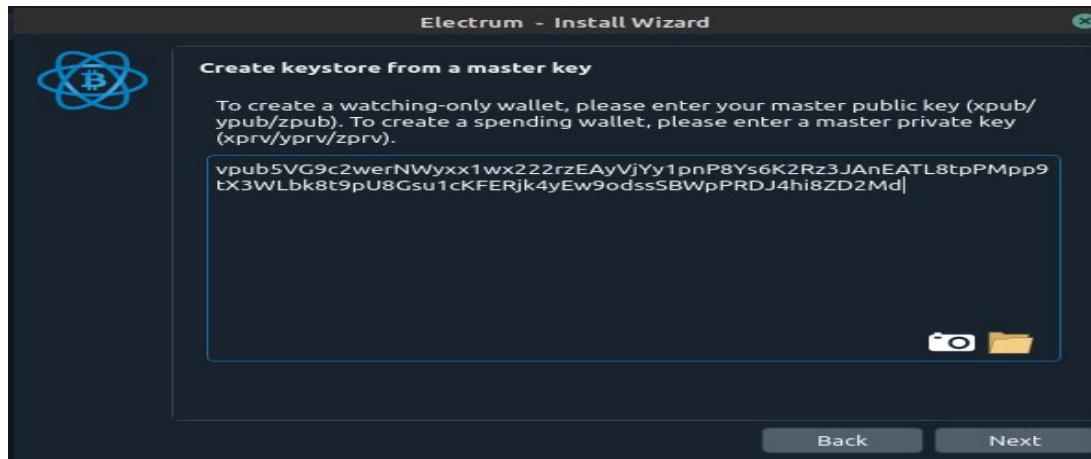


Figure 40 Copy the MPK of offline wallet

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)

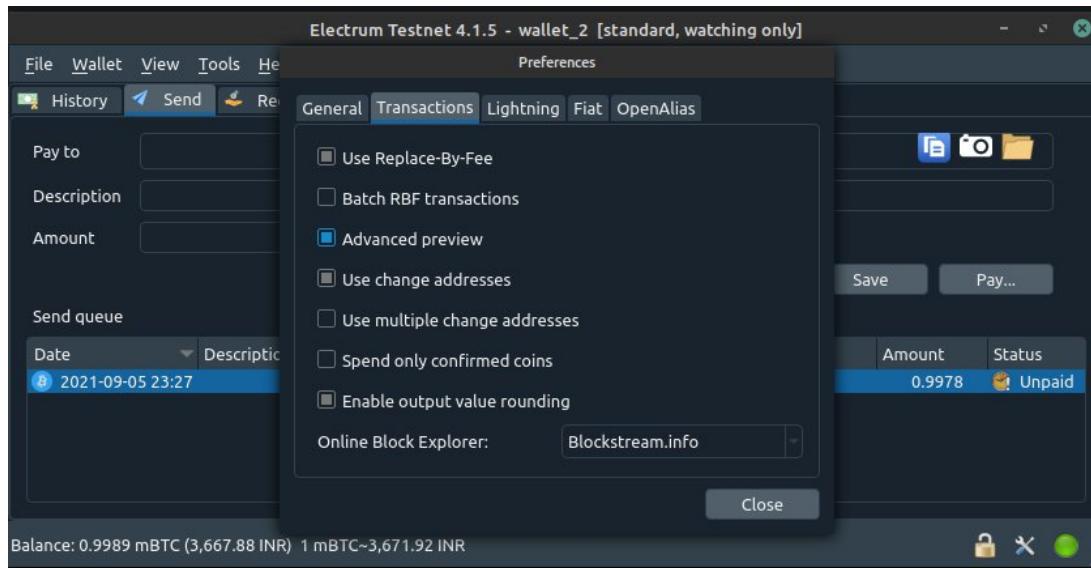


Figure 41 The setting to enable previews is not mentioned in the documentation, but needs to be enabled

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.

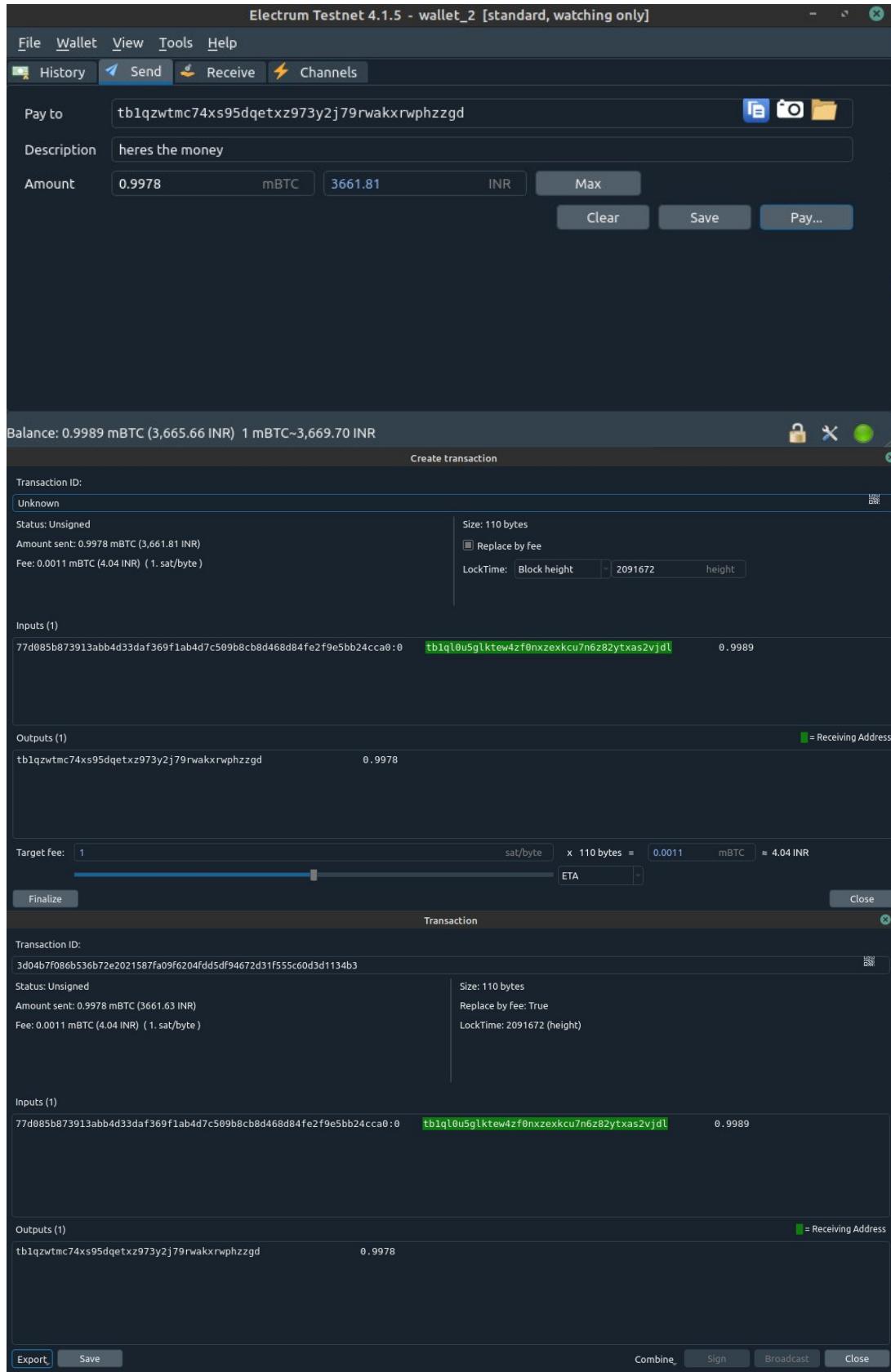


Figure 42 Click on Save, Finalize and Export (export to a psbt file)

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021

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

The screenshot shows a terminal window with two tabs. Tab 1 displays the command `curl http://192.168.43.230:8080 --output partial-transaction.psbt` being run, which outputs a large base64-encoded partial transaction file. Tab 2 shows the file being transferred via curl to an oneshot server at port 8080, with a progress bar indicating 100.0% completion. The transfer summary shows the file name is `wallet_2-3d04b7f0.psbt`, MIM type is `text/plain`, file size is 339 B, start time is 23:44:37.248 IST 5 Sep 2021, duration is 0s, rate is 1.608 MB/s, and client address is 192.168.43.120:45728.

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.

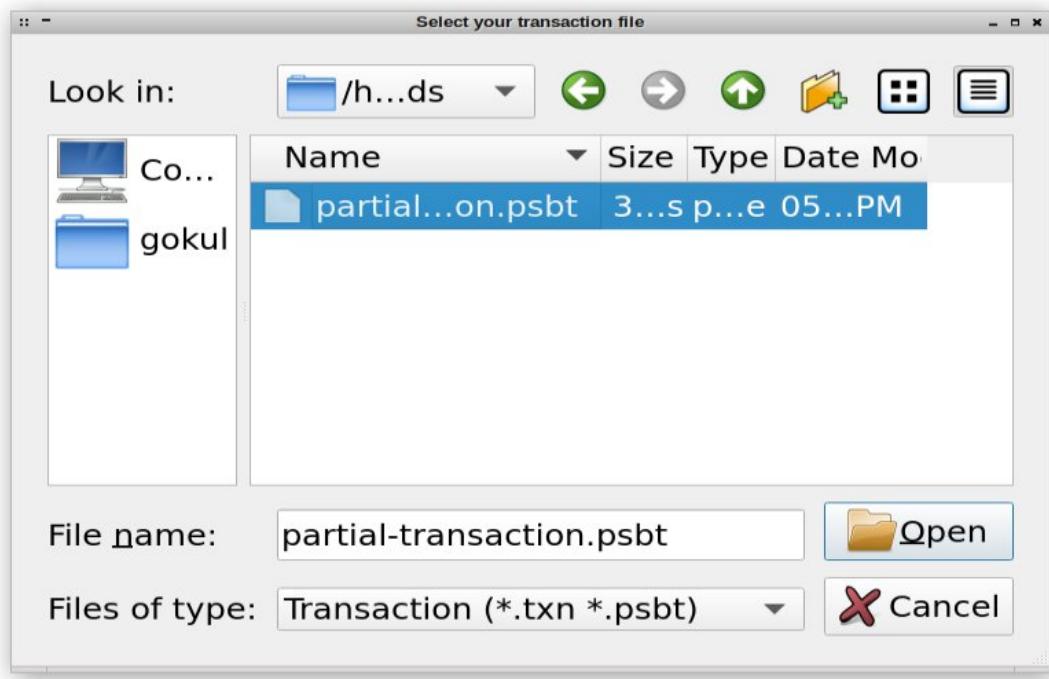


Figure 44 psbt file loaded on to offline wallet

Click on sign and then on the export option and have the saved .tx 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

Transaction ID:

3d04b7f086b536b72e2021587fa09f6204fdd5df94672d31f555c60d3d1134b3

Status: Unsigned Size: 110 bytes
Amount sent: 0.9978 mBTC Replace by fee: True
Fee: 0.0011 mBTC (1. sat/byte) LockTime: 2091672 (height)

Inputs (1)

`77d085b873913abb4d33daf369f1ab4d7c509b8cb8d468d84fe2f9e5bb24cca0:0
tb1ql0u5glktew4zf0nxzexku7n6z82ytxas2vndl` 0.9989

Outputs (1)

`tb1qzwmc74xs95dqetxz973y2j79rwakxrwpzzgd` 0.9978

Export Save Combine Sign Broadcast Close

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

Transaction	
77d085b873913abb4d33daf369f1ab4d7c509b8cb8d468d84fe2f9e5bb24cca0	
STATUS	24 Confirmations
INCLUDED IN BLOCK	0000000000000a1149fb3c747f79b4768640b04b569ff1192c300ca6159983
BLOCK HEIGHT	2091750
BLOCK TIMESTAMP	2021-09-05 22:51:30 GMT +5.5
TRANSACTION FEES	0.0000011 tBTC (1.0 sat/vB)
SIZE	191 B
VIRTUAL SIZE	110 vB
WEIGHT UNITS	437 WU
VERSION	2
LOCK TIME	2091749
REPLACE BY FEE	Opted in
SEGWIT FEE SAVINGS	This transaction saved 42% on fees by upgrading to native SegWit-Bech32
PRIVACY ANALYSIS	Possibly self-transfer ↗
View raw transaction	
77d085b873913abb4d33daf369f1ab4d7c509b8cb8d468d84fe2f9e5bb24cca0	
DETAILS +	

Figure 46 Transaction as seem on the explorer

Assignment 01 - Cryptowallet Exploration

Gokul G [CB.EN.P2CYS20017]

10/09/2021