A bitcoin lab.

- Practice receiving and sending bitcoin transactions
- Practice signing and verifying messages

Section Download, Configuration, and Installation:

Bitcoin terms:

Bitcoin has three types of network
Bitcoin Mainnet: live transactions take place (peer to peer)
Bitcoin Testnet: provides a test environment (peer to peer)
Bitcoin Regnet: for testing bitcoin applications (no peers)

Let's first use a test environment. You need to create a Bitcoin configuration file inside the Bitcoin core directory since the configuration file is not automatically created, and then we download and install the bitcoin application.
This configuration file helps to access the network and to provide directions to the Bitcoin core application. Bitcoin looks for the configuration file in the bitcoin data directory. The configuration tells the Bitcoin core to load the application into the Testnet or Regnet network rather than into the main network.

To create a configuration file **bitcoin.conf** file**.** The configuration file consists of `option=value` entries. One per line, and remove any whitespaces. A value of the given option is required e.g, `testnet=1`, or `regtest=1`.

Windows:

```
C:\Users\username\AppData\Roaming\Bitcoin\bitcoin.conf
Linux:
/home/username/.bitcoin/bitcoin.conf
macOS:
/Users/username/Library/Application\
Support/Bitcoin/bitcoin.conf
```

Click on the link below to download and install a bitcoin application
https://bitcoin.org/en/download

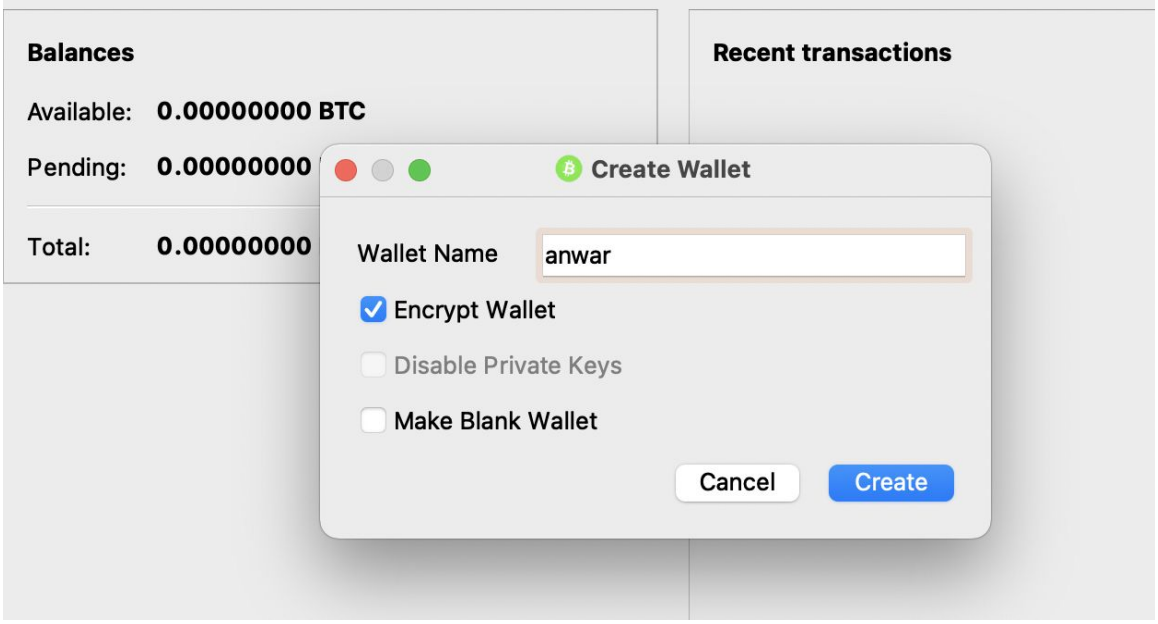
After you install the application and run it, you will see Bitcoin core [test].

Source

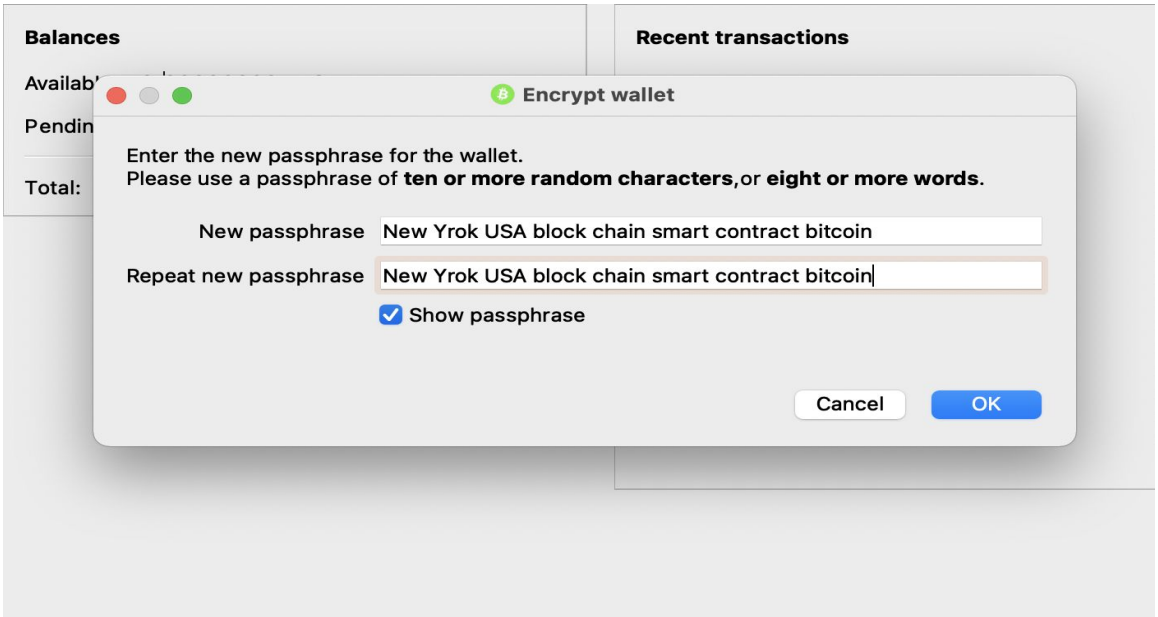
Section 2) Create Wallet and Receive bitcoin

Create a new wallet from the file menu choose create a wallet, and then write the name of the wallet :

File menu ➞ Create Wallet

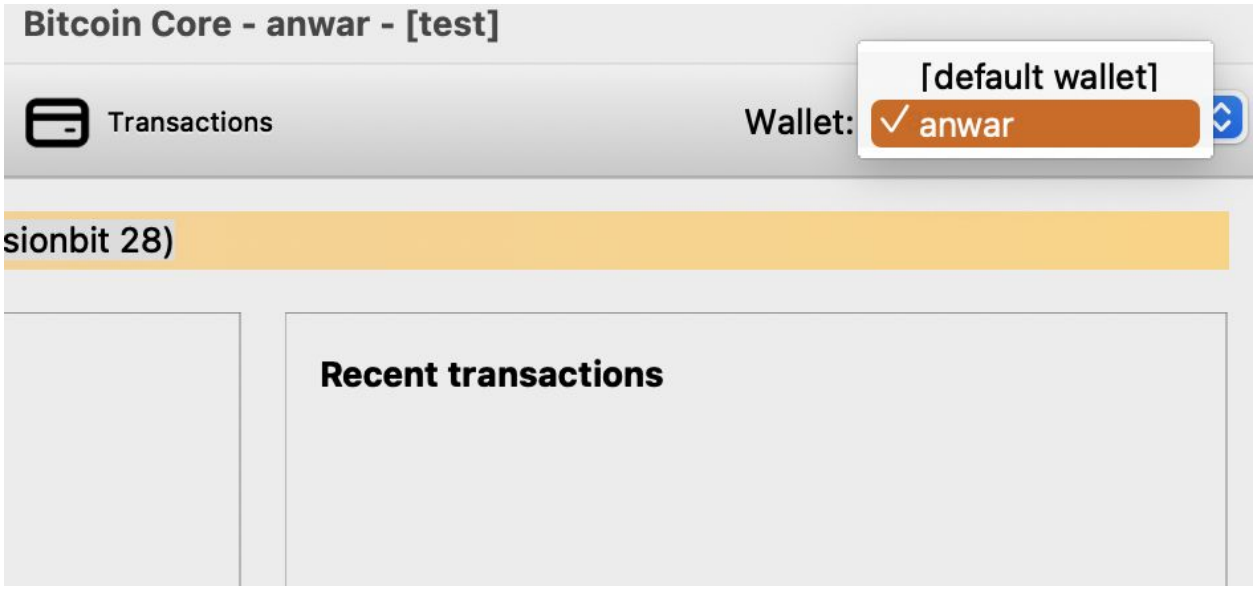


Encryption wallet allows you to enter a new passphrase for the wallet.



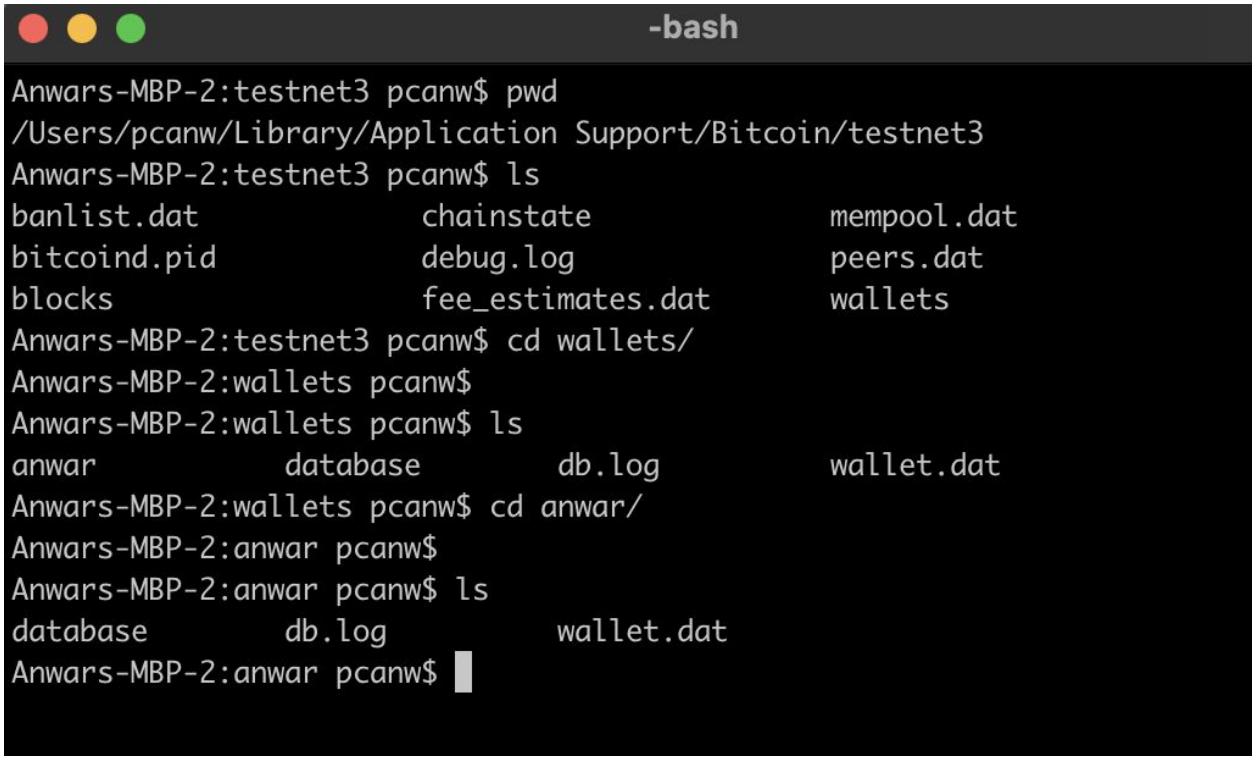
Warning:
The passphrase can be any word or sentence. It is case-sensitive. You MUST keep your passphrase in a safe place and if you lose it or forget it, you will LOSE your wallet and your coins. When you encrusted your wallet that does not mean that your coin is protected from being stolen by malware. It is better to keep your wallet in a safe place.

To backup your wallet you need to select the wallet name on the wallet selection:



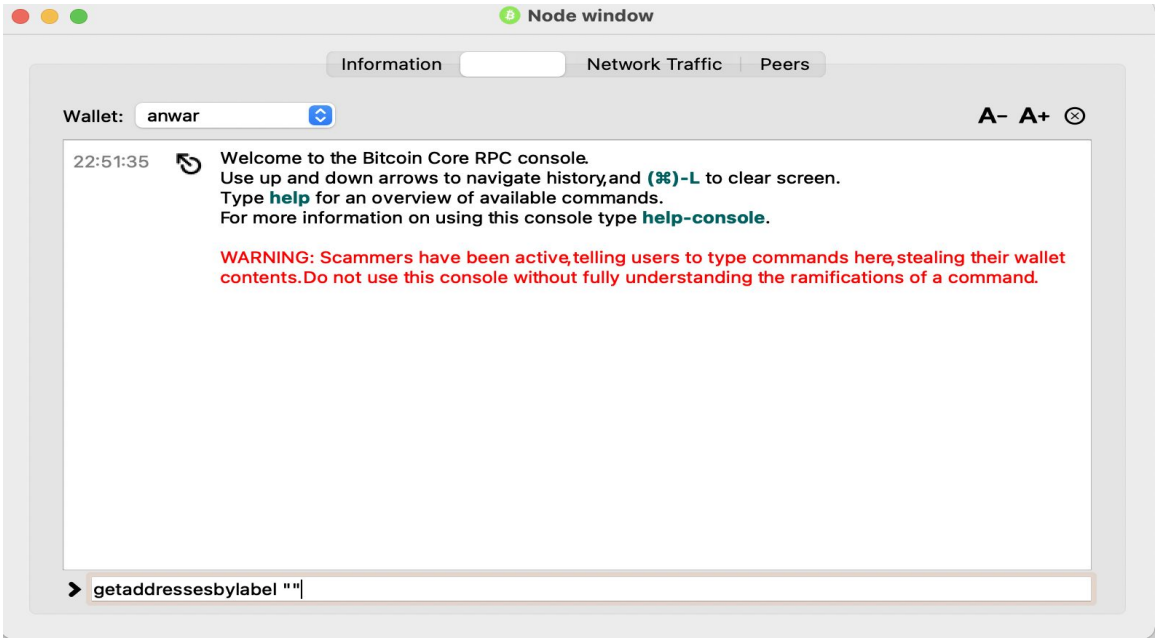
And then from the file menu ➞ Backup Wallet.
Also if you go to the bitcoin directory and since you use the testnet network you can find block and wallet files in /Bitcoin/testnet3/ directory.

So if you want to backup `wallet.dat` file since by default contains the private keys, you can find it under the wallets directory inside **the wallet name directory**.
Warning: if you lose or delete your wallet file, you lose your coins.



To receive bitcoin, you need your public address or you could create a new receiving address. You need to use the console in the application so you can write the command line which is the interface for the bitcoin application.

From Window menu ➙ Console, you will see this windows below.



You MUST select the wallet name then write **getaddressesbylabel ""** to show the default receiving wallet address.

You can create a new receiving address by:

getnewaddress ("label" "address_type")

Returns a new Bitcoin address for receiving payments.

```
Arguments:
1. Label: (string, optional, default="") The label name for the address to be
linked to. It can also be set to the empty string "" to represent the default
label. The label does not need to exist, it will be created if there is no
label by the given name.
2. Address_type: (string, optional, default=set by -addresstype) The address
type to use. Options are "legacy", "p2sh-segwit", and "bech32".
```

legacy is the P2PKH address type. The **P2PKH** stands for “Pay to Public Key Hash”. The Public Key Hash is one of many formats of the Bitcoin address.

Example:

getnewaddress "anwar" "legacy"

To return the new receiving wallet address:

```
getaddressesbylabel "label"

Returns the list of addresses assigned the specified label.
```

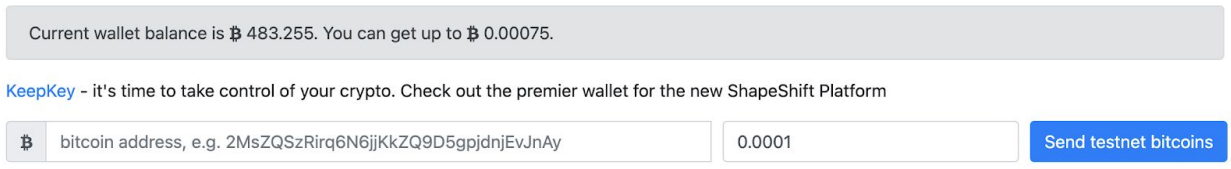
Example:

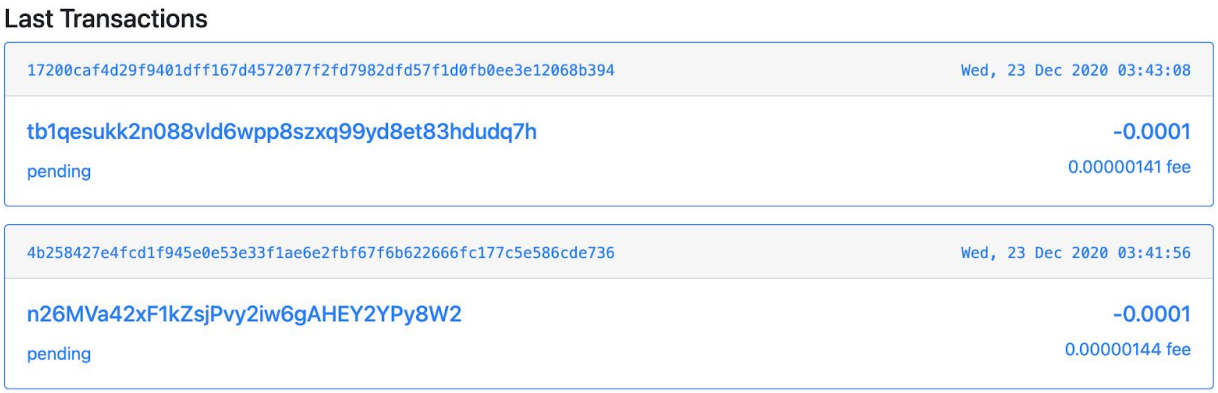
getaddressesbylabel "anwar"

```
22:51:59  getaddressesbylabel ""
22:51:59  {
            "tb1qesukk2n088vld6wpp8szxq99yd8et83hdudq7h": {
              "purpose": "receive"
            }
          }
22:56:00  getaddressesbylabel "anwar"
22:56:00  {
            "mt7JFYmEnijJ5myZ97MP4s4wr3CapXNU7h": {
              "purpose": "receive"
            },
            "n26MVa42xF1kZsjPvy2iw6gAHEY2YPy8W2": {
              "purpose": "receive"
            }
          }
```

In this example we have those addresses above.

To receive a test Bitcoin to yourself, you can use this website and put any one of the addresses
https://bitcoinfaucet.uo1.net/

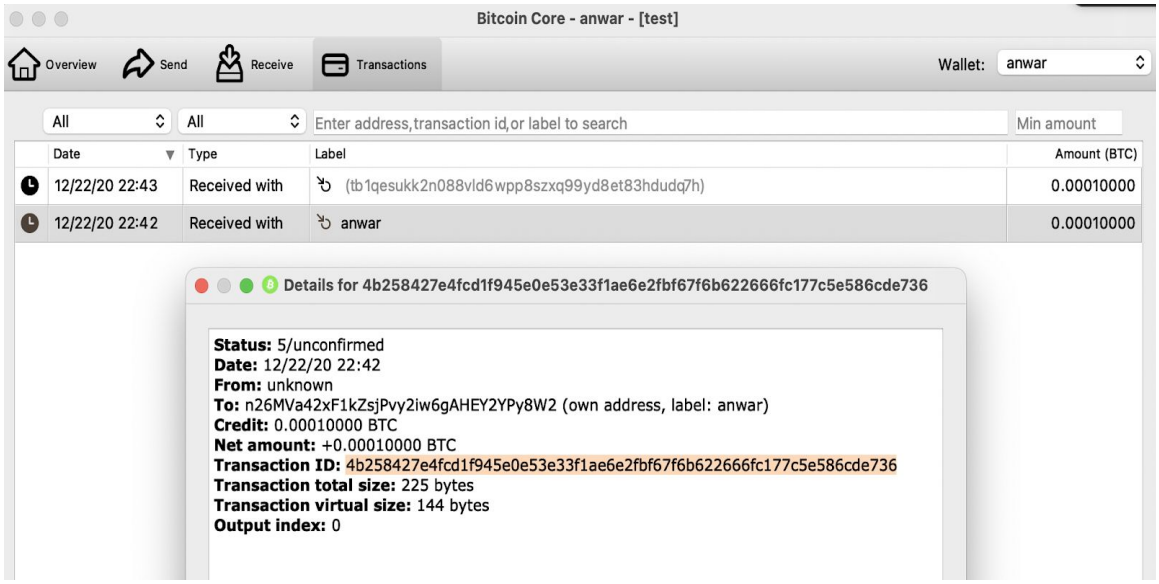


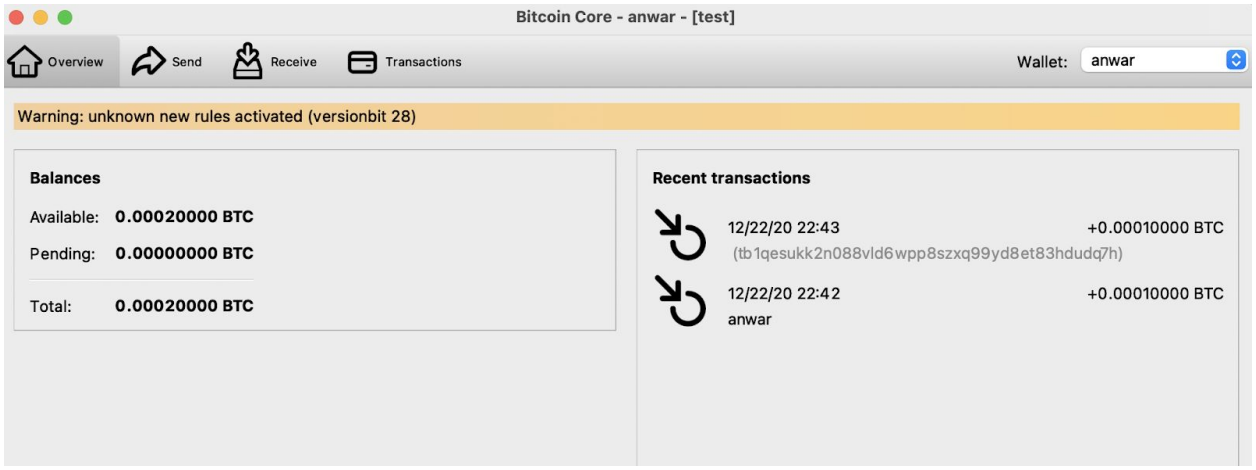


As you can see the transaction is pending on the memory pool “unconfirmed transaction” so we need to wait until the transactions get confirmed.

Pending transactions or unconfirmed transactions means these the transactions have not been included in a block so they have not been confirmed.

Transaction Details:
You can see transaction Id on the website and the bitcoin application:
4b258427e4fcd1f945e0e53e33f1ae6e2fbf67f6b622666fc177c5e586cde736





Nice! We received our test bitcoin.

You can see the details of the transaction on the bitcoin test explore. The advanced details show you more about block hash, index, size, and other details. You need to scroll further down so you can see more detail. It gives a breakdown of the transaction input and output, and you can navigate the network and get an idea of the structure of data available on the blockchain.

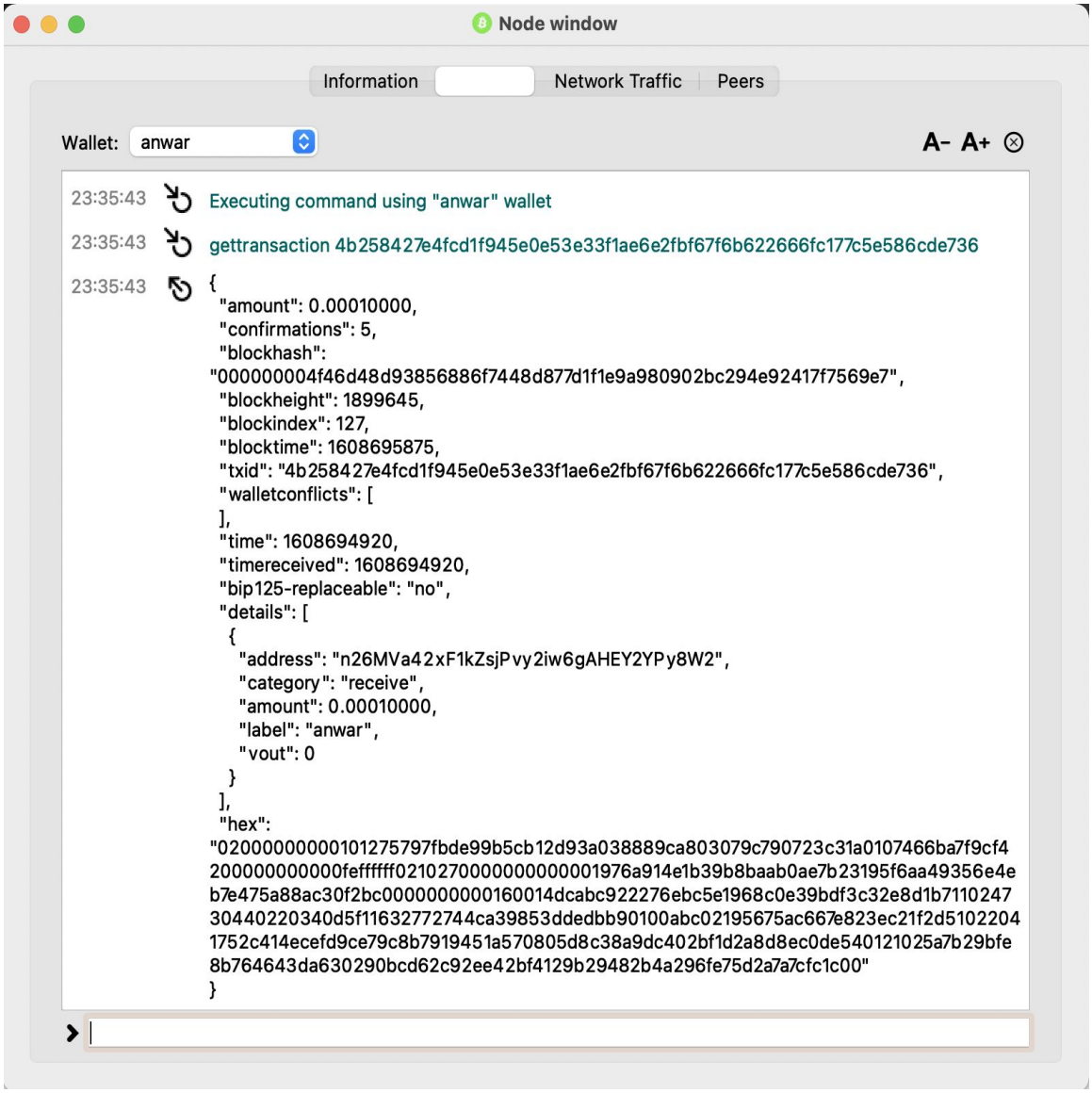
https://live.blockcypher.com/btc-testnet/tx/4b258427e4fcd1f945e0e53e33f1ae6e2fbf67f6b622666fc177c5e586cde736/

Also you can get detailed information about a transaction in the console

```
                          gettransaction "txid"
Get detailed information about in-wallet transaction <txid>
txid                      (string, required) The transaction id
```

Example:

gettransaction 4b258427e4fcd1f945e0e53e33f1ae6e2fbf67f6b622666fc177c5e586cde736

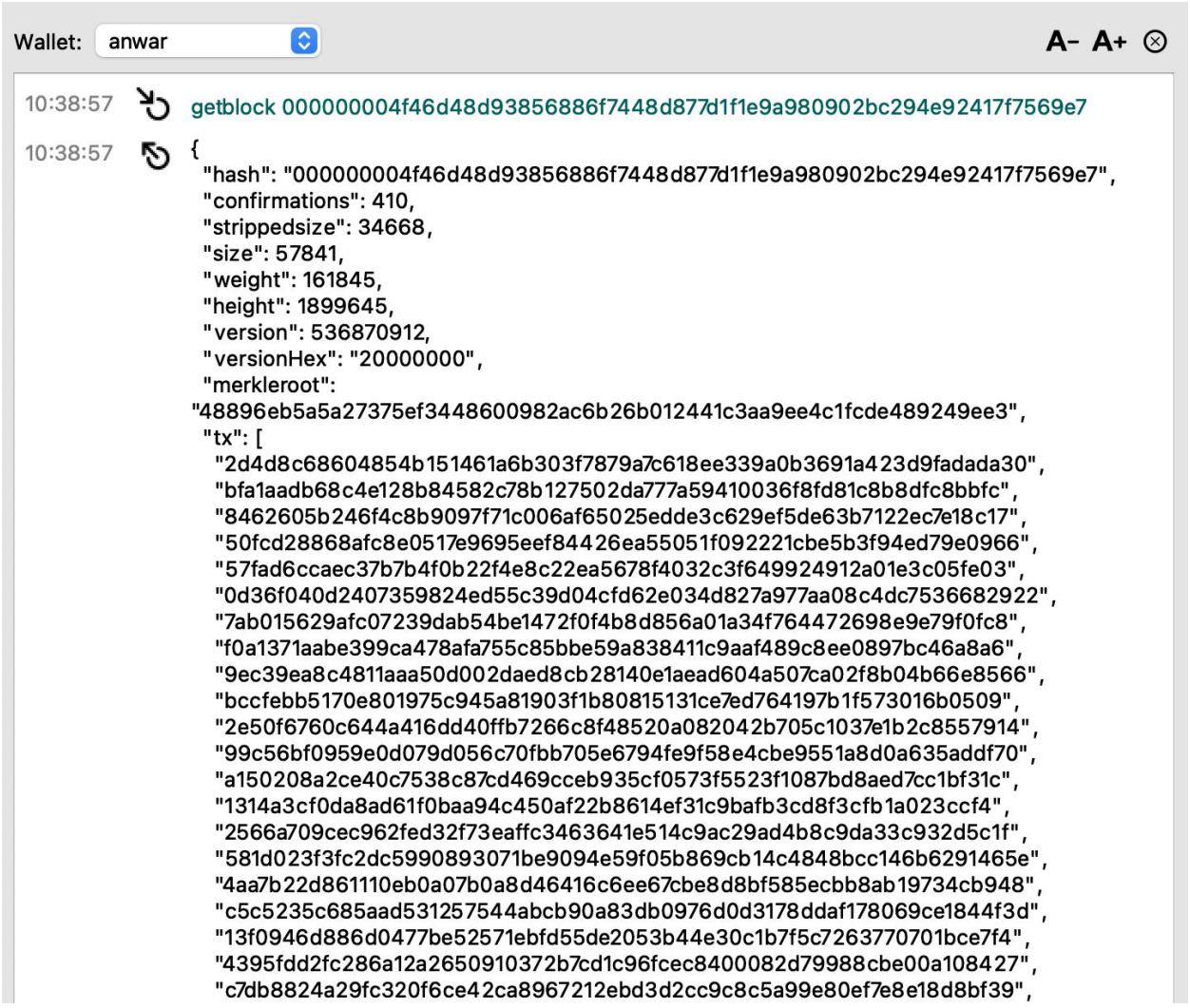


Note: make sure to select your wallet name.

Also you can get information about the block if you use the `getblock` command line.

Copy the `blockhash` value to get information about the block

getblock 000000004f46d48d93856886f7448d877d1f1e9a980902bc294e92417f7569e7



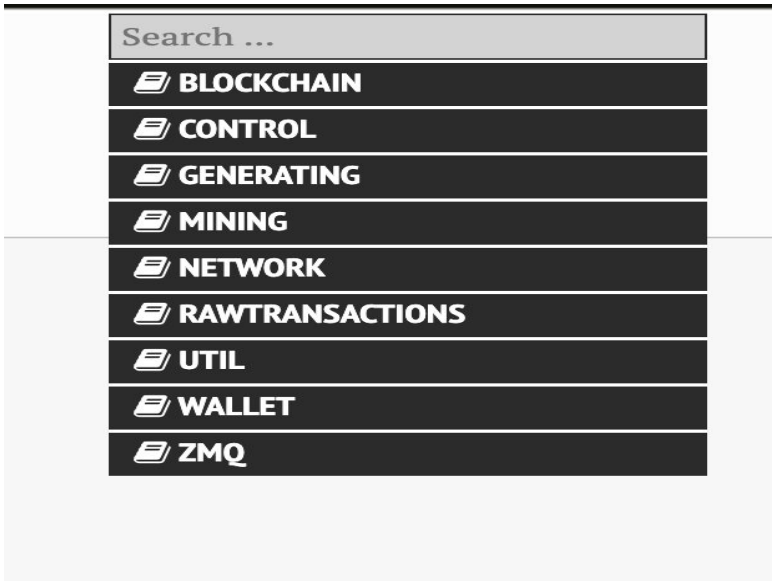
As you can see there are 410 transactions that were confirmed in this block.

In order to receive the coins, the Bitcoin core application has to be synchronized completely with the Tesnet network. It will be delayed until the network is synchronized.

All bitcoin core application command lines can be found in this website:

https://bitcoincore.org/en/doc/0.17.0/rpc/

We only used wallets and transaction information command lines, there are more such as general Information (getblockchaininfo, getmininginfo, getpeerinfo) and block information (getblockcount, getblock hash, getblockhash index)



Signing and Verifying message:

The digital signature is used for authenticating transactions by hashing a value of a message. When someone sent this message, that means this person had to be in possession of the private key in order to send the message, and the result anyone can verify the transaction.

The signing algorithm (s) ← Sign(message, private key) takes a message (m) and the private key (piv) and outputs a signature (s). This algorithm has to be processed on the client side because the users must use their (piv).

From the console, we need to create a P2PKH address to sign a message.
To create a new P2PKH address type

getnewaddress "anwar" "legacy"

To return your P2PKH address

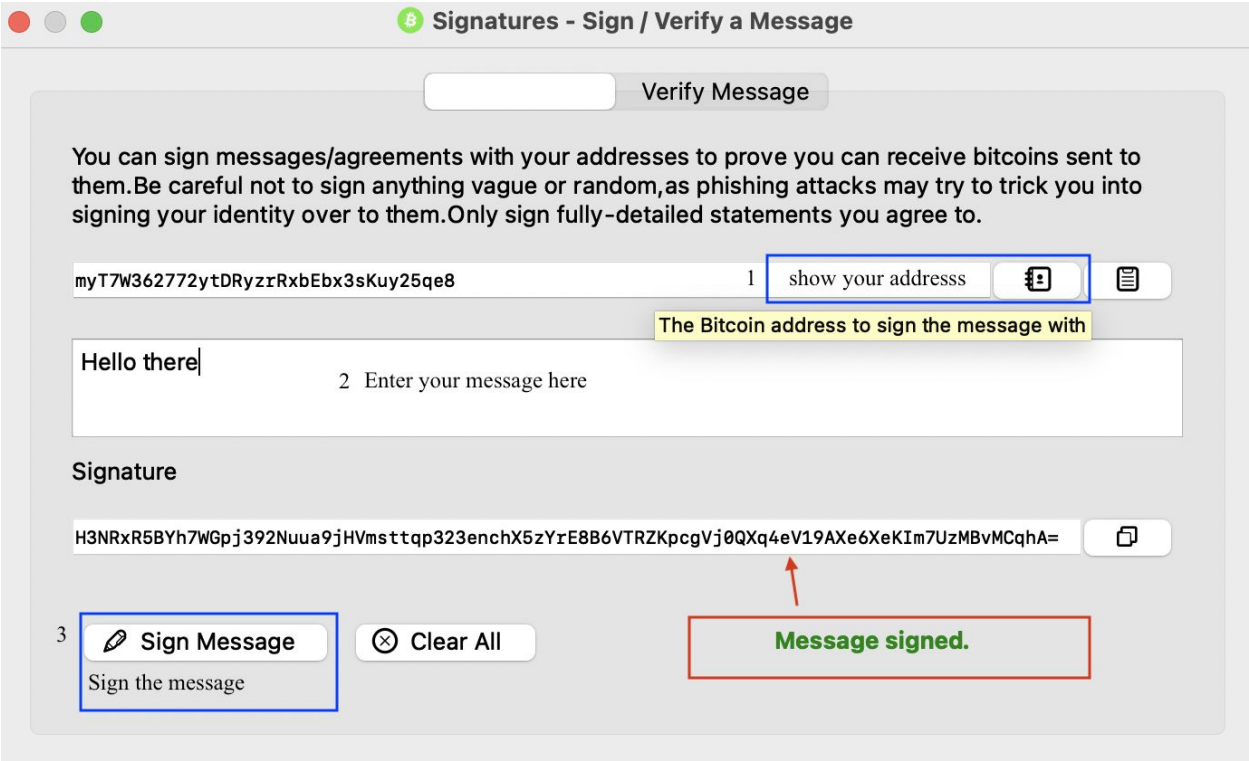
getaddressesbylabel "anwar"

Note: Make sure you write your wallet name.

Sign message:

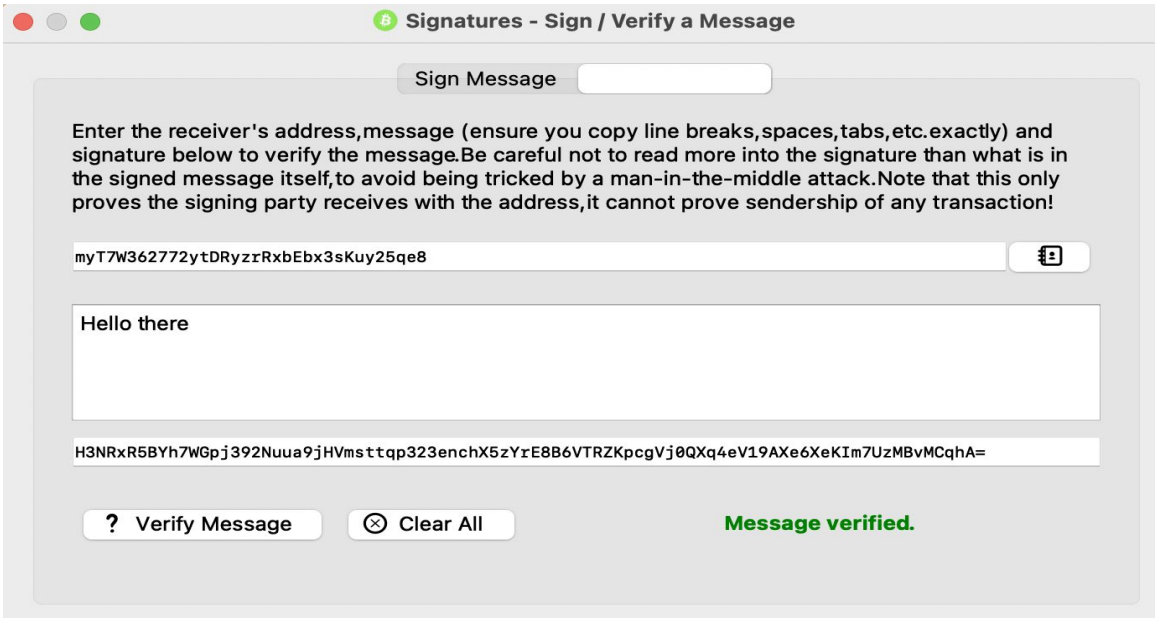
First you need to input the address and you can select any of your addresses from the contact book o

This is the P2PKH address:
`myT7W362772ytDRyzrRxbEbx3sKuy25qe8`
Message: Hello there
Signature:`H3NRxR5BYh7WGpj392Nuua9jHVmsttqp323enchX5zYrE8B6VTRZKpcgVj0QXq4eV19AXe6XeKIm7UzMBvMCqhA=`

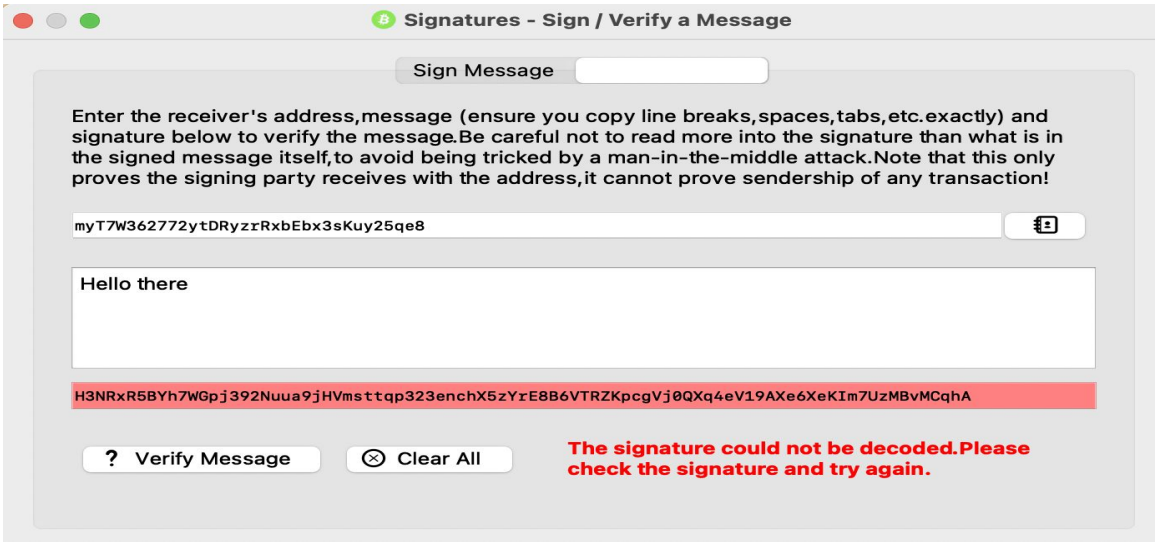


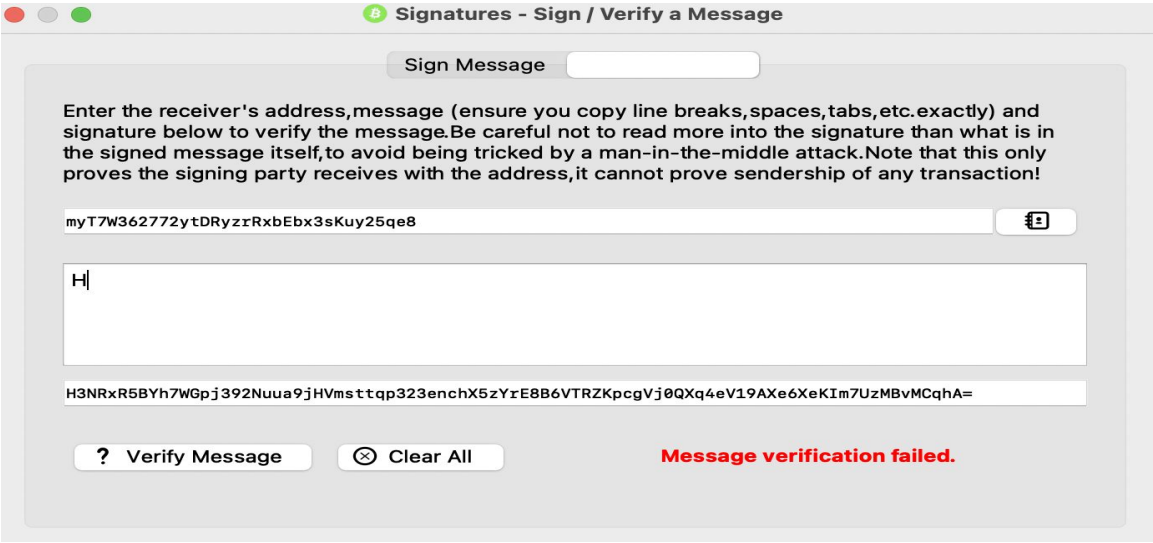
Verify a message:

The signature verification algorithm validity ← signVer(m, s) requires a public key (pub), a message (m), and a signature (s) and returns true if hash(m) corresponds with the signature. A string transaction is true if signVer(piv, m, Sign(pub, m)) = true] = true, and otherwise the output is invalid.



Look if we changed the signature that was given when the message was signed we received an error. Also if we changed the message, we got an error too.





Homework:

- Create your own wallet and send bitcoin test to your address and then send **0.0005 - 0.0001** bitcoin to this address :

  Also you need to find your transaction using this website https://live.blockcypher.com and share the link of your transaction.

Note: when you don't need the coins anymore please send them back: `tb1qm5tfegjevj27yvvna9elym9lnzcf0zraxgl8z2`

https://bitcoinfaucet.uo1.net