Intro to Bitcoin Command Line - Using Bitcoin!

Generate a wallet

Before we can do anything, you need to generate a wallet. Run the command below to create your first wallet.

```
$ bitcoin-cli createwallet my_wallet
```

Now that you have a wallet, you will need to get an address from your wallet. The wallet in this case is a hierarchical deterministic (HD) wallet which you can find out more about by reading Bitcoin Improvement Proposal (BIP) 32 https://github.com/bitcoin/bips/blob/master/bip-0032.me diawiki.

You can always get a new address by running this command.

```
$ bitcoin-cli getnewaddress "" legacy
```

Note: All bips proposed and implemented can be viewed at https://github.com/bitcoin/bips. These are often the best source of information when trying to learn about or understand a feature of Bitcoin without having to read the source code.

A HD wallet is a set of addresses which is generated from a single master seed using a hierarchical deterministic process. So a wallet can have many types of coins, such as Bitcoin, Bitcoin Testnet, Litecoin, et cetera. A wallet can also organize the addresses for a specific coin type into accounts. Where the account will have many addresses for that coin type. This may all seem confusing but you can learn more at https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki. But for now you can simply think of wallets as having many addresses.

Before we can use our newly generated wallet, we will need to get an address from our wallet. To do this you will need to get a receive address from your wallet. This can be done by using the `getnewaddress` command, but to make things simpler let's save the result as a variable so we can access it easily in the future.

```
unset NEW_ADDRESS_1
NEW_ADDRESS_1=$(bitcoin-cli getnewaddress "" legacy)
```

These commands clear the NEW_ADDRESS_1 variable, then fills it with the results of the bitcoin-cli getnewaddress command. The flag legacy is used here because some of the features of bitcoin-cli are not yet supported for "new" seg-wit addresses. You can tell that this is a legacy address because it starts with a n or m rather than a 2.

You can then use your shell's echo command to look at your (new) address:

```
echo $NEW_ADDRESS_1
2Mz6bRRUV5kfDUv87FFaPAgqMECDuMQn9ta
```

Get some 'testnet' Bitcoin

The following websites will send you some testnet bitcoin so that you can play around and learn how to use Bitcoin.

https://coinfaucet.eu/en/btc-testnet/
https://testnet-faucet.mempool.co/

Enter your bitcoin address and fill out the captcha. Once it shows that it is sending some testnet bitcoin you can check the status of the transaction at:

https://live.blockcypher.com/

Select "Bitcoin Testnet" in the search menu and enter your Bitcoin address.

You should see a test faucet transaction sending money to your wallet. If not, wait for a few minutes and refresh. The reason that you will potentially need to wait is that your transaction must be included in a block. Bitcoin blocks are found using a stochastic process of mining where blocks are typically found every 10 minutes. There can be a significant deviation in any given block time ranging from 1 minute to 60 minutes is not atypical. To check on the progress of the Bitcoin testnet blocks you can go to https://live.blockcypher.com/btc-testnet/ and see when the last block was found.

Explore your wallet while you wait

Now that you have a wallet you will need to back it up. Future assignments will require you to "turn-in" your work by executing a transaction on the blockchain from one of the addresses in this account. Therefore, you will need to back-up your wallet. You can do this by running:

```
bc backupwallet backup.dat
```

Locate this file on your computer and save it in a safe location. If in the future you need to restore your wallet, you can do so by importing the wallet using:

```
bc importwallet backup.dat
```

To better understand what is going on in the background we can dump the wallet in plaintext format. To do this run:

```
bc dumpwallet mywallet.txt
```

Open the resulting file in your text editor. You will see something like the following:

```
cQ4qpkvWCRvZaxGgbitZs5RJCBGoL9g1mqQmsssRw93F3WbybPqY 2019-01-20T04:37:24Z reserve=1#
addr=2N1cjToNQPX4JHQTJNLz94TFa8bcEseoPuQhdkeypath=m/0'/0'/195'
cRCvHdRZaEfjtgC15s7AV7nKwVxpUDumRJPeTfawwLEmhPVwCyHc 2019-01-20T04:37:24Z reserve=1#
addr=2NA18oui36UXLdeyzViDCtZxZSvKnRK6782hdkeypath=m/0'/0'/691'
cTrEobtoAuq3Zw6NbR1n6xvPSoCY5Bqu7FHHqmmbf3Yzc1zbEP2T 2019-01-20T04:37:24Z reserve=1#
addr=2NBbat4jgaLmXYnPPvdXhw1Gmanm66FWnJihdkeypath=m/0'/0'/672'
cTGS9GLiSmmFv6uUAGYhmAyBC5tfNv2uHQT4SDJMgD6mChb3dvD42019-01-20T04:37:24Z
reserve=1#addr=2N5TfhrXWVVg6g4GMfG99XBt84Fx2ZPhtM8hdkeypath=m/0'/0'/475'
cPTDBf9G73476d1k2BKRQptKUnF1NHeKcwqUeK66BNTHNnZty1o52019-01-20T04:37:24Z
reserve=1#addr=2N8oTz3XKpp7QVuSGiArxWYWF9PAYRn3Fcphdkeypath=m/0'/0'/430'
cTHDfPzfN6TKXFtXFnmCqEtq4erTR7t6n1VmnUJX1PJHcip3D2Tu2019-01-20T04:37:24Z
reserve=1#addr=2N1M9KJu6AtU62tEVstytZcZMYTrkYDDxoMhdkeypath=m/0'/0'/273'
cV1cYg7nYbyyAVWpiaEgXTT2XCGsDxiMVig9bdTbipFhGUHX6BZk 2019-01-20T04:37:24Z reserve=1#
addr=2NCHHAc7ykgpvsbVdsgy9jRPX6eth4nV65Khdkeypath=m/0'/0'/712'
cPsbCqpytvwtqe4YA1hJkopiK57bCCCcSBqRdiJJRxEuXcwvZ8hr 2019-01-20T04:37:24Z reserve=1#
addr=2MzUs5uaMXM8ye8WuovqBFW8h8v8p3rdHHghdkeypath=m/0'/0'/729'
cPULsxLpi6vHoxayNMuiGp9yRcCksftEPbWhojCFF3mpXKtdkjSU 2019-01-20T04:37:24Z reserve=1#
addr=2N1rTT1Uh8ChfvFF7QEAPUvwBZaQBaiFM46hdkeypath=m/0'/0'/307'
```

Go ahead and search for your address that you used to get your testnet bitcoin. If you already forget it you can find it by `echo NEW_ADDRESS_1`. You should find it and at the end of the line you will see that `hdkeypath=m/0'/0'/0'`. This is the first address in your HD wallet tree. The value to the left, (e.g. is the private key corresponding to this address which needs to be used to move Bitcoin from this address.

Let's get a new address and see what happens.

```
unset NEW_ADDRESS_2
NEW_ADDRESS_2=$(bitcoin-cli getnewaddress "" legacy)
```

```
echo $NEW_ADDRESS_2
mhpt1x5Ro8SSMAQrkXwCj4F6sZd1xL9nsT
```

If you search for this value in your wallet file you should find that it appears in the line with `hdkeypath=m/0'/0'/1'`. This is the second address in your HD wallet. You can use address in Bitcoin multiple times; however privacy is better preserved if addresses are not reused. This is one of the reasons for an HD wallet. If you look at the top of the file you will see that there is similar to:

```
# extended private masterkey:
tprv8ZgxMBicQKsPdNtUXyQ3x2h9AJpp1EDayZsSHFj52HXPjRdehHDqADx7kJ8ZNf7AhbchZXAj2CRc
zMYMVXxXeSNPkwWYhjESRRDR98TSwnN
```

The beauty of HD wallets is that as long as you have this line, you can re-derive the rest of the file by changing the `hdkeypath`. This gives Bitcoin users the ability to manage a single secret but derive an infinite number of addresses which they can use easily. I will also mention here, that if you have actual bitcoin in your wallet, you do NOT want to **share this file or your masterkey with anyone**. The ownership of bitcoin is the possession of this private key and the ability to sign transactions with it. Therefore, if anyone else gets your private keys, or your wallet.dat file, they have your Bitcoin.

Signing a message

Assignment Deliverable 1: You will need to sign a message, your UT ID, and submit the message and the account to your TA. You will use this same account for the deliverable 2.

The heart of Bitcoin, blockchain, and any cryptocurrency is signing messages with private keys that can then be verified by using a public key. A bitcoin address is a representation of a public key and a bitcoin transaction is like a check that tells the network to move money from one account to the other. The check is signed using an address's corresponding private key, and if the signature is valid, the miners update the ledger as long as the transaction meets the rules of consensus.

In addition to being able to sign Bitcoin messages, you can also use the same keypairs to sign any arbitrary message. This can be done using the following:

```
$ bc signmessage "mhpt1x5Ro8SSMAQrkXwCj4F6sZd1xL9nsT" "Hello, Bitcoin!"
H9v014McOc0+ESiyWMWJ2lBcGi+Vd+O0/IqSR0mdt15IEK7z158PKkD8JeRe5j4n29+Gyu2m5F3qPWwH QuAPS90=
```

The veracity of the message can be checked by using: `verifymessage "address" "signeddata" "message"`.

```
$ bc verifymessage "mhpt1x5Ro8SSMAQrkXwCj4F6sZd1xL9nsT"
"H9v014McOc0+ESiyWMWJ2lBcGi+Vd+O0/IqSR0mdt15IEK7z158PKkD8JeRe5j4n29+Gyu2m5F3qPWw HQuAPS90=" "Hello,
Bitcoin!"
true
```

Back to your transaction

Once you see the transaction on the testnet, go back to the command line and enter

```
bc getwalletinfo
```

you should see a result similar to the following:


```
{
   "walletname": "wallet.dat",
   "walletversion": 159900,
   "balance": 0.08203474,
   "unconfirmed_balance": 0.00000000,
   "immature_balance": 0.00000000,
   "txcount": 1,
   "keypoololdest": 1547959045,
   "keypoolsize": 999,
   "keypoolsize_hd_internal": 1000,
   "paytxfee": 0.00000000,
   "hdmasterkeyid": "3aee71d488da5d8547e22087f7d577136d7f4a80"
}
```


You will note that the "balance" field is no longer 0 because you have received testnet Bitcoins from the faucet. You can see that your wallet currently has 1 transaction and has a keypool of 999 keys. To learn more about the wallet. Try out these commands:

```
bc listtransactions

bc listunspent

bc getrawtransaction

bc decoderawtransaction
```

This will tell you information about the transactions, your unspent transaction outputs (UTXOs) as well as letting you see the raw transaction data.

Assignment Deliverable 2: Using the commands above, find your transaction ID and then use that to find the hex of your transaction as well as the JSON interpreted output of the transaction. You will submit the transaction ID, the hex, and the JSON output.

Making a simple transaction

Now that you have your wallet setup and you have some testnet Bitcoin lets go ahead and make a simple transaction. For this transaction, you will be sending money to your TAs using this address: `mg51NavnAmVsddKHQj4zSJVvM5wZ9RVp3h`.

You can send money simply by:

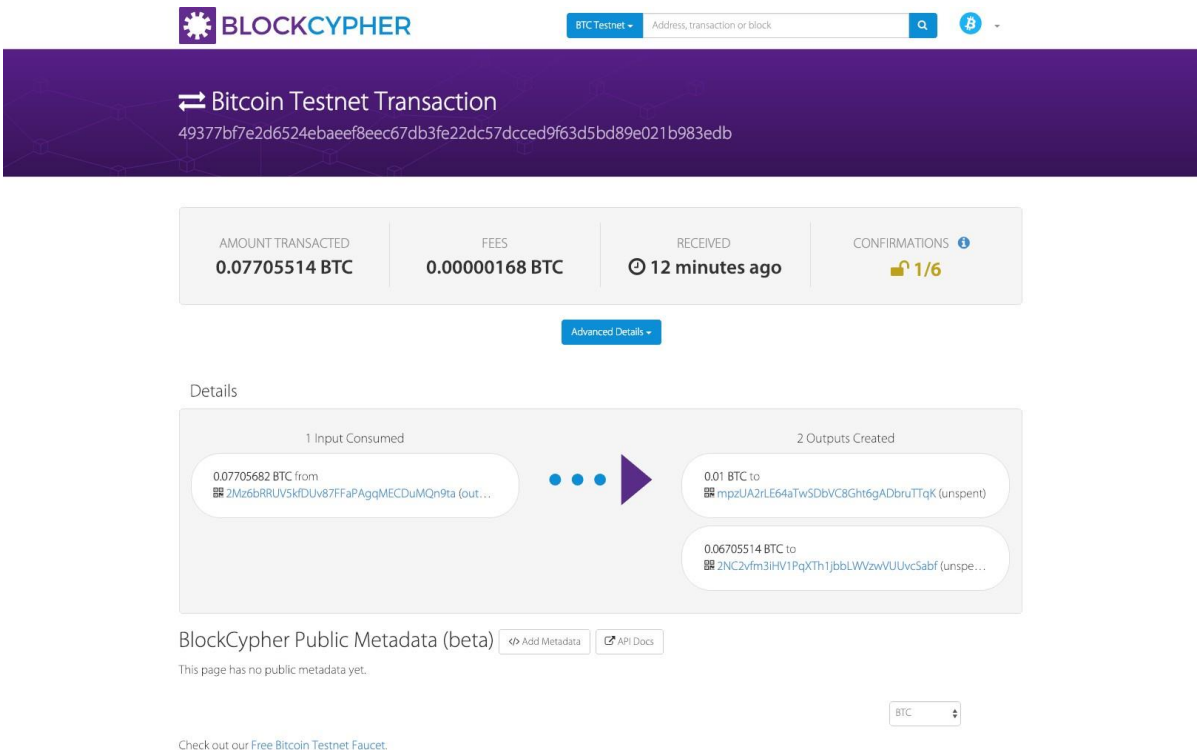
```
$ bc sendtoaddress mg51NavnAmVsddKHQj4zSJVvM5wZ9RVp3h 0.01
49377bf7e2d6524ebaeef8eec67db3fe22dc57dcced9f63d5bd89e021b983edb
```

The result is the transaction ID. You can learn more about the transaction ID by using:

```
  $ bc gettransaction 49377bf7e2d6524ebaeef8eec67db3fe22dc57dcced9f63d5bd89e021b983edb
{
   "amount": 0.00000000,
   "fee": -0.00000168,
   "confirmations": 0, "trusted":
   true,
   "txid": "49377bf7e2d6524ebaeef8eec67db3fe22dc57dcced9f63d5bd89e021b983edb", "walletconflicts": [
   ],
   "time": 1548631273,
   "timereceived": 1548631273, "bip125-
   replaceable":"no", "details": [
      {
         "account": "",
         "address": "mpzUA2rLE64aTwSDbVC8Ght6gADbruTTqK", "category": "send",
         "amount": -0.01000000,
         "label": "", "vout": 0,
         "fee": -0.00000168,
         "abandoned": false
      },
      {
         "account": "",
         "address": "mpzUA2rLE64aTwSDbVC8Ght6gADbruTTqK", "category": "receive",
         "amount": 0.01000000, "label": "",
         "vout": 0
```

```
        }
    ],
    "hex": "02000000000101611fbe48bfd8ec6b8c0cbdb66432486a7c9430dda1062af8b657615bc0ffcfa50
00000001716001 4467b8e642ef4659cc5c9f2dbeed51696ee98b458feffffff0240420f000000000
01976a91467ed4a521f8a3d32a373b7d33ea9b942fb215e3288ac6a5166000000000017a914ce16f
0dc82a233f433063018057c68be7a1dc1918702483045022100f6b9c8ae54aca7e9df3d6ae17e097
1e2ee7535bf0840b28c02e86ff8fd15dfa6022068c8441252a47c8294357246ee0dab4b977669530
ae31daec37e1922fd79883d0121030a830e465562709ddcdc1e397af9098c1ff862ba98f3a8cfd5f fb9c95c9141eb5d301600"
}
```

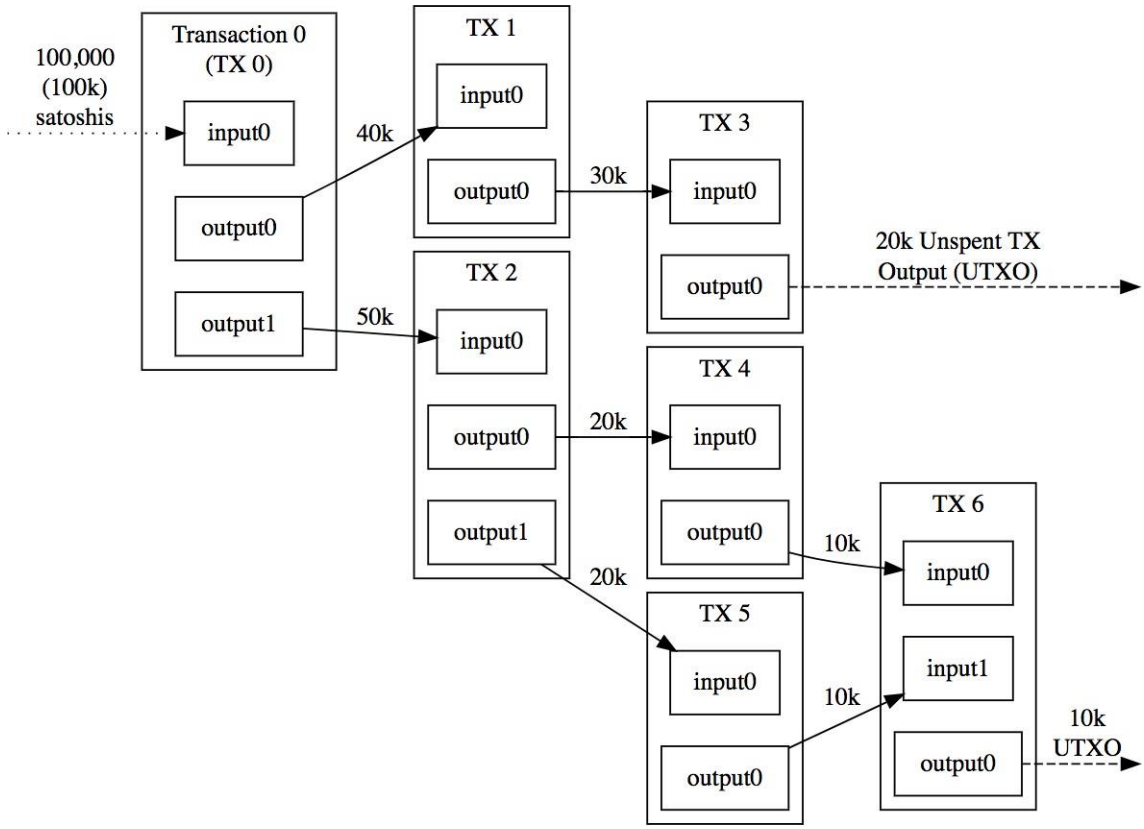

This shows that we sent 0.01 BTC to mg51NavnAmVsddKHQj4zSJVvM5wZ9RVp3h. However, there is a significant amount of information that is hidden in the transaction hex field. To learn more about it lets lookup the transaction on https://live.blockcypher.com/btc-testnet/ and see what we can figure out.



Congratulations! You have just made a Bitcoin transaction. You have seen this happen on a block explorer and can conjure the power of magic internet money!

Here is what we can see from the block explorer There is one input to the transaction and two outputs from the transaction. If we go use `bc listunspent` you will see that you no longer have one UTXO but you rather have one new one and the old UTXO is no longer displayed. This is because the UTXO was consumed in this transaction when it was used as an input. This also helps explains why there are two outputs. The first output is you sending the 0.01 to your TAs. The TAs now have that UTXO. The second output is the change that is being sent back to your wallet. If you wanted you could have sent the change back to the original address, however this is considered bad practice from a privacy perspective so by default bitcoin-cli sent the change to a new address in your wallet.

This is a visual summary of what you are seeing in your wallet. If you would like to read a bit more on UTXOs and how this differs from an account model, this is a reasonable intro https://medium.com/@sunflora98/utxo-vs-account-balance-model-5e6470f4e0cf



Triple-Entry Bookkeeping (Transaction-To-Transaction Payments) As Used By Bitcoin

In addition to the UTXOs we can also see that a fee was paid to the miners so that they include the transaction in a mined block. In this case the fee was 0.00000168. The value of the input was 0.07705682 and the value of the outputs was 0.06705514 and 0.01. The input value is equal to the sum of all of the output values and the transaction fee. Therefore, no money was created or destroyed in this transaction.

Assignment Deliverable 3: Find the transaction ID in which you sent your TAs 0.01 BTC and submit it.

What just happened?

To make this transaction happen a lot of things are being done for you by bitcoin-cli behind the scene. This includes selecting which UTXO to spend, calculating the transaction fee, building the
transaction, signing the transaction, and broadcasting the transaction. In the subsequent sections you will learn all about this process because we are going to build a transaction piece by piece.