Blockchain Lab - Bitcoins and Co.
January 16th, 2023
Yves Roudier - UCA / Polytech Nice Sophia

This lab's purpose is to make you experiment with various blockchains and their mechanisms. You will manipulate blockchain supported cryptocurrencies and program a small blockchain. You are expected to hand screenshots of your experiments (as requested within this document or beyond) together with comments and explanations about your findings. For the programming part, you are also requested to provide the *commented* code you develop within your report together with explanations about its design.

1) Proof of Work

1.1) Install Blockshell and experiment

First off, make sure you have a working Linux (e.g. Ubuntu) OS running with Python 2.7 installed (e.g. "sudo apt-get install python" if need be). Also install virtualenv ("sudo apt-get install virtualenv").

If you do not have a Linux, you may still install Blockshell onto your OS, but the rest of the lab might be harder to handle.

We will be using the Blockshell blockchain simulator in order to get a better understanding of how a blockchain works and how proofs of work are computed.

Head over to http://github.com/daxeel/blockshell and follow the installation procedure explained there.

Then run Blockshell, create a few blocks and examine how they are crafted. You can also start the Blockshell Web Explorer web user interface to examine blocks (this will start a web server on your own computer, accessible through the loopback interface 127.0.0.1 on port 5000).

You will find more reference about this tool and Blockshell basic commands on page https://github.com/daxeel/blockshell/wiki/Get-Started-with-BlockShell

2) Bitcoin

2.1) Examine transactions

Similar block explorer tools also exist for real, notably for the Bitcoin and Ethereum blockchains (and a few more). For instance, have a look at the following sites:

- https://www.blockchain.com/explorer (or https://blockchain.info/)

- https://blockstream.info
- https://live.blockcypher.com/btc/
- https://btc.com/

Take screenshots of both a mined bitcoin and a normal transaction as can be seen in these tools with details on the addresses you capture. How much does a miner earn?

You can also browse the Ethereum's TestNet at: https://ropsten.etherscan.io/.

You can also retrieve a transaction, for instance by decoding it through the BlockCypher implementation.

The hashrate describes the computing power available (from individuals or pools). What can you say about its distribution from what you can observe with these tools? Is it safe with respect to the trust blockchain participants put into the cryptographic algorithms?

2.2) Full Bitcoin node: Bitcoin core client

- Install: follow instructions at: https://bitcoin.org/en/full-node#ubuntu-1604
- The client can be downloaded from: https://bitcoin.org/en/download
- Create $HOME/.bitcoin/bitcoin.conf with the following content:
```
rpcuser=user_name
rpcpassword=your_password
server=1
testnet=1
prune=550
```
- Run "bitcoin-qt" and wait for the synchronization with the blockchain.
- You will also have access to the bitcoin client and its operations, like for instance "bitcoin-cli getnewaddress" to retrieve your address.

How long will the synchronization take and why? (alternately, you can also run "bitcoind", which does not provide a GUI). You can let the synchronization operation continue in the background and check the process before closing your computer.

Now proceed to the following tasks.

2.3) Electrum client

- Install: (see also https://electrum.org/#download):
  sudo apt-get install python3-pyqt5
  wget https://download.electrum.org/3.3.8/Electrum-3.3.8.tar.gz (the lab has been tested with this version of Electrum but you can probably safely install the latest version of the software, see the website)
  sudo apt-get install python3-setuptools python3-pip
  python3 -m pip install --user Electrum-3.3.8.tar.gz
- Run "~/.local/bin/electrum --testnet" (Auto Connect / default_wallet / Standard wallet /new seed / Segwit)

- Start the daemon: “electrum --testnet daemon start”, then “electrum --testnet daemon load_wallet”
- Retrieve the first address from the Electrum wallet (“electrum --testnet listaddresses” or check the GUI “Receive” tab), you will use it for obtaining Bitcoins

2.4) Receiving Bitcoins

Now connect to one of the following faucets and ask for being funded with some Bitcoins (you won’t need mining):

- https://coinfaucet.eu/en/btc-testnet/
- https://kuttler.eu/en/bitcoin/btc/faucet/
- https://bitcoinfaucet.uo1.net/
- https://testnet-faucet.com/btc-testnet/
- https://onchain.io/bitcoin-testnet-faucet

Carefully note down the time at which you received the bitcoins.

Using multiple faucets may be especially useful if you want to repeat the experience (be careful that faucets limit the number of bitcoins you may receive at once and/or during a certain timeframe).

If you cannot find any faucet available, ask your lab instructor or one of your colleagues that has received Bitcoins.

2.5) Examine the transaction bitcoins

Check your wallet by issuing “electrum –testnet listunspent”. You should see a new transaction, as well as in the Electron GUI “History” tab.
Also check the transaction on BlockCypher using your address (make sure you are browsing the Bitcoin testnet blockchain and not the real Bitcoin blockchain).
By the way, is the transaction immediately displayed? What can you deduce?
Let’s have a closer look:

- Retrieve the previous transaction with: electrum --testnet gettransaction "*transactionHash*”, then retrieve the hex value of the transaction
- Deserialize the transaction: “electrum --testnet deserialize *hex*”. Compare the result with what we obtained from BlockCypher. How can you explain that?

Ask one of your colleagues in the class to check if he/she can see your transaction based on the time at which it took place (the amount of bitcoins received might also help). Does that mean that your address remains anonymous? Is the content of the transaction private?

2.6) Sending bitcoins

Now place a transaction back to the faucet (and/or to one of your colleagues), either through the Electrum GUI “send” tab, or through the following CLI commands:

- “electrum --testnet payto <bitcoin_address> <amount> | electrum --testnet broadcast - ”

- The transaction is signed, you can check your keys through “electrum --testnet listaddresses | electrum --testnet getprivatekeys -”

Don’t forget to send your remaining BTCs back to the faucet you received them from once you are done with this part of the lab.

3) More Blockshell

In this part, you will get Blockshell closer to the Bitcoin implementation.

4.1) Introducing addresses

Add an address (an identity) into transactions to distinguish between multiple users (even though we create all transactions locally on the same “node” so to speak).

4.2) Designing our cryptocurrency

Introduce a syntax in order to account for cryptocurrency transfers (we will call our cryptocurrency the “NiceCoin”). Then create a few transactions, including double spending (assume that each user receives a limited amount of NiceCoins to begin with). Now implement a new operation for checking all recorded blocks, verifying transactions, and reporting that a new block cannot be validated if double spending is spotted.

4.3) Introducing multiple transactions per block

We want to record more than one transaction per block. Implement this behavior into Blockshell. First compute a block hash based on the concatenation of all transactions it contains. Then compute the hash using the Merkle tree technique in order to perform the block hash computation in a modular fashion.

4.4) The hash computation

Let’s have a look at how the difficulty parameter works. You will try to set up the blockchain with a difficulty parameter of 10 instead of the default 3. What happens? What is the principle of the algorithm in Blockshell? Try to implement the Bitcoin PoW instead. (Hint: have a look into the *blockchain* directory).

4) Appendix: Useful installs (utils)

You may have to install the following packages:

- Curl (“sudo apt-get install curl”)
- Pip (“sudo apt-get install python-pip”)
- Flask (“pip install -U Flask”)
- NPM (“sudo apt-get install npm”)
- Web3 (“npm install web3”)

- Ethereum:
  ```
  sudo apt-get install software-properties-common
  sudo add-apt-repository -y ppa:ethereum/ethereum
  sudo apt-get update
  sudo apt-get install ethereum
  ```
- Hex editor: Bless: “sudo apt-get install bless” or Okteta: “sudo apt-get install okteta”