Bitcoin Programming Assignment

Cyril Renevey, Lukas Pestalozzi, Joey Zenhäusern, Leonardo Wirz

December 15, 2017

Contents

1 Mobile wallet on testnet

We use the testnet address `mipVxuurz9XAapprSX5RquroiZfa4povRj` for this part.

2 Wallet management in C#

The `WalletManager` class of our application, as the name suggests, is capable of managing a set of wallets, which includes creation, storing to disk and loading from disk, as well as QR-code generation. There are three constructors, the first one simply takes a filename for the new wallet, and will either load this wallet or create it, if it does not exist yet. The other two constructors take either a private key, or a bitcoin secret, as an additional argument and may be used for easier handling of existing private keys in other parts of our application.

The saved wallets can be found inside the `wallets` folder in the root directory of the project. The first line of the wallet file contains the bitcoin secret, the second and third line contain the main and test address respectively. We also store the QR-codes of the testnet address for each wallet along with it in the same folder. The generation of the QR-code is implemented using a http request to a web API provided by *www.qrserver.com* which returns a picture of the code as a `PNG` file.

3 Vanity and Proof-of-work

We want to find a private-key which generates an address ending with a specific string, for example "nice". We choose to find an ending (as opposed to a beginning), because the beginning of a testnet address is not uniformly distributed, and therefore a lot of different combinations are impossible to find (in our case "nice" cannot be found at the beginning). Since we can't find the private-key for a given address (this is the whole point of asymmetric-key cryptography), we have no other choice than to randomly generate new private-keys and test if the corresponding address actually ends by the string we want. To decrease the expected waiting time we make no difference between uppercase and lowercase letters. For example, "NiCe" is equivalent to "nice".

We use the following steps to find a vanity-address:

1. Initialize the needed variables.
2. Randomly generate a new private-key.
3. Test if the last characters of the new address (to lower case) are equal to "nice" (the desired ending).
4. If it is not equal then go to 2.
5. Save the last generated private-key.

Here are 4 (testnet) addresses ending with the first 4 or 5 characters of our names. The respective private keys can be found in the wallet folder:

- Leonardo Address : `mknA88QyoqpsQu4eVN3rH1YNKfdmR7LEon`
- Cyril Address : `n4E2oFaeZ49k3aaX2JT3cLi4GEppJCYriL`
- Joey Address : `mkez7rL2AF8LvMMwr44K5BqTcUne5aJoeY`
- Lukas Address : `mksZSHNnEA81eh2EKwSQrYNEaDRe5LUKAs`

How is it related to a proof-of-work algorithm?
In fact, this algorithm can be seen as a proof-of-work algorithm. Instead of a hash function, we use asymetric-key cryptography as the 'one-way' function. It is easy to check whether an address (generated from a private-key) ends with the given string, but it is hard to find a private key generating a desired address. Finding a vanity address is the work we do, and the length of the string determines the difficulty. For more details about proof-of-work see 7.1.

How could you scale your algorithm to find the solution faster? Does that apply to mining?
The easiest way to find a solution faster, is to parallelize the algorithm. That is, we can increase the number of private-keys we check per second, by running it on several CPU's or even computers a the same time. This, of course, reduces the expected time it takes until we find a vanity-address. Clearly this also applies to mining, the more hashes a miner can try per second, the more likely is it that he finds one that is smaller than the desired difficulty.

The code we used to find the vanity-addresses above can be found in the `Vanity.cs` file. The user can determine how many CPUs it should run on and what addresses to search for.

4 Interaction with Qbitninja blockchain indexer

Choose a transaction at random on https://blockchain.info
We took the transaction "5e89acb5bb302098207ff1ac099e6ff00909fdc46f55ee65c5b3b0ae182d0fc3". We can explore it at this address.

Download all information available for this transaction from your code
From this transaction, we store the fees, the block id and the lists of received coins and spent coins. We also print all outputs and inputs from the transaction.

Download all information relative to your CoPay address from your code
We need to create the walletclient object in the TestNet network. For this, all active address' are added manually.

Print in the console the balance and the list of transactions related to this address
From the previously created WalletClient, we get all the transactions. For each transaction, the ID and the amount is printed. The amount is positive when receiving coins and negative when sending ones, then the total balance is calculated by summing all amounts.

Print the list of unspent transaction outputs, also known as UTXO or coins
For this part, we first need to get all address' of the wallet, then for each address we get the transactions with the "UnspentOnly" flag set to true. Finally, the ID ans the amount in printed.

Explain the relationship between the balance and the UTXO set.
The UTXO contains all unspent outputs. There is a subset of these outputs which correspond to a list of public keys that have been generated with the same private key. This subset is usually called the listunspent set. The total balance for this private key is computed by summing over the listunspent set.

5 Bitcoin Transaction in C#

Any Bitcoin transaction has a hash (the unique identifier for the transaction), some *inputs* and some *outputs*: Inputs describe which coins (that we received from previous transactions) we want to spend. Each input therefore has to reference a specific output from a past transaction, and needs to show a prove that the sender is the owner of those coins. This is done by providing a signature signed by the private-key corresponding to the receiving address of the output.

Each output describes how many coins we want to send to which address. Typically there are 2 outputs: One for the actual transaction we want to make and one for the change which we send back to an address we control.

The difference of the amounts between the inputs and the outputs is the transaction-fee (of course the difference must be positive for the transaction to be valid).

In our case, we send 0.001 tBTC from the vanity-address `mksZSHNnEA81eh2EKwSQrYNEaDRe5LUKAs` to the CoPay address `mipVxuurz9XAapprSX5RquroiZfa4povRj`. The fee is 0.001 tBTC. This results in the transaction shown in Figure 1.

The internal structure of the transaction is as follows (long values are shortened for readability):

hash: `4f48f94745dea2dab906be295a25be57b190b4a823e49409cdfbb01398cce98a`

in: **prev_out:** **N**: 1, **hash**: `ead...7c6`

scriptSig: `044...f01 030...2d1`

out: **out0:** **value**: 0.001, **scriptPubKey**: `OP_DUP OP_HASH160 243...6d4 OP_EQUALVERIFY OP_CHECKSIG`

out1: **value**: 0.996, **scriptPubKey**: `OP_DUP OP_HASH160 3ab...799 OP_EQUALVERIFY OP_CHECKSIG`

out2: **value**: 0.0, **scriptPubKey**: `OP_RETURN 546861742077617320656173792`

There is one input: the output number 1 (the second output) from another transaction with hash `ead...7c6`. The *scriptSig* is the signature created using the private-key of the vanity-address (which corresponds to the target address of the *prev_out*) to prove ownership of the coins in the following outputs.

There are 3 outputs: the first is the actual 'payment' we want to make, the second is the change-back and the third is a message we added, because we can.

The code used to create the transaction can be found in the `TransactionTask.cs` file. There is also a function which shows, and updates, the number of block-confirmations of a given transaction.

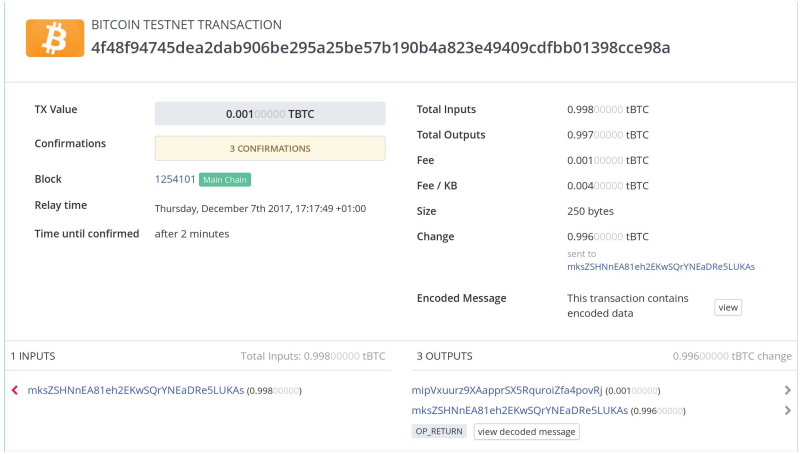

Figure 1: Screenshot of the transaction from `https://www.blocktrail.com/tBTC`

6 Investigate the blockchain

6.1 Satoshi Nakamotos Addresses & Transactions

There is a blog post that is well known within the community, written by a bitcoin developer named Sergio Lerner, which describes how one could go about attributing private keys to Satoshi Nakamoto[6]. The basic argument is that many of the first blocks seem to have been mined by the same machine, as they can be ordered by their extraNonce fields, which are incremented consecutively every time the nonce fields overflow.

Plotting those extraNonce values against the block number reveals an interesting pattern, which can be seen in Figure 2. The black lines restart periodically, indicating that the miner has reset his application at regular intervals, perhaps to backup the wallet, as it is speculated in the blog post. Furthermore, it is interesting that many of the lines appear to have a constant slope, which could be an additional hint that the blocks were mined by the same machine as it has taken the same time to mine each block, meaning that the mining performance did not change.

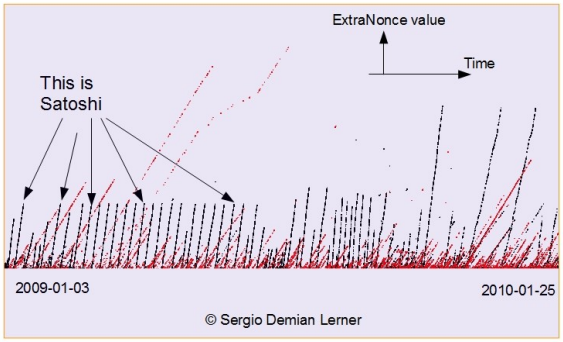

Figure 2: Image taken from:
`https://bitslog.wordpress.com/2013/04/17/the-well-deserved-fortune-of-satoshi-nakamoto/`

Once Satoshis addresses are known by means of the recipient of the coinbase transactions, it is merely an exercise of going through all blocks to identify the transactions that his addresses participated in.

6.2 Satoshi Nakamotos Wealth

Many sources (including the mentioned blogpost[6]) estimate that Nakamoto owns in the order of 1 million (10^6) bitcoins, which amounts to 6% of the current circulating supply. At the time of writing this, it would be worth almost 17′700′000′000 $ (17′692 $ per bitcoin). This would rank him at place 51 on forbes "The world's billionaires" list[7]. However, selling that many bitcoins would massively decrease the price, not to mention the "panic" it would cause when the *inventer of bitcoin* reappears and starts selling them.

6.3 The hidden message in the genesis block

The genesis block is the first block of a block chain. In the case of bitcoin it is numbered as block 0, but early versions of the bitcoin application numbered it as block 1. Satoshi Nakamoto left a message in this block, which reads:

> The Times 03/Jan/2009 Chancellor on brink of second bailout for banks

It is encoded as a hex value in the coinbase field, which is used as the sole input for coinbase transactions. A coinbase transaction is the first transaction in a block and is created by a miner in order to get the block reward. The original coinbase value is as follows:

> 04ffff001d0104455468652054696d65732030332f4a616e2f32303039204368616e63656c6c6f
> 72206f6e206272696e6b206f66207365636f6e64206261696c6f757420666f722062616e6b73

A normal transaction uses this section to refer to the parent's transaction outputs, but a coinbase transaction has no parent, as it is the first transaction in the block, and therefore the coinbase may contain an arbitrary value[5].

7 Research

7.1 Proof-of-Work : Purpose and Limitations

7.1.1 Definition, Properties and Purposes

Proof of work describes a system which requires a significant, but feasible amount of work in order to prevent a malicious use of the power of a computer. In the case of Bitcoin for example,

it prevents including unauthorized transactions in the blockchain or alter previous transactions already included in a block.

A key feature of proof of work systems is their asymmetry, the amount of work must be big enough, but easy to verify. For this purpose, as it is done for Bitcoin, one can use the property of hash functions. For a hash function $h(x)$ it is easy to find y such that $h(x) = y$ for a specific x but hard to find x such that $h(x) = y$ for a specific y (or, in fact any y)

Proof-of-work can have different purposes, but we will look at its use in the Bitcoin system. In this cryptocurrency, there is no central agent who verifies the validity of the transactions, thus the system needs to create an incentive for the miners to verify the transactions and to avoid wrong ones. Proof-of-work associated to the blockchain principles is a good help for that. Since only the longest chain is valid and since there are a lot of agents working with Bitcoin, it is very unlikely that a lonely agent could add an illicit transaction to the blockchain, because adding a block requires to solve the proof-of-work task more than 50% of the time to make the desired chain longer. Thus all the agents have the incentive to add verified blocks in order to keep their mined Bitcoins. This system can thus avoid unverified transactions or double-spent bitcoins, because following the majority is more profitable.

7.1.2 Limitations

Unfortunately this system has some limitations. Indeed proof-of-work requires a high computational power to be solved, thus the agents who want to contribute to the work need a powerful computer and a lot of electricity. In the case of Bitcoin, because the average time to find a block needs to be constantly around 10 minutes, the difficulty of the task increases with the number of agents. This leads to a huge power consumption required by Bitcoin, as it is in December the 6th, the cryptocurrency uses around 32TWh of energy per year[1] which corresponds to the consumption of Serbia, or 0.14% of the world consumption[1]. In comparison, VISA uses around 0.54TWh per year to execute 82.3 billion transactions in 2016 [1]. It is increasing at a high rate every day and in perspective of the actual energy crisis it may be crucial, for the survival of cryptocurrencies, to find less energy consuming alternatives. The huge demand in electricity rises an other problems, the countries with low energy cost are more attractive for miners, but one of the goal of Bitcoin is to reduce centralized agents. We observe that, as it was in June 2017, the Chinese miners possessed more than 70% of the mining power of Bitcoin (see Figure 3), which lead to a centralization to a specific country of the transactions choice for the block.

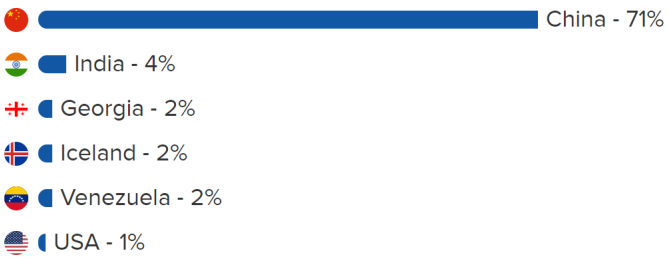

Figure 3: Proportion of Hash rate per Country [2].

An other limitation of the proof-of-work systems is their incentive to create groups of miners (so called mining-pools) in order to reduce the high volatility of returns by redistributing the gains over a group instead of a miner alone. An agglomeration of miners would have more hash power to solve the proof-of-work task and thus would receive bitcoins more often, which would be redistributed to the different miners. Each miner would receive less bitcoins but more often. This would keep their expectation of return constant, but reduces the volatility which leads to a increase in utility for the miner. However, the trust in Bitcoin depends on its decentralized property, thus if a mining-pool is close to controlling 50% of the power shares, the trust in Bitcoin would fall as much as its value compared to other currency, so mining could become non-profitable. For this reason, the pools try to avoid this situation to keep the trust in the cryptocurrency.

7.1.3 Alternatives

The alternatives to Proof-of-work must keep the same important incentives, such as: encourage verification, provide consensus, avoid invalid transactions, and decentralization. These alternatives also need to reduce the impact of the limitations of proof-of-work.

Proof-of-stake The choice of the next block, in a proof-of-stake system, is determined my a random selection depending on the wealth or age of a miner or mining account. This alternative gets rid of the energy problem, however it is an elitist system. The wealthier a miner is, the more voting power he has. Thus it is also profitable to create conglomerates of rich people which may have more than 50% of the voting power and thus create a centralized system. Some variation to the system, such as the age of a mining account, could make proof-of-stake less "money-elitist" by giving more voting power to older mining accounts (has been mining for a longer time).

Proof-of-burn In this system, instead of using the money for electricity consumption or computers, miners "burn" it, i.e. send it to an address which cannot spend coins, and in return they get to be randomly chosen for building the next block. The more coins you burn, the more chance you have to be selected, thus it works the same as proof-of-work, the more you spend your money in electricity and power, the more chance you have to solve the required proof-of-work task. Because of its similarity with proof-of-work, proof-of-burn does not solve the problems of conglomerates, but it drastically reduce the energy consumption of the system.

Proof-of-capacity Also known as proof-of-space, this system gives more chance of building a block to the miners who provides more storage space. This system is used by Burstcoin for its cryptocurrency. This alternative of proof-of-work is less energy consuming than running computers to solve a useless task and the provided storage space could be used for different purposes in different fields. There are more and more data created every day and huge data center are built all around the world, so any storage space is useful. Like for proof-of-burn, this alternative only get rid of the energy problem, by using the energy for a useful cause. However it is the most limiting problem, since there is an incentive in avoiding conglomerates to keep trust in a decentralized currency. The difference with proof-of-burn, which encourages wasting money as much as proof-of-work, is that the storage space can be used for other purposes.

7.2 Ethereum, an Alternative to Bitcoin

Bitcoin was the first created cryptocurrency in 2008, but since then and because of its success, numerous of different new decentralized cryptocurrencies have emerged. However one of them, Ether, or more precisely its associated protocol, Ethereum, have more success than the other ones. In the following lines we will see why it is the case by comparing it with Bitcoin.

7.2.1 Currency

Bitcoin is a deflationary currency, the mining reward is periodically divided by two and the total amount of bitcoins is limited to 21mio. Ether is not limited in its total amount and it is an inflationary currency. However, the main purpose of Ether is not to serve as an exchange currency, even though it can be used for that, but it is to help the realization of the famous smart contracts. We will discuss it later, but the main strength of Ethereum is its programmable part, which allows the creation of inventive contracts, the smart contracts, and Ether can be used as a payment method.

7.2.2 Block Size and Time

Bitcoin has blocks limited in size, around 2Mb per block, and new blocks are created in average every 10 minutes, which is a big limitation on the number of transactions you can carry out per

unit of time. In the case of Ethereum, new blocks are created in average every 14 seconds and there is no limitation in size, thus it can be as fast as we want it to be. However each block is limited by what we call a "gas limit", which correspond to the amount of Ether required to carry out the transactions and scripts of the block. This limit can be regularly increased by a vote between the miners.

7.2.3 Smart Contracts and Distributed Applications

The big advantage of Ethereum compared to its concurrent is its programing platform, which allows the creation of "smart contracts". It was also possible to do something similar with Bitcoin by attaching a physical contract or wealth to a portion of a bitcoin to easily exchange it, however it is very limited. A smart contract is a computer protocol which goal is to facilitate, verify and secure the terms of a contract, without intermediary agents (such as lawyer or a bank). Ethereum makes it easy to create one by giving the possibility to the user to develop an algorithm with the Ethereum programming language (Solidity). This protocol can then be added to the blockchain and the terms of the contract are thus fixed and easily verifiable. The number of different applications is almost limitless, such as a betting platform, self-executable contracts (such as transfer Ether at a certain time) or European options. It is a new way to create contracts, decentralized and customizable.

7.2.4 Proof-of-stake

The Ethereum company is actually working on a new system which would use proof-of-stake instead of proof-of-work, it is called Serenity. There are still some problems to create the good incentives for the users and miners, however it will be released soon and it can be a big step towards reduction of energy consumption.

7.3 What is an Oracle?

An oracle, in the context of blockchain technology, is a trusty agent, who's goal is to provide trustworthy informations and to attest the origin of a piece of information brought to the blockchain for its good operation. An oracle must deliver truth to the blockchain, so that executable contracts, that needs outside informations to run, can work properly. Reuters could be an oracle for financial data for example. Oracles can also be used in a multi-oracle protocol to sign off a transaction in order to be executed, we could think of the different keys needed to launch a nuclear weapon as an example, a certain number of keys, possessed by different people, are required to launch a nuclear bomb. In the blockchain technology, this would reduce the possibility of corruption or compromising data sources for a contract. Oracles are a key features to increase the possibilities of different smart contracts, that can now, using trustworthy oracles, be depending on outside of the blockchain information, such as weather, stock prices, exchange rates or scores of a sport games.

7.4 Application of Distributed Ledgers : Betting Platform

As an example of distributed ledgers, we will develop a protocol for a betting platform using Ethereum, which allows us to write more sophisticated smart contracts.

7.4.1 The Concept of the Platform

To construct our platform, we will consider tennis games in ATP world tour. The goal is to create a website, where people could suggest a bet on a game, who we will call proposer, and other people interested by a suggested bet would accept it, let's call them acceptor. So the interface of the website would show the list of the suggested bet and the acceptors can choose their favorite suggestions in the list. To explain more the process behind the bet, we need to define more clearly how a bet would look like. The proposer would need to specify for his bet :

- The underlying tennis game.
- The winner the proposer is expecting.
- The range of the price P of the bet (between 1 and 2 Ether for example).
- The return R for the acceptor in the case the proposer would be wrong in proportion of the price (for example if $R = 1.2$ and $P = 0.5$ eth then the acceptor can win 0.6 eth).

The acceptor would then see the content of the bet and would need to accept the bet by suggesting a price P in the range.

7.4.2 The Algorithm to be Included in the Blockchain

Once a suggestion found an acceptor the betting platform automatically generates a code in the Ethereum programming language that enclose the bet and adds it to the blockchain. As soon as the bet is included in the blockchain, this secure the rules of the bet and both players cannot alter the content of the bet anymore, thanks to the security brought by the blockchain principles. The algorithm would look like that :

1. Take P eth to the acceptor and $P \cdot R$ eth to the proposer and send it to an intermediary wallet at the time of the bet.
2. at the time of the game, if proposer win then send $P + P \cdot R$ eth to proposer.
3. Else send $P + P \cdot R$ to the acceptor.

With this algorithm, by taking the money from the beginning, we ensure that the loser of the bet can pay the winner.

7.4.3 The Role of the Oracle

A key process in the good functioning of the bet is the entry of the result of the tennis game. Indeed, in order for the all protocol to be secure, we need a trustworthy agent to verify and enter the relevant results of the tennis game in the algorithm. For this purpose the platform require an oracle to verify and enter the game results. As we talked about in the section 7.3 about oracles, it needs to be trustworthy. In our case the oracle could be a sport platform or the betting platform itself, because it would have a very high incentive to enter good results in order to gain trust and thus customers.

References

[1] *Bitcoin Energy Consumption Index*, https://digiconomist.net/bitcoin-energy-consumption, retrieved December 6th 2017.

[2] Jordan Tuwiner, *Bitcoin Mining in China*, https://www.buybitcoinworldwide.com/mining/china/, June 13th 2017.

[3] *Ethereum vs Bitcoin, quelles différences entre ces deux technologies?*, https://www.crypto-france.com/ethereum-vs-bitcoin/, May 24th 2016.

[4] Matthew De Silva, *What is an Oracle?*, https://www.ethnews.com/what-is-an-oracle, June 18th 2017.

[5] *Coinbase*, https://en.bitcoin.it/wiki/Coinbase, August 28th 2014.

[6] *The Well Deserved Fortune of Satoshi Nakamoto, Bitcoin creator, Visionary and Genius*, https://bitslog.wordpress.com/2013/04/17/the-well-deserved-fortune-of-satoshi-nakamoto/, April 17th 2013

[7] *Forbes World's Billionaires List*, https://www.forbes.com/billionaires/list/#version:static, December 2017