Handout 7 (Bitcoins)

In my opinion Bitcoins are an elaborate Ponzi scheme[1] —still the ideas behind them are really beautiful and not too difficult to understand. Since many colourful claims about Bitcoins float around in the mainstream and not-so-mainstream media, it will be instructive to re-examine such claims from a more technically informed vantage point. For example, it is often claimed that Bitcoins are anonymous and free from any potential government meddling. It turns out that the first claim ignores a lot of research in de-anonymising social networks, and the second underestimates the persuasive means a government has at its disposal.

There are a lot of articles, blogposts, research papers etc. available about Bitcoins. Below I will follow closely the very readable explanations from

http://www.michaelnielsen.org/ddi/how-the-bitcoin-protocol-actually-works/ and

http://www.imponderablethings.com/2013/07/how-bitcoin-works-under-hood.html

The latter also contains a link to a nice youtube video about the technical details behind Bitcoins.

Let us start with the question who invented Bitcoins? You could not make up the answer, but we actually do not know who the inventor is. All we know is that the first paper

https://bitcoin.org/bitcoin.pdf

is signed by Satoshi Nakamoto, which however is likely only a pen name. There is a lot of speculation who could be the inventor, or inventors, but we simply do not know. This part of Bitcoins is definitely anonymous. The paper above is from the end of 2008; the first Bitcoin transaction was made in January 2009. The rules in Bitcoin are set up so that there will only ever be 21 Million Bitcoins with the maximum reached around the year 2140. Currently there are already 11 Million Bitcoins in 'existence'. Contrast this with traditional fiat currencies where money can be printed almost at will. The smallest unit of a Bitcoin is called a Satoshi, which is the 10^{-8}th part of a Bitcoin. Remember a Penny is the 10^{-2}th part of a Pound.

The two main cryptographic building blocks of Bitcoins are cryptographic hashing functions (SHA-256) and public-private keys using the elliptic-curve encryption scheme for digital signatures. Hashes are used to generate 'fingerprints' of data that ensure integrity (absence of tampering). Public-private keys are used for signatures. For example sending a message, say msg, together with the encrypted version

$$msg, \{msg\}_{K^{priv}}$$

[1] http://en.wikipedia.org/wiki/Ponzi_scheme

allows everybody with access to the corresponding public key K^{pub} to verify the message came from the person who knew the private key. Signatures are used in Bitcoins for verifying the addresses where the Bitcoins are sent from. Addresses in Bitcoins are essentially the public keys. There are 2^{160} possible addresses, which is such a vast amount that there is not even a check for duplicates, or already used addresses. If you start with a random number to generate a public-private key pair it is very unlikely that you step on somebody else's shoes. Compare this with the email-addresses you always wanted to register with, say Googlemail, but which are already taken.

One main difference between Bitcoins and traditional banking is that you do not have a place, or places, that record the balance on your account. Traditional banking involves a central ledger which specifies the current balance in each account, for example

account owner	balance
Alice	£10.01
Bob	£4.99
Charlie	-£1.23
Eve	£0.00

Bitcoins work differently in that there is no such central ledger, but instead a public record of all transactions ever made. This means spending money corresponds to sending messages of the (oversimplified) form

$$\{\text{I, Alice, am giving Bob one Bitcoin.}\}_{K^{priv}_{Alice}} \tag{1}$$

These messages, called transactions, are the only data that is ever stored in the Bitcoin system (we will come to the precise details later on). The transactions are encrypted with Alice's private key so that everybody, including Bob, can use Alice's public key K^{pub}_{Alice} for verifying that this message came really from Alice, or more precisely from the person who knows K^{priv}_{Alice}.

The problem with such messages in a distributed system is that what happens if Bob receives 10, say, of these transactions. Did Alice intend to send him 10 Bitcoins, or did the message get duplicated by for example an attacker replaying a sniffed message? What is needed is a kind of serial number for such transactions. This means transaction messages look more like

$$\{\text{I, Alice, am giving Bob Bitcoin \#1234567.}\}_{K^{priv}_{Alice}}$$

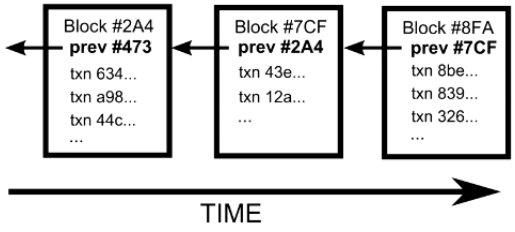
There are two difficulties, however, that need to be solved with serial numbers. One is who is assigning serial numbers to Bitcoins and also how can Bob verify that Alice actually owns this Bitcoin to pay him? In a system with a bank as trusted third-party, Bob could do the following:

- Bob asks the bank whether the Bitcoin with that serial number belongs to Alice and Alice hasn't already spent this Bitcoin.

- If yes, then Bob tells the bank he accepts this Bitcoin. The bank updates the records to show that the Bitcoin with that serial number is now in Bob's possession and no longer belongs to Alice.

But for this banks would need to be trusted and would also be an easy target for any government interference, for example. Think of the early days of music sharing where the company Napster was the single point of "failure" which was taken offline by law enforcement. Bitcoins is more a system like BitTorrent without a single central entity that can be taken offline.

Bitcoin solves the problem of not being able to rely on a bank by making everybody the "bank". Everybody who cares can have the entire transactions history starting with the first transaction made in January 2009. This history of transactions is called *blockchain*. Bob, for example, can use his copy of the blockchain for determining whether Alice owned the Bitcoin he received, and if she did, he transmits the message that he owns it now to every other participant on the Bitcoin network. An illustration of a three-block segment of the blockchain is (simplified) as follows



(2)

The chain grows with time. Each block contains a list of individual transactions, written txn in the picture above, and also a reference to the previous block, written prev. The data in a block (txn's and prev) is hashed so that the reference and transactions in them cannot be tampered with. This hash is the unique serial number of each block. Since this previous-block-reference is also part of the hash, the whole chain is robust against tampering. I let you think why this is the case?…But does it actually eliminate all possibilities of fraud?

We can check the consistency of the blockchain by checking whether all the references and hashes are correctly recorded. I have not tried it myself, but it is said that with the current amount of data (appr. 12GB) it takes roughly a day to check the consistency of the blockchain on a normal computer. Fortunately this "extended" consistency check usually only needs to be done once. Afterwards the blockchain only needs to be updated consistently.

Recall I wrote earlier that Bitcoins do not maintain a ledger, which lists all the current balances in each account. Instead only transactions are recorded. While a current balance of an account is not immediately available, it is possible to extract from the blockchain a transaction graph that looks like the picture shown in Figure 1. Each rectangle represents a single transaction. Take for example the rightmost lower transaction from Charles to Emily. This transaction has as receiver the address of Emily and as the sender the address of Charles.

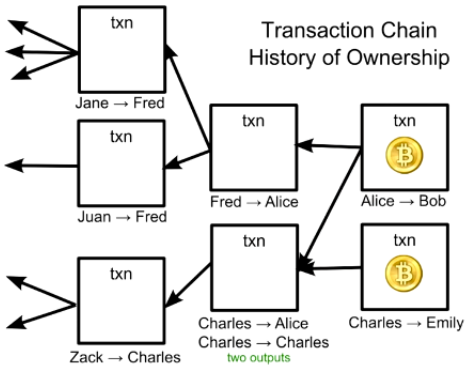


Figure 1: Transaction graph that is implicitly recorded in the public blockchain.

In this way no Bitcoins can appear out of thin air (we will discuss later how Bitcoins are actually generated). If Charles did not have a transaction of at least the amount he wants to give Emily to his name (i.e. send to an address with his public-private key) then there is no way he can make a payment to Emily. Equally, if now Emily wants to pay for a coffee, say, with the Bitcoin she received from Charles she can essentially only forward the message she received. The only slight complication with this setup in Bitcoins is that "incoming" Bitcoins can be combined in a transaction and "outgoing" Bitcoins can be split. For example in the leftmost upper transactions in Figure 1, Fred makes a payment to Alice. But this payment (or transaction) combines the Bitcoins that were send by Jane to Fred and also by Juan to Fred. This allows you to "consolidate" your funds: if it were only possible to split transactions, then the amounts would get smaller and smaller.

But in Bitcoins it is also important to have the ability to split the money from one or more incoming transaction to potentially more than one receiver. Consider again the rightmost transactions in Figure 1 and suppose Alice is a coffeeshop owner selling coffees for 1 Bitcoin. Charles received a transaction from Zack over 5 Bitcoins, say. How does he pay for the coffee? There is no explicit notion of *change* in the Bitcoin system. What Charles has to do instead is to make one single transaction with 1 Bitcoin to Alice and with 4 Bitcoins going back to himself, which then Charles can use to give to Emily, for example.

Let us consider another example. Suppose Emily received 4 Bitcoins from Charles and independently received another transaction (not shown in the picture) that sends 6 Bitcoins to her. If she now wants to buy a coffee from Alice for 1 Bitcoin, she has two possibilities: She could just forward the transaction from Charles over 4 Bitcoins to Alice split in such a way that Alice receives 1 Bitcoin and Emily sends the remaining 3 Bitcoins "back" to herself. In this case she would now be in the "posession" of two unspend Bitcoin transactions, one

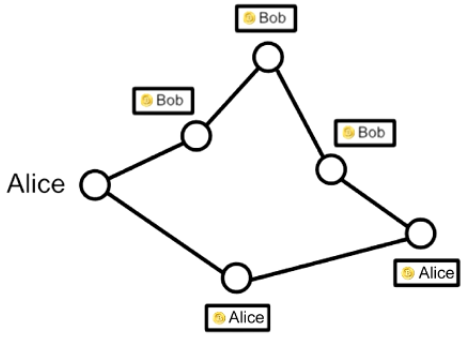
over 3 Bitcoins and the independent one over 6 Bitcoins. Or, Emily could combine both transactions (one over 4 Bitcoins from Charles and the independent one over 6 Bitcoins) and then split this amount with 1 Bitcoin going to Alice and 9 Bitcoins going back to herself.

I think this is a good time for you to pause to let this concept of transactions really sink in…You should see that there is really no need for a central ledger and no need for an account balance as familiar from traditional banking. The closest what Bitcoin has to offer for the notion of a balance in a bank account are the unspend transactions that a person (more precisely a public-private key address) received. That means transactions that can still be forwarded.

After the pause also consider the fact that whatever transaction is recorded in the blockchain will be in the "historical record" for the Bitcoin system. If a transaction says 1 Bitcoin goes from address A to address B, then this is what will be—B has then the possibility to spend the corresponding Bitcoins, whether the transaction was done fraudulently or not. There is no exception to this rule. Interestingly this is also how Bitcoins can get lost: One possibility is that you send Bitcoins to an address for which nobody has generated a private key, for example because of a typo in the address field—bad luck for fat fingers[2] in the Bitcoin system. The reason is that nobody has a private key for this erroneous address and consequently cannot forward the transaction anymore. Another possibility is that you forget your private key and you had messages forwarded to the corresponding public key. Also in this case bad luck: you will never be able to forward this message again, because you will not be able to form a valid message that sends this to somebody else (we will see the details of this later). But this is also a way how you can get robbed of your Bitcoins. By old-fashioned hacking-into-a-computer crime, for example, an attacker might get hold of your private key and then quickly forward the Bitcoins that are in your name to an address the attacker controls. You will never again have access to these Bitcoins, because for the Bitcoin system they are assumed to be spent. And remember with Bitcoins you cannot appeal to any higher authority. Once the Bitcoins are gone, they are gone. This is much different in traditional banking where at least you can try to harass the bank to roll back the transaction.

This brings us to back to problem of double spend. Suppose Bob is a merchant. How can he make sure that Alice does not cheat him? She could for example send a transaction to Bob. But also forward the "same" transaction to Charlie, or even herself. If Alice manages to get the second transaction into the blockchain, Bob will be cheated out of his money. The problem in such conflicting situations is how should the network update their blockchain? You might end up with a picture like this

[2] http://en.wikipedia.org/wiki/Typographical_error



where Alice convinced some part of the "world" that she is still the owner of the Bitcoin and some other part of the "world" thinks it's Bob's. How should such a disagreement be resolved? This is actually the main hurdle where Bitcoin really innovated. The answer is that Bob needs to convince "enough" people on the network that the transaction from Alice to him is legit.

What does, however, "enough" mean in a distributed system? If Alice sets up a network of a billion, say, puppy identities and whenever Bob tries to convince, or validate, that he is the rightful owner of the Bitcoin, then the puppy identities agree. Bob would then have no reason to not give Alice her coffee. But behind his back, however, she has convinced everybody else on the network that she is still the rightful owner of the Bitcoin. After being outvoted, Bob would be a tad peeved.

The reflex reaction to such a situation would be to make the process of validating a transaction as cheap as possible. The intention is that Bob will get enough peers to agree with him that he is the rightful owner. But such a solution has always the limitation of Alice setting up an even bigger network of puppy identities. The really cool idea of Bitcoin is to go into the other direction of making the process of transaction validation (artificially) as expensive as possible, but reward people for helping with the validation. This is really a novel and counterintuitive idea that makes the whole system of Bitcoins work so beautifully.

Proof-of-Work Puzzles

In order to make the process of transaction validation difficult, Bitcoin uses a kind of puzzles. Solving the puzzles is called *Bitcoin mining*, where whoever solves a puzzle will be awarded some Bitcoins. At the beginning this was 50 Bitcoins, but the rules of Bitcoin are set up such that this amount halves every 210,000 transactions or so. Currently you will be awarded 25 Bitcoins for solving a puzzle. Because the amount will halve again and then later again and again, around the year 2140 it will go below the level of 1 Satoshi. In that event no new Bitcoins will ever be created again and the amount of Bitcoins stays fixed. There will be still an incentive to help with validating transactions, because there is the possibility in Bitcoins to offer a transaction fee to whoever

solves a puzzle. At the moment this fee is usually set to 0, since the incentive for miners is the 25 Bitcoins that are currently awarded for solving puzzles.

What do the puzzles that miners have to solve look like? The puzzles can be illustrated roughly as follows: Given a string, say `"Hello, world!"`, what is the salt so that the hash starts with a long run of zeros? Let us look at a concrete example. Recall that Bitcoins use the hash-function SHA-256. Suppose we call this hash function `h`, then we could try the salt `0` as follows:

```
h("Hello, world!0") =
  1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64
```

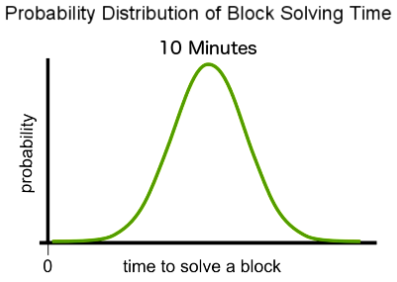
OK this does not have any zeros at all. We could next try the salt `1`:

```
h("Hello, world!1") =
  e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7d8
```

Again this hash value does not contain any leading zeros. We could now try out every salt until we reach

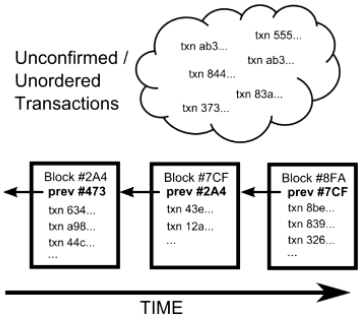
```
h("Hello, world!4250") =
  0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e9
```

where we have four leading zeros. If four zeros are enough, then the puzzle would be solved with this salt. The point is that we can very quickly check whether a salt solves a puzzle, but it is hard to find one. Latest research suggest it is an NP-problem. If we want the output hash value to begin with 10 zeroes, say, then we will, on average, need to try $16^{10} \approx 10^{12}$ different salts before we find a suitable one. In Bitcoins the puzzles are not solved according to how many leading zeros a has-value has, but rather whether it is below a *target*. The hardness of the puzzle can actually be controlled by changing the target according to the available computational power available. I think the adjustment of the hardness of the problems is done every two weeks. I am not sure whether this is an automatic process. The aim of the adjustment is that on average the Bitcoin network will most likely solve a puzzle within 10 Minutes.



It could be solved quicker, but equally it could take longer, but on average after 10 Minutes somebody on the network will have found a solution.

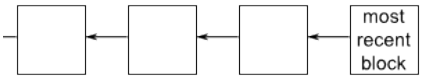
Remember that the puzzles are a kind of proof-of-work that make the validation of transactions artificially expensive. The validation and the derivation of the puzzle is done as follows:



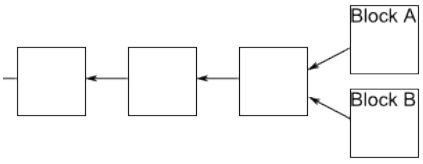
(3)

There are some unconfirmed transactions. Choosing some of them, the miner (i.e. the person/computer that tries to solve a puzzle) will form a putative block to be added to the blockchain. This putative block will contain the transactions and the reference to the previous block. The serial number of such a block is simply the hash of all the data. The puzzle can then be stated as the "string" corresponding to the block and which salt is needed in order to have the hashed value being below the target. Other miners will choose different transactions and therefore work on a slightly different putative block and puzzle.

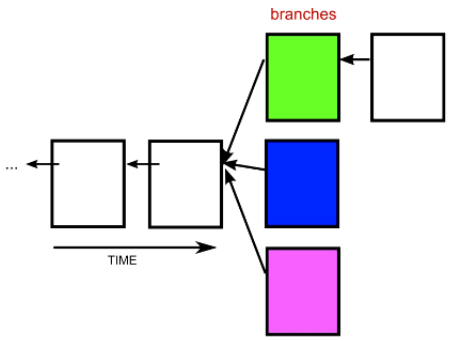
The intention of the proof-of-work puzzle is that the blockchain is at every given moment nicely linearly ordered, see the picture shown in (3). If we don't have such a linear ordering at any given moment then it may not be clear who owns which Bitcoins. Assume a miner David is lucky and finds a suitable salt to confirm the transactions. Should he celebrate? Not yet. Typically the blockchain will look as follows



But every so often there will be a fork



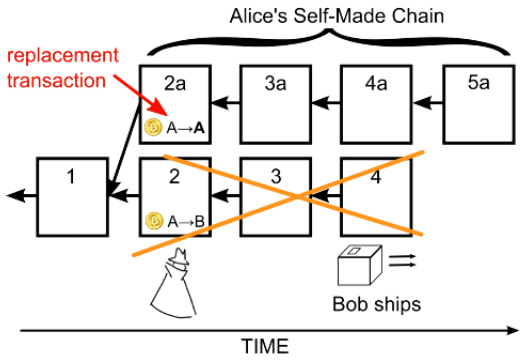
What should be done in this case? The tie is broken if another block is solved, like so:



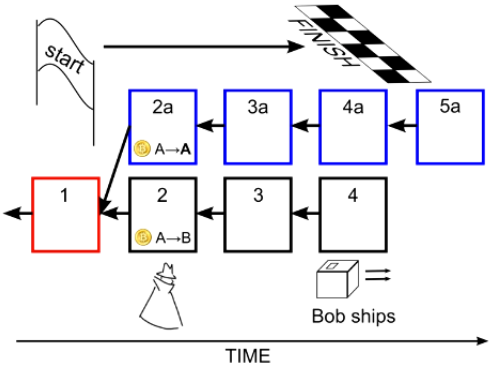
The rule in Bitcoins is: If a fork occurs, people on the network keep track of all forks. But at any given time, miners only work to extend whichever fork is longest in their copy of the block chain. Why should miners work on the longest fork? Well their incentive is to mine Bitcoins. If somebody else already solved a puzzle, then it makes more sense to work on a new puzzle and obtain the Bitcoins for solving that puzzle. Note that whoever solved a puzzle on the "loosing" fork will actually not get any Bitcoins as reward. Tough luck.

Alice against the Rest of the World

Let is see how the blockchain and the proof-of-work puzzles avoid the problem of double spend. If Alice wants to cheat Bob she would need to pull off the following ploy:



Alice makes a transaction to Bob for paying, for example, for an online order. This transaction is confirmed, or validated, in block 2. Bob ships the goods around block 4. In this moment, Alice needs to get into action and try to validate the fraudulent transaction to herself instead.



At this moment she is in a race against all the computing power of the "rest of the world". She has to solve the puzzles one by one, because the hash of a block is determined by all the data in the previous has. She might be very lucky to solve one puzzle for a block before the rest of the world, but to be lucky many times is very unlikely. In order to raise the bar for Alice, merchants accepting Bitcoin use the following rule of thumb: A transaction is "confirmed" if (1) it is part of a block in the longest fork, and (2) at least 5 blocks follow it in the longest fork. In this case we say that the transaction has "6 confirmations". A simple calculation shows that these number of confirmations can take up to 1 hour and more. While this seems excessively long, from the merchant's point of view it is not long at all. For this recall that ordinary credit cards can have their transactions been rolled-back for 6 months or so. The point however is that the odds for Alice being able to cheat are very low.

Connected with the 6-confirmation rule is an interesting phenomenon. On average, it would take several years for a typical computer to solve a proof-of-work puzzle, so an individual's chance of ever solving one before the rest of the world, which typically takes 10 minutes, is negligibly low. Therefore many people join groups called *mining pools* that collectively work to solve blocks, and distribute rewards based on work contributed. These mining pools act somewhat like lottery pools among co-workers, except that some of these pools are quite large, and comprise more than 20% of all the computers in the network. It is said that BTC, the largest mining pool, has limited its members to not solve more than 6 blocks in a row. Otherwise this would undermine the trust in Bitcoins, which is also not in the interest of BTC, I guess.

Bitcoins for Real