Distributed Trust and Blockchains	Date:	*28th August, 2018*
Instructor: *Sujit Prakash Gujar*	Scribes:	Rama Rohit Reddy Aditya Singh Bisht Yash Goyal

Lecture 8: Bitcoin Ledger Architecture and Transactions

1 Recap

In this lecture, we will recapitulate the content in the previous seven lectures and then move forward to discuss about Bitcoin Ledger Architecture, Bitcoin transactions and Bitcoin scripts.

1.1 Relevant terms

- *Blockchain:* The basic idea is relatively simple and consists of two elements, which the name suggests: the block and the chain. The block is a highly encrypted list of entries related to transactions of a business. All information that is important to the business can be documented in the block. The documentation is not only managed in one location, but it is distributed among the users of the blockchain. There are countless copies, potentially millions distributed throughout the network. The chaining of the blocks ensures that the content of the block remains trustworthy at all times. By combining encryption, decentralization, a multitude of stakeholders and community control, this system is nearly impossible for hackers to penetrate.
- *Trust:* When two parties execute an agreement, there are several moving parts. But what makes the transaction efficient is trust. Through the use of blockchain, all the parties involved in a transaction only have to trust the technology.
- *Bitcoin:* It is the first decentralized digital cryptocurrency, an application of the blockchain network in which the network is peer-to-peer and transactions take place between users directly without an an intermediary such as a central bank or single administrator.
- *Smart Contract* A smart contract is similar to a contract in the physical world, but it is digital and is represented by a tiny computer program stored inside a blockchain. A smart contract is a piece of software that stores rules for negotiating the terms of an agreement, automatically verifies fulfillment, and then executes the agreed terms. Since a smart contract removes reliance on a third party when establishing business relations, the parties making an agreement can transact directly with each other.

1.2 Evolution of currencies.

- *Barter System* The system of exchange where goods or services are directly exchanged for other goods or services without saying a medium of exchange, such as money is known as the Barter system. This system existed in olden times. The drawback, of course, is coordination arranging a group of people, whose needs and wants align, in the same place at the same time. Two systems emerged to solve coordination: credit and cash.
- *Credit system* This is a system for allowing people to purchase things on credit i.e. borrowed money that you can use to purchase goods and services when you need them.
- *Cash system* Cash accounting is an accounting method where receipts are recorded during the period they are received, and expenses are recorded in the period in which they are actually paid.

- A *cash-based* system needs to be bootstrapped with some initial allocation of cash, without which no trades can occur. A *credit-based* system doesnt need bootstrapping, but the drawback is that anyone whos owed a debt is taking on some risk. Theres a chance that the other person never comes back to settle the debt.
- Currently we use *fiat currency* and *Banks* are responsible for maintaining the transactions. Issues in Banking system: No *Anonymity* or *Decentralization*.
- *Digital Cash* David Chaum proposed the idea of DigiCash in 1982 for which he invented a cryptographic tool called *blind signature*. It is a form of digital signature in which the content of a message is disguised (blinded) before it is signed. The resulting blind signature can be publicly verified against the original, unblinded message in the manner of a regular digital signature. It solved the issue of anonymity but couldn't solve the issue of *centralization*.
- *Some more cryptocurrencies:*
  - *e-Gold*: This was the first cryptocurrency backed entirely by gold, and released to the public in 1995
  - *DigiGold*: A digigold is a hybrid cryptocurrency that was created for people to use as they want.
  - *HashCash*: Proposed by Adam Back in 1997, Hashcash is a proof-of-work algorithm, which has been used as a denial-of-service countermeasure technique in a number of systems. A hashcash stamp constitutes a proof-of-work which takes a parameterizable amount of work to compute for the sender. The recipient (and indeed anyone as it is publicly auditable) can verify received hashcash stamps efficiently.

1.3 Desirable properties of Cryptocurrencies

- Decentralized
- Anonymity
- Avoid/Detect Double Spending
- Minting should not be associated with any existing fiat currency or commodity

1.4 Techniques

To design cryptocurrencies to satisfy the above mentioned properties we require some cryptographic techniques -

- *Hash functions* A cryptographic hash function is a mathematical algorithm that maps data of arbitrary size to a bit string of a fixed size (a hash) and is designed to be a one-way function, that is, a function which is infeasible to invert. Properties:
  - *Pre-image resistance* Given a hash value h it should be difficult to find any message m such that $h = hash(m)$.
  - *Second pre-image resistance* Given an input m1, it should be difficult to find a different input $m2$ such that $hash(m1) = hash(m2)$
  - *Collision resistance* It should be difficult to find two different messages $m1$ and $m2$ such that $hash(m1) = hash(m2)$.
- *Digital Signatures* A digital signature is a mathematical scheme for presenting the authenticity of digital messages or documents. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, that the sender cannot deny having sent the message, and that the message was not altered in transit.

- *Properties*
  - Only you can make your signature, but anyone who sees it can verify that its valid.
  - The signature should be tied to a particular document so that the signature cannot be used to indicate your agreement or endorsement of a different document.
- *Algorithms*
  - *RSA* algorithm is an asymmetric cryptography algorithm which actually means that it works on two different keys i.e. *Public Key* and *Private Key*. As the name describes that the Public Key is given to everyone and Private key is kept private.
  - The *ElGamal* signature scheme is a digital signature scheme which is based on the difficulty of computing discrete logarithms.
  - The *Digital signature algorithm(DSA)* does not encrypt message digests using private key or decrypt message digests using public key. Instead, it uses unique mathematical functions to create a digital signature consisting of two 160-bit numbers, which are originated from the message digests and the private key.
  - The *Elliptic Curve Digital Signature Algorithm (ECDSA)* offers a variant of the Digital Signature Algorithm (DSA) which uses elliptic curve cryptography.

1.5 Simple Cryptocurrencies

1.5.1 Goofy Coin

- Goofy can create new coins by simply signing a statement that hes making a new coin with a unique coin ID.
- Whoever owns a coin can pass it on to someone else by signing a statement that saying, Pass on this coin to X (where X is specified as a public key).
- Anyone can verify the validity of a coin by following the chain of hash pointers back to its creation by Goofy, verifying all of the signatures along the way.
- GoofyCoin does not solve the *doublespending* attack and therefore its not secure.

1.5.2 Scrooge Coin

- It solves the problem of double spending introduced in GoofyCoin by maintaining an append-only ledger which contains the history of all the transactions that have happened.
- Scrooge can build a block chain, which consists of a series of data blocks, each with one transaction in it. Each block has the Id of a transaction, the transactions contents, and a hash pointer to the previous block.
- Scrooge will digitally sign the final hash pointer and a transaction is valid only if it is in the block chain signed by Scrooge.
- Anybody can verify that transaction was endorsed by Scrooge by checking Scrooges signature on the block that records the transaction. Scrooge will make sure that he doesnt endorse a transaction that attempts to double spend an already spent coin.
- The problem here is that Scrooge Coin is *centralized* as Scrooge has too much influence. So we need to figure out how to provide the same services of Scrooge, but in a decentralized way, in which no particular party is the only trusted one.

2 Blockchain : An immutable ledger

Immutability means means once data has been written to a blockchain no one can change it. This is helpful for auditing and as a provider of data we can prove that our data hasnt been altered, and as a recipient of data we can be sure that the data hasnt been altered. These benefits are useful for databases of financial transactions.

Blockchains are essentially databases with some inbuilt pre-agreed technical and business logic criteria, kept in sync via peer-to-peer mechanisms and pre-agreed rules about what new data can be added. With respect to immutability, there are two key ideas that help to make tampering easy to detect: cryptographic functions and blocks.

1. **Hash and Blocks**: If a miner tries to change a transaction from history, he will have to re-mine all the blocks from that block till the current block and this will have to be reflected in every copy of the ledger in the network.Now, since the next block stores the hash of this block, the next block will also have to be re-mined. This is because the next block will have to be edited with the new previous block hash. This change will result in a different block hash. The same process will have to be propagated to the latest block in the chain. While this miner is busy remaining old blocks, there will be new blocks getting added to the chain. Thus to edit a historical record, the miner will have to re mine the old blocks and keep up with the pace of newly generated blocks too.

   In Bitcoins blockchain,transactions are bundled into blocks before being added to the blockchain database. Blocks contain some bitcoin transactions (payments) and also some other data including the previous blocks hash. As each block includes the previous blocks hash as part of its data, a chain of blocks is formed.

   - Each blocks hash is derived from the contents of the block
   - Each block refers to the previous blocks hash, not a sequential number
   - Data in a blockchain is internally consistent, that is you can run some checks on it, and if the data and hashes dont match up, there has definitely been some tinkering.

3 Why Blockchain can achieve consensus

Key technical challenge of Decentralised e-cash: Distributed consensus
Why consensus protocols:

- Reliablity in distributed systems
- Distributed key-value store enables various applications:
- DNS,public key directory,stock trades

Assume social networking sites like Facebook have millions of servers and form massive distributed database that records all actions that happens on the system like user comments,likes,posts,etc.Suppose new post comes than it may record copies of this in different nodes. Now, what the server needs to make sure is that that comment either gets recorded in all copies of that database, or none of them.If system follows atomicity,it works fine else it produces inconsistent data.

3.1 Distributed Consensus Protocol

There are n nodes that each have an input value. Some of these nodes are faulty or malicious. A distributed consensus protocol has the following two properties:

- The protocol terminates and all correct nodes decide on the same value.
- The value must have been generated by an correct node.

Bitcoin is a peertopeer system. When Alice wants to pay Bob, what she actually does is broadcast that transaction to all of the Bitcoin nodes that comprise the peertopeer network. There is no requirement that Bob should be listening on the network; Running a node is not necessary for Bob to receive the funds. The bitcoins will be whether or not hes operating a node on the network. At any given point, all the nodes in the peertopeer network have a ledger consisting of a sequence of blocks, each containing a list of transactions, that theyve reached consensus on.
Why consensus is hard:

- Nodes might crash, and nodes might outright be malicious.
- Network is imperfect and not all pairs of nodes connected
- Faults in network
- Latency(no notion of global time)

Impossibility Results in Consensus:

1. **Byzantine Generals Problem:** Byzantine army separated into divisions, each commanded by a general. Some generals may be traitors.The nodes which are malicious or faulty and try to prevent the consensus of a distributed system are called byzantine nodes. It is proven that it is impossible to achieve consensus if one-third or more byzantine nodes are present.
2. **Fischer-Lynch-Paterson:** It is showed that under some conditions, which include the nodes acting in a deterministic manner, it is impossible to reach consensus even a single faulty process.
3. **Paxos:** Paxos makes certain compromises. On one hand, it never produces an inconsistent result. On the other hand, it accepts the trade-off that under certain conditions, albeit rare ones, the protocol can fail to make any progress.

3.2 Bitcoin - Consensus

Bitcoin achieves consensus in the following way:

- Broadcast transaction to all the nodes.
- Each node collects new transactions into a block.
- A random node which solves a given puzzle given in equation below gets a chance to write to the ledger. It broadcasts its block.

$$H(nonce||prev\_hash||tX_1||tX_2||...) < target \tag{1}$$

- Nodes express their acceptance of the block by including its hash in the next block they create.
- Always longest chain is considered to be valid.

In this method:

- *Stealing* is not possible.
- No *Denial Of Service* as malicious node can only pause at max one block.
- *Double Spending* is possible because of orphan block. To avoid this a transaction is considered not valid until 6 blocks are added to it.

Previous when the puzzles are easy, there were a lot of orphan blocks, but as the difficulty increased, number of orphan blocks decreased.

3.3 Incentivize nodes

We need to incentivize nodes to behave honestly by paying them in units of this currency. Two separate incentive mechanisms in BitCoin: Block reward and Transaction fee.

1. **Block Reward:** According to rules of bitcoin,creater of block gets to include special transaction in that block and the node can also choose the recipient address of this transaction.This is analogous to payment to the node in exchange for the service of creating a block on the consensus chain.

   The value of this coin creation transaction is currently fixed at 25 bitcoins, it actually halves every 210,000 blocks and the rate drops roughly every four years. Were now in the second period. For the first four years of Bitcoins existence, the block reward was 50 bitcoins; now its 25 and it will continue to keep halving.

   Node gets the block reward regardless of whether it proposes a valid block or behaves maliciously but this node collect its reward only if the block in question ends up on the longterm consensus branch because just like every other transaction, the coincreation transaction will only be accepted by other nodes if it ends up on the consensus chain.

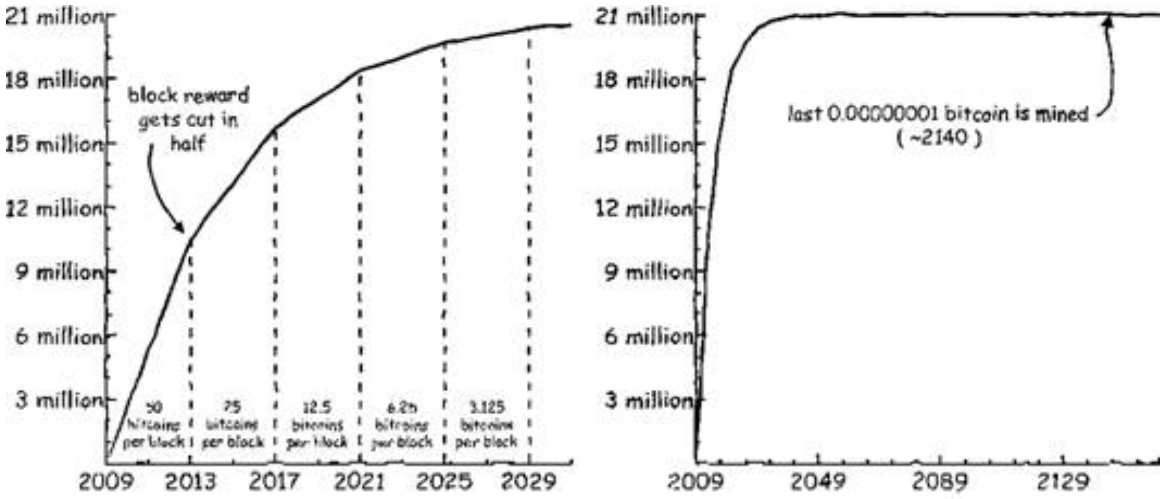

Figure 1: The block reward is cut in half every 4 years[1]

2. **Transaction fee:** It is important to note that this is the only way in which new bitcoins are allowed to be created. There is no other coin generation mechanism, and thats why 21 million is a final and total number (as the rules stand now, at least) for how many bitcoins there can ever be. This new block creation reward is actually going to run out in 2140.The second incentive mechanism is called the Transaction fee.

   The creator of any transaction can choose to make the total value of the transaction outputs less than the total value of its inputs. Whoever creates the block that first puts that transaction into the block chain gets to collect the difference, which acts as a transaction fee. So if a node thats creating a block that contains, say, 200 transactions, then the sum of all those 200 transaction fees is paid to the address that we put into that block. The transaction fee is purely voluntary, but we expect, based on understanding of the system, that as the block reward starts to run out, it will become more and more important, almost mandatory, for users to include transaction fees in order to get a reasonable quality of service.

[1]This image is taken from `https://ebrary.net/imag/edu/bar_bfbuf/image128.jpg`

4 Bitcoin Ledger Architecture

Generally in banks, individual transactions are added to the ledger one at a time as shown in figure2. We cannot maintain bitcoin transactions using such ledger because anyone who wants to determine if a transaction is valid will have to keep track of these account balances and there can be millions of people per block.

If we look at Figure 2. First four transactions are done and now for the fifth transaction, the question is whether Amit has 12 coins that he is trying to transfer to Sam. To figure this out, we have to look backwards in time forever to see every transaction affecting Amit and whether of not his net balance at the time when he is trying to transfer 12 coins to Sam is greater than 12 coins. This can be made a little bit more efficient by keeping track of Amit's balance after each transaction but that requires a lot of extra housekeeping besides the ledger itself.

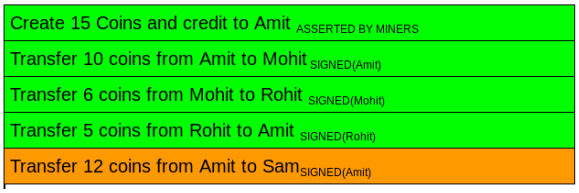

Figure 2: an account-based ledger[2]

So Bitcoin instead uses a ledger that keeps tracks of transactions similar to that of ScroogeCoin. As we can see in figure 3, these transactions specify a number of inputs and outputs. We can think inputs as coins being consumed and outputs as coins being created. For transactions in which new currency is being minted, there are no coins being consumed. Each transaction has a unique identifier and is signed by the owner to authorize the transaction.

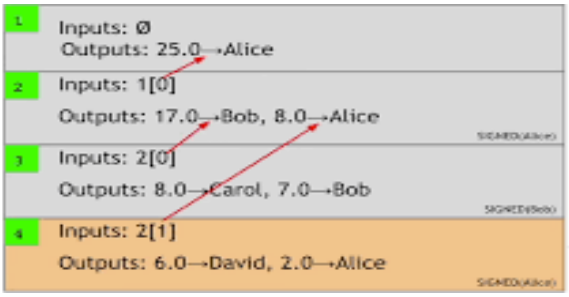

Figure 3: a transaction-based ledger, close to Bitcoin[3]

5 Bitcoin Transaction

Public keys are the identities in bitcoin transactions. In bitcoin, the entirety of a transaction output must be consumed by another transaction or none of it.
Eg: Mohit wants to pay 0.5BTC to Rohit, but the output that he owns is worth 1BTC. So Mohit needs to create a new output where 0.5BTC is sent back to himself and 0.5BTC to Rohit. Mohit should sign this whole thing to authorize the transaction.
The advantages of this type of ledger architecture is:

- **Efficient Verification:** To check the validity of new transaction, we just need to look up the transaction output that Mohit mentioned and make sure that he hasn't already been spent it.

[2]This image is created by us by taking class notes as reference.
[3]This image is taken from `http://haelchan.me/images/btc/ledger2.jpg`

Looking up the transaction output is easy as we use hash pointers. To ensure that it hasn't been spent, we need to scan the block chain between the referenced transaction and the latest block. This doesn't need to go all the way up to beginning of block chain and doesn't require keeping additional data structures.

- **Consolidating funds:** Since transactions can have many inputs and many outputs, splitting and merging value is easy. He creates a transaction with the two inputs and one output, with the output address being one that he owns. This lets him consolidate those two transactions.
- **Joint Payments:** Similarly, joint payments are also easy to do. They can create a transaction with two inputs and one output, but with the two inputs owned by two different people. The transaction will need two separate signatures for authorization.

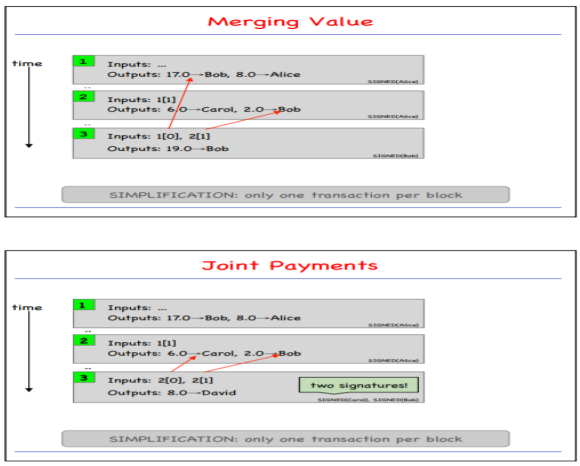

Figure 4: Examples of merging values and joint payments[4]

Transaction syntax: As we can see in Figure 5, there are three parts to a bitcoin transaction:

- **Metadata:** It contains some housekeeping information such as the size of the transactions, the number of inputs and the number of outputs, the hash of the entire transaction which servers as unique ID for the transaction. It allows us to use hash pointers to reference transactions. There is a "$lock_time$" field tells miners not to publish the transaction until the specified lock time.
- **Inputs:** The transaction inputs form an array and each input has the same form. An input specifies a previous transaction and contains a hash of that transaction, which acts as a hash pointer to it. It also contains the index of the previous transaction's outputs that's being claimed and a signature.
- **Outputs:** The outputs are also an array. Each output has just two fields. They each have a value, and the sum of all the output values has to be less than or equal to the sum of all the input values. The difference between sum of all output values and sum of all input values is a transaction fee to the miner who publishes this transaction.

[4]http://haelchan.me/2017/12/28/bitcoin-and-cryptocurrencies/

Figure 5: An actual Bitcoin transaction[5]

6 Bitcoin Scripts

The most common type of transaction in Bitcoin is to redeem a previous transaction output by signing with the correct key. Generally, transaction output says that "this can be redeemed by a signature from the owner of address X." Address is a hash of public key. So merely specifying the address X doesn't tell us what public key is and doesn't give us a way to check the signature. So for this purpose we use scripts.

- Bitcoin scripting has many similarities to a language called *froth*, which is an *old, simple, stack based* programming language.
- Bitcoin scripting language is very small and supports maximum of 256 instructions.
- Of those 256, 15 are currently disabled and 75 are reserved. The reserved instruction codes havent been assigned any specific meaning yet.
- Support for cryptography.
- Limits on time/memory.
- No looping.

The inputs also contain scripts instead of signatures. To validate that a transaction redeems a previous transaction output correctly, we combine new transaction's input script and the earlier one's output script. The resulting script obtained by concatenating them should run successfully in order for the transaction to be valid. These two scripts are called **scriptPubKey** and **scriptSig**, the output script just specifies a public keyor(an address to which public key hashes) and the input script specifies signature with that public key.

$$Structure of script :< \textbf{scriptSig} >< \textbf{scriptPubKey} > \tag{2}$$

There are only two possible outcomes when a Bitcoin script is executed. It either executes successfully with no errors implying that the transaction is valid. Or in case of any error while the script is executing, the whole transaction will be invalid and shouldnt be accepted into the block chain.
Important Commands:

- **DUP:** Duplicate the top of the stack.
- **HASH160:** RIPEMD-160(SHA1(top of the stack)).
- **PUSHDATA:** Push the # of bytes mentioned after this into the stack.
- **EQUALVERIFY:** Compares top two elements of stack and return true only if they are equal.

[5]http://haelchan.me/images/btc/transaction.jpg

- **CHECKSIG:** Checks that the input signature is a valid signature using the input public key for the hash of the current transaction.
- **CHECKMULTISIG:** Checks that the k signatures on the transaction are valid signatures from k of the specified public keys.
- **Other Control Statements:** IF, NOTIF, NEGATE, AND.

6.1 Example

Let us consider an example, where Amit wants to pay 1BTC to Mohit and later Mohit wants to pay 0.5BTC of it to Rohit. Figure 7 will be transaction block of Amit.

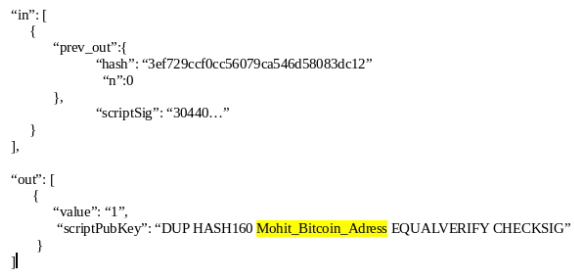

Figure 6: Amit to Mohit Transaction[6]

Say Hash pointer to this is "$0x1234$..."

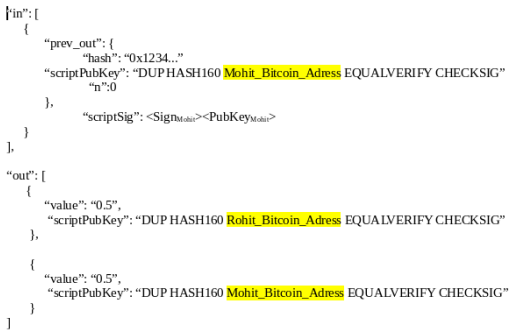

Figure 7: Mohit's Transaction[7]

Mohit redeems this by executing script:

$$< Sign_{Mohit} >< PubKey_{Mohit} > DUP\ HASH160\ BitcoinAdress_{Mohit}\ EQUALVERIFY\ CHECKSIG \tag{3}$$

If the script executes successfully with no errors, then transaction is valid and is accepted into block chain else the transaction is considered invalid.

References

[1] S. Nakamoto, "A peer-to-peer electronic cash system." https://bitcoin.org/bitcoin.pdf.

[2] learningspot, "Proof of work." http://learningspot.altervista.org/incentives-and-proof-of-work.

[6] We drew this image by taking class notes as reference.
[7] We drew this image by taking class notes as reference.

[3] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," 06 2017.

[4] E. F. A. M. S. G. Arvind Narayanan, Joseph Bonneau, "Bitcoin and cryptocurrency technologies.," 10 2015.

[5] T. A. U. Slides, "Mechanics of bitcoin." http://faculty.cs.tamu.edu/bettati/Courses/489CryptoCurrencies/2017A/S