Lightning Network

Ante Sosa
Erasmus exchange student
Student number: 70081480
Faculty of Computer and Information Science
University of Ljubljana

***Abstract*—Bitcoin was created to be decentralized cash system. One of the problems that is keeping Bitcoin from widespread adoption is scalability. Lightning Network is one of the proposed solutions to the scalability problem. The main topic of this paper is Lightning Network which is made of payment channels, to gain more transactions to be processed.**

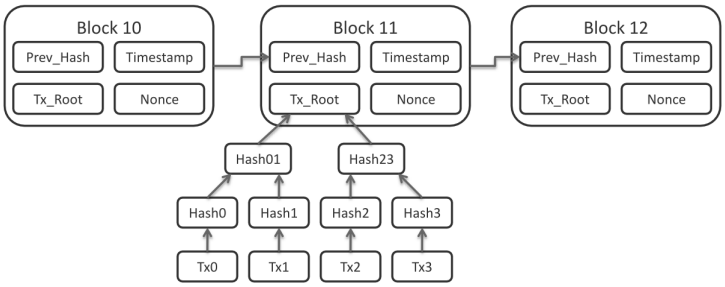
I. Bitcoin

Bitcoin is a peer to peer electronic payment system which allow parties to exchange value without need of third party being involved. Bitcoin solves issues that physical money has:

- Independent - no need of trusted third party.
- Impossible to counterfeit - there is no software that can generate fake coins.
- Inflation - Bitcoin inflation is done through halving mining rewards every 4 years and thus is much more stable then money inflation.
- Limited supply - only 21 million coins will ever exist. (Compared with phisical money which can be printed as much as goverment wants.)
- Fast transactions - Bitcoin transactions take up to few hours, which is very fast compared to bank to bank transfers which last for days.
- Low fees - much less then using bank transfers.

Bitcoin network is made of nodes which are mutually connected. Each node has publicly distributed digital ledger, downloaded locally. The digital ledger is a chain of blocks. Each block has a header and a body. In the block's body information of transactions are stored. In the block's header several information are stored:

- Version (4 bytes) - it is used for checking if the mined block is mined using up to date version of consensus.
- Hash of previous block (32 bytes)
- Nonce (4 bytes) - it is used for miners to generate correct hash.
- Merkle root (32 bytes) - a hash of the root of the merkle tree of this block's transactions.
- Timestamp (4 bytes) - the timestamp of the block in UNIX.
- Difficulty target (4 bytes) - the target for the block.



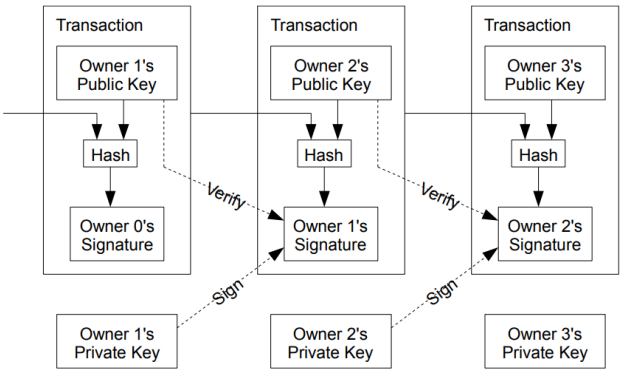
The bitcoin protocol uses a proof-of-work concept for generating new blocks. The proof-of-work involves changing nonce until calculated SHA-256 value of block is less then difficulty target in block header. Once the miner has found correct block hash, he broadcasts that block to other nodes in network and gets mining reward and transaction fees. The longest chain of blocks is considered as the correct one and network will keep working on extending it. To modify past block, attacker needs to do the proof-of-work of that block and all blocks that have been mined after it. Also, attacker would need to keep up with mining new blocks faster then the rest of honest network. If attacker doesn't have more than 51% of networks computational power, it has very low chances to mine more blocks then the rest of network.

II. Transaction

While using Bitcoin software wallet, it seems like coins are moving from wallet to wallet, but actually

they are moving from transaction to transaction. Basically, each transaction input is output of another transaction. Each transaction output is input to exactly one transaction, if that is not the case, same coins are spent twice, which is known as double-spending. Every transaction output stored on Bitcoin blockchain can be divided in two sets:

- Unspent transaction outputs
- Spent transaction outputs



A coin owner sends a coin to the next owner by digitally signing a hash of the previous transaction and the public key of the next owner. A new owner can verify the chain of ownership by verifying the signatures. Only Unspent Transaction Outputs (UTXOs) can be spent. For transaction to be valid it is sufficient just to check if transaction input is unspent. If transaction output exceeds its input, transaction would be rejected, but if transaction input exceeds its output, the difference is transaction fee credited to miner which include that transaction into his block.

III. Scaling problem

When Satoshi Nakamoto published withepaper of Bitcoin, one of the first public comments was that Bitcoin seems not to scale to required size. After a decade of existing, scalability is the main problem for Bitcoin and the other cryptocurrencies. Bitcoin can proceed at most 7 transaction per second, which in the beginning was more than enough, but today transactions take a long time to process and transaction fees are getting higher. If Bitcoin will ever become alternative to Visa, it will need to compete with them in sense of transactions per second. For now, the gap between the two is huge. Visa can process 20 000 transactions per second on average and even 50 000 at its peak. There were various ideas to improve Bitcoin's scaling, but the overall consensus haven't still been met.

IV. Lightning Network

The basic idea which led to construction of Ligthining Network is that not every transaction should be stored on the blockchain. Instead of keeping every record of transaction on the blockchain, the Ligthining Network, adds second layer to Bitcoin's blockchain and enables users to create payment channels.

Two parties who want to use Lightning Network to exchange funds back and forth need to open a multisignature channel by committing a funding transaction to base blockchain, which is also considered as layer 1. That channel is represented as an entry on the Bitcoin public ledger. This is followed by making number of Lightning transactions. Each one of these transactions update the distribution of the channel's funds without broadcasting it to the blockchain (layer 1).

The current balance is stored as the most recent transaction signed by both parties, which are sending funds in the channel. To make a final payment both parties sign an exit transaction and by doing so all old exit transactions became invalid. The Lightning Network does not require both parties to agree about exiting the channel. Both parties have the option to close the channel at any moment if they want to do so. If one party want to close the channel it just need to broadcast the latest transaction signed by both parties to the blockchain.

The network is formed of parties which have multiple payment channels with many different users. Even if two parties are not connected directly, they can exchange funds if there exists a path of payment channels between them in the network.

V. How Lightning Network works?

To understand how Lightning Network actually work, one should understand payment channels and background behind them. Firstly, let us introduce building blocks.

A. Smart contracts

Bitcoin supports Script, a smart contract programming language. Each Bitcoin transaction contains a field in itself which has a small script written in Script. This script checks whether the output of the transaction can be spend. The abilities of Bitcoin's smart contracts are highly limited, because programming language Script is not a Turing-complete language.

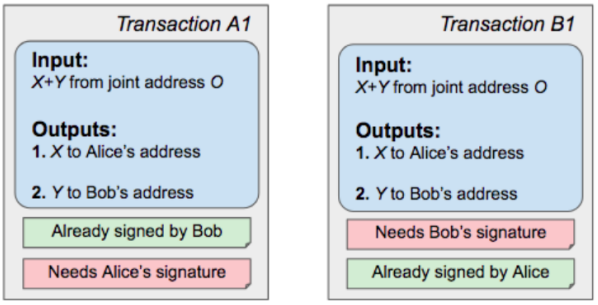
B. Operations

There are a couple of important operations of Bitcoin scripts:

- OP_CHECKSIG - this operation is checking whether some party have the key to spend money from some address, which is previously funded with some transaction.
- OP_SHA256 - computes the hash of the provided paramater.
- OP_EQUALVERIFY - this operation is checking equality. Combining it with OP_SHA256 it can be checked that one party has a secret number, the hash of which is equal to some value.
- OP_CHECKMULTISIG - this operation allows to spend funds from address only when multiple keys sign it. This is used for multiple signature wallet.
- OP_CHECKSEQUENCEVERIFY - this operations allows to spend funds only after some fixed number of blocks.

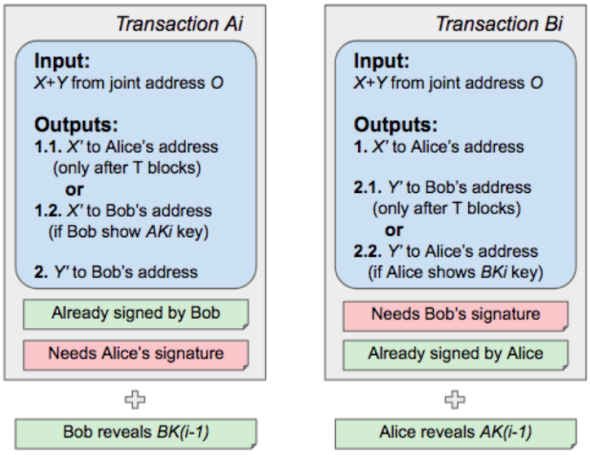
C. Payment channel

Let us consider two parties, called Alice and Bob, want to open the payment channel. Firstly, they need to create transactions, by which they transfer their funds to joined deposit address. The funds from this joined address can be spent only when the transaction is signed with two keys from Alice and Bob. They did not sign transaction yet, because one party can refuse to cooperate and the other party will lose access to his funds. Instead of signing transaction, they both create two new refund transactions(A1 and B1). The input of these transactions is output of the joined address. These transactions has two outputs: x amount of bitcoins is going to be sent to Alice and y to Bob. Then they sign each others transactions and exchange it. Only now they can safely publish initial deposit transaction to the main blockchain. If ether party refuses to cooperate, the other just need to sign his transaction and record it in the blockchain and receive access to his money.



To make a payment in a channel, parties need to rewrite transactions A1 and B1. Let us say that Bob pays to Alice z bitcoins, so Alice agrees to receive $x + z$ and Bob agrees to receive $y - z$. Since, A1 and B1 are changed, now we can consider two newest transactions A2 and B2.

If Alice does not want to cooperate, how can Bob be sure that she will not publish A1 instead of newer transaction A2? This problem is solved by selecting one-time private keys each time they make new transaction. That key can be used by one party to receive the deposit of another party, by using scripting operation OP_CHECKSIG. Each time they make new transaction, Bob sends to Alice his previous private key and creates a new key. Alice does the same with her keys. The party will not accept payment without receiving the previous key from the other party. If Alice knows Bob's private key she can spend all Bob's money, but only in case Bob decides to record an incorrect transaction (not the last one) to the blockchain. The small modification is done using scripting operation OP_CHECKSEQUENCEVERIFY which gives the party, who closed channel, their deposit after some fixed number of blocks (T).



If Alice signs and broadcast her transaction, which is previously signed by Bob two different scenarios are possible:

- Alice is trying to record her old transaction instead of her last one. That means she is trying to cheat, so Bob knows Alice's key and can use it to get all the Alice's funds. Clearly, this case is not interesting for Alice.
- Alice submits her last transaction. In this case Bob does not have Alice's last private key, so this is valid closure of the channel. Finally, Bob should get his unspent money immediately and Alice after some fixed number of blocks.

D. Payment channel with intermediary

Let us consider again Alice and Bob which does not have a direct payment channel but want to exchange funds. They both have payment channels with Charlie, so Alice can send funds to Charlie and Charlie can send funds to Bob and the other way around. The problem that arises is how can Alice be sure that Charlie will forward the funds to Bob. The solution is given by using secret number. After the payment is initiated, Bob selects a secret number e, computes hash of it He and sends it to Alice. Alice creates transaction AC for transferring funds to Charlie but only in case Charlie knows secret e, what she checks by using script operations: OP_EQUALVERIFY, OP_SHA256. Charlie creates similar transaction CB transferring money to Bob only if he provides secret number e. Bob now can see that he will be able to receive funds and shows him secret e. Now Charlie can prove to Alice he has transferred funds and receive transaction AC.

One possible scenario is that Bob decides to cheat by not revealing secret number to Charlie. If Bob wants to get the money, he will need to close the channel with Charlie and record transaction CB on blockchain but to be able to spend the money he will need to show his secret number and Charlie would also see it.

E. Routing

In order for a payment to work over the Lightning Network, at least one path of connected payment channels must exist between a payer and a payee. The routing problem is problem of finding the most optimal path. The cost of each path is determined by various dynamic factors like: network topology, channels throughput, availability of nodes, intermediaries fees. One of the most respected peoples in Bitcoin community Andreas M. Antonopoulos says: "Routing is not a solved problem." He explains that routing is not part of core specification of Lightning Network and is by itself independent of construction of payment channels. He also claims that the problem of routing is not necessary to solve, since the network is so small and the optimization would be just a waste of resources for now.

VI. Implementation

Lightining Network was published in whitepaper, written by Joseph Poon and Thaddeus Dryja in 2005. Today, there are three teams doing the most development of the Ligthining Network: Lightining Labs, Blockstream and ACINQ. Each of the three is using their own implementation of Lightening Network written in different programming languages. Lightining Labs is implementing their version of Lightening Network in Golang, Blockstream in C and ACINQ in Scala. There are plenty of other implementations currently in process of developing. Recent interoperable tests showed that three major implementation can work in the same time with one another without any problems.

VII. Bad consequence of Lightning Network

There is a lot of negative comments on Lightning Network. Here are some of them:

1) The most popular one is that Lightning Network is producing centralization which violates one of the main principles of Bitcoin. While creating payment channels nodes want to connect to large hubs which are well funded and connected to the rest of the network. This can be solved by using algorithm for automatic creating payment channels without node's ability of choosing node to connect.
2) Lightning network can not be used with cold storage. Funds that are locked in payment channels are controlled by online wallets.
3) Efficient routing on large number of nodes is nearly impossible. This problem will come into place when network grows to larger scale. For now, it is just far future.
4) Lightning transactions are not as safe as on-chain transactions. That's why for bigger transfers it is recommended to use on-chain transaction.
5) Node needs to be online to use Lightning Network. This is not entirely true, because if node is offline other nodes can monitor its channels that no one is cheating.

VIII. Conclusion

The Lightning Network also introduces better anonymity of transactions since transactions are done off chain and it becomes very hard to trace them. Although, the Lightning Network is still not mature software and has some bugs, the number of nodes, channels and network capacity are growing from day to day. Today, there are more then 5 500 Nodes, 23 000 channels filled up with more then 620 bitcoins. Development is in very early phase and a huge work should be done before we can use Lightning payments in everyday life.

References

[1] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System (2008) *https://bitcoin.org/bitcoin.pdf*.

[2] Bitcoin developer guide, *https://bitcoin.org/en/developer-guide*

[3] G. D. Stasi, S. Avallone, R. Canonico, G. Ventre, Routing payments on the Lightning Network, University Federico II of Napoli (Italy)

[4] J. Poon, T. Dryja, The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments *https://lightning.network/lightning-network-paper.pdf*