| | | | |
|---|---|---|---|
| Distributed Trust and Blockchains | | Date: | 11/09/18 |
| Instructor: *Sujit Prakash Gujar* | | Scribes: | *Satyam Mittal, Geethika Dudyala, Aditya Arora* |

# Lecture 10: Mechanics of Bitcoin

## 1 Recap

### 1.1 Bitcoin Script Execution

- We discussed about scripts scriptSig and scriptPubKey used to redeem a transaction.
- The ScriptSig contains two components, a signature and a public key. The signature along with the public key is used to prove that the transaction is from the owner of the mentioned address only.
- The ScriptPubKey consists of taking the public key that was pushed on the stack, duplicating it, hashing it and comparing it to the hash of the public key the output was destined for.

### 1.2 Bitcoin payment methods

#### 1.2.1 Pay-to-Public-Key-Hash

- The pay-to-public-key-Hash is the basic form of making a transaction and is the most common form of transaction on the Bitcoin network. Here the transaction specifies an address to which the payment is to be sent in the form of the hash of the public key.

#### 1.2.2 Pay-to-Public-Key

- The pay to public key is a very similar process as the pay to public key hash. The only difference is that the presented data is slightly different as the public key is presented as the elliptic curve point and so the hashing and duplication operators are not needed.

#### 1.2.3 Pay-to-ScriptHash

- Pay-to-Script-hash, or P2SH, is an extension of the multisignature idea but reducing the burden on the Bitcoin infrastructure in terms of storage required.
- The details of the transaction script are not stored in the blockchain, only the hash, and therefore the creator of the transaction must keep a copy of the script or remember the route to generate the specific hash.

### 1.3 Few Applications of Bitcoin Scripts

#### 1.3.1 Escrow transactions

- Escrow just means that when there is a trade between two people, there is a middle man who ensures both parties uphold their parts of the deal.
- Escrow transactions can be implemented quite simply using MULTISIG. Say you want to buy a stuffed teddy bear. You doesn't send the money directly to store owner, but instead creates a MULTISIG transaction that requires you, store owner and one new member (Judy) to sign in order to redeem the coins. Store owner will give the teddy bear. When you receive teddy bear, you and store owner both sign a transaction redeeming the funds from escrow, and sending them to store owner.

- In case of dispute, Judy needs to get involved. Judy's going to have to decide which of these two people deserves the money.

### 1.3.2 Green transactions

- Green transactions are used when one of the user is offline or he doesn't have have time to look at blockchain and he wants to checks whether the transaction is already there. For normal transaction to be part of blockchain, it has to be confirmed by 6 blocks. Clients such as street vendors can not wait for an hour for transaction confirmation and deliver their service to their users.
- To solve this problem, A third party such as bank is introduced. Users will communicate with bank to ask for help regarding transactions. Bank will pay the clients on behalf of users from their bank controlled address, green address. Bank will guarantee it will not double spend the money.
- It puts too much trust on bank. Bank can tell users to look at their transactions history to check whether they double spend. If the bank ever does double-spend, people will stop trusting its green address(es).

### 1.3.3 Micro Payment

- One of the most prominent benefits of using Bitcoin is being able to send money across the world without any hassle. Most traditional financial services are limited when it comes to sending micropayments or large payments with huge fees attached to them, but Bitcoin is very different.
- Say that Alice is a customer who wants to continually pay Bob small amounts of money for some service that Bob provides. We start with a MULTISIG transaction that pays the maximum amount Alice would ever need to spend to an output requiring both Alice and Bob to sign to release the coins. Each time Alice uses the service, Alice will keep sending transactions with added fees upto that time. Eventually, Alice will finish using the service. At this point Alice will stop signing additional transactions and only sign the last transaction. The final transaction that Bob redeems pays him in full for the service that he provided and returns the rest of the money to Alice. Here all the initial transactions that Alice makes are double-spent transactions and only the last transaction is valid.

# 2 Introduction

In this lecture, the following topics were discussed :

- Merkel Tree
- Bitcoin block structure
- CoinBase Transaction
- Ways to join Bitcoin network
- Thin or SPV client
- How to join Bitcoin network
- Role of Bitcoin Node
- UTXO Set Size

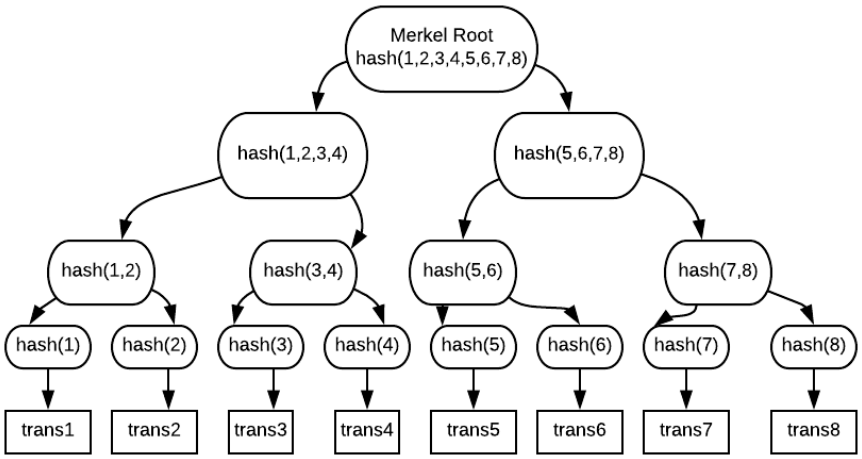


Figure 1: Merkle tree schematic representation

# 3 Merkle Trees

## 3.1 Introduction

- A binary tree built with hash pointers is known as a *Merkle tree*.
- A Merkle tree summarizes all the transactions in a block by producing a digital fingerprint of the entire set of transactions, thereby enabling a user to verify whether or not a transaction is included in a block.[1]

## 3.2 Construction of Merkle tree

- Merkle trees are created by repeatedly hashing pairs of nodes until there is only one hash left (this hash is called the *Merkle root*). They are constructed in a bottom up manner.
- Each leaf node of the tree is a hash of transactional data. Each non-leaf node is a hash of its previous hashes.
- The data blocks are grouped into pairs of two. For each pair, the hash of both of the blocks is stored in a parent node. These parent nodes make the next level up of the tree.
- Subsequently, the parent nodes are grouped in pairs and their hashes stored one level higher. This continues all the way up the tree until there is only one hash left. (Refer to Figure 1)
- Merkle trees are binary and therefore require an even number of leaf nodes. If the number of transactions is odd, the last hash will be duplicated once to create an even number of leaf nodes.

## 3.3 Tampering of data

- If an adversary tries to tamper with a transaction at the bottom of the tree (leaf node), the hash pointer stored one level higher in the tree will not match.
- Though he can try to cover up this change by changing that node as well, any change to a previous transaction would travel up the tree changing each hash along the way. His strategy will fail because he cannot tamper with the *Merkle root* as it is global.

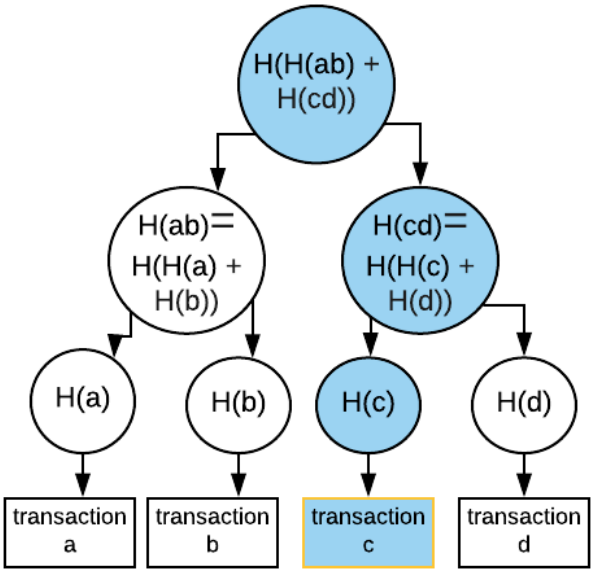


Figure 2: Verification of transaction 'C', reconstruction only of path from root to node

- Thus Merkle tree ensures that the transactions are tamper-proof. It maintains the integrity of the data. If a single detail in any of the transactions or the order of the transactions changes, so does the Merkle Root.

### 3.4 Proof of membership

- Proof of membership means - verifying whether a certain data block is present in the Merkle tree or not.
- For example
  - In Figure 2, we want to verify whether c is present in the tree. We need to be given $H(d)$ and $H(H(a)+H(b))$. Then we can hash $c$ to get $H(c)$, then concatenate hash $H(c)$ with $H(d)$, then concatenate and hash the result of that with $H(H(a) + H(b))$. If the result was the same string as the root hash (this is assumed to be known), it would imply that $c$ is a part of the data in the Merkle Tree.
  - The entire dataset need not be downloaded to verify the integrity of a particular transaction. Only the branch from the root to the piece of data being verified needs to be reconstructed. (the blue nodes in Figure 2)
  - Thus, using a Merkle tree reduces the amount of data that has to be maintained for verification purpose, as it separates the validation of the data from the data itself.
  - *Complexity discussion* If there are n nodes in the tree, only about $log(n)$ items need to be shown. Verification thus runs in time and space that's logarithmic in the number of nodes in the tree.

## 4 Block Structure in Bitcoin

In bitcoin, many transaction are grouped together into a single block.

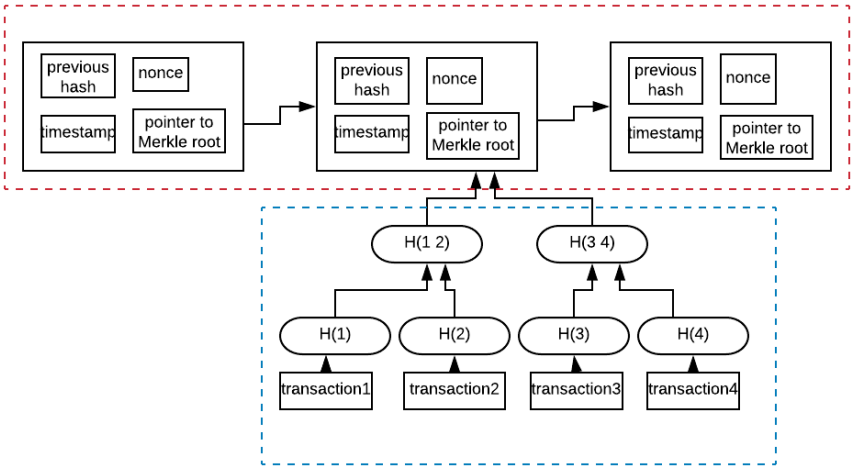


Figure 3: Schematic representation of structure of the blockchain - 2 different hash structures shown

### 4.1 Motivation for grouping transactions into blocks

- Consider the case where one block only had one transaction. Then the miners have to come to a distributed consensus for each block (or in this case transaction) separately. The rate at which new transactions are accepted are very low when compared to the case in which each block has many transactions.
- A hash chain of blocks is much shorter than a hash chain of transactions as many transactions are put into a single block. Hence it becomes more easier to verify the block chain data structure.

### 4.2 Structure

- The block chain is a combination of 2 hash-based data structures. The hash chain of blocks is represented in red dotted lines and the merkel tree structure is represented in blue dotted lines in Figure-3.
- There is a hash chain of blocks. (Refer to the Figure 2) Every block in the chain has a block header, a hash pointer to the previous block in the sequence and a hash pointer to the merkel tree structure.
- The block header contains
  - *nonce* that miners can change
  - a time stamp
  - *bits*, which is an indication of how difficult this block was to find.
- The root of the transaction tree is stored the “mrkl-root” field of the block header.

## 5 Coinbase Transaction

- A coinbase transaction is a unique type of bitcoin transaction that can only be created by a miner.

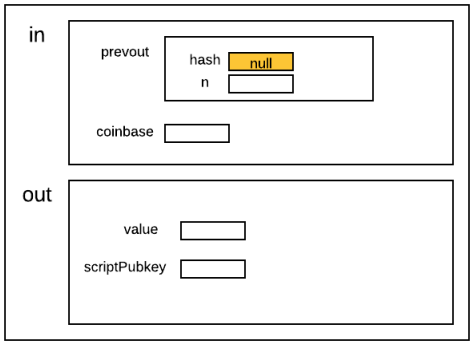


Figure 4: Schematic Representation of a Coinbase transaction

- It is used to create new coins.
- It has a single input and a single output.
- The input doesn't redeem a previous output. As can be seen in Figure-4, the *prevout* field contains a null hash pointer. This is because a coinbase transaction mints new bitcoins and doesn't spend existing coins.
- the value of the output is currently a little over 25 Bitcoins. The output value is the miner's revenue from the block. It consists of two components:
  * a mining reward, which is set by the system and which halves every 210,000 blocks (about 4 years). The Bitcoin block reward is dependent on the number of blocks from the genesis block and the number of fees included in the transactions of the block.
  * the transaction fees collected by the miner from every transaction included in the block.
  * The value of the coinbase transaction is the block reward plus all of the transaction fees included in this block.

# 6 Ways to join Bitcoin network

- There are various approaches to join Bitcoin network.
  - Can join just as a thin client to receive payments.
  - Can join as a miner or a full node.
- Almost 90% run *Core Bitcoin C++*.
- Other Implementation running services:
  - BitCoinJ (Java)
  - libbitcoin (C++)
  - btcd (Go)
- *Original Satoshi Client*: Bitcoin Core is free and open-source software that serves as a bitcoin node (the set of which form the bitcoin network) and provides a bitcoin wallet which fully verifies payments[2].

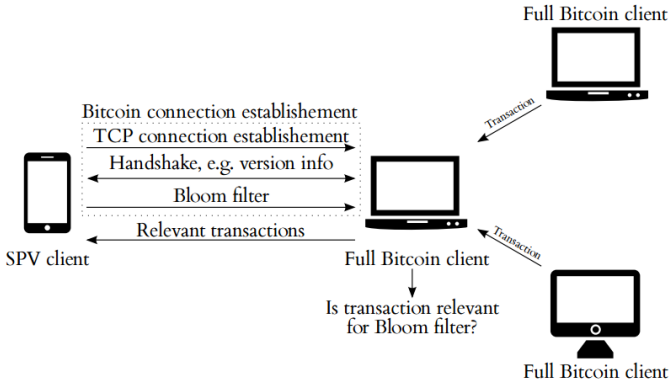


Figure 5: Relationship between SPV and full node[4]

# 7 Thin or Simple Payment Verification (SPV) Clients

**Definition 1** (Lightweight node)**.** *A lightweight node, also called thin or SPV client does not download the whole blockchain, instead they download the block headers only to validate the authenticity of the transactions and proof of work.[3]*

## 7.1 Introduction

- The SPV clients, also known as *lightweight nodes* differ from full nodes by the fact that they don't download the whole blockchain, instead they download the block headers only to validate the authenticity of the transactions.
- They form the majority of the bitcoin network.
- Most of the bitcoin wallets maintains a lightweight node.
- They use the *Simple Payment Verification (SPV)* method to verify the transaction.
- As per the SPV protocol, lightweight nodes can only verify the proof of work by the miner but cannot verify the transactions inside the block as it only stores the header of a block.
- The full nodes allow the lightweight nodes to connect to the bitcoin network and thus they are essential for functioning of a SPV node.
- The SPV nodes are dependent on a full node for verifying the transactions that affect that node as a full node notifies the SPV client about the authenticity of its transactions.

## 7.2 Advantages of a lightweight node

- The cost savings of being a lightweight node are huge. The block headers are only about $\frac{1}{1000}$ the size of the block chain. So instead of storing a few tens of gigabytes, its only a few tens of megabytes. Even a smart phone can easily act as a lightweight node in the Bitcoin network.

## 7.3 Downsides of a lightweight node

- If a lightweight node is not connected to its trusted full node, it is vulnerable to a lot of risks as listed below:
  - The lightweight nodes are not self sufficient to verify its transactions, thus they can accept fake or used bitcoins.

- Mostly a lightweight node sends its addresses to a third party to check wallet balance and history which hijacks their privacy.

**Note** - All above mentioned risks can be avoided if a lightweight node is connected to its own full node.

# 8 How to join Bitcoin network

**Definition 2** (Gossip Protocol)**.** *A gossip protocol is a procedure or process of computer-computer communication that is based on the way social networks disseminate information or how epidemics spread[5].*

- The Bitcoin network is a randomly-wired gossip network. This means that all nodes make arbitrary connections to other peers (using various ways to discover new addresses) using a custom TCP protocol, usually using port 8333.
- To connect to the bitcoin network, you need to contact a seed node i.e. an already existing node in the network and send a special message requesting entry into the network.
- You can repeat the process with the new nodes you learn about as many times as you want.
- You can choose the members to peer with, and then you become a fully functioning member in the network.
- If network finds you are not active for 3 hours, network forgets you.

# 9 Role of bitcoin node

- The most important role of a bitcoin node is to maintain the peer to peer network network of bitcoin by relaying the transactions.
- The full nodes also serve the lightweight nodes by allowing them to transmit their transactions on the bitcoin network thus maintaining decentralization for lightweight nodes.
- The bitcoin nodes also verify the proof of work by miners and the individual transactions inside the block by performing a list of checks as mentioned below
  - Check syntactic correctness
  - Check if block hash satisfies claimed nBits proof of work
  - Check if the transaction inputs are in UTXO (Unspent Transactions Output)
- The nodes will ensure it's not a double spend transaction.
- The nodes will execute the output script of input transaction along with ScriptSig and verify the output of script.
- The nodes perform all these checks and then broadcast the block on the network thus maintaining consensus in the network. If transaction is repeated, It will not broadcast to avoid flooding.

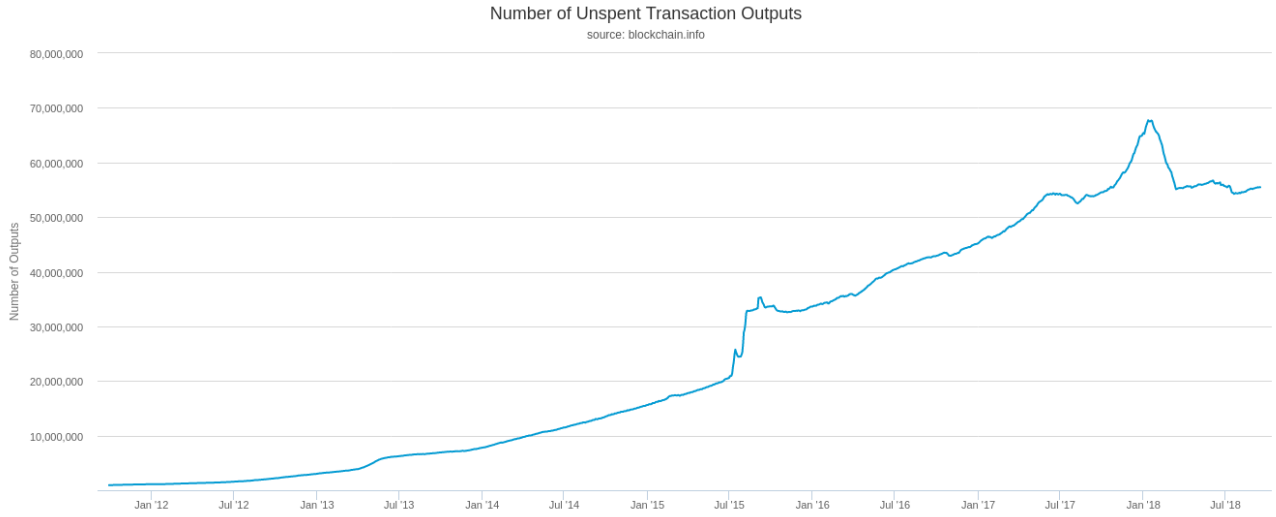


Figure 6: The number of unspent Bitcoin transactions outputs, also known as the UTXO set size.[7]

# 10 Unspent Transaction Output (UTXO) Set

**Definition 3** (UTXO Set)**.** *Bitcoin makes use of the Unspent Transaction Output (UTXO) set in order to keep track of output transactions that have not been yet spent and thus can be used as inputs to new transaction[6].*

- All full nodes need the UTXO set in order to validate transactions so if the UTXO set is too large, it becomes more expensive to retrieve data from the UTXO set which thus increases the costs of running a full node.
- A small UTXO set size allows devices to run a bitcoin full node on low end hardware devices which is beneficial for the strength of the network.

# References

[1] https://hackernoon.com/merkle-trees-181cb4bc30b4

[2] https://en.wikipedia.org/wiki/Bitcoin_Core

[3] J. B. Arvind Narayanan and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.* 2016.

[4] A. Gervaist, "Privacy provisions of bloom filters in lightweight bitcoin clients,"

[5] https://en.wikipedia.org/wiki/Gossip_protocol

[6] G. N.-A. J. H.-J. . Sergi Delgado-Segura, Cristina P´erez-Sol'a, "Analysis of the bitcoin utxo set,"

[7] https://www.blockchain.com/en/charts/utxo count?timespan=all