Ethereum Structure: Difference from Bitcoin Structure

Table of Content

S. No	Topic
1	Introduction
2	What is a Block in Blockchain?
3	What is Block in Ethereum?
4	Structure of Blocks
5	Difference between Ethereum Structure and Bitcoin Structure
6	Difference in Bitcoin's state and Ethereum world state
7	Summing up

Introduction

In blockchain technology, a block is a record of new transactions that have been added to the blockchain. Each block contains a unique code called a "hash" that allows it to be distinguished from every other block, as well as a "hash" of the previous block in the chain, linking the two. This creates a chain of blocks, or a "blockchain," that cannot be altered or tampered with because any change to a block would also change its hash and would therefore no longer match the hash of the previous block. This is what makes blockchain technology secure and tamper-proof.

What is a Block in Blockchain?

In the blockchain, a block is a core concept that can be considered a page in a ledger. Blocks contain transactions and critical data, such as previous hash, that ensure immutability and security in the blockchain network. Each Block stores a previous hash sequentially, so it is almost infeasible to reverse and tamper data. The Block will be created when a miner finds a specific value(nonce) by calculating.

What is Block in Ethereum?

In Ethereum, a block is a collection of transactions and other data that are added to the Ethereum blockchain. Each block contains a unique code called a "hash" that allows it to be distinguished from every other block, as well as a "hash" of the previous block in the chain, linking the two.

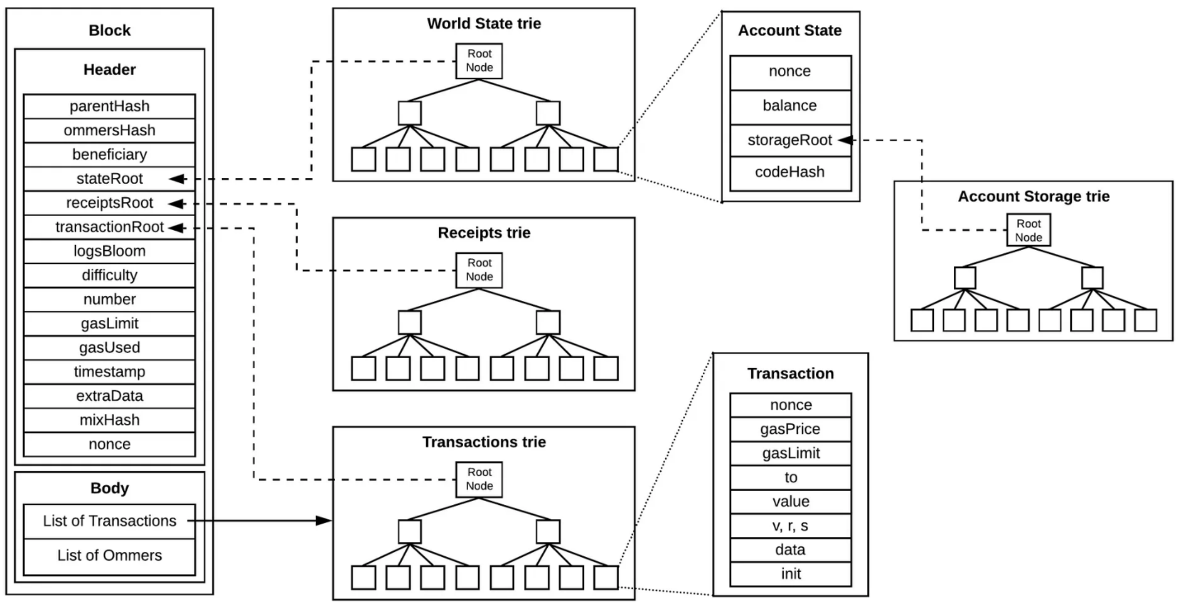
- In addition to transactions, blocks in Ethereum also contain other types of data such as smart contract code and the results of that code being executed. Each block also includes a timestamp and information about the miner who mined the block.
- The blocks in Ethereum blockchain are added through a consensus mechanism called Proof of Stake, which is different from Bitcoin's Proof of Work mechanism.
- Ethereum blocks are mined at a fixed rate of around 15 seconds, which makes the Ethereum blockchain faster than Bitcoin's, which has a block time of 10 minutes.

- An Ethereum block is a collection of transactions that are processed and verified by the network's nodes. Each block contains a block header and body.
- The block also contains the transactions themselves, which are grouped into a single Merkle tree. This allows for efficient verification of transactions without having to include the entire block data in the header.

Structure of Blocks

Every block in Ethereum consists of 2 main parts:

- Header
- Body



Block, transaction, account state objects and Ethereum tries

Header

An Ethereum block header contains several fields that provide information about the block, miner, and current state of the network including:

1. Parent Block's hash: The parent block's hash, also known as the "previous block hash," is a reference to the hash of the previous block in the blockchain. It is included in the header of each block in the Ethereum blockchain and is used to link blocks together in a chain. This creates a tamper-evident and transparent way to verify the integrity of the entire blockchain.

2. Ommers/ Uncle Hash: An Uncle Hash is a reference to the hash of a block that is not included in the main blockchain but is still considered valid. In Ethereum, when a miner finds a new block, other miners may also be working on finding a new block at the same time. If two miners find a new block at the same time, the one whose block gets added to the main blockchain first is called the "main block", while the other is called an "uncle block".

- The Uncle Hash is included in the header of the main block and is a reference to the hash of the uncle block. Uncle blocks are rewarded with a smaller amount of Ether than main blocks, as a way to incentivize miners to continue to mine even if their blocks do not make it into the main blockchain.

- By including the Uncle Hash in the block header, Ethereum allows for the recognition of work done by miners even if the block is not included in the main chain, and also helps to promote network security.

3. State Root: The state root is a reference to the root of the state trie in the Ethereum blockchain. The state trie is a data structure that stores the current state of the Ethereum network, including the balance of all accounts, the storage of all contracts, and the nonce of all accounts. The state trie is a modified version of the Merkle trie, a data structure that allows for efficient verification of the contents of the trie.

- The state root is included in the header of each block in the Ethereum blockchain. It is a 32-byte hash that serves as a summary of the entire state trie at a specific point in time. This allows the entire state of the Ethereum network to be verified by only looking at the state root in the block header, rather than having to download and verify the entire state trie.
- By including the state root in the block header, Ethereum allows for a compact way to verify the integrity of the state of the network and allows for a more efficient way to sync a node with the Ethereum network.

4. Transaction root: The transaction root is a reference to the root of the transaction trie in the Ethereum blockchain. The transaction trie is a data structure that stores all the transactions included in a block. The trie allows for efficient verification of the contents of the transactions in a block.

- The transaction root is included in the header of each block in the Ethereum blockchain. It is a 32-byte hash that serves as a summary of all the transactions in the block. This allows the entire transactions in a block to be verified by only looking at the transaction root in the block header, rather than having to download and verify the entire transaction trie.
- By including the transaction root in the block header, Ethereum allows for a compact way to verify the integrity of the transaction in a block and allows for a more efficient way to sync a node with the Ethereum network. This ensures that the transactions in the block are valid and authorized.

5. Receipt root: The receipts root is a reference to the root of the receipt trie in the Ethereum blockchain. The receipt trie is a data structure that stores the receipts of the transactions included in a block. The receipt for a transaction contains information about the outcome of the transaction, such as whether it was successful, the amount of gas used, and the contract address if the transaction created a new contract.

- The receipts root is included in the header of each block in the Ethereum blockchain. It is a 32-byte hash that serves as a summary of the receipts of all the transactions in the block. This allows the entire receipts of the transactions in a block to be verified by only looking at the receipt root in the block header, rather than having to download and verify the entire receipt trie.
- By including the receipts root in the block header, Ethereum allows for a compact way to verify the outcome of the transactions in a block and allows for a more efficient way

to sync a node with the Ethereum network. This ensures that the receipts in the block are valid and authorized.

6. Logs bloom: The logs bloom is a filter that is included in the header of each block in the Ethereum blockchain. It is used to efficiently check if a log event from a contract execution is included in the block. A log event is a record of an event that occurred during the execution of a smart contract, such as a transfer of funds or a change in the state of the contract.

- The logs bloom is a bit array that is constructed by taking the hashes of the log topics and setting the corresponding bits in the array. The log topics are the indexed data, such as the address of the contract that emitted the event and the event signature. The bloom filter allows for quick checking of whether a log event is included in a block, without the need to scan through all the logs in the block.
- By including the logs bloom in the block header, Ethereum allows for efficient searching of logs events without having to scan through all the logs in the block. This makes it easy to look up events and improves the performance of searching.

7. Difficulty: The difficulty in the Ethereum block header refers to the difficulty level of the proof-of-work algorithm that is used to validate new blocks in the Ethereum blockchain. The difficulty level is a measure of how hard it is to find a valid block, and it is adjusted dynamically based on the current state of the network.

- The difficulty is a value that is adjusted to control the rate at which new blocks are added to the Ethereum blockchain. The goal is to maintain a consistent block time of around 15 seconds. If the block time is too fast, the difficulty is increased, making it harder to find new blocks, and if the block time is too slow, the difficulty is decreased, making it easier to find new blocks.
- The difficulty is encoded in the block header and is a 256-bit value, which represents a very large number. Miners use this value to adjust their mining difficulty in order to find a new block.
- By including the difficulty in the block header, Ethereum allows the network to automatically adjust the mining difficulty to maintain a consistent block time and to ensure that the blockchain remains secure and efficient.

8. Block Number: The number of blocks, also known as the block number, is included in the header of each block in the Ethereum blockchain. It is a scalar value that represents the position of the block in the blockchain. The first block in the Ethereum blockchain, also known as the Genesis block, has a block number of 0.

- The block number is a monotonically increasing value, which means that it increases with every new block that is added to the blockchain. It is used to identify and reference specific blocks in the Ethereum blockchain and to determine the current state of the blockchain.
- By including the block number in the header, Ethereum allows for easy referencing and identification of specific blocks in the blockchain, and it allows for determining the current state of the blockchain. This helps to ensure that the blockchain remains secure and efficient.

9. Gas limit: The gas limit in the Ethereum block header is a scalar value that represents the maximum amount of gas that can be used by the transactions in a block. Gas is the internal pricing mechanism used in Ethereum to pay for the computation of smart contracts and transactions on the Ethereum network.

- Each transaction and smart contract execution requires a certain amount of gas, and the total gas used in a block is limited by the gas limit. The gas limit is set by the miner who creates the block, and it is included in the block header.
- The gas limit is used to prevent a situation where the network becomes overwhelmed by too many transactions, causing delays and increased fees. By including the gas limit in the block header, Ethereum allows for a mechanism to control the rate at which new transactions are processed and to ensure that the network remains secure and efficient.
- The gas limit can be adjusted by the miner who mines the block, and it can be changed based on the current state of the network. The Ethereum protocol also has a mechanism called block gas limit, which is a dynamic algorithm that adjusts the gas limit based on the gas usage of recent blocks.

10. Gas used: The gas used in an Ethereum block header refers to the total amount of gas that was consumed by all the transactions included in the block. Gas is the internal pricing mechanism used in Ethereum to pay for the computation of smart contracts and transactions on the Ethereum network.

- Each transaction and smart contract execution requires a certain amount of gas, and the total gas used in a block is limited by the gas limit. The gas used is included in the block header, and it is a scalar value that represents the total amount of gas consumed by all the transactions included in the block.
- The gas used is an important metric because it helps to determine the state of the network. If the gas used is approaching the gas limit, it may indicate that the network is congested and that transactions are taking longer to be processed.
- By including the gas used in the block header, Ethereum allows for a mechanism to track the total amount of gas consumed by the transactions in a block and to ensure that the network remains secure and efficient. This information can be used to monitor network usage and to make decisions about adjustments to the gas limit or other network parameters.

11. Timestamp: The timestamp in an Ethereum block header is a scalar value that represents the time at which the block was mined. It is a Unix timestamp, which is the number of seconds that have elapsed since January 1, 1970, at 00:00:00 UTC.

- The timestamp is included in the block header, and it is used to order the blocks in the Ethereum blockchain. The timestamp is set by the miner who mines the block, and it is a consensus value, meaning that all nodes in the network should have the same value for the timestamp.
- The timestamp is an important value because it helps to ensure that the blockchain is chronological and that blocks are added in the correct order. It also helps to prevent the possibility of a miner creating multiple blocks at the same time, which would lead to a fork in the blockchain.

- By including the timestamp in the block header, Ethereum allows for a mechanism to order the blocks in the blockchain and to ensure that the blockchain remains secure and efficient. This information can be used to monitor network usage and to make decisions about adjustments to the block time or other network parameters.

12. Extra data: The extra data field in an Ethereum block header, also known as the "extra data" or "extra field," is a 32-byte field that can be used to include additional data in the block header. The extra data field is not used by the Ethereum protocol for any specific purpose and is intended for use by miners or other users of the network. It can be used to include a message, signature, or other data that may be useful for the miner or other users of the network.

- The extra data field is not used by the Ethereum protocol for any specific purpose, and it is not included in the consensus rules, which means that the nodes in the network do not need to validate the contents of the extra data field. Miners can put whatever they want in this field, usually, it's used to put a message, or sometimes it's used to put a miner's address.
- The extra data field is an optional field, and its usage is not required. Miners can choose to leave it empty or to include any data they wish in it. The extra data field is included in the block header, and it is a 32-byte value that can be used to include additional data in the block header.
- By including the extra data field in the block header, Ethereum allows for a mechanism to include additional information in the block header that may be useful for miners or other users of the network. This information can be used for various purposes such as adding a message, signature, or other data that may be useful for the miner or other users of the network.

Body

The body of an Ethereum block, also known as the "block payload" or "block data," is a collection of data that contains all the information necessary to execute the transactions included in the block. The main components of the block body are the list of transactions and the list of uncles (stale blocks).

- The list of transactions in the block body contains all the transactions that were included in the block. Each transaction includes information such as the sender's address, the recipient's address, the amount of Ether to be transferred, and the amount of gas to be consumed.
- The list of uncles in the block body contains all the stale blocks that were included in the block. These stale blocks are included in the block as a reward for the miner who mined them, even though they were not included in the main blockchain.
- The body of the block is not included in the header of the block and is not used for the consensus mechanism but is crucial for the execution of the block's transactions and for the update of the state trie. The block body contains the information necessary to execute the transactions included in the block, and it is used to update the state of the Ethereum network.
- By including the body of the block, Ethereum allows for a mechanism to include all the information necessary to execute the transactions included in the block and to update the state of the Ethereum network. The body of the block contains the list of transactions

and the list of uncles, which are used to update the state of the Ethereum network and to reward miners for their work on the network.

The main fields in the Ethereum block body are:

1. Transactions: A list of transactions included in the block. Each transaction includes information such as the sender's address, the recipient's address, the amount of Ether to be transferred, and the amount of gas to be consumed.

2. Uncles: A list of stale blocks that were included in the block. These stale blocks are included in the block as a reward for the miner who mined them, even though they were not included in the main blockchain.

3. Transactions Root: A field that contains the Merkle root of the list of transactions in the block. The Merkle root is a hash of all the transactions in the block, and it is used to prove that a specific transaction is included in the block without having to include all the transactions in the block header.

4. Uncle Root: A field that contains the Merkle root of the list of uncles in the block. The Merkle root is a hash of all the uncles in the block, and it is used to prove that a specific uncle is included in the block without having to include all the uncles in the block header.

5. Gas Limit: A field that contains the maximum amount of gas that can be used by the transactions in the block. The gas limit is set by the miner who mines the block and is used to prevent the network from being overwhelmed by too many transactions.

6. Gas used: A field that contains the amount of gas that was actually used by the transactions in the block.

7. Block Reward: A field that contains the reward given to the miner who mined the block. This reward is a combination of the block reward, the uncle reward, and the transaction fee rewards.

8. Value: The "value" field in a transaction within the Ethereum block body is used to specify the amount of Ether that is being transferred from the sender to the recipient in the transaction. It is an important field as it represents the value being transferred and it is used to calculate the total value transferred in a block.

- The value field is a 64-bit word (8 bytes) and the unit of value is wei, the smallest unit of Ether. It's important to note that the value field is optional in a transaction, if the value is not specified then it is assumed to be zero.
- The value field is used in combination with the "to" field, which specifies the address of the recipient, to create a standard transaction. The value and the "to" fields are used to transfer Ether from the sender's address to the recipient's address.
- It is important to note that the value field is not used in contract creation transactions because contracts do not have an address and therefore cannot receive Ether.
- The value field plays a key role in the functionality of the Ethereum network, as it represents the amount of Ether that is being transferred between addresses, which is used to facilitate various types of transactions and smart contract execution.

9. Data: The data field in the block body contains all the transactions that were included in the block by the miner. These transactions can include various types of transactions such as contract creation, contract execution, and token transfer.

- It's important to note that the block body only contains the transaction data and not the transaction results, the transaction results are stored in the "receipts" field which is included in the block header.
- The data field plays a critical role in the functionality of the Ethereum network, as it stores the transactions that are executed on the Ethereum blockchain, which are used to transfer value, create and execute smart contracts, and more.

10. To: The "to" field in a transaction within the Ethereum block body is used to specify the recipient address to which the Ether is being transferred. It is an important field as it represents the address of the recipient, and it is used to calculate the total value transferred to a particular address in a block.

- The "to" field is a 20-byte address in Ethereum, which is a unique identifier for an Ethereum account. It can be a user account or a smart contract.
- The "to" field is used in combination with the "value" field, which specifies the amount of Ether that is being transferred, to create a standard transaction. The "to" and "value" fields are used to transfer Ether from the sender's address to the recipient's address.
- It is important to note that the "to" field is not used in contract creation transactions because contracts are created by the sender and don't have an address. In this case, the "to" field is set to null. The "to" field plays a key role in the functionality of the Ethereum network, as it represents the address of the recipient, and it is used to facilitate various types of transactions and smart contract execution, including transfer of value and contract execution.
- These fields in the block body are used to execute the transactions included in the block, and it is used to update the state of the Ethereum network. The block body contains the information necessary to execute the transactions included in the block, and it is used to update the state of the Ethereum network.

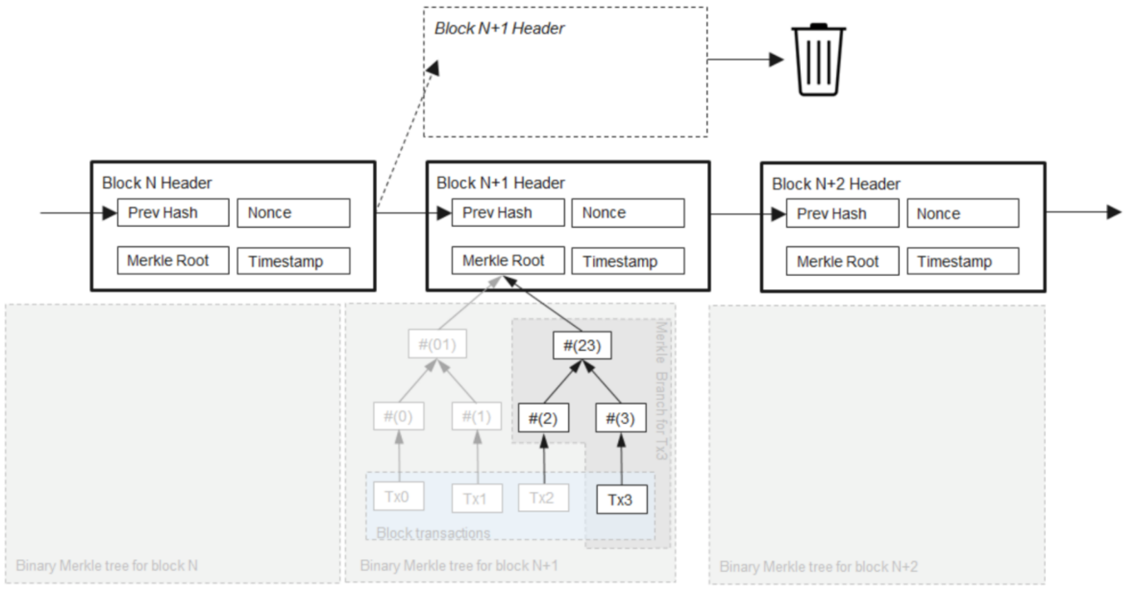
In short:

- A Block comprises of Block Header and transactions.
- Block Header comprises of 15 fields.
- State, Transactions, Receipts header fields have pointers to the root hash.

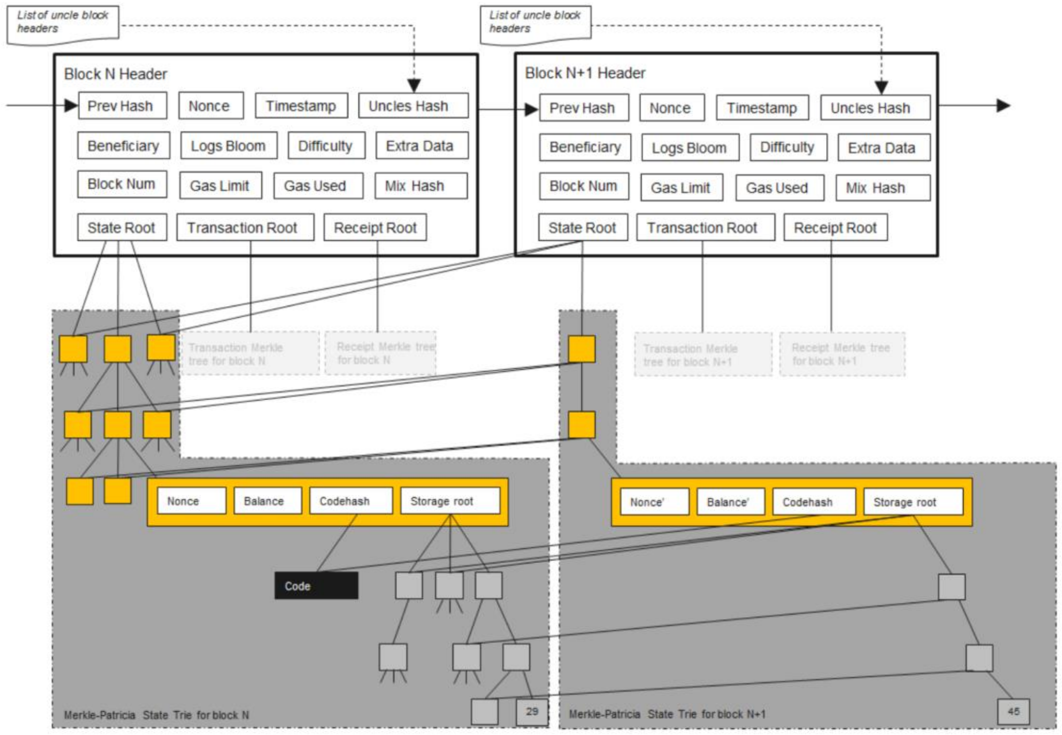
Difference between Ethereum Structure and Bitcoin Structure

The Figures given below diagrammatically show how Ethereum Block Structure differ from Bitcoin Block Structure:

Bitcoin Block Structure



Ethereum Block Structure


List of uncle block headers
Block N Header
Block N+1 Header
Prev Hash
Nonce
Timestamp
Uncles Hash
Beneficiary
Logs Bloom
Difficulty
Extra Data
Block Num
Gas Limit
Gas Used
Mix Hash
State Root
Transaction Root
Receipt Root
Transaction Merkle tree for block N
Receipt Merkle tree for block N
Transaction Merkle tree for block N+1
Receipt Merkle tree for block N+1
Nonce
Balance
Codehash
Storage root
Code
29
45
Merkle-Patricia State Trie for block N
Merkle-Patricia State Trie for block N+1

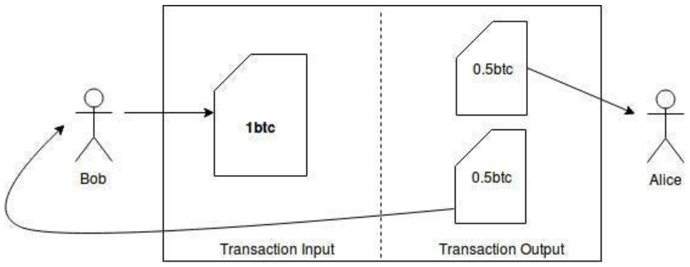
Difference in Bitcoin's state and Ethereum world state

Bitcoin State

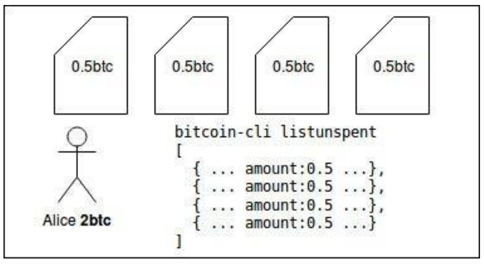
Bitcoin's "state" is represented by its global collection of Unspent Transaction Outputs (UTXOs). The transfer of value in bitcoin is actioned through transactions. More specifically, a bitcoin user can spend one or more of their UTXOs by creating a transaction and adding one or more of their UTXOs as the transaction's input.

The following two bitcoin examples will provide contrast between bitcoin's UTXO model and the concept of Ethereum's world state.

Firstly, bitcoin UTXOs cannot be partially spent. If a bitcoin user spends 0.5 bitcoin (using their only UTXO which is worth 1 bitcoin) they have to deliberately self-address (send themselves) 0.5 bitcoin in return change. If they don't send themselves change, they will lose the 0.5 bitcoin change to the bitcoin miner who mines their transaction.



Secondly, at the most fundamental level, bitcoin does not maintain user account balances. With bitcoin, a user simply holds the private keys to one or more UTXO at any given point in time. Digital wallets make it seem like the bitcoin blockchain automatically stores and organizes user account balances and so forth. This is not the case.



```
bitcoin-cli listunspent
[
  { ... amount:0.5 ...},
  { ... amount:0.5 ...},
  { ... amount:0.5 ...},
  { ... amount:0.5 ...}
]
```

A user account balance in bitcoin is an abstract notion. Realistically a user's account balance is the sum total of each individual UTXO (for which that user holds the corresponding private key). The key[s] which a user holds can be used to individually sign/spend each of the UTXOs.

The UTXO system in bitcoin works well, in part, due to the fact that digital wallets are able to facilitate most of the tasks associated with transactions. Including but not limited to:

- handling UTXOs
- storing keys
- setting transaction fees
- providing return change addresses
- aggregating UTXOs (to show available, pending and total balances)

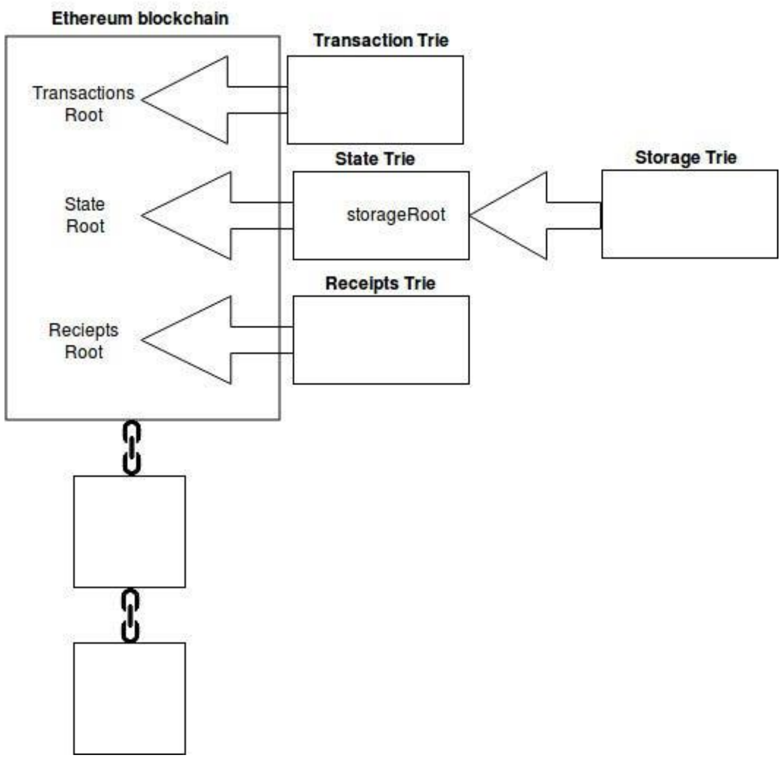
Interestedly, a back-up of a non-deterministic wallet (like the bitcoin core wallet pictured above) only provides a snapshot of the UTXOs (at that point in time). If a user performs any transactions (sending or receiving), the original back up, which they made, will be out of date.

Ethereum World State

In contrast to the information above, the Ethereum world state is able to manage account balances, and more. The state of Ethereum is not an abstract concept. It is part of Ethereum's base layer protocol. As the yellow paper mentions, Ethereum is a transaction-based "state" machine; a technology on which all transaction-based state machine concepts may be built.

Let's start at the beginning. As with all other blockchains, the Ethereum blockchain begins life at its own genesis block. From this point (genesis state at block 0) onward, activities such as transactions, contracts and mining will continually change the state of the Ethereum blockchain. In Ethereum, an example of this would be an account balance (stored in the state trie) which changes every time a transaction, in relation to that account, takes place.

Importantly, data such as account balances are not stored directly in the block headers of the Ethereum blockchain. Only the root node hashes of the transaction trie, state trie and receipts trie are stored directly in the block headers. This is illustrated in the diagram below.



Above diagram shows that the root node hash of the storage trie (where all of the smart contract data is kept) actually points to the state trie, which in turn is hashed and stored in the block header. There are two vastly different types of data in Ethereum; permanent data and ephemeral data. An example of permanent data would be a transaction. Once a transaction has been fully confirmed, it is recorded in the transaction trie; it is never altered. An example of ephemeral data would be the balance of a particular Ethereum account address. The balance of an account address is stored in the state trie and is altered whenever transactions against that particular account occur. It makes sense that permanent data, like mined transactions, and ephemeral data, like account balances, should be stored separately. Ethereum uses trie data structures, to manage data.

There are many Merkle Tries (referenced in each block) within the Ethereum blockchain:

- **World State Trie:** The world state trie contains the mapping between addresses and account states. The hash of the root node of the world state trie is included in a block (in the stateRoot field) to represent the current state when that block was created. We only have one world state trie.

- **Storage Trie:** The account storage trie contains the data associated to a smart contract. The hash of the root node of the Account storage trie is included in the account state (in the storageRoot field). We have one Account storage trie for each account.
- **Transaction Trie:** The transaction trie contains all the transactions included in a block. The hash of the root node of the Transaction trie is included in the block header (in the transactionsRoot field). We have one transaction trie per block.
- **Receipts Trie:** The transaction receipt trie contains all the transaction receipts for the transactions included in a block. The hash of the root node of the transaction receipts trie is included in also included in the block header (in the receiptsRoot field); We have one transaction receipts trie per block.

Summing Up

In this chapter, we have not only covered Ethereum Block Structure, but also various component of Ethereum and how they differ from Bitcoin, like:

- World state: the hard drive of the distributed computer that is Ethereum. It is a mapping between addresses and account states.
- Account state: stores the state of each one of Ethereum's accounts. It also contains the storageRoot of the account state trie, that contains the storage data for the account.
- Transaction: represents a state transition in the system. It can be a funds transfer, a message call or a contract deployment.
- Block: contains the link to the previous block (parentHash) and contains a group of transactions that, when executed, will yield the new state of the system. It also contains the stateRoot, the transactionRoot and the receiptsRoot, the hash of the root nodes of the world state trie, the transaction trie and the transaction receipts trie, respectively.

Reference

https://medium.com/cybermiles/diving-into-ethereums-world-state-c893102030ed

https://consensys.io/blog/ethereum-explained-merkle-trees-world-state-transactions-and-more

https://medium.com/@eiki1212/ethereum-block-structure-explained-1893bb226bd6

https://www.geeksforgeeks.org/ethereum-block-structure/