Lecture 2: Structure of the blockchain and how cryptography plays a role

Total de pontos 11/48

Let's consider the motivation for why something like Bitcoin exists:

The root problem with conventional currency is all the trust that's required to make it work. The central bank must be trusted not to debase the currency, but the history of fiat currencies is full of breaches of that trust. Banks must be trusted to hold our money and transfer it electronically, but they lend it out in waves of credit bubbles with barely a fraction in reserve. We have to trust them with our privacy, trust them not to let identity thieves drain our accounts. - Satoshi Nakamoto

He worked on Bitcoin during the Great Financial Crisis as evident by posting the whitepaper on 31st October 2008 and the code on 3rd January 2009. Banks were at the heart of this crisis. Briefly, they sold mortgages to citizens with bad credit ratings with the expectation that if the price of houses kept going up, then both parties could profit. The profit did not materialise when the housing market's bubble burst in 2006 and instead it triggered defaults as the initial loans were worth more than the purchased house. The toxic loans and risk spread throughout the markets (i.e. securities, pension funds) and nearly took them down. As Satoshi Nakamoto identifies, this was all possible because banks can lend more money than they held in deposits and in a way they are effectively gambling with our deposits. Even worse and to Satoshi Nakamoto's frustration, the banks got away with it as national governments bailed them out:

The Times 03/Jan/2009 Chancellor on brink of second bailout for banks. - Genesis Block

Satoshi Nakamoto was clearly versed in the Cypherpunk's Manifesto1 which focuses on how cryptography can be used to remove power-imbalances in society.

We can speculate that during the financial crisis, Satoshi Nakamoto spent time working out how cryptography can build a new global currency that empowers the individual and not any single nation state. In spite of 30+ years research, Satoshi Nakamoto was the first to work out how to replace a central authority (currency issuer) with a peer-to-peer network. His discovery led to the blockchain which is a cryptographic audit log that lets anyone compute the contents of a database and Nakamoto Consensus that lets financially motivated peers compete to update the database (and thus remove the need for any appointed authority).

For this homework, we focus on how cryptography provides integrity to the blockchain, how the database is structured and how transactions are processed.

Email *

lalanda.alberto@gmail.com

Two Cryptographic Primitives

✗ What is a hash function? * ···/2

An hash function is a way to transform an input message (preimage) into a fixed size unpredictable output (hash). The hash gives no information on the preimage; if the preimage changes 1 bit the hash will be completely different, it will give no clues on the chage that was made; the hash fuction is deterministic, the same input always gives the same output; and it is required that different preimage have different hashes on a 1:1 match relation. ✗

Feedback

A hash function is simply a one-way function. It takes an input of arbitrary sized, and computes a fixed, pseudo-random output.

✗ Identify and explain the three properties that make a hash function cryptographic? * ···/6

Preimage resistance, given a hash H, it should be computationally difficult to find the pre-image m such that h = H(m).

Second-preimage resistance, with the information of an input, m1, it should be difficult to find a different input m2, that will have the same hash, H(m1) === H(m2).

Collision resistance, it should be difficult to find any two messages, m1 and m2, such that H(m1) === H(m2).

Feedback

Preimage Resistance: Given a hash h, it should be difficulty to compute the pre-image x. In other words, the hash gives no clue to the preimage.

Second Preimage Resistance: Whoever computes h = H(x), should not be able to compute another preimage x' such that h = H(x'). In other words, it can act as a real commitment to a secret value.

Collision resistance: Given any hash h, it should be impossible to compute any pair of inputs x, x' that corresponds to h. In other words, there can never be a collision!

✗ What is a Merkle tree and a Merkle tree root? * ···/2

A merkle tree is a tree in which every leaf (node) is the hash of an item in a list. Every inner node is the hash of its child nodes. The end result is a single root (merkle root) that represents a commitment to all items in the list. A ordered list of transactions will always have the same merkle root. If the order of the list is changed the merkle root will change. ✗

Feedback

A merkle tree has a list of leaf nodes representing real data items (i.e. trans- actions), and every non-leaf node is a hash of its child nodes. A merkle tree root is simply a commitment to the entire tree and every data item.

✓ How are the blocks chained together to form the blockchain? * 1/1

- ○ Every hash in the chain is the hash of the previous block
- ◉ Every hash in the chain is the hash of the previous block header ✓
- ○ Every hash in the chain is the hash of the previous list of transactions

✗ What information is required for a light client to verify the inclusion of a transaction? * 0/1

- ◉ Block header, list of transactions for the block ✗
- ○ Block header, transaction included in the block
- ○ Block header, merkle branches of the tree, and the transaction
- ○ Block header, merkle branches of the tree, and all transactions

Resposta correta

- ◉ Block header, merkle branches of the tree, and the transaction

✓ What field of the block header is used by a light client to verify a transaction was included in a block? * 1/1

- ○ Nonce
- ○ Logs
- ○ Previous block hash
- ◉ Transaction merkle root ✓
- ○ Access list
- ○ Payload

Why does simplified payment verification in Bitcoin have weaker security guarantees compared to a full node who processes the entire blockchain? Check all that apply. *

- ○ SPV client cannot follow the longest and heaviest chain
- ◉ SPV client must trust the block producers to protect the integrity of the blockchain
- ○ SPV client cannot verify whether a transaction was included in a block
- ○ SPV client cannot verify the validity of a transaction in the blockchain

✗ What is a digital signature? And what digital signature algorithms are typically used in cryptocurrencies? * ···/2

A digital signature is used for the authentication of the data on cryptocurrencies. It relies on public key cryptography where the user generates a private key d and a public key P. The digital signature should provide: Authentication, Integrity of message and non-repudiation. Anyone should be able to verify that the associated public key has indeed digitally signed the message and that the message was not tampered with. The digital signature algorithms most typically used in cryptocurrencies are Elliptic Curve Digital Signature Algorithm (ECDSA) and Schnorr Signature. ✗

Feedback

A digital signature is an electronic, encrypted, stamp of authentication on digital information such as email messages, macros, or electronic documents. A signature confirms that the information originated from the signer and has not been altered. It relies on a private key to produce signatures. Algorithms include ECDSA and Schnorr Signature.

✗ Why is it important to always use fresh randomness when computing a digital signature? * ···/1

If the same value is used for multiple signatures it is possible to break the signature scheme and derive the private key. ✗

Resposta correta

Correct answer

Feedback

The randomness used in a digital signature (r,s) is public knowledge. Anyone can detect if two signatures rely on the same randomness. If they do, then it is straight-forward to break it by computing the private key.

UTXO model (Bitcoin)

✗ What is the role of an input and output in a Bitcoin transaction? * ···/2

In bitcoin the transaction input identifies a row in the database and provides evidence that should satisfy the spending script. If valid, the row in the database is deleted. A transaction output creates a new row in the database. It is associated with a set of coins and a script. ✗

Feedback

An output specifies the spending conditions before a coin can be spent, and an input specifies the cryptographic evidence of why the spender is authorised to spend the coin.

✗ Why is there no such concept of a "Coin" in Bitcoin? * ···/1

In truth in Bitcoin there are only unspent transaction outputs with some number of bitcoins assossiated with each entry. For the bitcoins to be sent someone must prove evidence to sastify the spending script. ✗

Feedback

A transaction can combine the coins from all inputs into a single output or split the coins into several outputs. As such, there is no fixed coin that we can follow.

✓ Pick 3 operations supported in Bitcoin script. * 3/3

- [x] Digital signature ✓
- [ ] Decrypt value X
- [x] check X is the pre-image of a hash ✓
- [x] If statements ✓
- [ ] Loop

✓ Can we represent the condition "TRUE after time T" in Bitcoin script? * 1/1

- (●) Yes ✓
- ( ) No

Feedback

Yes we can! You can use OP_CSV for it
https://bitcoin.stackexchange.com/questions/38845/what-does-op-checksequenceverify-op-csv-do/38846#38846

✗ Let's pretend 10 transactions are accepted into a block, and each transaction has an output that sends 1 coin to the bitcoin address A. How many entries in the UTXO set will be recorded? * 0/1

- (●) 1 ✗
- ( ) 10

Resposta correta

- (●) 10

Feedback

Every transaction creates at least one new unspendable transaction output (UTXO) in the database.

✗ Why is it difficult to assume if a transaction is spending two or more sets of coins, then they all belong to the same user? * ···/2

One transaction can have more than one set of inputs and create more than one output, but, that same transaction can be signed by more than one entity.

Feedback

Multiple users can collaborate to combine their inputs into a single trans- action. This can be done in such a way where users don't need to trust each other, and can be used to make it difficult to link the inputs/outputs.

✗ What technique takes advantage of this difficulty to help obscure coin ownership? * ···/1

.

Feedback

CoinJoin

Ethereum's account model

✓ What information is stored for an account on Ethereum? * 5/5

- [x] Account hash ✓
- [ ] Transaction hash
- [ ] Script
- [x] Nonce ✓
- [x] Storage ✓
- [x] Code ✓
- [x] Balance ✓
- [ ] Private key
- [ ] NFT

✗ What is a replay attack and why does the transaction nonce stop it? * ···/2

A replay attack is when a transaction that occourred in the past is replayed. The nonce, a number sent in every transaction, protects from replay attacks since it is impossible to send a transaction with the same nonce of a previous transaction, for a specific account.

Feedback

A replay attack lets an attacker re-submit the same transaction to the network several times and as a result it is executed several times. The transaction nonce ensures that a transaction is only accepted once into the blockchain.

✗ When processing an Ethereum transaction that interacts with a contract, what basic validation checks are performed on the transaction and how does the software update the account database? * ···/6

The first check done is of the balance of the transaction signer, to check if is has sufficient balance to cover the cost of the transaction. Then the trasaction is processed and there are three outcomes possible, Success and failed, because of lack of gas or the conditions of the smart contract were not satisfied. The account database is updated, charging the user for the gas (always happens), and updating the accounts rows changed by the transaction (if the transaction is successfull).

Feedback

*Balance check. Does the user have a sufficient balance to cover the maximum fee? i.e. userBalance > gasPriced * gasLimit. Does this trans- action have a larger nonce than what is already stored for the user? i.e. to prevent replay attacks.*

Update contract storage. The contract's function is executed to compute the contract's new state. If the transaction does not run out of gas, then the database is updated to store the contract's new state. i.e. if I play the winning move in a game, then the outcome (I've won) should be recorded in the contract! If the transaction runs out of gas while executing the function, it simply fails and there is no update for the contract.

Update user's account. The transaction signer's balance should be up- dated to deduct the transaction fee (and any coins sent to the smart con- tract). This transaction's nonce is also stored (i.e. to prevent replay attacks).

✕ What are the subtle differences between Bitcoin's UTXO model and Ethereum's account-based model? * ···/8

Updating the database
UTXO - A new entry for every new payment
Account based - Updates the data stored in the same entry
Stateness
UTXO - Transaction is self-contained and it does not execute on shared data.
Account based - Transaction has self-contained data and fetches data from the database during execution.
Replay protection
UTXO - Database entry is deleted upon transacting.
Account based - Nonce for an account is incremented.
Parallel execution
UTXO - All inputs of a transaction can be executed independently of each other.
Account based - Transactions with strictly different access lists can be executed in parallel.
Mental model for coding
UTXO - Consume and produce a series of scripts
Account based - Stateful programs (smart contracts)
Number of signers
UTXO - A single transaction can include payments from one or more signers.
Account based - Only a single party can pay the gas for the transaction, but they can carry execution instructions for anyone (depending on smart contract implementation).
Failed transactions
UTXO - A transaction is successful or it is not included in the blockchain.
Account based - A transaction's execution can fail and the user is still charged for it.

Feedback

Any four of the following (and there are more on the relevant presentation deck):

Number of signers.
In Bitcoin's UTXO model, there can be one or more signers per transaction. So far, Ethereum's account-based model only sup- ports one signer per transaction.

Evidence supplied when spending a coin.
In Bitcoin, the signer needs to specify which coins they are spending (i.e. what unspent transaction output do we care about?). In Ethereum, no evidence is required - we just deduct from the user's balance.

Entries in the database.
In Bitcoin, the balance of a user is not aggregated and every time a user receives a coin it will create a new entry in the database. In Ethereum, the balance of the account is simply updated.

Protection against replay attacks.
In Bitcoin, there is no risk of a replay attack. If a transaction is signed, then it will always spend the desired inputs and always have the same outcome. In Ethereum - a nonce is required to prevent the same transaction being accepted more than once.

Stateless vs Stateful.
In Bitcoin - the state and transaction are intertwined. When we spend an UTXO - the state is destroyed. While there are techniques like Covenants that may someday allow an

unspent output restrict how a future output is created, right now it doesn't exist in Bitcoin. In Ethereum - the state and transaction are not intertwined. All state is stored in the database, and the contract is responsible for self-enforcing how the state can be transitioned.