Lesson 14

ADDRESSES, TRANSACTIONS, PROOF OF WORK

Laura Ricci
21/04/2017
COMMON TYPES OF PAYMENTS

Common characteristic?

Trust to a financial institution
TRUSTING TO A CENTRAL AUTHORITY
THE BITCOIN REVOLUTION

Dipartimento di Informatica
Università degli Studi di Pisa
Laura Ricci
Transactions, Addresses,
Proof of Work
BATCH: THE ORIGIN

• design of the protocol released in 2008 under a pseudonym Satoshi Nakamoto.

• Satoshi Nakamoto releases his/her white paper “Bitcoin: a Peer to Peer Electronic Cash System” in October 2008:
  • idea for a purely peer-to-peer version of electronic cash to the world
  • he/she manages to solve the problem of money being copied, solving a foundation problem for Bitcoin to grow legitimately.
BITCOIN: A BRIEF HISTORY

• January 2009:
  • the first block, the “Genesis Block” is launched allowing the initial “mining” of Bitcoins to take place.
  • later that month, the first transaction takes place between Satoshi and Hal Finney, a developer and cryptographic activist.

• May 2010: first known Bitcoin purchase for real goods
  • Laszlo Hanyecz, from Florida, offered, on the Bitcointalk forum, 10,000 BTC to whom would have delivered him “a couple of pizzas”
  • The request was satisfied from a guy from the west coast, who received 10,000 BTC in exchange for $25 worth of pizza.
THE "PIZZA TRANSACTION" ON BLOCKCHAIN.INFO

Transazione: Ottieni informazioni su una transazione bitcoin

<table>
<thead>
<tr>
<th>Sommario</th>
<th>Input e Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensione 23620 (byte)</td>
<td>Totale Input 10,000.99 BTC</td>
</tr>
<tr>
<td>Ora di Ricezione 2010-05-22 18:16:31</td>
<td>Totale Output 10,000 BTC</td>
</tr>
<tr>
<td>Incluso nei Blochi 57043 (2010-05-22 18:16:31 + 0 minuti)</td>
<td>Tasse 0.99 BTC</td>
</tr>
<tr>
<td>conferme 405698 conferme</td>
<td>Costo per byte 4,191.363 sat/B</td>
</tr>
<tr>
<td>Inoltrato dall'IP 0.0.0.0 (whois)</td>
<td>Stima dei BTC scambiati 10,000 BTC</td>
</tr>
<tr>
<td>Visualizza Osserva il Grafico ad Albero</td>
<td>Script Mostra gli script e la coinbase</td>
</tr>
</tbody>
</table>
BITCOIN: MOTIVATION

• why it is worth using Bitcoin? why don't use PayPal or electronic credit cards?

• some peculiar characteristics of Bitcoin:
  • anonymity & privacy
    • possibility of transacting without identity disclosures
    • public keys as pseudonyms
    • like “paying cash”,
    • risk: exploit the cryptocurrency for illicit purposes
  • openness: everyone having a connection to the Internet can participate to Bitcoin
    • all you need is a Bitcoin client or a third party offering the service, no banking account, no credit card
  • small fees
BITCOIN: INTRODUCTION

- some peculiar characteristics of Bitcoin:
  - decentralization
    - no central trusted authorities that process transactions
    - transactions do not go to a third party
    - no centralized entity controls the money supply
    - disadvantages: no central authority defending against the classical security threats:
      - fraud
      - double spending

- Bitcoin have to solve these security threats without the ability to have trust of anyone else on the network.
  - solution: exploiting a lot of cryptographic techniques: cryptocurrency
  - validation performed by
BITCOIN: MOTIVATION

- Least, but not last: more and more people accept transactions payed in Bitcoin over the last years
  - a real Bitcoin economy
  - existence of Bitcoin exchangers: place where bitcoin are exchanged for mainstream fiat currencies
    - MT.Gox: closed in February 2014
    - other exchangers: CoinDesk, BPI, Bitstamp, Bitfinex, Coinbase, itBit, OKCoin
  - current price: 1,186.33 $
  - a big fluctuation in exchange value

- Note: Bitcoin, the system, bitcoin, the currency
BITCOIN: PRICE FLUCTUATION

- Highest price: 1216 USD (17 Nov. 2013) and 1150 Dec 2016
- https://blockchain.info/it/charts/
BITCOIN: TOTAL VALUE OF BITCOIN

- **Total value:** = 6,282,925 (20 Apr. 2017)
- [https://blockchain.info/it/charts/](https://blockchain.info/it/charts/)
BITCOIN: TOTAL NUMBER OF BITCOINS

- Total number = 20,638,607,438 (19 Apr. 2017)
- https://blockchain.info/it/charts/
BITCOIN: INTRODUCTION

- Economics
- Cryptography
- Distributed systems
BITCOIN ADDRESSES

• a Bitcoin **address** is derived from the public key
  • through a transformation to reduce the length of the public key
  • network identifier byte, hash of the public key, checksum
  • 20 byte address

• an address is used to identify the recipient of the funds in bitcoin

• the private/public key is used to uniquely identify the owner of the funds
  • verify that a transaction is really created by the owner of the funds

• each user may generate any number of pairs (private key, public key)
  • may have more addresses
  • increases anonymity
TRANSACTIONS

- Centralized digital currencies account-based
  - Alice account number is 43569 and the current balance is 300 EUROs

- Bitcoin does not exploit accounts, but records only transactions
  - move value from transaction inputs to transaction outputs.

- Input of the transaction defines where the coin value is coming from:
  - usually a previous transaction’s output

- Outputs from one transaction can be used as inputs in a new transaction, thus creating a chain of ownership as the value is moved from address to address
BITCOIN TRANSACTIONS

- transaction represent funds exchange between Bitcoin addresses.

- each transaction is composed by two lists
  - **TxOut**, a list of transaction outputs.
  - **TxIn**, a list of transaction inputs
TRANSACTIONS OUTPUTS

each output holds:

- the recipient address
- an amount (the value in parenthesis)
- a spending condition: determines the conditions that need to be met in order for a transaction to be spent.
  - most common condition: presence of a valid signature
TRANSACTIONS INPUTS

- transaction input: a tuple consisting of
  - a reference to a previously created output
  - hash of the transaction that created the output
  - index of the output within that transaction
  - arguments to the spending condition: to verify that the transaction creator has the permission to spend that output
A possible scenario

- Alice, may have received two payment from two friend recorded in Transaction1 and Transaction2.
- Output of these transactions are sent to two different Alice's addresses
- Alice performs Transaction 3 to pay something, taking the change for herself
Each transaction completely uses the input funds: no change is left in the input addresses.

Change = difference between input sums and the sum we actually want to pay including fees

• can be kept by using an owned address between the outputs
BITCOIN TRANSACTIONS VALIDITY

- A first condition for validity: \( \Sigma(\text{input funds}) \geq \Sigma(\text{output funds}) \). The transaction must not spend more than the available inputs

- \( \Sigma(\text{input funds}) - \Sigma(\text{output funds}) = \text{transaction fee} \).
  - collected by the miners as a fee to include the transaction in a block
  - paying a fee is optional
  - fair practice to shorten the validation time of the transaction (to be seen later)
COMMON TRANSACTIONS

- the most common form of transaction: a simple payment from one address to another
- often includes some “change” returned to the original address.
- this type of transaction has one input and two outputs
AGGREGATING FUNDS

- A transaction aggregating several inputs into a single output
  - the equivalent of exchanging a pile of coins for a single larger note

- may be generated to clean up lots of smaller amounts that were received as change for payments (generated by wallet applications)

- merging funds belonging to the same user in the output of the transaction, but exploited also for joint payments (multisignature transactions)
transactions distributing one input to multiple outputs representing multiple recipients

used to distribute funds, for instance processing payroll payments to multiple employees
Asymmetric keys are used to verify transaction ownership.

- You create a private key $p$ and use it to generate a public key $P$

  ![Key generation diagram]

- Anything encrypted with $p$ can only be decrypted with $P$

  ![Encryption and decryption example 1]

- Anything encrypted with $P$ can only be decrypted with $p$

  ![Encryption and decryption example 2]
SIGNING TRANSACTIONS

A simplified scenario (with respect to real Bitcoin machinery)
- use your private key to encrypt a hash of the transaction

- Alice and Bob have their own private and public keys

Alice's private key
She keeps this secret

Alice's public key
Anyone can see this

Bob's private key
He keeps this secret

Bob's public key
Anyone can see this

Alice wants to authorize this transaction by signing it.
- this binds Alice's identity with the transaction record itself
SIGNING TRANSACTIONS

- Alice takes a hash of the transaction

\[
\text{SHA256(Pay 5 BT to } B) = 01495e7d19001c89fd3a47b6d1f0f6a7075cd1f89d91e8c1b7814d51fa9a544
\]

- Alice encrypts the hash using the private key

\[
61495e7d19001c89fd3a47b6d1f0f6a7075cd1f89d91e8c1b7014d51fa9a544 + a = Xlkjlkj4ghhj4#jue5iu 4ucmiou4 #jhlkRT34K4t4&%3iopt3#o%75%
\]

- Alice attaches the encrypted hash and her public key to the transaction and transmits it to the P2P network
Verifying the signature through Alice's public key it is possible to validate that only Alice could have created this blo

\[
\text{Pay 50 BTto B} \quad \rightarrow \quad \text{SHA256} \quad \rightarrow \quad \text{Decrypt} \quad \rightarrow \quad \text{Equal?}
\]

\[
\begin{align*}
\text{A:} & \quad \text{Xlkjlgkj4hghj4#}jwo5iu 4ucmiou4 \\
& \quad \#jhlkjRT34K4t4\&\#%opit3\#%75\
\text{B:} & \quad \text{61495e7d19001c89fd3a47bfd1f0f6a7} \\
& \quad \text{0755cd1f89d91e8c1b7014d51fa9a544}
\end{align*}
\]
Transactions Signatures

Transaction from A to B
- txn contents
- B's address
- sign
- A's signature
- A's public key
- verify
- sign
- A's private key

Transaction from B to C
- txn contents
- C's address
- sign
- B's signature
- B's public key
- verify
- sign
- B's private key

Transaction from C to D
- txn contents
- D's address
- sign
- C's signature
- C's public key
- verify
- sign
- C's private key
To make a payment a peer

- creates a correct transaction
- broadcast it to the peer's neighbors, which would broadcast it to theirs neighbors and so on
- after a while, the entire (reachable) network knows of the new transaction
The peers must agree on the order in which transactions happened:

- all must see the same order of the transactions, difficult because of network delays, no global time...

- local replica of the state may eventually diverge, but consistency is reestablished by distributed consensus
  - allows to keep a distributed, replicated consistent ledger including all the transactions (we will see in the next lesson)
THE P2P NETWORK

- a gossip-like broadcast of messages.

- bootstrap
  - new peers connect to the network using nodes
    - published by voluntary users or services,
    - using secure bootstrap nodes hard coded in the protocol.
THE P2P NETWORK

- a gossip-like broadcast of messages.
- overlay maintenance
  - each node sends the other peers, upon request, the list of its neighbors to allow new peers discovery
  - a protocol similar to Cyclon or TMAN

Sampled from local "addrMan" database

Propagate information about other peers

ADDR messages sent in response to GETADDR, also spontaneously
TRANSACTION BROADCAST

• The transactions are broadcasted on the network

• Each node may verify that the transaction is valid

• Validity check:
  • the previous output references by the transaction exist and they have not been spent
  • the sum of the input values is greater or equal to the sum of the outputs
  • the signatures for the transaction input are valid
    • each input is signed with the private key corresponding to the public key associated with the address it reference

• If the transaction is valid, it is broadcasted on the network
UNSPENT TRANSACTION OUTPUTS

- outputs of each transaction may be either in the spent or unspent state

- unspent output are those that are not input of any further transaction
  - an address balance is the sum of bitcoins in unspents outputs

- **Unspent transaction outputs (UTXO):** represents the shared space of the Bitcoin network

- effect of a transaction
  - move the input (output of a previous transaction) to the spent state
  - add the output of the new transaction to the UTXO.

- We can say that “**Bitcoins reside in the unspent outputs of the transactions**"
UNSPENT TRANSACTION OUTPUTS CACHE

• Bitcoin client maintains an **unspent transaction output cache**
  • a cache containing only unspent transactions
  • useful to check validity of new transactions

• Advantage of using the UTXO: it is much smaller than the whole transaction database (the block chain)

• UTXO can be kept in RAM, which speeds the validity check

• When checking the validity of a new transaction
  • look for its input in the UTXO
  • if all the inputs are found, the input correspond to previous outputs
  • otherwise, discard the transaction
UNSPENT TRANSACTION OUTPUT CACHE

Diagram showing the relationship between UTXO, blocks, and transactions. The diagram illustrates how transactions (Tx) are linked to output (TxOut) through blocks. Each block contains transactions that spend UTXOs, and the output of these transactions becomes another input (UTXO) in subsequent transactions.

Laura Ricci
Transactions, Addresses, Proof of Work

Dipartimento di Informatica
Università degli Studi di Pisa
TRANSACTIONS: SUMMARY

- Alice starts a transaction
  - constructs a transaction record signs with her secret key, and put in the transaction her public verification key

- transaction is broadcasted to all the nodes in the Bitcoin network for verification

- all the nodes receive information about all the transactions which occurred in the Bitcoin network

- transactions are incorporated, after their validation into a global ledger

- the global ledger contains all the transactions which have ever taken place in the Bitcoin system, starting from the genesis block
RECEIVING A TRANSACTION

Receive transaction $t$

for each input $(h, i)$ in $t$ do
  
  if output $(h, i)$ is not in local UTXO or signature invalid
  then
    Drop $t$ and stop
  end if

end for

if sum of values of inputs $<$ sum of values of new outputs then
  Drop $t$ and stop
end if

for each input $(h, i)$ in $t$ do
  Remove $(h, i)$ from local UTXO
end for

Append $t$ to local memory pool

Forward $t$ to neighbors in the Bitcoin network
RECEIVING A TRANSACTION

- all the Bitcoin nodes execute the previous algorithm when receiving a transaction

- the algorithm describes the local acceptance policy
  - the transaction which are locally accepted by executing this algorithm may not be globally accepted
  - the transaction considered unconfirmed and added to a pool, called the local memory pool
  - they are added to the Bitcoin blockchains when they are globally confirmed

- different local memory pool

- different Unspent transaction outputs in different nodes because of double spending
THE DOUBLE SPENDING PROBLEM

• Doublespend:
  • multiple transactions attempt to spend the same output.
  • only one transaction is valid since each output (amount) can be spent only once.

• Possible scenarios:
  • an output co-owned by multiple users
  • more frequently... a double spend attack: an attacker pretends to transfer an output to a victim, but it transfer the same output to her/himself
THE DOUBLE SPENDING PROBLEM

Consider a double spending attack

- Alice performs two transactions, \( T_1 \) and \( T_2 \)
  - \( T_1 \) sends 10 bitcoins to Bob
  - \( T_2 \) sends the same 10 bitcoins to herself.
- the two conflicting transactions are propagated on the network
- the validity of a transaction depends on the order in which they arrive
- consider the local acceptance algorithm:
  - each node accepts the first transaction it receives, then removes the corresponding input form the local cache, so that the second transaction is not accepted
  - The order in which the transactions are seen by different nodes may be not the same for all the nodes
    - the shared state diverges!
  - Eventual consistency: a mechanism is required to decide which of the two conflicting transactions is to be confirmed
THE MINING PROCESS

• new transactions are constantly flowing into the network from Bitcoin clients

• as these are received by the bitcoin peers, they get added to a temporary pool of unverified transactions maintained by each node

• some special peers, called miners,
  • take unverified transactions from this temporary pool and create a new transaction block
  • attempt to solve a very hard problem, the proof of work
  • if this succeeds, the transaction in the block are confirmed and the block is added to a blockchain and propagated on the network

• a transaction is not confirmed until it becomes part of the global distributed ledger, the blockchain.
to be seen in the next lessons:

- how the mining process guarantees to achieve distributed consensus?
- what is a proof of work?
- what is the structure of the blockchain?
- how the mining property guarantees the tamper-free property of the blockchain?
- what are the incentives for performing mining?
- how are bitcoin minted?
PROOF OF WORK

- A proof of work (PoW) is a mechanism that allows a party to prove to another party that a certain amount of computational resources has been utilized for a period of time.

- puzzles that can be solved, but require a considerable effort which cannot be short-circuited

- it must be possible to verify the effort made to solve a PoW in a easy way
  - verification requires less time with respect to the time needed to conduct the PoW

- many applications of the this technique:
  - deterring DOS attacks
  - deterring e-mail SPAM
  - Bitcoin mining
deterring of denial-of-service attacks:

- allow users access to a particular service only if they solve a very computationally expensive computational problem

- the computational effort required is effectively a way to throttle the requesters

- it must be easy for the service provider to check if the requester carried out the required work
  - only if that work was carried out the service is given.
PROOF OF WORK AND EMAIL SPAMS

- a proof of work protocol may be tied to an email message

- like affixing a post stamp to a message
  - rather than paying for that stamp using money, pay for the stamp through CPU cycles

- if someone is sending out a small number of messages
  - the proof of work will not amount to a huge amount of work, because it is only executed a very small number of times

- for a spammer, who might be sending hundreds of thousands or millions of messages
  - it might be prohibitively expensive to use so many CPU cycles for each message it sends
PROOF OF WORK: HOW DOES IT WORK?

- given challenge string, $c$, define a proof which is tied to this string

- the challenge string may be:
  - for spams, the whole message.
  - for Bitcoin: the header of a block (set of transactions) of the blockchain

- the prover must find and return a response string
  - the prover must give a response that has a specific mathematical property in relation to the challenge

- for instance: given the challenge $c$ find a response $p$ such that
  - the output of a cryptographic hash (HASH-256) applied to the concatenation of $c$ and $p$ satisfies some property
PROOF OF WORK: HOW DOES IT WORK?

- example of property: the output must have a prefix of 0 of predefined length
PoW: A FORMAL DEFINITION

- Consider following notation:
  - \(d\) (difficulty), a positive number which is used to adjust the time to execute the proof
  - \(c\) (challenge) is a bitstring
  - \(x\) (nonce) is a bitstring

- A Proof-of-Work is a function:

\[
F_d(c, x) \rightarrow \{\text{true}, \text{false}\}
\]

which satisfies the following properties:

- fixed \(d\) and \(c\), finding \(x\) such that \(F_d(c, x) = \text{true}\) is computationally difficult, but feasible.
- \(F_d(c, x)\) is fast to compute, if \(d\), \(c\), and \(x\) are fixed
THE BITCOIN PROOF OF WORK

\[
F_d(c, x) \rightarrow SHA256(SHA256(c|x)) \leq \frac{2^{224}}{d}.
\]

- double hash of the concatenation of the challenge \(c\) and the nonce \(x\), using SHA256

- The output of SHA256 is a cryptographic hash with a numeric value in \(\{0\ldots 2^{266-1}\}\) which is compared to a target value \(2^{224}/d\), which gets smaller when increasing the difficulty.
A SIMPLIFIED PROOF OF WORK

• $c = \text{"Hello, world!"}$ and nonce $x = 0$ then (output is in hexadecimal)
$h(\text{"Hello, world! 0"}) = \text{1312Af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64}$

• first with $x = 0$ is a failure, since the output does not begin with any zeroes at all, try nonce $x = 1$
$h(\text{"Hello, world! 1"}) = \text{e9af424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7d8}$

• keep trying different values for the nonce, $x = 2, 3, \ldots$ finally, at $x = 4250$ obtain:
$h(\text{"Hello, world! 4250"}) = \text{0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e9}$

success if the proof of work requires 4 leading zeros!
if you exploit a good cryptographic hash function then the only known way to find the response string is to try a huge number of different possibilities.

- **brute force**: trying a lot different proposed strings until you find one that works

- if it is required that the output have a prefix of 40 consecutive zeros this would require, on the average, to try $2^{40}$ possibilities
  - but if you are lucky, you will resolve it faster

- a lot of computational power!
brute force is the only possible approach?

- what makes proof of work hard to solve?
  - the output from a cryptographic hash function behaves like a random number
  - each output bit looks like coin flips (0/1)
  - change the input a bit and the output from the hash function changes completely

- so no better way of finding the correct output than trying by brute force
  - the probability to find out the right output is very small!
PROOF OF WORK: PROPERTIES

- **it can be tuned to obtain the desired computational difficulty**
  - if you require even more computational effort, increase or decrease the number of leading zeros in the prefix.
  - adding an zero, double the computational effort, on the average
  - remove a zero, reduce the computational effort of one half

- **ease to be verified:** if someone proposes a nounce, it is very easy to validate that it is a correct proof of work
  - take the challenge and the nounce and hash them together
  - if the output produced by the PoW has the required number of leading zeros, the PoW is valid