P2P Systems and Blockchains
Spring 2021,
instructor: Laura Ricci
laura.ricci@unipi.it

Lesson 13:
BITCOIN TRANSACTIONS AND SCRIPTS

01/04/2021
Cryptocurrencies = “virtual” currencies that can be used for digital payments
FROM CENTRALIZED TO DECENTRALIZED PAYMENT SYSTEMS

Balance sheet

Mesopotamia

2040 BC

Centralized Banking

HSBC

1848

Bank of America

1936

Turing

Privacy-Preserving Banking

VISA

1958

mastercard

Decentralized Banking

ECash

1983

Chaum

Bitcoin

Ethereum

2008

CASH

2015

1998

PayPal
DIGITAL VERSUS PAPER CURRENCIES

Paper:

Digital:

Very useful if is also digital.
characteristics:

- a trusted server (a financial institution), is needed for each transaction
- “digital money” doesn’t “circulate”
- high transaction fees,
- no anonymity
PAYMENT TRANSACTION WITH INTERMEDIARIES

Checks account balance, updates account ("debit")

Giro network transfer, RT1/EBA Clearing

Buyer's bank

Seller's bank

Updates account ("credit")

Triggers SEPA transfer of the invoice amount

Intermediary confirms receipt of payment

Buyer

Purchase agreement

Delivery promise

Delivery

Seller

IBAN, Invoice amount

Dipartimento di Informatica
Università degli Studi di Pisa
Laura Ricci
Bitcoin Transactions and Scripts
WHO TRUST WHOM?

• pre-payment:
  • the buyer trusts the seller: does he/she keep his/her delivery promise?

• both parties trust their banks

• banks trust among themselves

• a central ledger managed by the buyer's bank prevents double-spending
  • buyer can only spend money once!
• born in December 1998.

• In December 2000 fusion with Elon Musk

• An intermediary service
  • A company sits between you and the seller
  • Give your credit card to the intermediary, instead of the seller
  • Needed at the end of the nineties
    • Standard for protocol-level encryption protocols were just emerging

• Basically centralized:
  • Requires an intermediary to enroll
  • Link the service to your bank account or a credit/debit card
BACKGROUND: E-CASH

• the main problem is double spending
• Alice is sending a coin to Bob
• a coin is just a bitstring
• reach two contrasting goals:
  • how to prevent double-spend?
  • how to make the money transfer untrackable
BACKGROUND: E-CASH

- David Chaum, mid-1980s
  - first ideas of applying cryptography to cash
  - “Blind Signatures for Untraceable Payments”
- E-cash
  - Anonymous
  - Central Bank required
  - Digicash: mismanagement lead to bankruptcy in 1998
- basic idea:
  - RSA/Public Key cryptography
  - Elliptic Curve Cryptography
  - Hash Functions
  - blind signatures to guarantee anonymity
THIS WAS THE SITUATION UNTIL 2008.......
THE INFANCY OF BITCOIN

- **January 2009:**
  - the “Genesis Block” is launched starting the initial “mining” of Bitcoins
  - later that month, first transaction between Satoshi and Hal Finney, a developer and cryptographic activist.

- **May 2010** The first real “pizza transaction” from the bitcointalk forum

---

**Pizza for bitcoins?**

I'll pay 10,000 bitcoins for a couple of pizzas. I have some left over for the next day. I like having left over pizza to nibble on later. You can make the pizza yourself and bring it to my house or order it for me from a delivery place, but what I'm aiming for is getting food delivered in exchange for bitcoins where I don't have to order or prepare it myself, kind of like ordering a 'breakfast platter' at a hotel or something, they just bring you something to eat and you're happy!

I like things like onions, peppers, sausage, mushrooms, tomatoes, pepperoni, etc., just standard stuff no weird fish topping or anything like that. I also like regular cheese pizzas which may be cheaper to prepare or otherwise acquire.

If you're interested please let me know and we can work out a deal.

Thanks,
Laszlo
May 2010: first known Bitcoin purchase for real goods:

- Laszlo Hanyecz, from Florida, offered, on the Bitcointalk forum, 10,000 BTC (he had received as a mining reward) 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 (80 M$, today) in exchange for $25 worth of pizza.
**Transazione** Ottieni informazioni su una transazione bitcoin

<table>
<thead>
<tr>
<th>Sommario</th>
<th>Input e Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensione</td>
<td>Totale Input: 10,000.99 BTC</td>
</tr>
<tr>
<td>Ora di Ricezione</td>
<td>Totale Output: 10,000 BTC</td>
</tr>
<tr>
<td>Incluso nei Blocchi</td>
<td>Tasse: 0.99 BTC</td>
</tr>
<tr>
<td>conferme</td>
<td>Costo per byte: 4,191.363 sat/B</td>
</tr>
<tr>
<td>Inoltrato dall'IP</td>
<td>Stima dei BTC scambiati: 10,000 BTC</td>
</tr>
<tr>
<td>Visualizza</td>
<td>Script: Mostra gli script e la coinbase</td>
</tr>
</tbody>
</table>

`a1075db55d416d3ca199f55b6084e2115b9345e16c5cf302fc80e9d6f56d48d`

1XPTgDRhN8RFnzniWCddobD9iKZatrvtH4 → 17SkEw2md5avVNyYgj6RiXuQKNwkXaxFyQ

10,000 BTC
BITCOIN IN THE LAST TEN YEARS

- **2011-2013**: Bitcoin popular for buying illegal goods
  - **SilkRoad** the most sophisticated criminal online marketplace
  - **Mt. Gox** a famous exchanger, got hacked for the first time: 2000 BTC stolen

- **2013**
  - Market price skyrockets and the world notices the new cryptocurrency
  - Mt. Gox hacked for the second time

- **2017**
  - Bitcoin enters a new bull market
  - At the end of the year, Bitcoin highest price around $20000

- **2018**
  - Bitcoin enters bear market
  - Bitcoin falls down at $3250
BITCOIN: PRICE FLUCTUATION

Irrational exuberance
BIPTE: PAYMENT WITHOUT INTERMEDIARY

- **Create Transaction using the wallet**
- **Processing of transaction through P2P network**
- **Buyer**
- **Seller**
- **Purchase agreement**
- **Instead of IBAN: Bitcoin address Invoice amount**
- **Seller finds transaction in the blockchain**
- **Delivery**
- **Delivery promise**
BITCOIN: MOTIVATIONS

- a payment transaction system without intermediaries
- a purely peer-to-peer version of “digital cash”
  - no controlling authority, no server
- default setting
  - suspicion in a completely untrusted environment
- challenge: identify evil intentions, find consensus
- he/she (Nakamoto) manages to solve the problem of money being copied
  - a solution for double spending
  - solves a foundation problem for Bitcoin to grow legitimately
BITCOIN: MOTIVATIONS

Problems of previous approaches

- a trusted server is needed, a financial institution
- no anonymity
- high transaction fees,

Bitcoin

- openness:
  - all you need is a Bitcoin client
  - no banking account, no credit card
    - transactions do not go to a third party
    - no centralized entity controls the money supply
- pseudo anonymity
  - transactions without identity disclosures
    - like “paying cash”,
  - addresses as pseudonyms
- small fees
WHAT IS BITCOIN?

- Economics
- Cryptography
- Distributed systems
WHY BITCOIN BECAME SO POPULAR?

• ideological reasons
  • cryptoanarchy (nobody controls the money)
  • cyberpunk movement

• good timing due to financial crisis in 2008
  • no money printing in Bitcoin
  • a “child of the crisis”?

• trading of illegal goods due to seeming anonymity (pseudonymity)

• payments can be cheap
  • almost no fees for long time (PayPal 2-10%)

• novel technology for distributed systems
WHY BITCOIN BECAME SO POPULAR?

- more and more people accept transactions payed in Bitcoin over the last years
- existence of a real Bitcoin economy
- presence of exchangers: places where bitcoin are exchanged for mainstream fiat currencies
  - MT.Gox: closed in February 2014
  - others: CoinDesk, BPI, Bitstamp, Bitfinex, Coinbase, itBit, OKCoin
- a big fluctuation in exchange value
  - today price: 5.758,39 euro: a big slow down due to COVID
- Note that
  - Bitcoin: is the system
  - bitcoin: is the currency
BITCOIN: PROBLEMS

- no central authority defending against the classical security threats:
  - like cash: exploit the cryptocurrency for illicit purposes

While these types of innovations may pose risks related to law enforcement and supervisory matters, there are also areas in which they may hold long-term promise, particularly if the innovations promote a faster, more secure and more efficient payment system.
BITCOIN IN CONTEXT

**Bitcoin:**
- Protocol
- Client software
- Data (blockchain)

**Bitcoin Ecosystem**

- Exchanges
- Mining pools
- Remote wallets

**Financial Sector**
- Agents
- Goods
- Markets (legal/illegal)
- Externalities

**Real Economy**

**Centralized**
- Banks
- Fonds
- Regulators
- Treasury
WHAT WE WILL SEE

- Bitcoin identity management
- transactions
- scripts
- blockchain
- consensus
  - miners
  - mining pools
- attacks
  - double spending,....
DECENTRALIZED IDENTITY MANAGEMENT

- if I want to send bitcoin to someone, the first point is
  - how to represent identities?
- Bitcoin uses a decentralized identity system
- an easy way to generate new identities in a cryptographic system
  - create a new, random key-pair
    - \textit{sk} private (secret) key
    - \textit{pk} public key
  - if \textit{pk} “looks random”, nobody needs to know who you are
  - \textit{pk} is the public “name” of a user, “speak for” the “user's identity”
    - more often is used Hash(Pk)
- only the owner of \textit{sk} can control the identity
- if you see a signature \textit{sig}, such that \textbf{verify}(pk, data, sig)== true, think that \textit{pk} has generated the transaction
the private key (k) is a number, usually picked at random.

- ownership control over the private key is fundamental to control all funds associated with the corresponding bitcoin address

- from the private key k, generate a public key K
  - through elliptic curve multiplication, a one-way cryptographic function

- from K generate a bitcoin address (A) through a one-way cryptographic hash
Base58 encoding

- uses the upper- and lowercase
- set of lower and capital letters and numbers without (0, O, I, l)
- omits some characters that can appear identical when displayed in certain fonts
BITCOIN ADDRESSES: RECAP

- identities called **addresses** which, in the majority of cases,
  - represent the owner of a private/public key pair
  - are generated from the public key and corresponds to it
    - but not in all cases: the **address may also represent a script**
  - anybody can make a new identity at any time
    - make as many identities as you want!
    - used like the beneficiary name on a check: “pay to the order of xxx”
  - as far as concerns privacy...
    - addresses (public keys) not directly connected to real world identities
    - but an observer can link together an address's activity over time, and make inferences...
BITCOIN PAYMENT WORKFLOW

- merchant Bob shares its address A out of band (not using Bitcoin P2P)
- customer Alice broadcasts on the P2P networks transaction t which pays A
- miners collect broadcasted transactions into a candidate block
  - one of the candidate blocks containing t is mined
- merchant waits for confirmations on t before providing goods
- all this to be described in more details in the next lesson
A TRANSACTION: ONE INPUT, TWO OUTPUTS

- simplify: one address for each wallet
- a simple payment
- the most typical transaction
- one of the output is the “change”
- returned to the same address or to another address of the issuer
A BITCOIN TRANSACTION

Transaction Fees
\[ \sum \text{inputs} \geq \sum \text{outputs} \]
Difference are fees

Inputs
- 1 BTC @Bob
- 0.75 BTC @Alice

Outputs
- 0.25 BTC @Bob
- Change address @Alice

Sender address
Recipient address
Change address

Alice
Bob
Wallet
Wallet
BITCOIN TRANSACTIONS

- each transaction completely uses the input funds
  - no change is left in the input addresses

- basic condition for transaction validity

\[ \Sigma(\text{input funds}) \geq \Sigma(\text{output funds}) \]

- transaction fee
  - \[ \Sigma(\text{input funds}) - \Sigma(\text{output funds}) \]
  - an optional amount
  - collected by the miners as a reward to include the transaction in a block
  - fair practice to shorten the validation time of the transaction
• a chain of ownership is registered on the blockchain

• value is moved from address to address
• a **chain of ownership** is registered on the blockchain

• value is moved from address to address
MERGING FUNDS: MORE INPUT, ONE OUTPUT

• the equivalent of exchanging a pile of coins for a single larger note
  • aggregating several inputs into a single output
• may be generated to clean up lots of smaller amounts that were received as change for payments
  • generated by wallet applications
• exploited also for joint payments
  • multisignature transactions
DISTRIBUTING FUNDS: ONE INPUT, MORE OUTPUTS

- Transactions distributing the value in input to multiple recipients
- Used to distribute funds
  - Example: processing payroll payments to multiple employees
TRANSACTIONS: RECAP

- map input addresses to output addresses
- each input spends a previous output
- usual transaction: one input, two outputs
- only unspent output are significative: maintained in the UTXO
BITCOIN TRANSACTIONS

- centralized currencies are account-based and transfer is between accounts
  - Alice account number is 43569 and the current balance is 300 EUROs
- instead, Bitcoin does not exploit accounts, records only transactions
  - transactions inputs and outputs are not accounts
  - transaction output serves as an input source of bitcoins for a later transaction
- bitcoin belonging to a user are spread along the blockchain
  - unspent output addresses: bitcoin sent to an address and not spent in any transaction
  - to find the total amount owned by a address: search the blockchain for its unspent output addresses
- but transactions are more complex: they do not contain only value but also code, as we will see in the next slides!
BITCOIN TRANSACTIONS

• there is more than values in a transactions

• each transaction contains some scripts

• what is a script?
  • a simple program written in a simple programming language
  • language is not Turing complete
  • limitation is intentional to prevent infinite or endless looping and error execution.
  • used mainly to verify the ownership of the transferred coins
 TRANSACTION METADATA JSON

```json
{
  "hash": "5a42590...b8b6b",
  "ver": 1,
  "vin_sz": 2,
  "vout_sz": 1,
  "lock_time": 0,
  "size": 404,
...
}
```

- hash of the entire transaction, an unique identifier
- version: allow different interpretation of some fields
- locktime defines the earliest time that a transaction can be added to the blockchain
  - set to zero in most transactions to indicate immediate execution
  - used in escrow and for the lightning network
TRANSACTION INPUT JSON

```
"in": [
  {
    "prev_out": {
      "hash": "3be4...80260",
      "n": 0
    },
    "scriptSig": "30440...3f3a4ce81"
  },
  ...
],
```

- a JSON array: each element contains a key value pair
  - hash pointer to a previous transaction and index of the previous transaction's output
  - an unlocking script
**TRANSACTION OUTPUT JSON**

```
"out": [  
  
  {  
    "value": "10.12287097",
    "scriptPubKey": "OP_DUP OP_HASH160 69e...3d42e
                     OP_EQUALVERIFY OP_CHECKSIG"
  },

  ...
  
  (more outputs)
  
  ]

  }
```

- a JSON array where each element contains a pair
  - value to be transferred in that output
  - a **locking script** containing the address where that value has to be transferred
UNLOCKING AND LOCKING BITCOINS

- an address $A_1$ sends some bitcoin to some other address $A_2$
  - $A_1$ unlocks some of its bitcoin
  - $A_1$ puts a small piece of code (script) on top of the sent bitcoin to lock the transferred bitcoin
  - when $A_2$ wants to spend its bitcoin it has to unlock them
    - provide another small piece of code
    - run the locking code together with the unlocking code
    - allow $A_2$ to spend the received bitcoin

![Diagram showing unlocking and locking of bitcoins]

every bunch of bitcoin on the blockchain is locked
• an amount of bitcoin is generally sent to a bitcoin address, which is the hash of the public key

• more simple scenario

  • send bitcoins to a public key

  • we will use this case in the following slides
UNLOCKING AND LOCKING BITCOINS

- B notifies its public key to A
- A sends its bitcoin to the public key of B
  - A unlocks some of its bitcoin, that were locked by a previous transaction
  - creates a lock on the bitcoin sent to B (to the public key of B)
  - only B it will be able to unlock the bunch of received bitcoins, with its private key
LOCKING AND UNLOCKING BITCOINS

- Green public key (lock)
- Red: is signature (unlock)
- locking refers to green, unlocking to red
THE STRUCTURE OF BITCOIN TRANSACTIONS

**Output Format**

- Previous Regular Tx with Tx ID = \( l_1 \)
  - One or more inputs
  - Amount \( x_1 \) Locking Script \( C_1 \)

- Previous Coinbase Tx with Tx ID = \( l_2 \)
  - Amount \( x_3 \) Locking Script \( C_3 \)

**Regular Transaction**

- Tx ID = \( l_1 \)
  - Output Index = 0
  - Unlocking Script \( R_1 \)

- Tx ID = \( l_1 \)
  - Output Index = 1
  - Unlocking Script \( R_2 \)

- Tx ID = \( l_2 \)
  - Output Index = 0
  - Unlocking Script \( R_3 \)

- Amount \( y_1 \)
  - Locking Script \( C_4 \)

- Amount \( y_2 \)
  - Locking Script \( C_5 \)

**Input Format**

- Input 0
  - hash
  - n
  - scriptSigLen
  - scriptSig
  - nSequence

- Input 1
  - Output 0
  - Output 1
COINBASE TRANSACTIONS

0 inputs, only outputs

• fresh bitcoin generated by the system
• generated for rewarding miners for having solved the proof of work
• transfer the reward to one of the addresses of the miners

Coinbase Transaction

\[
\begin{array}{c}
\text{Amount } x_1 \\
\text{locking Script } C_1
\end{array} \quad \left\{ \begin{array}{c}
\text{Output 0} \\
\text{Output 1}
\end{array} \right.
\begin{array}{c}
\text{Amount } x_2 \\
\text{locking Script } C_2
\end{array}
\]
A “REAL” BITCOIN TRANSACTION

```
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nVersion</td>
<td>Number of Inputs N</td>
</tr>
<tr>
<td>hash</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
</tr>
<tr>
<td>scriptSigLen</td>
<td></td>
</tr>
<tr>
<td>scriptSig</td>
<td></td>
</tr>
<tr>
<td>nSequence</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>hash</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
</tr>
<tr>
<td>scriptSigLen</td>
<td></td>
</tr>
<tr>
<td>scriptSig</td>
<td></td>
</tr>
<tr>
<td>nSequence</td>
<td></td>
</tr>
<tr>
<td>Number of Outputs M</td>
<td></td>
</tr>
<tr>
<td>nValue</td>
<td></td>
</tr>
<tr>
<td>scriptPubkeyLen</td>
<td></td>
</tr>
<tr>
<td>scriptPubkey</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>nValue</td>
<td></td>
</tr>
<tr>
<td>scriptPubkeyLen</td>
<td></td>
</tr>
<tr>
<td>scriptPubkey</td>
<td></td>
</tr>
<tr>
<td>nLockTime</td>
<td></td>
</tr>
</tbody>
</table>
```

- encoded in JSON
- further housekeeping not reported
BITCOIN SCRIPTING LANGUAGE: DESIGN GOALS

- Bitcoin has its own scripting language for transactions
  - a stack based, non-Turing complete language
  - built for Bitcoin, FORTH-LIKE
- not Turing complete:
  - no possibility to create “infinite loops”
  - all Bitcoin “full nodes” (miners and full nodes, not mobile) have to validate these scripts
    - they do not have to fall in a loop
  - preventing the transaction validation mechanism from being used as a vulnerability.
- executable on a wide range of hardware
BITCOIN SCRIPTING LANGUAGE: DESIGN GOALS

- **stateless**: there is no state prior to execution of the script, or state saved after execution of the script.
  
  - all the information needed to execute a script is contained within the script
  
  - different from Ethereum

- **deterministic**: a script will predictably execute the same way on any system.

- **simple, compact**:
  
  - one byte OPCODE, 256 instructions
  
  - basic arithmetic, basic logic (IF...THEN...ELSE), special purpose instructions to support cryptography
    
    - hashes
    
    - signature verification
    
    - multisignature verification
1 byte opcodes in total: 15 disabled, 75 working

- **arithmetic**: `OP_ABS`, `OP_ADD`, ...
- **stack**: `OP_DROP`, `OP_SWAP`, ...
- **flow control**: `OP_IF`, `OP_ELSE` (not while!)
- **bitwise logic**: `OP_EQUAL`, `OP_EQUALVERIFY`, ...
- **crypto instructions** for:
  - hashing, e.g. `OP_SHA1`, `OP_SHA256`
  - (Multiple) Signature Verification, `OP_CHECKMULTISIG`, ...
  - Locktime: `OP_CHECKLOCKTIMEVERIFY`, `OP_CHECKSEQUENCEVERIFY`
A STACK BASED EXECUTION

<table>
<thead>
<tr>
<th>Remaining Script</th>
<th>Stack State</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP_2  OP_3  OP_ADD</td>
<td></td>
</tr>
<tr>
<td>OP_3  OP_ADD</td>
<td>2</td>
</tr>
<tr>
<td>OP_ADD</td>
<td>3 2</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>
INSIDE BITCOIN SCRIPTS

- a piece of code that verifies a set of arbitrary conditions that must be met in order to spend coins

- script types
  - most are simple signatures checks
    - redeem a previous transaction by signing it with the correct key
  - MultiSig
  - Pay-to-Script-Hash
  - proof-of-burn

- more complex scripts encode more complex spending conditions
  - escrow transactions
  - green addresses
  - micro payments
LOCKING AND UNLOCKING SCRIPTS

- Pay to PubKey is the simpler script: the one in the next slides
P2PK (PAY-TO-PUBKEY)

unlocking script

locking script
SCRIPT EXECUTION

Locking script : <Public Key> CHECKSIG
Unlocking script : <Signature>
LOCKING SCRIPT

Unlocking script: <Signature>

Unlocking script: <Signature>
SCRIPT EXECUTION

Locking script : <Public Key> CHECKSIG

Unlocking script : <Signature>
PAY-TO-PUBLIC-KEY-HASH (P2PKH)

- the most popular script
  - more operators are needed
  - locking script contain hash of the key, instead of the public key
PAY-TO-PUBLIC-KEY-HASH (P2PKH)

- the unlocking script must provide
  - a public key which returns that hash of the address contained in the locking script
  - a signature corresponding to the public key

- script execution
  - hash the public key and compare to the hash in the locking script
  - check if the signature is valid (as in Pay-to-Key script)
P2PKH SCRIPT IN PRACTICE

- *Subway* is accepting payment in bitcoin

- Bob buys a sandwich: how to pay?
  - the payment has to be locked by a P2PKH challenge script.

- *Subway* provides the corresponding P2PKH address to Bob
  - the address can be encoded in a QR code
  - can be scanned by Bob using a mobile phone camera

- other scenarios:
  - the address can be sent by e-mail
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
TRANSACTION LIFE CYCLE

- each node verifies the transaction validity:
  - the previous outputs referenced by the transaction exist and 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 referenced in output script
  - only if the transaction is valid, it is passed to the neighbours
TRANSACTIONS LIFECYCLE

• starts with the transaction’s creation
• then the transaction is signed with one or more signatures indicating the authorization to spend the funds referenced by the transaction.
• it is broadcasted on the bitcoin P2P network
• each network node (participant) validates and propagates the transaction until it reaches (almost) every node in the network.
• the transaction is verified by a mining node and included in a block of transactions recorded on the blockchain.
• once recorded on the blockchain and confirmed by sufficient subsequent blocks (confirmations), the transaction is a permanent part of the blockchain
• the funds allocated to a new owner by the transaction can then be spent in a new transaction, extending the chain of ownership
UNSPENT TRANSACTION OUTPUTS

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

- unspent output (UTXO) are those that are not input of any further transaction
  - the bitcoin belonging to a user might be scattered as UTXO amongst hundreds of transactions and hundreds of blocks in the blockchain
  - there is no such thing as a stored balance of a bitcoin address or account
  - only scattered UTXO

- the concept of a user’s bitcoin balance is a derived construct created by the wallet application.
  - the wallet calculates the user’s balance by scanning the blockchain and aggregating all UTXO belonging to that user.
  - an address balance is the sum of bitcoins in unspents outputs
UTXO CACHE

• Unspent transaction outputs (UTXO): represents the shared space of the Bitcoin network
  “the state of Bitcoin reside in the unspent outputs of the transactions”
• more complex representation needed for Ethereum state
• Bitcoin UTXO is a cache containing unspent output
  • useful to check validity of new transactions
• advantage of using the UTXO cache
  • much smaller than the whole transactions database (the block chain)
  • 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
RECEIVING THE 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 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 (waiting for confirmation)

Forward $t$ to neighbors in the Bitcoin network
MANAGING 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 are added to a pool, called the local memory pool
  - they are added to the Bitcoin blockchains when they are globally confirmed
- different local memory pools
- different unspent transaction outputs in different nodes because of double spending
- eventual consistency