P2P Systems and Blockchains

Spring 2018,
instructor: Laura Ricci
laura.ricci@unipi.it

Lesson 13: BITCOIN: ADDRESSES, TRANSACTIONS

18/4/2018

OUTLINE

- What we will see in the next lessons
  - Bitcoin Addresses
  - Transactions
  - Bitcoin P2P Network
  - Miners
  - Blockchain

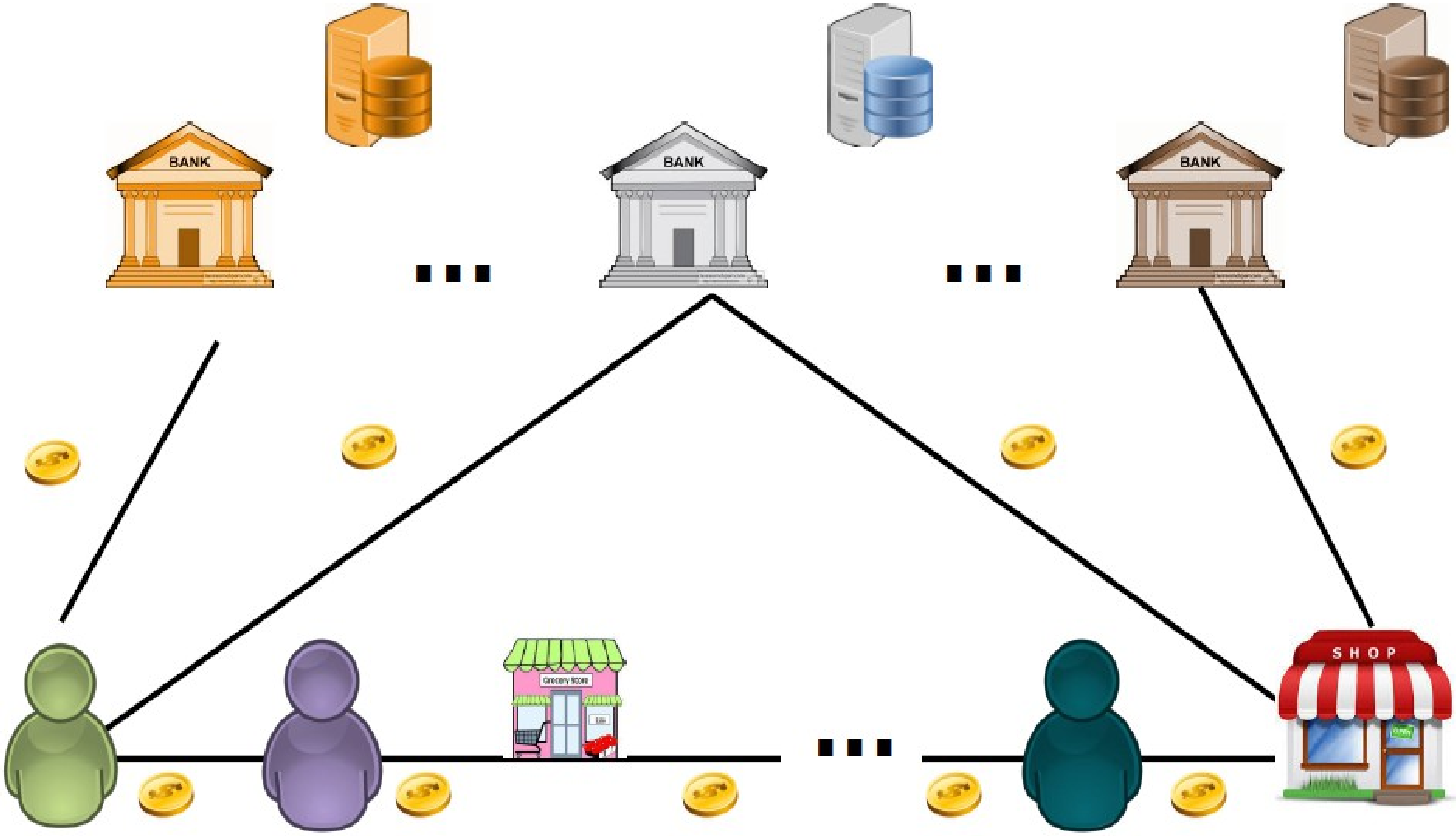
COMMON TYPES OF PAYMENTS

PayPal

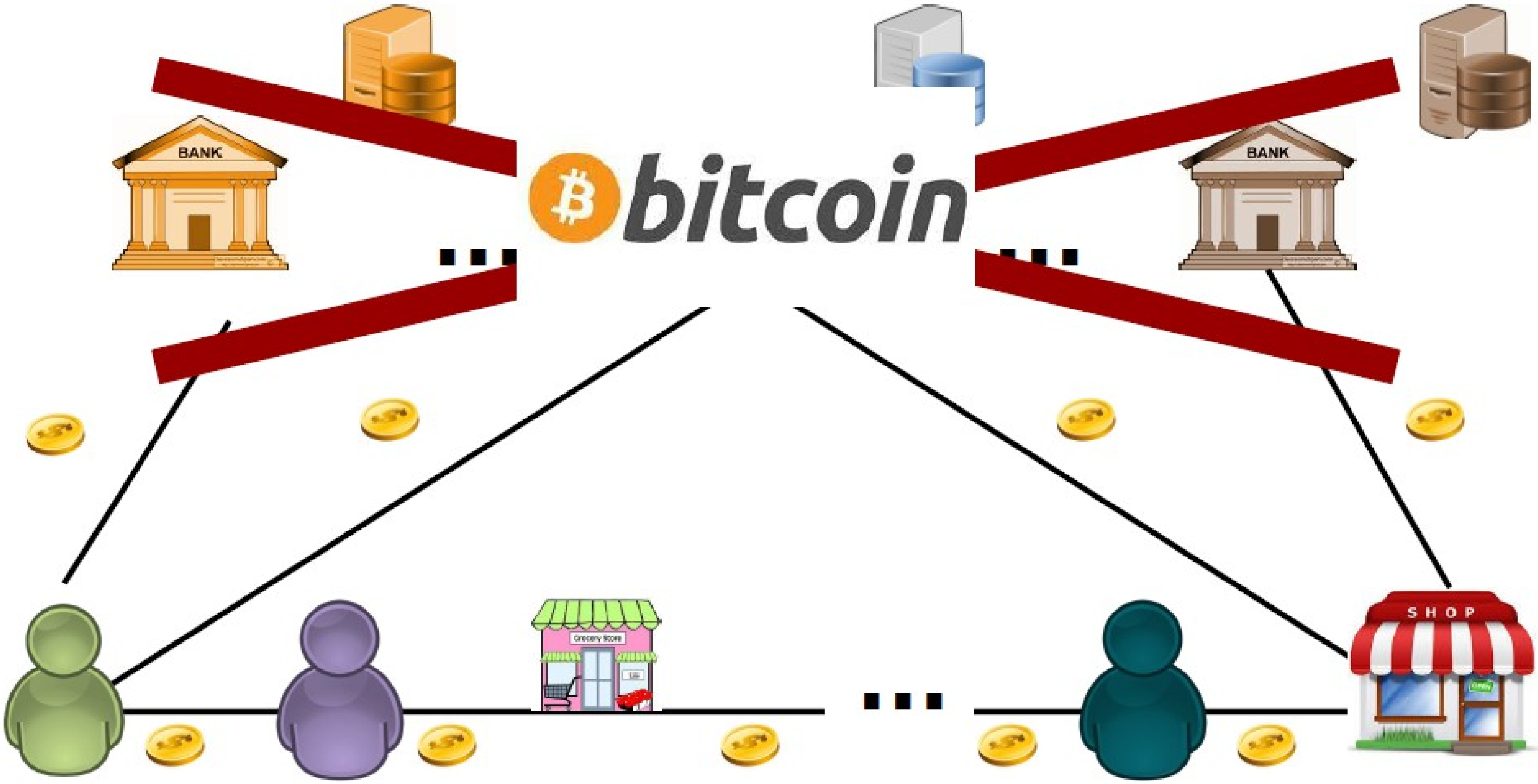
Common characteristic?

Trust to a financial institution

TRUSTING A CENTRAL AUTHORITY



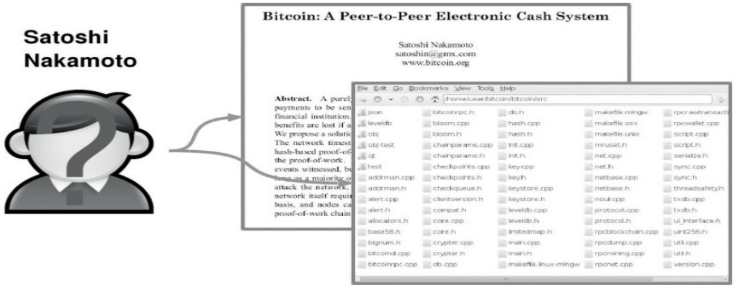
THE BITCOIN (AND BLOCKCHAIN) REVOLUTION



BITCOIN: 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

a1075db55d416d3ca199f55b6084e2115b9345e16c5cf302fc80e9d5fbf5d48d

1XPTgDRhN8RFnzniWCddobD9iKZatrvH4 → 17SkEw2md5avVNyYgj6RiXuQKNwkXaxFyQ 10,000 BTC

10,000 BTC

Sommario	
Dimensione	23620 (byte)
Ora di Ricezione	2010-05-22 18:16:31
Incluso nei Blocchi	57043 (2010-05-22 18:16:31 + 0 minuti)
conferme	405698 conferme
Inoltrato dall'IP	0.0.0.0 (whois)
Visualizza	Osserva il Grafico ad Albero

Input e Output	
Totale Input	10,000.99 BTC
Totale Output	10,000 BTC
Tasse	0.99 BTC
Costo per byte	4,191.363 sat/B
Stima dei BTC scambiati	10,000 BTC
Script	Mostra gli script e la coinbase

BITCOIN: MOTIVATION

- why it is worth using Bitcoin? why don't use PayPal or electronic credit cards?
- some peculiar characteristics of Bitcoin:
  - anonimity & privacy
    - possibility of transacting without identity disclosures
    - addresses computed from public keys as pseudonyms
    - like "paying cash",
    - risk: exploit the cryptocurreency for illicit purposes
  - openness: everyone having a connection to te 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 autority 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
  - vadidation performed by

BITCOIN: MOTIVATION

- Least, but noy 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: $8,127.76
    - a big fluctuation in exchange value

- Note: Bitcoin, the system, bitcoin, the currency

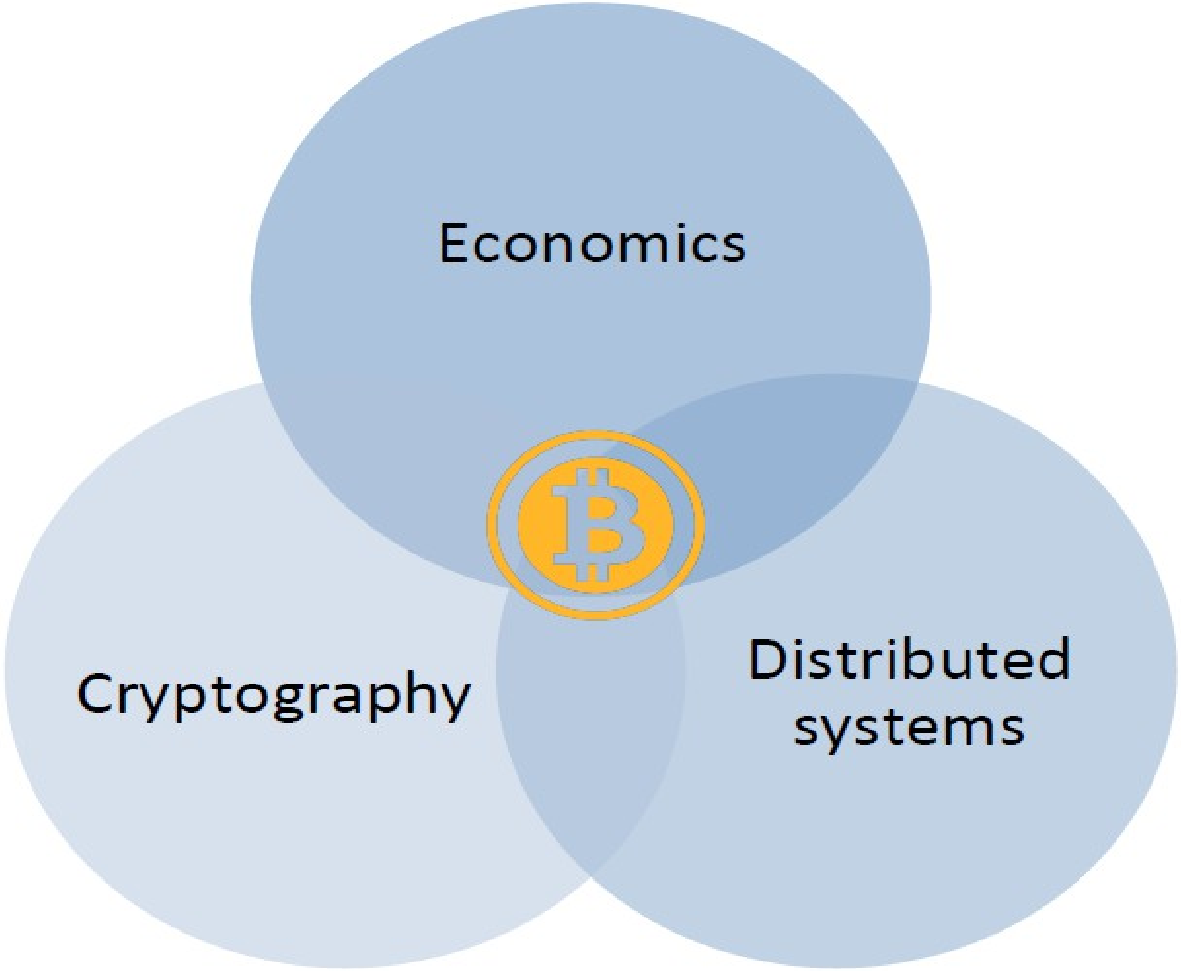
BITCOIN: PRICE FLUCTUATION

- https://blockchain.info/it/charts/

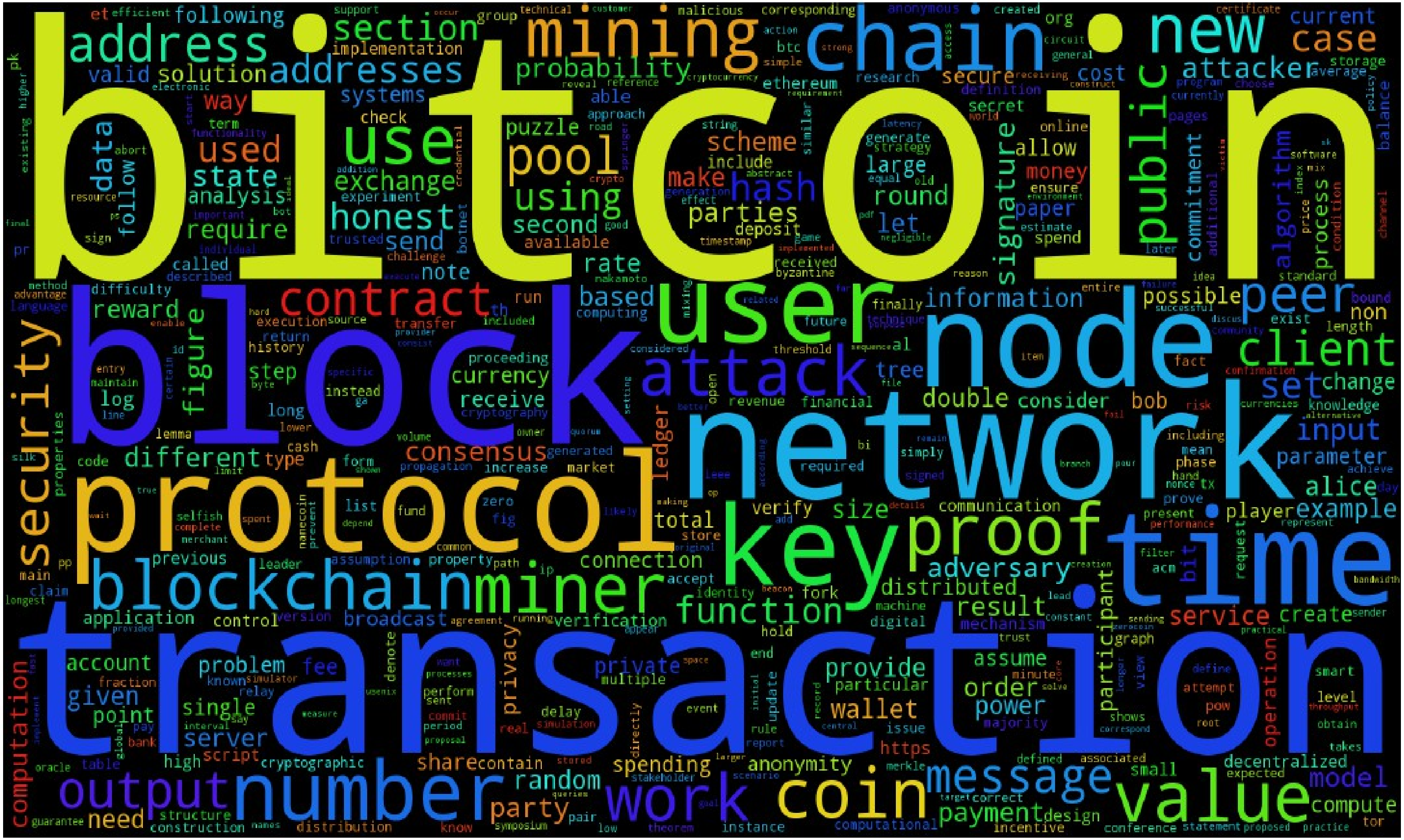
BITCOIN: TOTAL VALUE OF BITCOIN

- Total value: = 134.093.392.445 (18 Apr. 2018)
- https://blockchain.info/it/charts/

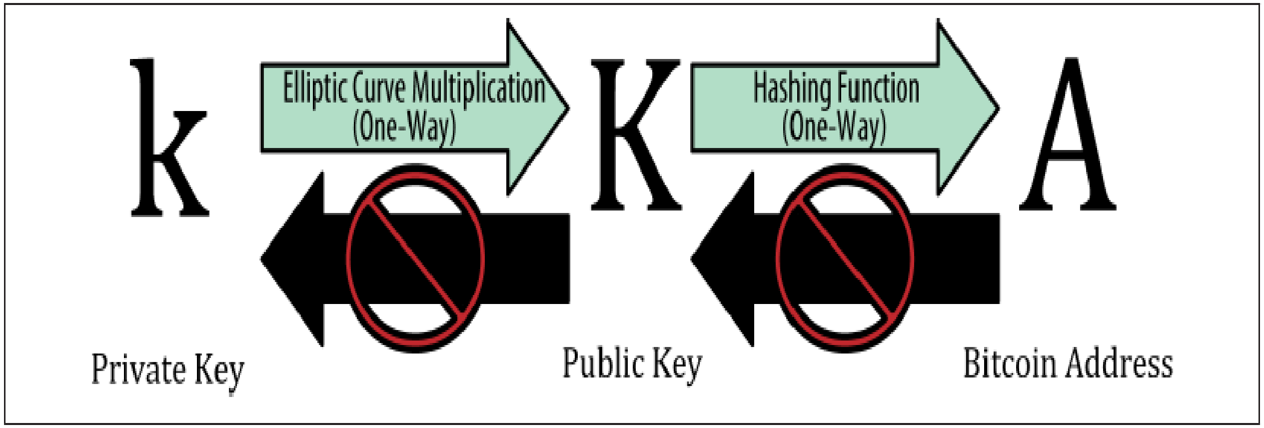
BITCOIN: INTRODUCTION



WHAT ARE THE NEXT TOPICS?



PUBLIC/PRIVATE KEYS AND ADDRESSES



- the private key (k) is a number, usually picked at random.
  - ownership and control over the private key is fundamental to control all funds associated with the corresponding bitcoin address.
- from the private key k, elliptic curve multiplication, a one-way cryptographic function, are exploited to generate a public key (K).
  - digital key are stored in a wallet
- from the public key (K), a one-way cryptographic hash function is used to generate a bitcoin address (A).

DECENTRALIZED IDENTITY MANAGEMENT

- how to make a new identity in a cryptographic system:
  - useful trick Public Keys == Identity
  - create a new, random key-pair (pk private key, Pk public key)
  - Pk is the public "name" a user can use (usually better to use Hash(Pk))
  - pk lets you "speak for" the identity

- you control the identity, because only you know sk

- if you see sig, such that verify(pk, msg, sig)== true, think of it as

  pk says "[msg]"

- if Pk "looks random", nobody needs to know who you are

DECENTRALIZED IDENTITY MANAGEMENT

- anybody can make a new identity at any time

  make as many identities as you want!

- no central point of coordination

- these identities are called addresses in Bitcoin

- 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 PUBLIC/PRIVATE KEYS

- the private/public key pair is used to uniquely identify the owner of funds of an address.

- when spending bitcoins, the current bitcoin owner presents
  - his/her public key and a signature (different each time, but created from the same private key) in the transaction spending those bitcoins.

  - to verify that a transaction is really created by the owner of the funds

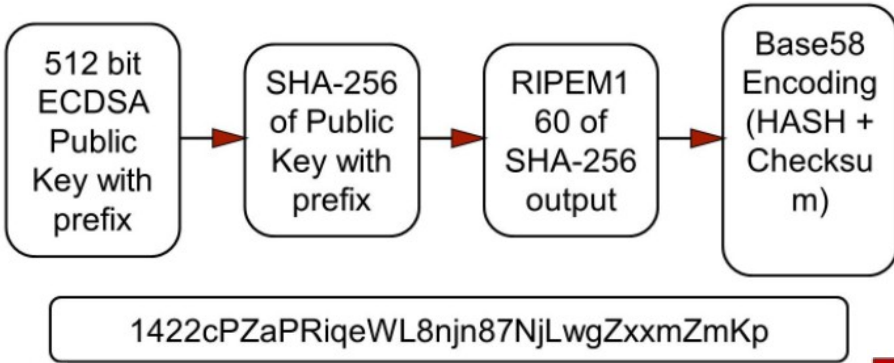
- through the presentation of the public key and signature, everyone in the bitcoin network can verify and accept the transaction as valid,
  - confirms that the person transferring the bitcoins owned them at the time of the transfer.

BITCOIN ADDRESSES

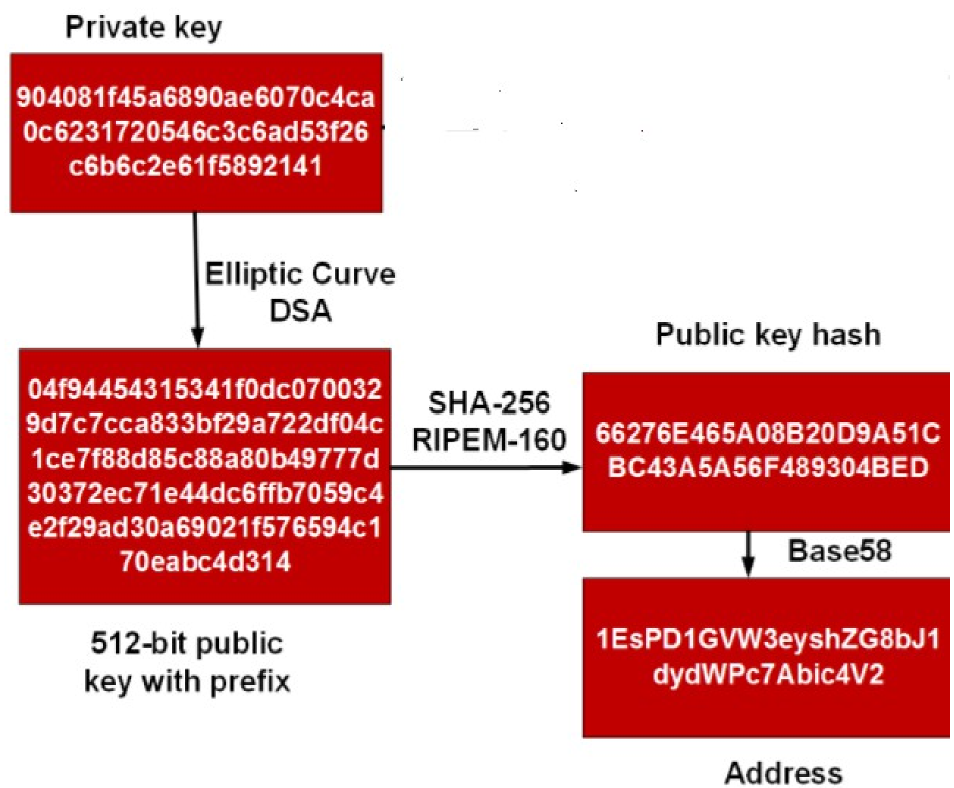
1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy

- derived from a public key
- not an hexadecimal value!
- may be used to identify the recipient of funds in Bitcoin.
  - compare a bitcoin transaction to a paper check, the bitcoin address is the beneficiary, which is what we write on the line after “Pay to the order of.”
- can be shared with anyone who wants to send you money.
- each user may have more addresses
  - easily generate any number of pairs (private key, public key)
  - increases anonimity

BITCOIN ADDRESSES GENERATION



Bitcoin Address



BASE 58 ENCODING

- Base-64 text-based binary encoding
  - uses 26 lower-case letters, 26 capital letters, 10 numerals, and two more characters such as “+” and “/”

- Base 58 is a subset of Base-64
  - a binary-to-text-encoding that uses, that uses only the alphanumeric characters (except 0,O,I and l).

valore	base58
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	A
10	B
11	C
12	D
13	E
14	F

valore	base58
15	G
16	H
17	J
18	K
19	L
20	M
21	N
22	P
23	Q
24	R
25	S
26	T
27	U
28	V
29	W

valore	base58
30	X
31	Y
32	Z
33	a
34	b
35	c
36	d
37	e
38	f
39	g
40	h
41	i
42	j
43	k
44	m

valore	base58
45	n
46	o
47	p
48	q
49	r
50	s
51	t
52	u
53	v
54	w
55	x
56	y
57	z

BASE 58 ENCODING

- Add a 4 byte checksum at the end.
- Encode the final result is then encoded with base58 encoding.

Algorithm 4.1 Construction of Bitcoin addresses from ECDSA public keys

Input : ECDSA public key pk
Output : Bitcoin address $\mathcal{A}$ e.g., 1DR8mXZpK75q7Vipkb1tmp8Wyjz6gDHZBL

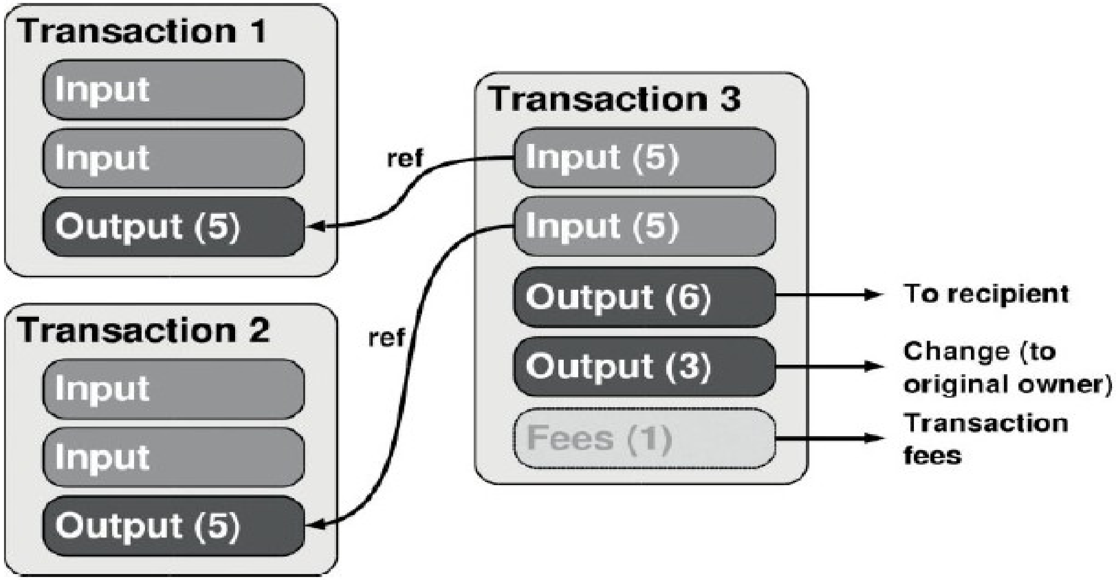
1: $a = \text{0x00} \,||\, \text{RIPEMD160}(\,\text{SHA256}\,(pk)\,)$
2: $h = \text{SHA256}(\,\text{SHA256}(\,a\,))$
3: $\mathcal{A} = \text{Base58}(\,a \,||\, h[251:255]\,)$

TRANSACTIONS

- Centralized 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.
  - transactions inputs and outputs as not related to accounts
  - think of them as bitcoin amounts being locked with a specific secret that only the owner can unlock
- 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

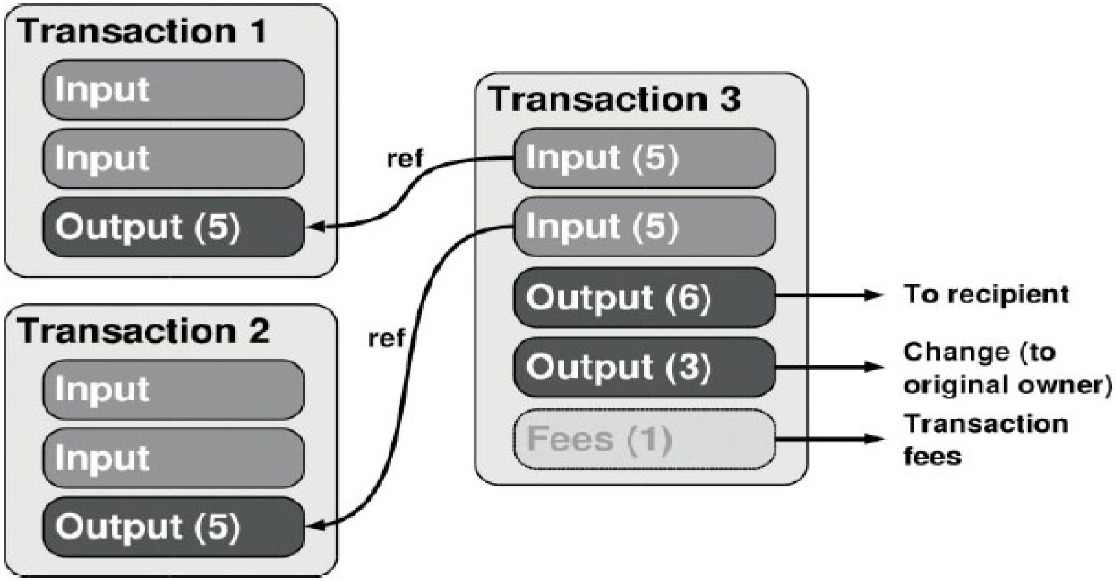
BITCOIN TRANSACTIONS

- transaction represent funds exchange between Bitcoin addresses.
- each transactions is composed by two lists
  - **TxOut,** a list of transaction outputs.
  - **TxIn,** a list of transaction inputs



TRANSACTIONS INPUTS

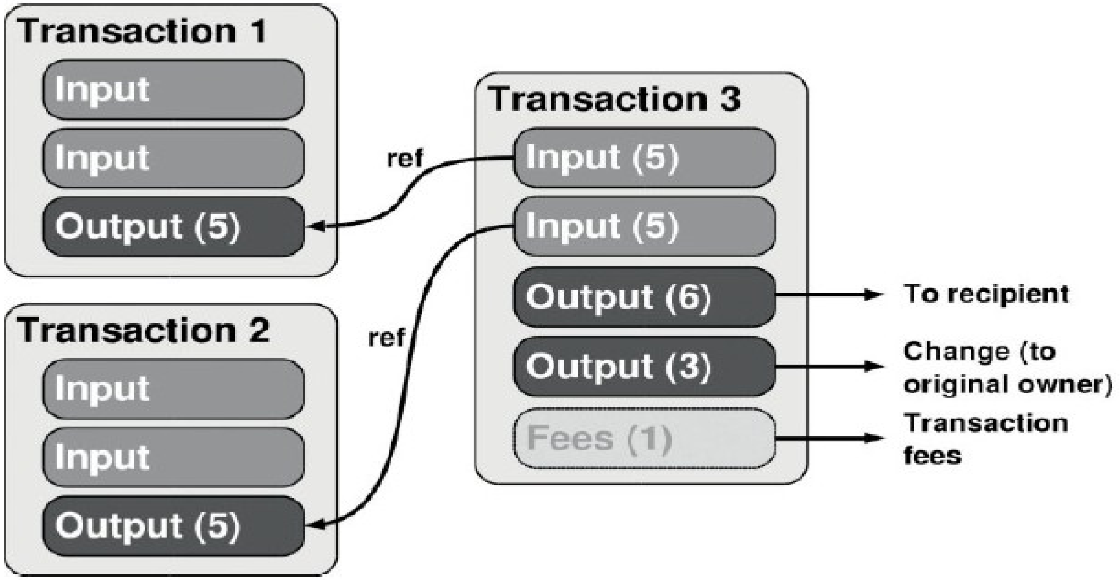
- transaction input: a tuple consisting of
  - a reference to a previously created output
    - hash of the transaction that created the output
    - index of the output within that transaction
  - arguments to the spending condition: to verify that the transaction creator has the permission to spend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



TRANSACTIONS OUTPUTS

each output holds:

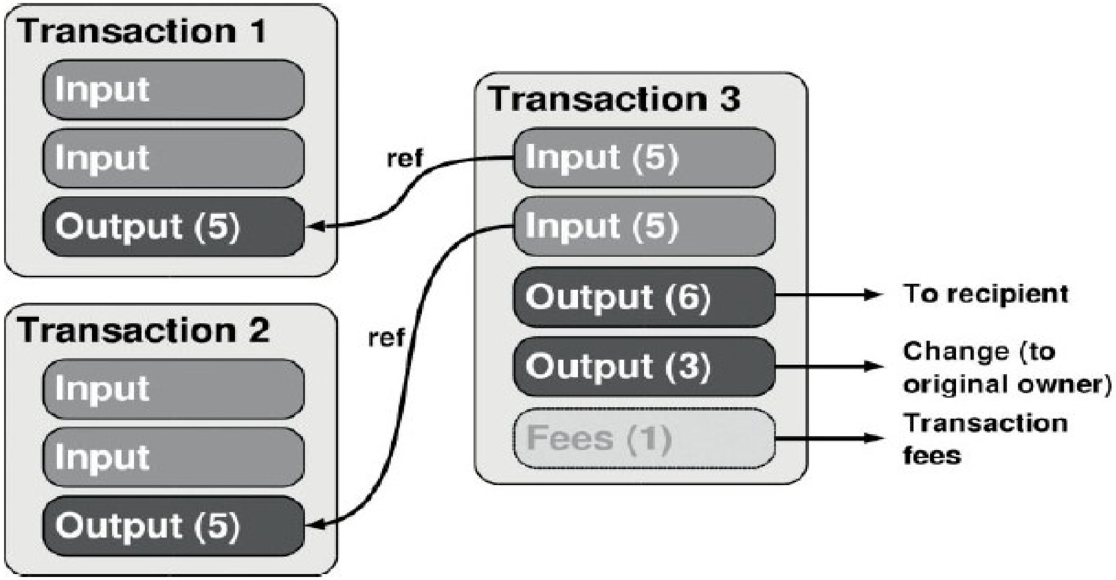
- the recipient address
- an amount (the value in parenthesis)
- a spending condition: determines the conditions that need to be met in order for a transaction to be spent.
  - most common condition: presence of a valid signature



BITCOIN TRANSACTIONS

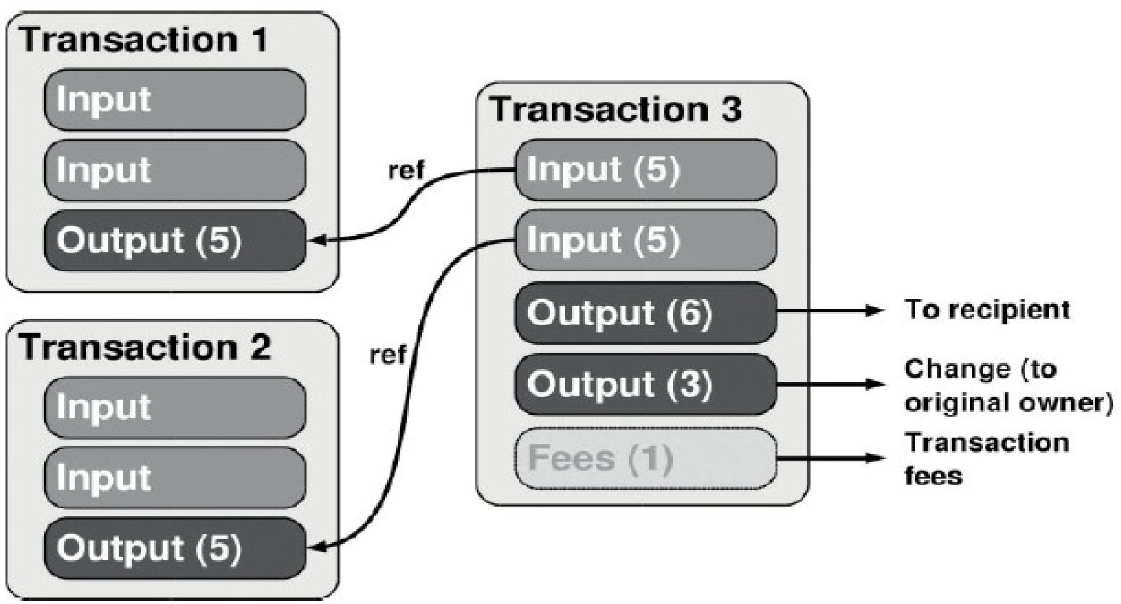
A possible scenario

- 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



TRANSACTIONS CHANGE

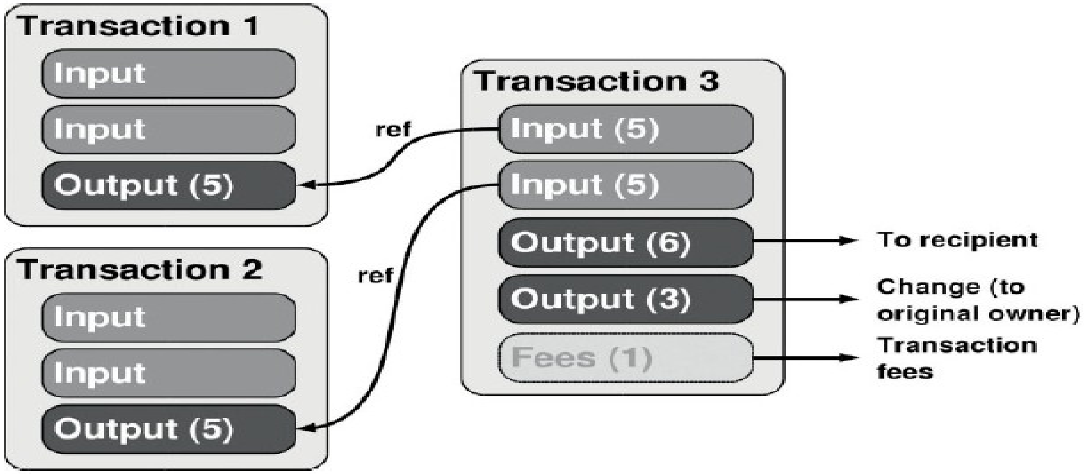
- Each transaction completely uses the input funds: no change is left in the input addresses.
- Change = difference between input sums and the sum we actually want to pay including fees
  - can be kept by using an owned address between the outputs



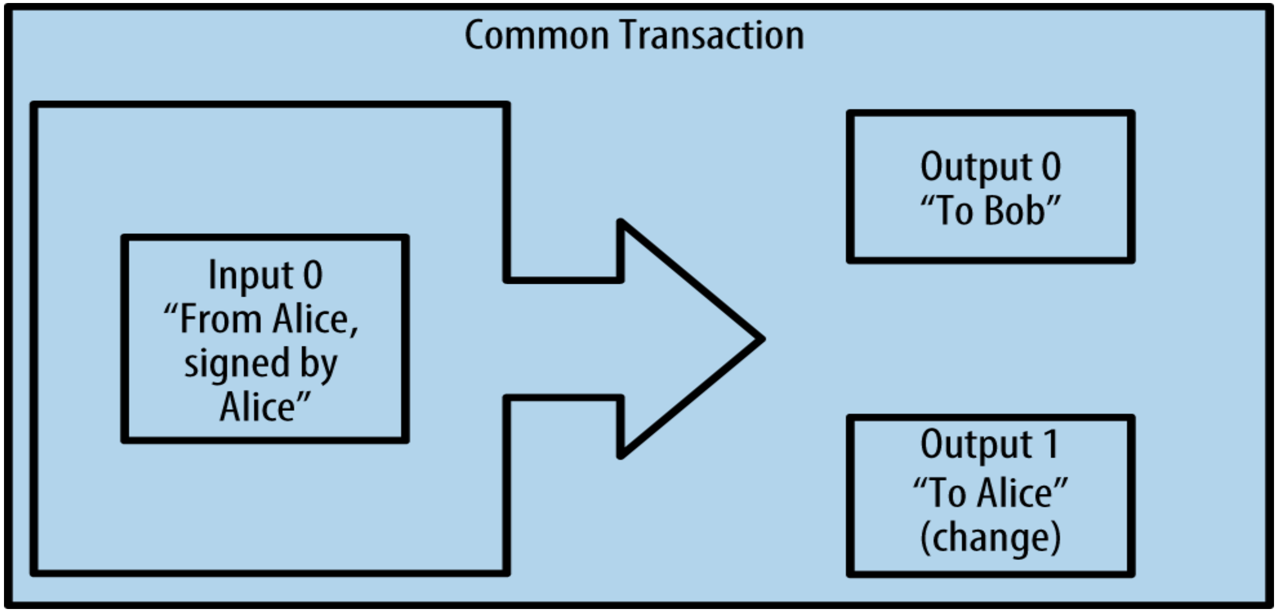
BITCOIN TRANSACTIONS VALIDITY

- A first condition for validity: Σ(input funds) ≥ Σ(output funds). The transaction must not spend more than the available inputs

- Σ(input funds) – Σ(output funds) = transaction fee.
  - collected by the miners as a fee as a reward 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)

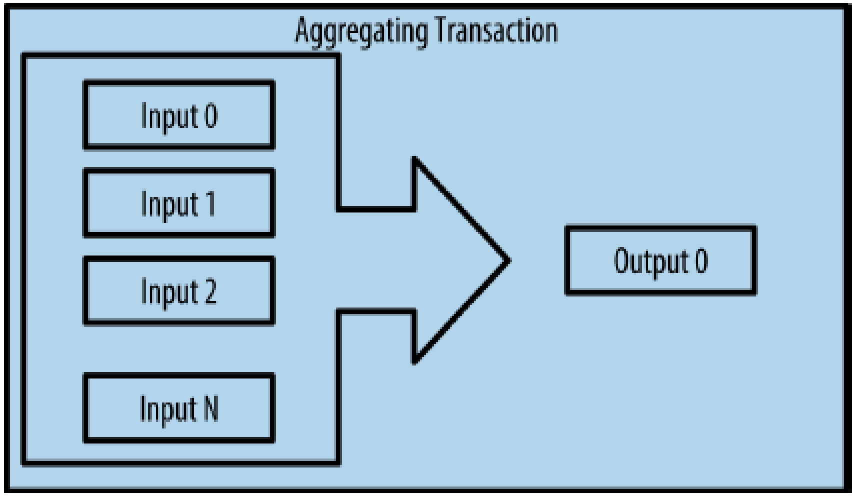


COMMON TYPE OF TRANSACTIONS



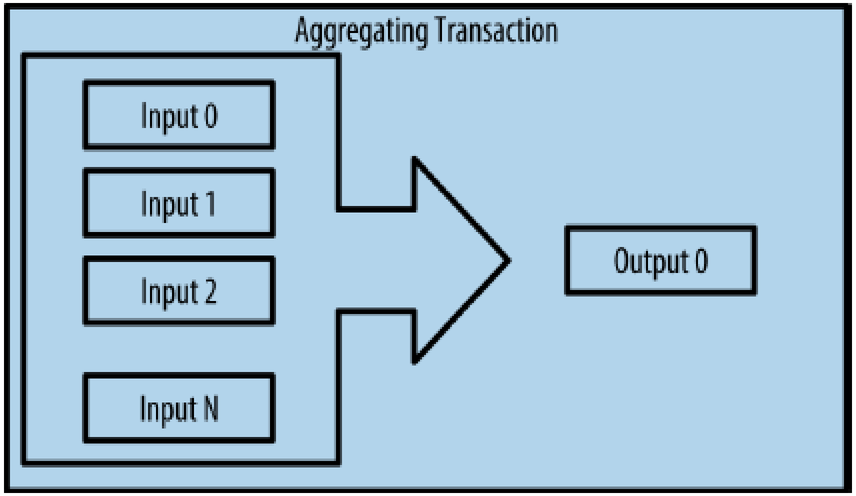
- the most common form of transaction: a simple payment from one address to another
- often includes some "change" returned to the original address.
- this type of transaction has one input and two outputs

AGGREGATING FUNDS



- A transaction aggegating several inputs into a single output
  - the equivalent of echanging 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)

DISTRIBUTING FUNDS



- transactions distributing one input to multiple outputs representing multiple recipients

- used to distribute funds, for instance processing payroll payments to multiple employees

A "REAL" BITCOIN TRANSACTION

metadata

input(s)

output(s)


```
{
  "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
  "ver":1,
  "vin_sz":2,
  "vout_sz":1,
  "lock_time":0,
  "size":404,
  "in":[
   {
    "prev_out":{
     "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
     "n":0
    },
      "scriptSig":"30440..."
   },
   {
    "prev_out":{
     "hash":"7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",
     "n":0
    },
    "scriptSig":"3f3a4ce81...."
   }
  ],
  "out":[
   {
    "value":"10.12287097",
    "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"
   }
  ]
}
```


encoded in JSON

TRANSACTION METADATA

	{
transaction hash	"hash":"5a42590...b8b6b"
housekeeping	"ver":1, "vin_sz":2, "vout_sz":1,
"not valid before"	"lock_time":0,
housekeeping	"size":404,
	...
	}

among other housekeeping information:

- hash of the entire transaction, its unique identifier
- locktime defines the earliest time that a transaction can be added to the blockchain. Set to zero in most transactions to indicate immediate execution

TRANSACTION INPUT

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

previous transaction

signature

(more inputs)

a JSON array

- each element contains a pointer to a previous transaction (its hash), the index of the previous transaction's output
- a script: scripSig

TRANSACTION OUTPUT

output value

recipient address

(more outputs)


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


a JSON array

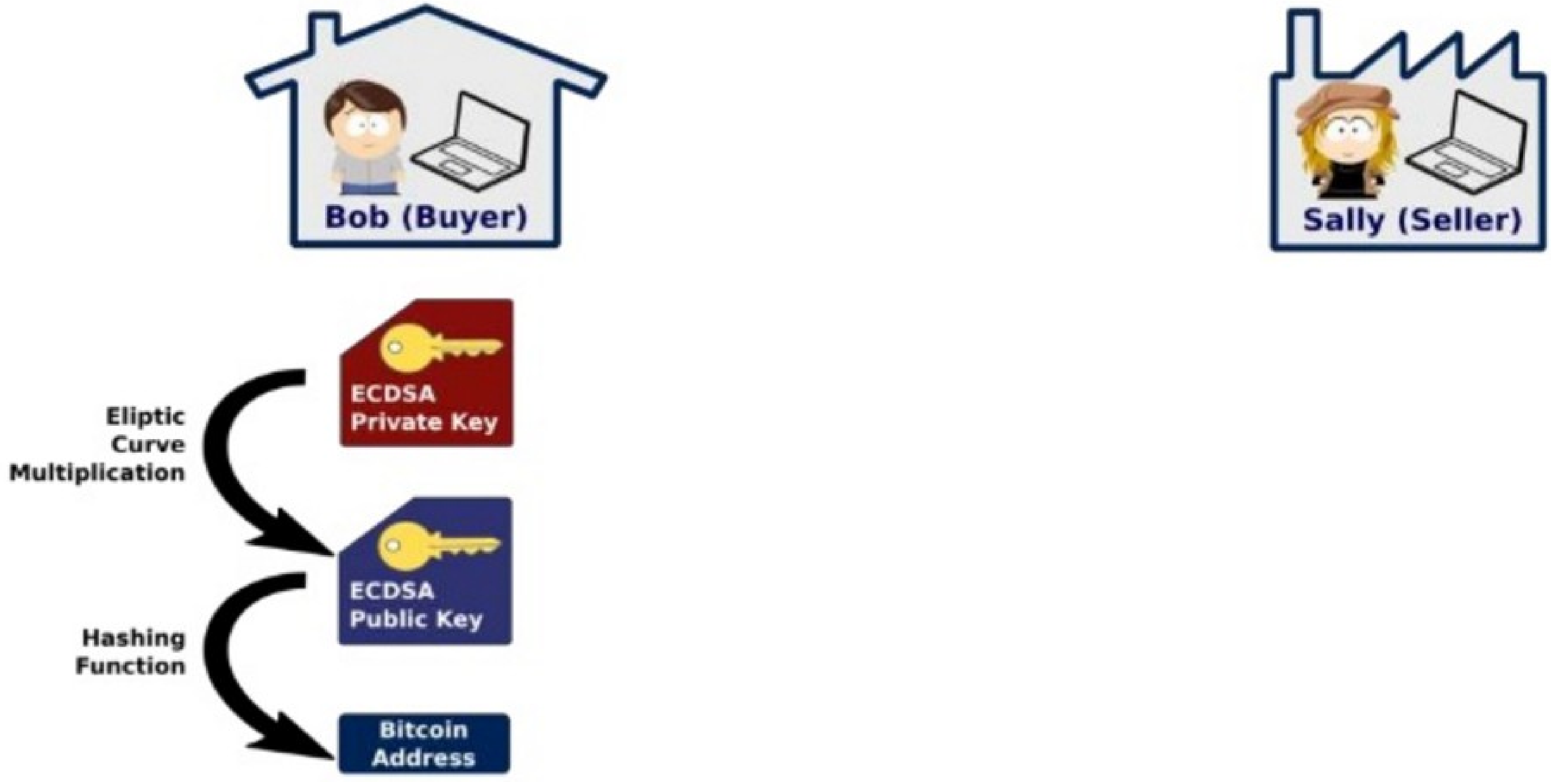
- each element contains the value to be transferred
- a script which may include:
  - the hash address of the receiver
  - the publicKey of the receiver (shown in the example) from which the address is computed

BITCOIN SCRIPTS

- a script is a piece of code that verifies a set of arbitrary conditions that must be met in order to spend coins
- most common type of script: redeem a previous transactio by signing it with the correct key
  - 99% are simple signatures checks
  - 0.01% are MULTISIG
  - 0.01% are Pay-to-Script-Hash
  - remainder proof-of-burn
- scripts have been introduced to specify also more complex spendingconditions
  - escrow transactions
  - green addresses
  - micro payments

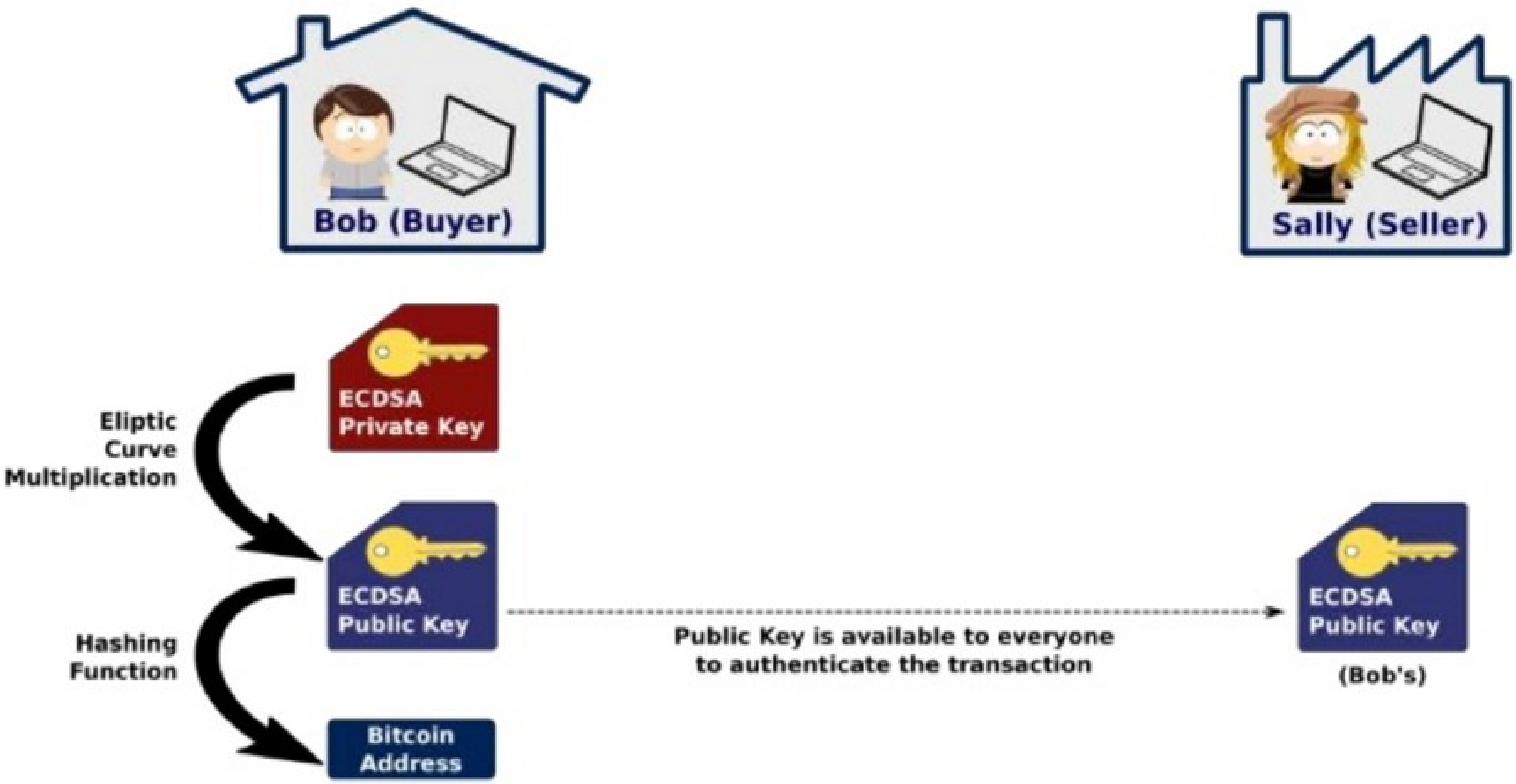
WHAT IS THE TASK OF A SCRIPT?

- Proving that someone has the right to spend the bitcoins



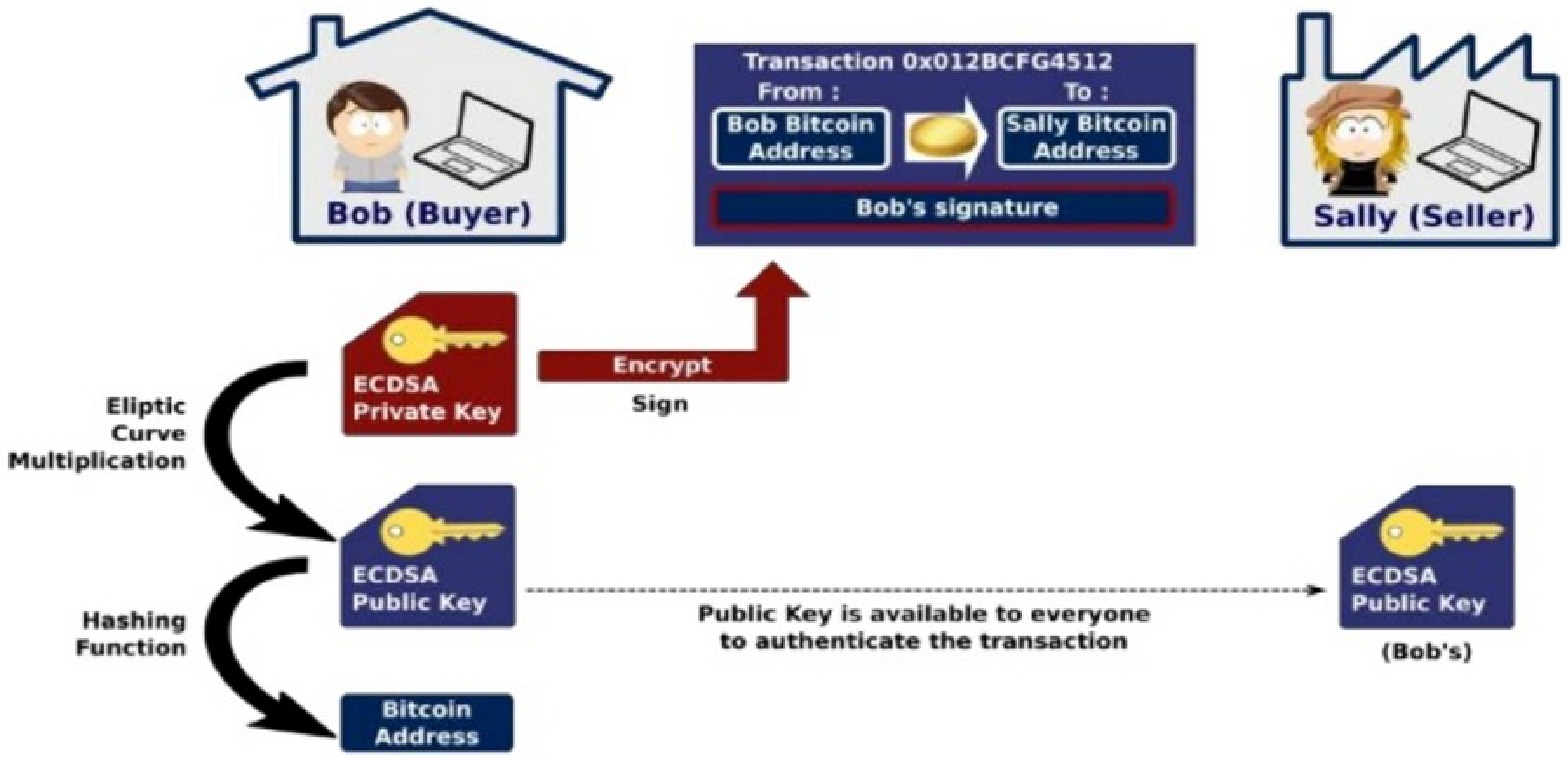
WHAT IS THE TASK OF A SCRIPT?

- Proving that someone has the right to spend the bitcoins



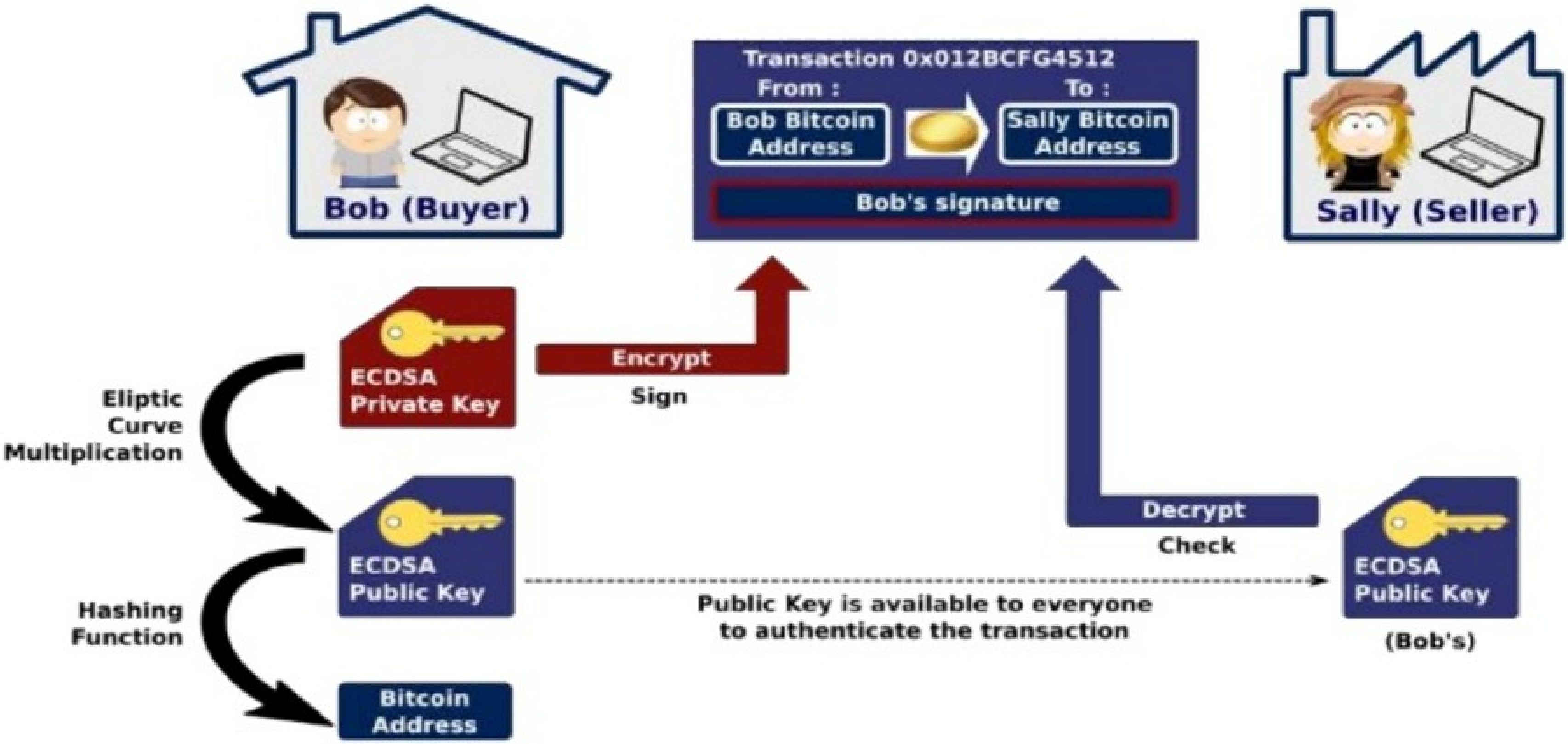
WHAT IS THE TASK OF A SCRIPT?

- Proving that someone has the right to spend the bitcoins



WHAT IS THE TASK OF A SCRIPT?

- Proving that someone has the right to spend the bitcoins



BITCOIN SCRIPTS

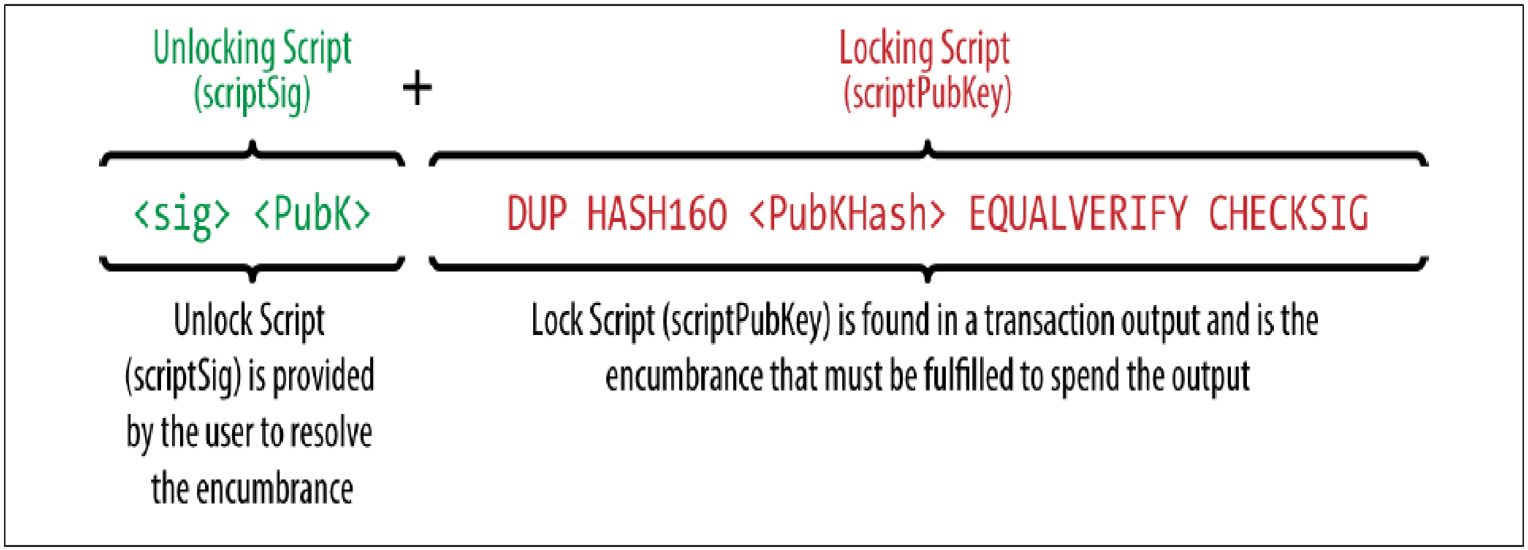
- Bitcoin's transaction validation engine relies on two types of scripts to validate transactions:
  - a locking script
  - an unlocking script.
- In a transaction output there is a locking script
  - specifies the conditions that must be met to spend the output in the future.
  - *ScriptPubKey:* usually contains a public key
- In a transaction input there is an unlocking script
  - "solves," or satisfies, the conditions placed on an output by a locking script
  - allows the output to be spent.
  - *ScriptSig:* it usually contained a digital signature.

BITCOIN SCRIPTS

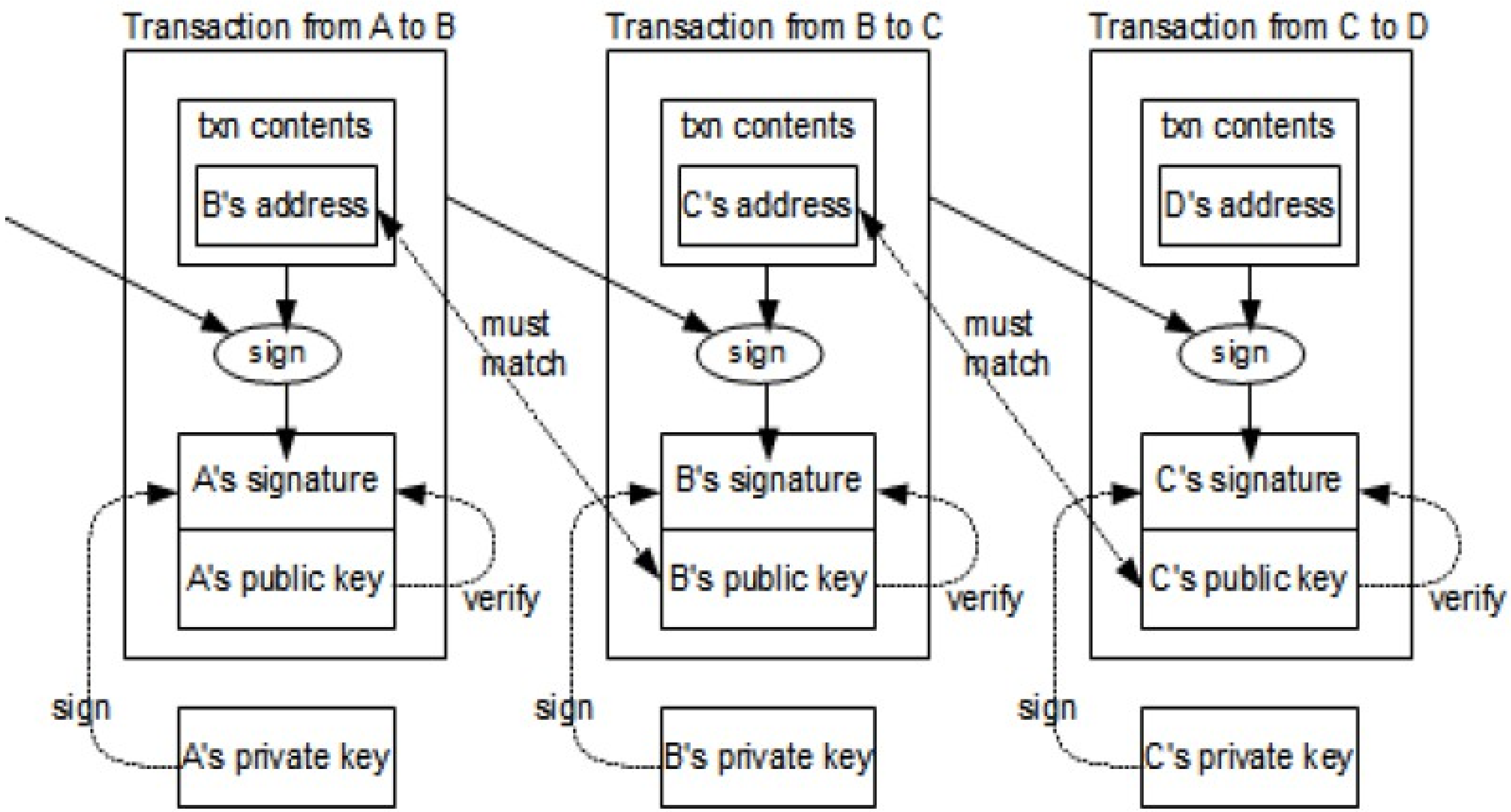
- Every bitcoin client will validate transactions by concatenating and then executing the locking and unlocking scripts together.

- For each input in the transaction, the validation software
  - retrieves the referenced output
  - that output contains a locking script defining the conditions required to spend it.
  - generally the public key of the owner
  - take the unlocking script contained in the input that is attempting to spend and execute the two scripts.
  - verify the signature

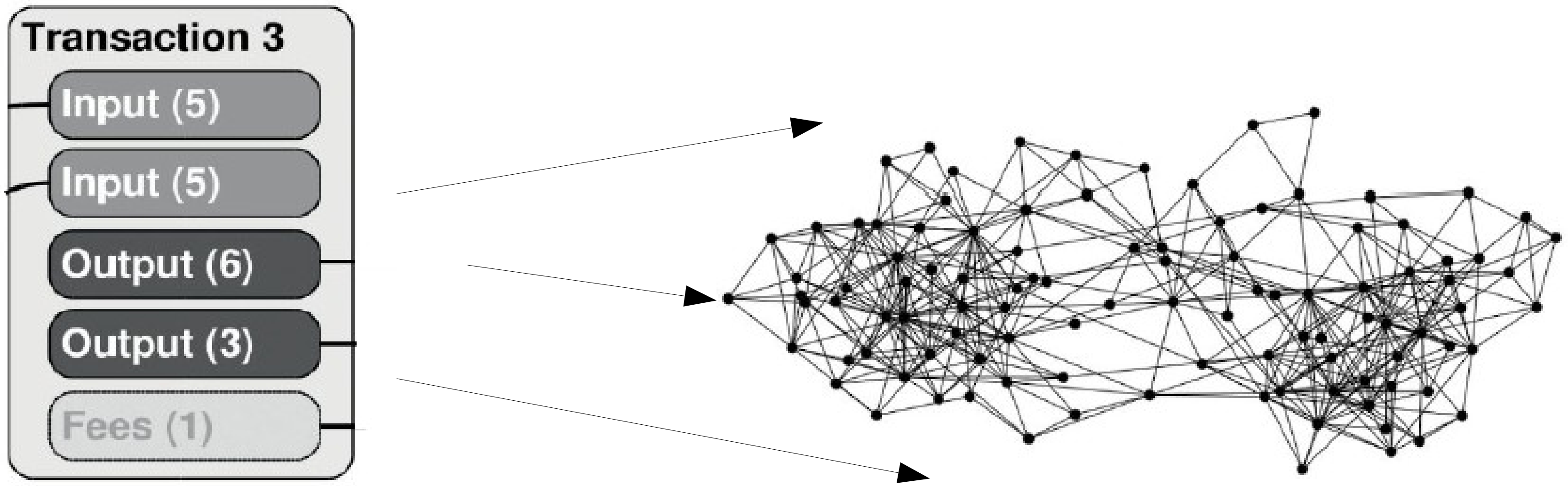
BITCOIN SCRIPTS



TRANSACTIONS SIGNATURES



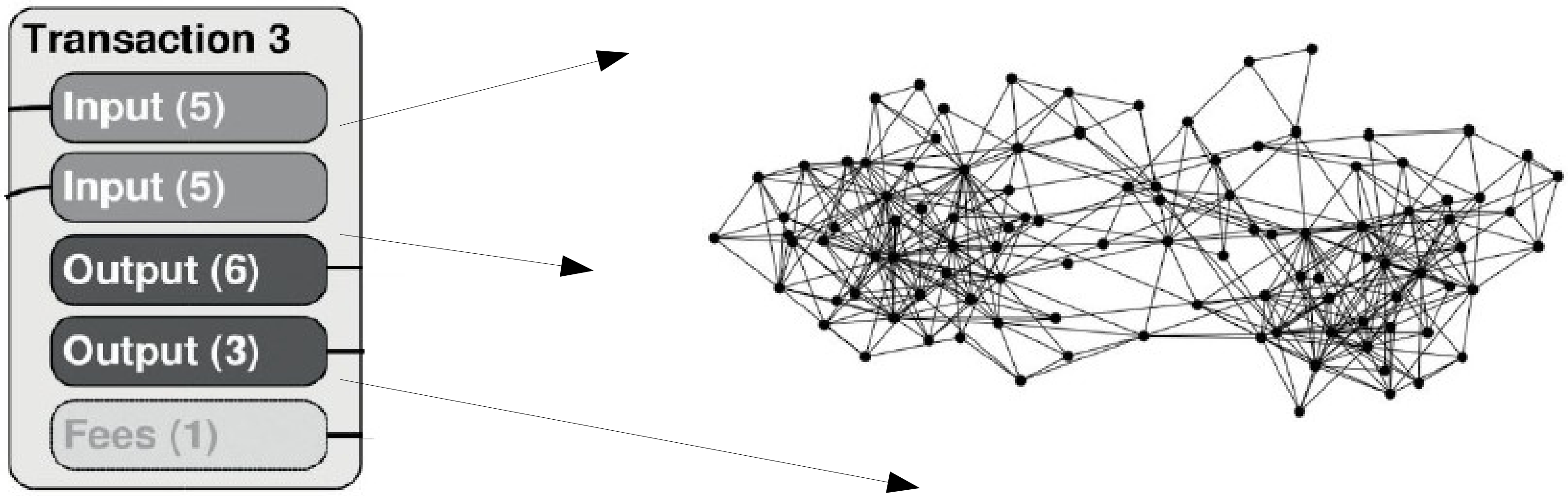
THE P2P NETWORK



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 P2P NETWORK



- The peers must agree on the order in which transactons 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 restablished by distributed consensus
    - allows to keep a distributed, replicated consistent ledger including all the transactions (we will see in the next lesson)

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 inut 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

TRANSACTIONS LIFECYCLE

- A transaction's lifecycle starts with the transaction's creation
- The transaction is then signed with one or more signatures indicating the authorization to spend the funds referenced by the transaction.
- The transaction is then broadcast on the bitcoin network
- Each network node (participant) validates and propagates the transaction until it reaches (almost) every node in the network.
- Finally, the transaction is verified by a mining node and included in a block of transactions that is recorded on the blockchain.
- Once recorded on the blockchain and confirmed by sufficient subsequent blocks (confirmations), the transaction is a permanent part of the bitcoin ledger
- 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
- 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

UNSPENT TRANSACTION OUTPUTS

- **Unspent transaction outputs (UTXO)**: represents the shared space of the Bitcoin network
- We can say that "the state of Bitcoin reside in the unspent outputs of the transactions"
- More complex representation will be needed to represent tha state of the Ethereum network

UNSPENT TRANSACTION OUTPUTS CACHE

- Bitcoin client maintains an **unspent transaction output cache**
  - a cache containing only transactions having UUTXO
  - useful to check validity of new transactions
- Advantage of using the UTXO cache : it is much smaller than the whole transactions 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

MANAGING 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 (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 pool
- different unspent transaction outputs in different nodes because of double spending
- eventual consisteny

THE DOUBLE SPENDING PROBLEM

A transaction is valid if:

- the transaction is structurally correct (output funds do not exceed input funds, …)

- input funds are used by its rightful owner

- the input funds do exist and were not already spent in a previous transaction
    - double spending problem!

- the digital signature guaranties that only the rightful owner can spend the funds, but it does not prevent it from spending them more than once in different transactions
    - a different mechanism is required
    - one of the most important challange to define a cryptocurrency
    - we will see in the next lessons