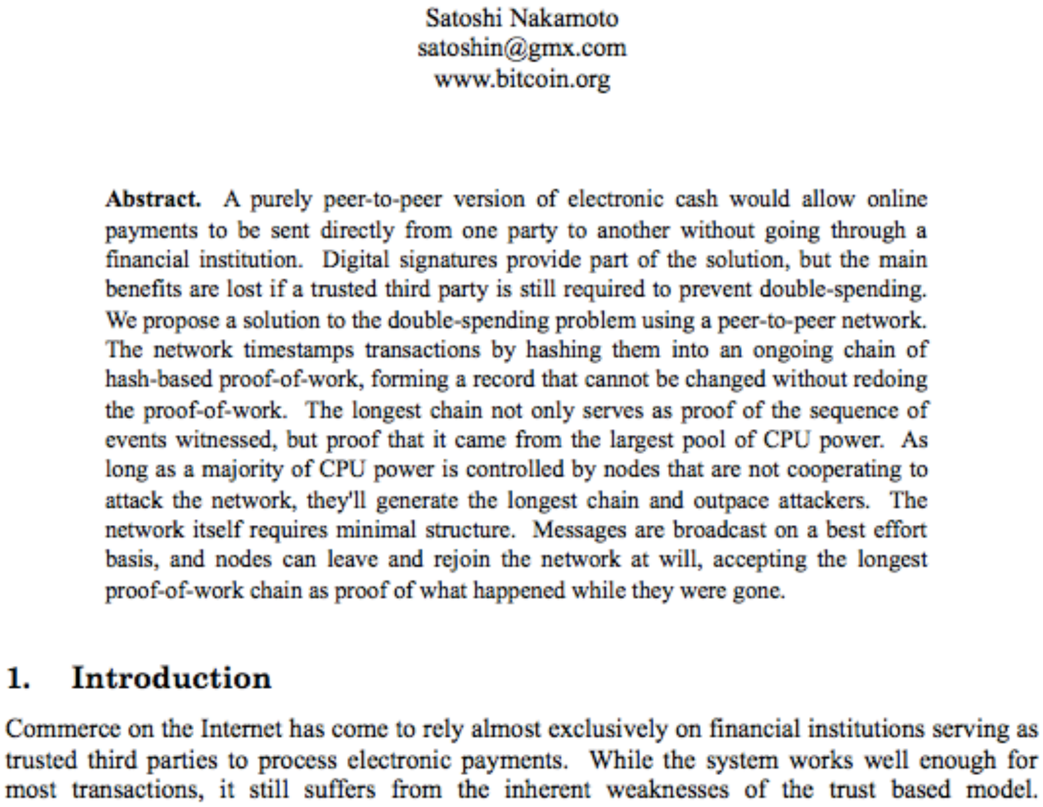
BITCOIN

Murat Osmanoglu

BITCOIN



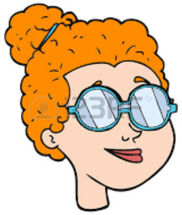
Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model.

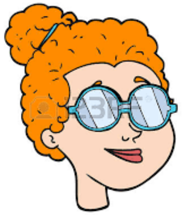
BITCOIN



Necla	13.2
Ali	23.2
Bulent	15
. . .	. . .

BITCOIN

Ali sends 3 bitcoins to Bulent



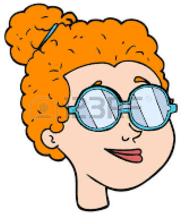
DIGITAL BANK

Necla	13.2
Ali	23.2
Bulent	15
. . .	. . .

BITCOIN

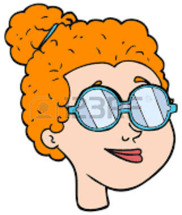
Ali sends 3 bitcoins to Bulent

Bulent sends 2 bitcoins to Necla



Necla	13.2
Ali	23.2
Bulent	15
. . .	. . .

BITCOIN



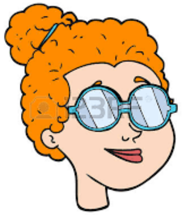
Ali sends 3 bitcoins to Bulent

Bulent sends 2 bitcoins to Necla

Ali sends 5 bitcoins to Necla

Necla	13.2
Ali	23.2
Bulent	15
. . .	. . .

BITCOIN



Ali sends 3 bitcoins to Bulent

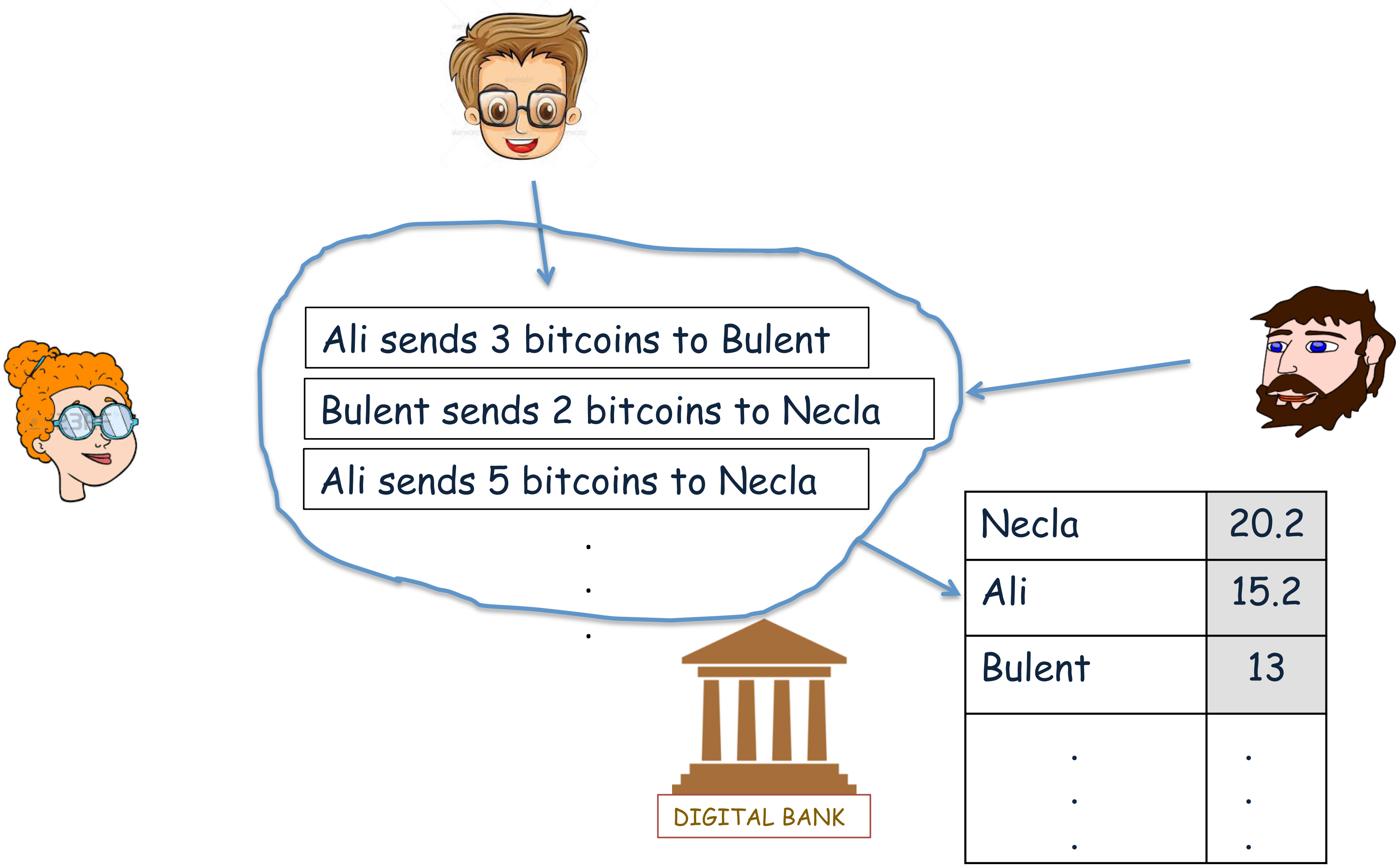
Bulent sends 2 bitcoins to Necla

Ali sends 5 bitcoins to Necla

.
.
.

Necla	13.2
Ali	23.2
Bulent	15
. . .	. . .

BITCOIN



Ali sends 3 bitcoins to Bulent

Bulent sends 2 bitcoins to Necla

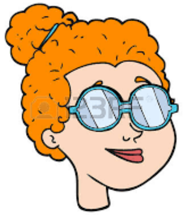
Ali sends 5 bitcoins to Necla

.
.
.

DIGITAL BANK

Necla	20.2
Ali	15.2
Bulent	13
. . .	. . .

BITCOIN

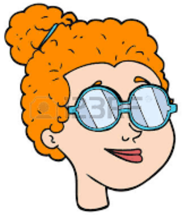


Ali sends 3 bitcoins to Bulent

Necla	13.2
Ali	23.2
Bulent	15
. . .	. . .

DIGITAL BANK

BITCOIN

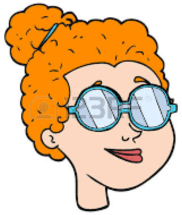


Ali sends 3 bitcoins to Bulent

Ali sends 2 bitcoins to Bulent

Necla	13.2
Ali	23.2
Bulent	15
. . .	. . .

BITCOIN



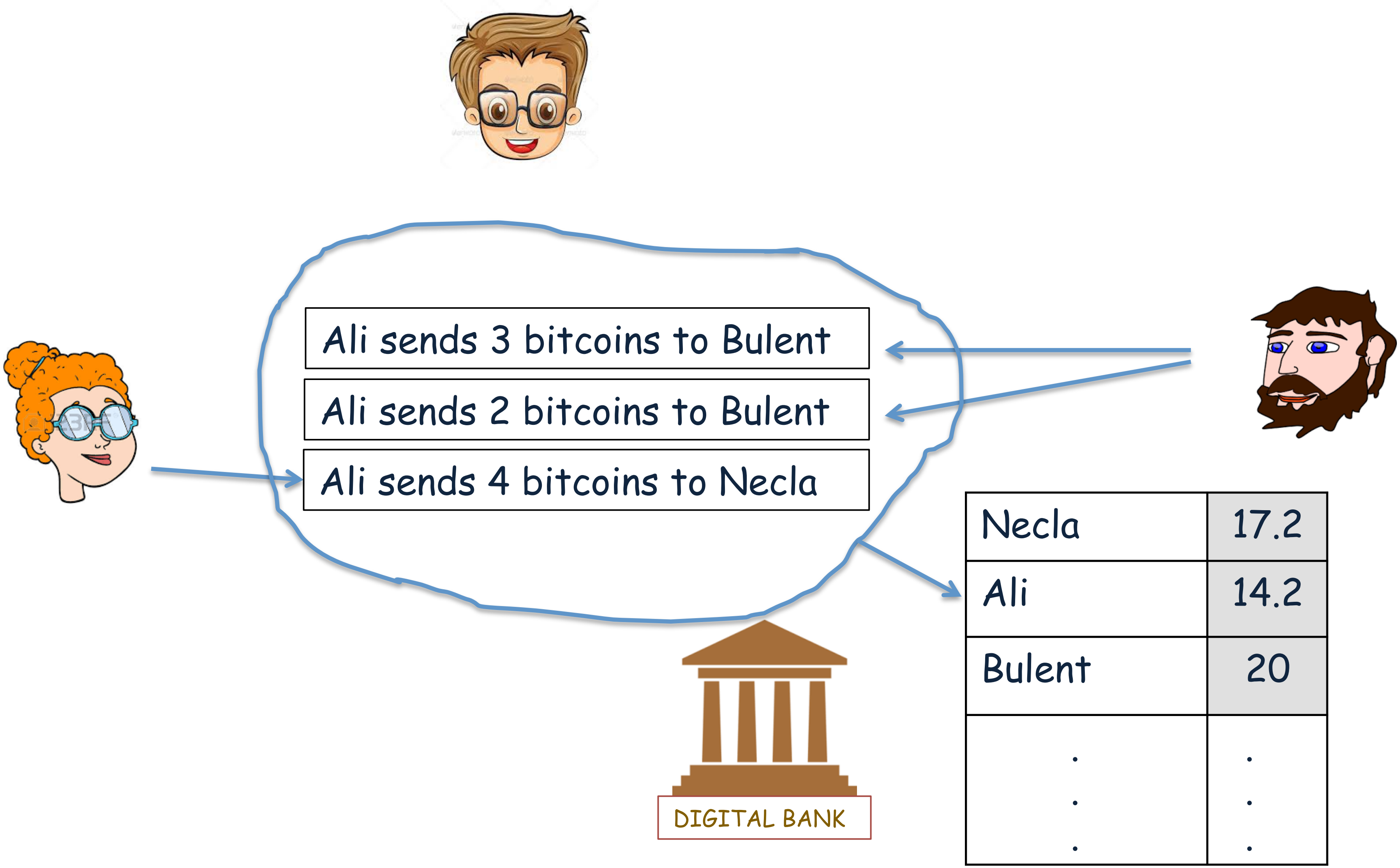
Ali sends 3 bitcoins to Bulent

Ali sends 2 bitcoins to Bulent

Ali sends 4 bitcoins to Necla

Necla	13.2
Ali	23.2
Bulent	15
. . .	. . .

BITCOIN



Ali sends 3 bitcoins to Bulent

Ali sends 2 bitcoins to Bulent

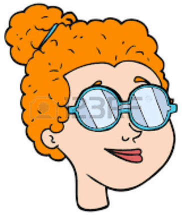
Ali sends 4 bitcoins to Necla

Necla	17.2
Ali	14.2
Bulent	20
. . .	. . .

DIGITAL BANK

BITCOIN

digital signature

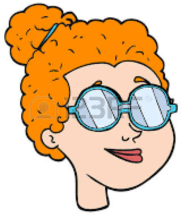


Necla	13.2
Ali	23.2
Bulent	15
. . .	. . .

BITCOIN

digital signature

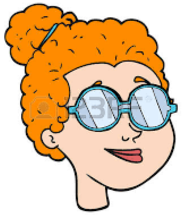
Ali sends 3 bitcoins to Bulent



Necla	13.2
Ali	23.2
Bulent	15
. . .	. . .

BITCOIN

digital signature



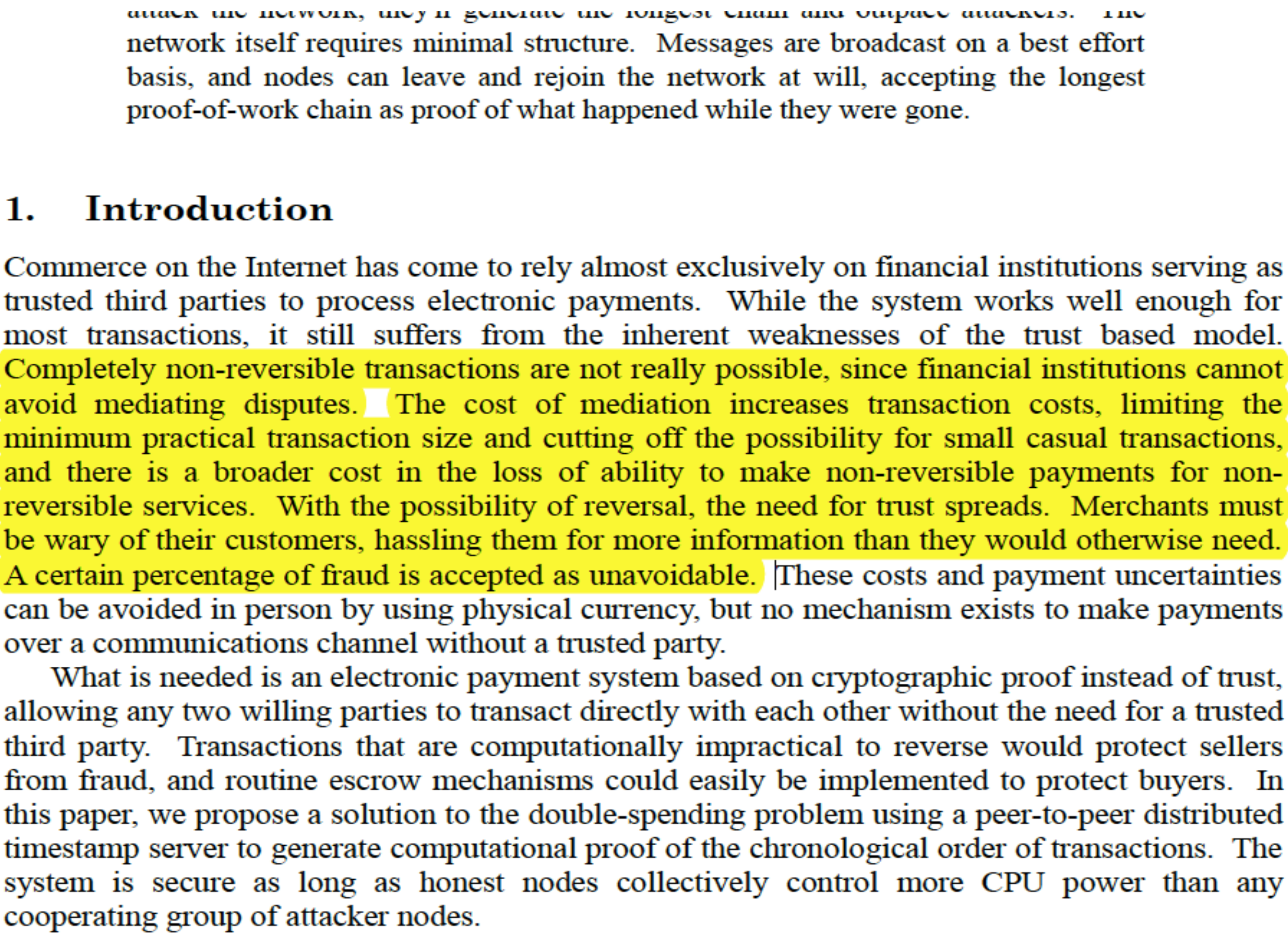
Ali sends 3 bitcoins to Bulent

Bulent sends 2 bitcoins to Necla

DIGITAL BANK

Necla	13.2
Ali	23.2
Bulent	15
. . .	. . .

BITCOIN



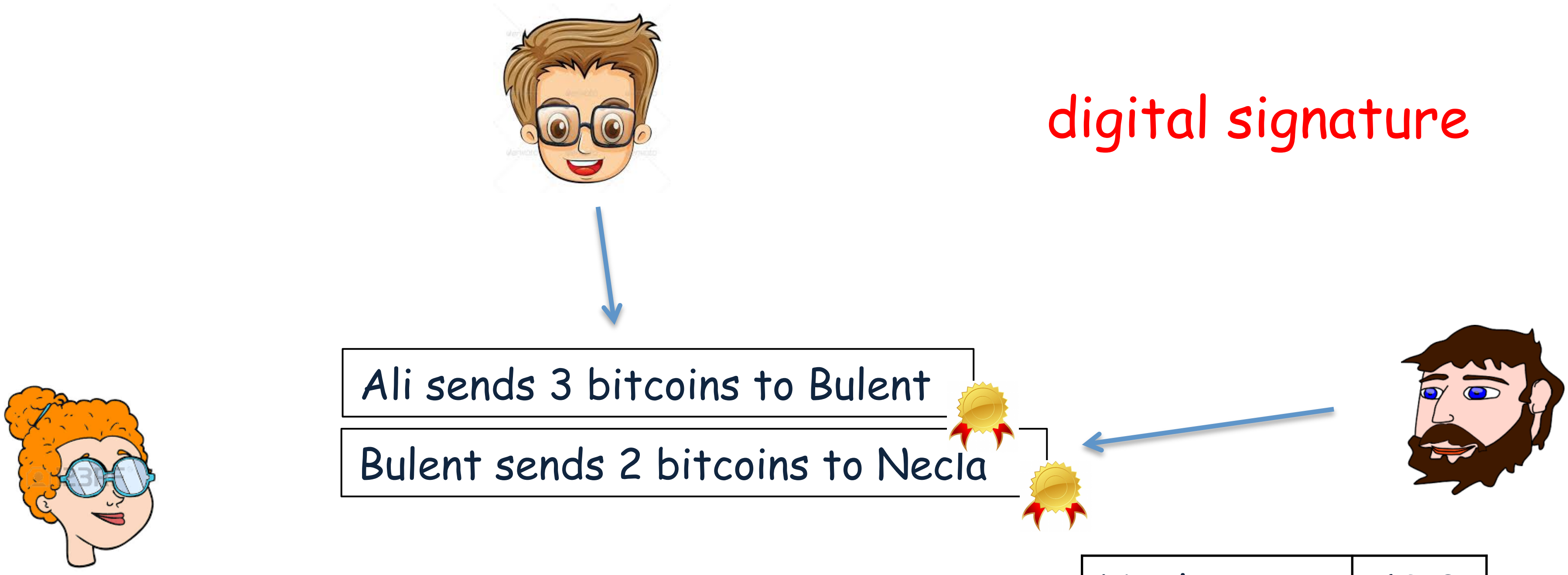
attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for non-reversible services. With the possibility of reversal, the need for trust spreads. Merchants must be wary of their customers, hassling them for more information than they would otherwise need. A certain percentage of fraud is accepted as unavoidable. These costs and payment uncertainties can be avoided in person by using physical currency, but no mechanism exists to make payments over a communications channel without a trusted party.

What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

BITCOIN



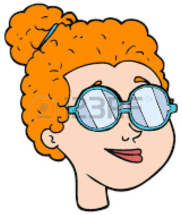
Ali sends 3 bitcoins to Bulent

Bulent sends 2 bitcoins to Necla

digital signature

Necla	13.2
Ali	23.2
Bulent	15
⋮	⋮

BITCOIN

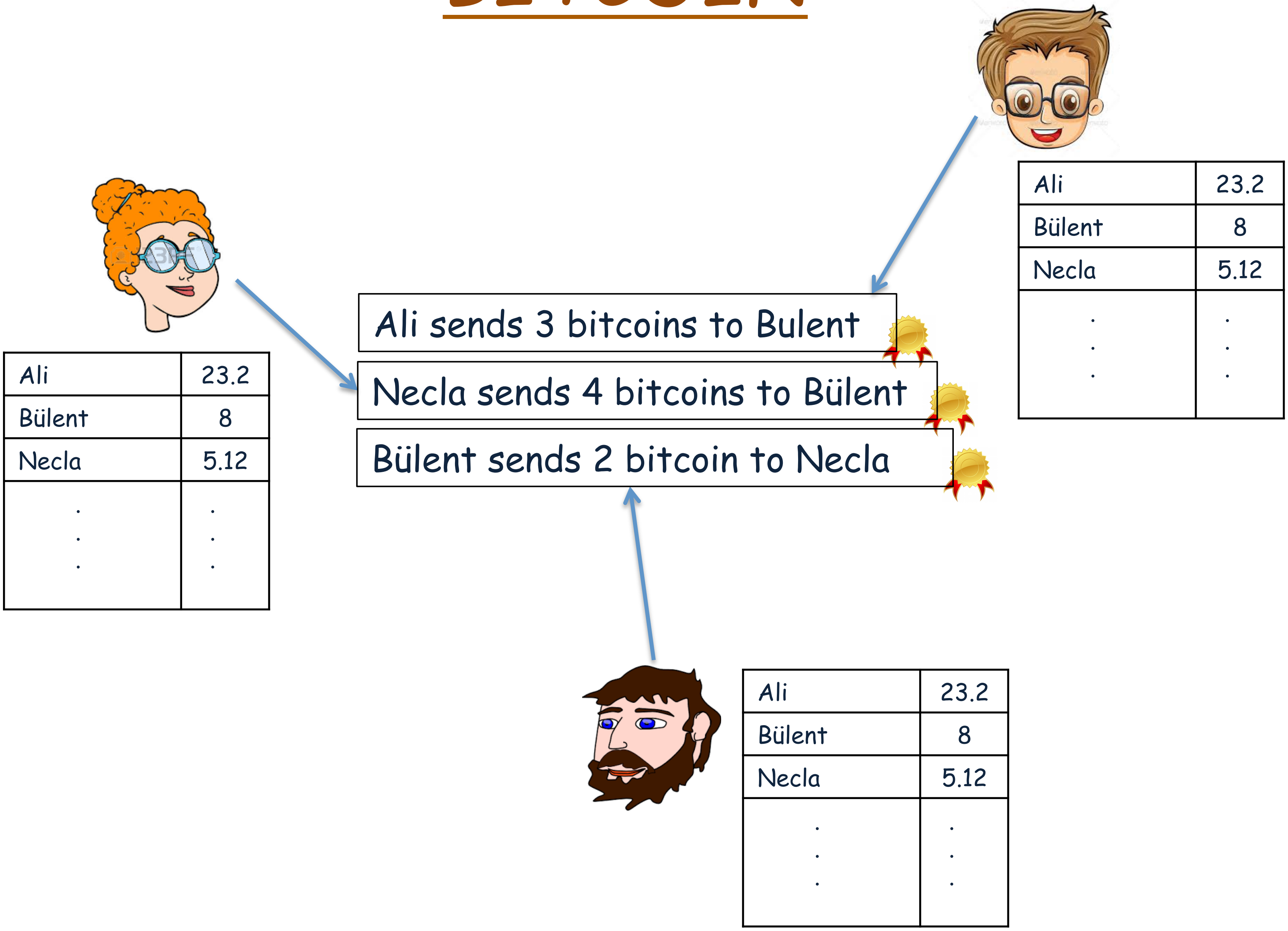


Ali	23.2
Bülent	8
Necla	5.12
. . .	. . .

Ali	23.2
Bülent	8
Necla	5.12
. . .	. . .

Ali	23.2
Bülent	8
Necla	5.12
. . .	. . .

BITCOIN



Ali sends 3 bitcoins to Bulent

Necla sends 4 bitcoins to Bülent

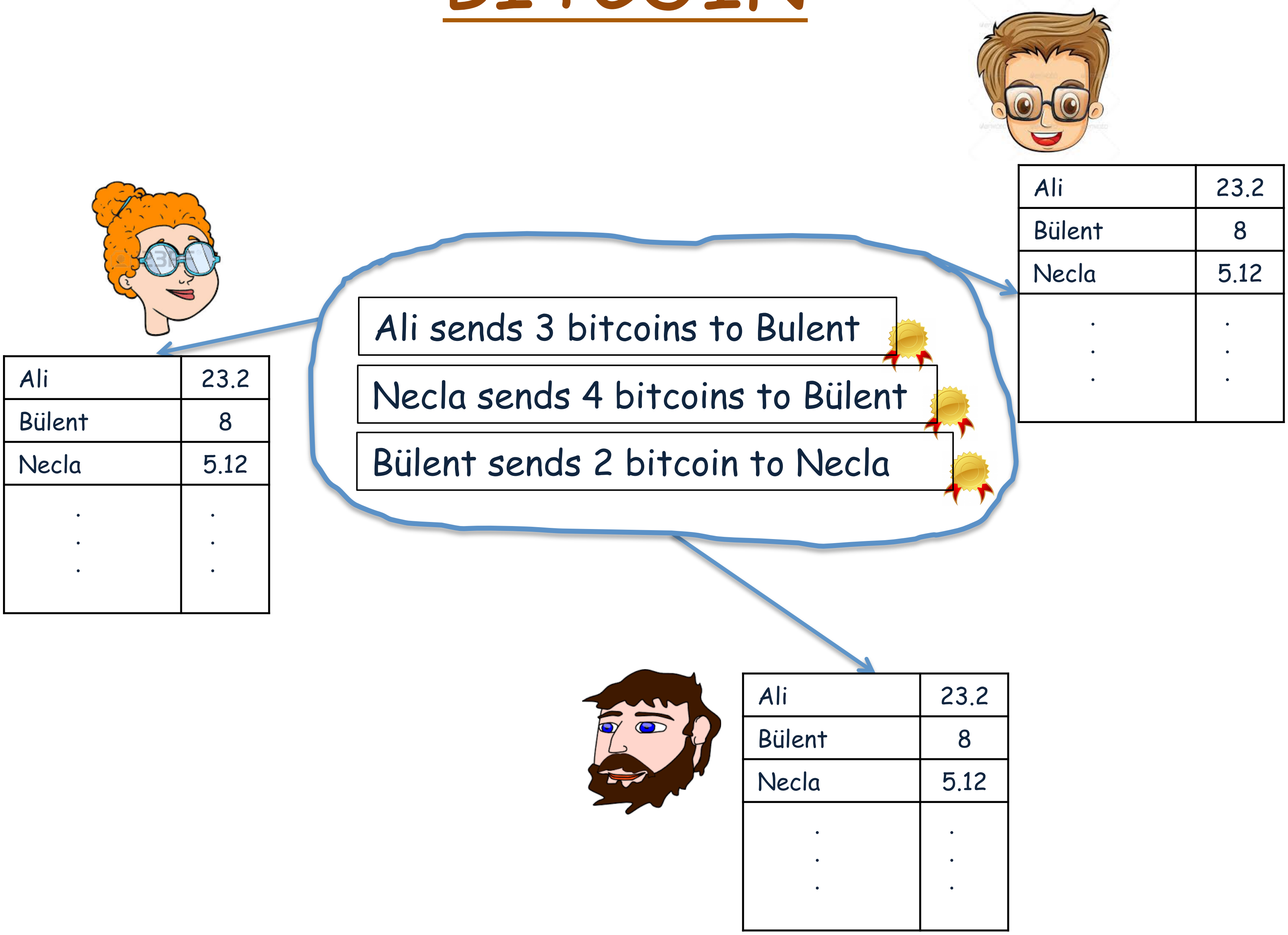
Bülent sends 2 bitcoin to Necla

Ali	23.2
Bülent	8
Necla	5.12
.	.
.	.
.	.

Ali	23.2
Bülent	8
Necla	5.12
.	.
.	.
.	.

Ali	23.2
Bülent	8
Necla	5.12
.	.
.	.
.	.

BITCOIN



Ali sends 3 bitcoins to Bulent

Necla sends 4 bitcoins to Bülent

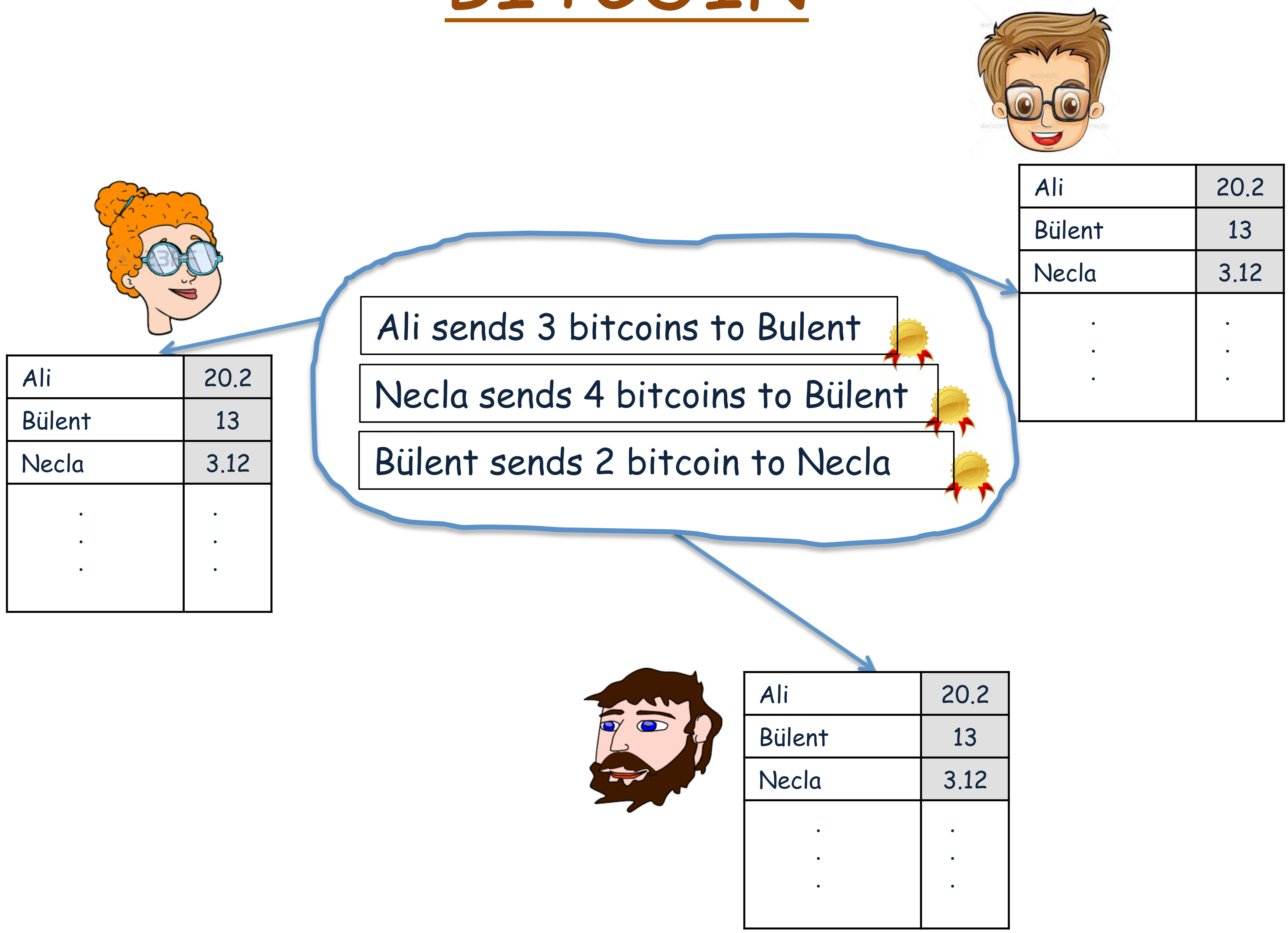
Bülent sends 2 bitcoin to Necla

Ali	23.2
Bülent	8
Necla	5.12
⋮	⋮

Ali	23.2
Bülent	8
Necla	5.12
⋮	⋮

Ali	23.2
Bülent	8
Necla	5.12
⋮	⋮

BITCOIN



Ali sends 3 bitcoins to Bulent

Necla sends 4 bitcoins to Bülent

Bülent sends 2 bitcoin to Necla

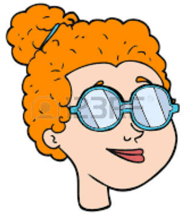
Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

BITCOIN

Ali	20.2
Bülent	15
Necla	1.12
⋮	⋮



Ali	23.2
Bülent	10
Necla	3.12
⋮	⋮

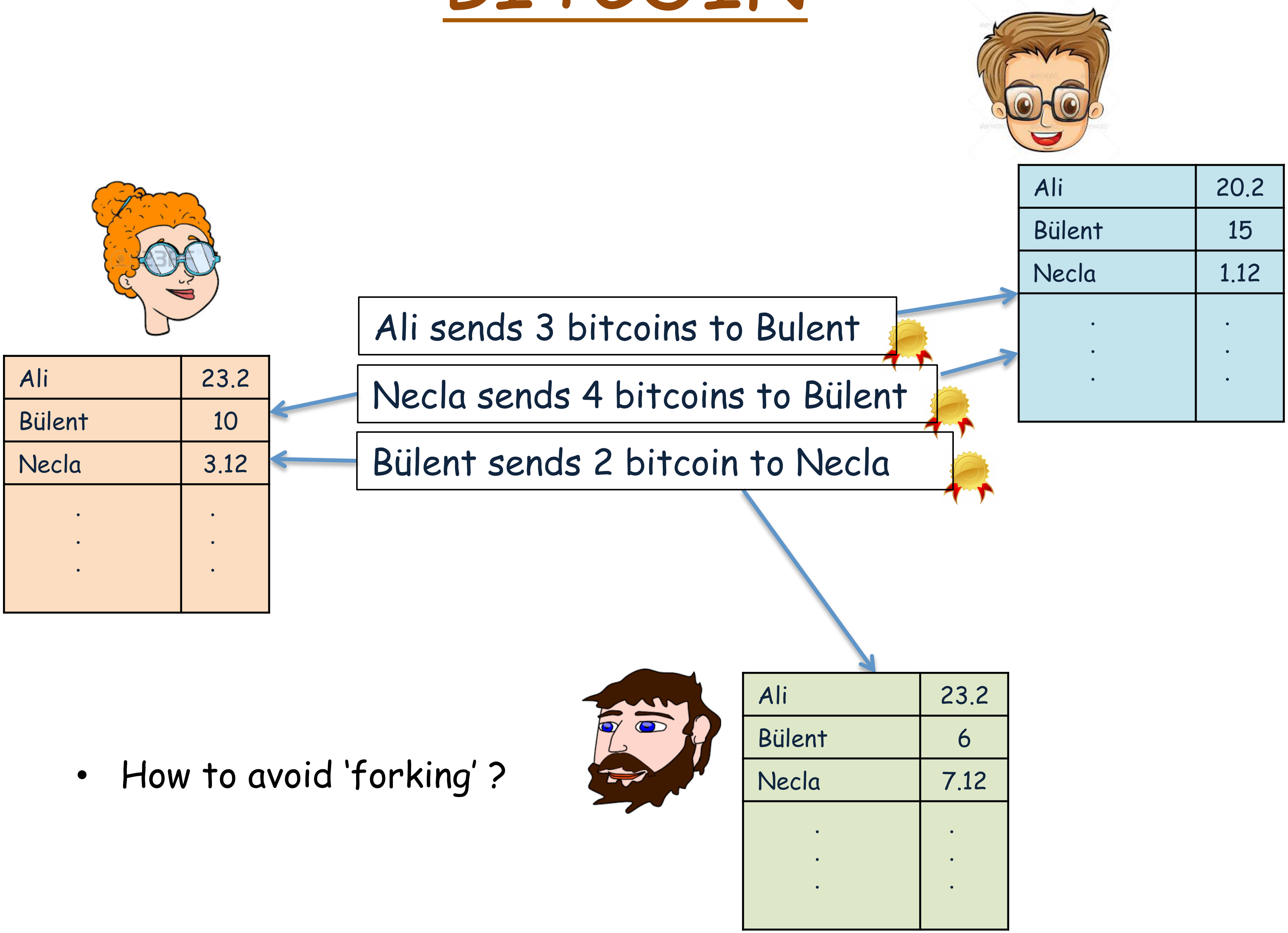
Ali sends 3 bitcoins to Bulent

Necla sends 4 bitcoins to Bülent

Bülent sends 2 bitcoin to Necla

Ali	23.2
Bülent	6
Necla	7.12
⋮	⋮

BITCOIN



Ali sends 3 bitcoins to Bulent

Necla sends 4 bitcoins to Bülent

Bülent sends 2 bitcoin to Necla

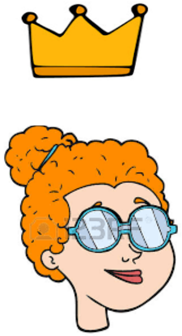
Ali	23.2
Bülent	10
Necla	3.12
⋮	⋮

Ali	20.2
Bülent	15
Necla	1.12
⋮	⋮

Ali	23.2
Bülent	6
Necla	7.12
⋮	⋮

- How to avoid ‘forking’ ?

BITCOIN

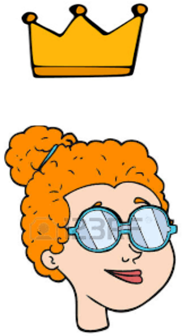


Ali	23.2
Bülent	8
Necla	5.12
⋮	⋮

Ali	23.2
Bülent	8
Necla	5.12
⋮	⋮

Ali	23.2
Bülent	8
Necla	5.12
⋮	⋮

BITCOIN



Ali	23.2
Bülent	8
Necla	5.12
⋮	⋮

Ali sends 3 bitcoins to Bulent

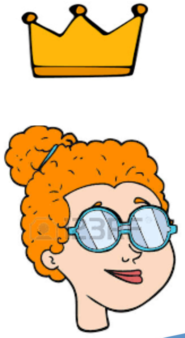
Necla sends 4 bitcoins to Bülent

Bülent sends 2 bitcoin to Necla

Ali	23.2
Bülent	8
Necla	5.12
⋮	⋮

Ali	23.2
Bülent	8
Necla	5.12
⋮	⋮

BITCOIN



Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

Ali sends 3 bitcoins to Bulent

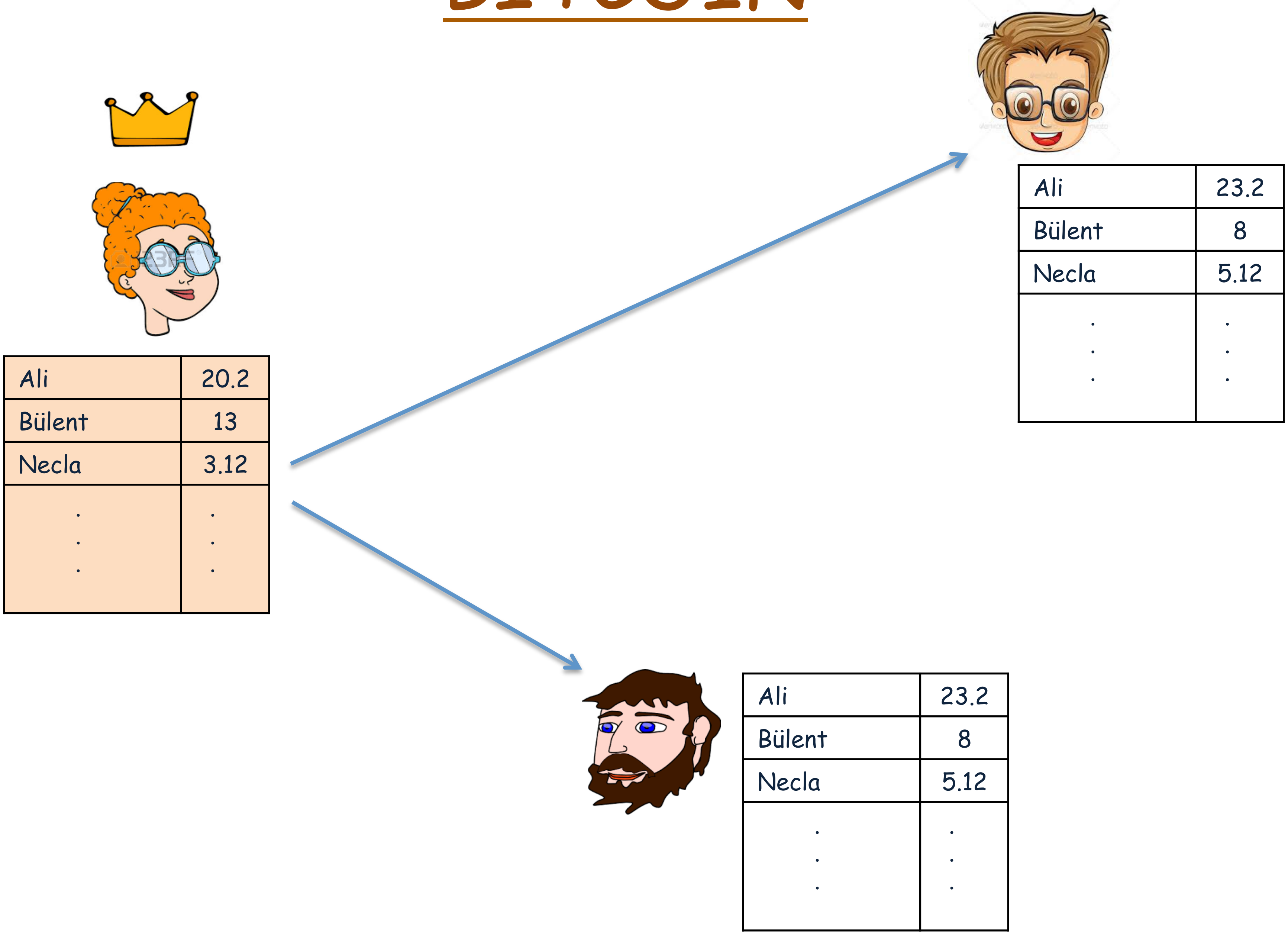
Necla sends 4 bitcoins to Bülent

Bülent sends 2 bitcoin to Necla

Ali	23.2
Bülent	8
Necla	5.12
⋮	⋮

Ali	23.2
Bülent	8
Necla	5.12
⋮	⋮

BITCOIN

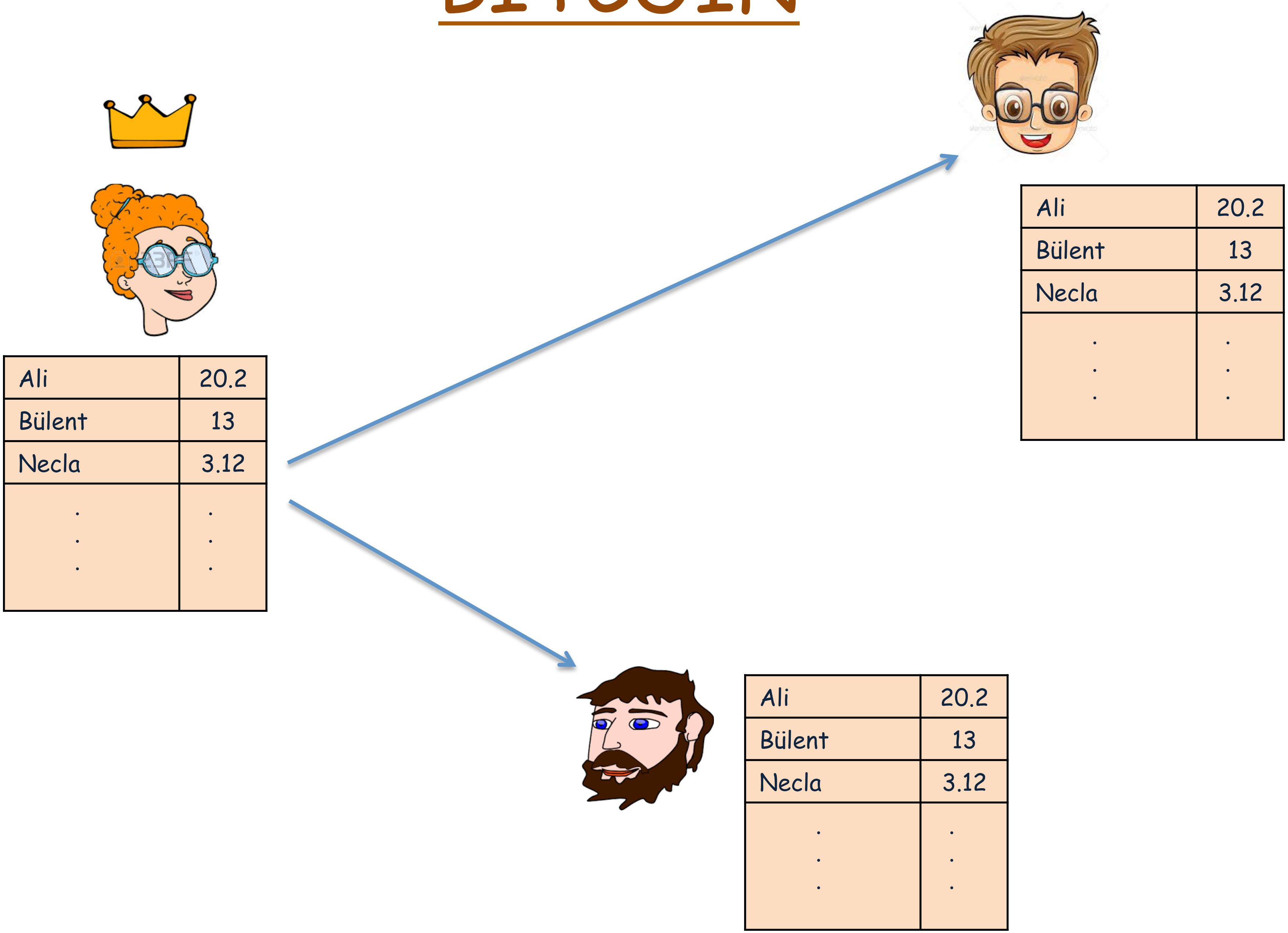


Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

Ali	23.2
Bülent	8
Necla	5.12
⋮	⋮

Ali	23.2
Bülent	8
Necla	5.12
⋮	⋮

BITCOIN

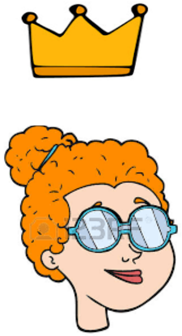


Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .

Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .

Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .

BITCOIN



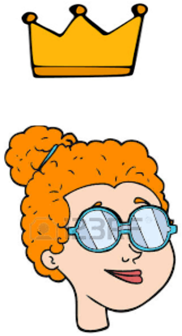
Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

✓ - How to avoid 'forking' ?

BITCOIN



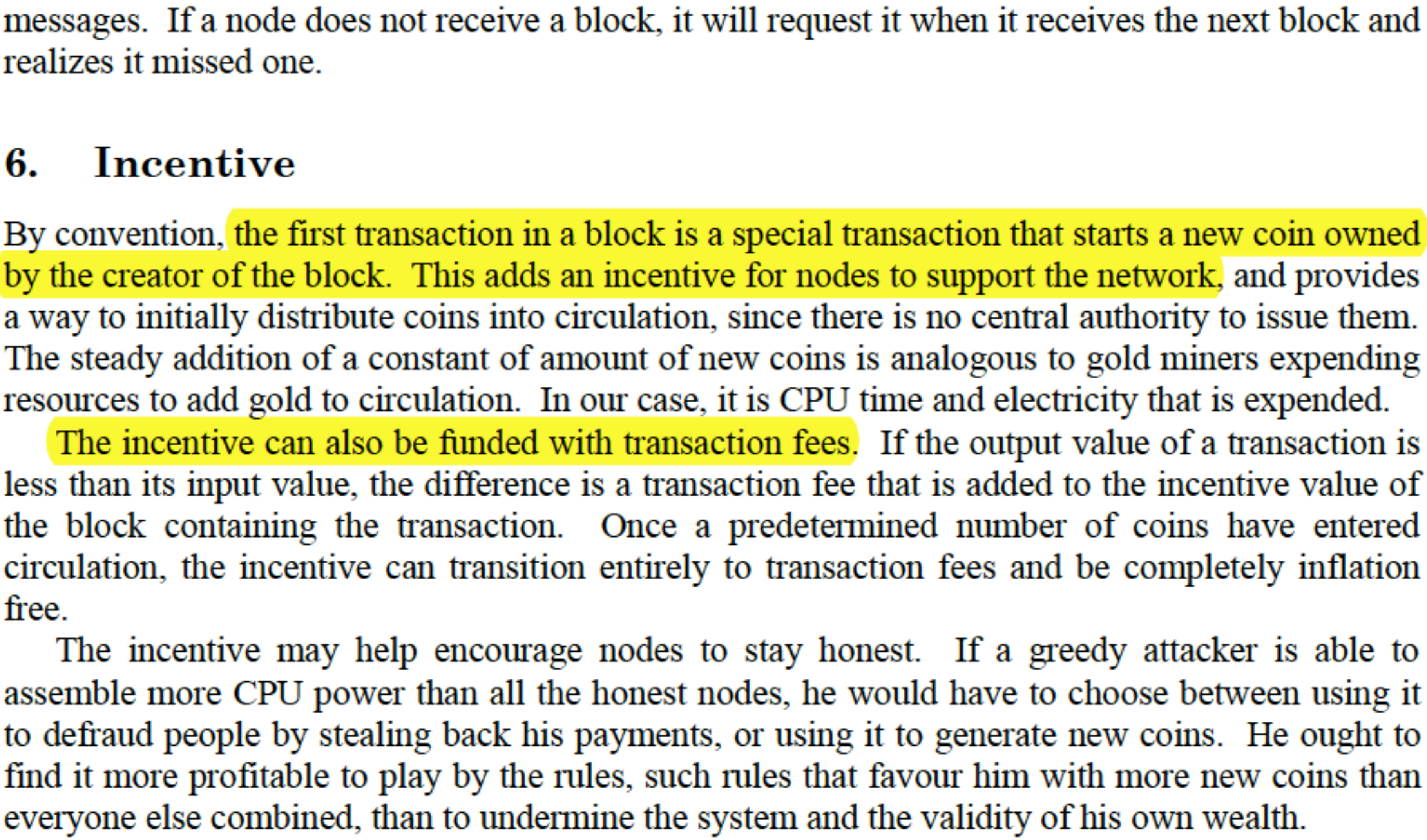
Ali	20.2
Bülent	13
Necla	3.12
.	.
.	.
.	.

Ali	20.2
Bülent	13
Necla	3.12
.	.
.	.
.	.

Ali	20.2
Bülent	13
Necla	3.12
.	.
.	.
.	.

✓
- How to avoid 'forking' ?
- Incentive ?

BITCOIN



messages. If a node does not receive a block, it will request it when it receives the next block and realizes it missed one.

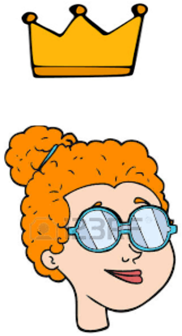
6. Incentive

By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. The steady addition of a constant of amount of new coins is analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that is expended.

The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free.

The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

BITCOIN



Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

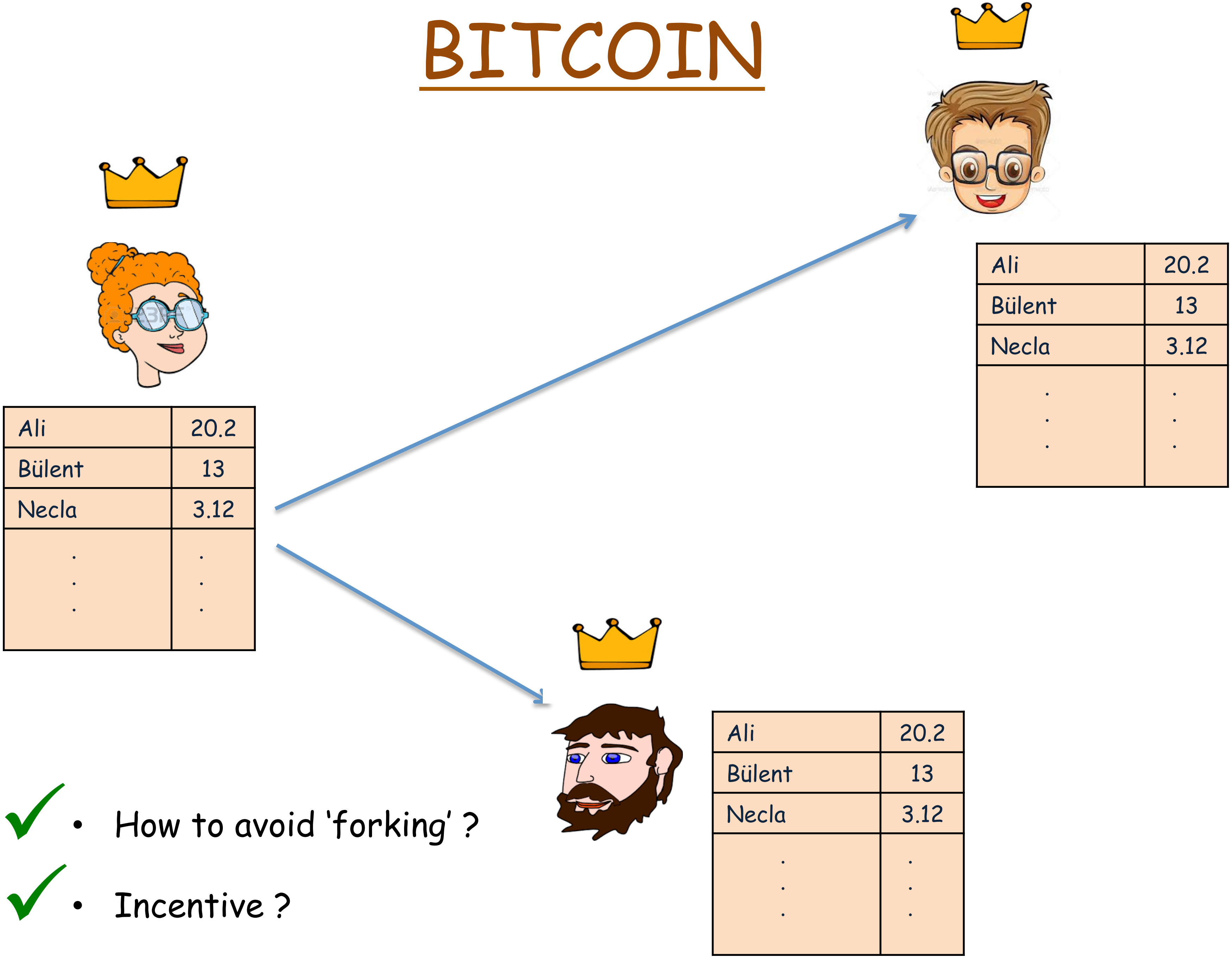
Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

✓ • How to avoid ‘forking’ ?

✓ • Incentive ?

BITCOIN



Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

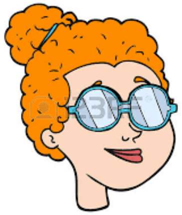
Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

- ✓ How to avoid ‘forking’ ?
- ✓ Incentive ?

BITCOIN

Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .



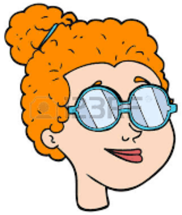
Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .

Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .

BITCOIN

?

Puzzle



Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

?

Puzzle

Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

?

Puzzle

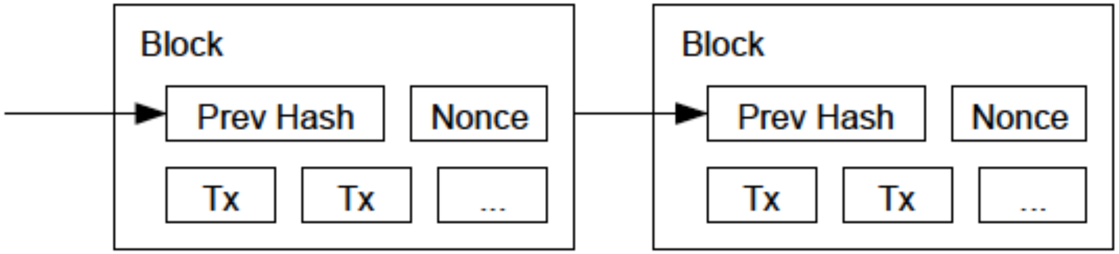
Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

BITCOIN

4. Proof-of-Work

To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof-of-work system similar to Adam Back's Hashcash [6], rather than newspaper or Usenet posts. The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash.

For our timestamp network, we implement the proof-of-work by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits. Once the CPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.



The proof-of-work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-CPU-one-vote. The majority

BITCOIN

- A hash function is any function that can be used to map digital data of arbitrary size to digital data of fixed size(MD5, SHA1, SHA256)

BITCOIN

- A hash function is any function that can be used to map digital data of arbitrary size to digital data of fixed size(MD5, SHA1, SHA256)

- slight differences in input data producing very big differences in output data.

BITCOIN

- A hash function is any function that can be used to map digital data of arbitrary size to digital data of fixed size(MD5, SHA1, SHA256)

- slight differences in input data producing very big differences in output data.

- For example, the MD5 hashes of 'abc' compared to 'abC'

BITCOIN

- A hash function is any function that can be used to map digital data of arbitrary size to digital data of fixed size(MD5, SHA1, SHA256)
- slight differences in input data producing very big differences in output data.
- For example, the MD5 hashes of 'abc' compared to 'abC'

abc

0bee89b07a248e27c83fc3d5951213c1

abC

2217c53a2f88ebadd9b3c1a79cde2638

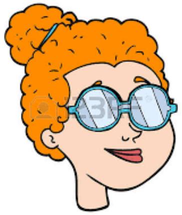
BITCOIN

?

Puzzle

?

Puzzle



HASH

Ali	20.2
Bülent	13
Necla	3.12
.	.
.	.
.	.

Ali	20.2
Bülent	13
Necla	3.12
.	.
.	.
.	.

?

Puzzle

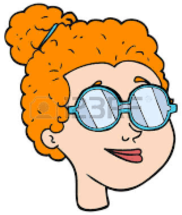
Proof of Work

Ali	20.2
Bülent	13
Necla	3.12
.	.
.	.
.	.

BITCOIN

?

Puzzle



Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .

m_1

m_2

.

.

HASH

?

Puzzle

Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .

?

Puzzle

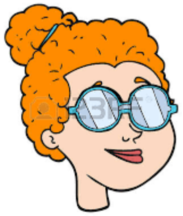
Proof of Work

Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .

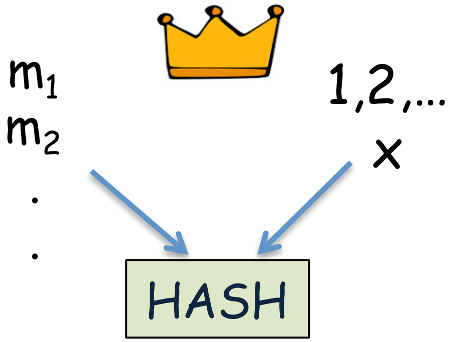
BITCOIN

?

Puzzle



Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮



?

Puzzle

Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

?

Puzzle

Proof of Work

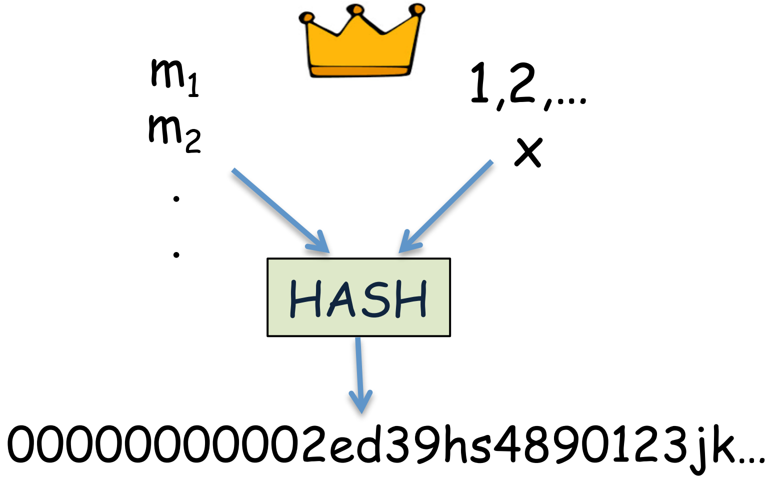
Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

BITCOIN

?

Puzzle

Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .



?

Puzzle

Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .

?

Puzzle

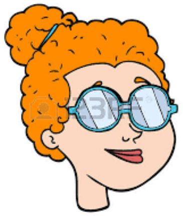
Proof of Work

Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .

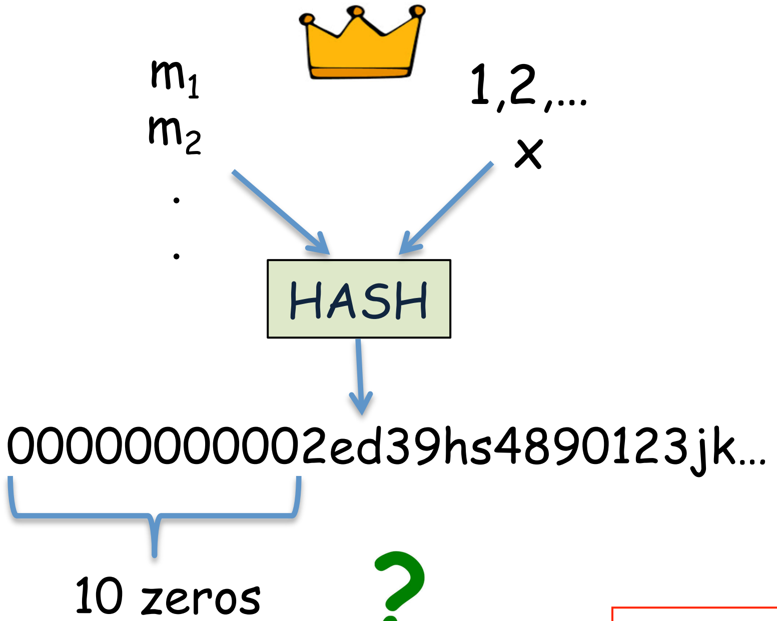
BITCOIN

?

Puzzle



Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮



?

Puzzle

?

Puzzle

Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

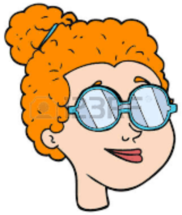
Proof of Work

Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

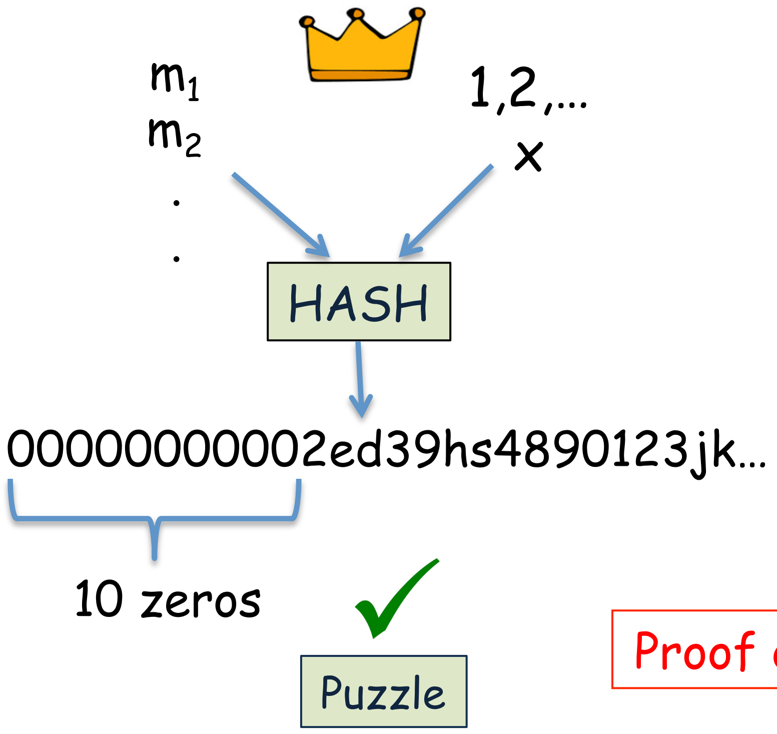
BITCOIN

?

Puzzle



Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮



m_1
m_2
⋮

1,2,...
x

HASH

00000000002ed39hs4890123jk...

10 zeros

✓

Puzzle

?

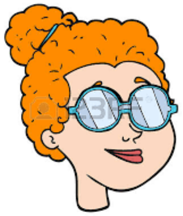
Puzzle

Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

Proof of Work

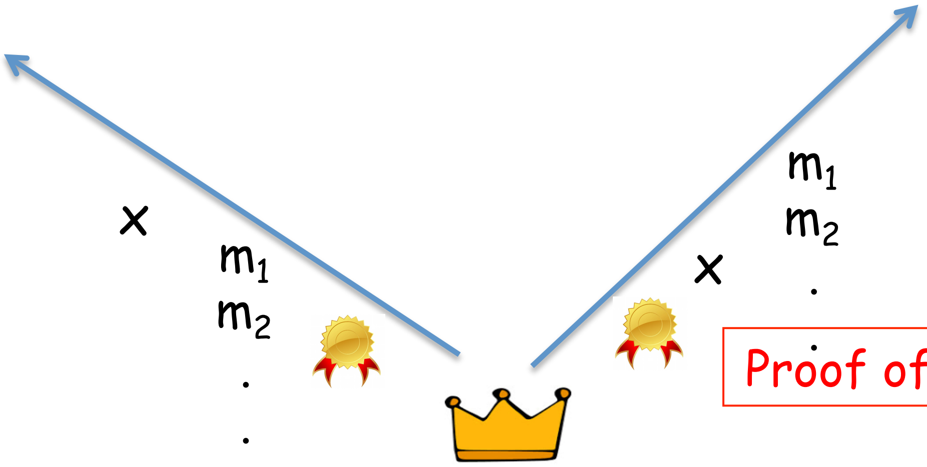
Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

BITCOIN



Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

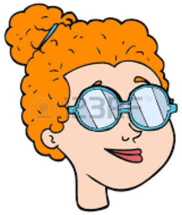


Proof of Work

Ali	20.2
Bülent	13
Necla	3.12
⋮	⋮

BITCOIN

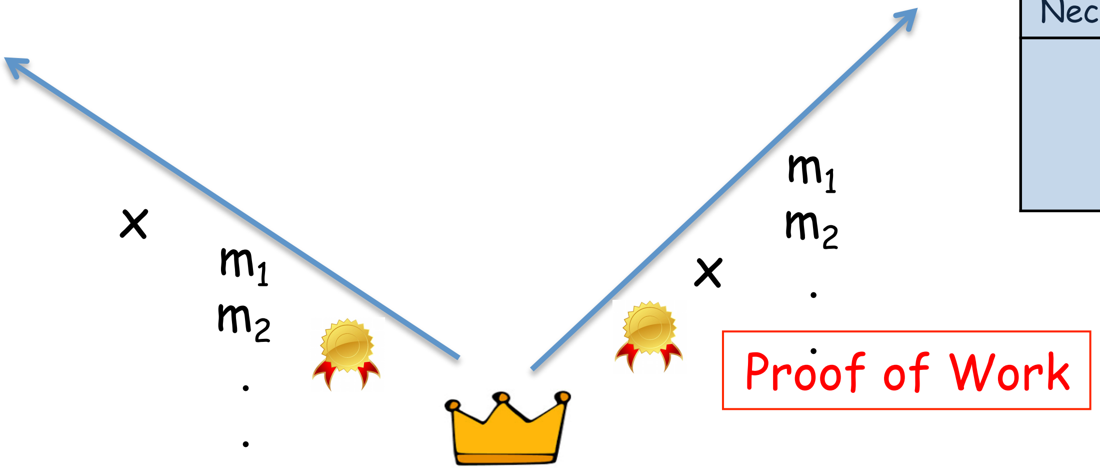
$H(m_1,\ldots \| x) < T$



Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .

$H(m_1,\ldots \| x) < T$

Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .

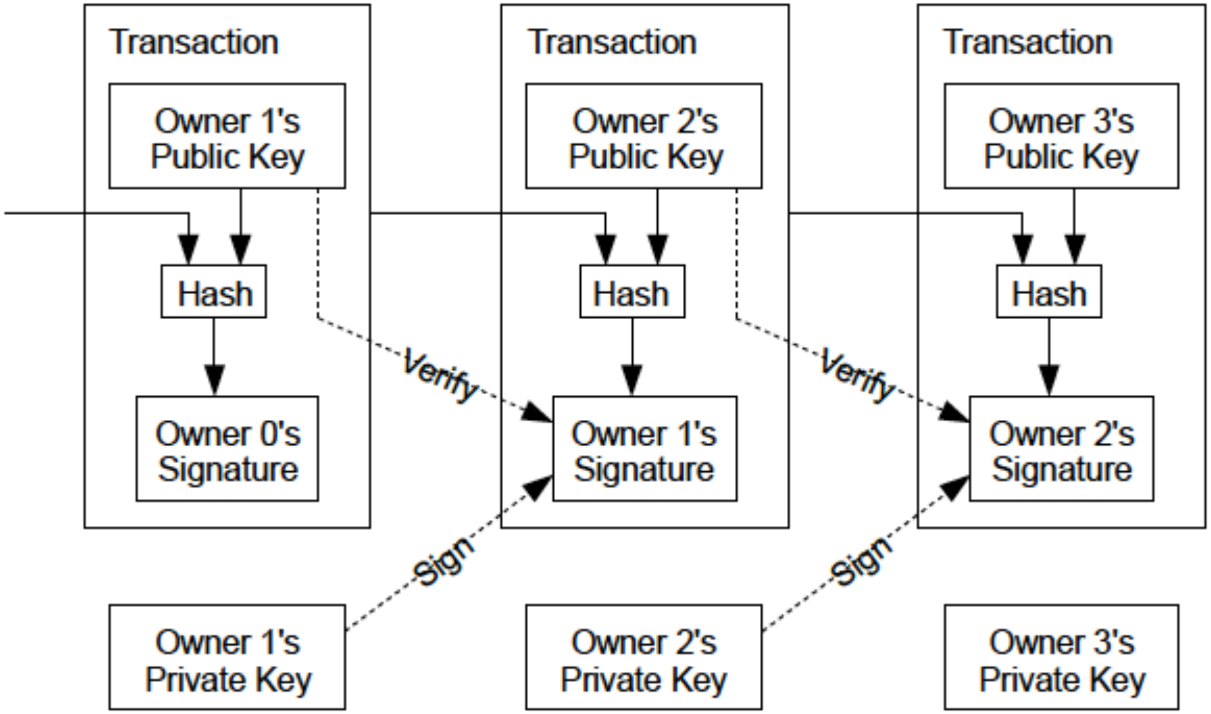


Ali	20.2
Bülent	13
Necla	3.12
. . .	. . .

BITCOIN

2. Transactions

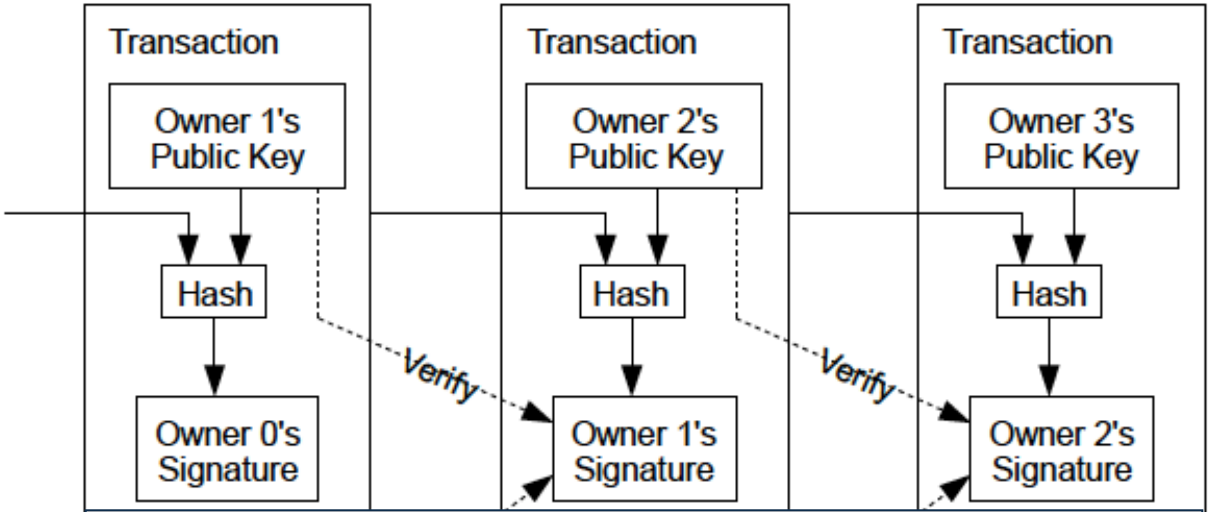
We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.



BITCOIN

2. Transactions

We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.

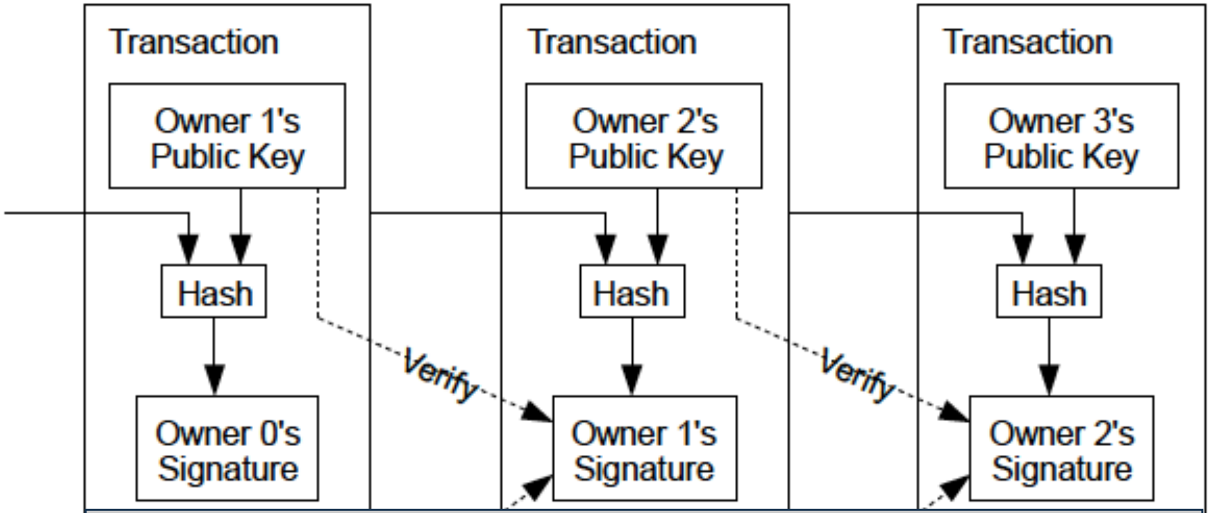


$\text{Sign}_{sk\_1}[\ \text{Hash}(\ \text{prev tx},\ pk_2)\ ]$

BITCOIN

2. Transactions

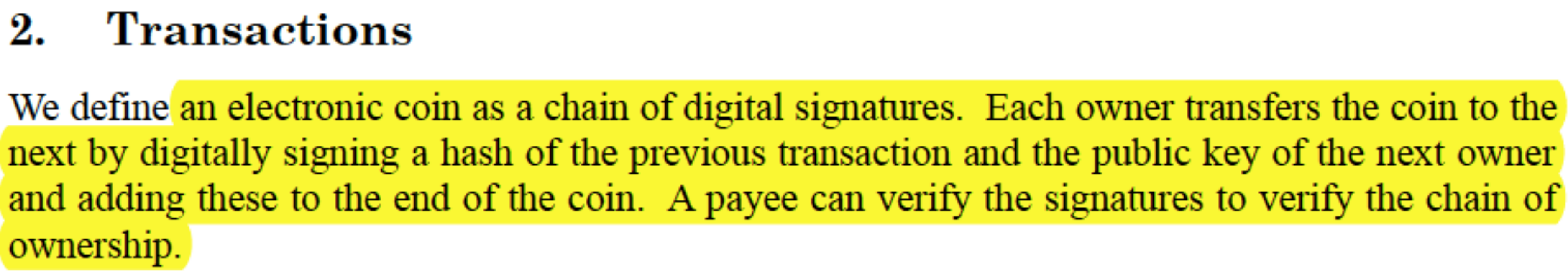
We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.



$$\mathrm{Sign}_{sk\_1}[\ \mathrm{Hash}(\ \mathrm{prev\ tx},\ pk_2)\ ]$$

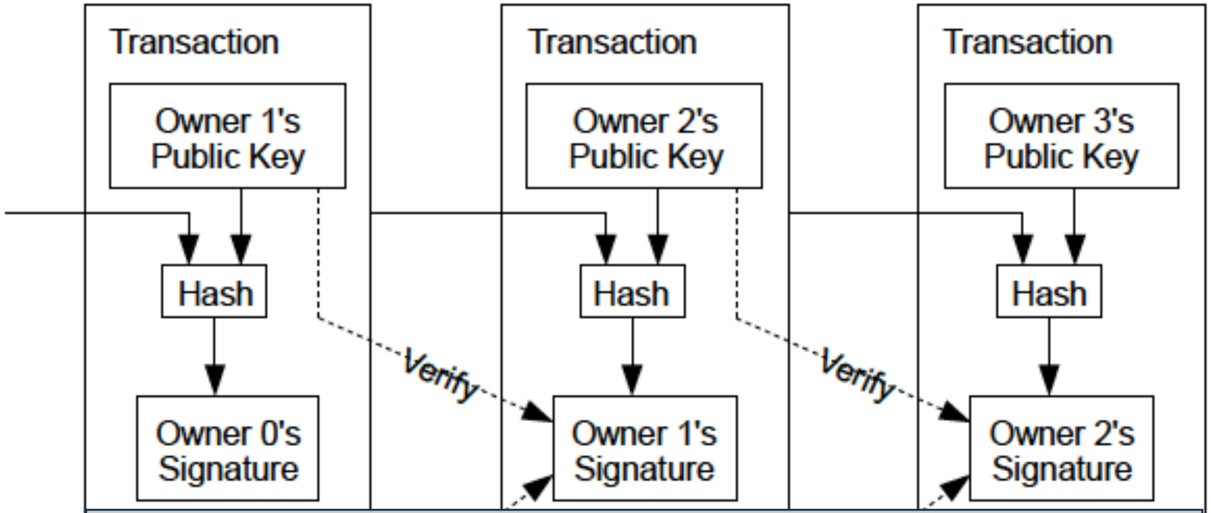
proving the ownership of the coin

BITCOIN



2. Transactions

We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.

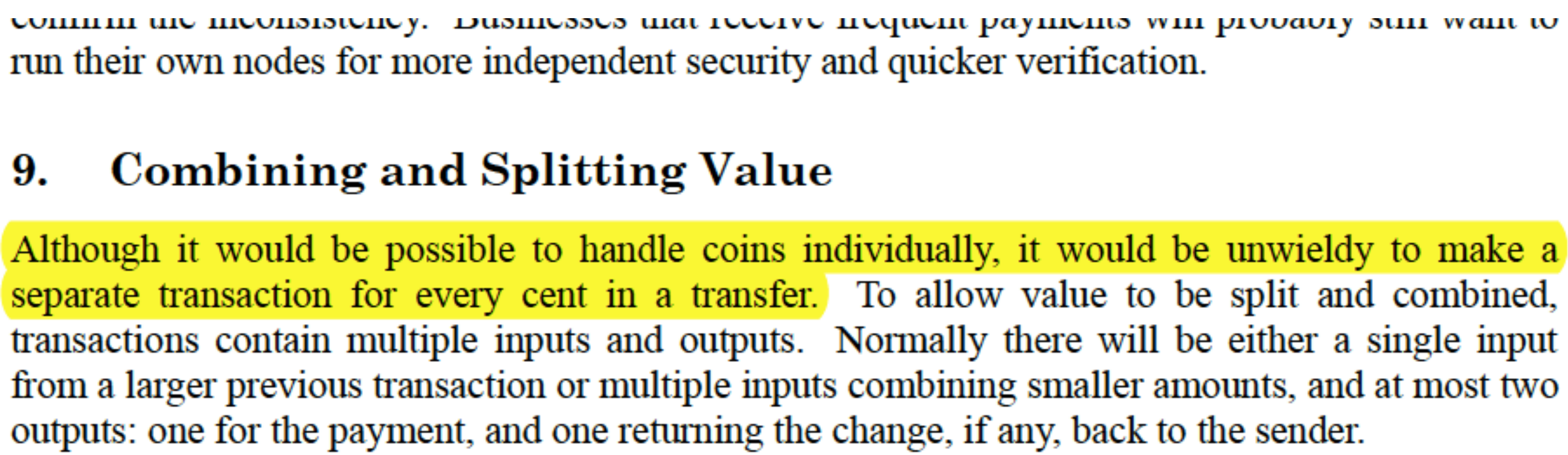


$$\text{Sign}_{sk\_1}[\text{ Hash(prev tx}, pk_2) \text{]}$$

proving the ownership of the coin

indicating the payee

BITCOIN

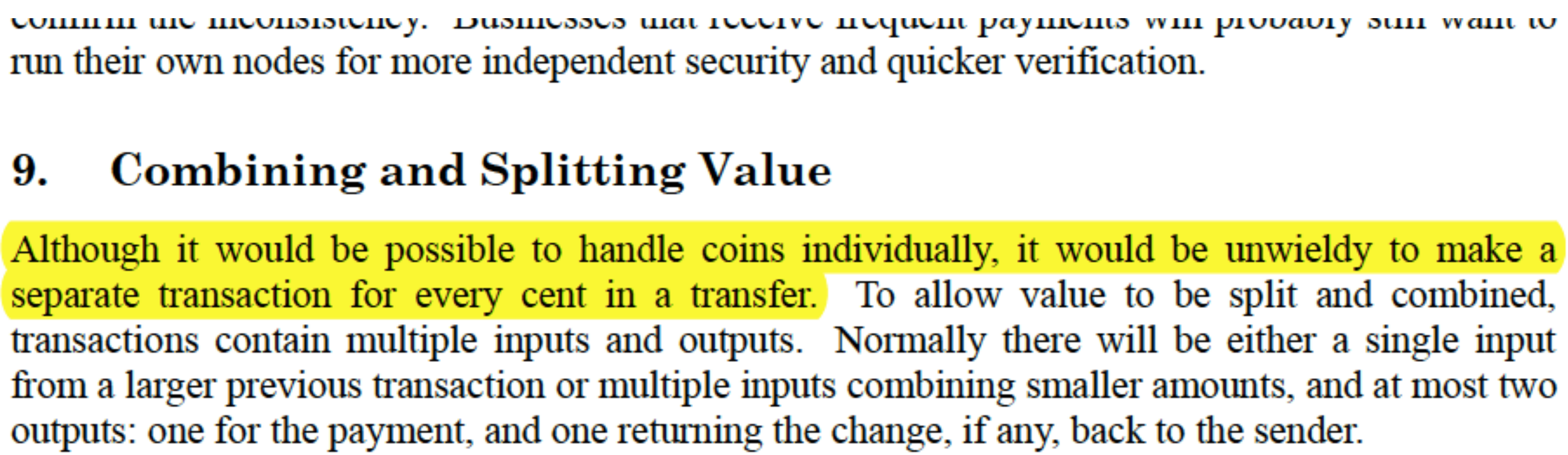


confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

9. Combining and Splitting Value

Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.

BITCOIN



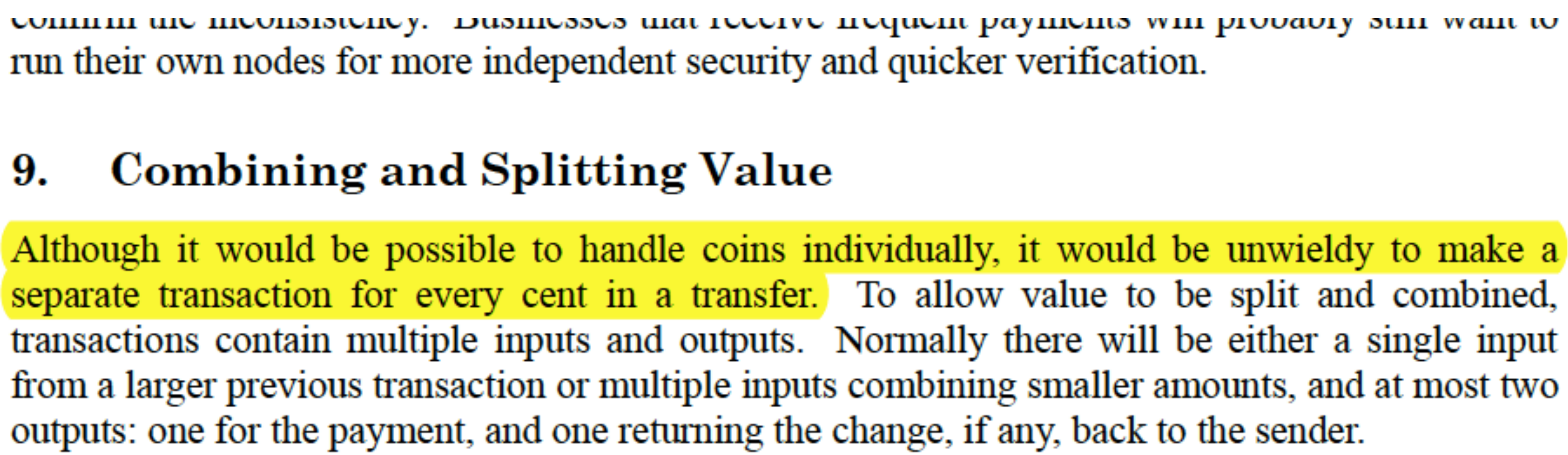
confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

9. Combining and Splitting Value

Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.

- A bitcoin can be divided down to 8 decimal places.

BITCOIN



confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

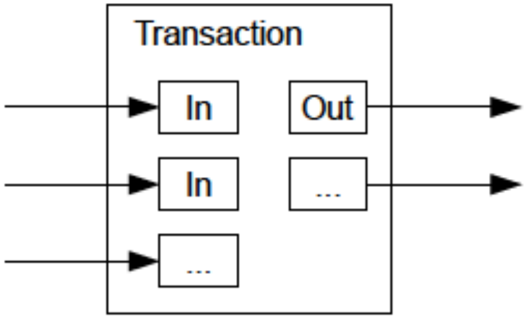
9. Combining and Splitting Value

Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.

- A bitcoin can be divided down to 8 decimal places.
- 0.00000001 BTC (Satoshi) is the smallest amount that can be handled in a transaction

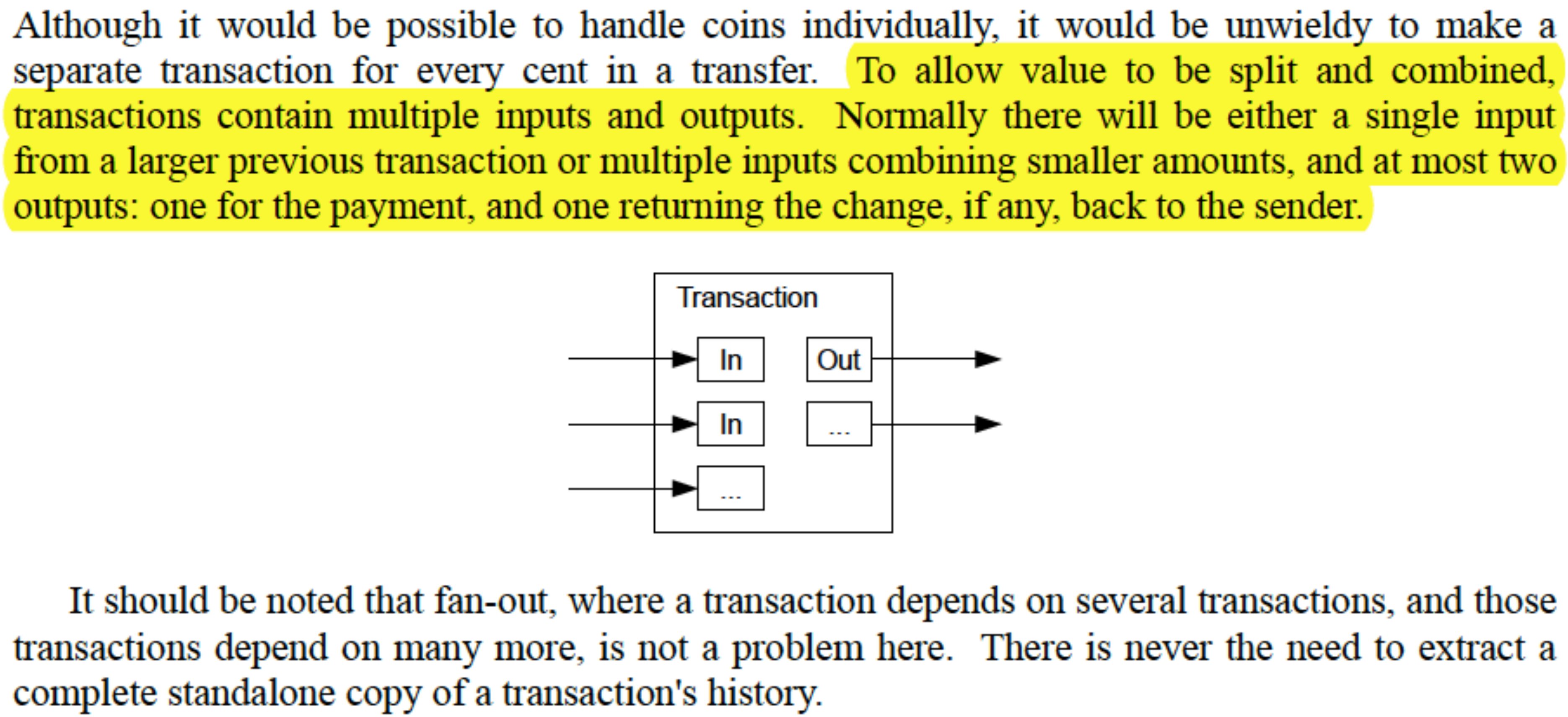
BITCOIN

Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.



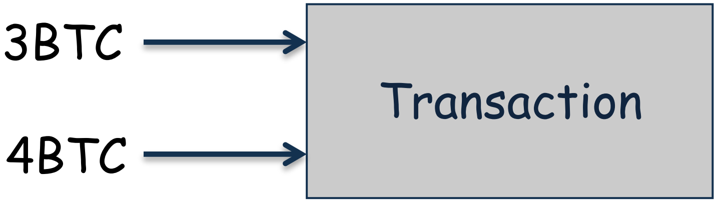
It should be noted that fan-out, where a transaction depends on several transactions, and those transactions depend on many more, is not a problem here. There is never the need to extract a complete standalone copy of a transaction's history.

BITCOIN



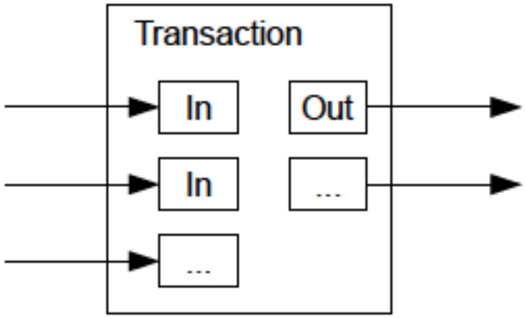
Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.

It should be noted that fan-out, where a transaction depends on several transactions, and those transactions depend on many more, is not a problem here. There is never the need to extract a complete standalone copy of a transaction's history.

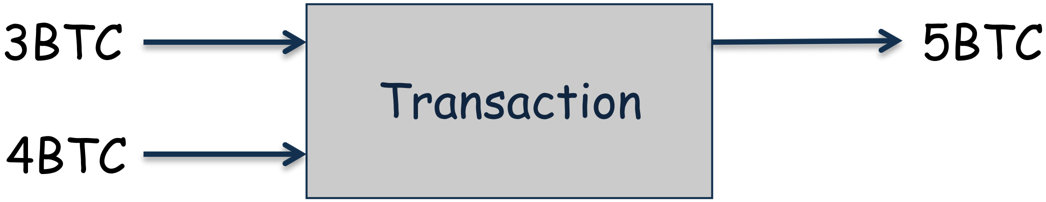


BITCOIN

Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.

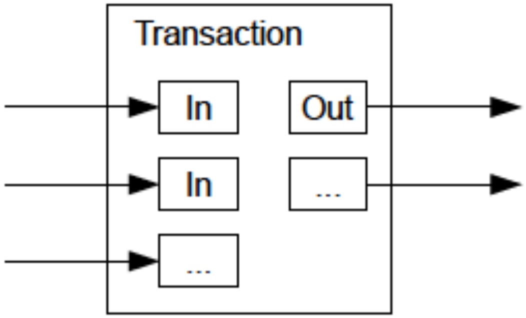


It should be noted that fan-out, where a transaction depends on several transactions, and those transactions depend on many more, is not a problem here. There is never the need to extract a complete standalone copy of a transaction's history.

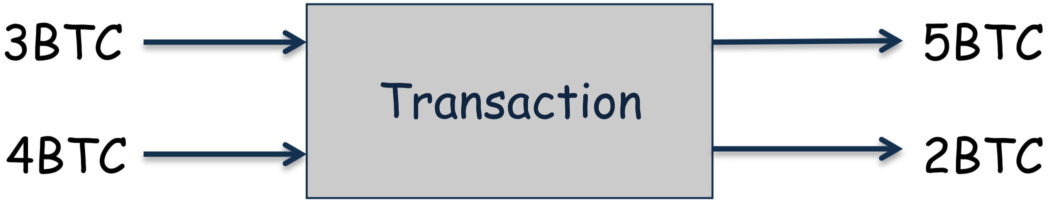


BITCOIN

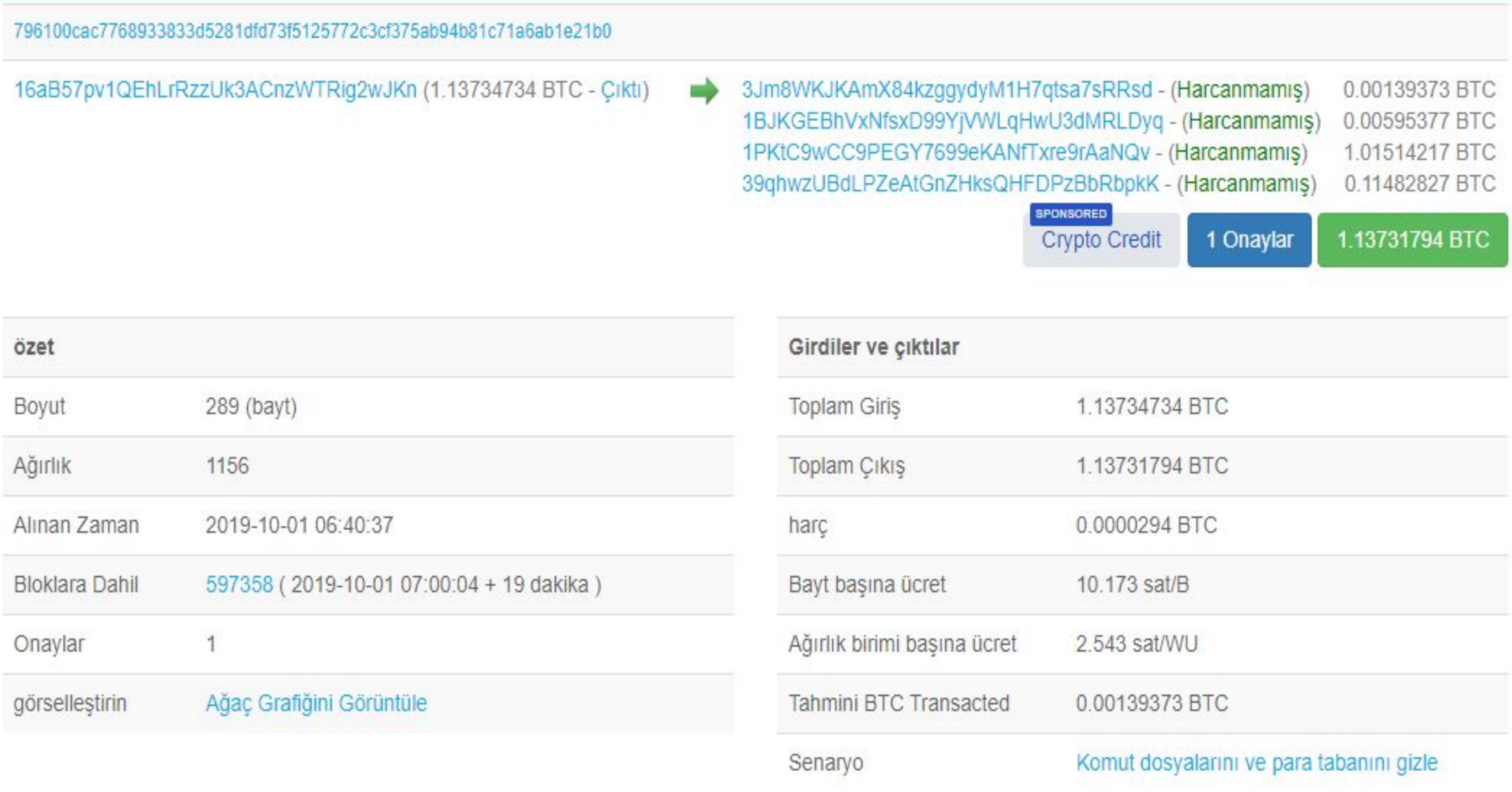
Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.



It should be noted that fan-out, where a transaction depends on several transactions, and those transactions depend on many more, is not a problem here. There is never the need to extract a complete standalone copy of a transaction's history.



BITCOIN



796100cac7768933833d5281dfd73f5125772c3cf375ab94b81c71a6ab1e21b0

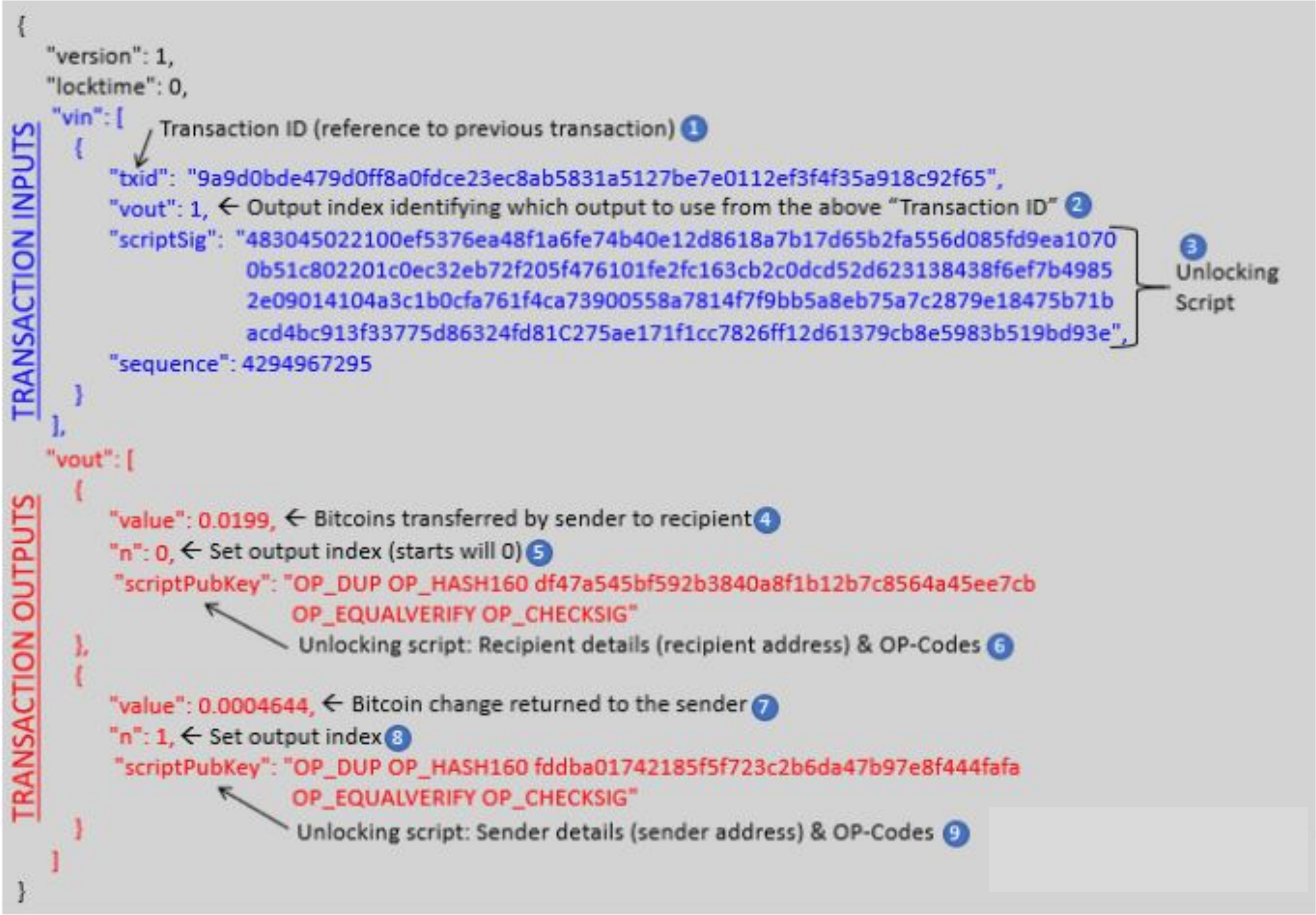
16aB57pv1QEhLrRzzUk3ACnzWTRig2wJKn (1.13734734 BTC - Çıktı) →

3Jm8WKJKAmX84kzggydyM1H7qtsa7sRRsd - (Harcanmamış) 0.00139373 BTC
1BJKGEBhVxNfsxD99YjVWLqHwU3dMRLDyq - (Harcanmamış) 0.00595377 BTC
1PKtC9wCC9PEGY7699eKANfTxre9rAaNQv - (Harcanmamış) 1.01514217 BTC
39qhwzUBdLPZeAtGnZHksQHFDPzBbRbpkK - (Harcanmamış) 0.11482827 BTC

özet	
Boyut	289 (bayt)
Ağırlık	1156
Alınan Zaman	2019-10-01 06:40:37
Bloklara Dahil	597358 (2019-10-01 07:00:04 + 19 dakika)
Onaylar	1
görselleştirin	Ağaç Grafiğini Görüntüle

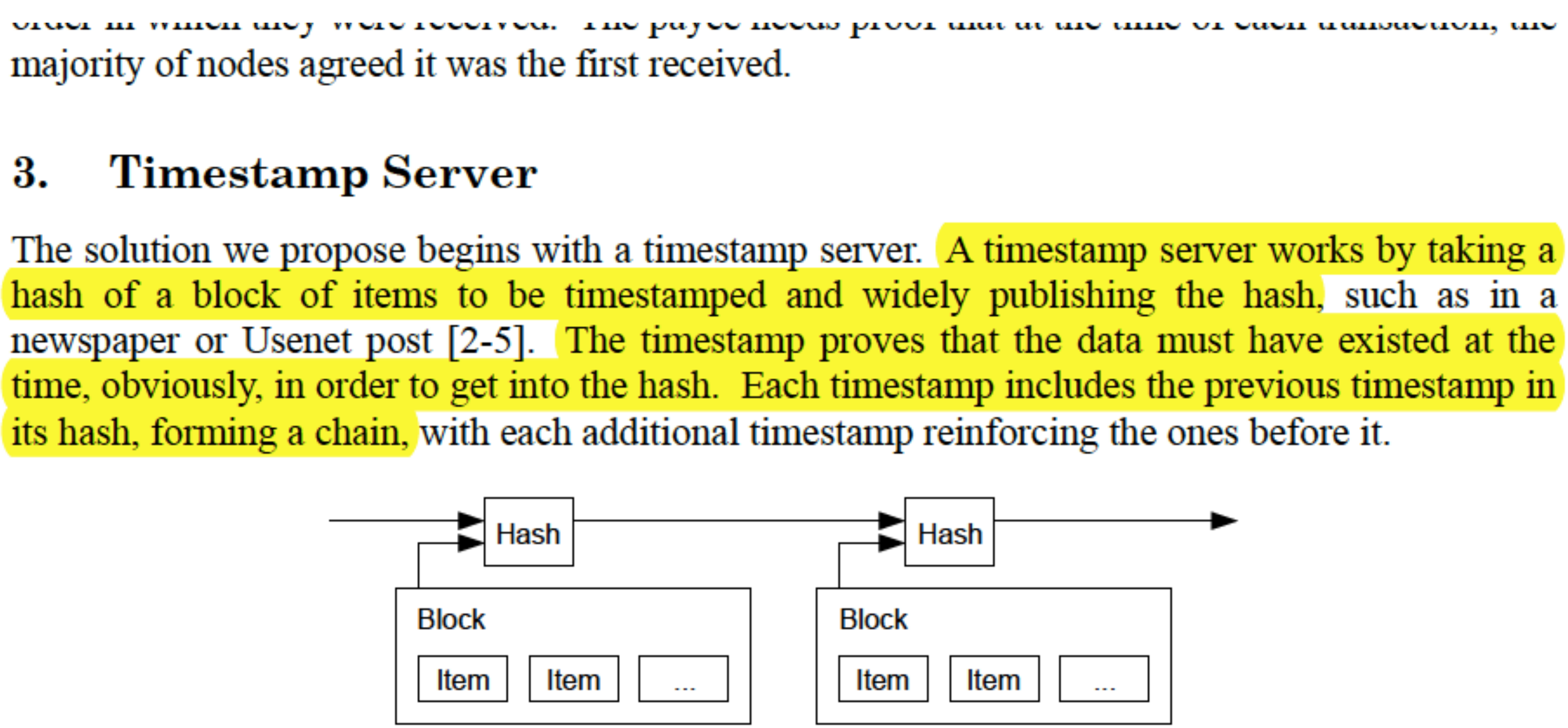
Girdiler ve çıktılar	
Toplam Giriş	1.13734734 BTC
Toplam Çıkış	1.13731794 BTC
harç	0.0000294 BTC
Bayt başına ücret	10.173 sat/B
Ağırlık birimi başına ücret	2.543 sat/WU
Tahmini BTC Transacted	0.00139373 BTC
Senaryo	Komut dosyalarını ve para tabanını gizle

BITCOIN



```
{
  "version": 1,
  "locktime": 0,
  "vin": [        Transaction ID (reference to previous transaction) 1
    {
      "txid": "9a9d0bde479d0ff8a0fdce23ec8ab5831a5127be7e0112ef3f4f35a918c92f65",
      "vout": 1, ← Output index identifying which output to use from the above "Transaction ID" 2
      "scriptSig": "483045022100ef5376ea48f1a6fe74b40e12d8618a7b17d65b2fa556d085fd9ea1070
                    0b51c802201c0ec32eb72f205f476101fe2fc163cb2c0dcd52d623138438f6ef7b4985
                    2e09014104a3c1b0cfa761f4ca73900558a7814f7f9bb5a8eb75a7c2879e18475b71b
                    acd4bc913f33775d86324fd81C275ae171f1cc7826ff12d61379cb8e5983b519bd93e",   3 Unlocking Script
      "sequence": 4294967295
    }
  ],
  "vout": [
    {
      "value": 0.0199, ← Bitcoins transferred by sender to recipient 4
      "n": 0, ← Set output index (starts will 0) 5
      "scriptPubKey": "OP_DUP OP_HASH160 df47a545bf592b3840a8f1b12b7c8564a45ee7cb
                       OP_EQUALVERIFY OP_CHECKSIG"
                       Unlocking script: Recipient details (recipient address) & OP-Codes 6
    },
    {
      "value": 0.0004644, ← Bitcoin change returned to the sender 7
      "n": 1, ← Set output index 8
      "scriptPubKey": "OP_DUP OP_HASH160 fddba01742185f5f723c2b6da47b97e8f444fafa
                       OP_EQUALVERIFY OP_CHECKSIG"
                       Unlocking script: Sender details (sender address) & OP-Codes 9
    }
  ]
}
```

BITCOIN

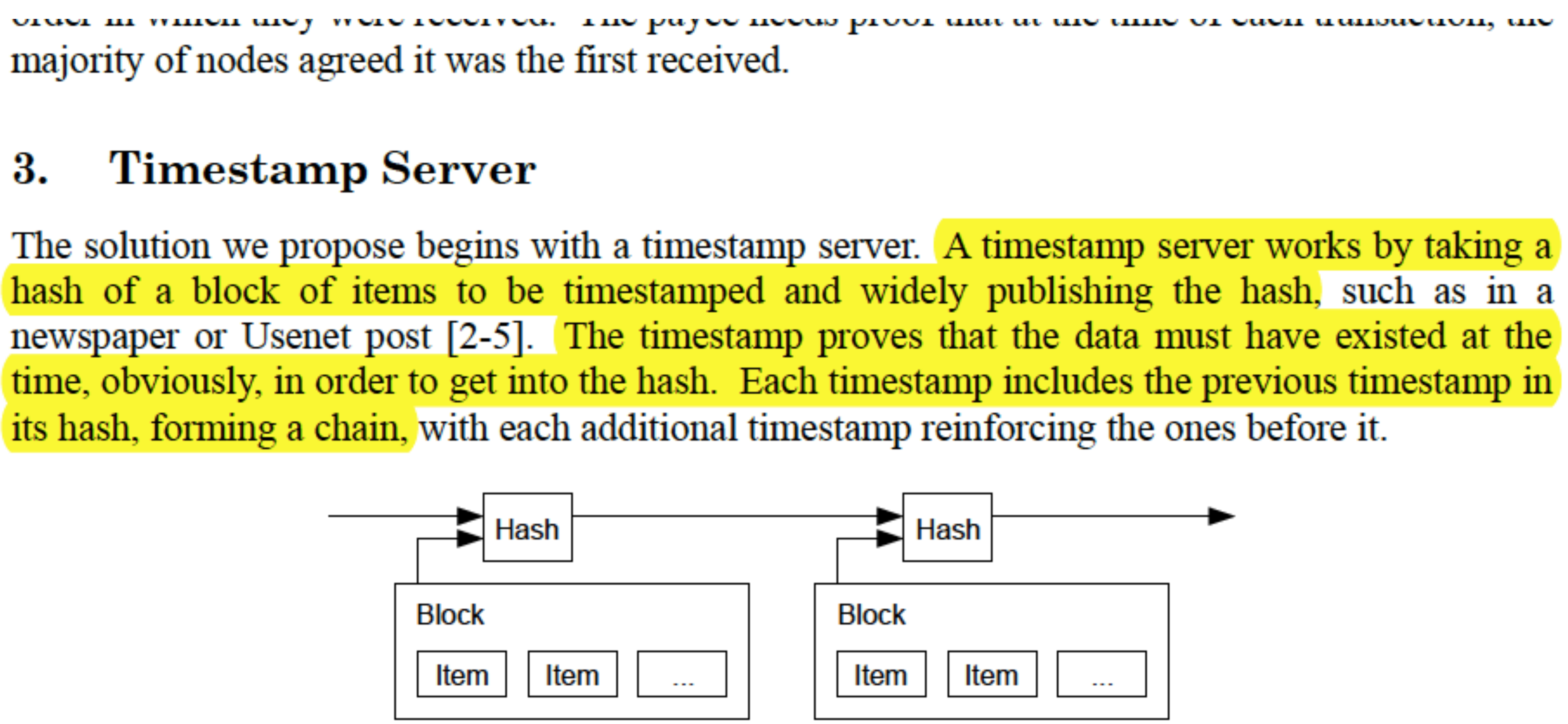


order in which they were received. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

3. Timestamp Server

The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5]. The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.

BITCOIN



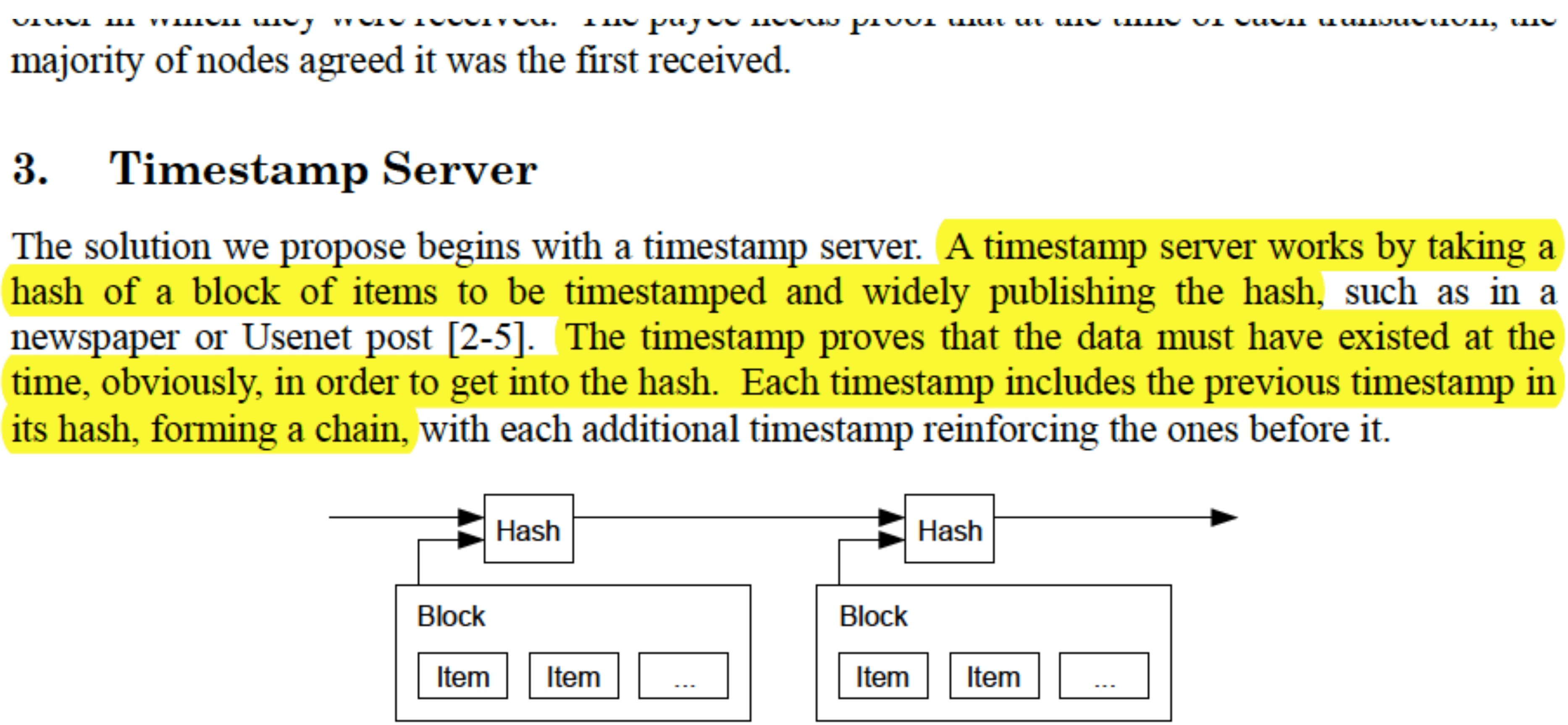
order in which they were received. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

3. Timestamp Server

The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5]. The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.

- SHA-256 is used as hash function, which is collision-resistant

BITCOIN



order in which they were received. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

3. Timestamp Server

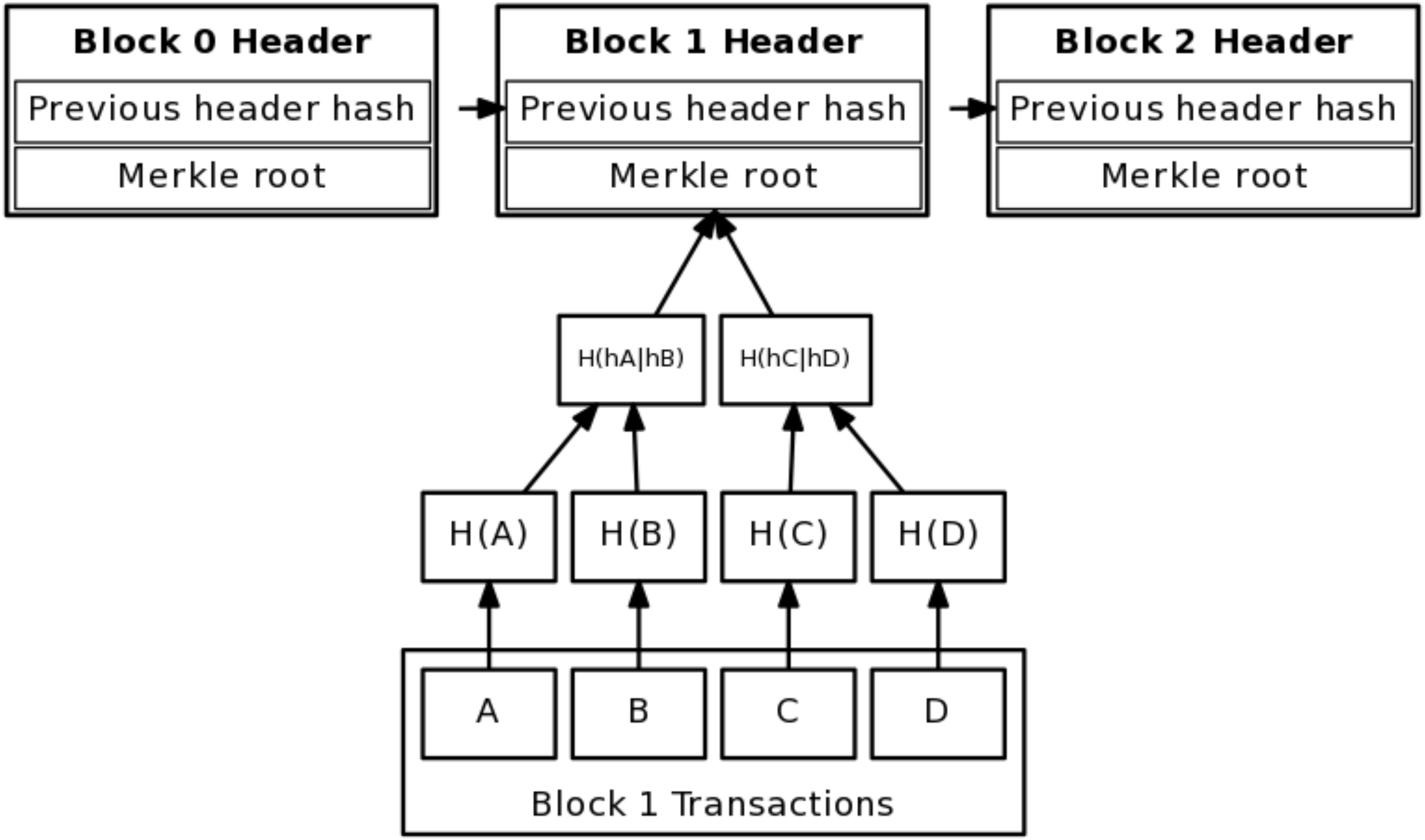
The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5]. The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.

- SHA-256 is used as hash function, which is collision-resistant

- It's hard to find x and y such that H(x) = H(y)

BITCOIN

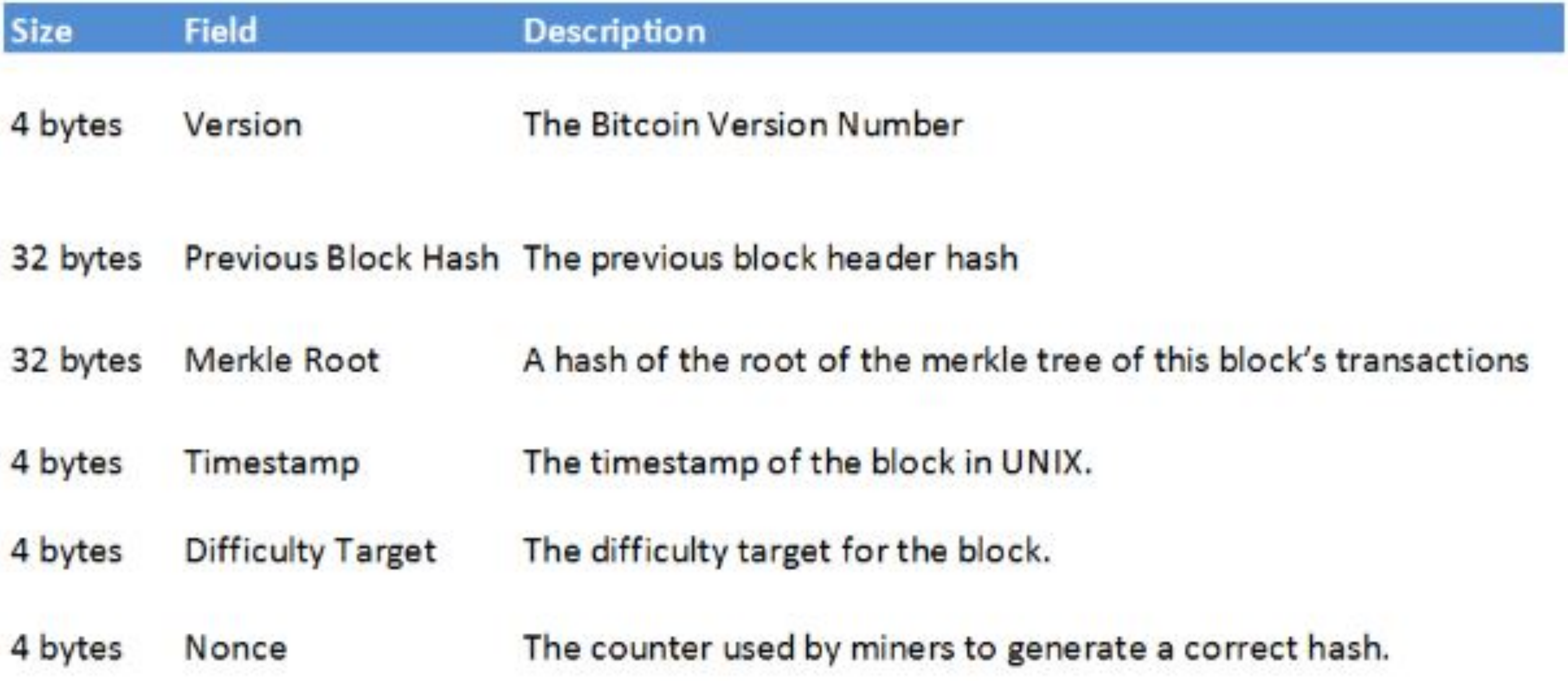
- Full Node vs Lightweight Node



Merkle tree connecting block transactions to block header merkle root

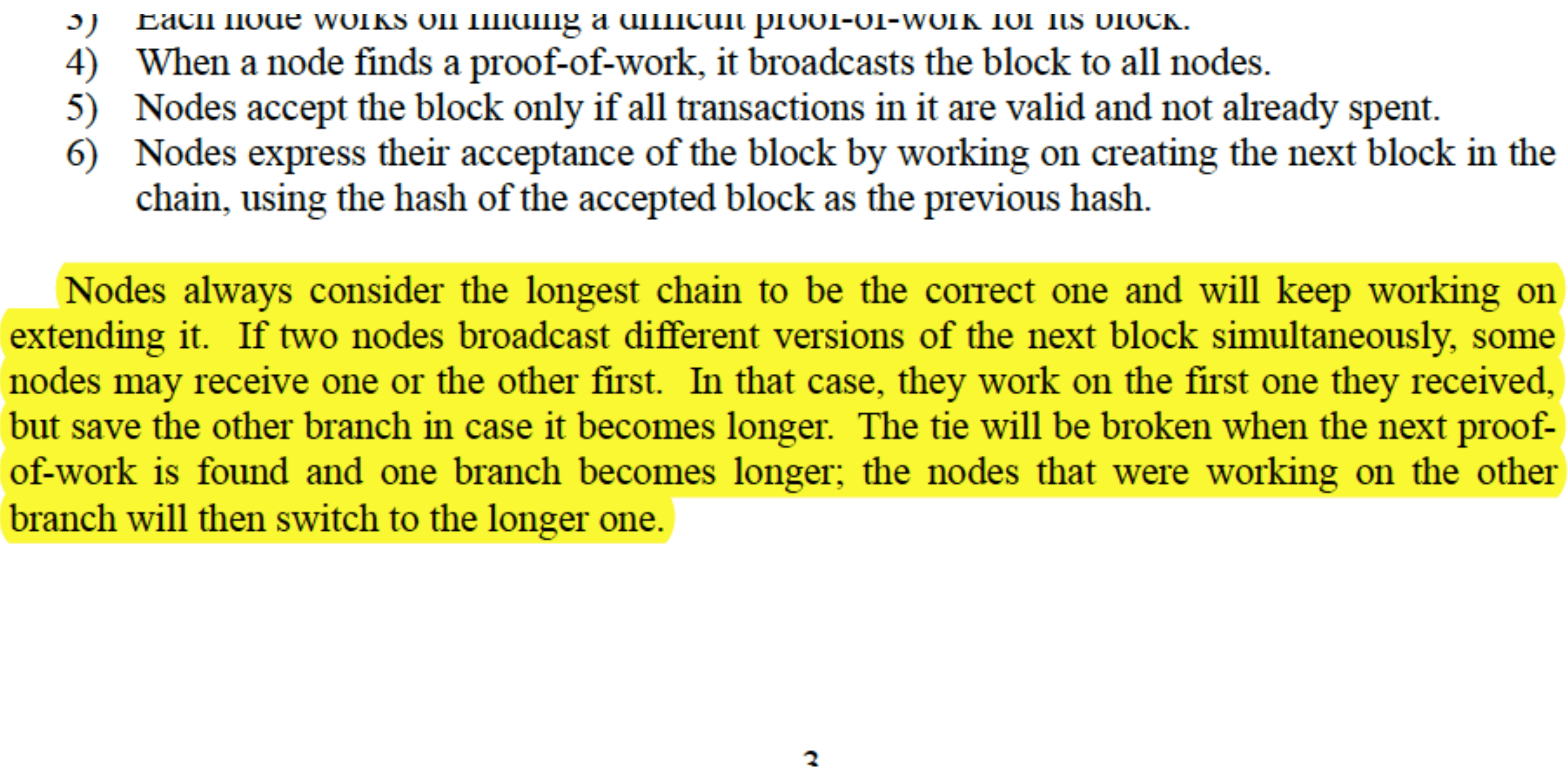
BITCOIN

- Block Header



Size	Field	Description
4 bytes	Version	The Bitcoin Version Number
32 bytes	Previous Block Hash	The previous block header hash
32 bytes	Merkle Root	A hash of the root of the merkle tree of this block’s transactions
4 bytes	Timestamp	The timestamp of the block in UNIX.
4 bytes	Difficulty Target	The difficulty target for the block.
4 bytes	Nonce	The counter used by miners to generate a correct hash.

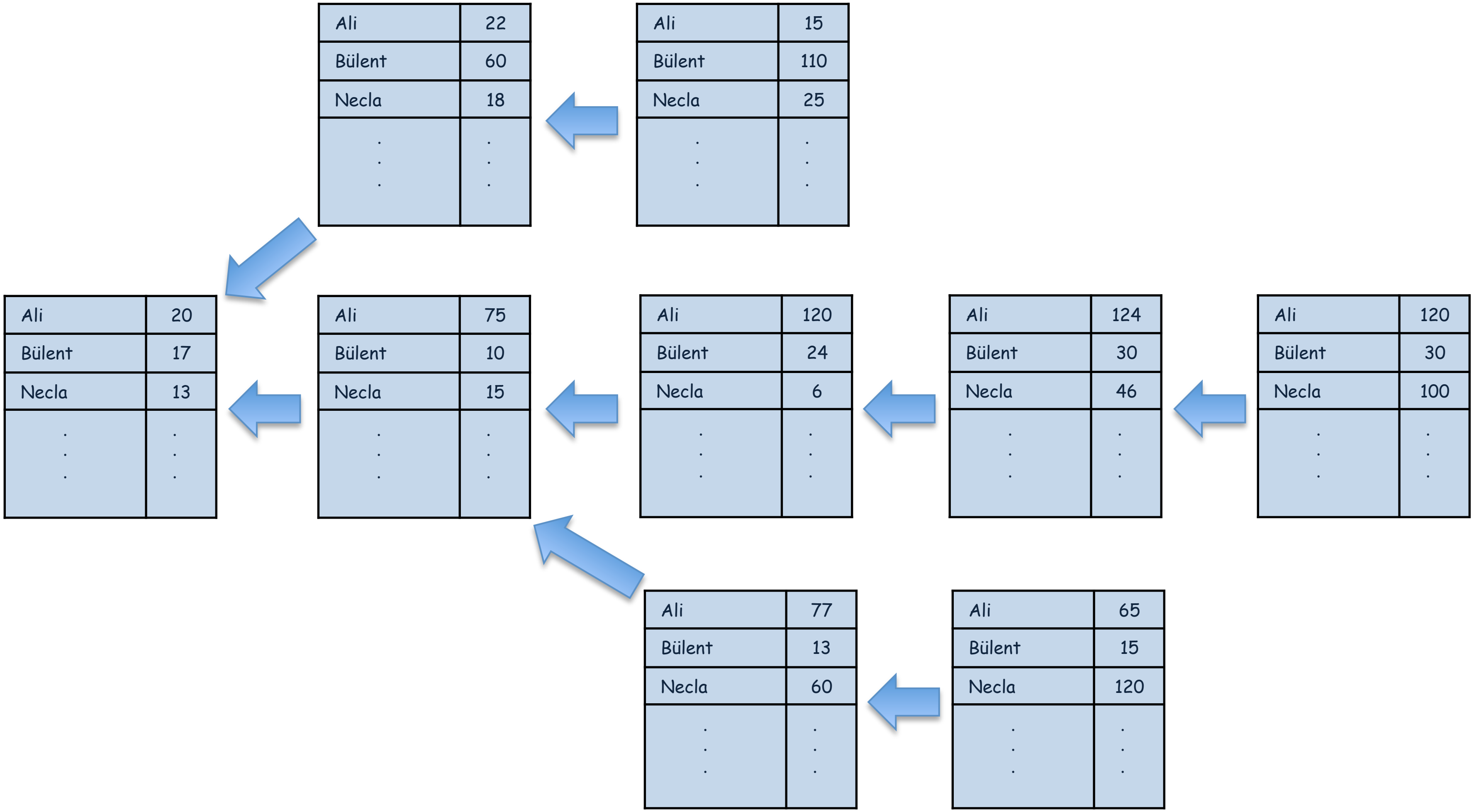
BITCOIN



3) Each node works on finding a difficult proof-of-work for its block.
4) When a node finds a proof-of-work, it broadcasts the block to all nodes.
5) Nodes accept the block only if all transactions in it are valid and not already spent.
6) Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Nodes always consider the longest chain to be the correct one and will keep working on extending it. If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received, but save the other branch in case it becomes longer. The tie will be broken when the next proof-of-work is found and one branch becomes longer; the nodes that were working on the other branch will then switch to the longer one.

BITCOIN



Ali	22
Bülent	60
Necla	18
⋮	⋮

Ali	15
Bülent	110
Necla	25
⋮	⋮

Ali	20
Bülent	17
Necla	13
⋮	⋮

Ali	75
Bülent	10
Necla	15
⋮	⋮

Ali	120
Bülent	24
Necla	6
⋮	⋮

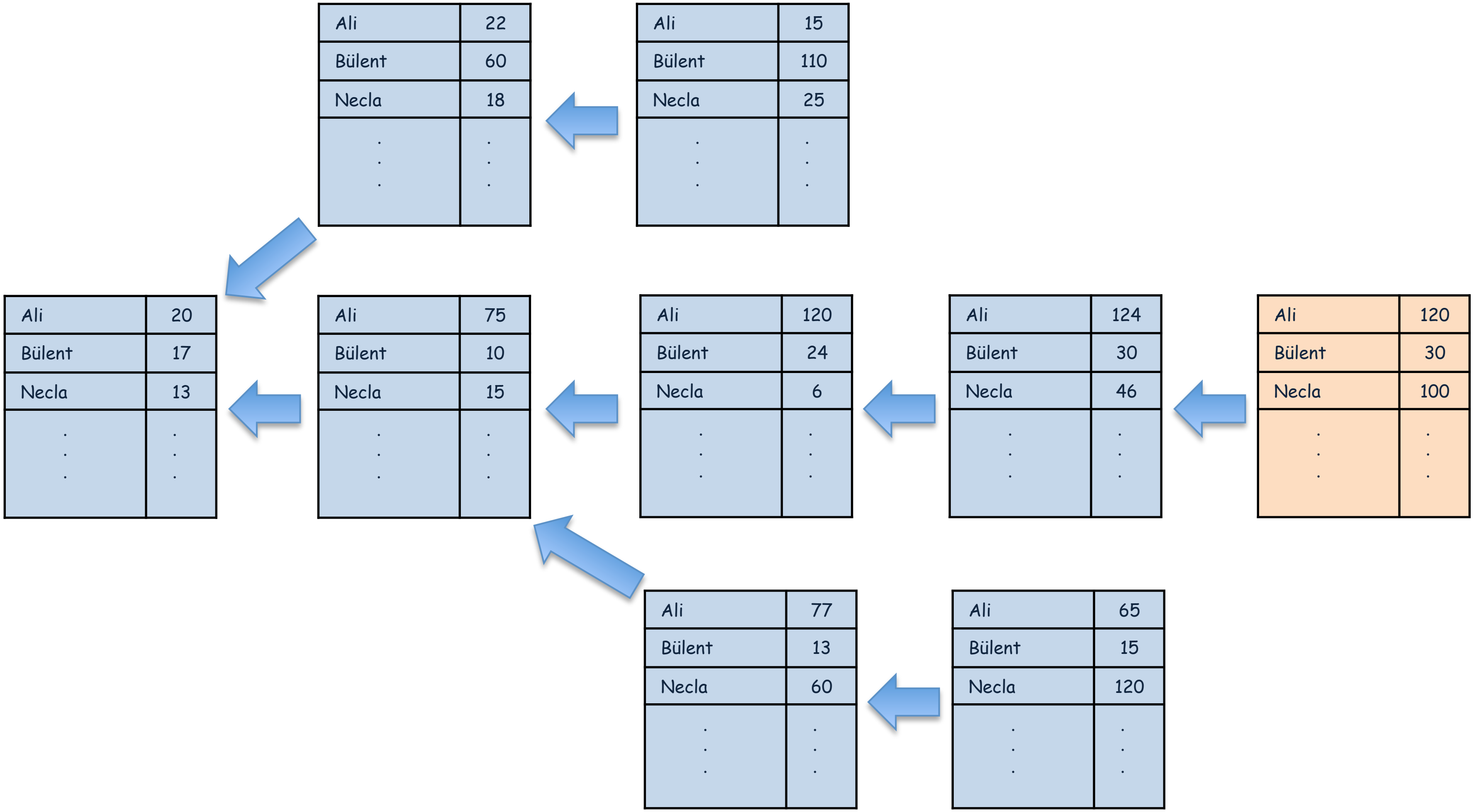
Ali	124
Bülent	30
Necla	46
⋮	⋮

Ali	120
Bülent	30
Necla	100
⋮	⋮

Ali	77
Bülent	13
Necla	60
⋮	⋮

Ali	65
Bülent	15
Necla	120
⋮	⋮

BITCOIN



Ali	22
Bülent	60
Necla	18
⋮	⋮

Ali	15
Bülent	110
Necla	25
⋮	⋮

Ali	20
Bülent	17
Necla	13
⋮	⋮

Ali	75
Bülent	10
Necla	15
⋮	⋮

Ali	120
Bülent	24
Necla	6
⋮	⋮

Ali	124
Bülent	30
Necla	46
⋮	⋮

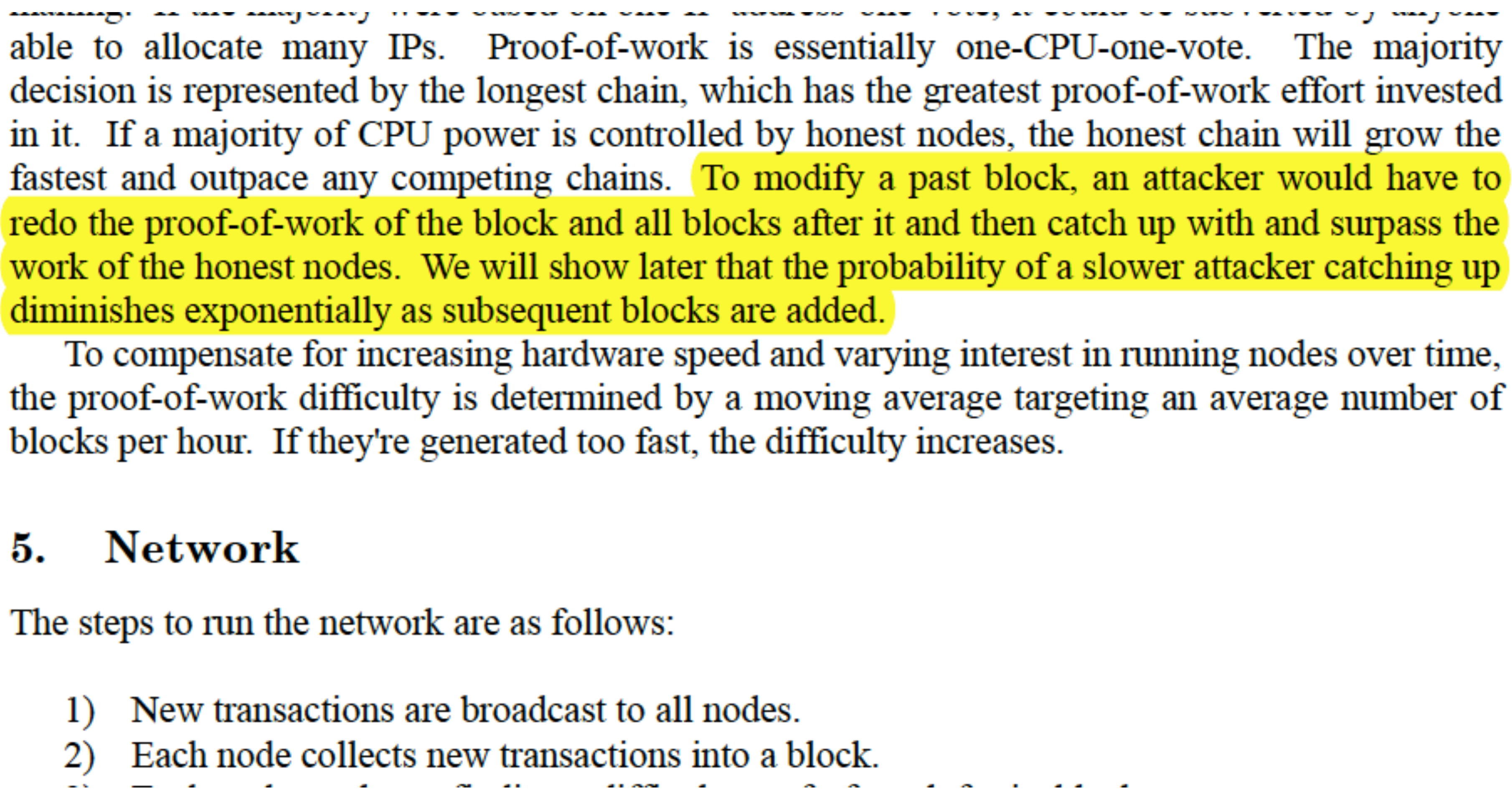
Ali	120
Bülent	30
Necla	100
⋮	⋮

Ali	77
Bülent	13
Necla	60
⋮	⋮

Ali	65
Bülent	15
Necla	120
⋮	⋮

BITCOIN

Double-spending



making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-CPU-one-vote. The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes. We will show later that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.

To compensate for increasing hardware speed and varying interest in running nodes over time, the proof-of-work difficulty is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases.

5. Network

The steps to run the network are as follows:

1) New transactions are broadcast to all nodes.
2) Each node collects new transactions into a block.
3) Each node works on finding a difficult proof-of-work for its block.

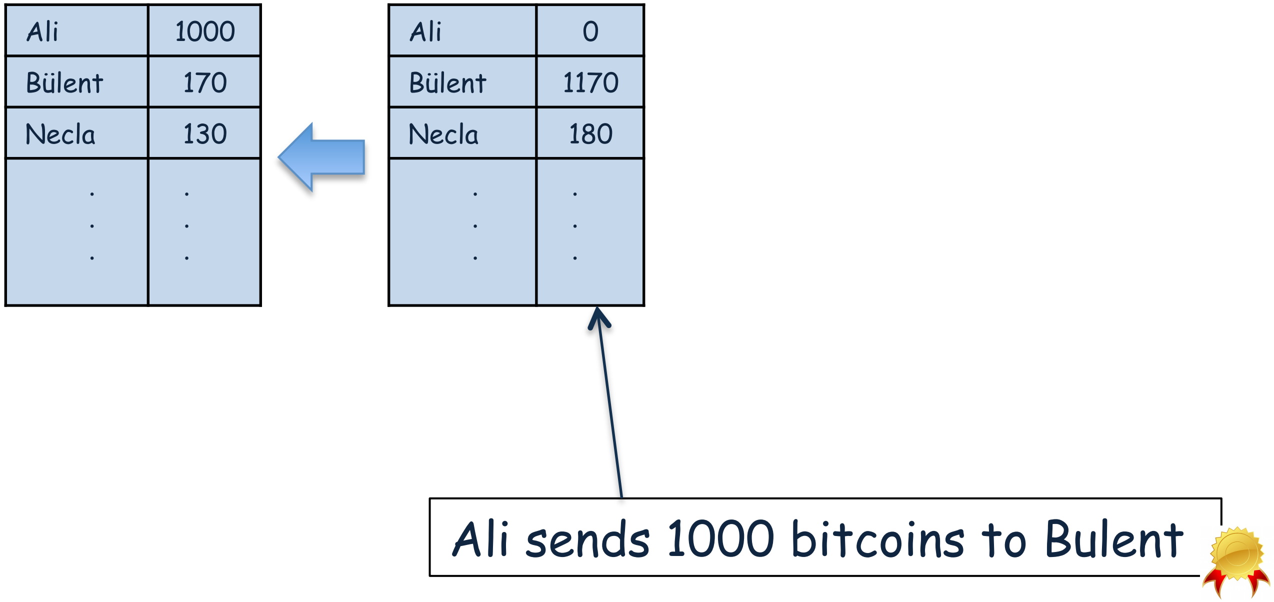
BITCOIN

Double-spending

Ali	1000
Bülent	170
Necla	130
. . .	. . .

BITCOIN

Double-spending



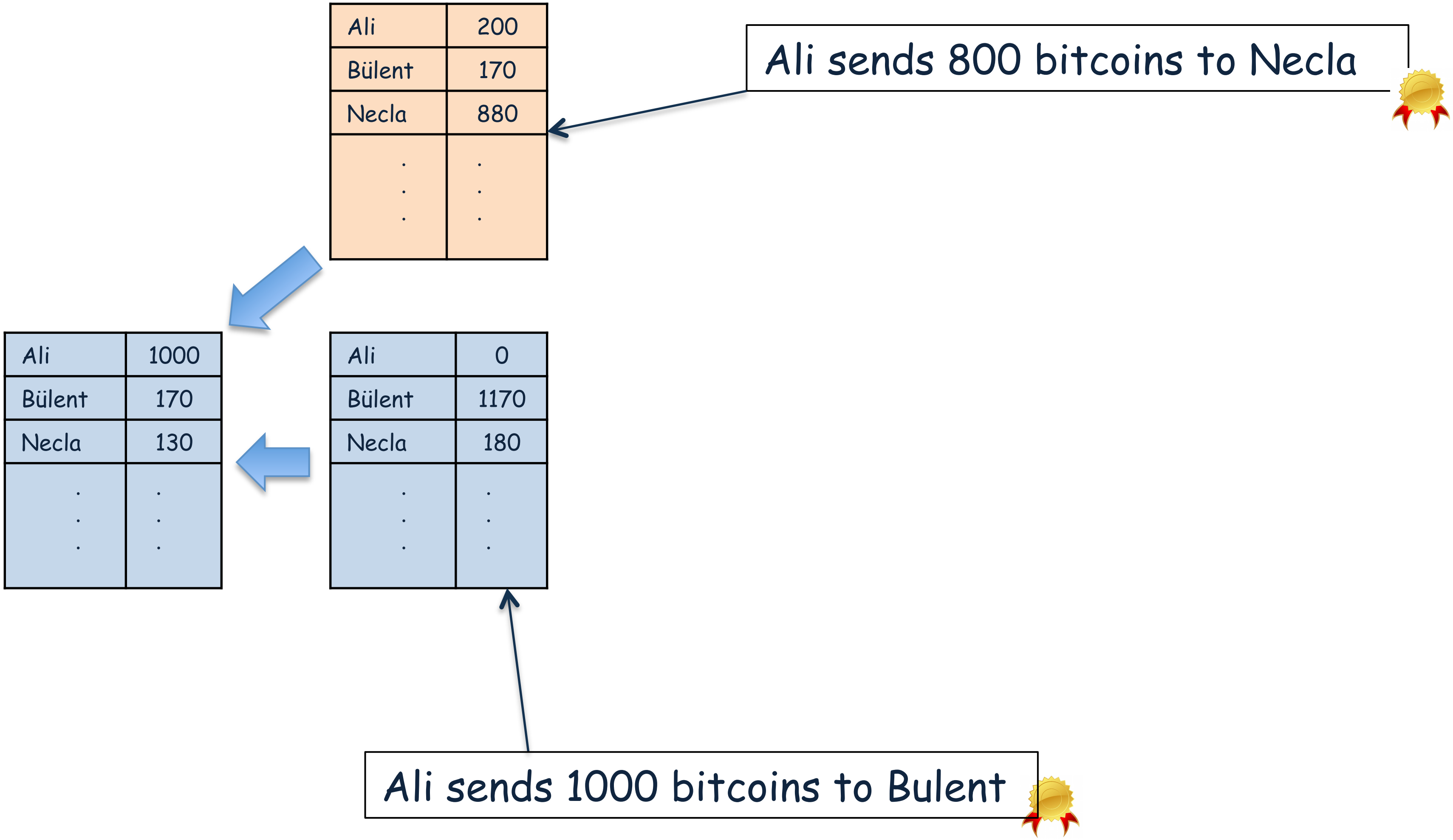
Ali	1000
Bülent	170
Necla	130
. . .	. . .

Ali	0
Bülent	1170
Necla	180
. . .	. . .

Ali sends 1000 bitcoins to Bulent

BITCOIN

Double-spending



Ali	200
Bülent	170
Necla	880
⋮	⋮

Ali sends 800 bitcoins to Necla

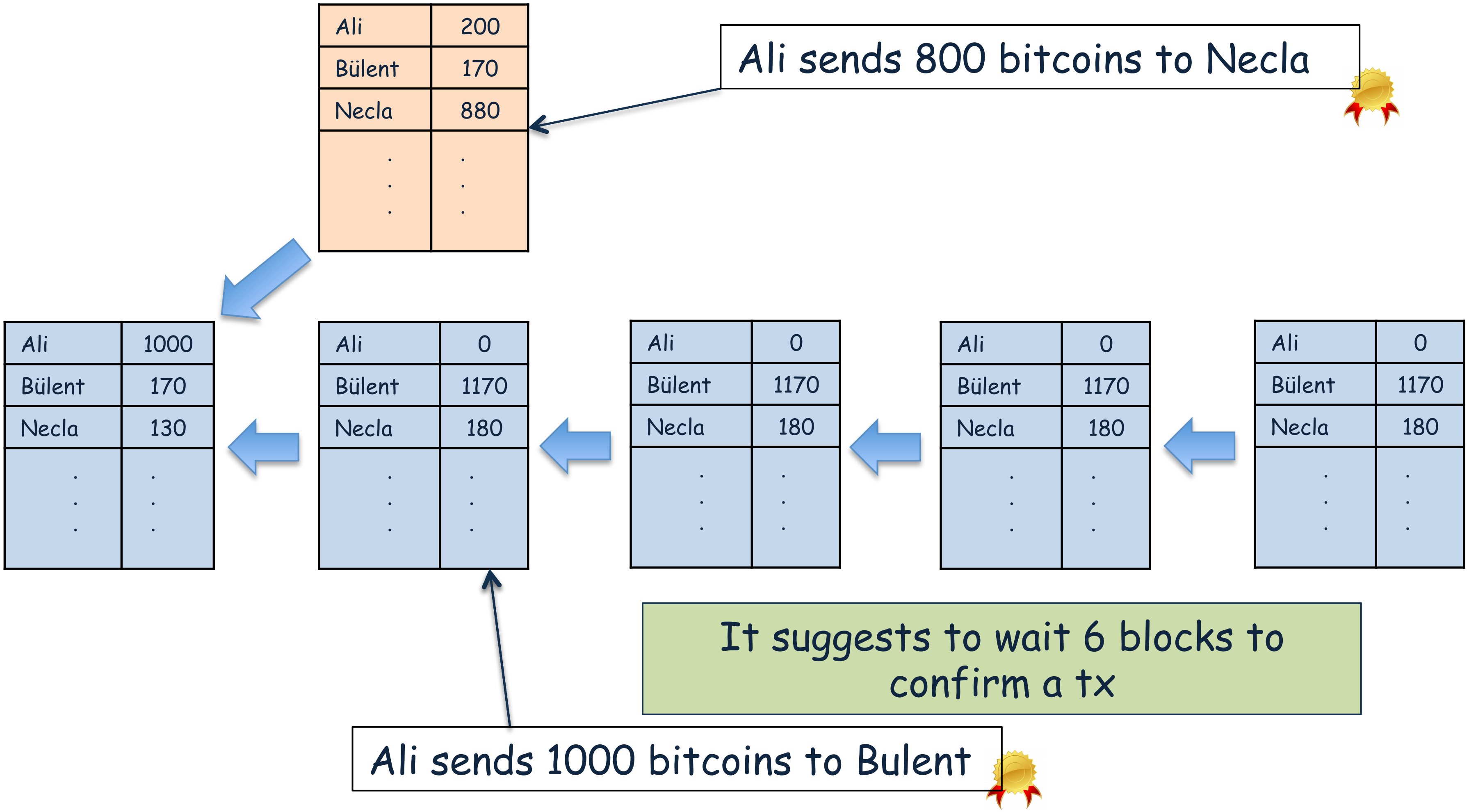
Ali	1000
Bülent	170
Necla	130
⋮	⋮

Ali	0
Bülent	1170
Necla	180
⋮	⋮

Ali sends 1000 bitcoins to Bulent

BITCOIN

Double-spending



Ali	200
Bülent	170
Necla	880
⋮	⋮

Ali sends 800 bitcoins to Necla

Ali	1000
Bülent	170
Necla	130
⋮	⋮

Ali	0
Bülent	1170
Necla	180
⋮	⋮

Ali	0
Bülent	1170
Necla	180
⋮	⋮

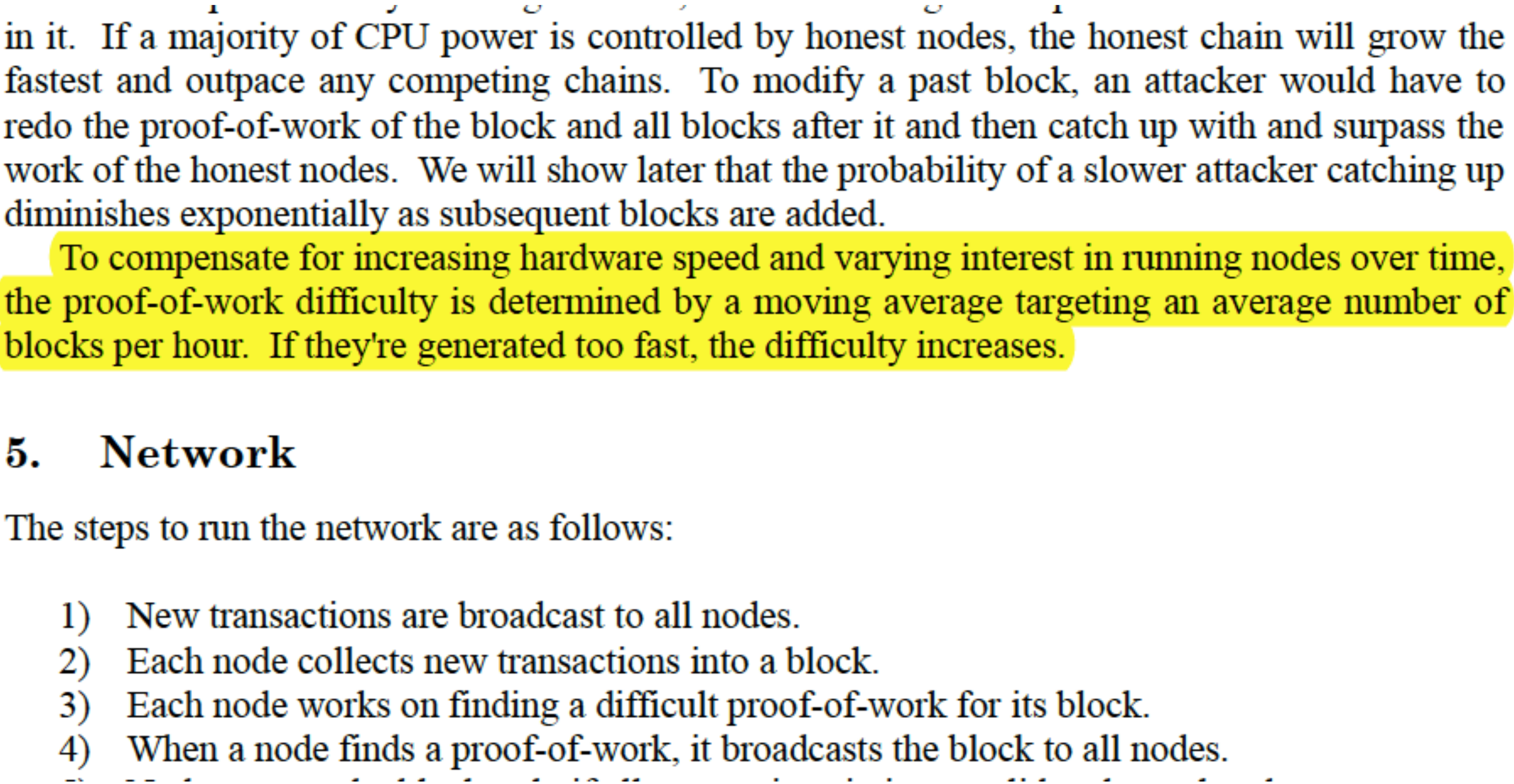
Ali	0
Bülent	1170
Necla	180
⋮	⋮

Ali	0
Bülent	1170
Necla	180
⋮	⋮

It suggests to wait 6 blocks to confirm a tx

Ali sends 1000 bitcoins to Bulent

BITCOIN



in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes. We will show later that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.

To compensate for increasing hardware speed and varying interest in running nodes over time, the proof-of-work difficulty is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases.

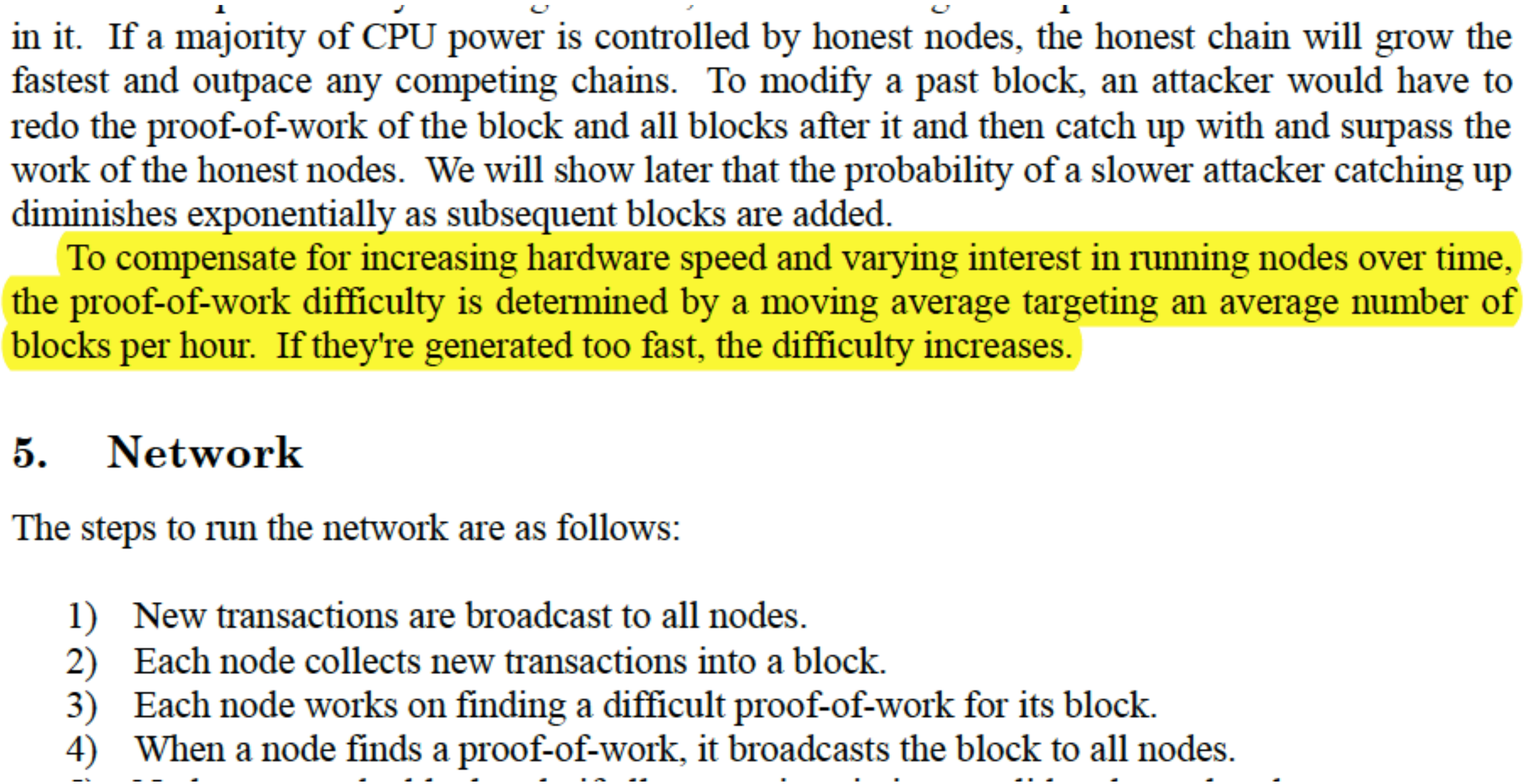
5. Network

The steps to run the network are as follows:

1) New transactions are broadcast to all nodes.
2) Each node collects new transactions into a block.
3) Each node works on finding a difficult proof-of-work for its block.
4) When a node finds a proof-of-work, it broadcasts the block to all nodes.

- The difficulty is adjusted for every 2016 blocks (~2 weeks) based on the time it took to mine the previous 2016 blocks

BITCOIN



in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes. We will show later that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.

To compensate for increasing hardware speed and varying interest in running nodes over time, the proof-of-work difficulty is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases.

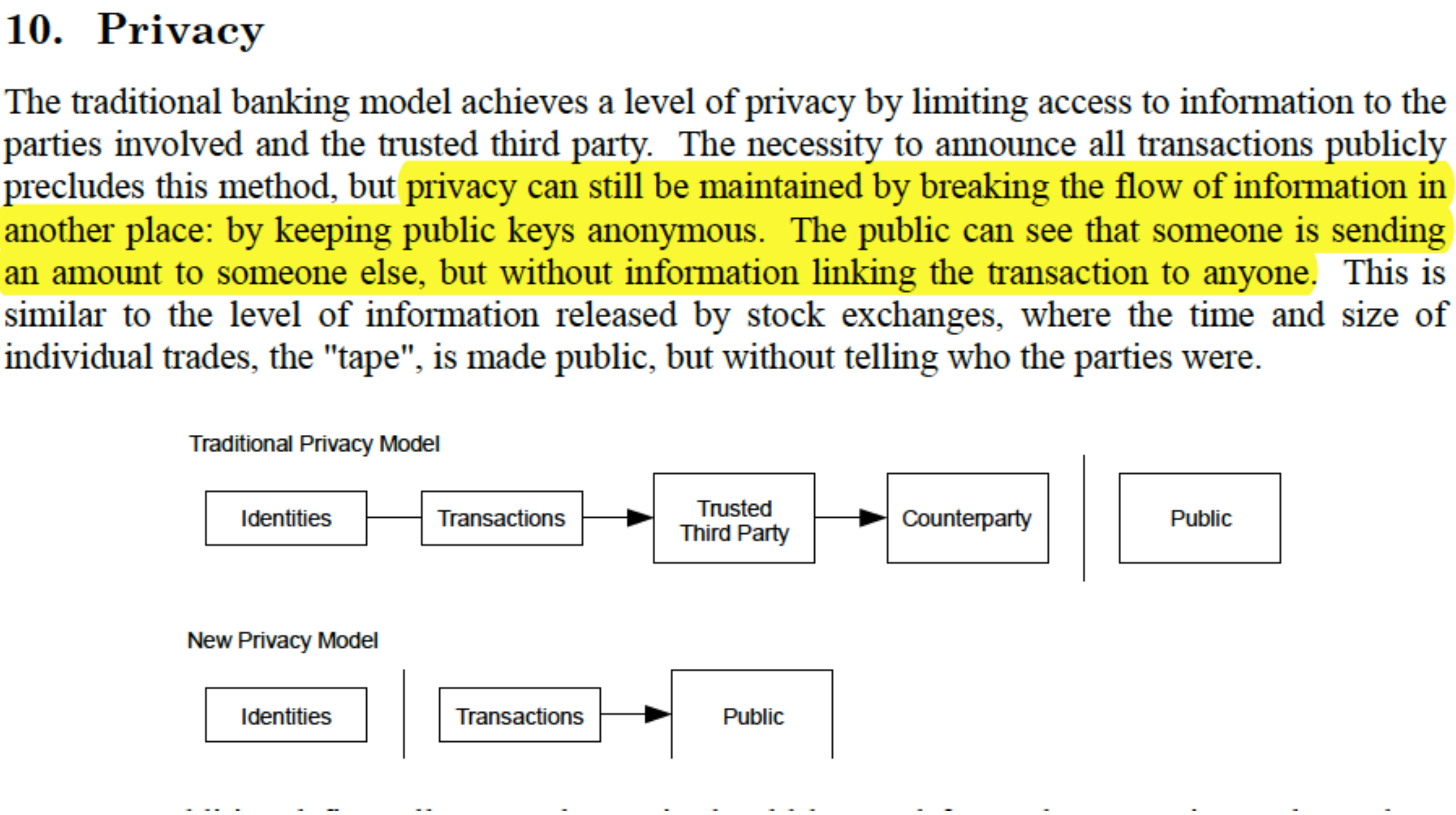
5. Network

The steps to run the network are as follows:

1) New transactions are broadcast to all nodes.
2) Each node collects new transactions into a block.
3) Each node works on finding a difficult proof-of-work for its block.
4) When a node finds a proof-of-work, it broadcasts the block to all nodes.

$$H(tx_1 \ldots tx_n \,\|\, x) < T$$

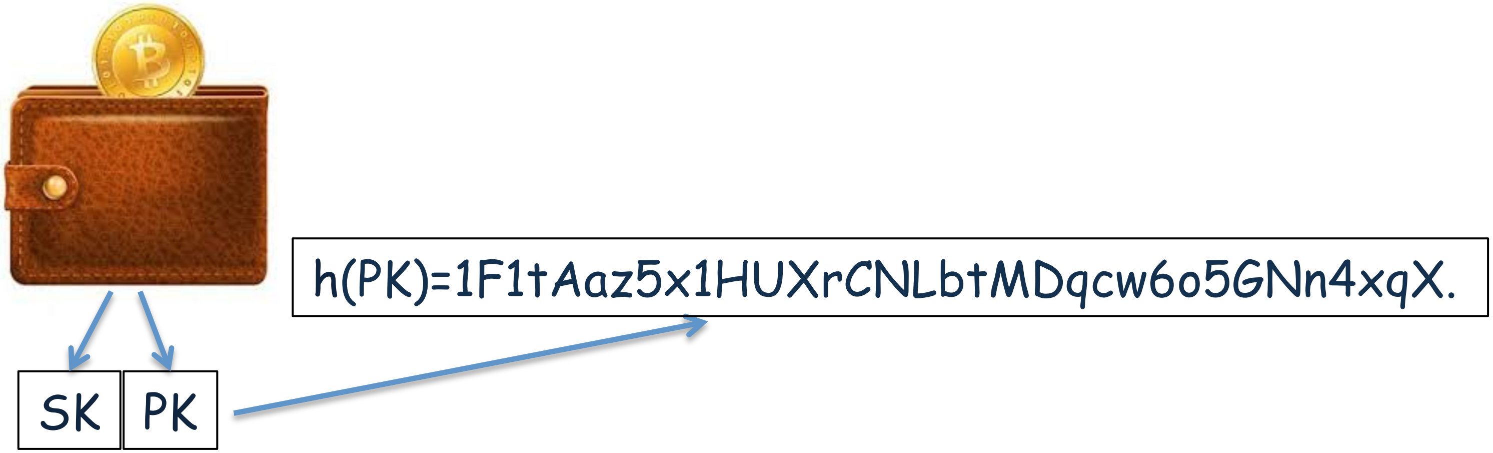
BITCOIN



10. Privacy

The traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. The necessity to announce all transactions publicly precludes this method, but privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous. The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone. This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the "tape", is made public, but without telling who the parties were.

BITCOIN

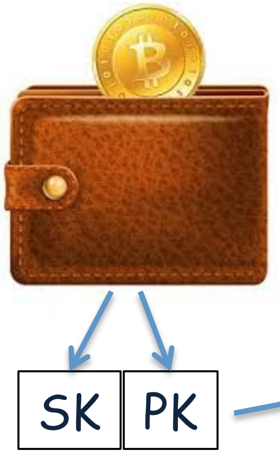


SK

PK

h(PK)=1F1tAaz5x1HUXrCNLbtMDqcw6o5GNn4xqX.

BITCOIN



SK PK

h(PK)=1F1tAaz5x1HUXrCNLbtMDqcw6o5GNn4xqX.

1F1tAaz5x1HUXrCNLbtMDqcw6o5GNn4xqX	32
18RcfsA8S2G3AQcJqBEnaDFc2Mb16SKRsE	12
12gtdfsgNvrDWxgDrJi38M5M2oAUGJcNMS	45
. . .	. . .

- identities are hidden

BITCOIN

- The reward is cut in half every four years (210.000 blocks)

BITCOIN

- The reward is cut in half every four years (210.000 blocks)
- At the beginning (2009), the reward was 50 BTC. (Now 6.25 BTC)

BITCOIN

- The reward is cut in half every four years (210.000 blocks)
- At the beginning (2009), the reward was 50 BTC. (Now 6.25 BTC)
- Around 2140, it drops below 1 satoshi, and no more creation after that block (1 BTC = 100.000.000 satoshi)

BITCOIN

- The reward is cut in half every four years (210.000 blocks)

- At the beginning (2009), the reward was 50 BTC. (Now 6.25 BTC)

- Around 2140, it drops below 1 satoshi, and no more creation after that block
  (1 BTC = 100.000.000 satoshi)

- It promises scare token economy with an eventual cap of about 21M bitcoins

BITCOIN

- A protocol that supports a decentralized, pseudo-anonymous, peer-to-peer digital currency

BITCOIN

- A protocol that supports a decentralized, pseudo-anonymous, peer-to-peer digital currency

- A publicly disclosed linked ledger of transactions stored in a blockchain

BITCOIN

- A protocol that supports a decentralized, pseudo-anonymous, peer-to-peer digital currency

- A publicly disclosed linked ledger of transactions stored in a blockchain

- A reward driven system for achieving consensus (mining) based on "Proofs of Work" for helping to secure the network

BITCOIN

History

BITCOIN

History

2008

- August 18 Domain name "bitcoin.org" registered

BITCOIN

History

2008

- August 18 Domain name "bitcoin.org" registered
- October 31 Bitcoin design paper published

BITCOIN

History

2008

- August 18 Domain name "bitcoin.org" registered
- October 31 Bitcoin design paper published
- November 09 Bitcoin project registered at SourceForge.net

BITCOIN

History

2008

- August 18 Domain name "bitcoin.org" registered
- October 31 Bitcoin design paper published
- November 09 Bitcoin project registered at SourceForge.net

2009

- January 3 Genesis block established at 18:15:05 GMT

BITCOIN

History

2008

- August 18 Domain name "bitcoin.org" registered
- October 31 Bitcoin design paper published
- November 09 Bitcoin project registered at SourceForge.net

2009

- January 3 Genesis block established at 18:15:05 GMT
- January 9 Bitcoin v0.1 released and announced on the cryptography mailing list

BITCOIN

History

2008

- August 18 Domain name "bitcoin.org" registered
- October 31 Bitcoin design paper published
- November 09 Bitcoin project registered at SourceForge.net

2009

- January 3 Genesis block established at 18:15:05 GMT
- January 9 Bitcoin v0.1 released and announced on the cryptography mailing list
- January 12 First Bitcoin transaction, in block 170 from Satoshi to Hal Finney