Bitcoin
A Technical Perspective

October 24, 2017

Martín Ugarte

Bitcoin: A Peer-to-Peer Electronic Cash System

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. 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.

P2P Electronic Cash: Main Challenges

Alice	400
Bob	200

P2P Electronic Cash: Main Challenges

Transfer 100
to Bob

Alice	400
Bob	200

P2P Electronic Cash: Main Challenges

Transfer 100
to Bob

Alice	300
Bob	300

P2P Electronic Cash: Main Challenges

Alice	300
Bob	300

P2P Electronic Cash: Main Challenges

Alice	300
Bob	300

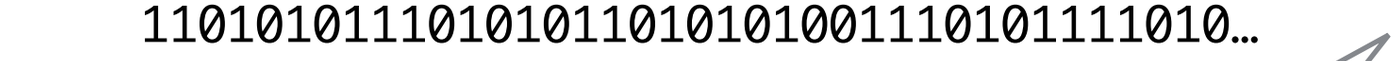
P2P Electronic Cash: Main Challenges

P2P Electronic Cash: Main Challenges

P2P Electronic Cash: Main Challenges

P2P Electronic Cash: Main Challenges

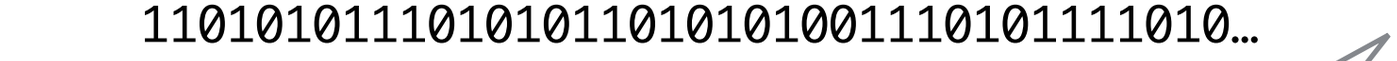
Here Bob, I'm paying you 10



Is this Alice?
Does Alice own 10?
Is she using these 10 elsewhere?

P2P Electronic Cash: Main Challenges

Here Bob, I'm paying you 10



Is this Alice?
Does Alice own 10?
Is she using these 10 elsewhere?

Also: where does money come from?

Digital Signatures

Digital Signatures

Digital Signatures

Is this really from Alice?

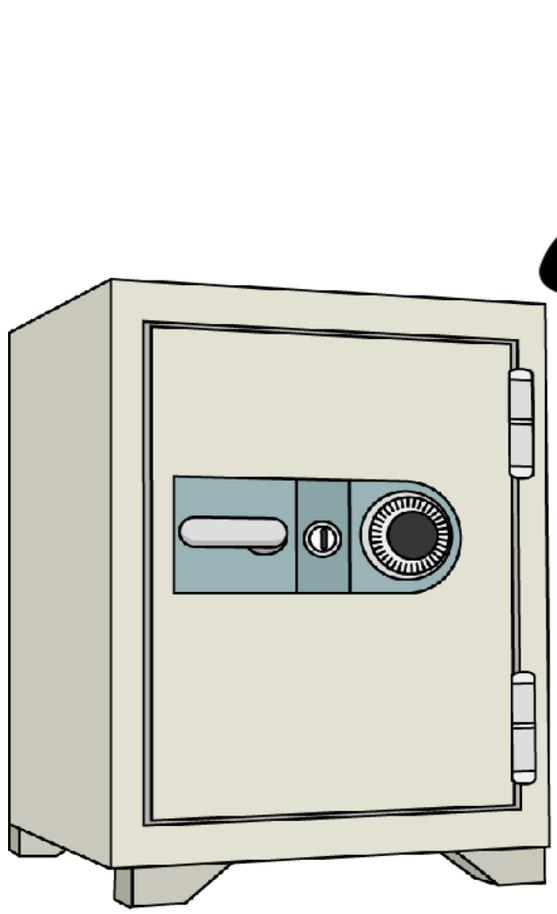
Digital Signatures

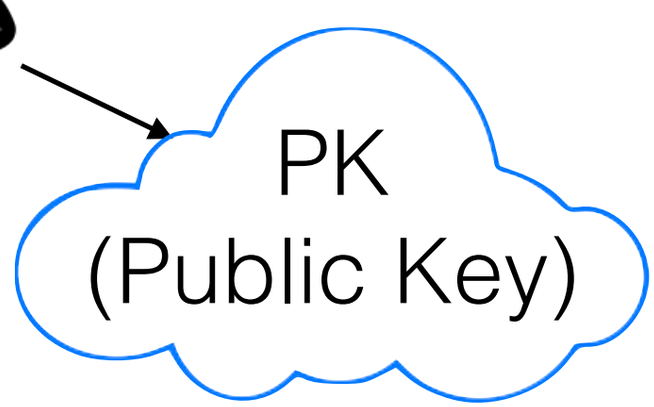
Digital Signatures

SK
(Secret Key)

PK
(Public Key)

Digital Signatures

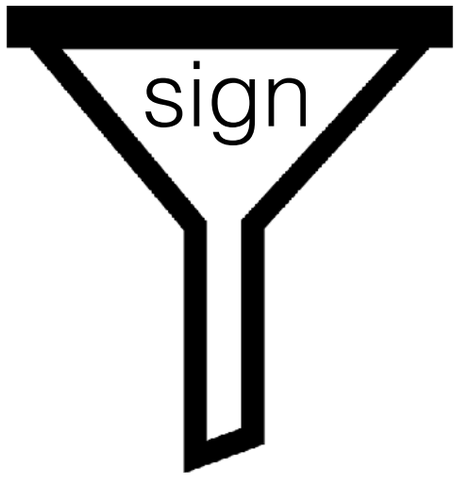




Digital Signatures

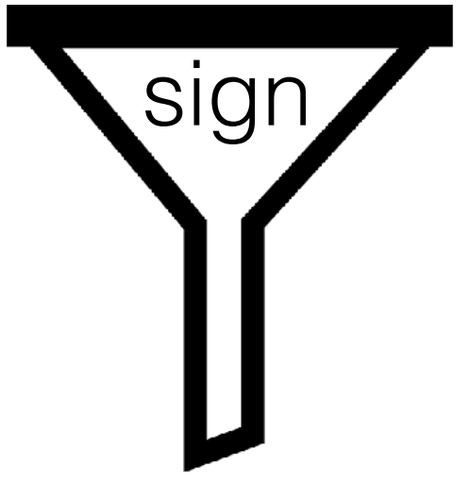
Digital Signatures

SK



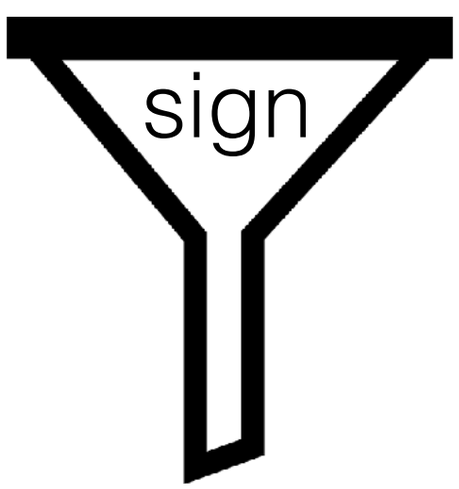
Digital Signatures

SK



Digital Signatures

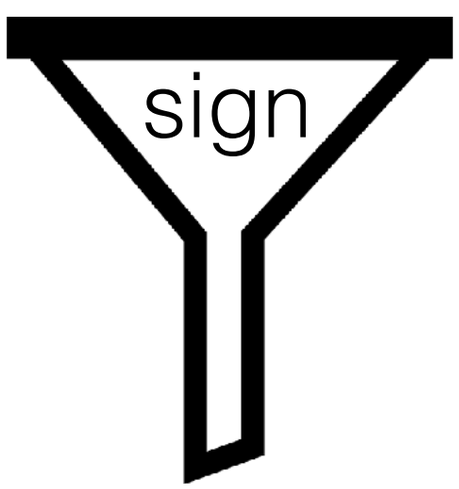
SK

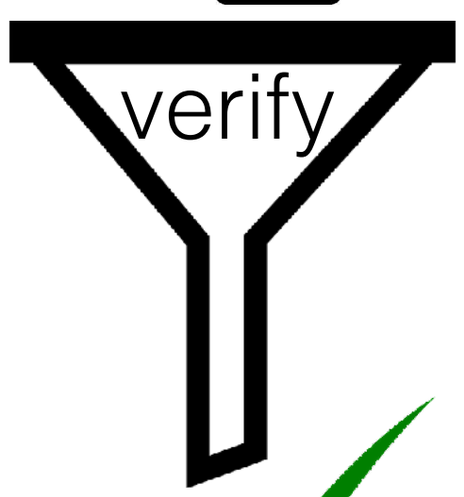


Digital Signatures

SK

PK



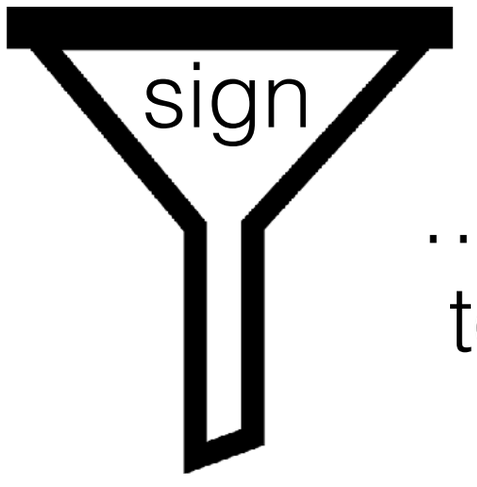


Digital Signatures

SK

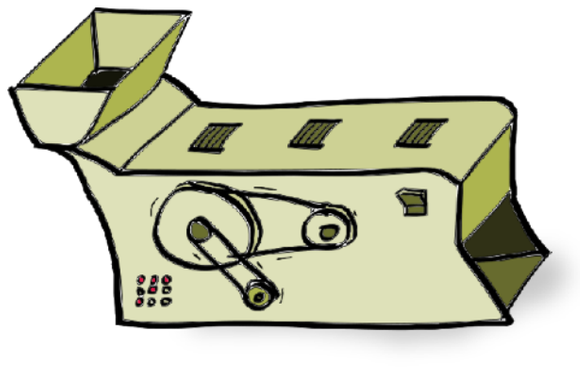
Impossible to sign without SK…

PK

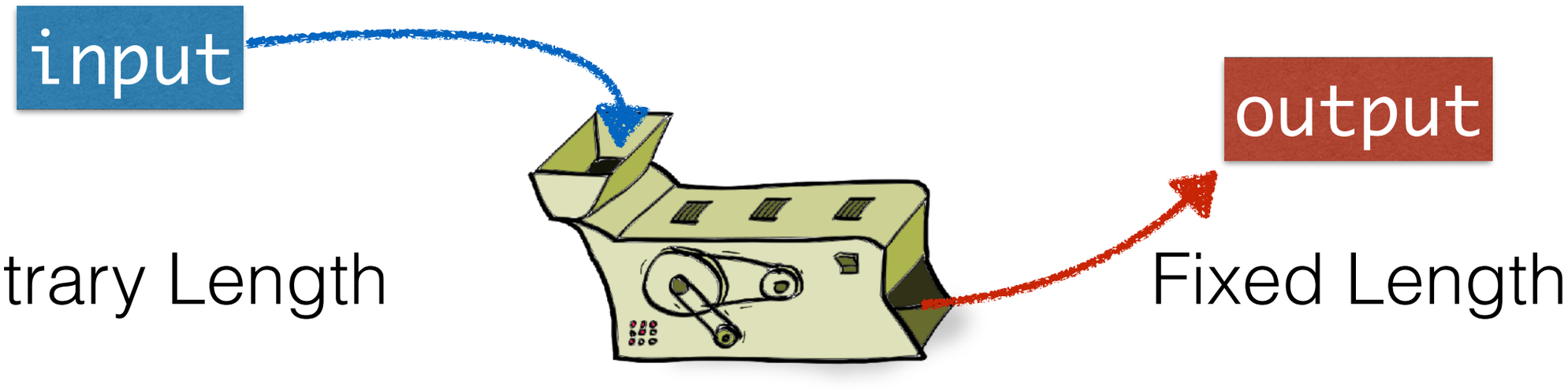


…even with PK and access to other signed messages

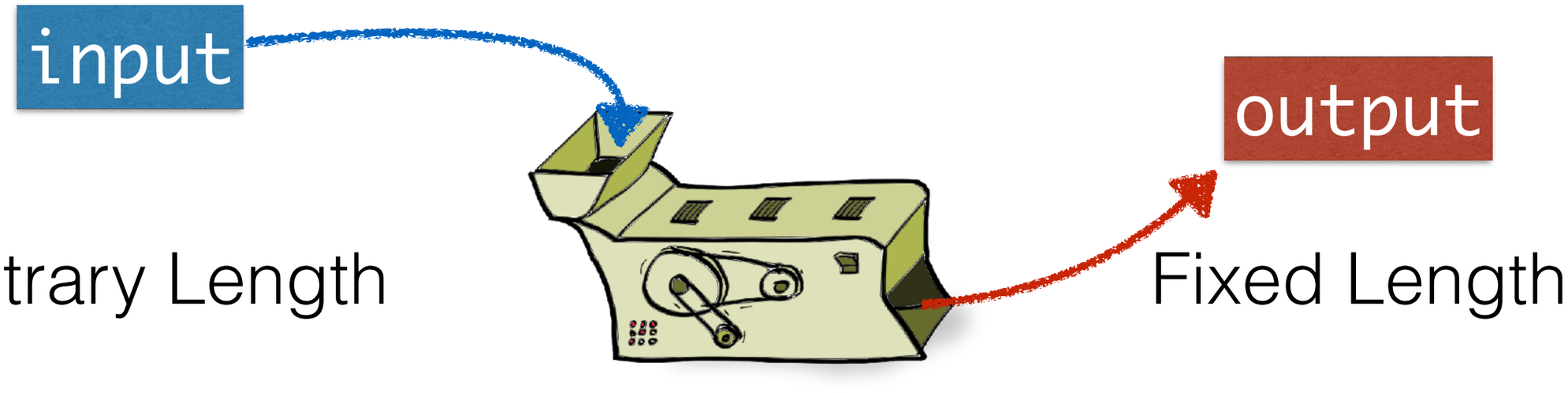
Cryptographic Hash Functions



Cryptographic Hash Functions

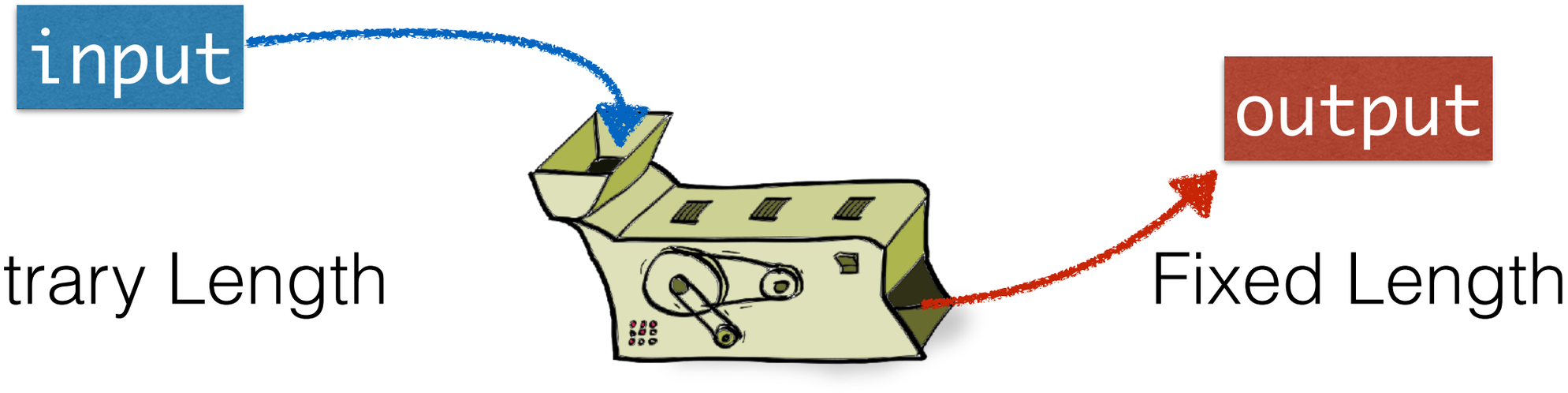


Cryptographic Hash Functions



Output gives no information about input(s)

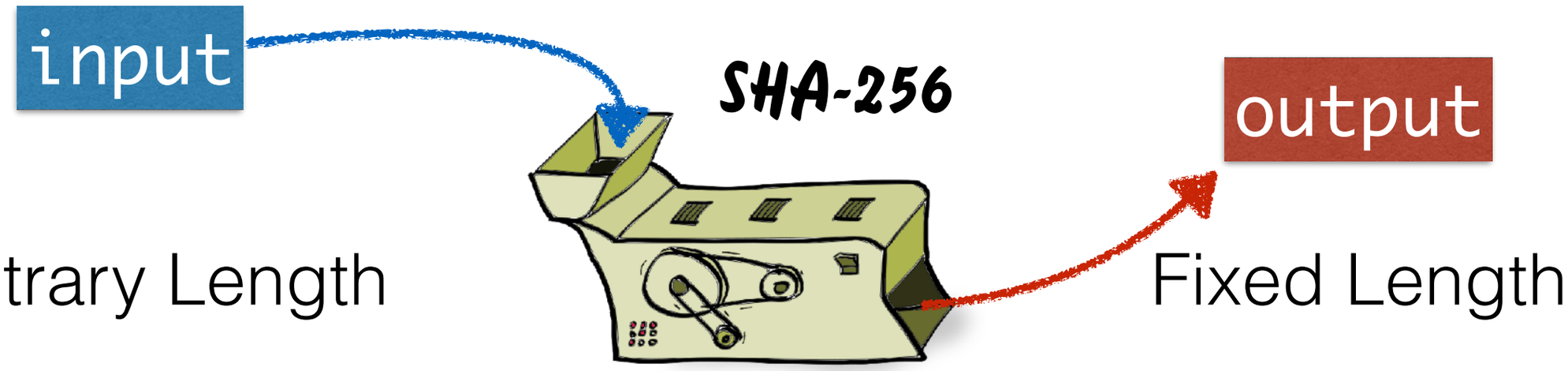
Cryptographic Hash Functions



Output gives no information about input(s)

Collision resistance

Cryptographic Hash Functions



Arbitrary Length

Fixed Length

Output gives no information about input(s)

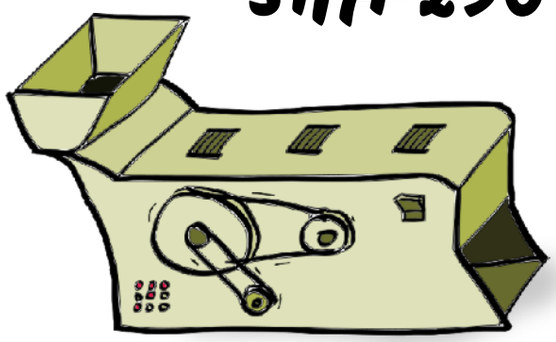
Collision resistance

Cryptographic Hash Functions

Cryptographic Hash Functions

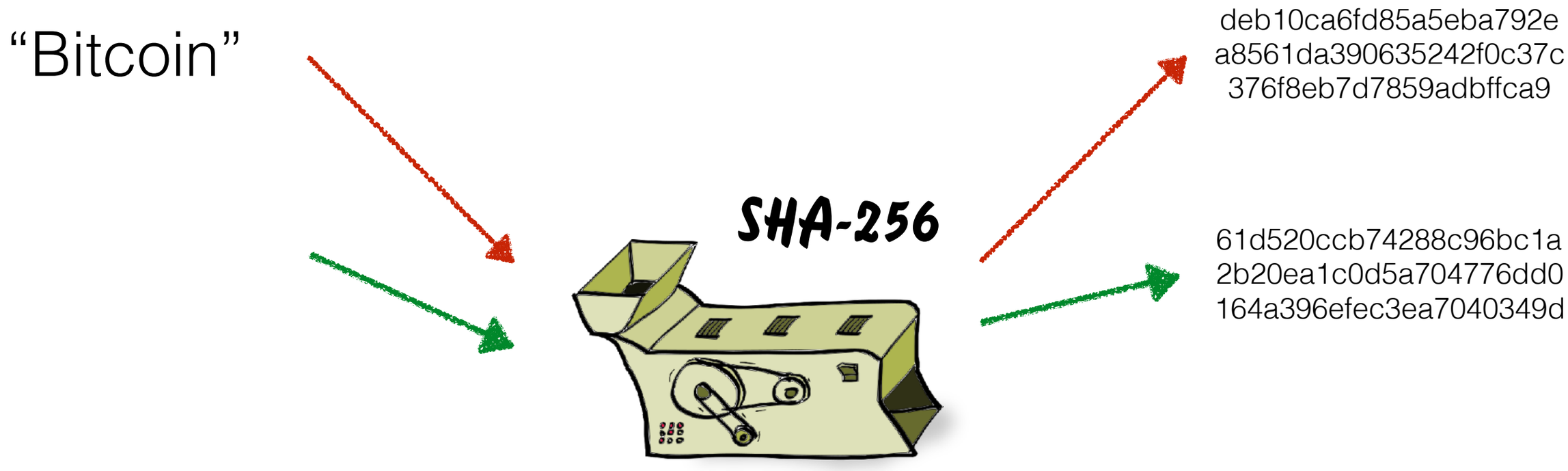
“Bitcoin”

SHA-256

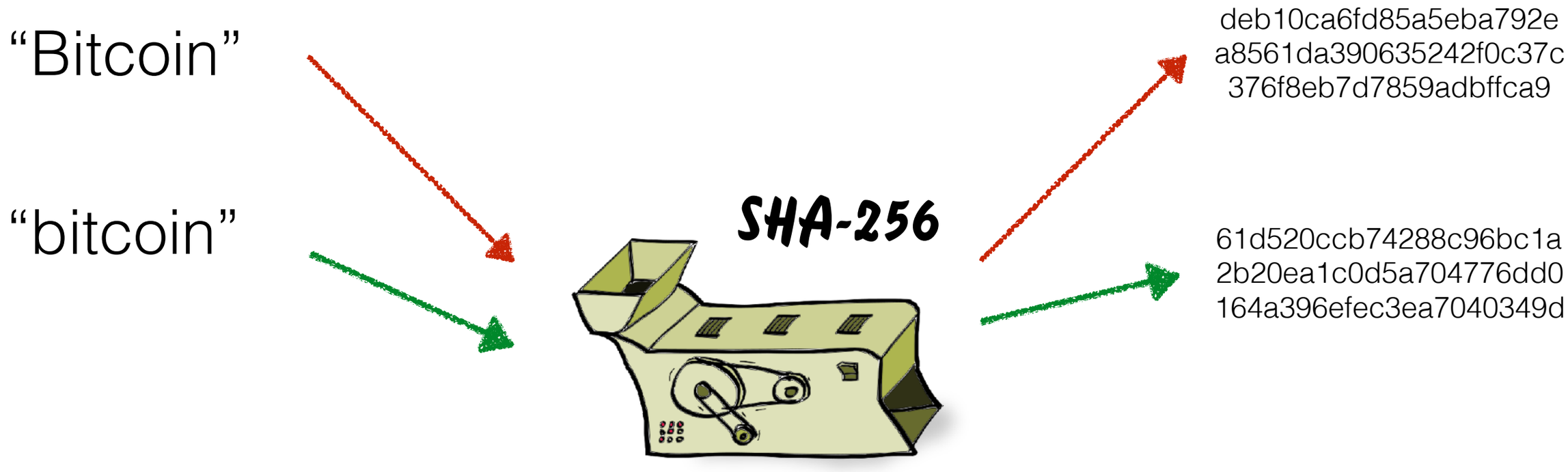


deb10ca6fd85a5eba792e
a8561da390635242f0c37c
376f8eb7d7859adbffca9

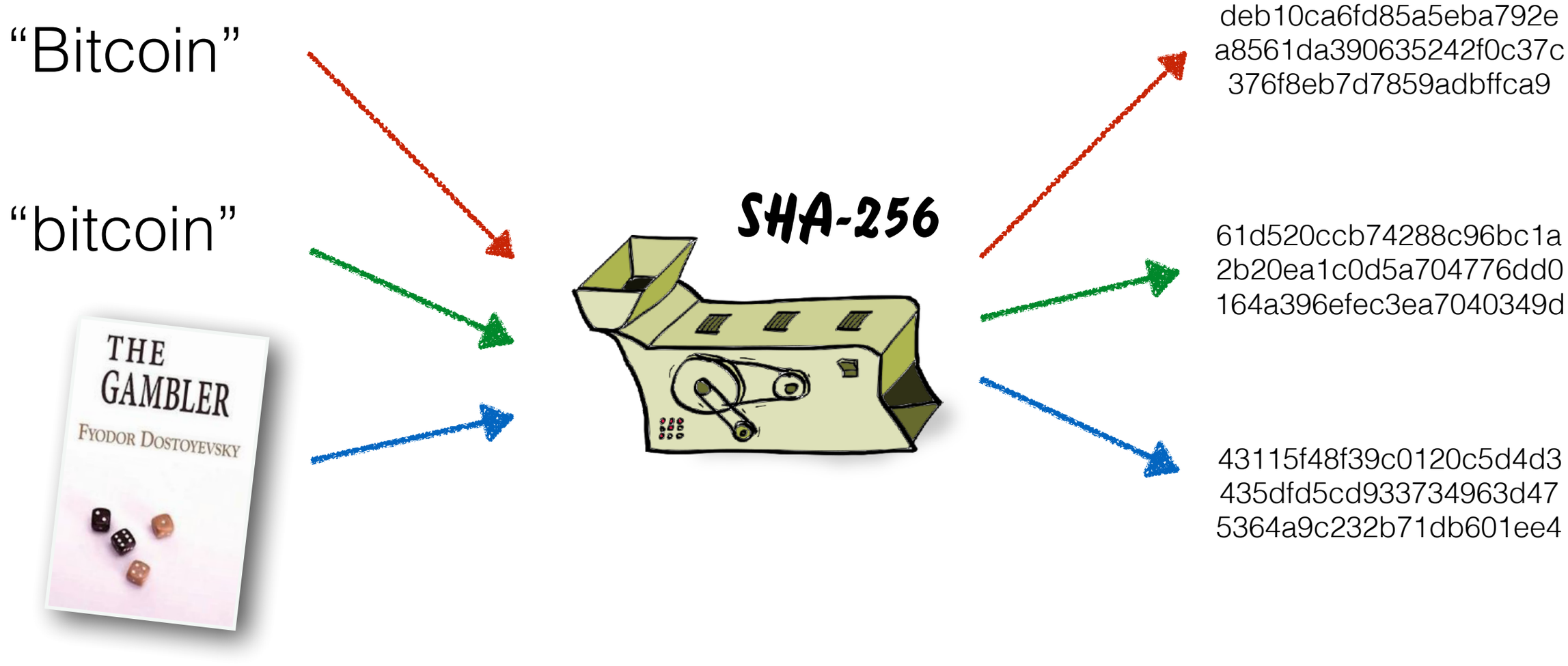
Cryptographic Hash Functions



Cryptographic Hash Functions



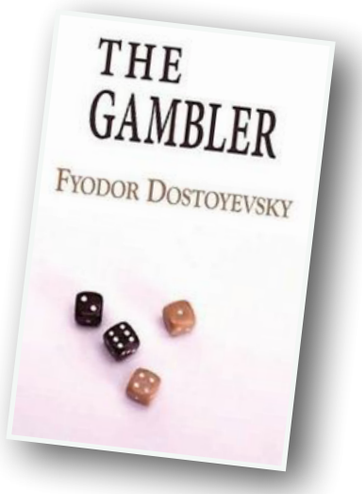
Cryptographic Hash Functions

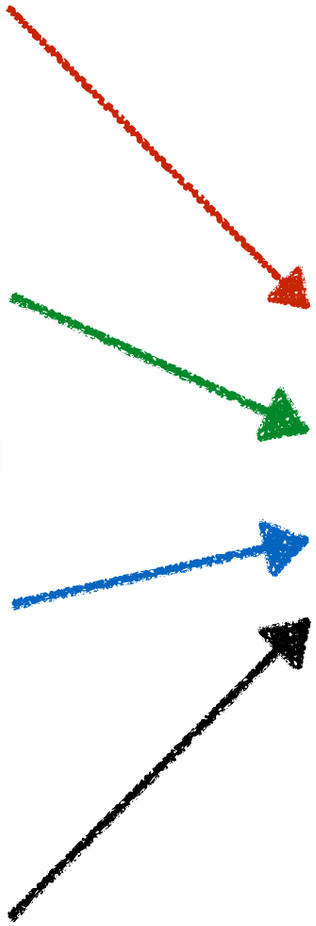


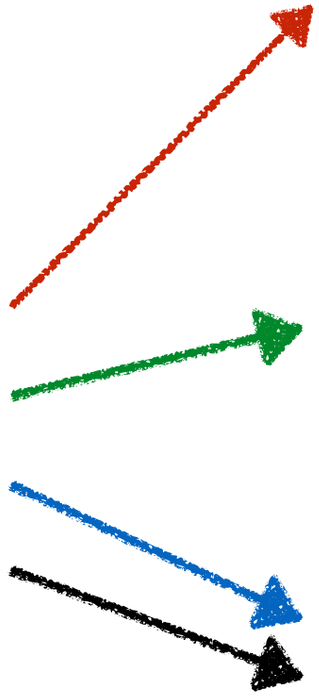
Cryptographic Hash Functions

“Bitcoin”

“bitcoin”





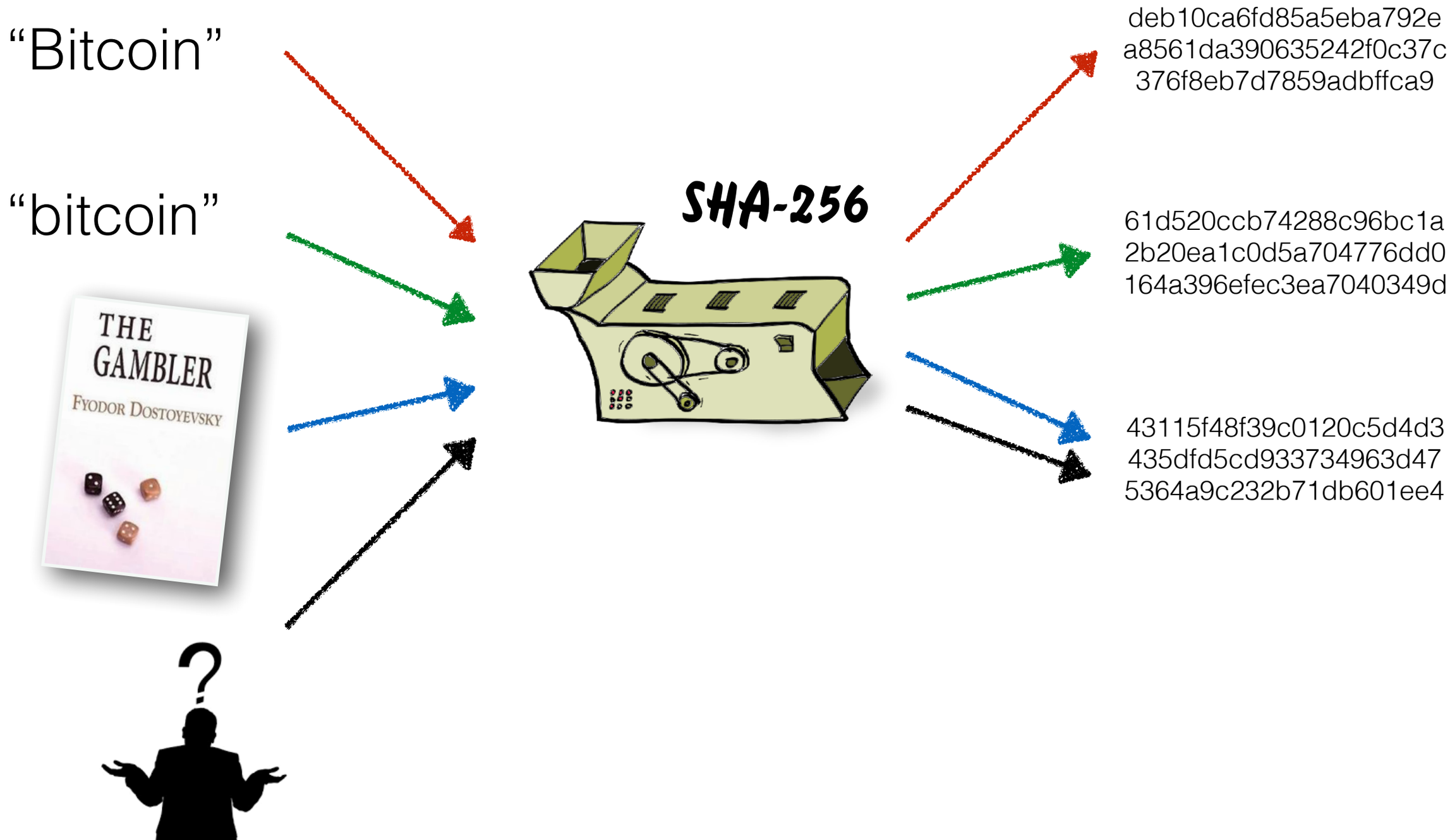


deb10ca6fd85a5eba792e
a8561da390635242f0c37c
376f8eb7d7859adbffca9

61d520ccb74288c96bc1a
2b20ea1c0d5a704776dd0
164a396efec3ea7040349d

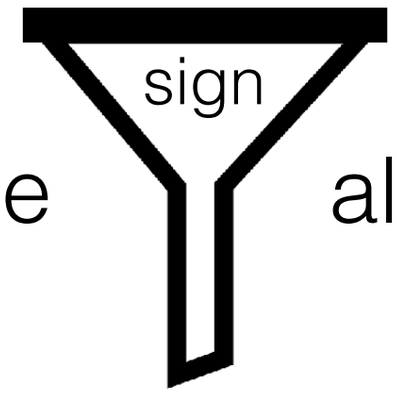
43115f48f39c0120c5d4d3
435dfd5cd933734963d47
5364a9c232b71db601ee4

Cryptographic Hash Functions



An Application to Digital Signatures

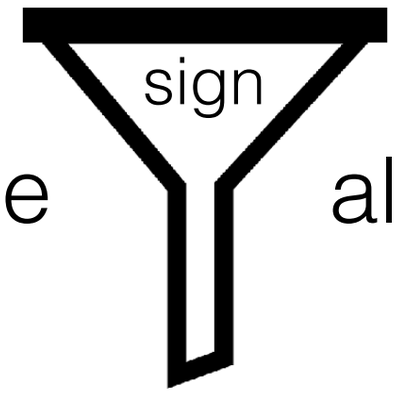
An Application to Digital Signatures

If the 

algorithm is computationally demanding:

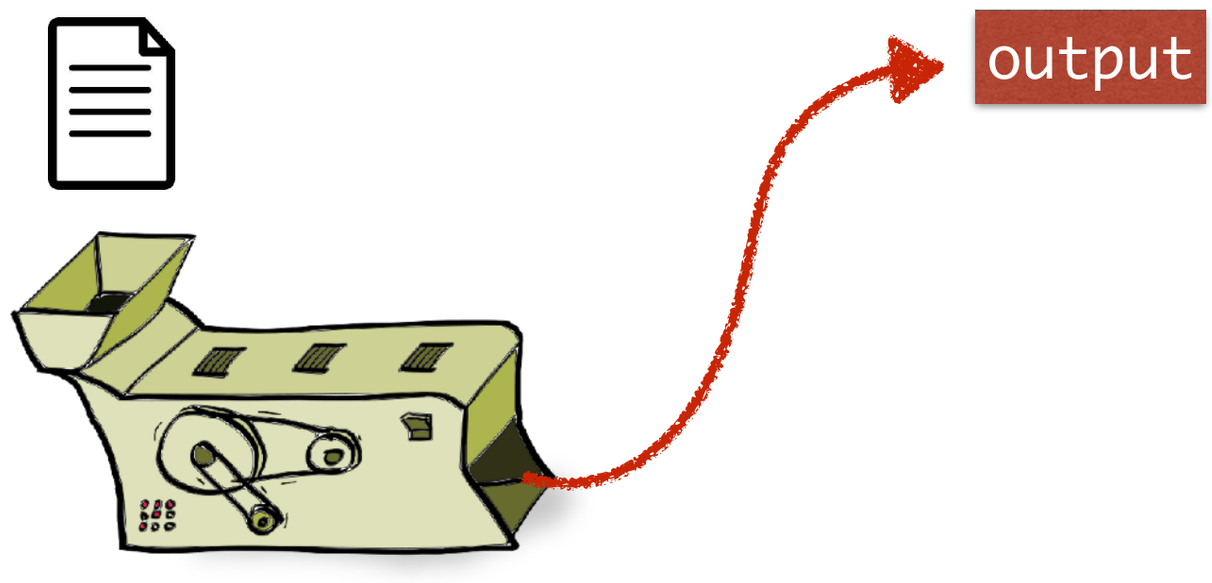
How can we speed up the signing process?

An Application to Digital Signatures

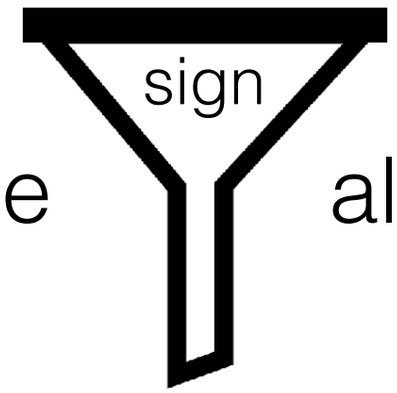


If the algorithm is computationally demanding:

How can we speed up the signing process?

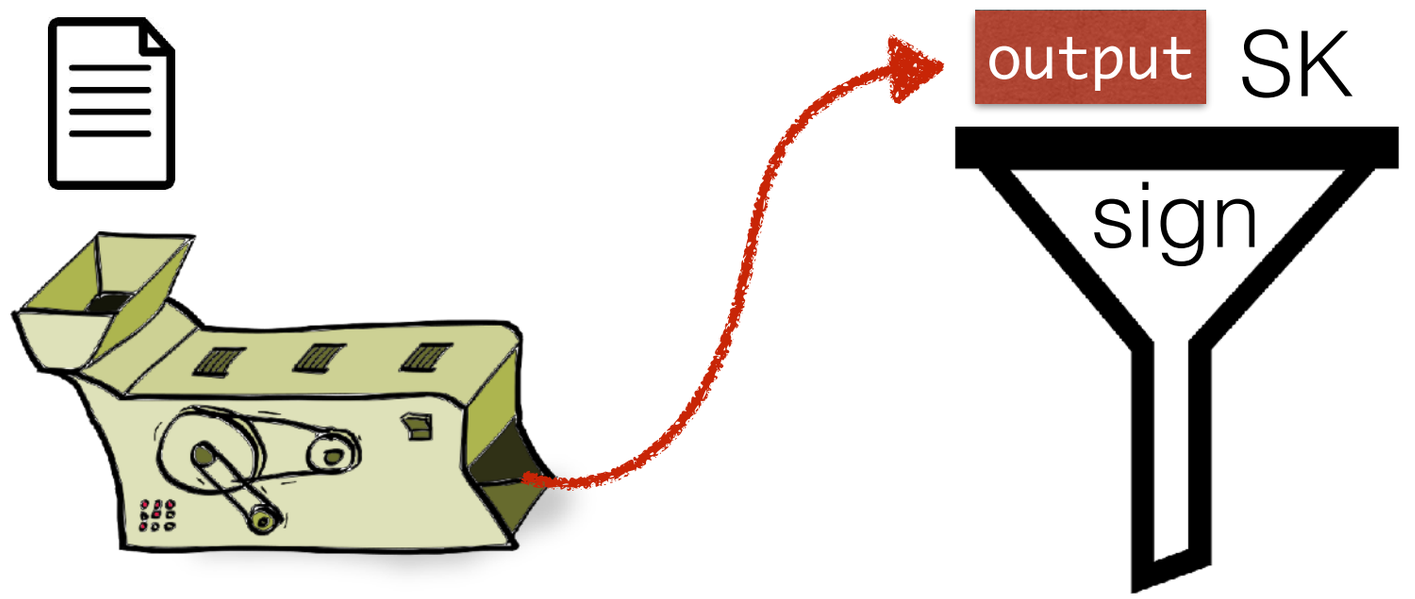


An Application to Digital Signatures

If the 

algorithm is computationally demanding:

How can we speed up the signing process?



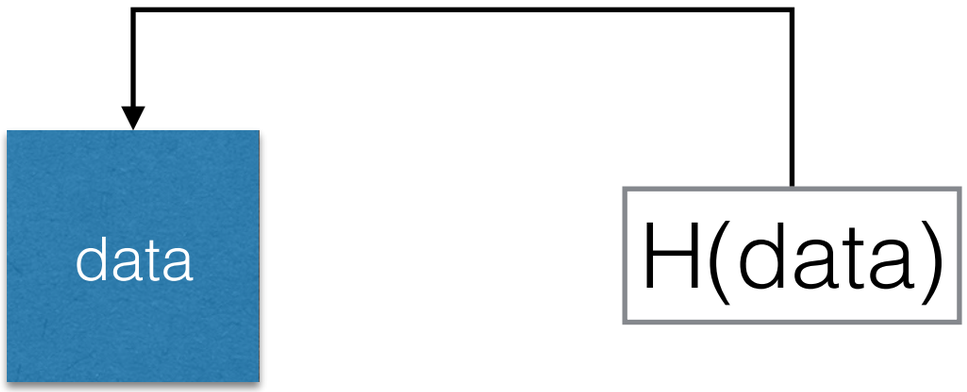
Hash Pointers and Integrity

Hash Pointers and Integrity

We can later verify that the data has not been modified.

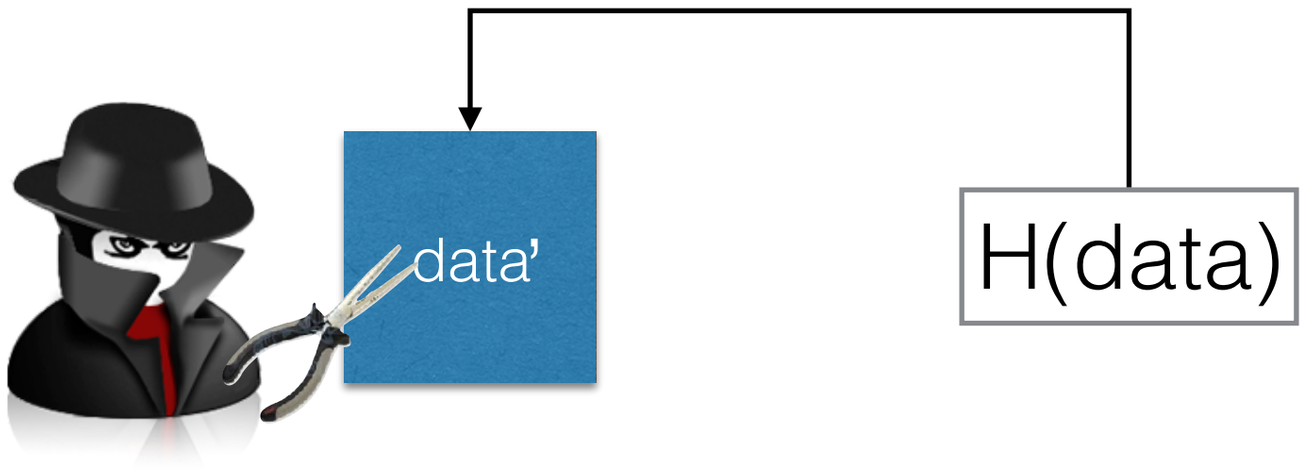
Hash Pointers and Integrity

Hash Pointers and Integrity



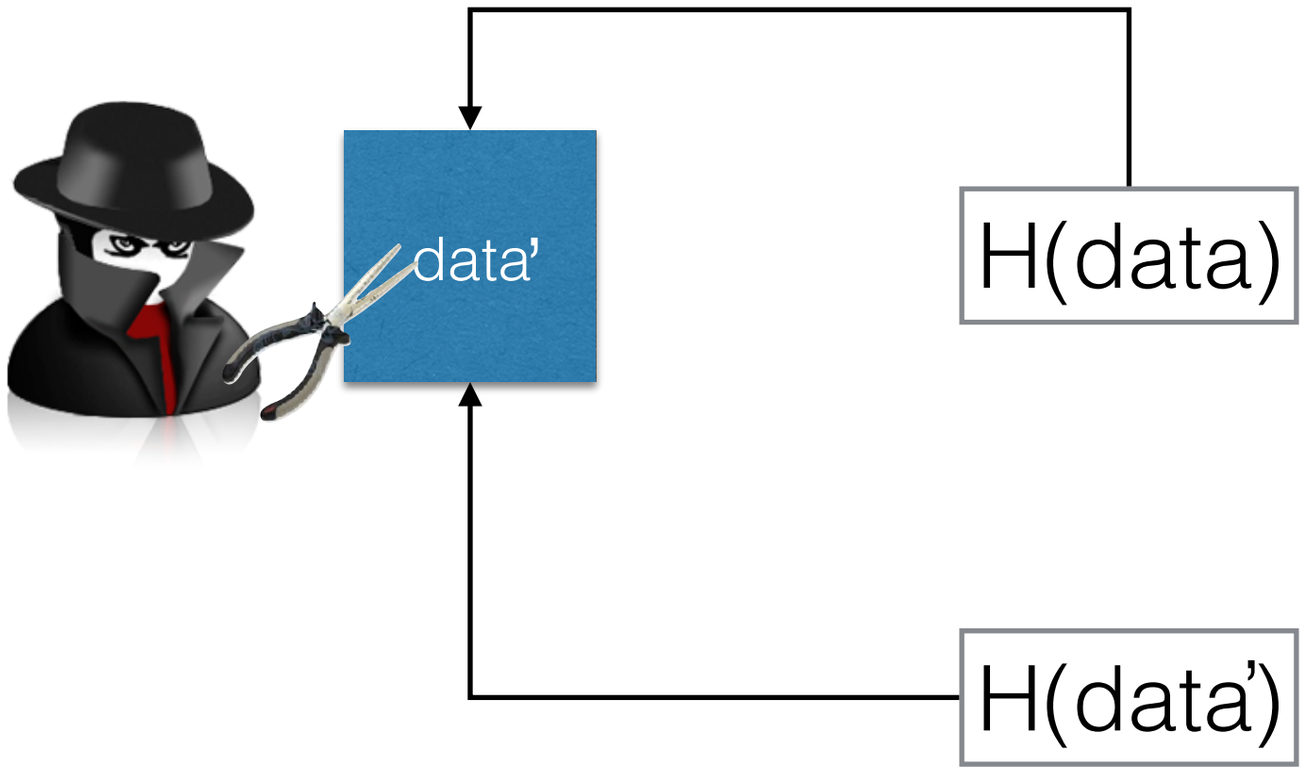
H(data)

Hash Pointers and Integrity

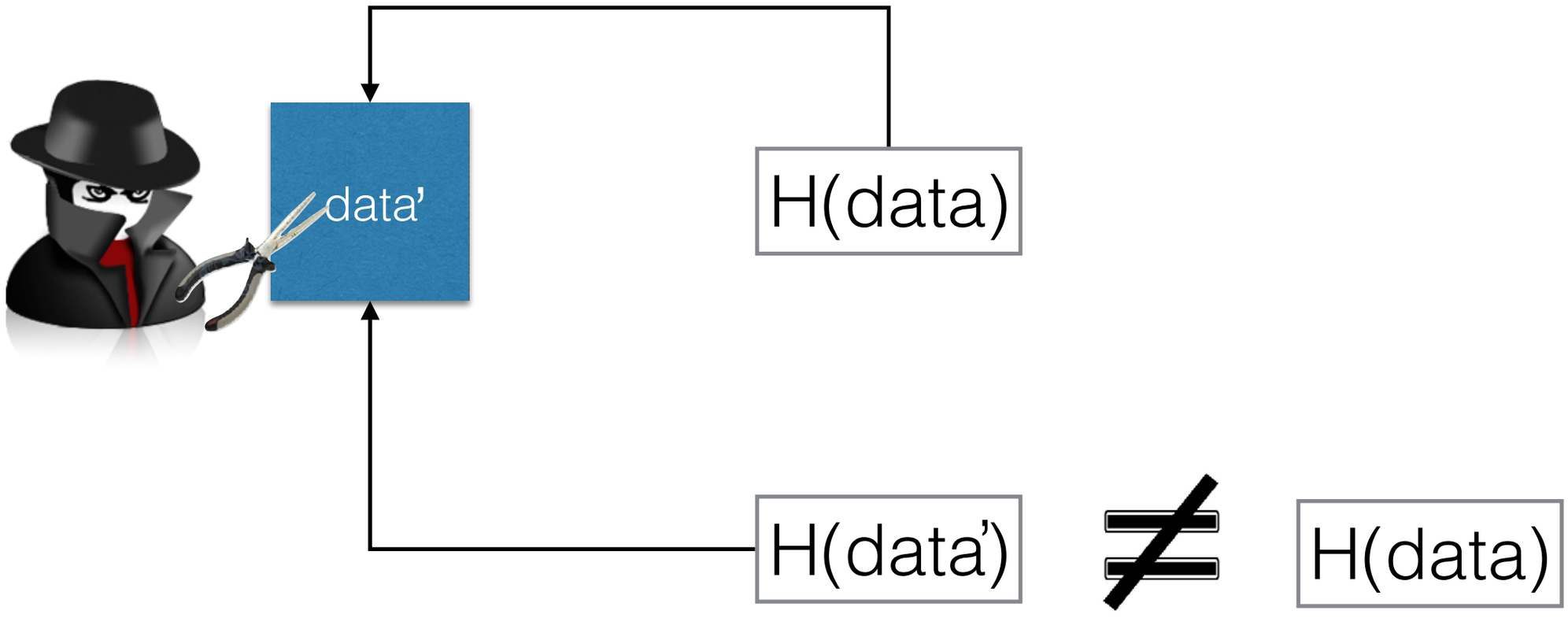


H(data)

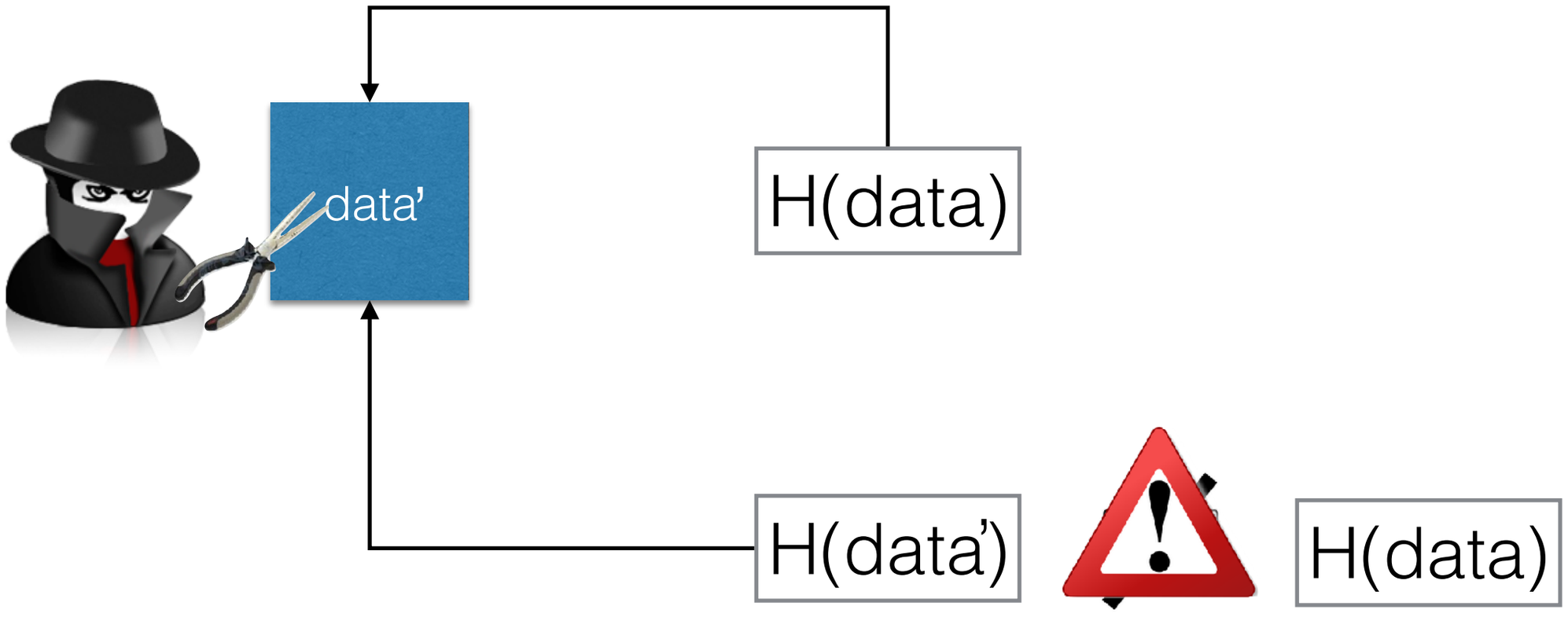
Hash Pointers and Integrity



Hash Pointers and Integrity

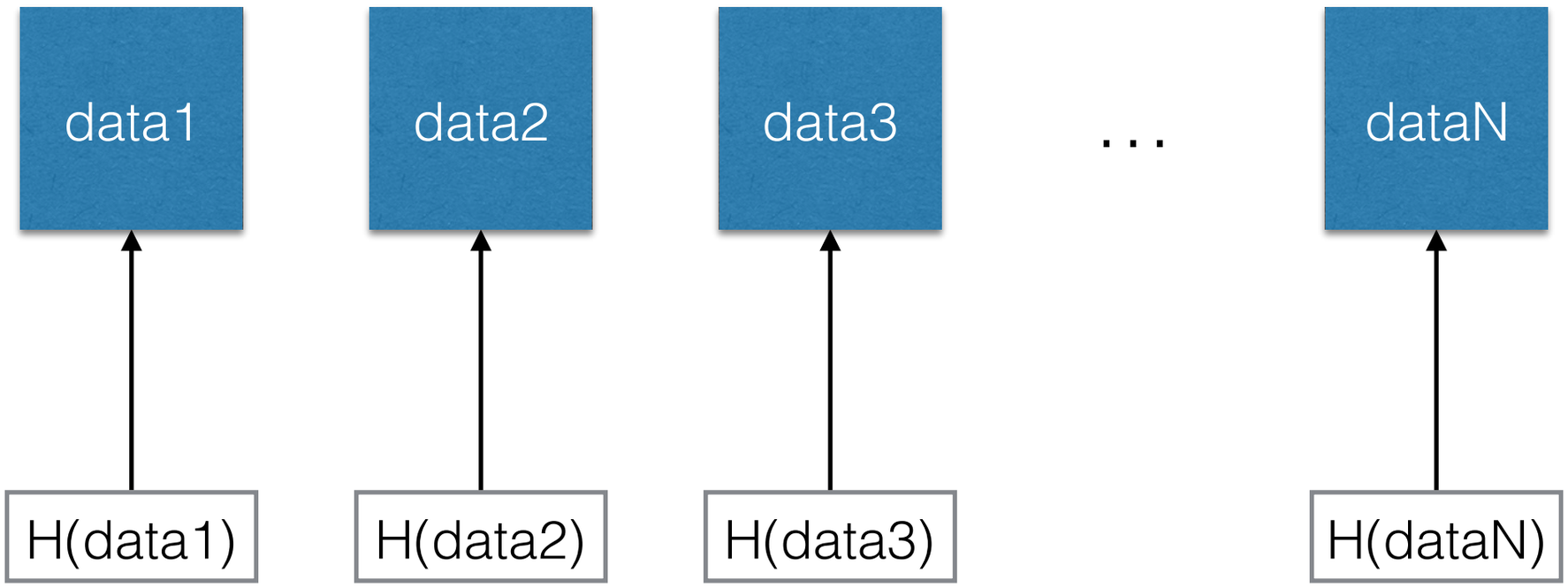


Hash Pointers and Integrity

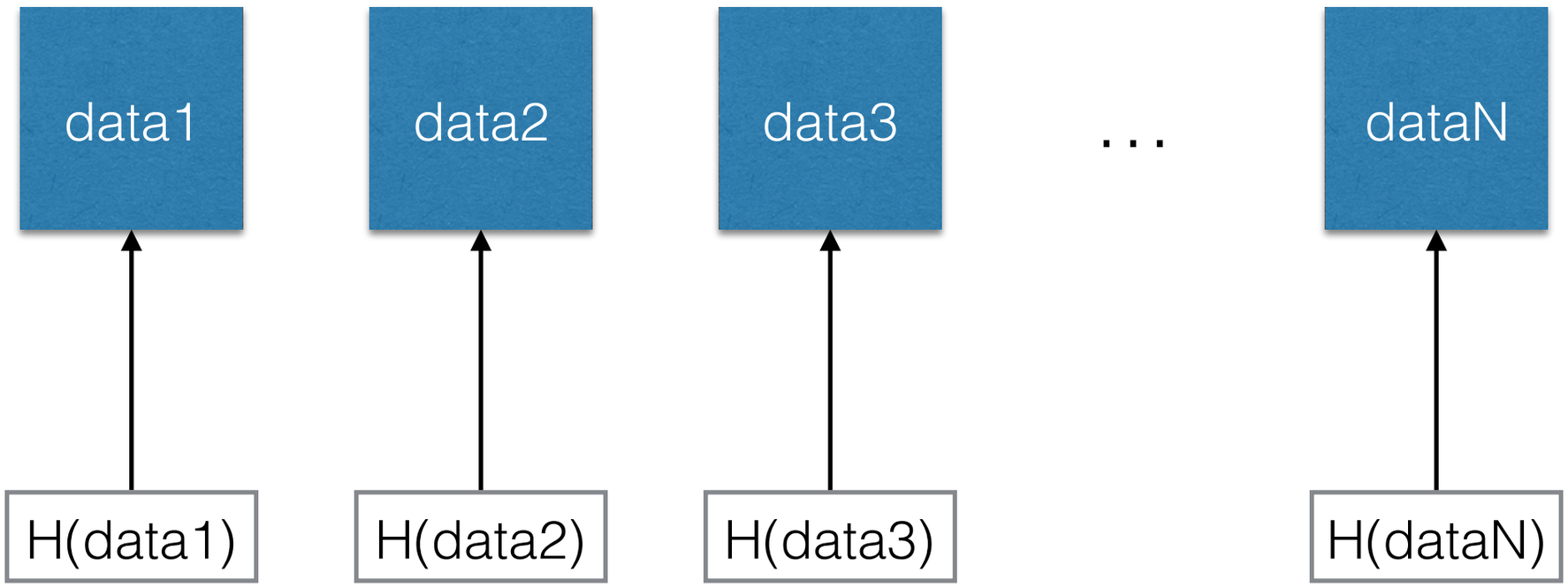


What if data is added continuously?

What if data is added continuously?



What if data is added continuously?

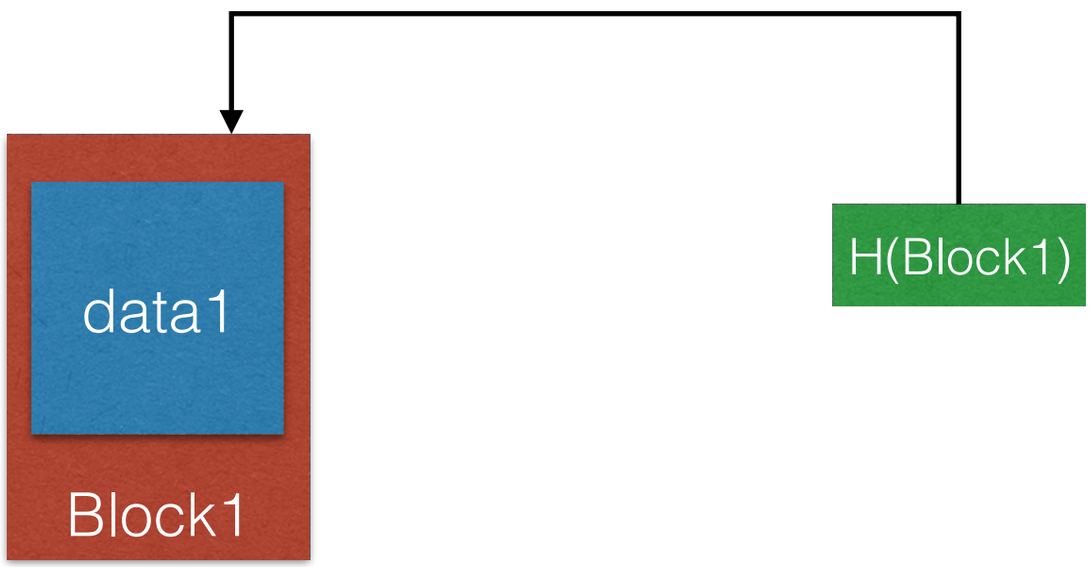


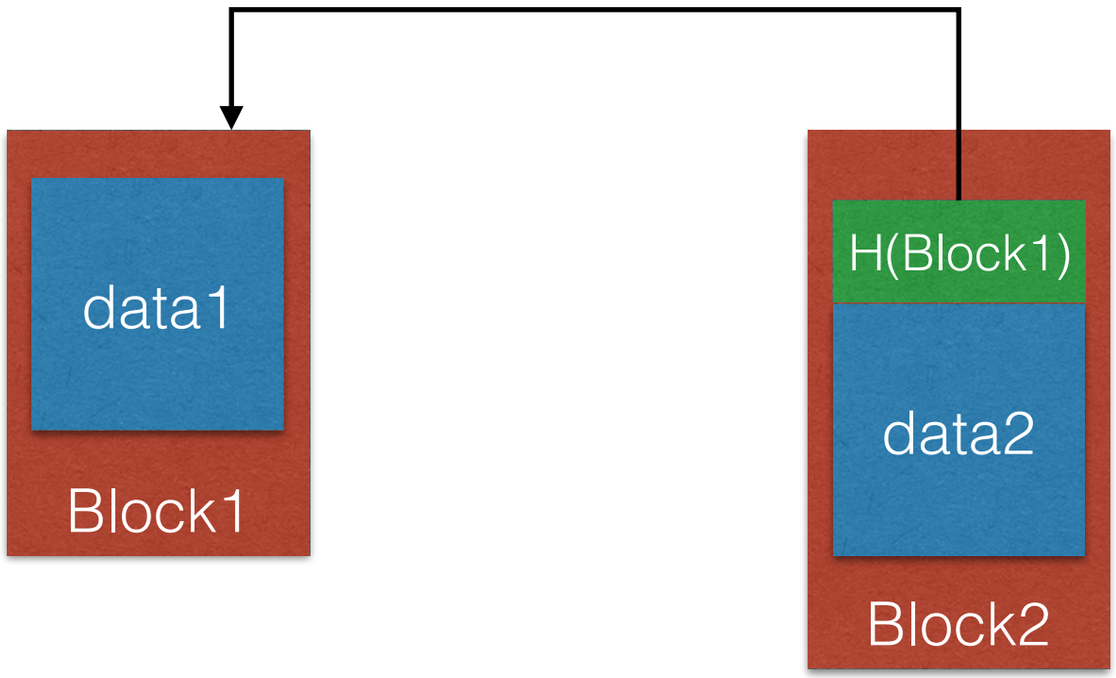
We need to store N hashes to
make sure no data has changed

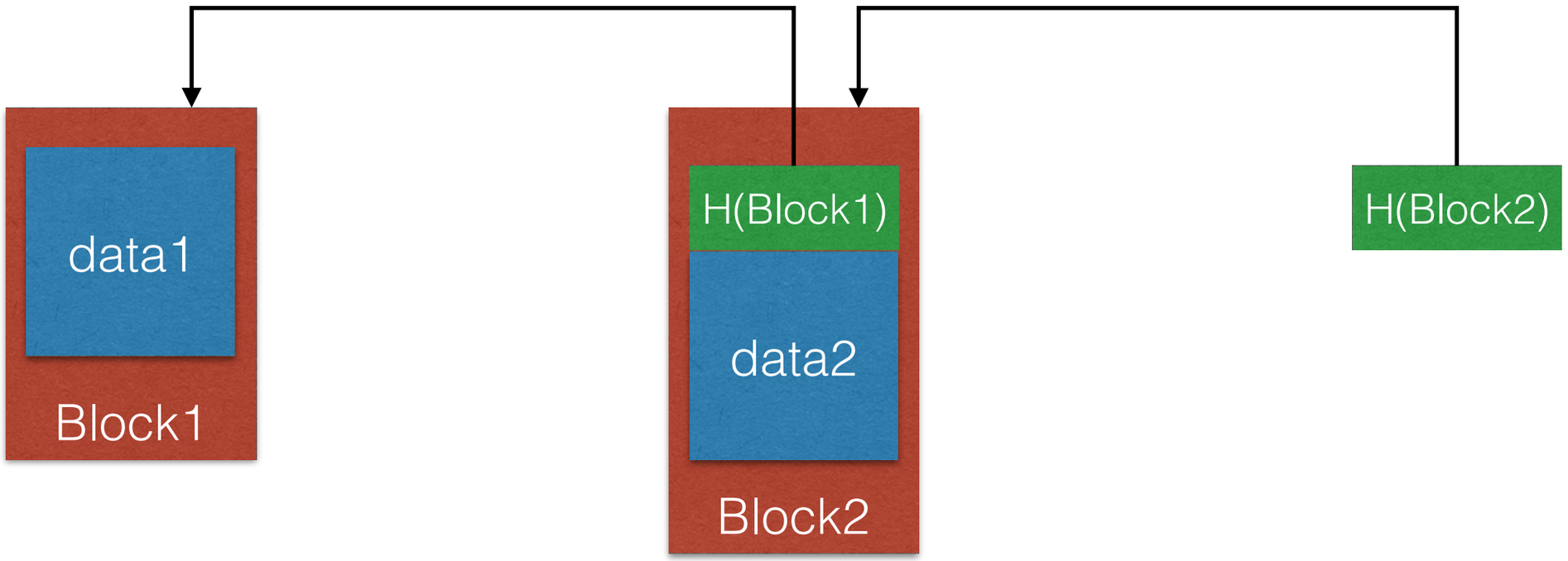
Can we do it by storing just one hash,
but without hashing all data together?

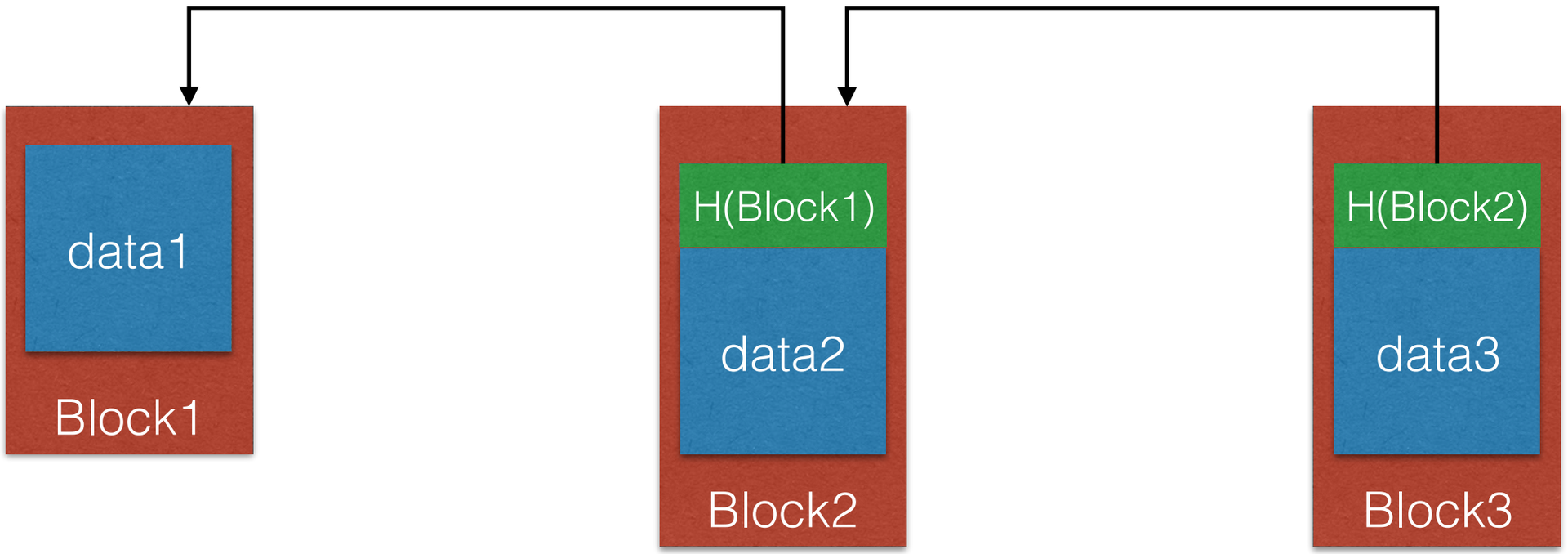
data1

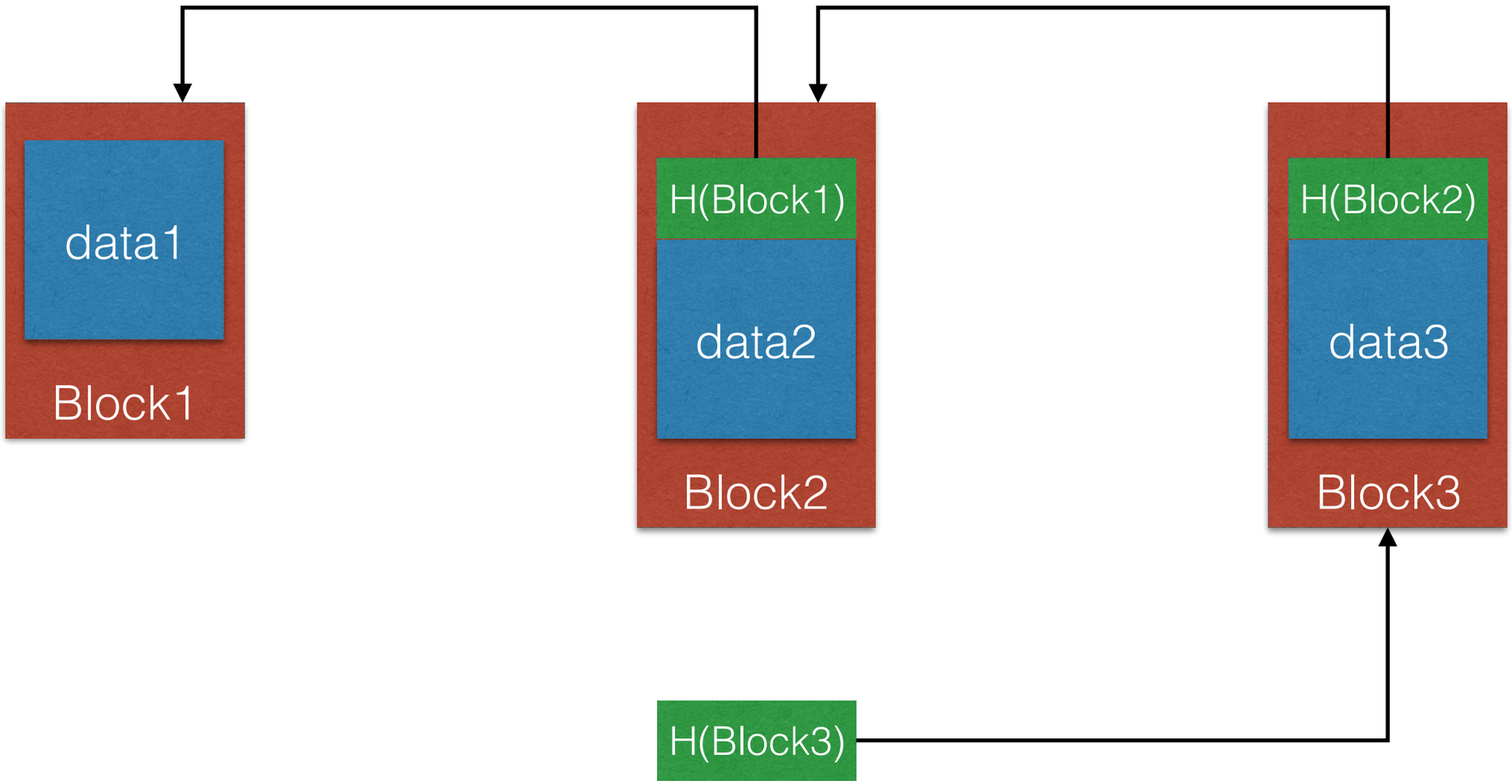

data1
Block1

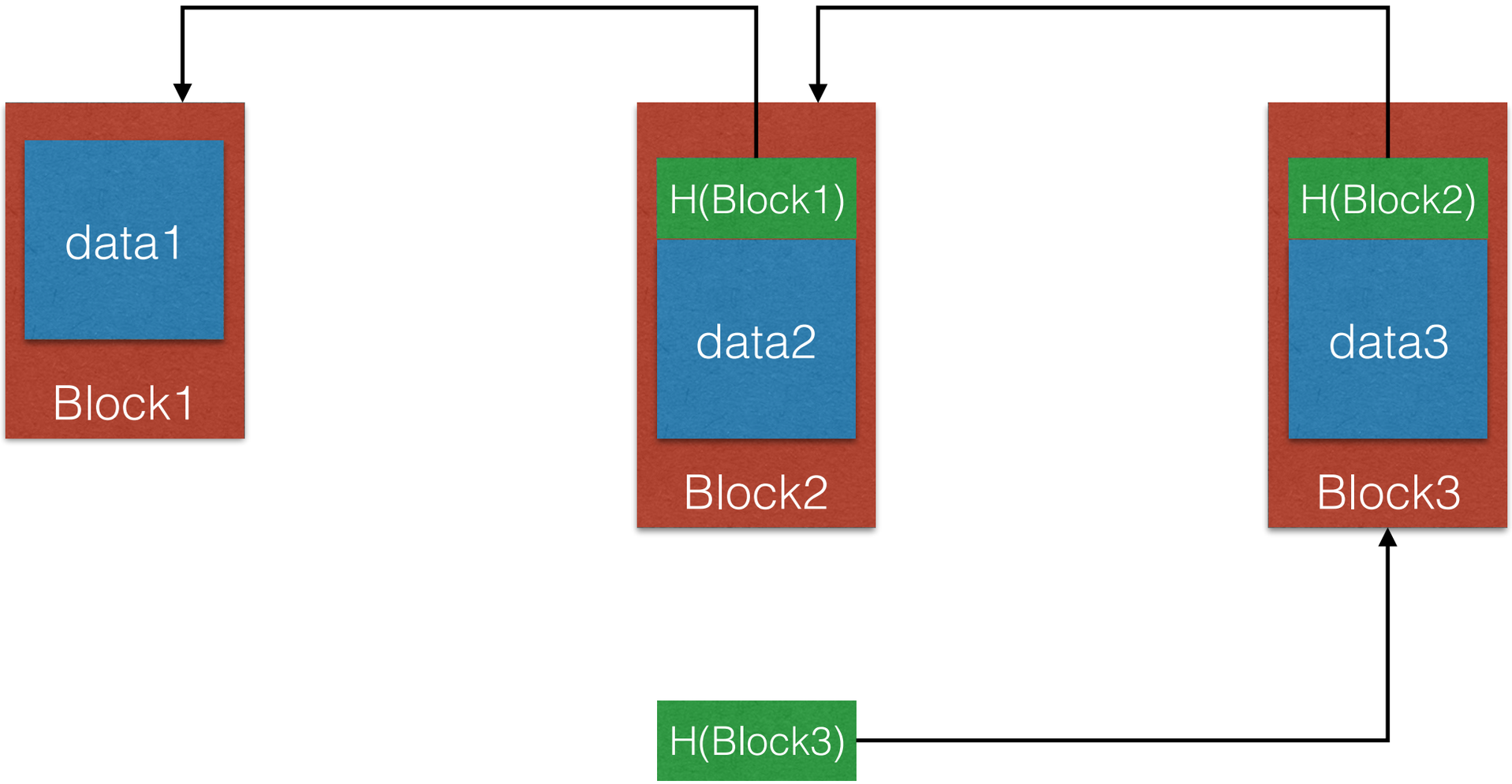

data1
Block1
H(Block1)


data1
Block1
H(Block1)
data2
Block2


data1
Block1
H(Block1)
data2
Block2
H(Block2)

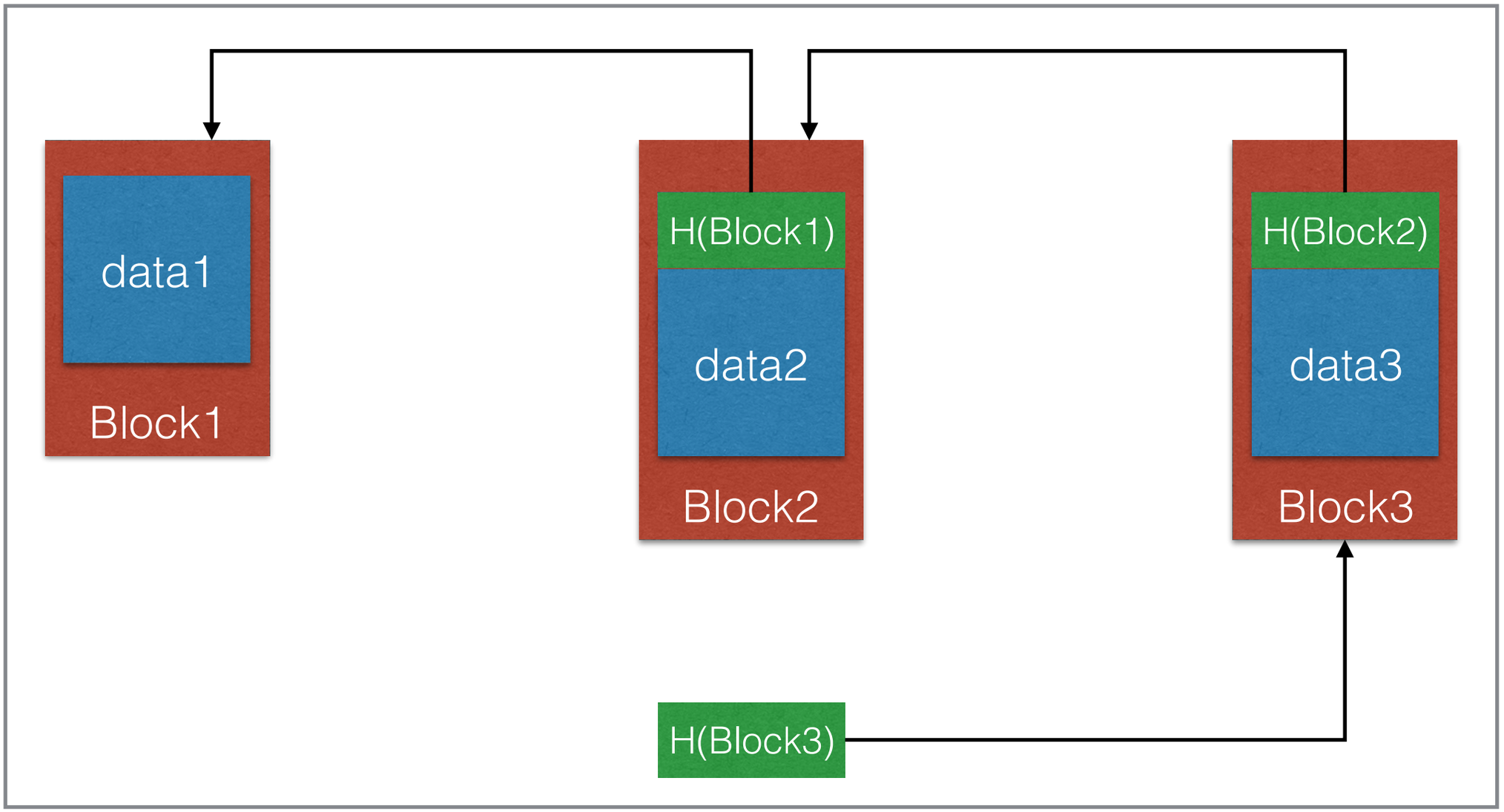

data1
Block1
H(Block1)
data2
Block2
H(Block2)
data3
Block3


data1
Block1
H(Block1)
data2
Block2
H(Block2)
data3
Block3
H(Block3)



If we only store H(Block3), we can later verify that no data has been modified

Blockchain



If we only store H(Block3), we can later verify that no data has been modified

Digital Signatures (impossible to sign documents without secret key)

Cryptographic Hash Functions (irreversible, collision resistance)

Hash Pointers, Blockchain

Questions

A Centralized Cryptocurrency

A Centralized Cryptocurrency

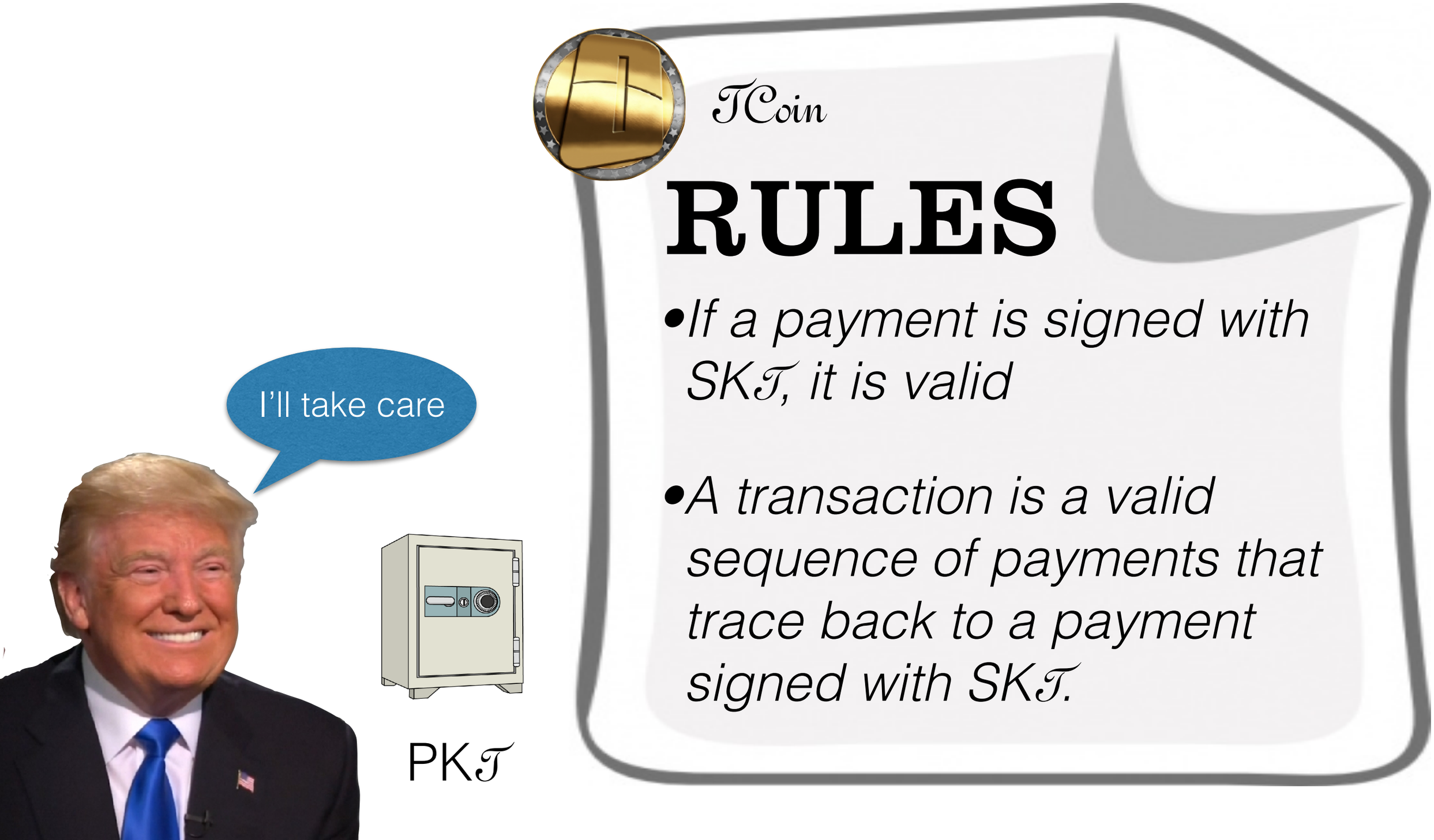
$\mathsf{SK}_{\mathcal{T}}$

$\mathsf{PK}_{\mathcal{T}}$

A Centralized Cryptocurrency

$\mathrm{PK}_{\mathcal{T}}$

A Centralized Cryptocurrency

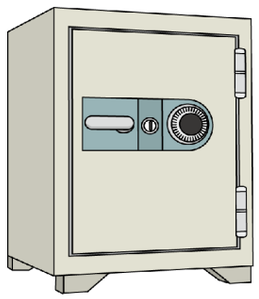


A Centralized Cryptocurrency

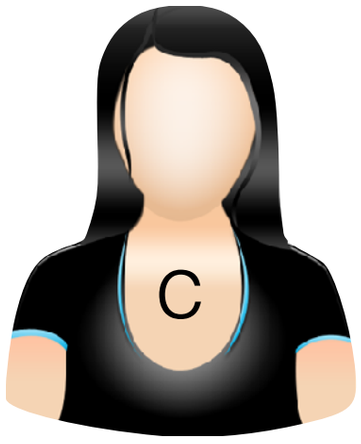
TCoin

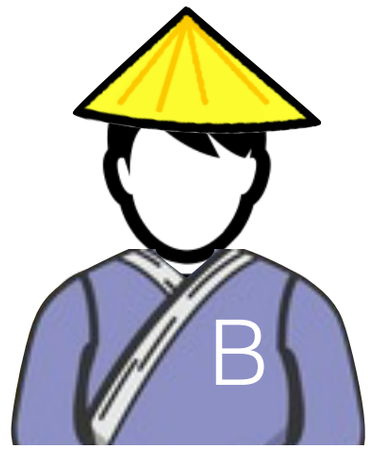
RULES

- *If a payment is signed with $SK_{\mathcal{T}}$, it is valid*
- *A transaction is a valid sequence of payments that trace back to a payment signed with $SK_{\mathcal{T}}$.*

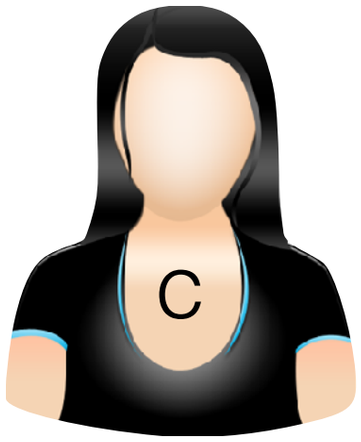


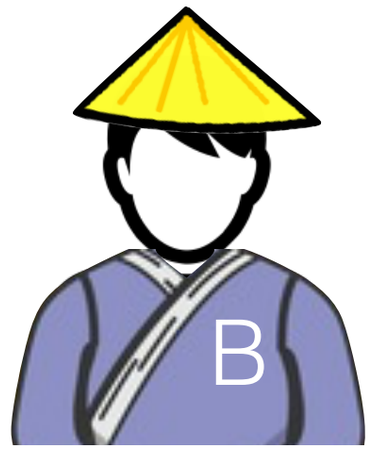
$PK_{\mathcal{T}}$



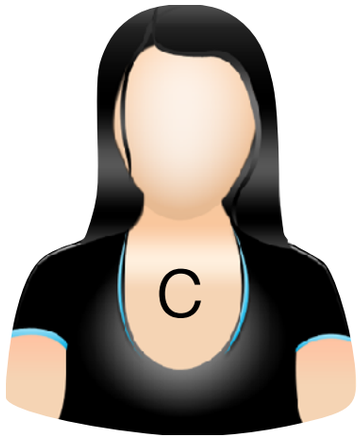


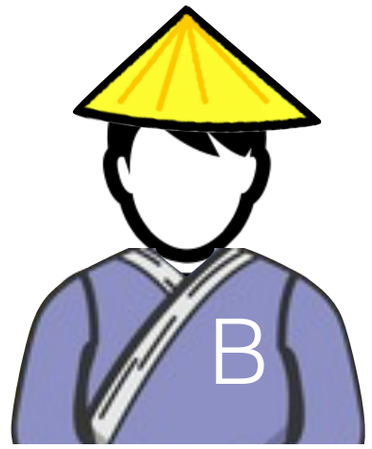
Alice, I want to pay you.
What's your public key?





My public key is PK_A



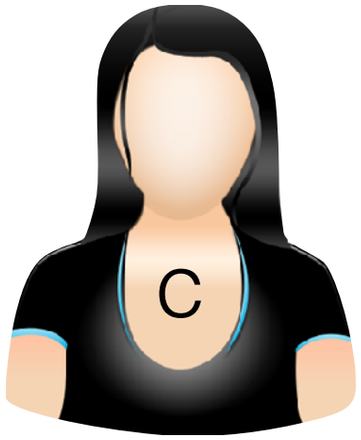


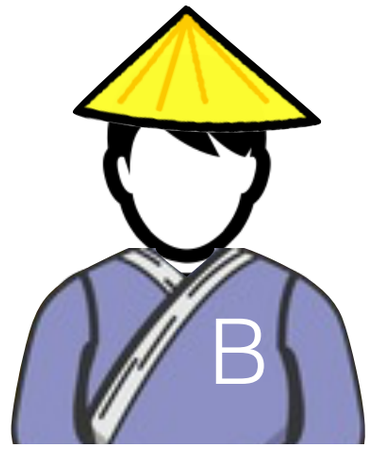
Pay 10 to the owner of the secret key corresponding to PK_A

Signed with $SK_{\mathcal{T}}$

Tx1

My public key is PK_A

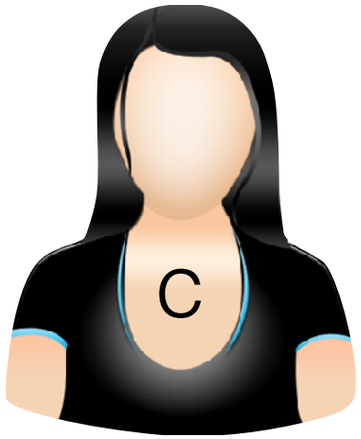


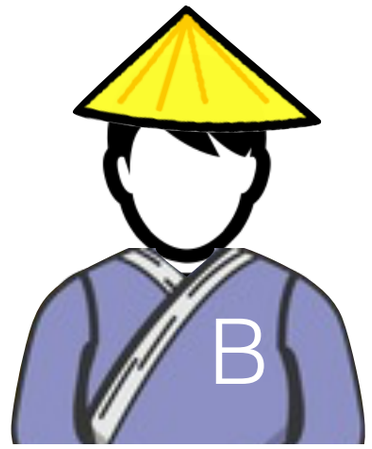


Pay 10 to the owner of the secret key corresponding to PK_A

Signed with $SK_{\mathcal{T}}$

Tx1





Pay 10 to the owner of the secret key corresponding to PK_A

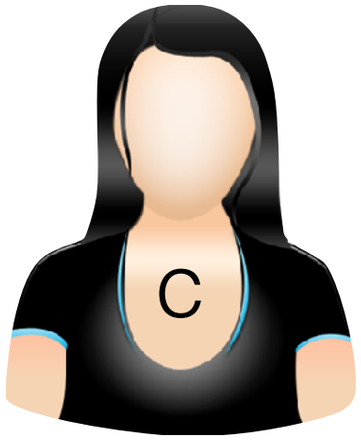
Signed with $SK_{\mathcal{T}}$

Tx1

Input: H(Tx1)
Pay 10 to the owner of the secret key corresponding to PK_B

Signed with SK_A

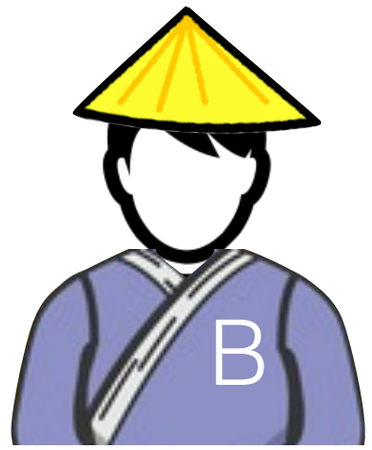
Tx2



Input H(Tx2)
Pay 5 to PK_C and 5 to PK_B.

Signed with SK_B

Tx3



Pay 10 to the owner of the secret key corresponding to PK_A

Signed with $SK_{\mathcal{T}}$

Tx1

Input: H(Tx1)
Pay 10 to the owner of the secret key corresponding to PK_B

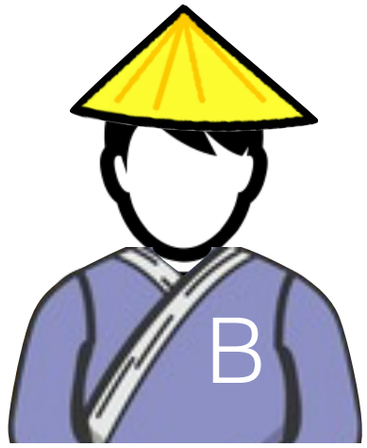
Signed with SK_A

Tx2

Input H(Tx2)
Pay 5 to PK_C and 5 to PK_B.

Signed with SK_B

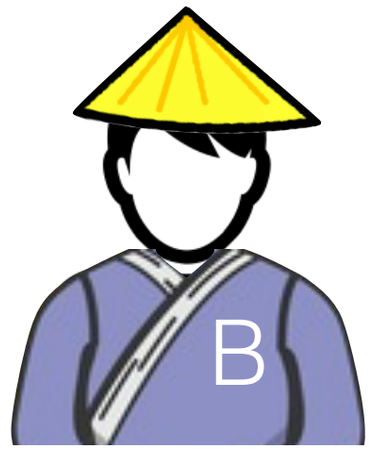
Tx3



Input H(Tx2)
Pay 5 to PK_C and 5 to PK_B.

Signed with SK_B

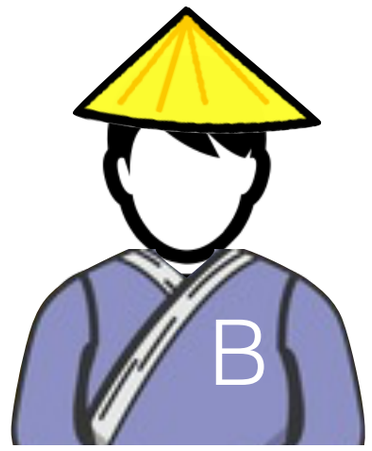
Tx3



Input H(Tx2)
Pay 5 to PK_C and 5 to PK_B.

Signed with SK_B

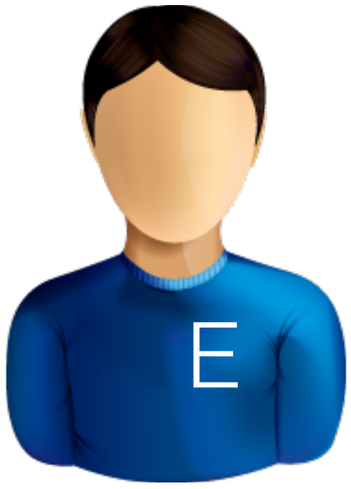
Tx3



Input H(Tx3)
Pay 5 to PK_D

Signed with SK_B

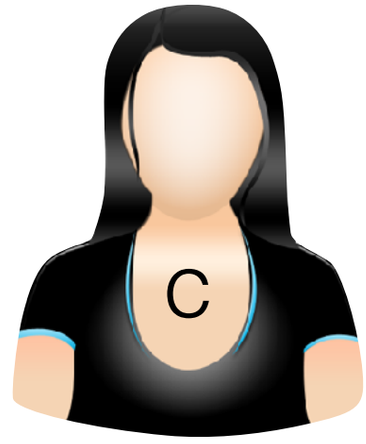
Tx4

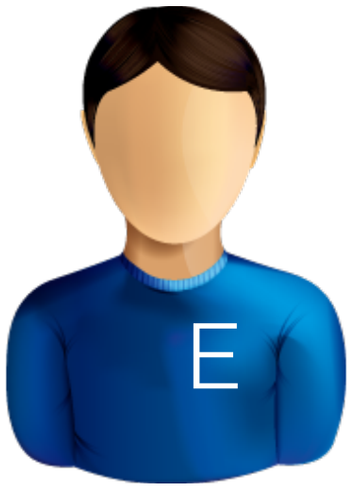


Input H(Tx2)
Pay 5 to PK_C and 5 to PK_B.

Signed with SK_B

Tx3

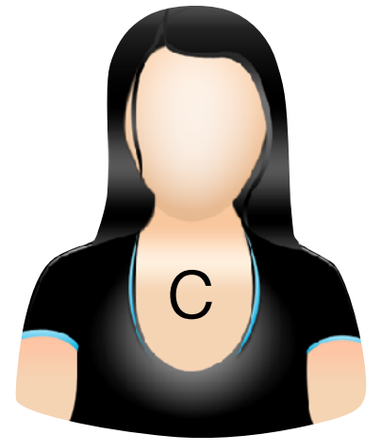




Input H(Tx2)
Pay 5 to PK_C and 5 to PK_B.

Signed with SK_B

Tx3



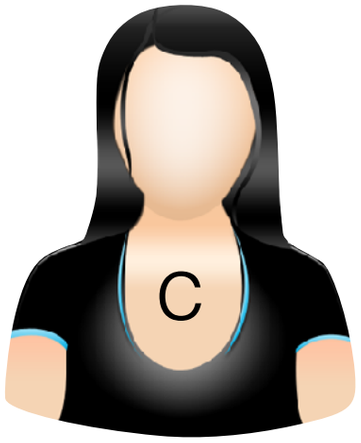
Input H(Tx3)
Pay 5 to PK_E

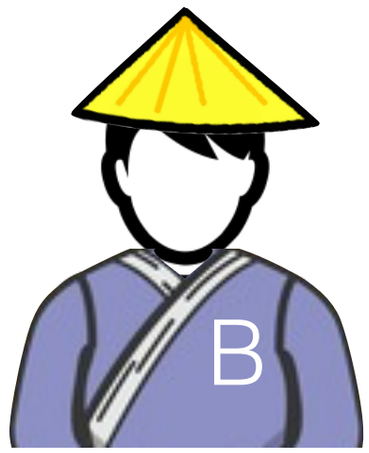
Signed with SK_C

Tx5

We can now “split” transactions

Can we merge them?





Pay 10 to PK_B

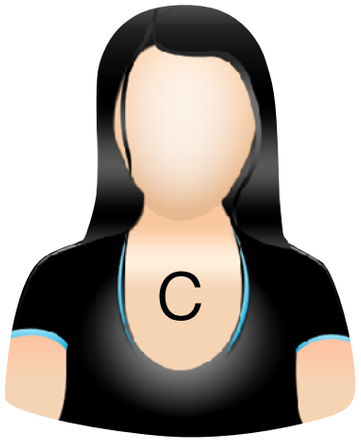
Signed with $SK_{\mathcal{T}}$

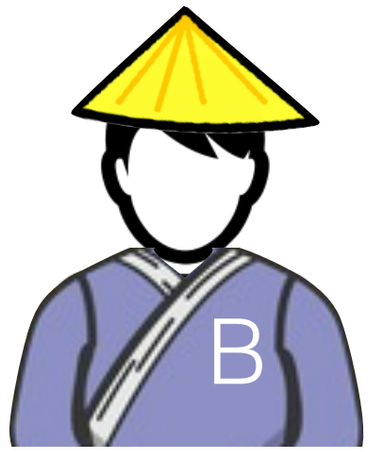
Tx1

Pay 5 to PK_B

Signed with $SK_{\mathcal{T}}$

Tx2





Input H(Tx1), H(Tx2)
Pay 15 to PK_C

Signed with SK_B

Tx3

Pay 10 to PK_B

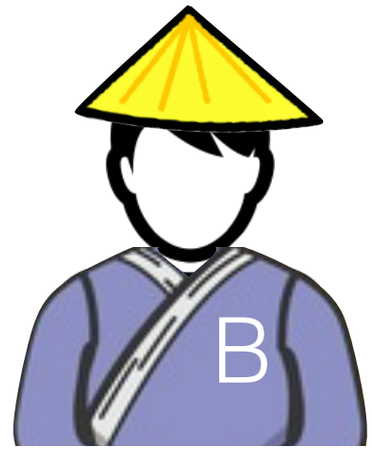
Signed with $SK_{\mathcal{T}}$

Tx1

Pay 5 to PK_B

Signed with $SK_{\mathcal{T}}$

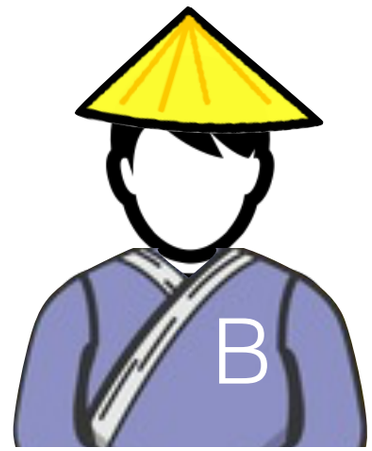
Tx2



Pay 10 to the owner of the secret key corresponding to PK_A

Signed with $SK_{\mathcal{T}}$

Tx1



Input: H(Tx1)
Pay 10 to the owner of the secret key corresponding to PK_B

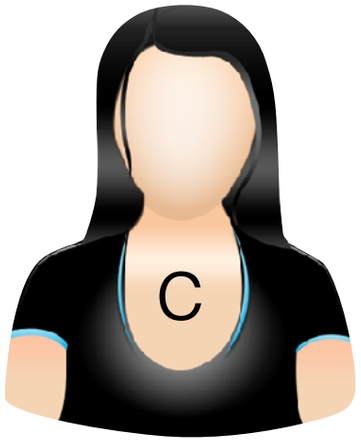
Signed with SK_A

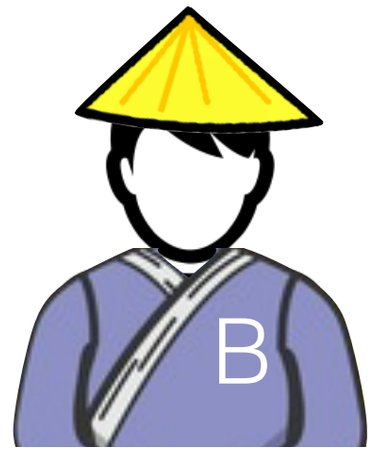
Tx2

Pay 10 to the owner of the secret key corresponding to PK_A

Signed with $SK_{\mathcal{T}}$

Tx1





Input: H(Tx1)
Pay 10 to the owner of the secret key corresponding to PK_B

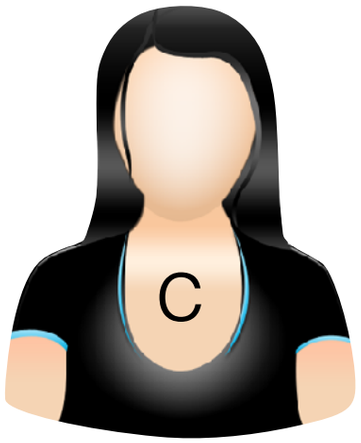
Signed with SK_A

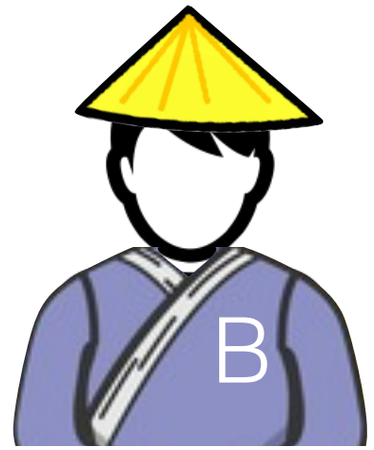
Tx2

Pay 10 to the owner of the secret key corresponding to PK_A

Signed with $SK_{\mathcal{T}}$

Tx1





Input: H(Tx1)
Pay 10 to the owner of the secret key corresponding to PK_C

Signed with SK_A

Tx3

Input: H(Tx1)
Pay 10 to the owner of the secret key corresponding to PK_B

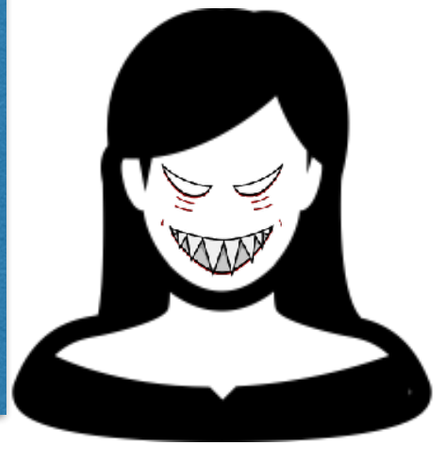
Signed with SK_A

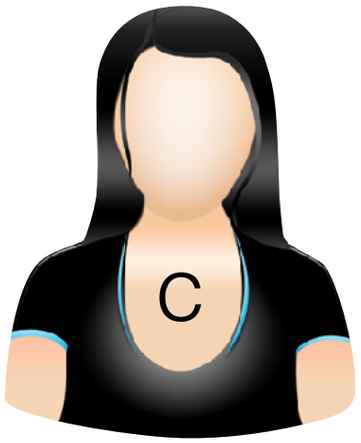
Tx2

Pay 10 to the owner of the secret key corresponding to PK_A

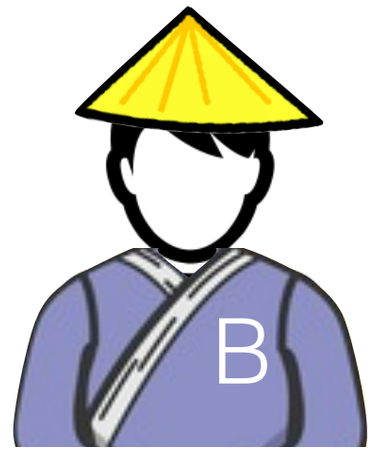
Signed with $SK_{\mathcal{T}}$

Tx1





I don't trust
this system anymore,
I'm out



Input: H(Tx1)
Pay 10 to the owner of the secret key corresponding to PK_C

Signed with SK_A

Tx3

Input: H(Tx1)
Pay 10 to the owner of the secret key corresponding to PK_B

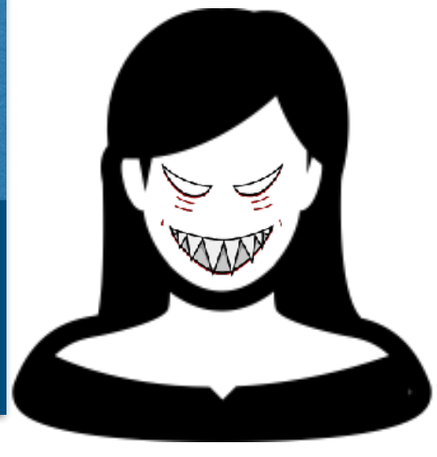
Signed with SK_A

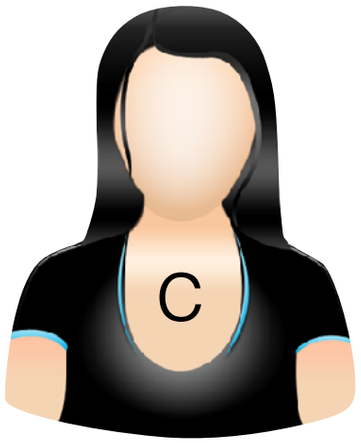
Tx2

Pay 10 to the owner of the secret key corresponding to PK_A

Signed with $SK_{\mathcal{T}}$

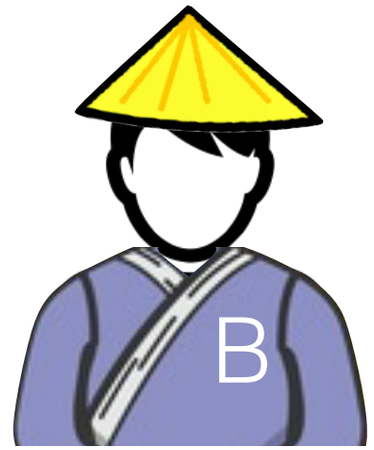
Tx1





I don't trust
this system anymore,
I'm out

We need
someone really
trustworthy



Input: H(Tx1)
Pay 10 to the owner of the secret key corresponding to PK_C

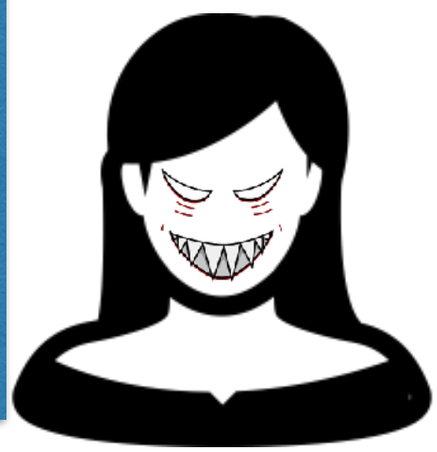
Signed with SK_A

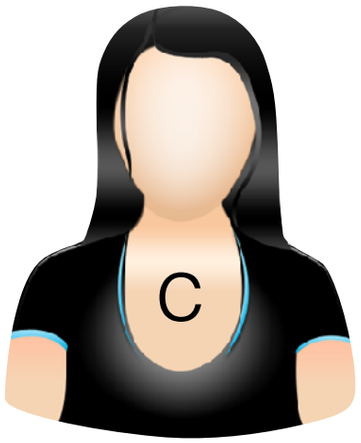
Input: H(Tx1)
Pay 10 to the owner of the secret key corresponding to PK_B

Signed with SK_A

Tx2

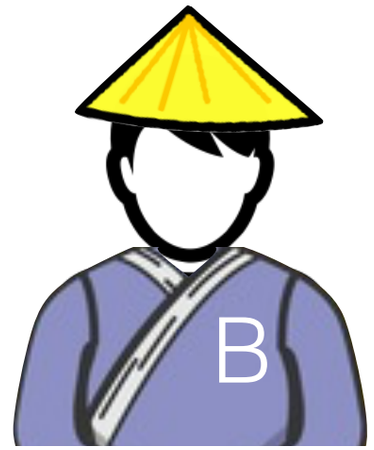
Pay 10 [illegible] he secret
key [illegible] o PK_A





We need someone really trustworthy

I don't trust this system anymore, I'm out



Input: H(Tx1)
Pay 10 to the owner of the secret key corresponding to PK_C

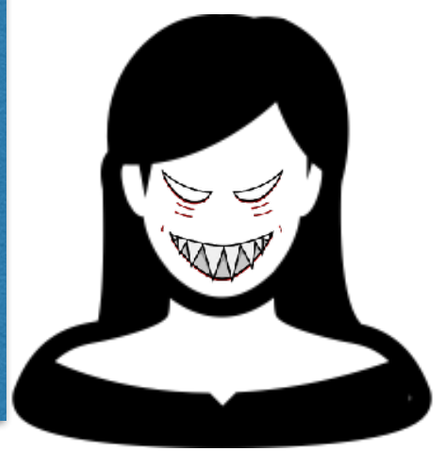
Signed with SK_A

Input: H(Tx1)
Pay 10 to the owner of the secret key corresponding to PK_B

Signed with SK_A

Pay 10 [illegible] he secret key [illegible] o PK_A

I'll take care of this



TCoin 2.0

RULES

- *If a payment is signed with PK$\mathcal{T}$, it is valid.*
- *A transaction is a sequence of valid payments which trace back to a payment signed with PK$\mathcal{T}$.*
- *Valid transactions are recorded in a public ledger signed periodically with PK$\mathcal{T}$.*

Pay 10 to the owner of the secret key corresponding to PK_A

Tx1 Signed with $SK_{\mathcal{T}}$

Pay 10 to the owner of the secret key corresponding to PK_B

Tx2 Signed with $SK_{\mathcal{T}}$

Pay 10 to the owner of the secret key corresponding to PK_A

Tx1 Signed with $SK_{\mathcal{T}}$

Pay 10 to the owner of the secret key corresponding to PK_B

Tx2 Signed with $SK_{\mathcal{T}}$

Block 1, Signed with $SK_{\mathcal{T}}$

Pay 10 to the owner of the secret key corresponding to PK_A

Tx1 Signed with $SK_{\mathcal{T}}$

Pay 10 to the owner of the secret key corresponding to PK_B

Tx2 Signed with $SK_{\mathcal{T}}$

Block 1, Signed with $SK_{\mathcal{T}}$

Input: H(Tx1)
Pay 10 to PK_B

Tx3 Signed with SK_A

. . .

Pay 10 to the owner of the secret key corresponding to PK_A

Tx1 Signed with $SK_{\mathcal{T}}$

Pay 10 to the owner of the secret key corresponding to PK_B

Tx2 Signed with $SK_{\mathcal{T}}$

Block 1, Signed with $SK_{\mathcal{T}}$

Input: H(Tx1)
Pay 10 to PK_B

Tx3 Signed with SK_A

...

H(Block 1)

Pay 10 to the owner of the secret key corresponding to PK_A

Tx1 Signed with $SK_{\mathcal{T}}$

Pay 10 to the owner of the secret key corresponding to PK_B

Tx2 Signed with $SK_{\mathcal{T}}$

Block 1, Signed with $SK_{\mathcal{T}}$

Input: H(Tx1)
Pay 10 to PK_B

Tx3 Signed with SK_A

. . .

H(Block 1)

Block 2, Signed with $SK_{\mathcal{T}}$

Pay 10 to the owner of the secret key corresponding to PK_A

Tx1 Signed with $SK_{\mathcal{T}}$

Pay 10 to the owner of the secret key corresponding to PK_B

Tx2 Signed with $SK_{\mathcal{T}}$

Block 1, Signed with $SK_{\mathcal{T}}$

Input: H(Tx1)
Pay 10 to PK_B

Tx3 Signed with SK_A

. . .

H(Block 1)

Block 2, Signed with $SK_{\mathcal{T}}$

Pay 10 to the owner of the secret key corresponding to PK_A

Tx1 Signed with $SK_{\mathcal{T}}$

Pay 10 to the owner of the secret key corresponding to PK_B

Tx2 Signed with $SK_{\mathcal{T}}$

Block 1, Signed with $SK_{\mathcal{T}}$

Input: H(Tx1)
Pay 10 to PK_B

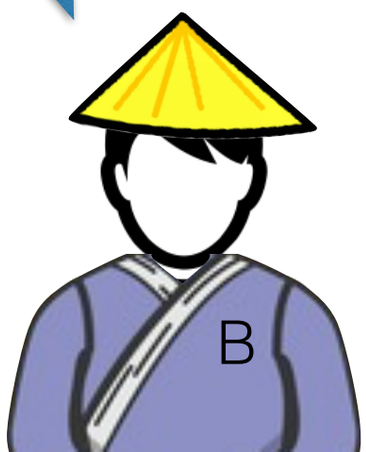
Tx3 Signed with SK_A

. . .

H(Block 1)

Block 2, Signed with $SK_{\mathcal{T}}$

Cool, now I have 20 TCoins!



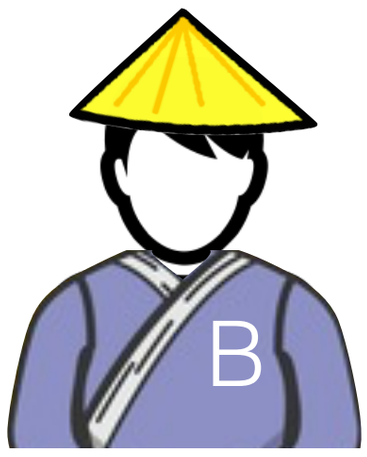
Don't worry, you can check I never include double spends

And we are all very happy (specially me)

Hey Theresa, why is C getting that amount of money?!

Uhm… I just like her a lot

Hey Theresa, why is C getting that amount of money?!

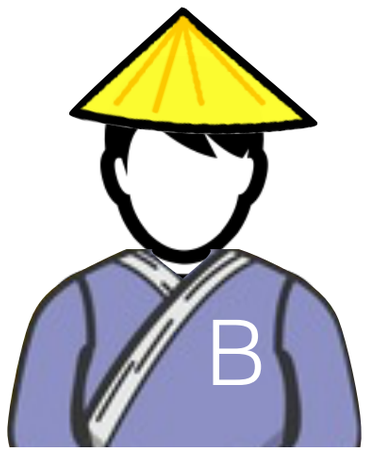


There's no new block since last week!

Sorry guys, I was on vacation

Uhm… I just like her a lot

Hey Theresa, why is C getting that amount of money?!



There's no new block since last week!

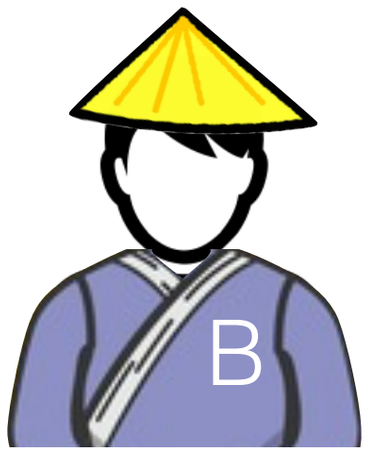
Sorry guys, I was on vacation

Uhm… I just like her a lot

Just a week?! my transactions haven't been confirmed in over a month!

It's because I don't like you :)

Hey Theresa, why is C getting that amount of money?!



There's no new block since last week!

Hey people, bad news, someone stole my private key

Uhm… I just like her a lot

Sorry guys, I was on vacation

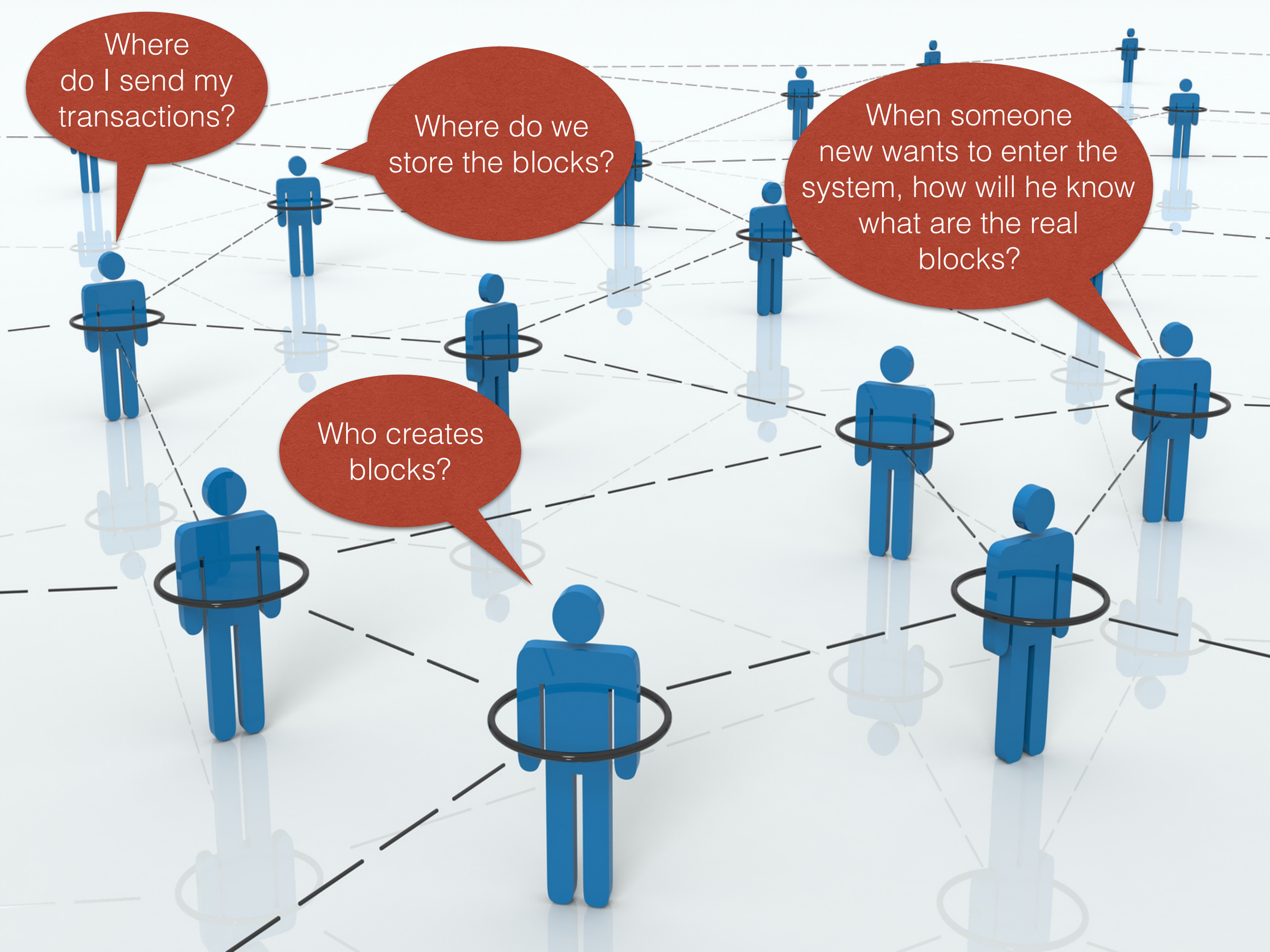
Just a week?! my transactions haven't been confirmed in over a month!

It's because I don't like you :)

Bitcoin: A Peer-to-Peer Electronic Cash System

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.


Where do I send my transactions?
Where do we store the blocks?
When someone new wants to enter the system, how will he know what are the real blocks?
Who creates blocks?

Where do I send my transactions?

Where do we store the blocks?

When someone new wants to enter the system, how will he know what are the real blocks?

Who creates blocks?

Let me help

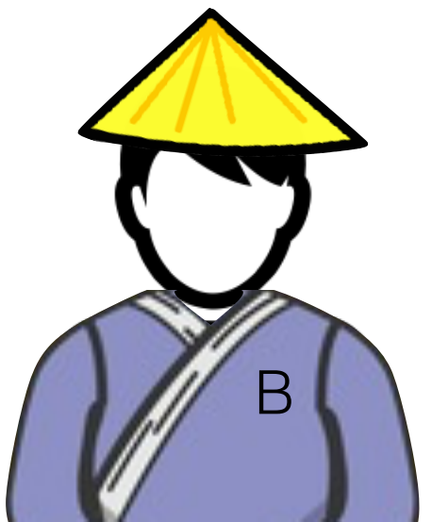
RULES

- *Each node should store all blocks, no trust!*
- *If you want to pay, just tell your neighbors*
- *If you receive a new transaction, broadcast it to your neighbors*
- *…*

RULES

- *Each node should store all blocks, no trust!*
- *you want to pay, just tell your neighbors*
- *If you receive a new transaction, broadcast it to your neighbors*
- *…*

How do I know when a transaction is valid? How are blocks generated?

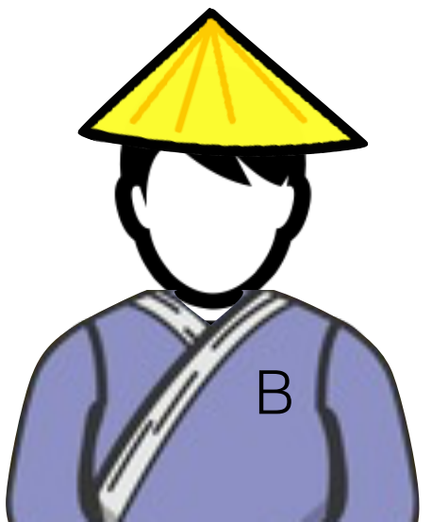


RULES

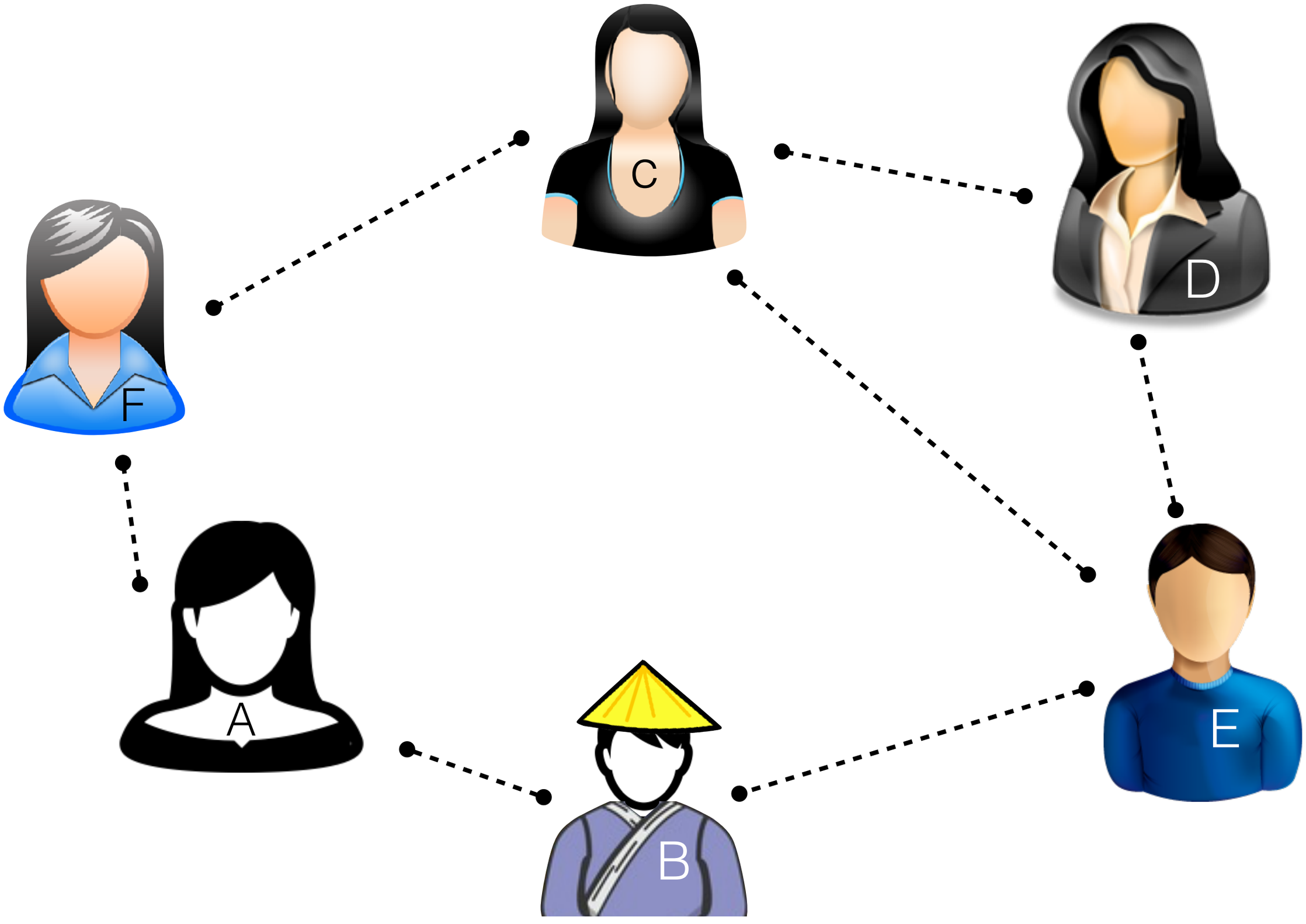
- *Each node should store all blocks, no trust!*
- *If you want to pay, just tell your neighbors*
- *If you receive a new transaction, broadcast it to your neighbors*
- *…*

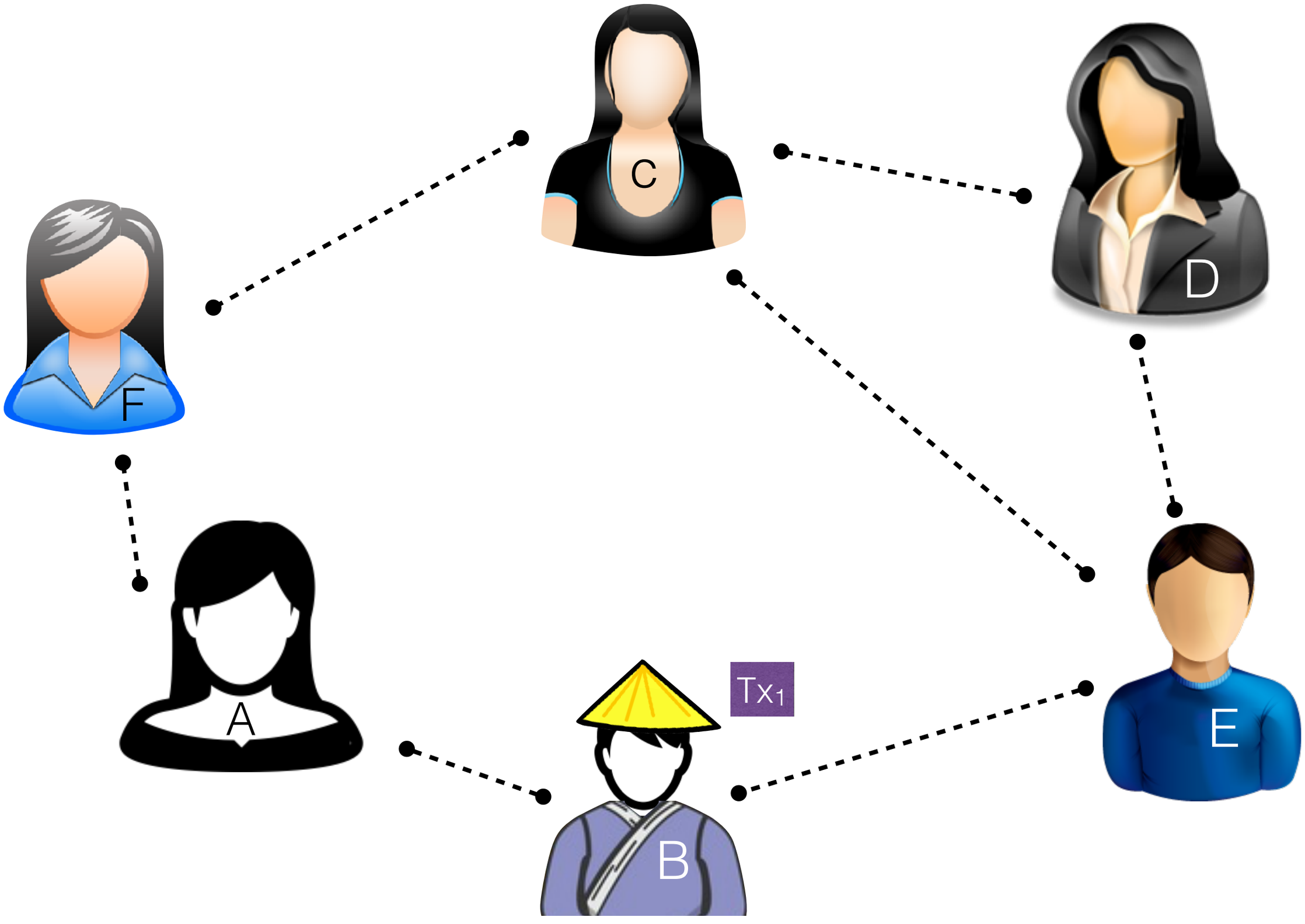
More on that later.

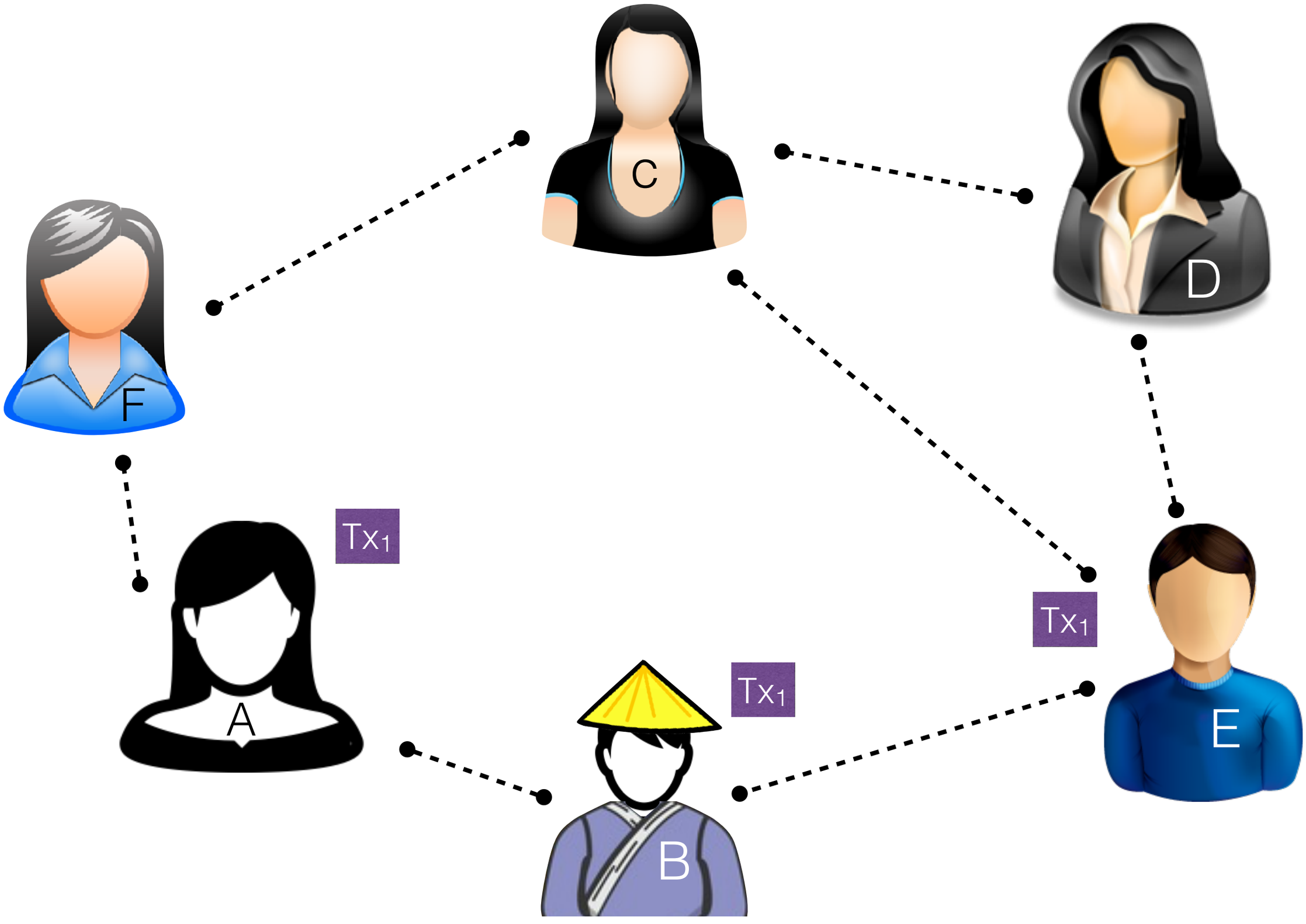
How do I know when a transaction is valid? How are blocks generated?

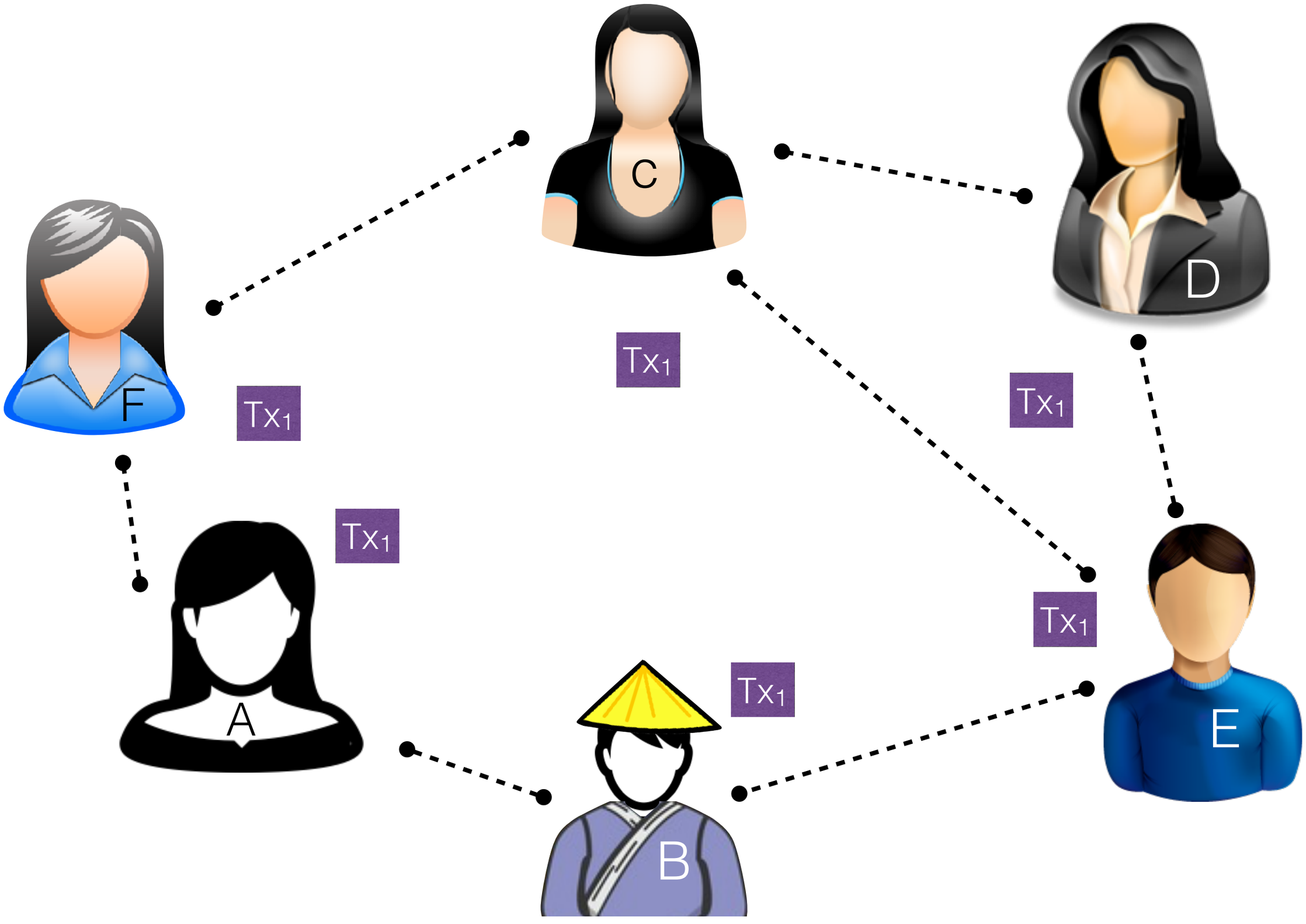


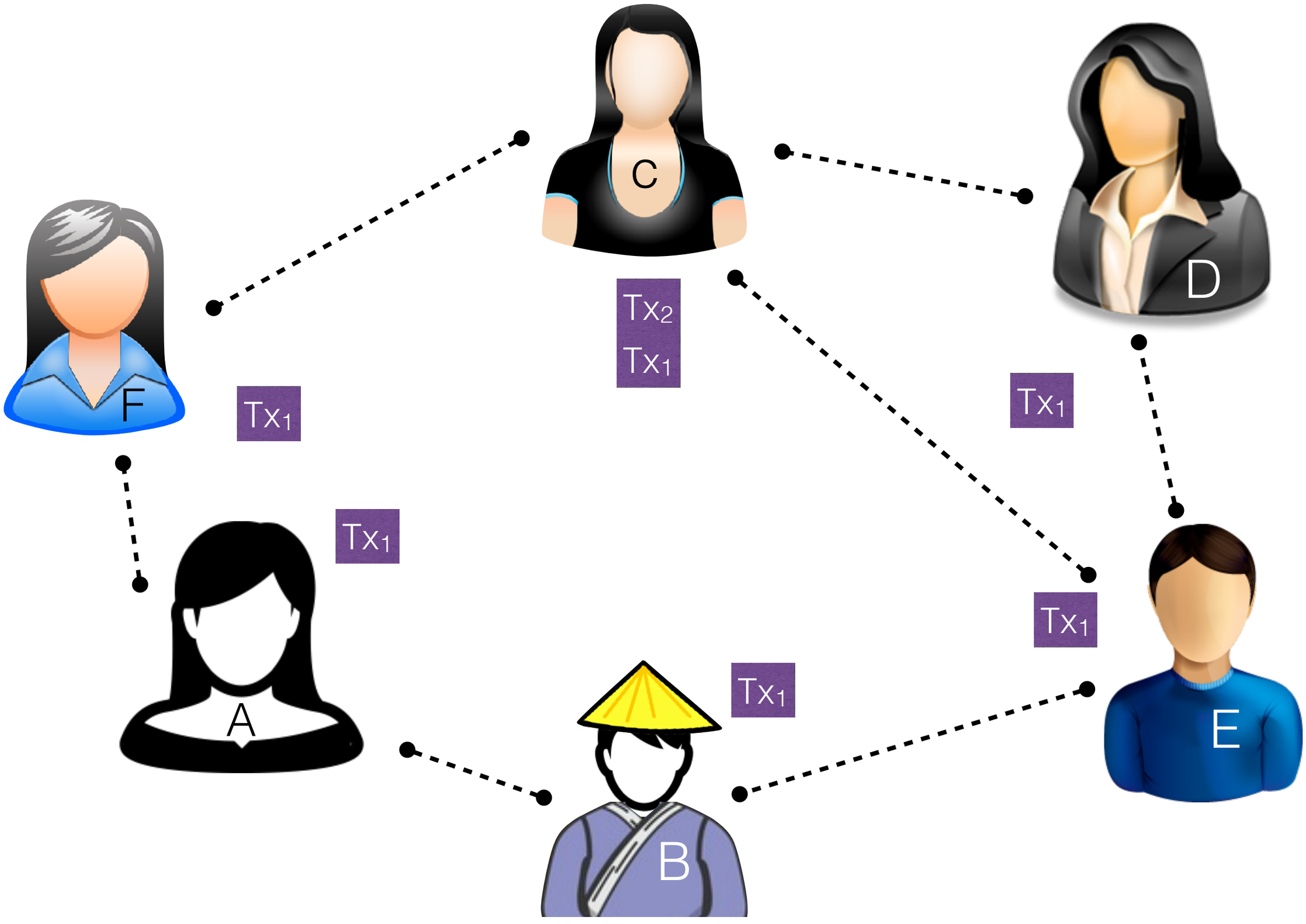
Let us assume for the moment that Satoshi can arbitrarily generate coins, like in TCoin.

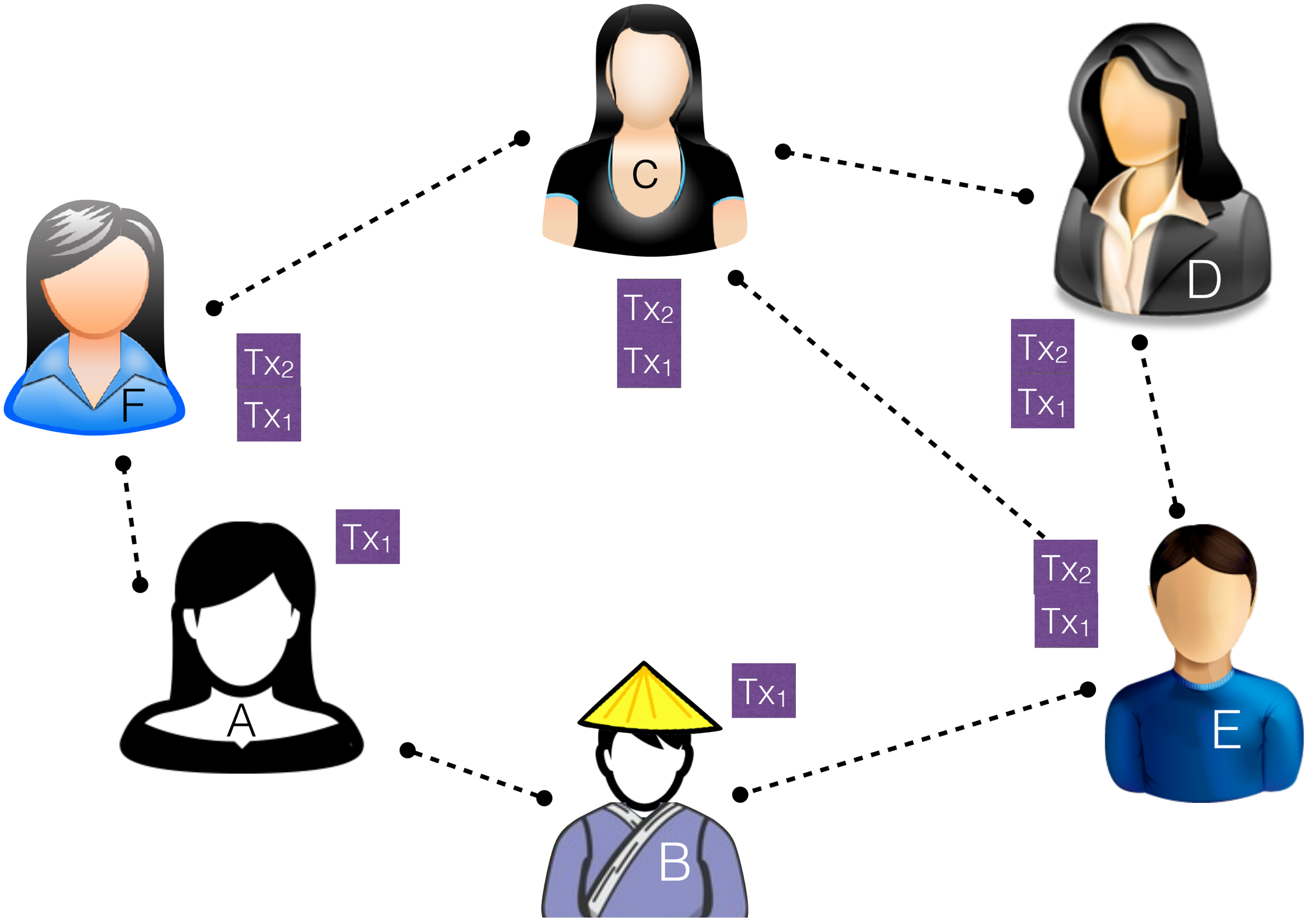

C
D
F
A
B
E

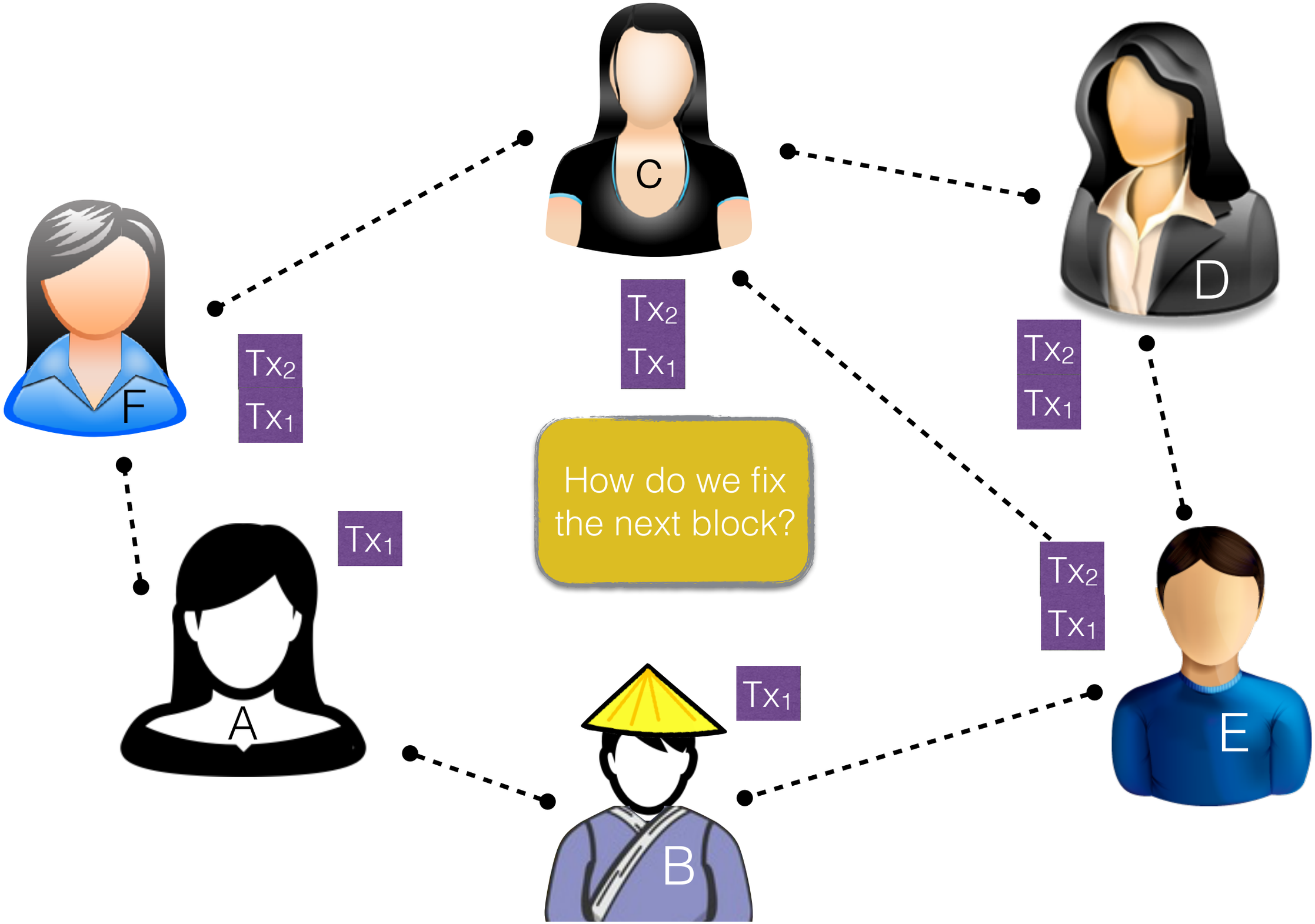

C
D
F
A
B
Tx1
E


C
D
F
Tx1
A
Tx1
B
Tx1
Tx1
E


C
D
F
Tx1
Tx1
Tx1
A
Tx1
Tx1
B
E

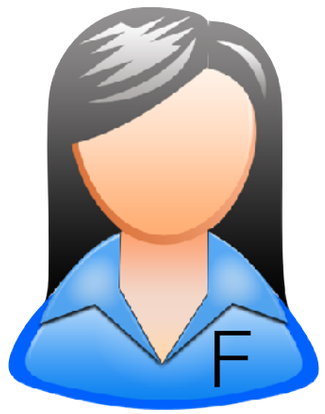

C
D
F
Tx2
Tx1
Tx1
Tx1
Tx1
Tx1
A
B
E

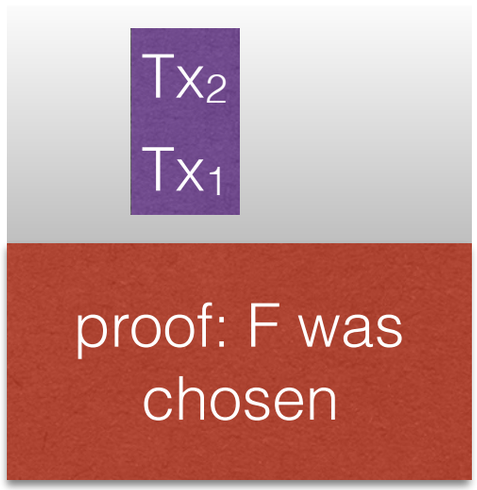

F
Tx2
Tx1
C
Tx2
Tx1
D
Tx2
Tx1
Tx2
Tx1
A
Tx1
B
Tx1
E


C
D
F
Tx2
Tx1
Tx2
Tx1
Tx2
Tx1
How do we fix the next block?
Tx1
Tx2
Tx1
A
Tx1
B
E

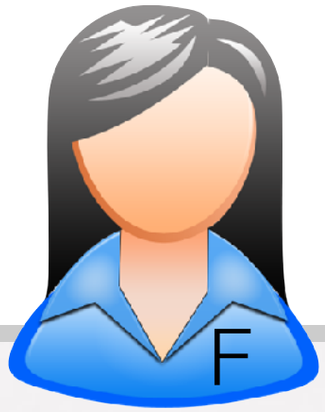
Assume that participants are randomly chosen to generate the next block

Assume that participants are randomly chosen to generate the next block





Assume that participants are randomly chosen to generate the next block



Tx_2

Tx_1

proof: F was chosen

- *…*
- *When you form a block, you must broadcast it*
- *If you receive a block, you must check that it is correct and broadcast it*

Assume that participants are randomly chosen to generate the next block

Why should I do this correctly?

Tx_2

Tx_1

proof: F was chosen

- …
- *When you form a block, you must broadcast it*
- *If you receive a block, you must check that it is correct and broadcast it*

Assume that participants are randomly chosen to generate the next block

Why should I do this correctly?

Tx_2

Tx_1

proof: F was chosen

- …
- *When you form a block, you must broadcast it*
- *If you receive a block, you must check that it is correct and broadcast it*

Mhh.. If only there was a currency that we could use as incentive

Assume that participants are randomly chosen to generate the next block

Why should I do this correctly?

F

Tx_2 Tx_1 Tx_3

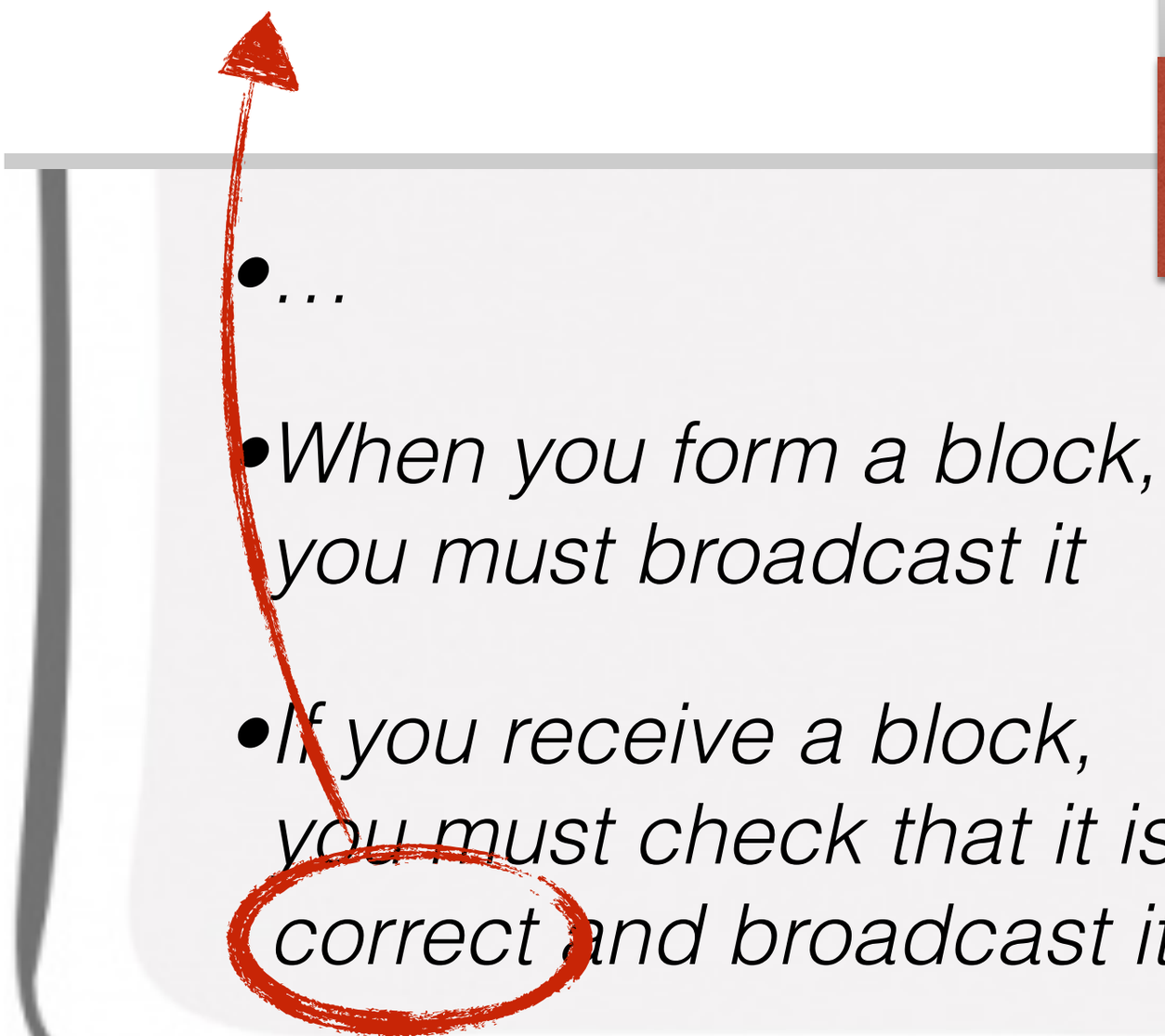
proof: F was chosen

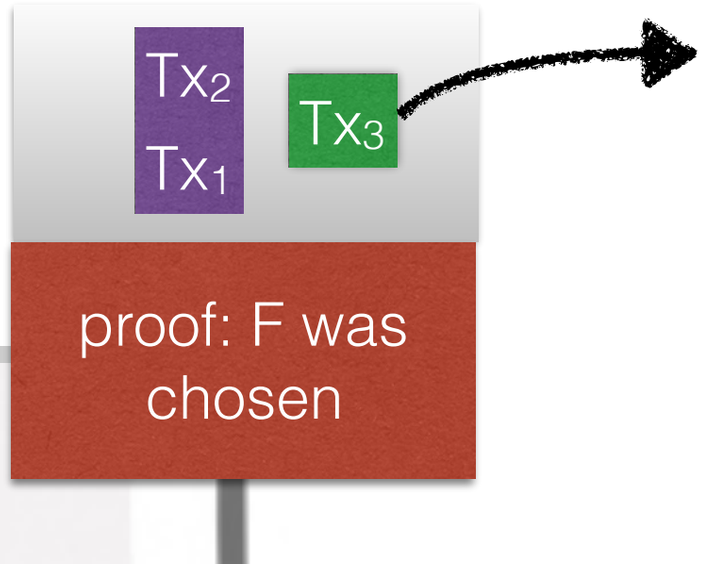
Coinbase transaction (~Fixed)

- …
- *When you form a block, you must broadcast it*
- *If you receive a block, you must check that it is correct and broadcast it*

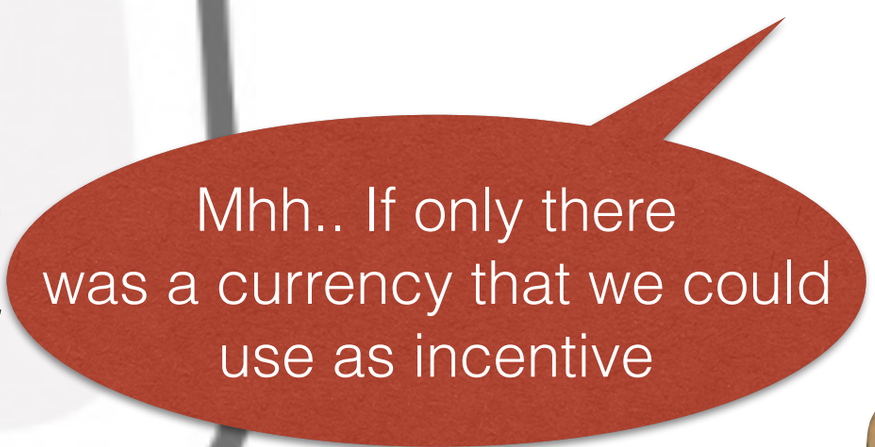
Mhh.. If only there was a currency that we could use as incentive

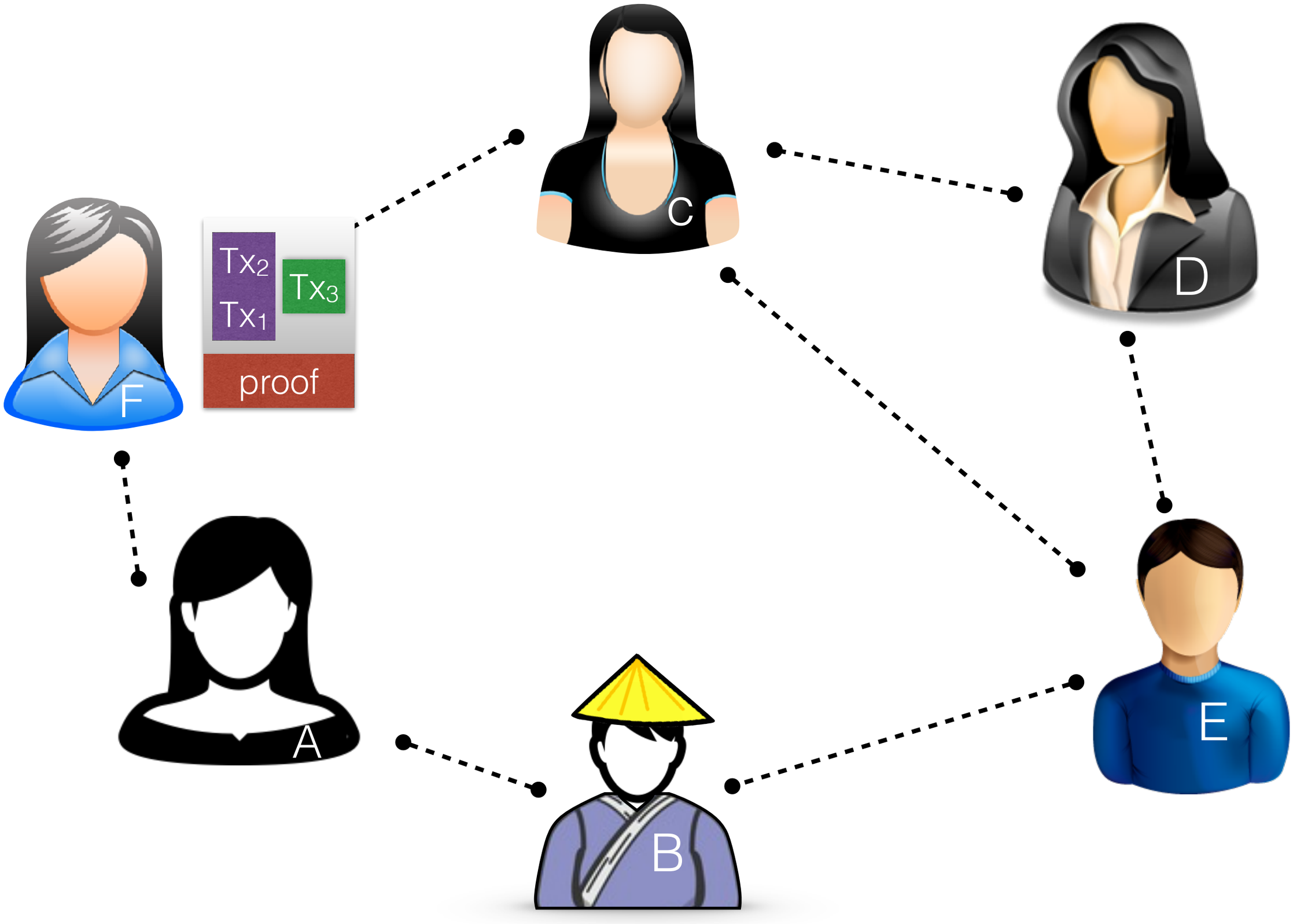
- The proof is correct
- All transactions are correctly signed
- The amounts are correct
- No double spends occur

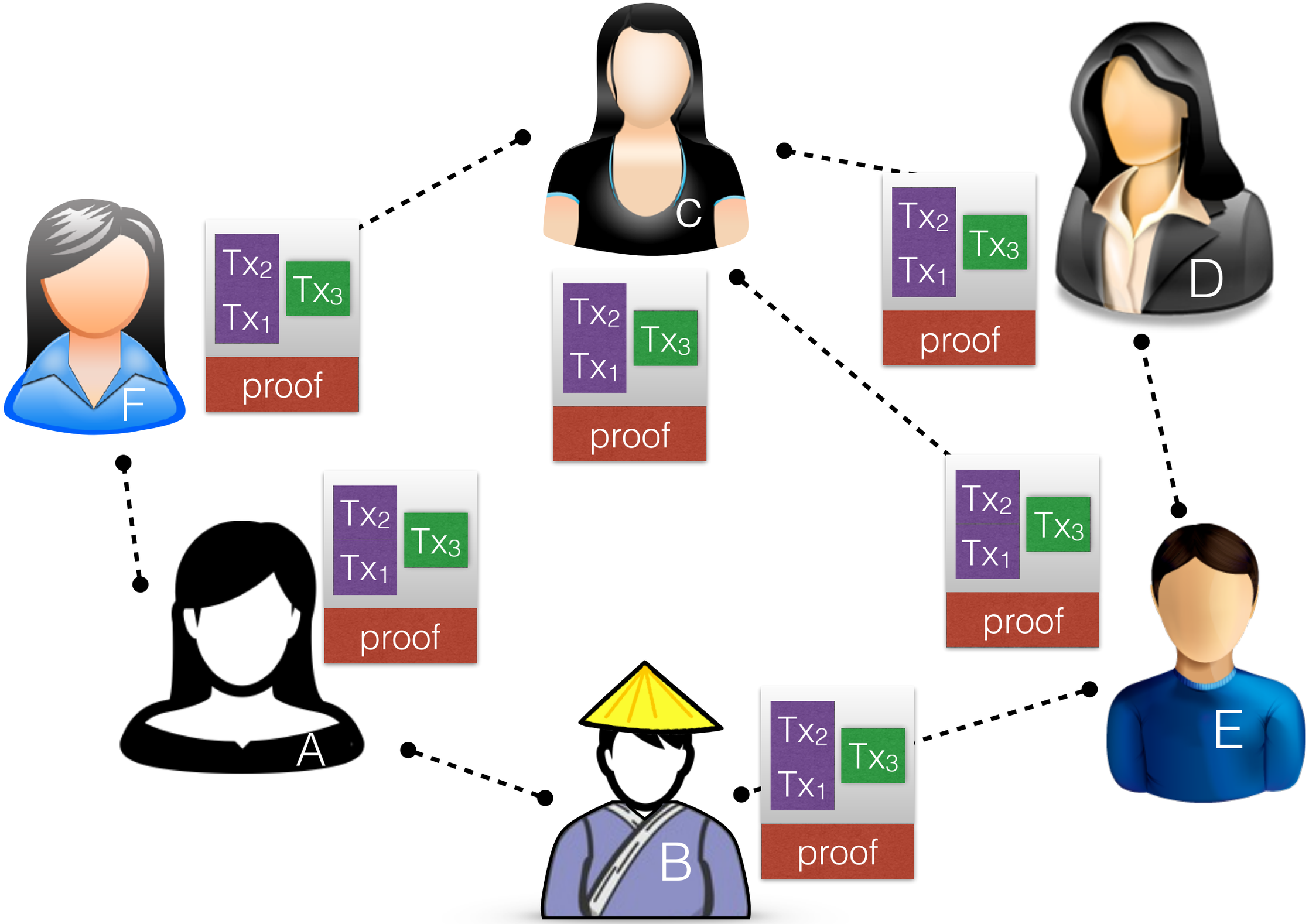




Coinbase transaction (~Fixed)




C
D
F
Tx2
Tx1
Tx3
proof
A
B
E


C
D
F
A
B
E
Tx2
Tx1
Tx3
proof

What's my motivation to check the correctness of the block?

C

Tx_2 Tx_1 Tx_3 proof

F

Tx_2 Tx_1 Tx_3 proof

D

I never generate blocks :(

Tx_2 Tx_1 Tx_3 proof

A

Tx_2 Tx_1 Tx_3 proof

Tx_2 Tx_1 Tx_3 proof

B

E

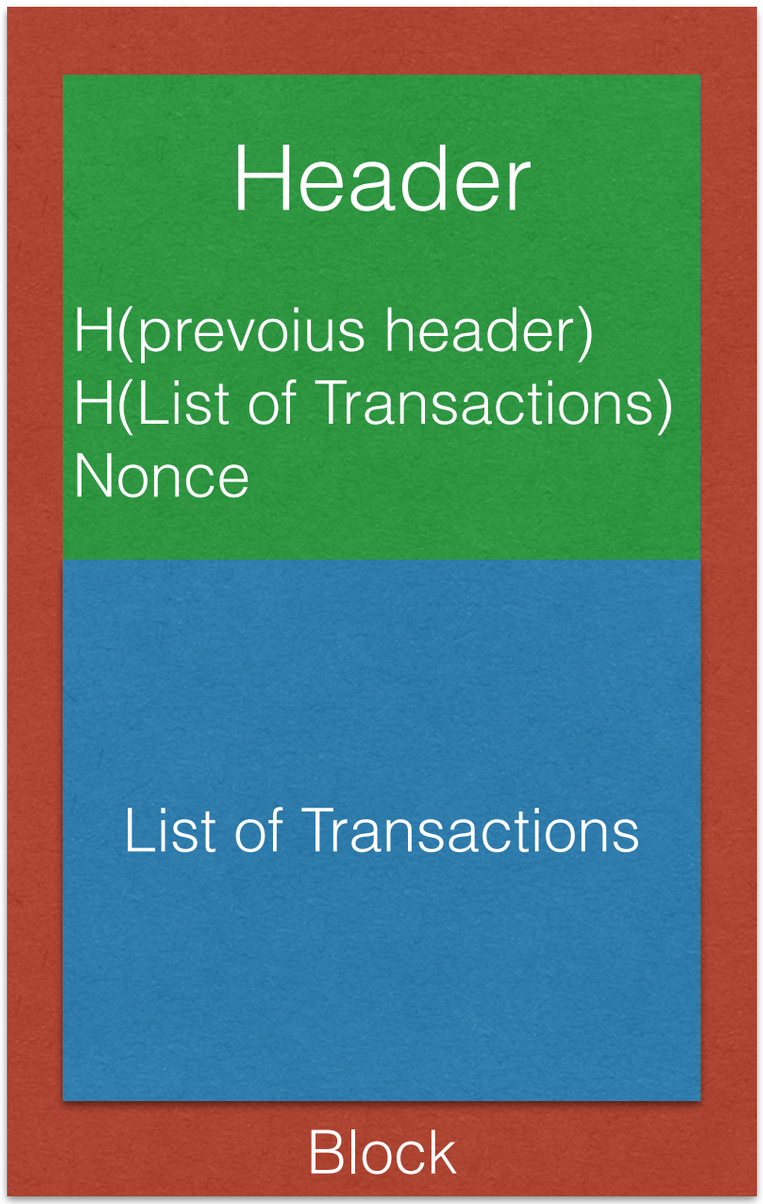
Why should I accept and broadcast this block?

I just joined: Is this really the last valid block?

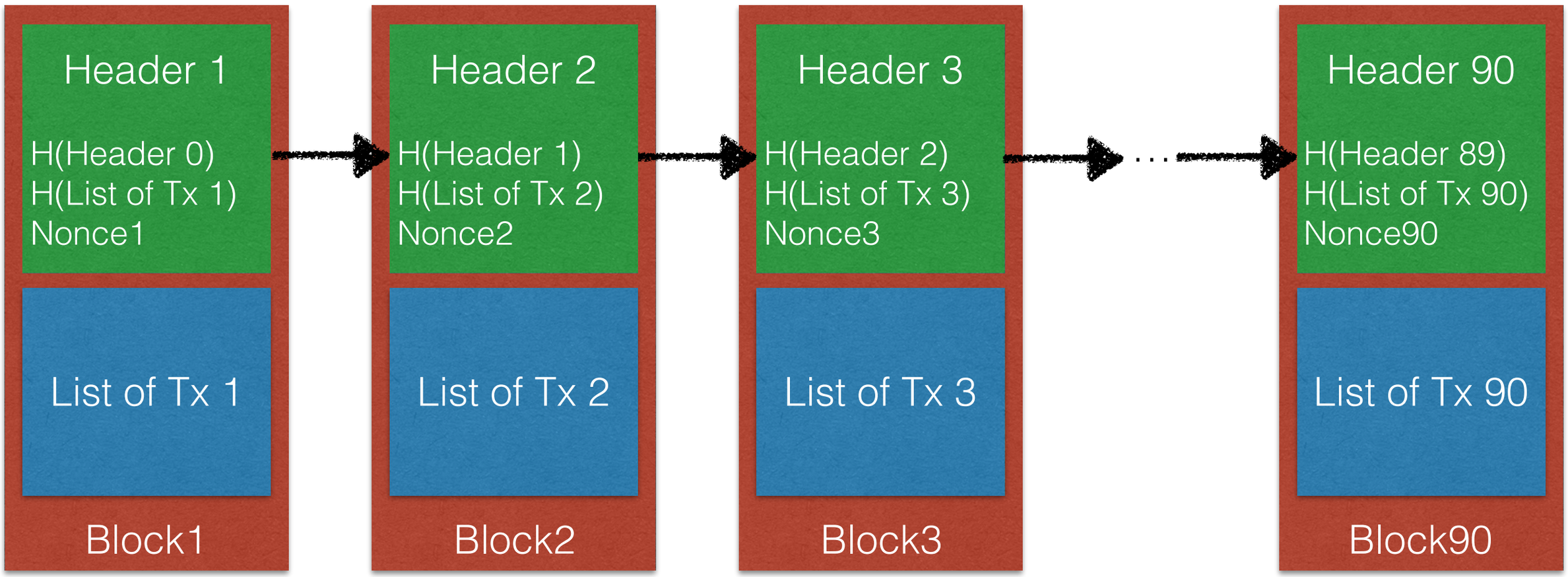
Let’s get practical…

How do Blocks really look like?

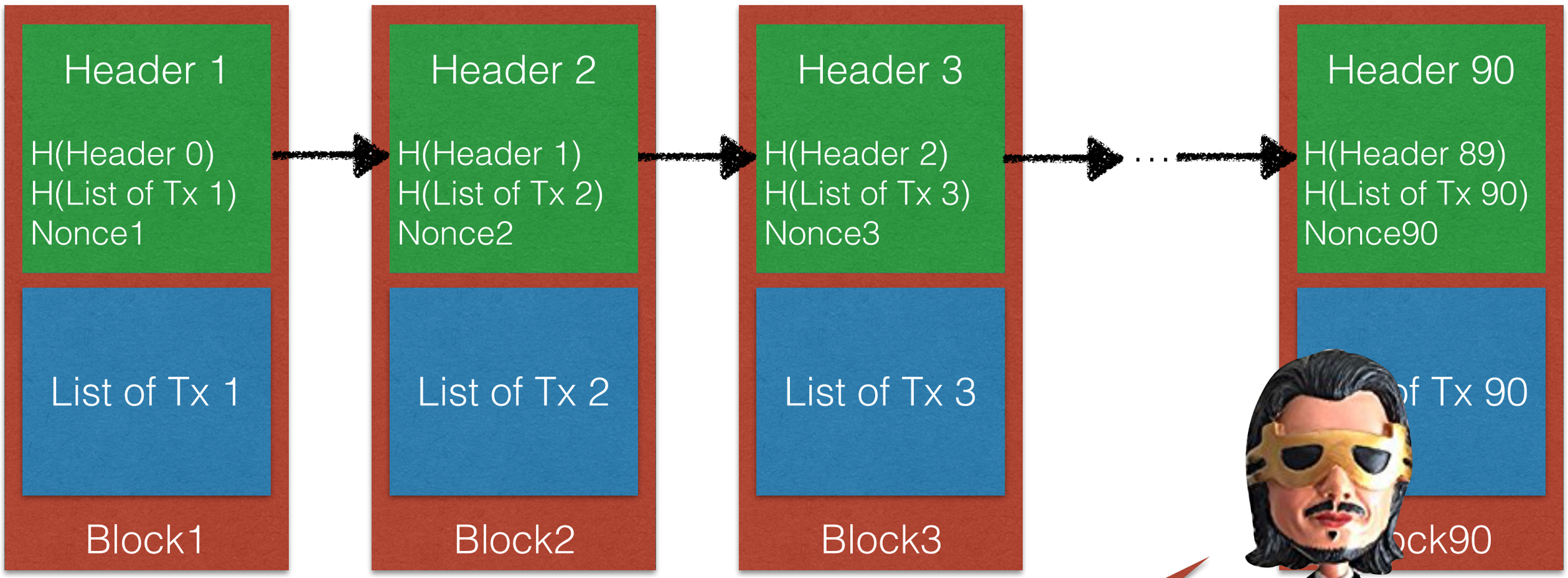
Bitcoin Blocks



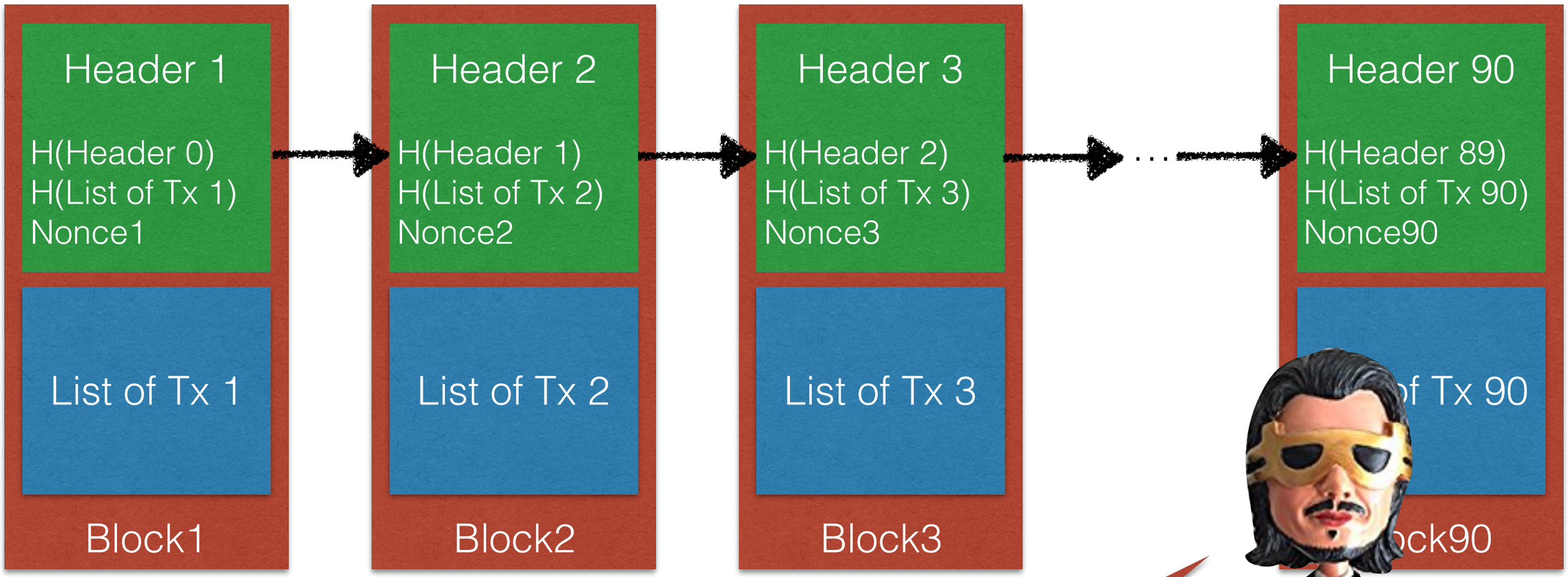
The Bitcoin Blockchain



The Bitcoin Blockchain



The Bitcoin Blockchain



I'm still not chosen, this is not fair!

Not so complicated, huh?

Generating Blocks

A fair, verifiable source of randomness?

Randomness over what?

Then how do we decide who creates the next block?

Generating Blocks

A fair, verifiable source of randomness?

Randomness over what?

Then how do we decide who creates the ne ck?

You can create a block if you can hash its header and get a string starting with, let's say, fifteen zeros

Generating Blocks

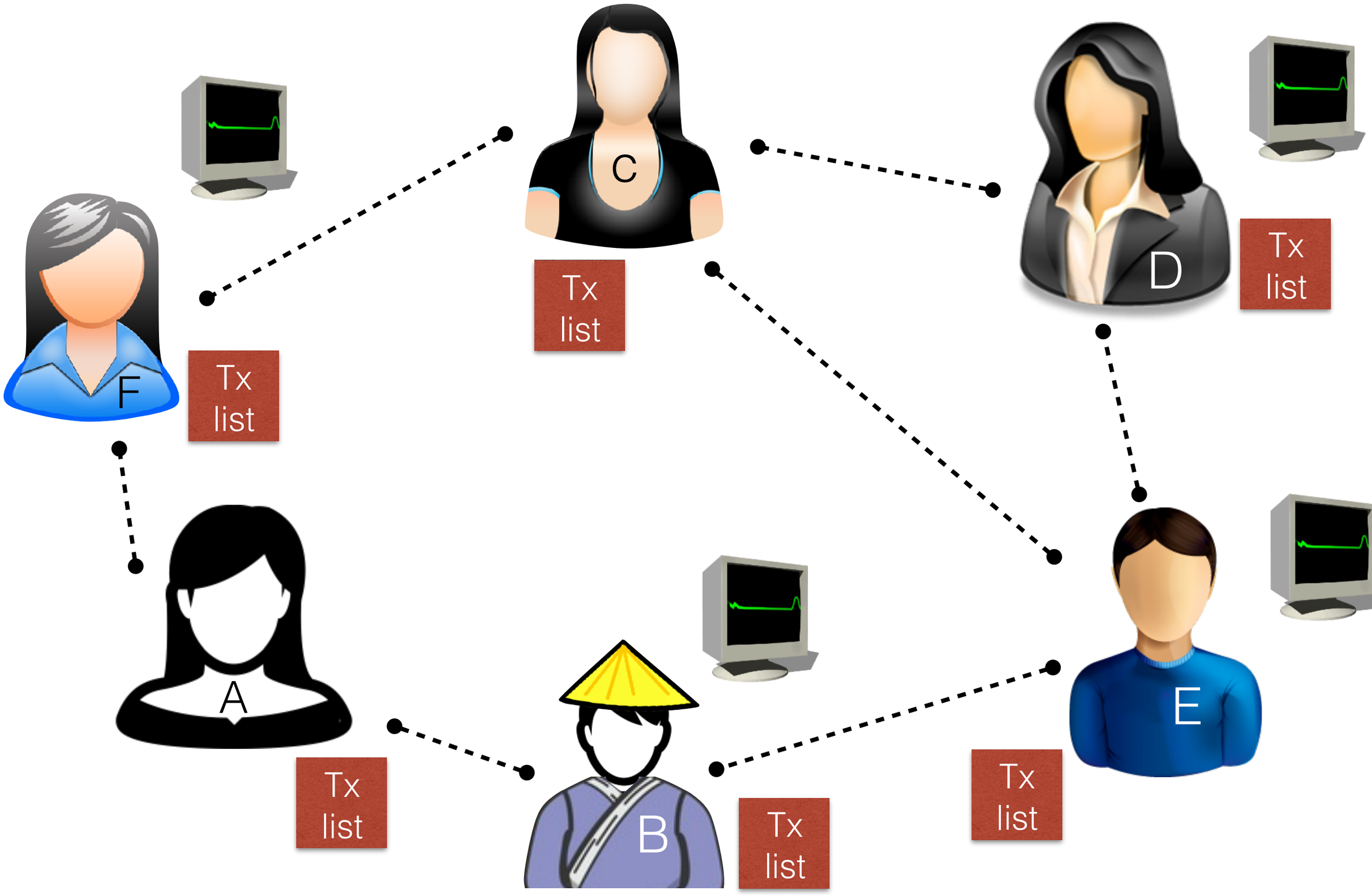
A fair, verifiable [illegible] of randomness?

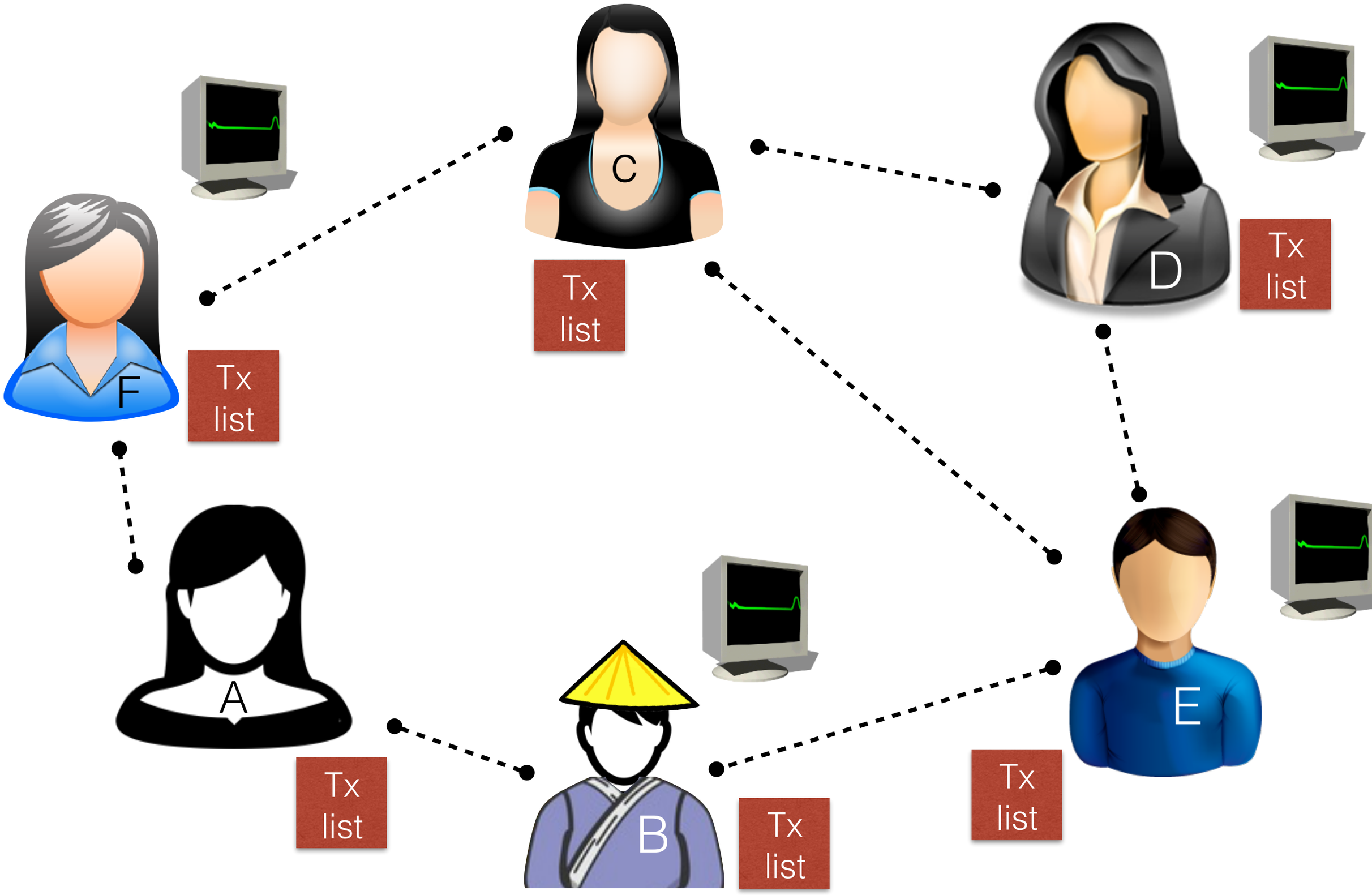
Ra[illegible]hat?

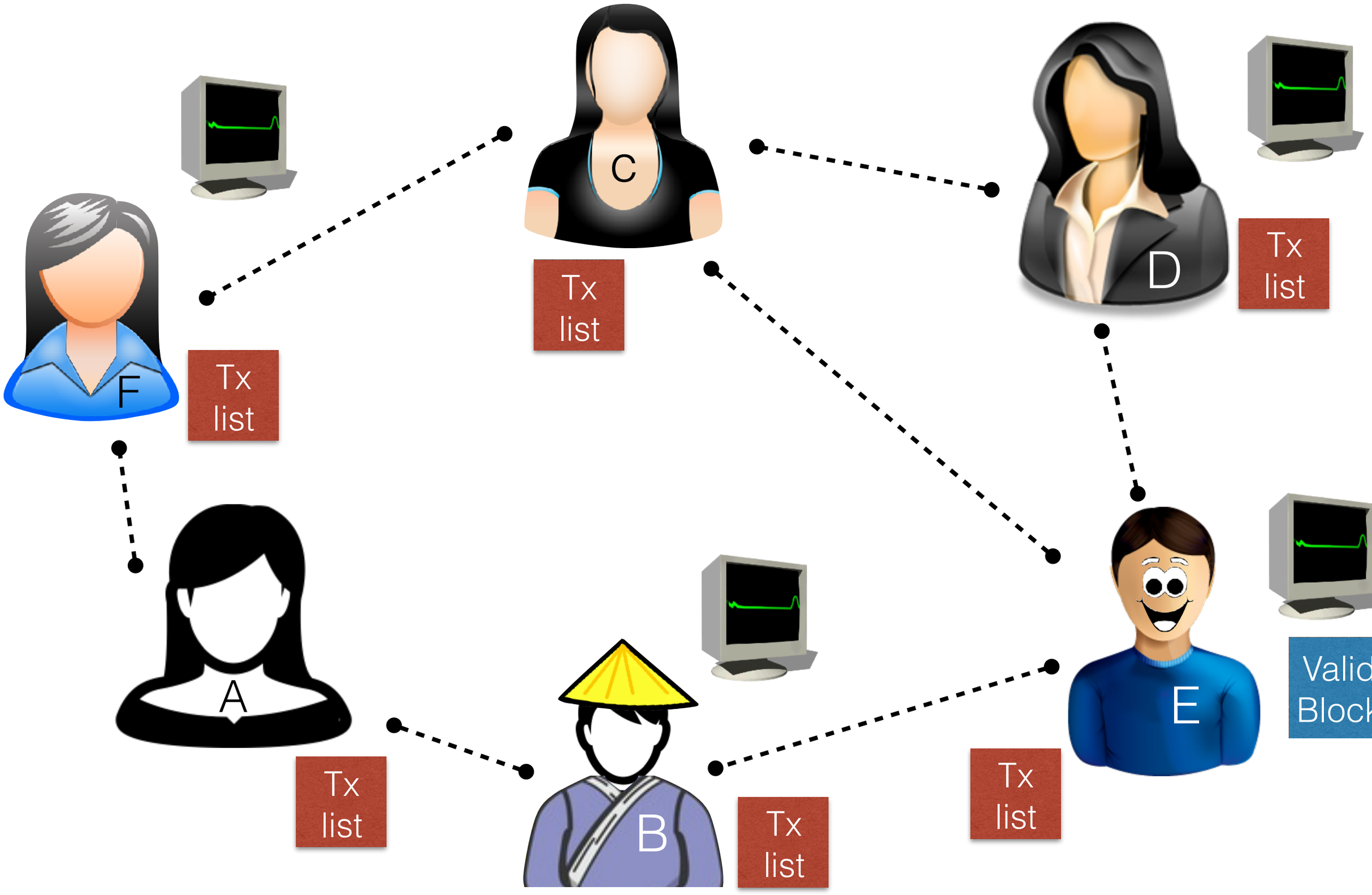
The nonce is the key!

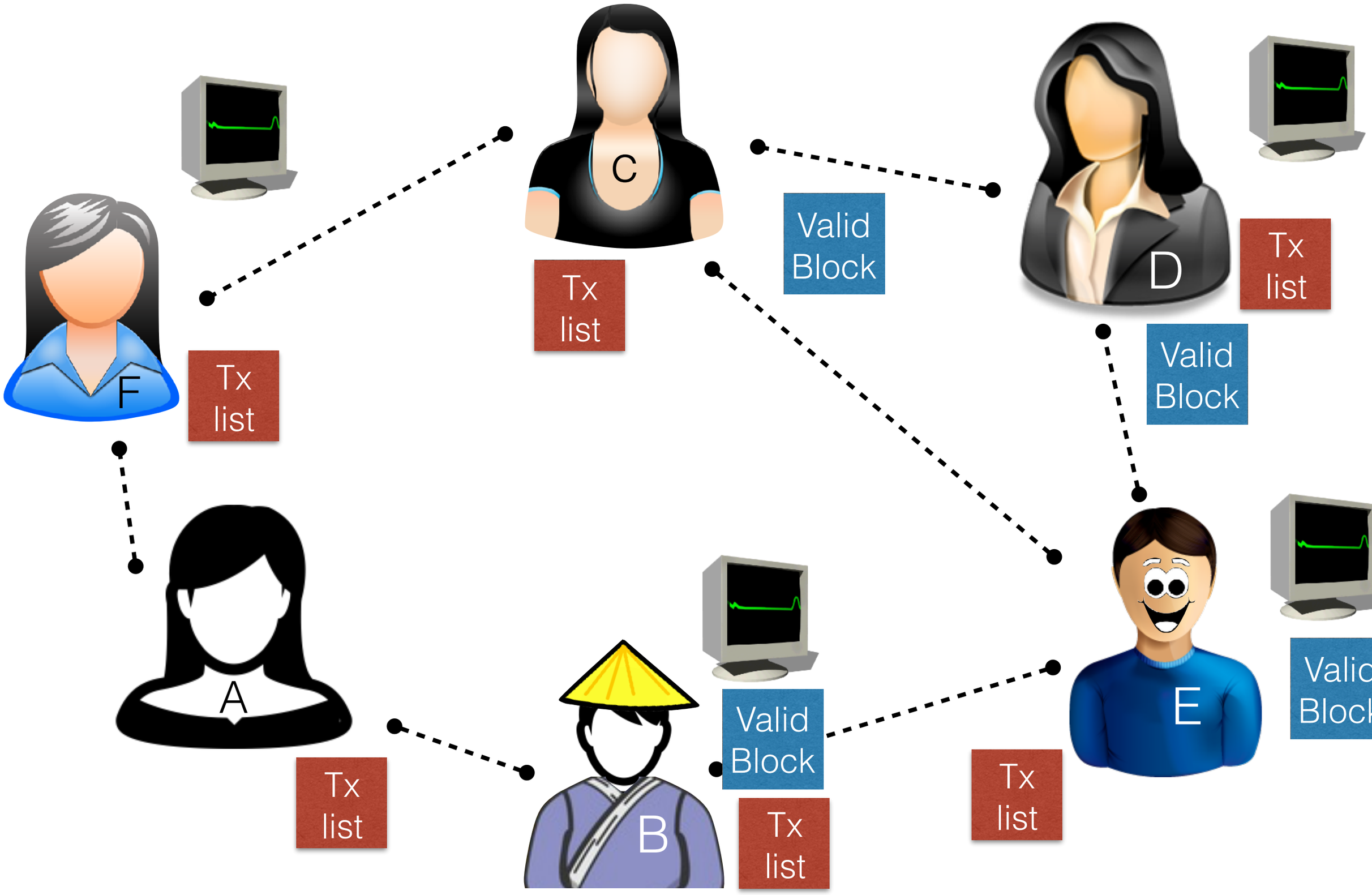
Then how do we decide who creates the ne[illegible]ck?

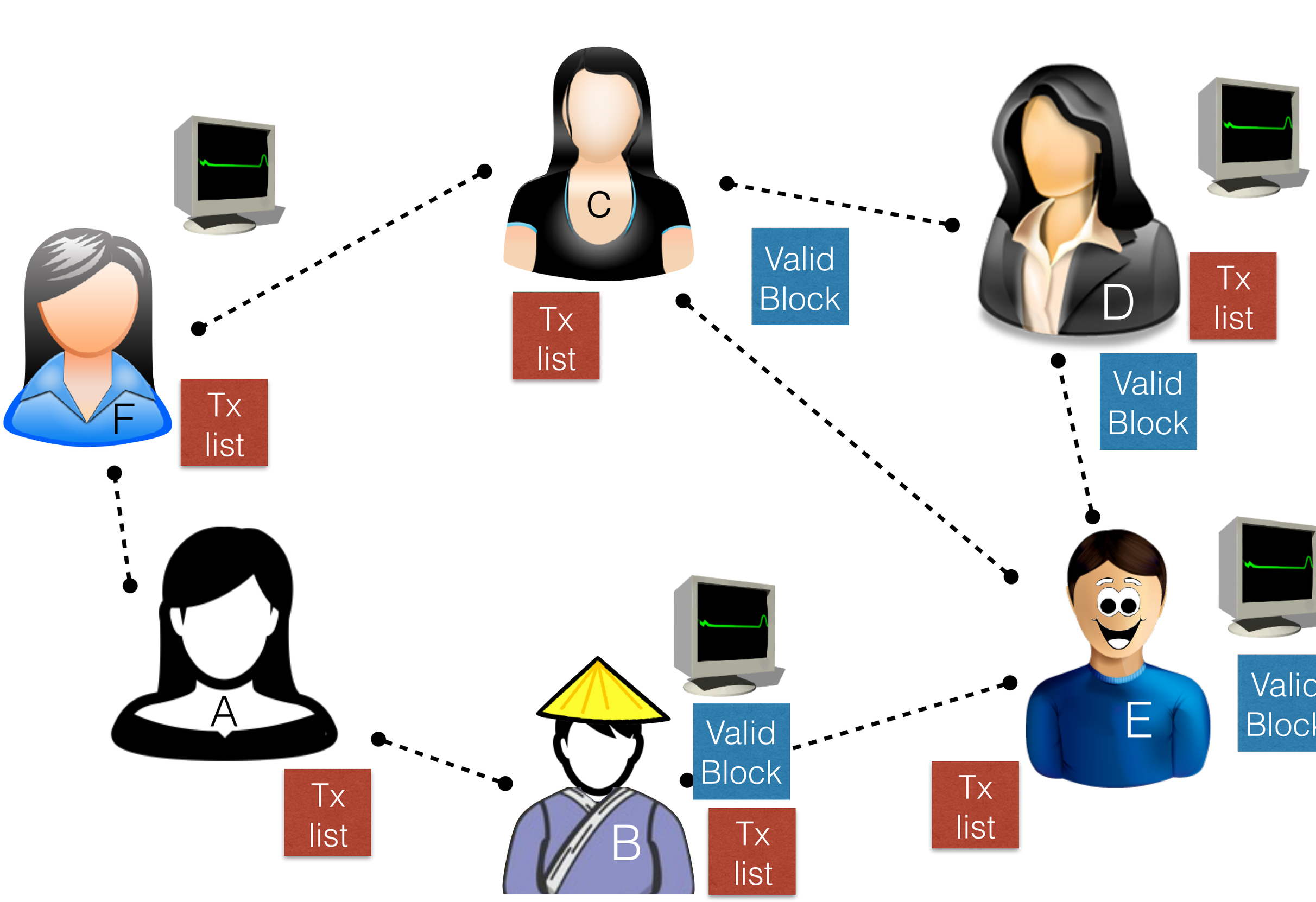
You can create a block if you can hash its header and get a string starting with, let's say, fifteen zeros


C
Tx list
D
Tx list
F
Tx list
A
Tx list
B
Tx list
E
Tx list


C
Tx list
D
Tx list
F
Tx list
A
Tx list
B
Tx list
E
Tx list


C
Tx list
D
Tx list
F
Tx list
A
Tx list
B
Tx list
Tx list
E
Valid Block


C
Tx list
Valid Block
D
Tx list
Valid Block
F
Tx list
A
Tx list
B
Valid Block
Tx list
Tx list
E
Valid Block



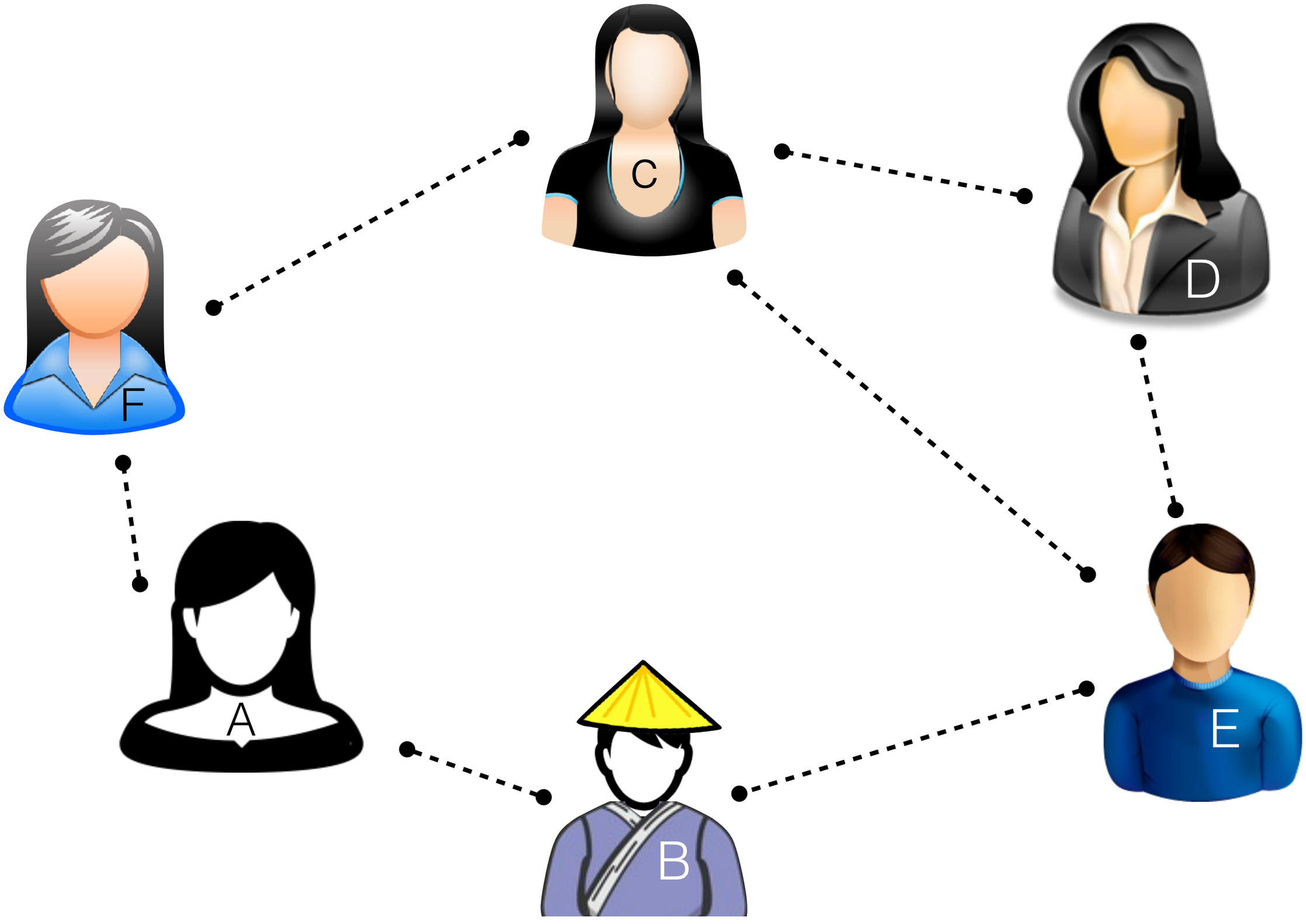
Bitcoin Mining

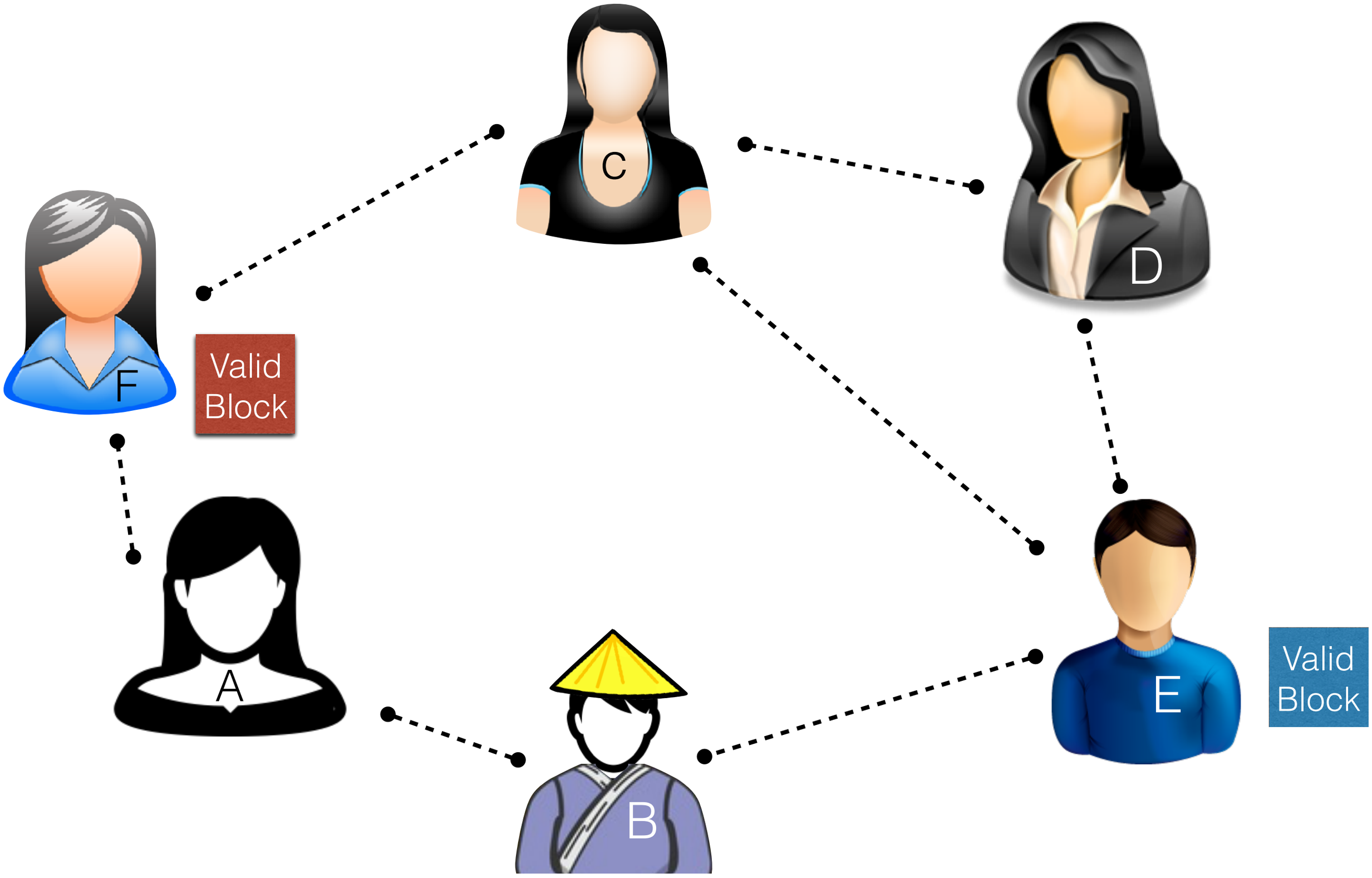
Bitcoin blocks are not actually signed

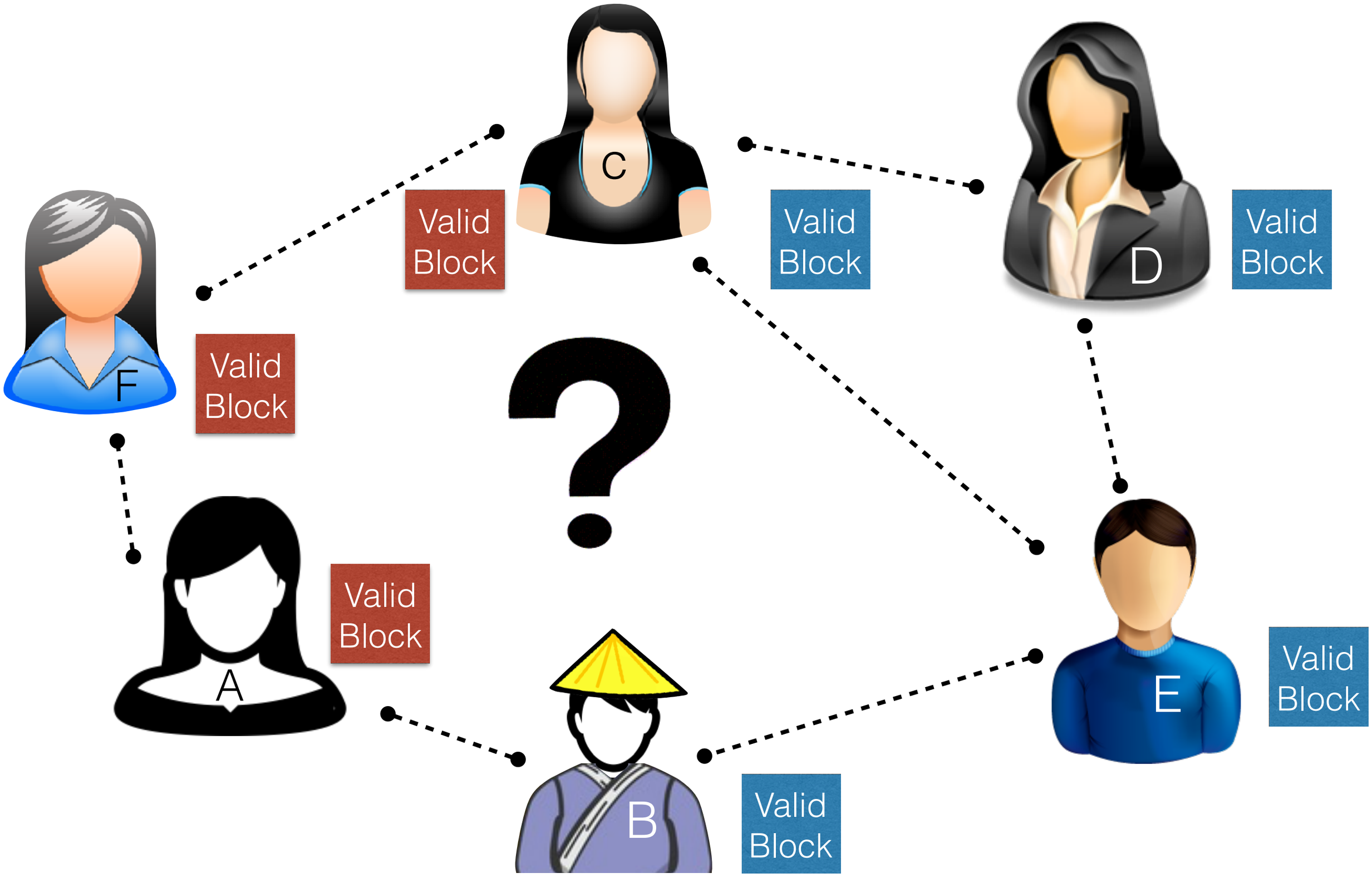
Valid blocks are simply generated by finding a hash with a certain coinbase transaction

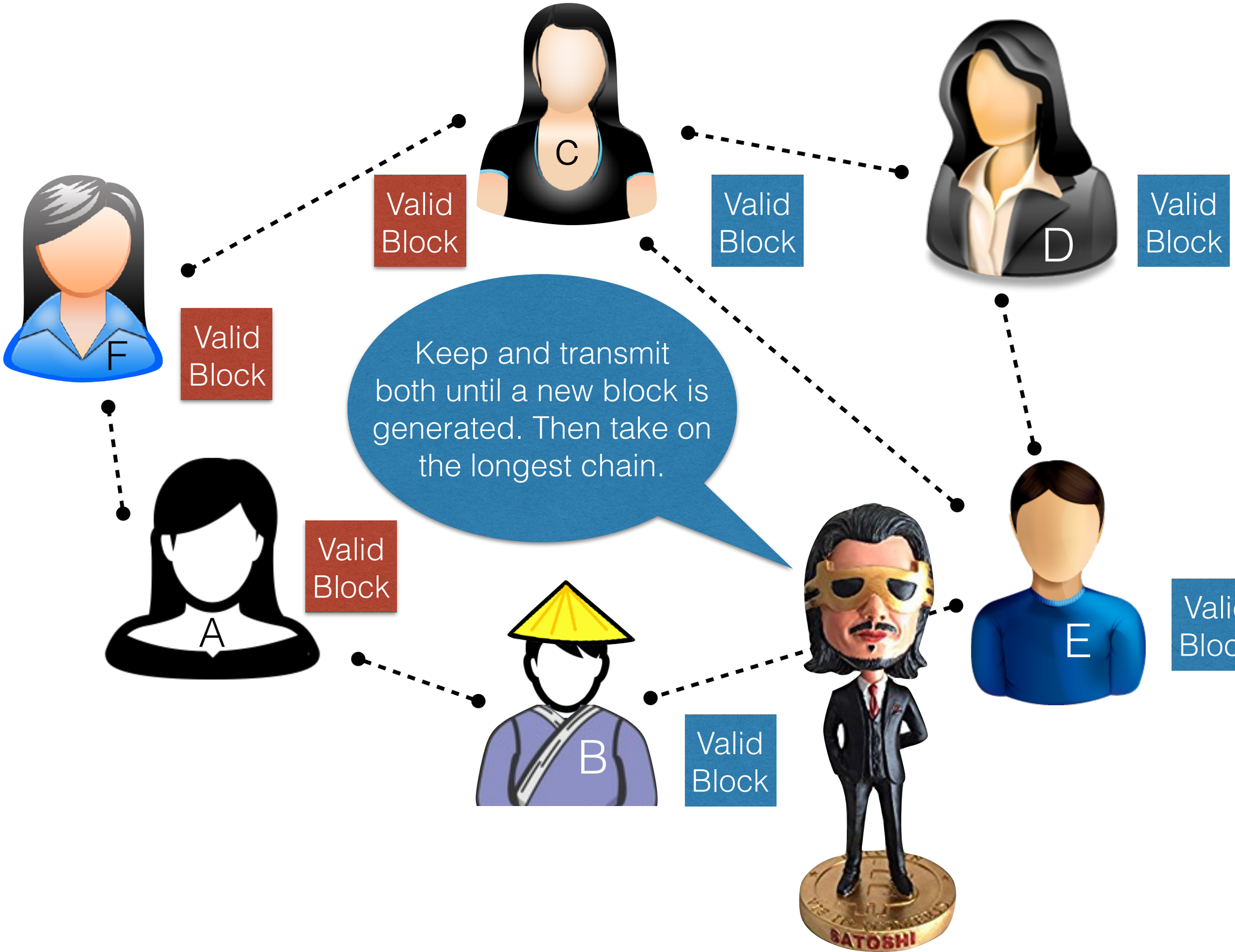
The recipient of the coinbase transaction can *prove* that he worked hard

Proof of Work (PoW)

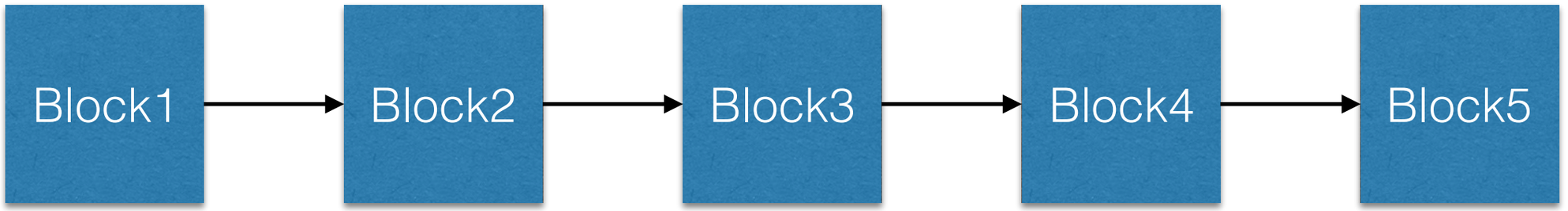

C
D
F
A
B
E


C
D
F
Valid
Block
A
B
E
Valid
Block


C
Valid Block
Valid Block
D
Valid Block
F
Valid Block
?
A
Valid Block
E
Valid Block
B
Valid Block

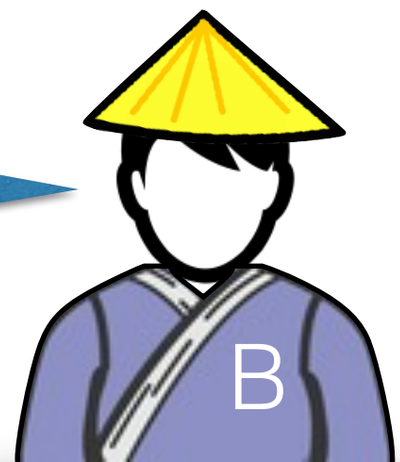

C
Valid Block
Valid Block
D
Valid Block
F
Valid Block
Keep and transmit both until a new block is generated. Then take on the longest chain.
A
Valid Block
B
Valid Block
SATOSHI
E
Valid Block

How to cheat?

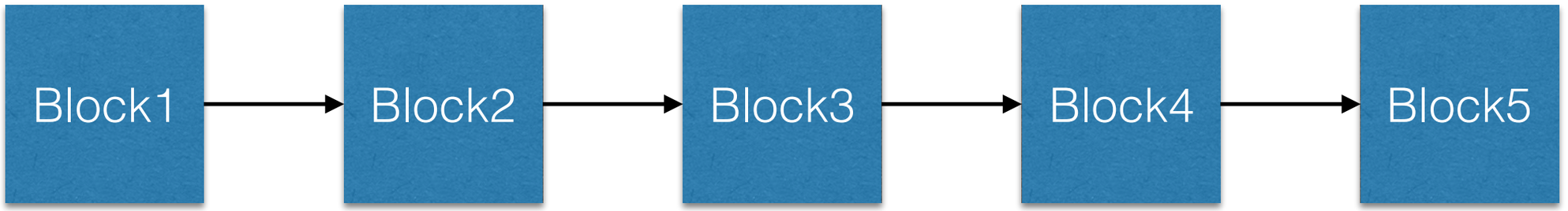


Thanks!

Hey Alice,
look at block 5, I
have transfered
you 10

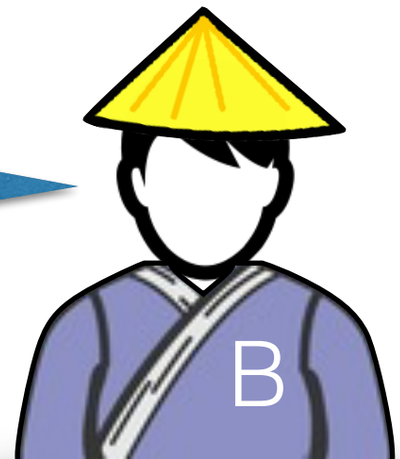


How to cheat?

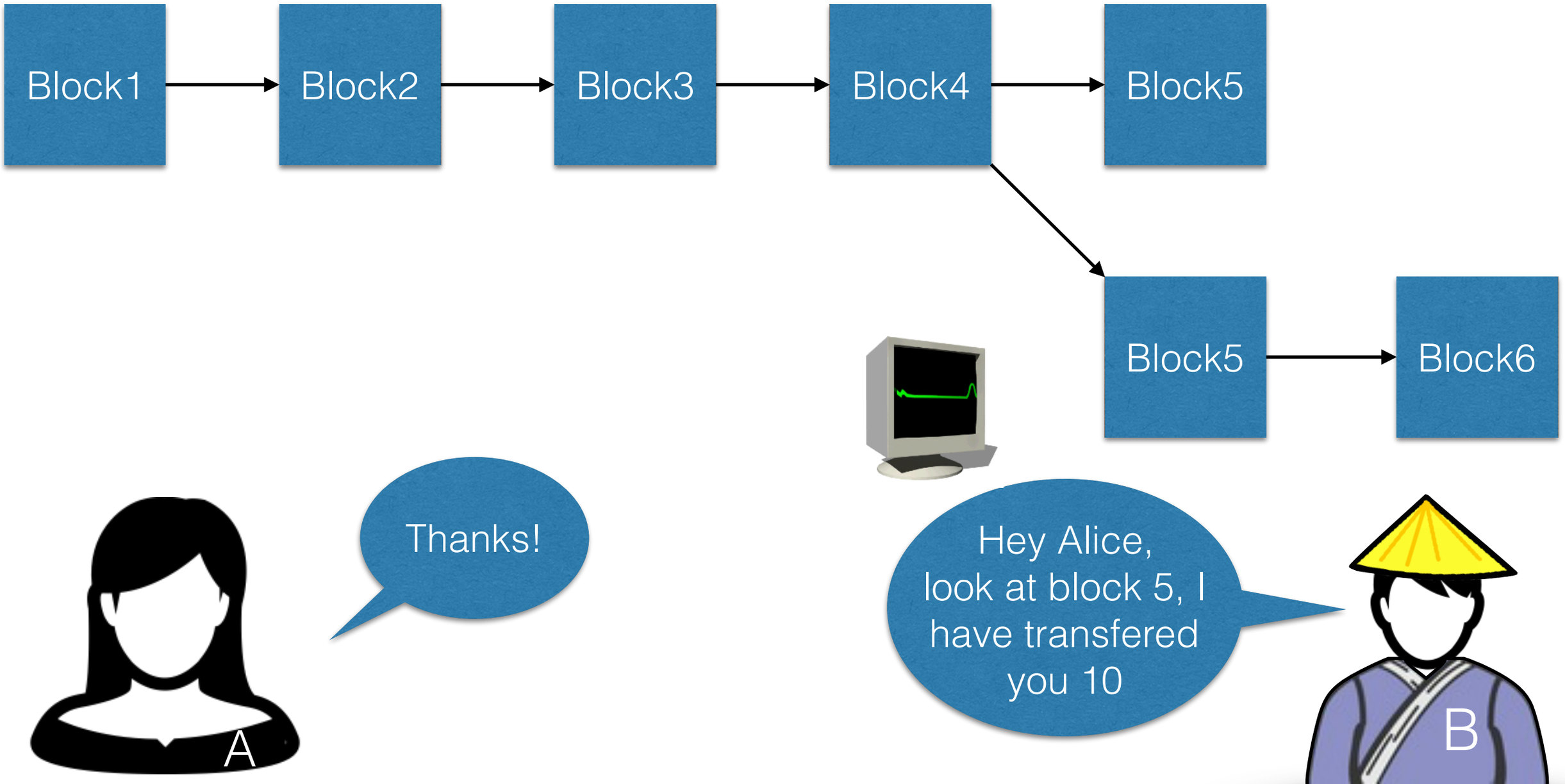


Thanks!

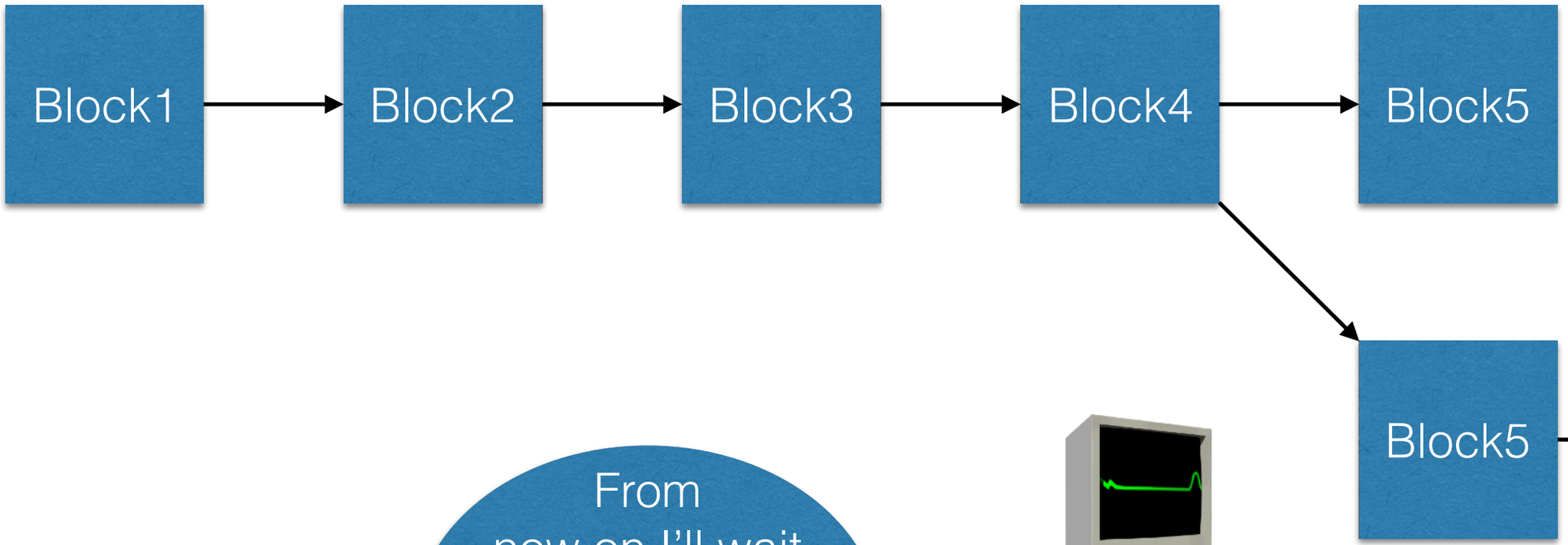
Hey Alice,
look at block 5, I
have transfered
you 10

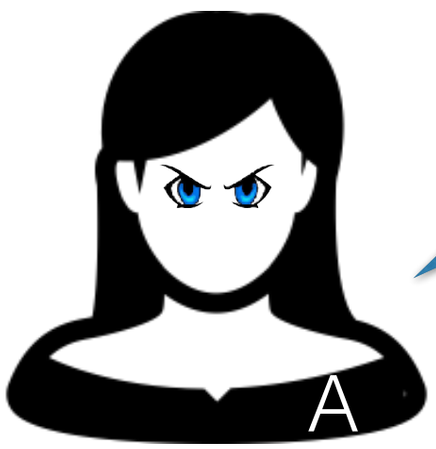


How to cheat?


Block1
Block2
Block3
Block4
Block5
Block5
Block6
Thanks!
A
Hey Alice,
look at block 5, I
have transfered
you 10
B

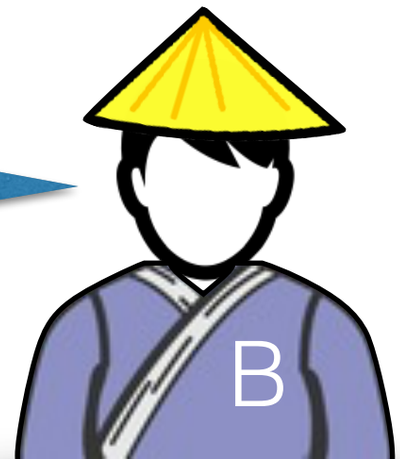
How to cheat?





From
now on I'll wait
until the transaction
is well buried into
the blockchain

Hey Alice,
look at block 5, I
have transfered
you 10

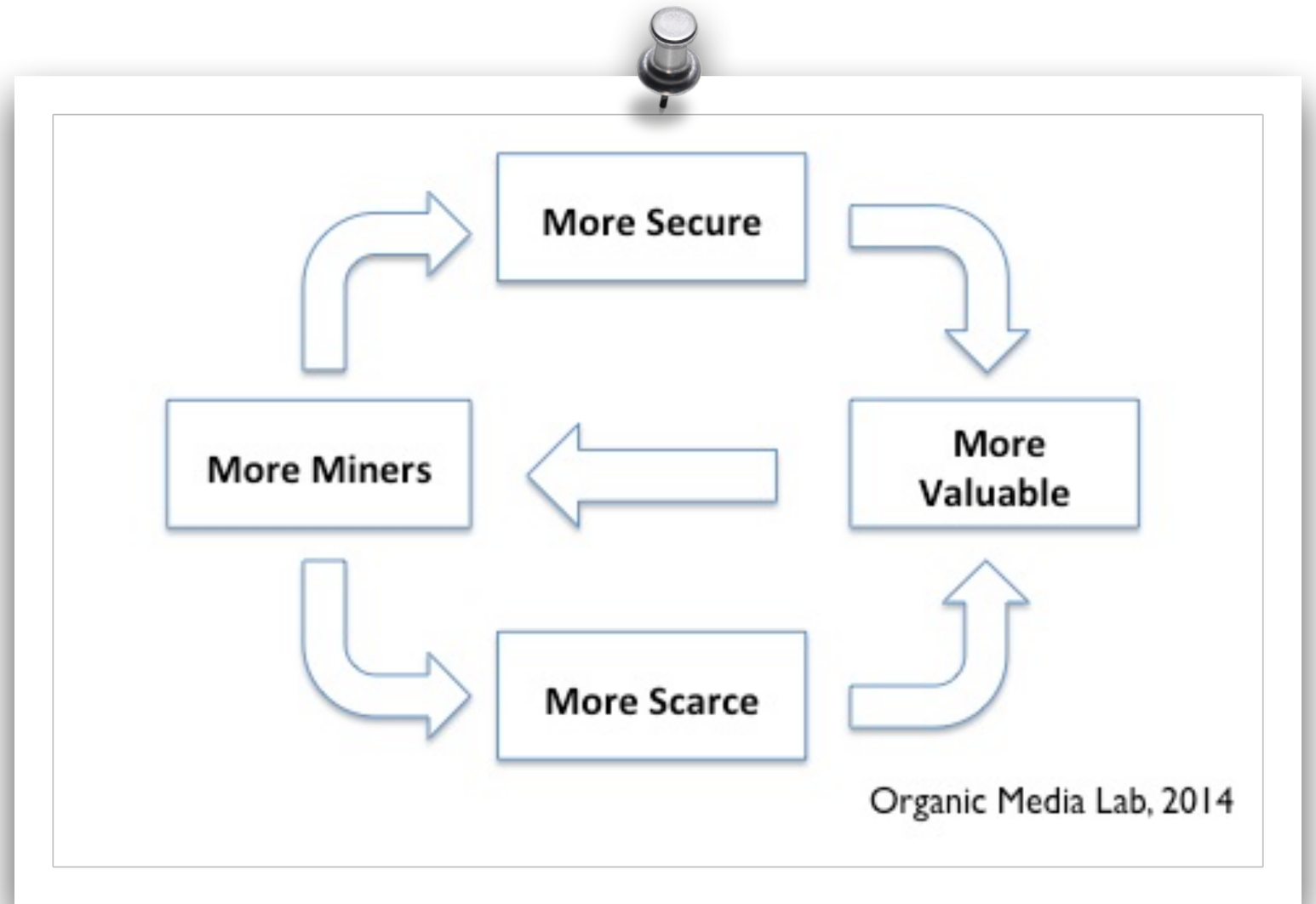


Bitcoin: A Peer-to-Peer Electronic Cash System

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.

The Virtuous Cycle



The Bitcoin Blockchain

The Bitcoin Protocol

Mining

Transactions

Questions

Thanks.