

IIC3253

Criptomonedas

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

Mandar 10 a
Bob



Name	Balance
Alice	100
Bob	100



Alice te
mandó 10

Name	Balance
Alice	90
Bob	110

Perfect👌
mando las
paltas



¿Cómo sé que estoy hablando con alguien que tiene 1฿?

¿Cómo sé que me está transfiriendo 1฿?

¿Cómo sé que no se lo está gastando en otro lado?

¿De dónde salió ese bitcoin, quién lo creó?

Ingredientes necesarios

Funciones de hash

Firmas digitales

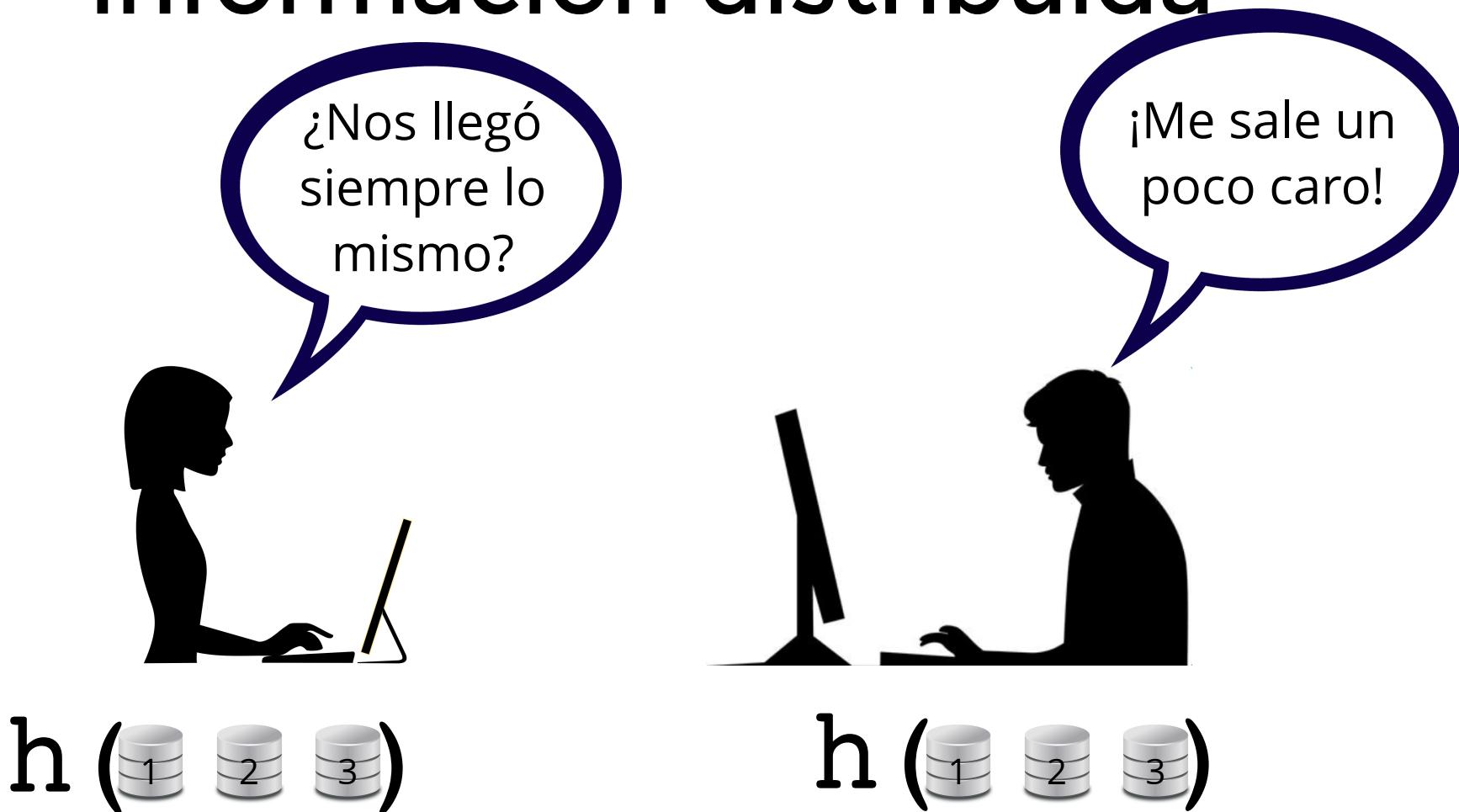
Funciones de Hash e información distribuida



Funciones de Hash e información distribuida



Funciones de Hash e información distribuida

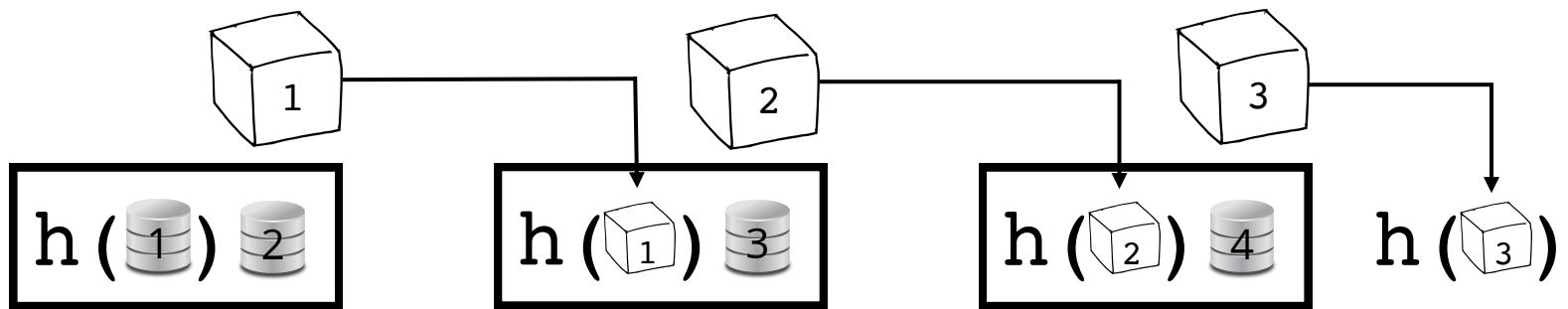


Funciones de Hash e información distribuida



$$h(h(\text{h}(\text{1})) \text{h}(\text{h}(\text{2})) \text{h}(\text{h}(\text{3})))$$

$$h(h(\text{h}(\text{1})) \text{h}(\text{h}(\text{2})) \text{h}(\text{h}(\text{3})))$$



Nos basta con compartir $h(3)$

¡EL FUTURO ES HOY!



BLOCK
CHAIN

Dude...
seriously?



¿Qué es lo nuevo?

¿Funciones de hash criptográficas?

1973

¿Firmas digitales?

1978



¿El ✨Blockchain✨?

1970s

**iHagamos una
criptomoneda!**

Alice



Bob



Carol



Dan



Eve



Frank



SK_A

SK_B

SK_C

SK_D

SK_E

SK_F

PK_A

PK_B

PK_C

PK_D

PK_E

PK_F



$= SK$



$= PK$

Transaction 1 (Tx1)

- Output 1: 200 to PK_A
- Output 2: 50 to PK_B
- Output 3: 120 to PK_C
- ...

¡Bacán,
tengo 320!



¡Bacán,
tengo 250!



Tx3

Input 1: Tx2, Output 1

Output 1: 200 to PK_C

J. Pérez
(Firmado con SK_B)

Transaction 1 (Tx1)

- Output 1: 200 to PK_A
- Output 2: 50 to PK_B
- Output 3: 120 to PK_C
- ...

Tx2

Input 1: Tx1, Output 1

Output 1: 200 to PK_B

J. Pérez
(Firmado con SK_A)





Toda la plata es una cadena
originada en la Transacción 1

Transaction 1 (Tx1)

- Output 1: 200 to PK_A
- Output 2: 50 to PK_B
- Output 3: 120 to PK_C
- ...

¿Siempre tengo que gastar
outputs completos?

¿Qué pasa si quiero combinar
outputs o gastar sólo una parte?



¿Cómo
pago 210?



Tx3

Input 1: Tx1, Output 2

Input 2: Tx2, Output 1

Output 1: 210 to PK_C

Output 2: 40 to PK_B

A handwritten-style signature of the letter 'J' followed by 'Kun'.

(Firmado con SK_B)



Transaction 1
(Tx1)

- Output 1: 200 to PK_A
- Output 2: 50 to PK_B
- Output 3: 120 to PK_C
- ...

Tx2

Input 1: Tx1, Output 1

Output 1: 200 to PK_B

A handwritten-style signature of the letter 'J' followed by 'Kun'.

(Firmado con SK_A)





Una transacción puede tener tantos inputs y outputs como queramos

Transaction 1 (Tx1)

- Output 1: 200 to PK_A
- Output 2: 50 to PK_B
- Output 3: 120 to PK_C
- ...

Los outputs siempre se gastan completos



¿Cómo lo
arreglamos?



Transaction 1 (Tx1)

- Output 1: 200 to PK_A
- Output 2: 50 to PK_B
- Output 3: 120 to PK_C
- ...

Tx2

Input 1: Tx1, Output 1

Output 1: 200 to PK_C

A handwritten signature in black ink.

(Firmado con SK_A)

¡Me
micho!



Tx2

Input 1: Tx1, Output 1

Output 1: 200 to PK_B

A handwritten signature in black ink.

(Firmado con SK_A)

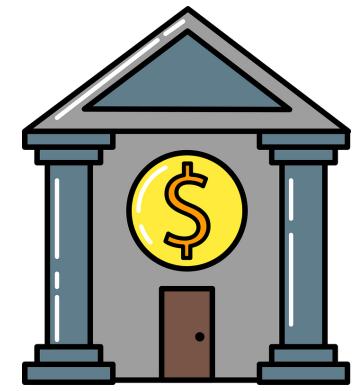
¡Yo puedo
ayudar!





BankCoin

1. La transacción inicial Tx1, se respeta igual que antes
2. Una transacción es una secuencia de pagos que llega a Tx1, igual que antes
3. El banco publicará en un blockchain firmado todas las transacciones válidas



SK

PK

Input 1: Tx1, Output 1

Output 1: 200 to PK_B

Tx2

(Firmado con SK_A)

Input 1: Tx1, Output 2

Output 1: 50 to PK_c

Tx3

(Firmado con SK_B)

J. Huerta (Firmado con SK_{Escuela})

Input 1: Tx1, Output 3

Input 2: Tx3, Output 1

Output 1: 170 to PK_D

Tx4

J. Huerta (Firmado con SK_C)

Input 1: Tx2, Output 1

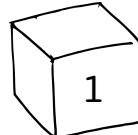
Output 1: 200 to PK_c

Tx5

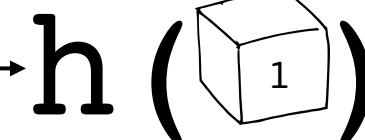
J. Huerta (Firmado con SK_B)

Transaction 1
(Tx1)

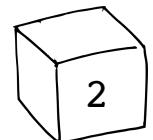
- Output 1: 200 to PK_A
- Output 2: 50 to PK_B
- Output 3: 120 to PK_C
- ...



Cool!



¡Está
funcionando
perfect!





¿Ha estado como lento el sistema los últimos días o no?

Sorry, problema técnico...

Vale, tranqui
estamos
empezando



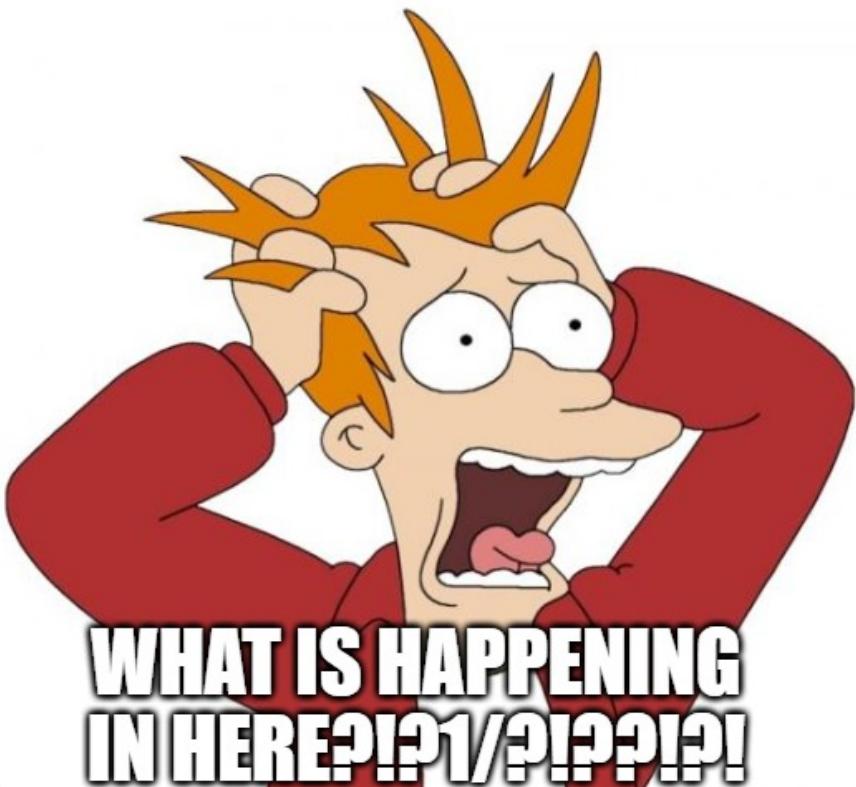


Qué raro, como hace una semana que no logro meter transacciones

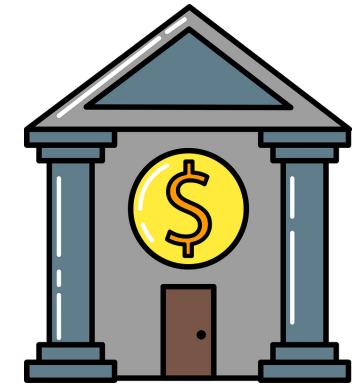
Sí, es que no me caes tan bien...

Pero por un pago se todo se puede





WHAT IS HAPPENING
IN HERE?!?!



Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshi@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

¿A quién le mando mis transacciones?



Acabo de llegar, ¿qué hago?



¿Quién creará los bloques ahora?



Yo puedo ayudar

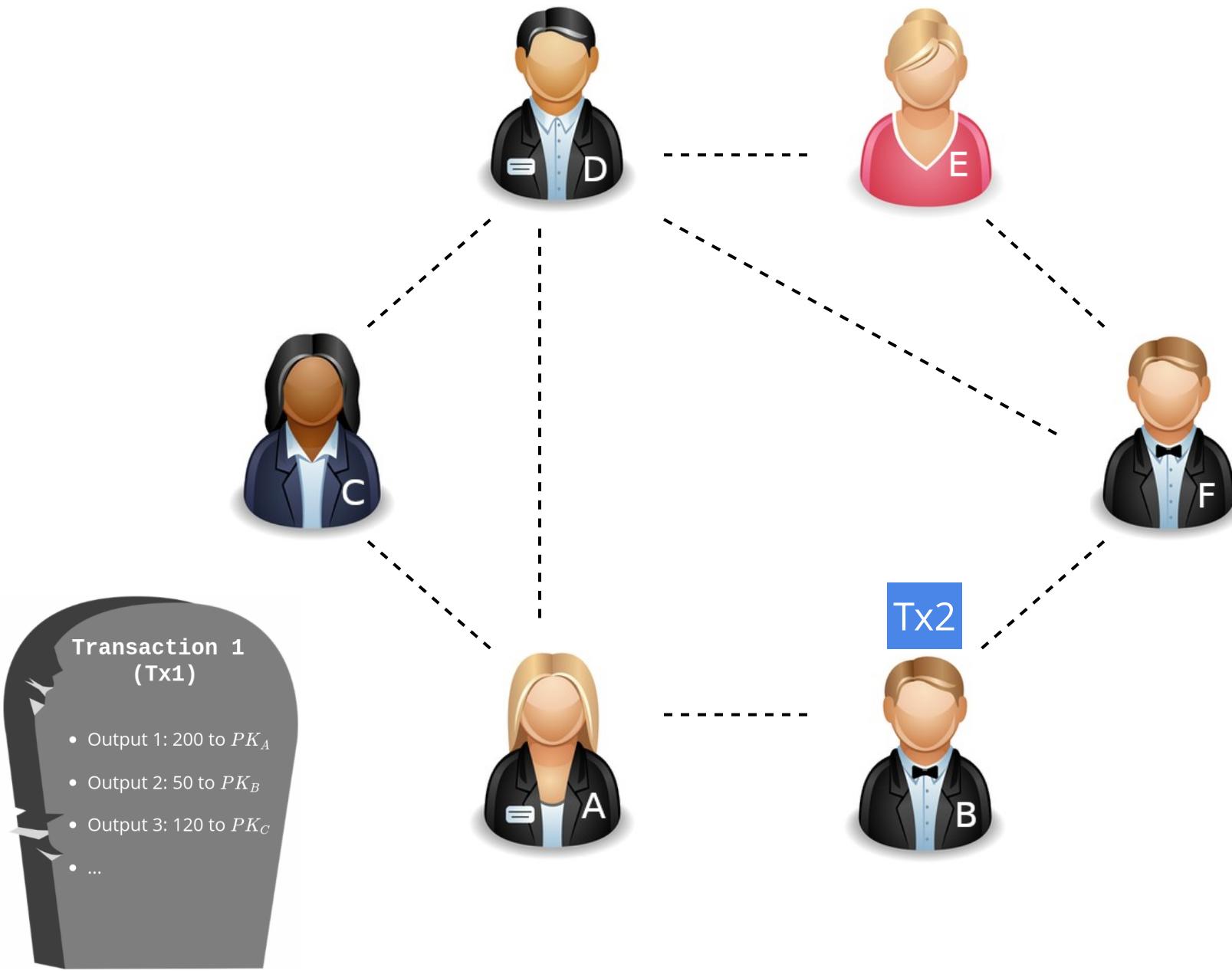


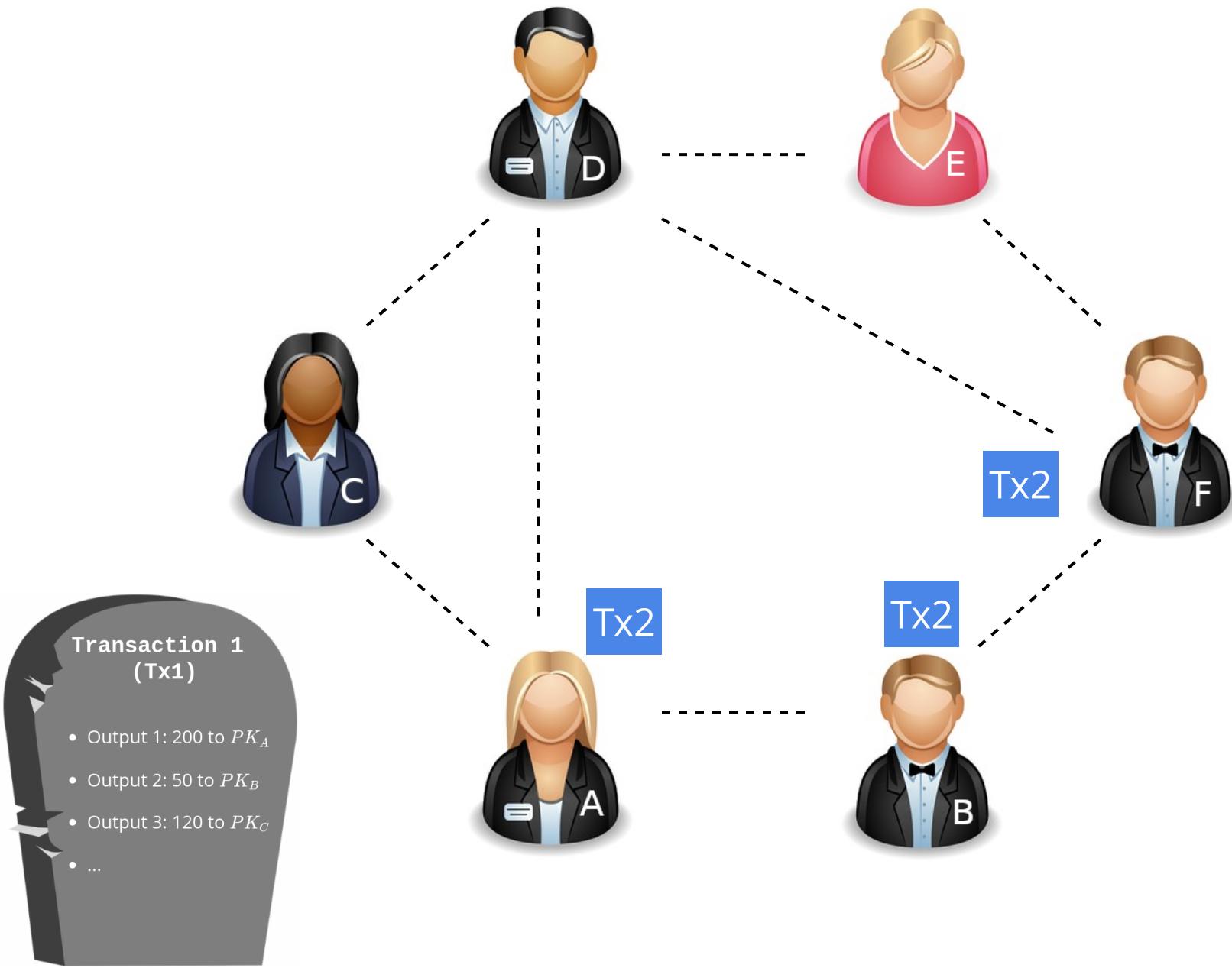
bitcoin

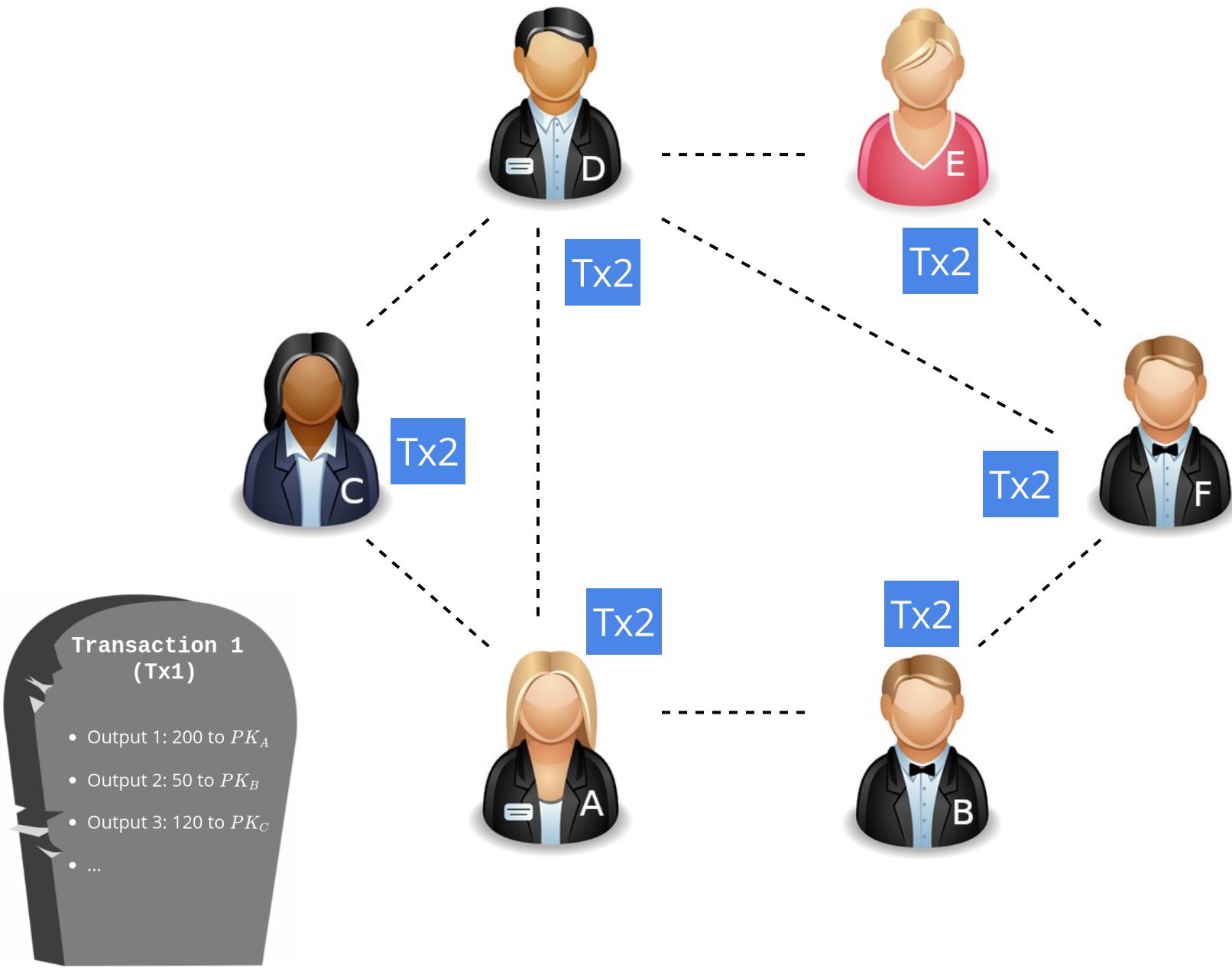
- La transacción inicial se respeta igual que antes, y le llamaremos bloque 0
- Si quieres pagar, mandale transacciones a tus vecinos
- Si recibes una nueva transacción, mándasela a tus vecinos

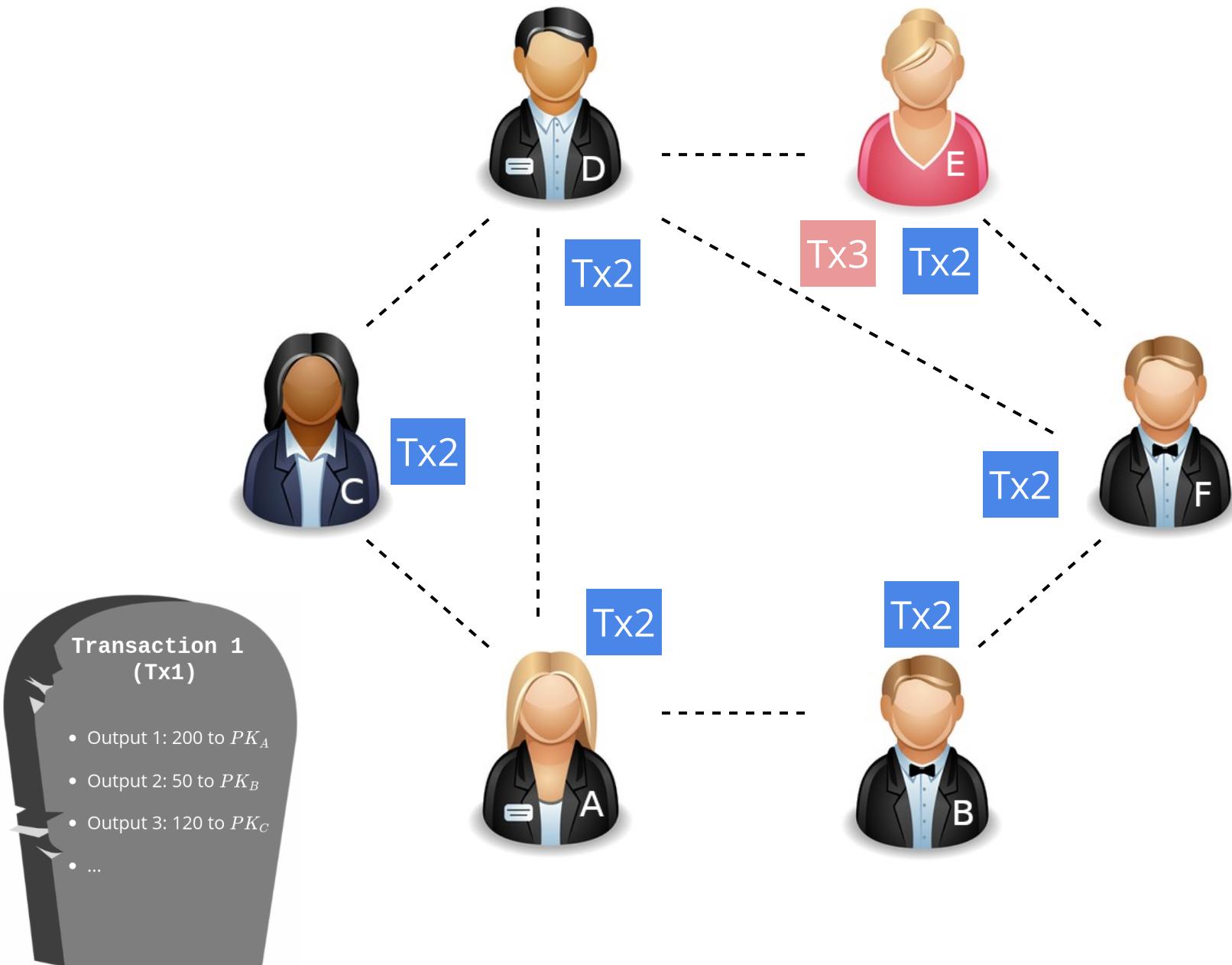
Nada complicado por ahora...



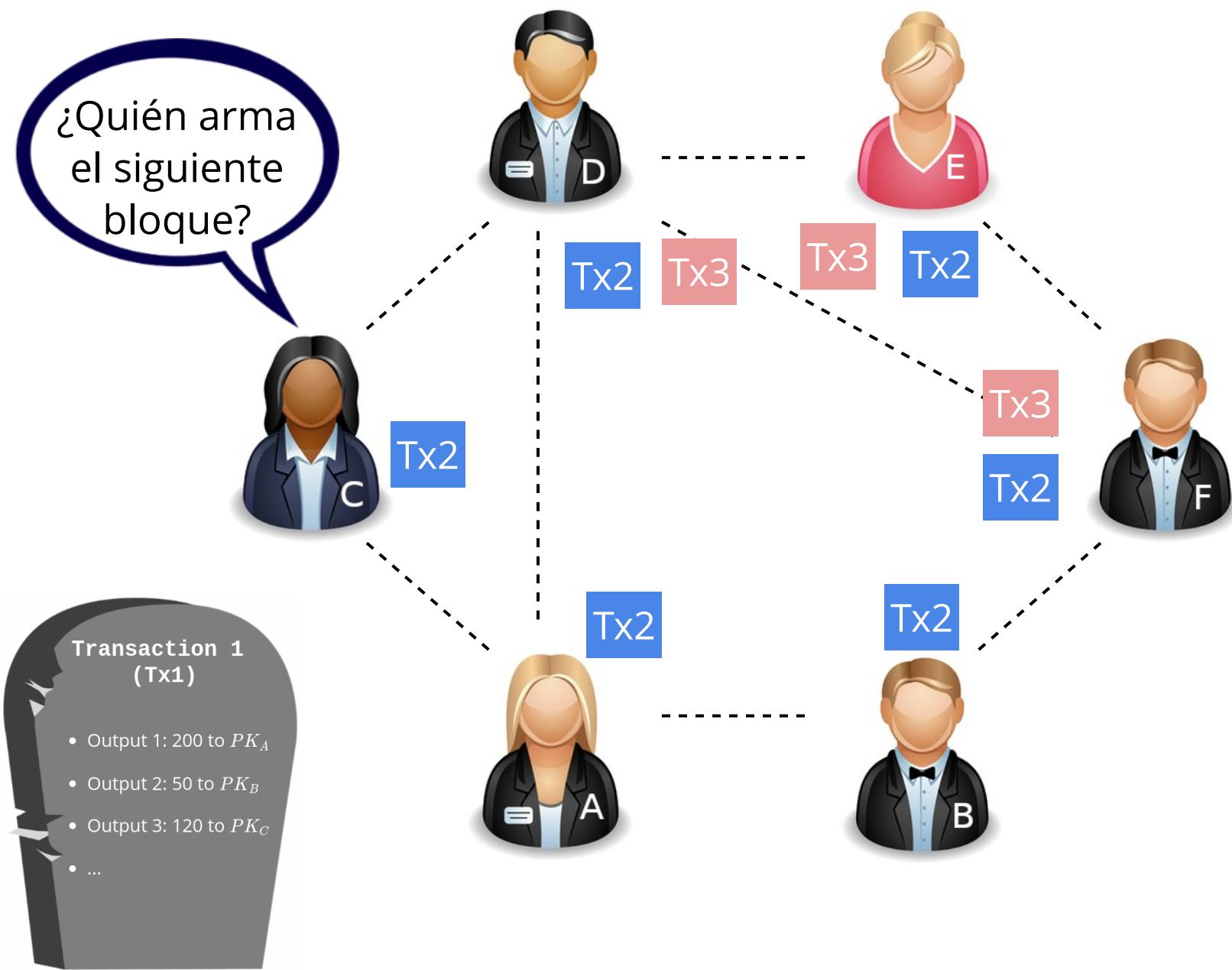






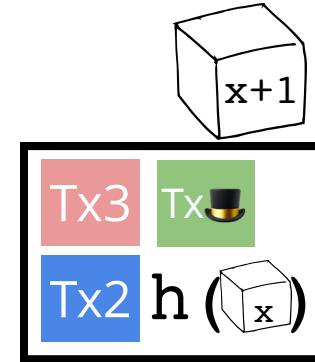


¿Quién arma el siguiente bloque?



- Cuando te elijan, forma un bloque con lo que tengas y mándalo a tus vecinos
- Si recibes un bloque, verifica que sea correcto y mándalo a tus vecinos
- ...

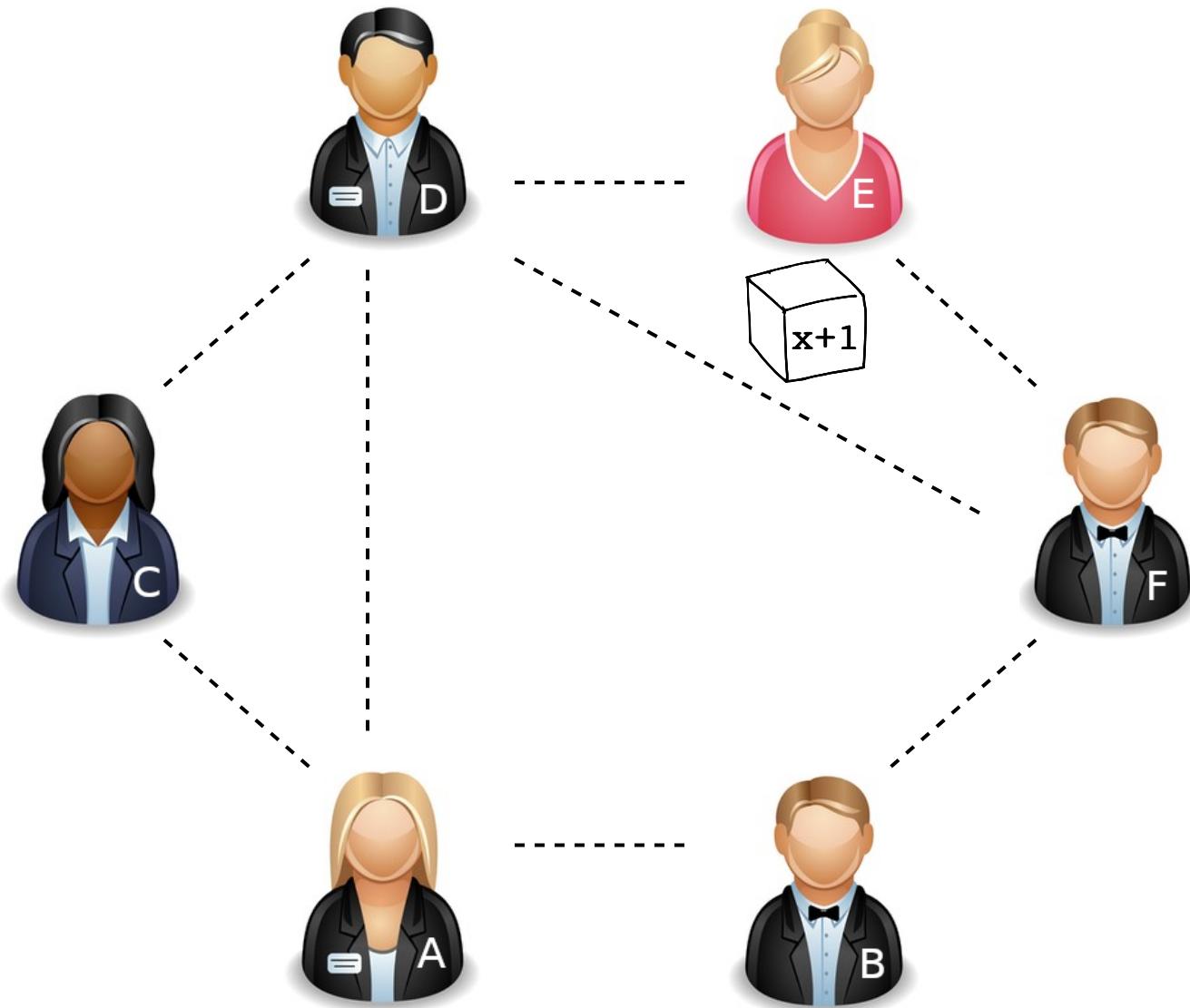
¿Por qué
haría yo
esto bien?

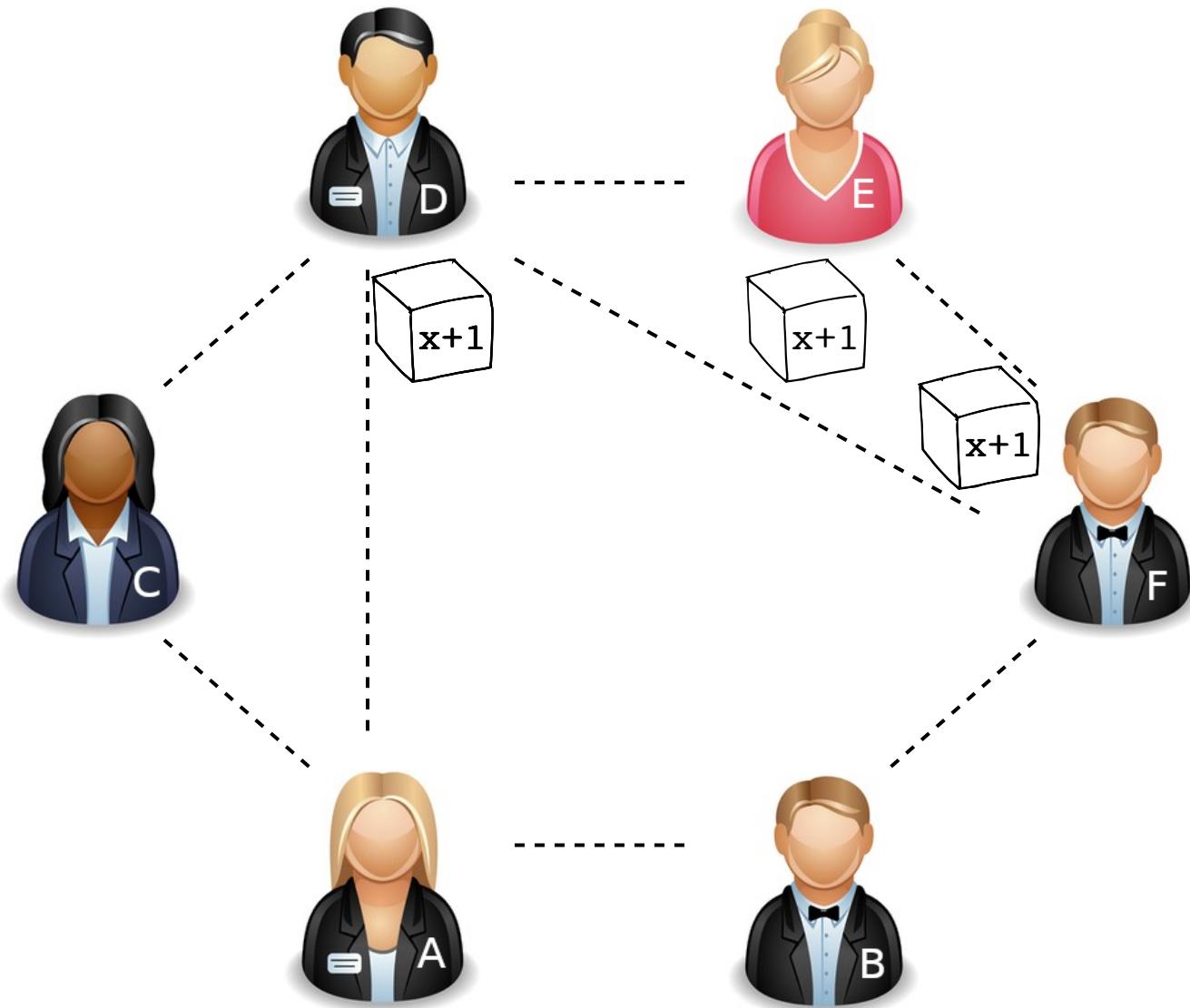


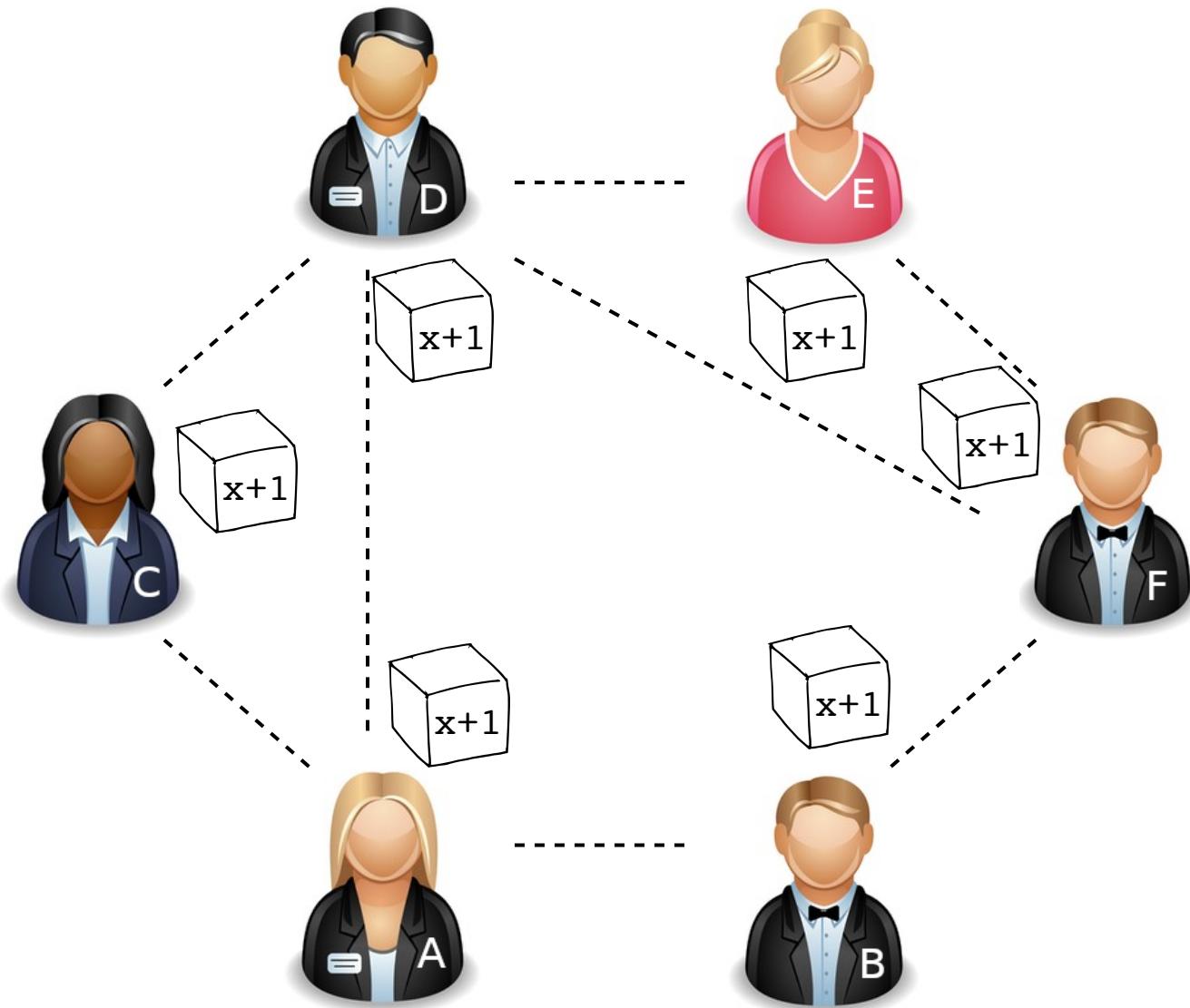
Yo voy a
elegir a una
persona

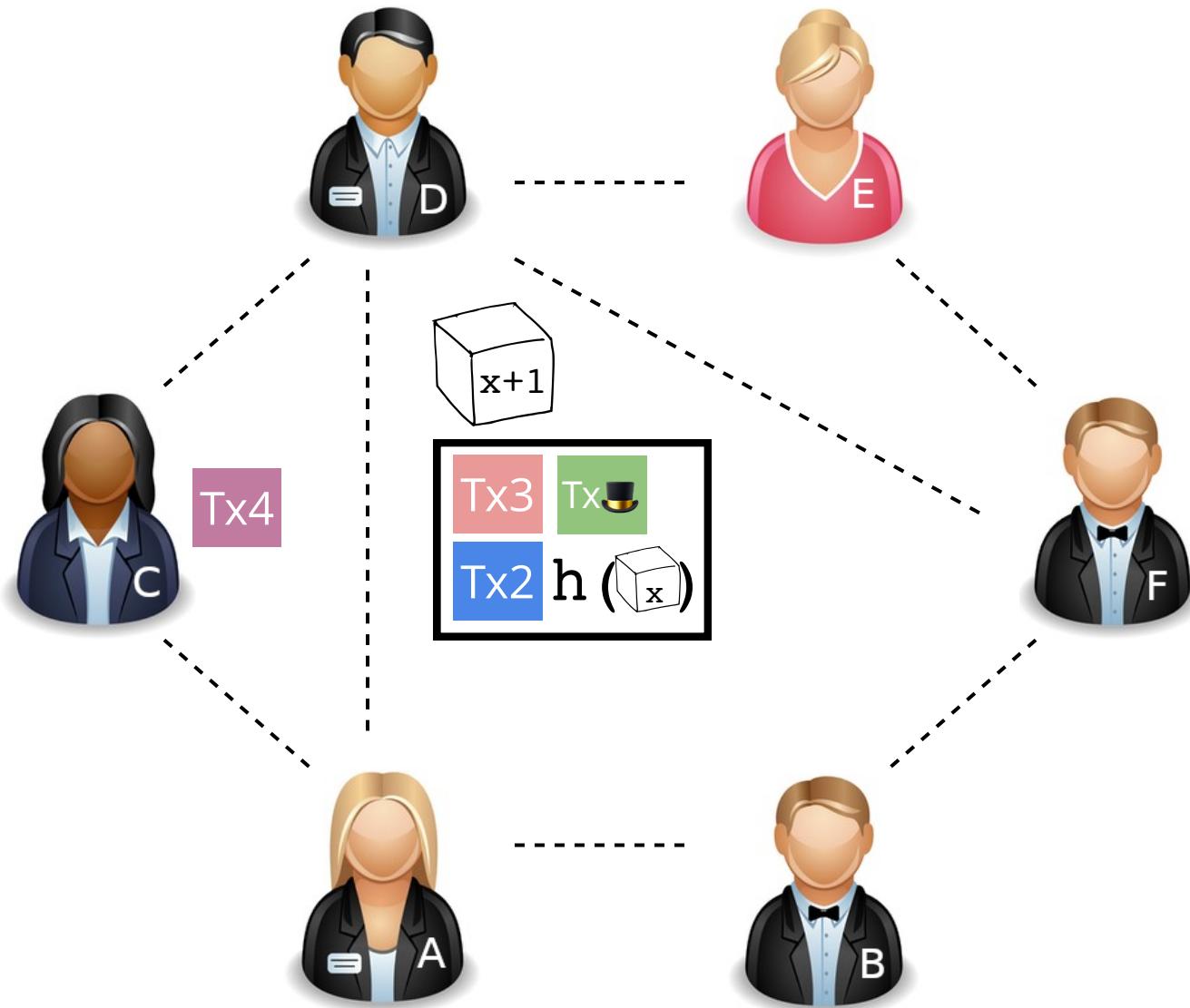
Monto fijo,
a quién
quieras

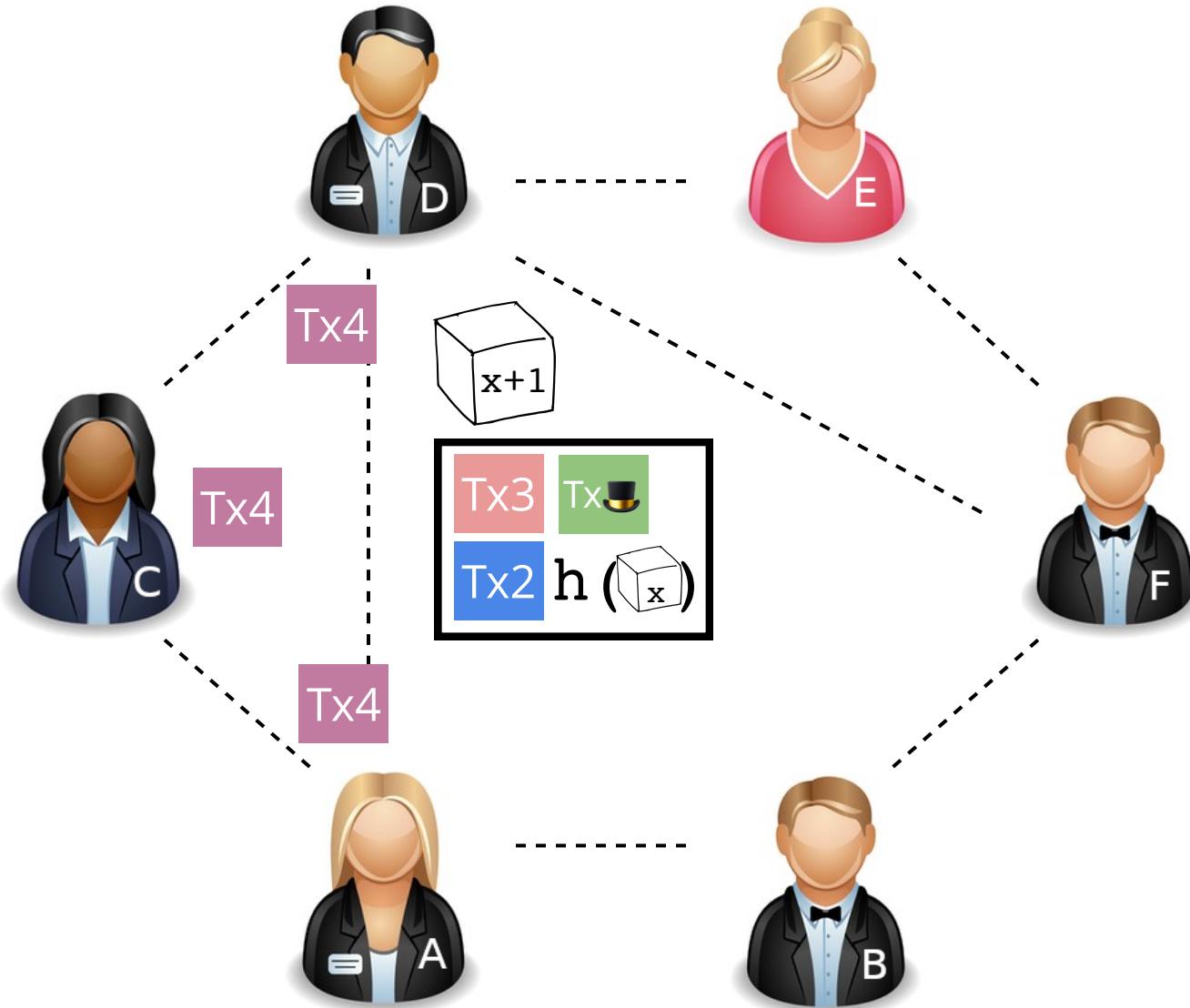


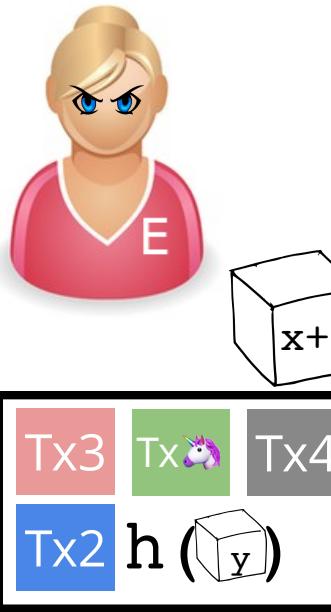








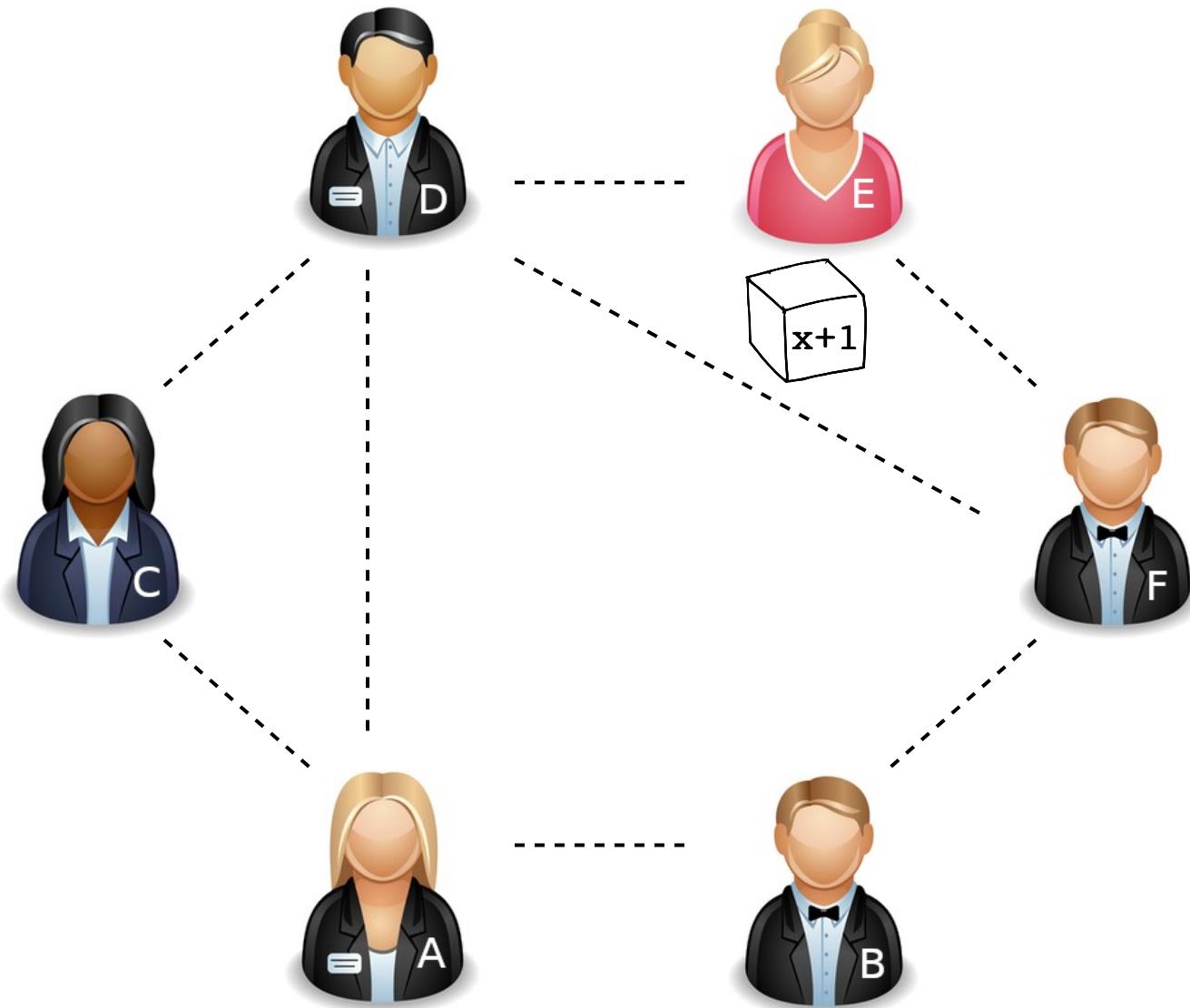




Yo voy a
elegir a una
persona

Monto fijo,
a quién
quieras

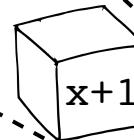
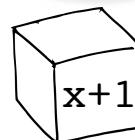
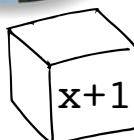


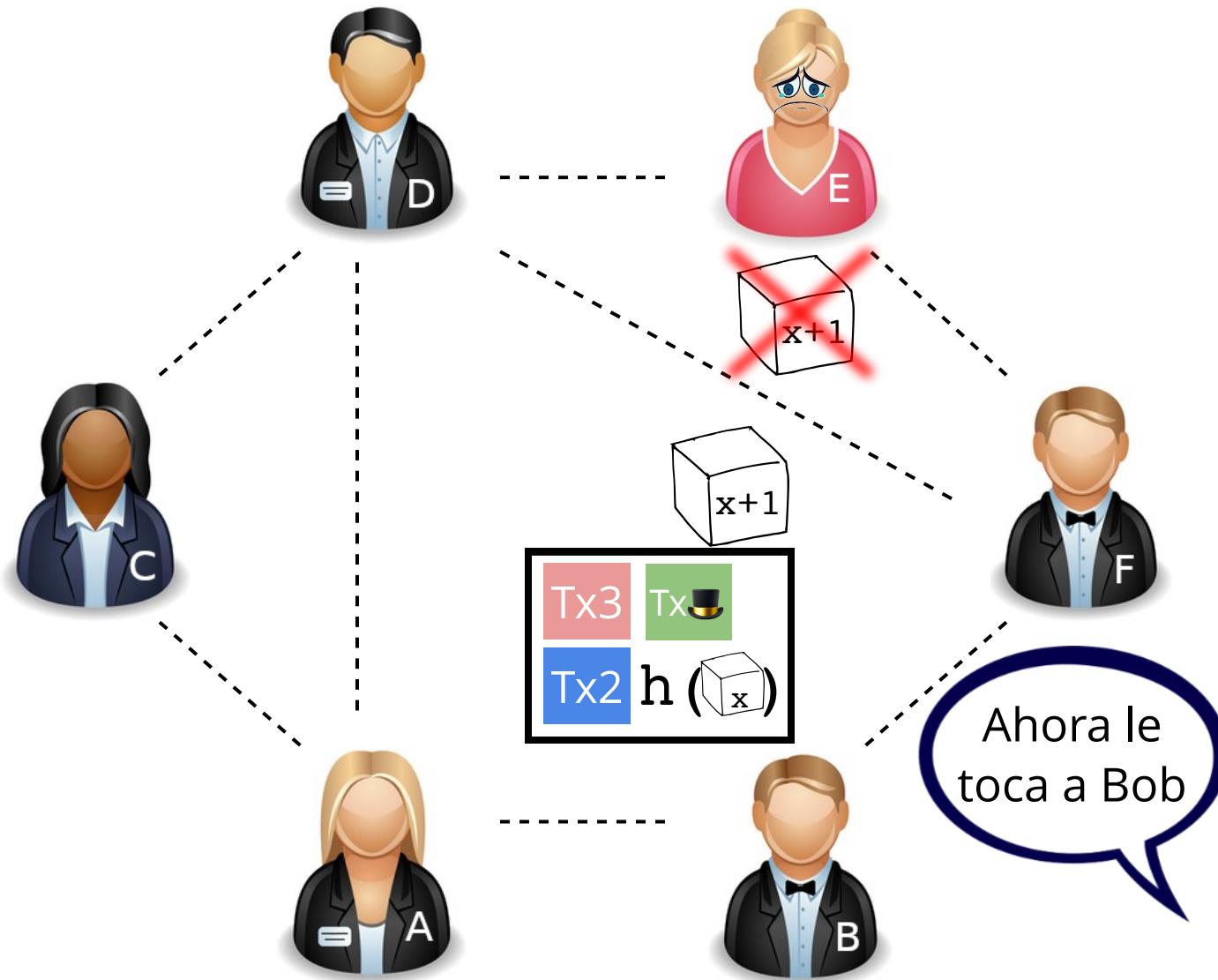


¡No me
salga con
esto vecina!



¡Nos está
intentando
estafar!







Todo bien Satoshi
pero... na que ver que
elijas tanto a tus
amigos



¡Siii a mi nunca
me ha elegido!

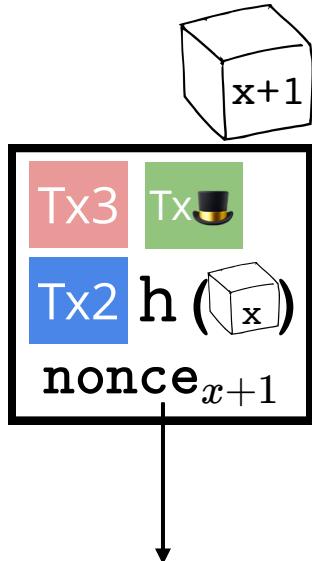


Ya, hagámosla
bien

¿Podemos elegir "aleatoriamente" a quien genera el siguiente bloque?

¿Cómo nos aseguramos de que la fuente de aleatoriedad sea justa?

Y aunque pudiésemos
¿Aleatoriedad sobre qué?



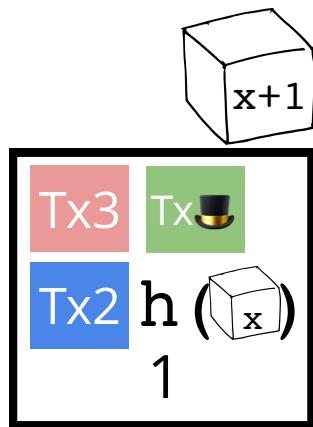
Es un valor *arbitrario*

Este bloque es
"válido" si su
hash parte con
veinte 0's

La 1^a persona
con bloque válido
es "elegida"



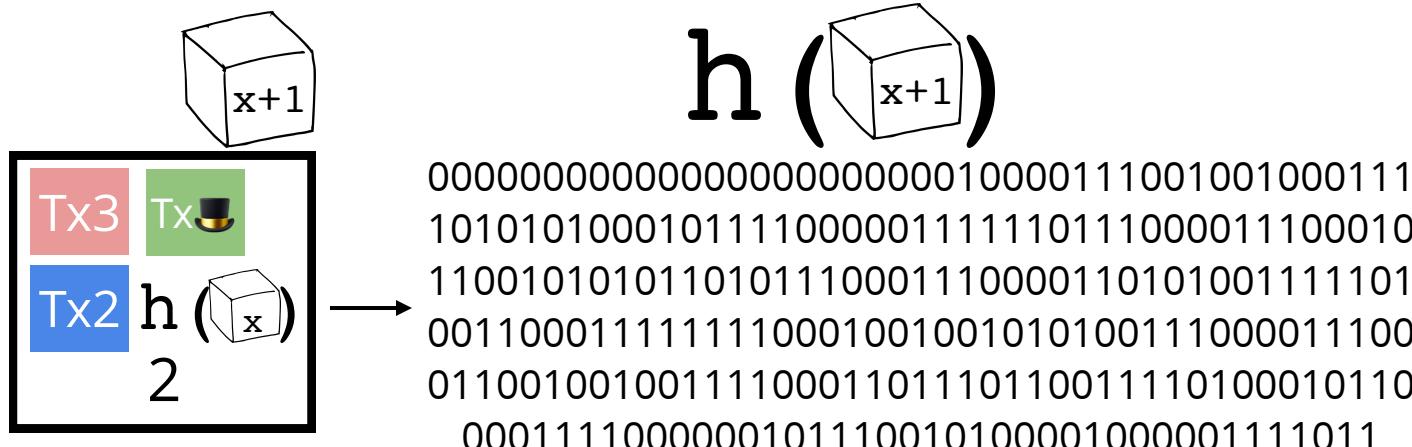
¡Bacán, voy a
tratar de hacer
un bloque válido!

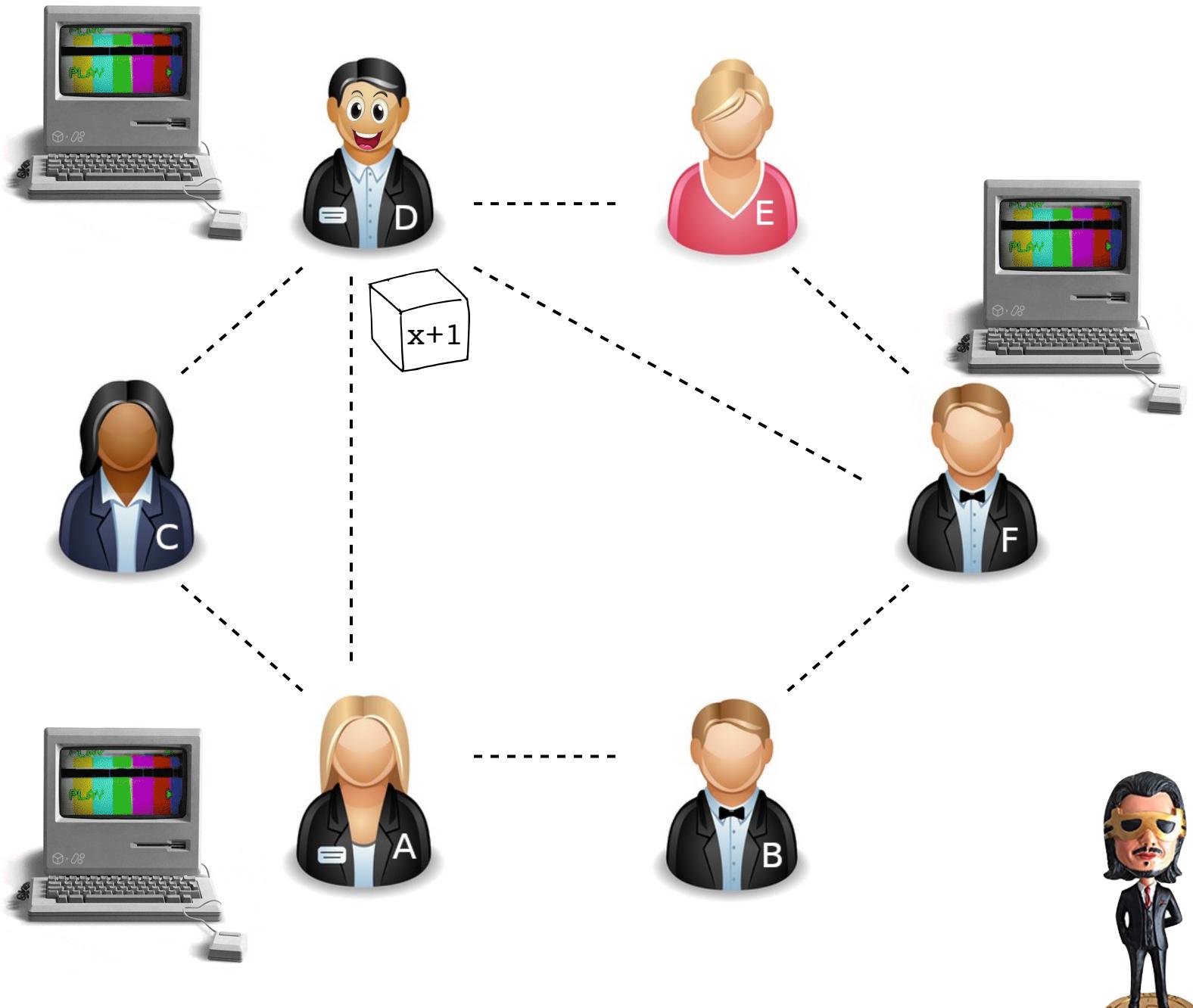


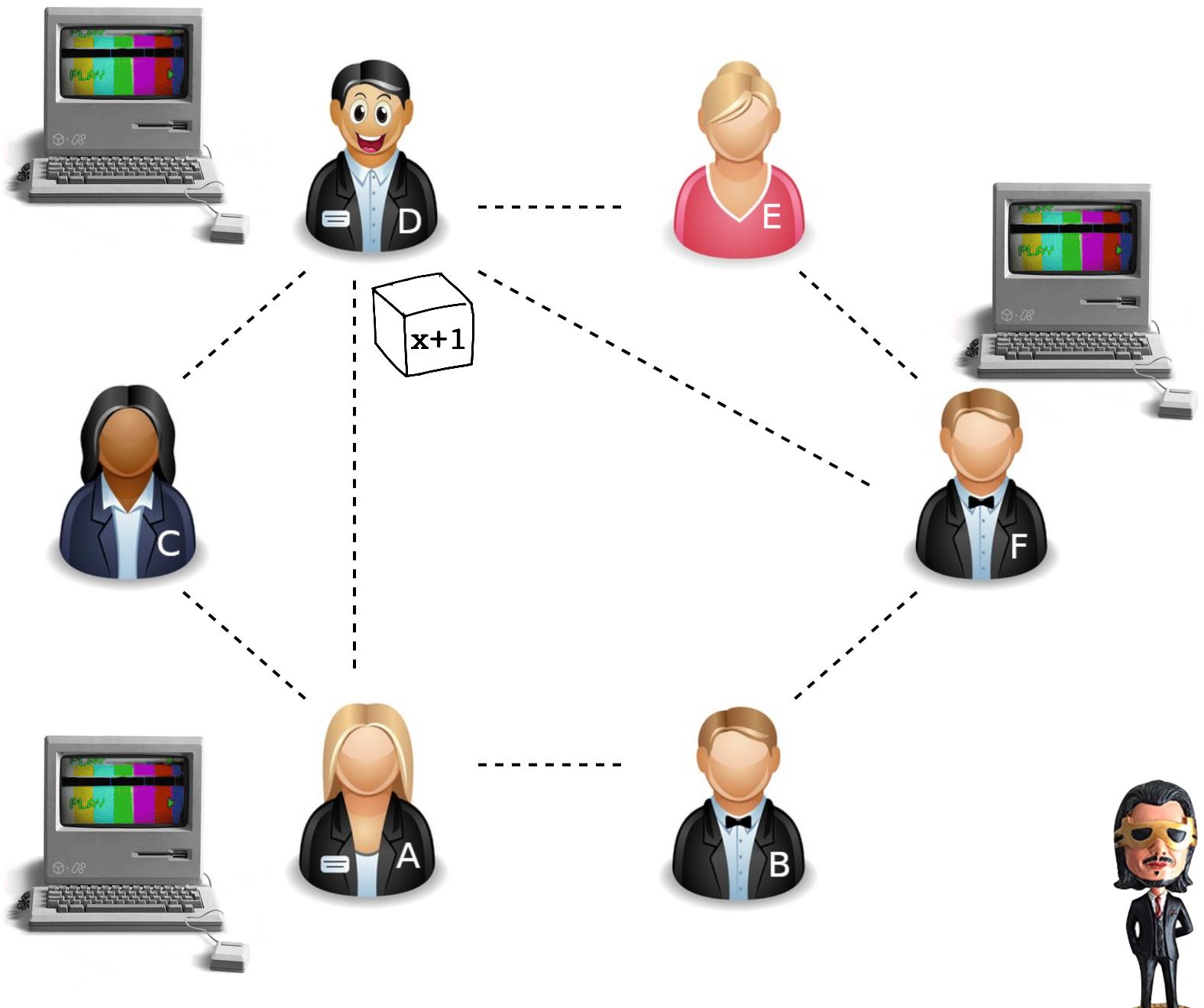
$h (x+1)$

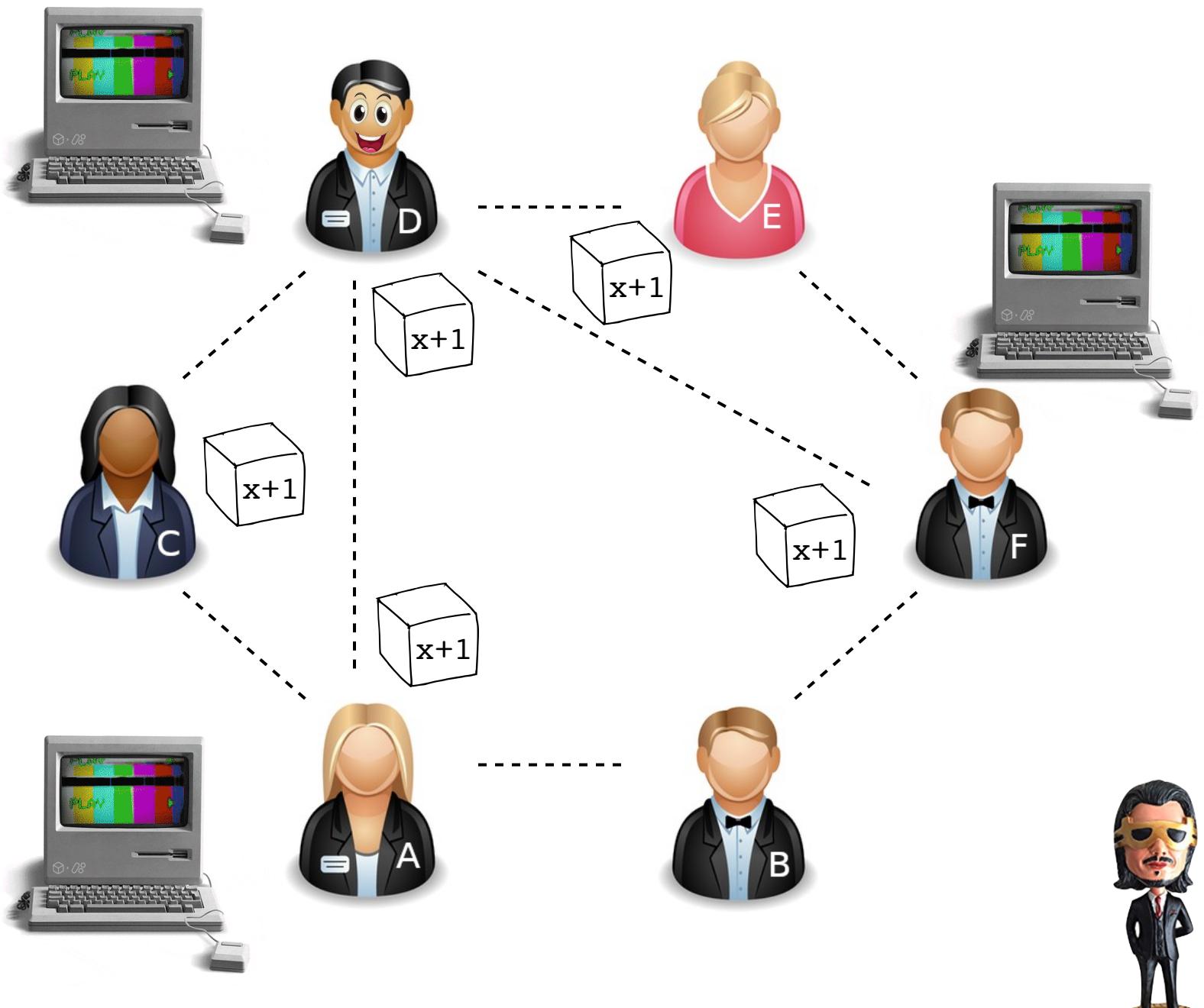
1101011100010001100000010000111001001000111
101010100010111100000111110111000011100010
110010101011010111000111000011010100111101
0011000111111100010010010100111000011100
0110010010011110001101110110011110100010110
00011110000001011001010000100000111011

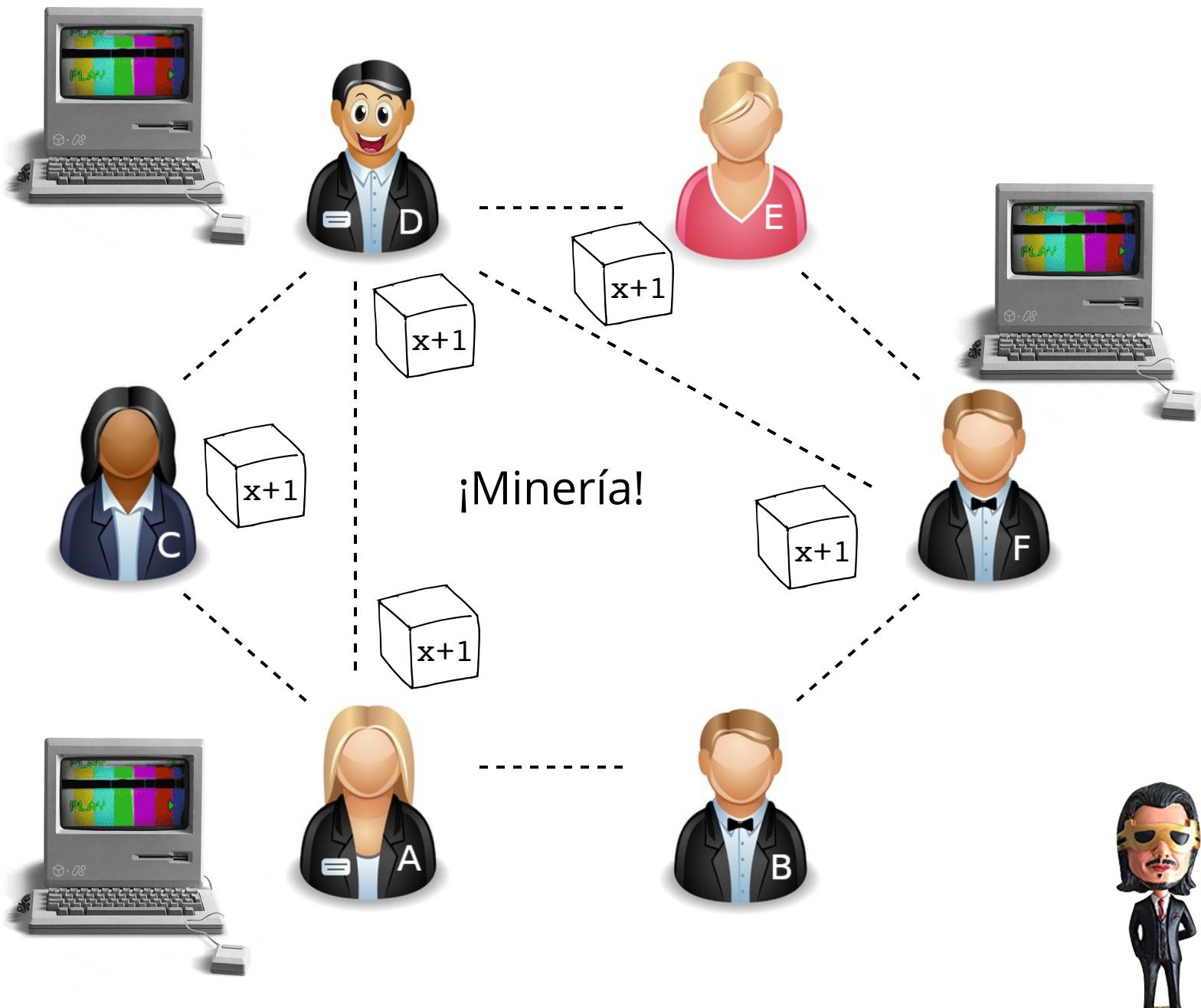
No me funcionó,
¡voy por otro!











Bitcoin: Repaso

¡tengo 320!



¡tengo 250!



Tx3

Input 1: Tx2, Output 1

Output 1: 200 to PK_C

J. Pérez
(Firmado con SK_B)

Transaction 1 (Tx1)

- Output 1: 200 to PK_A
- Output 2: 50 to PK_B
- Output 3: 120 to PK_C
- ...

Tx2

Input 1: Tx1, Output 1

Output 1: 200 to PK_B

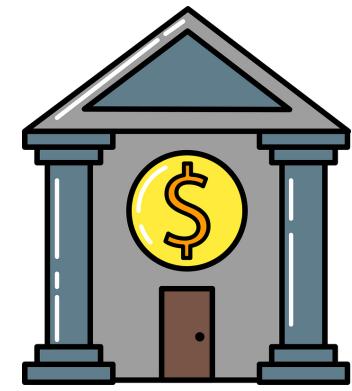
J. Pérez
(Firmado con SK_A)





BankCoin

1. La transacción inicial Tx1, se respeta igual que antes
2. Una transacción es una secuencia de pagos que llega a Tx1, igual que antes
3. El banco publicará en un blockchain firmado todas las transacciones válidas



SK

PK

Input 1: Tx1, Output 1

Output 1: 200 to PK_B

Tx2

(Firmado con SK_A)

Input 1: Tx1, Output 2

Output 1: 50 to PK_c

Tx3

(Firmado con SK_B)

J. Huerta (Firmado con SK_{Bank})

Input 1: Tx1, Output 3

Input 2: Tx3, Output 1

Output 1: 170 to PK_D

Tx4

J. Huerta (Firmado con SK_C)

Input 1: Tx2, Output 1

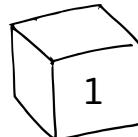
Output 1: 200 to PK_c

Tx5

J. Huerta (Firmado con SK_B)

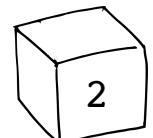
Transaction 1
(Tx1)

- Output 1: 200 to PK_A
- Output 2: 50 to PK_B
- Output 3: 120 to PK_c
- ...



$h(1)$

J. Huerta (Firmado con SK_{Bank})

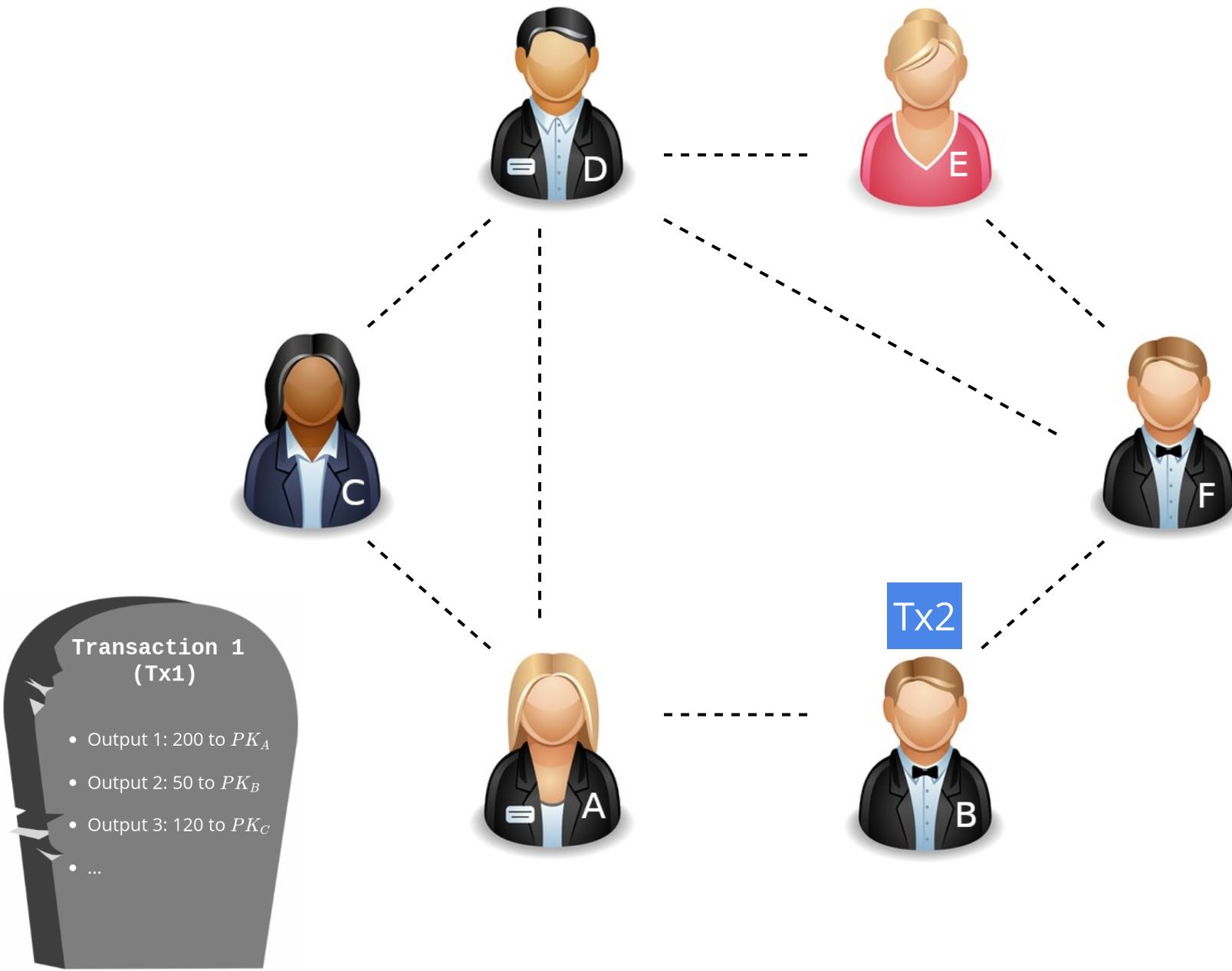


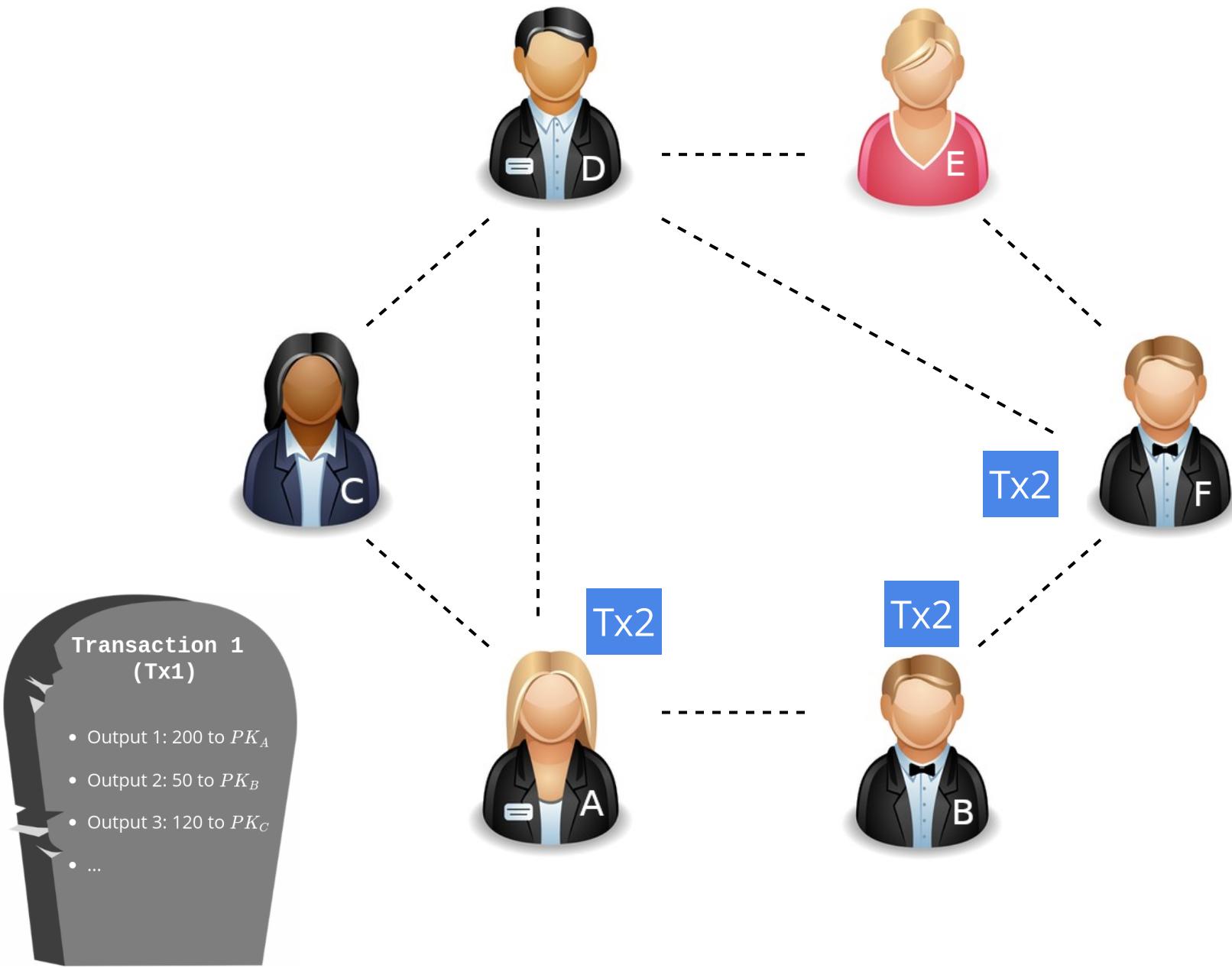
bitcoin

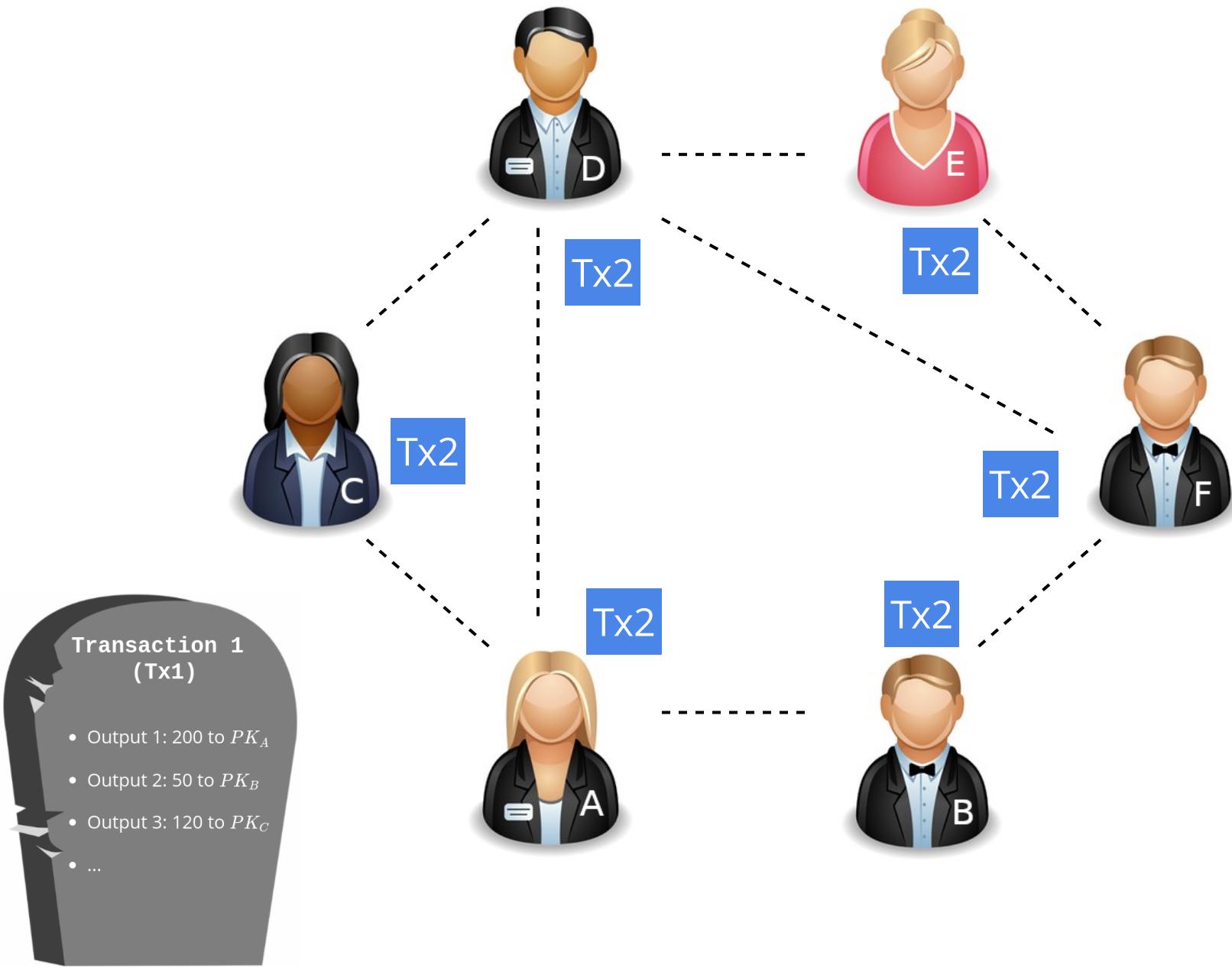
- La transacción inicial se respeta igual que antes, y le llamaremos bloque 0
- Si quieres pagar, mandale transacciones a tus vecinos
- Si recibes una nueva transacción, mándasela a tus vecinos

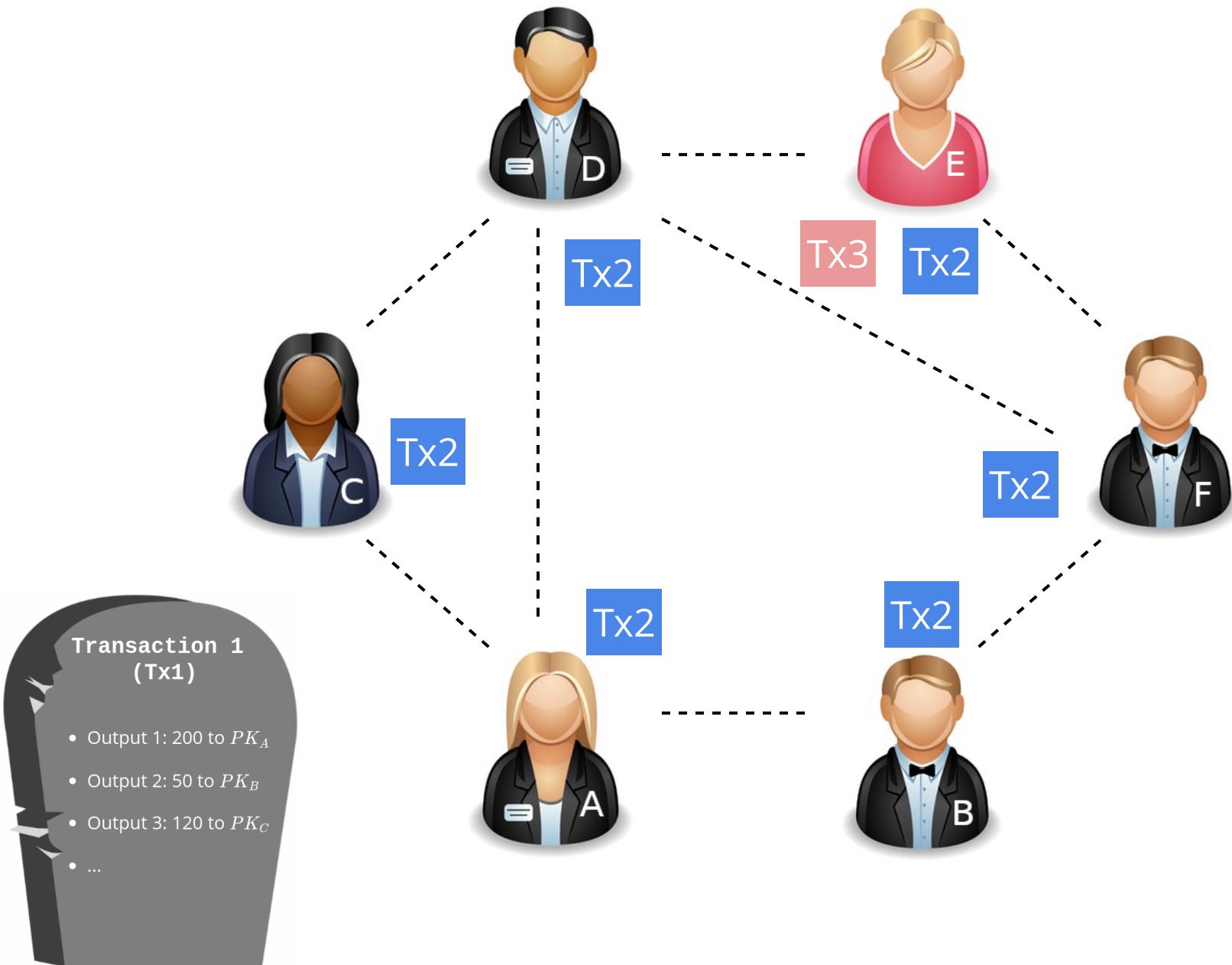
Nada complicado por ahora...



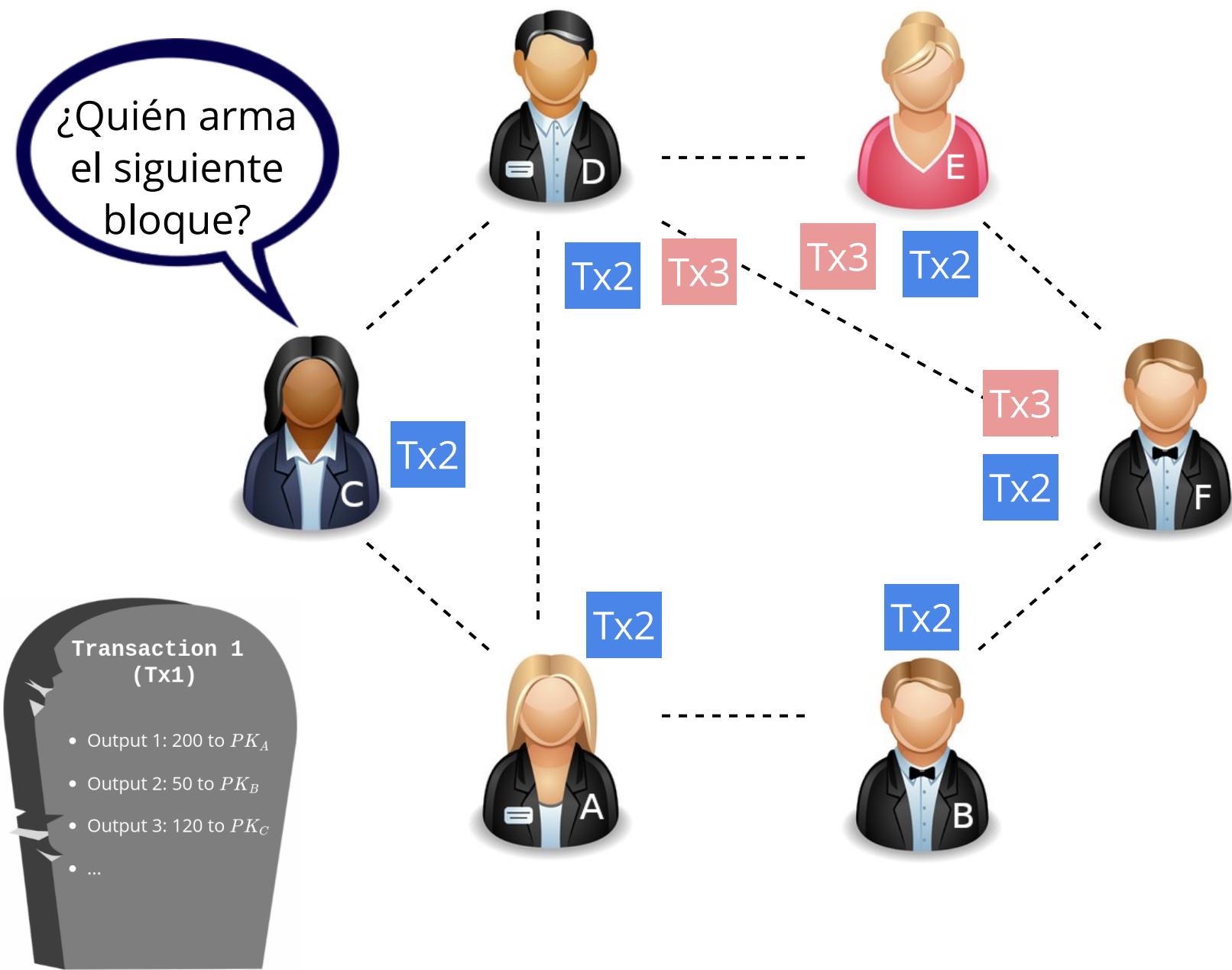






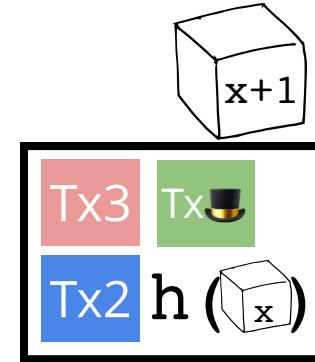


¿Quién arma el siguiente bloque?



- Cuando te elijan, forma un bloque con lo que tengas y mándalo a tus vecinos
- Si recibes un bloque, verifica que sea correcto y mándalo a tus vecinos
- ...

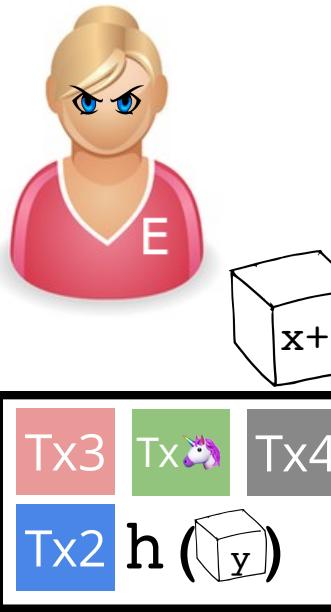
¿Por qué
haría yo
esto bien?



Yo voy a
elegir a una
persona

Monto fijo,
a quién
quieras

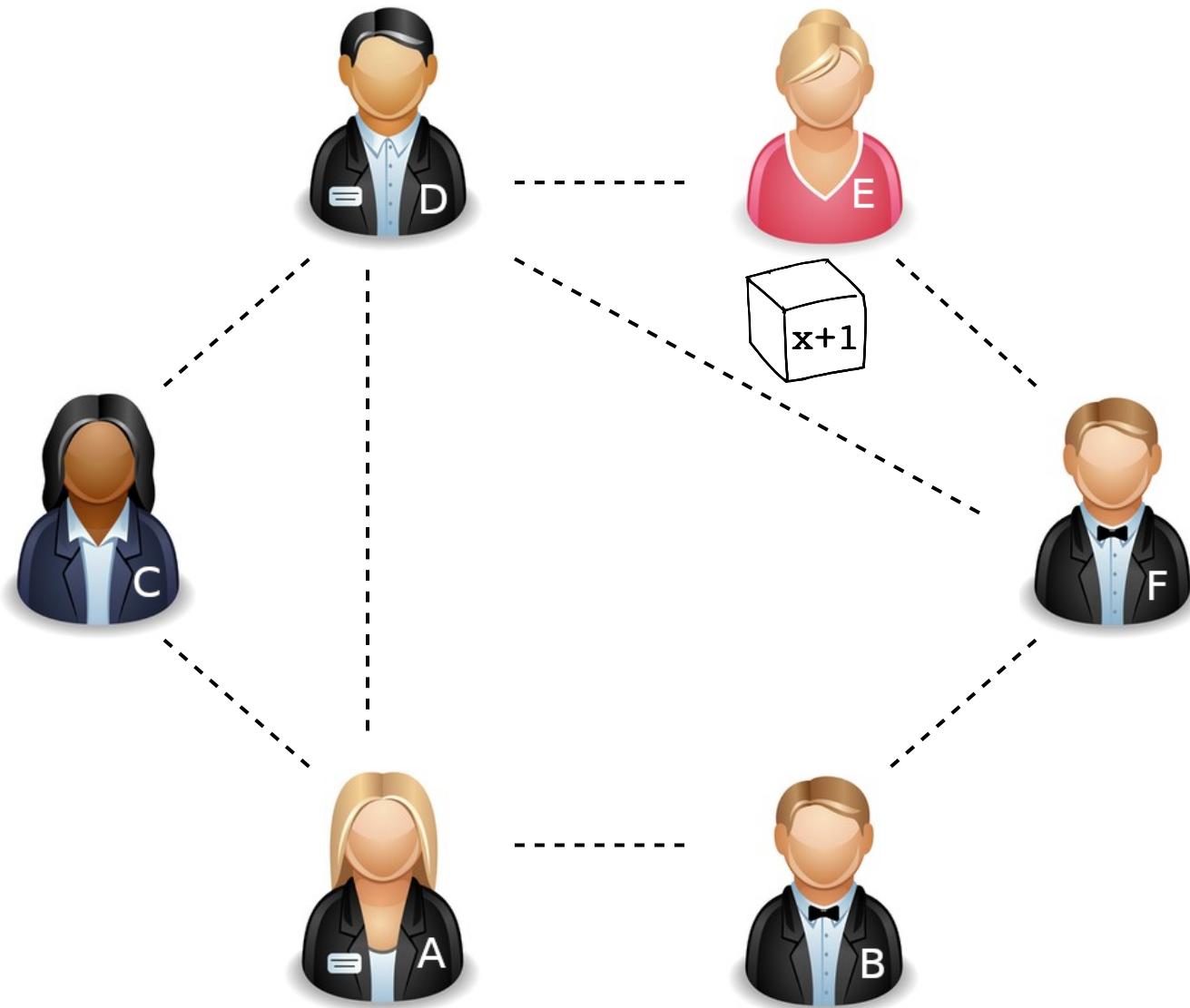




Yo voy a
elegir a una
persona

Monto fijo,
a quién
quieras

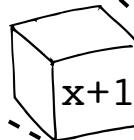
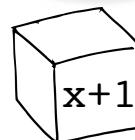
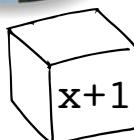


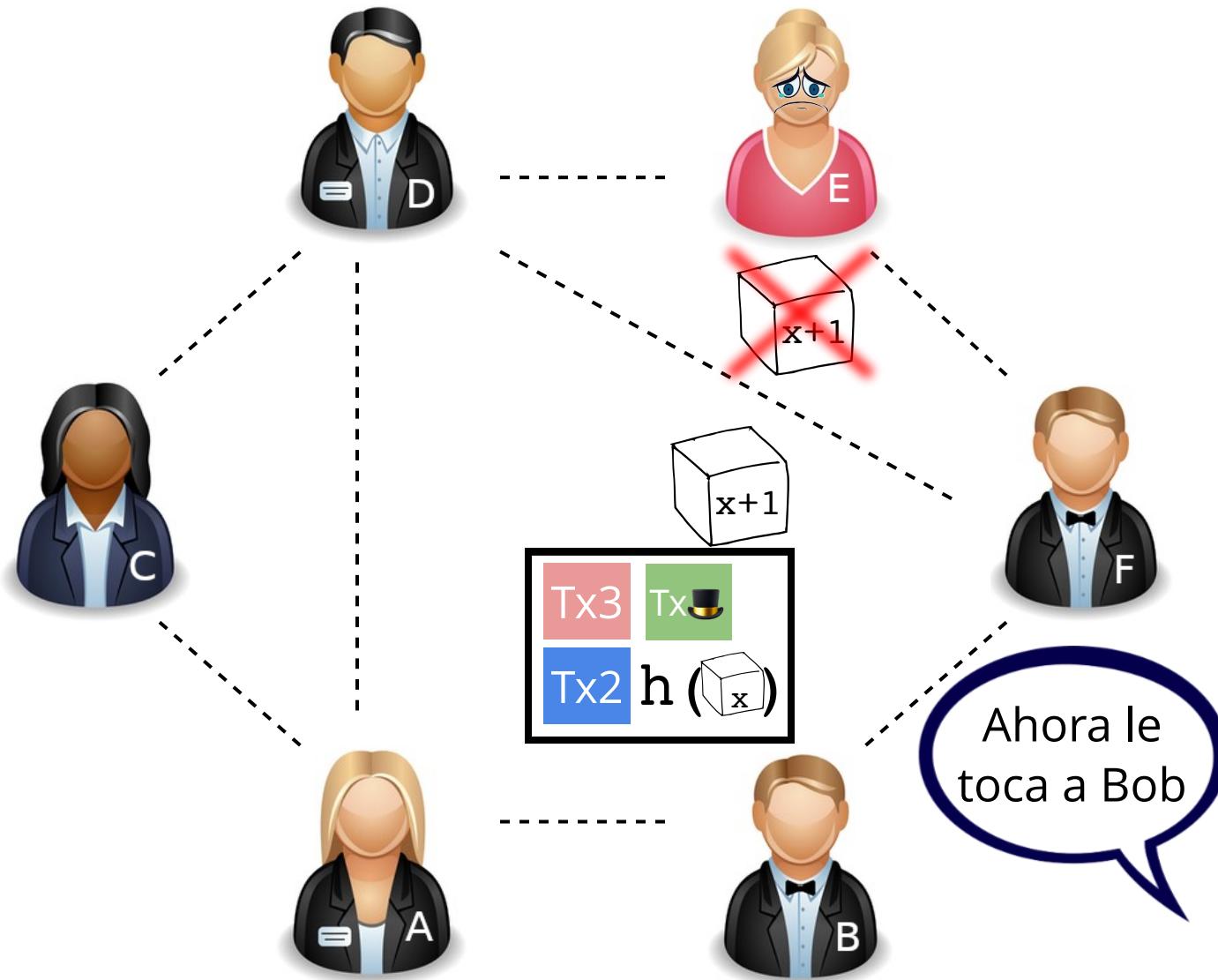


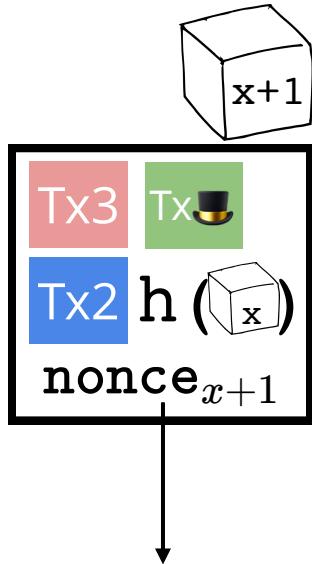
¡No me
salga con
esto vecina!



¡Nos está
intentando
estafar!







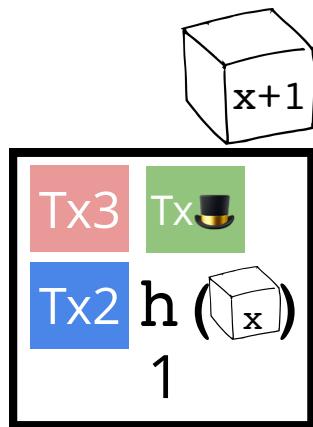
Es un valor *arbitrario*

Este bloque es
"válido" si su
hash parte con
veinte 0's

La 1^a persona
con bloque válido
es "elegida"



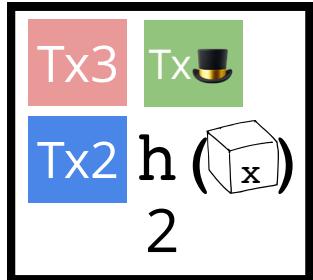
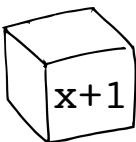
¡Bacán, voy a
tratar de hacer
un bloque válido!



$h(x+1)$

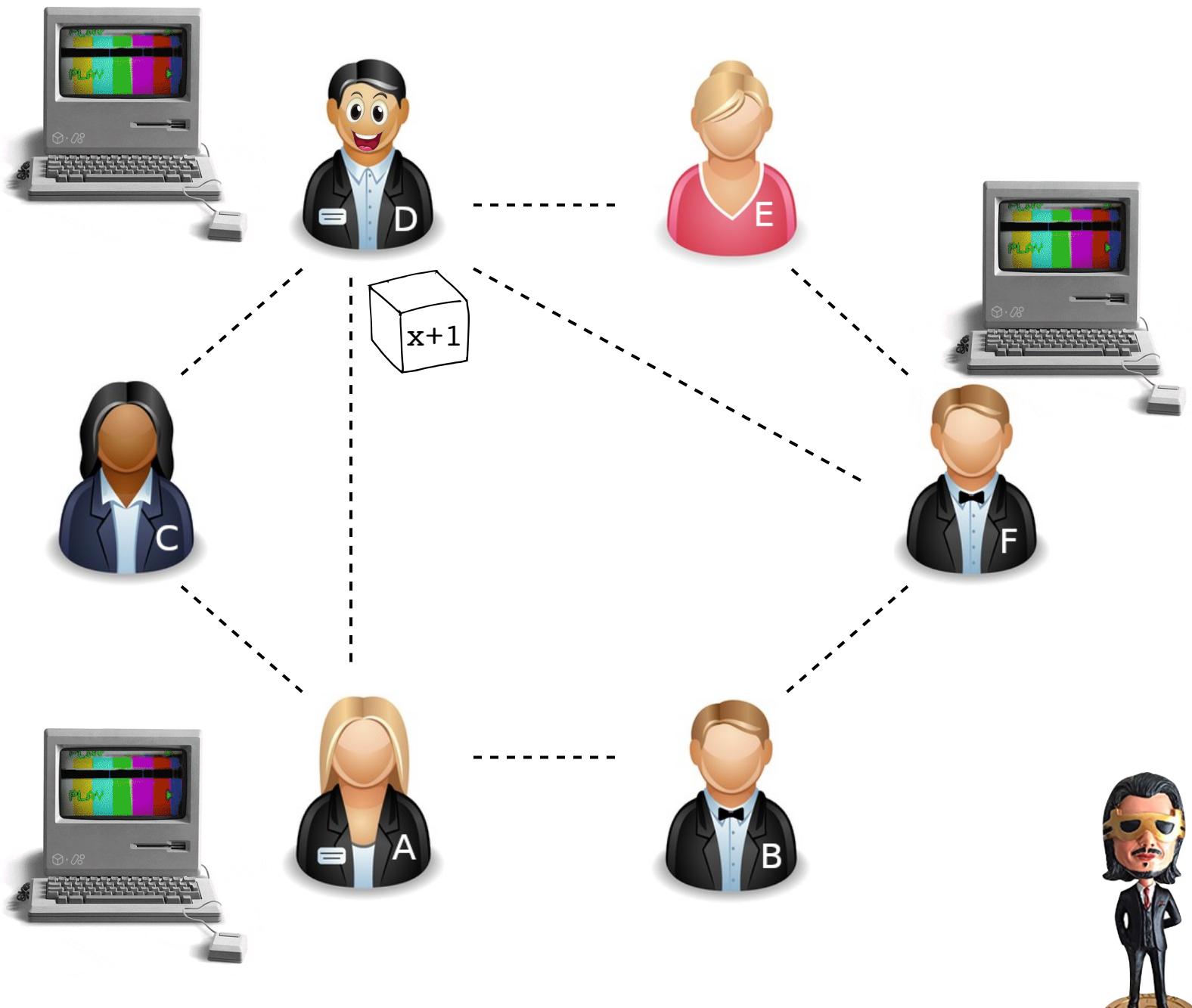
1101011100010001100000010000111001001000111
101010100010111100000111110111000011100010
110010101011010111000111000011010100111101
0011000111111100010010010100111000011100
0110010010011110001101110110011110100010110
00011110000001011001010000100000111011

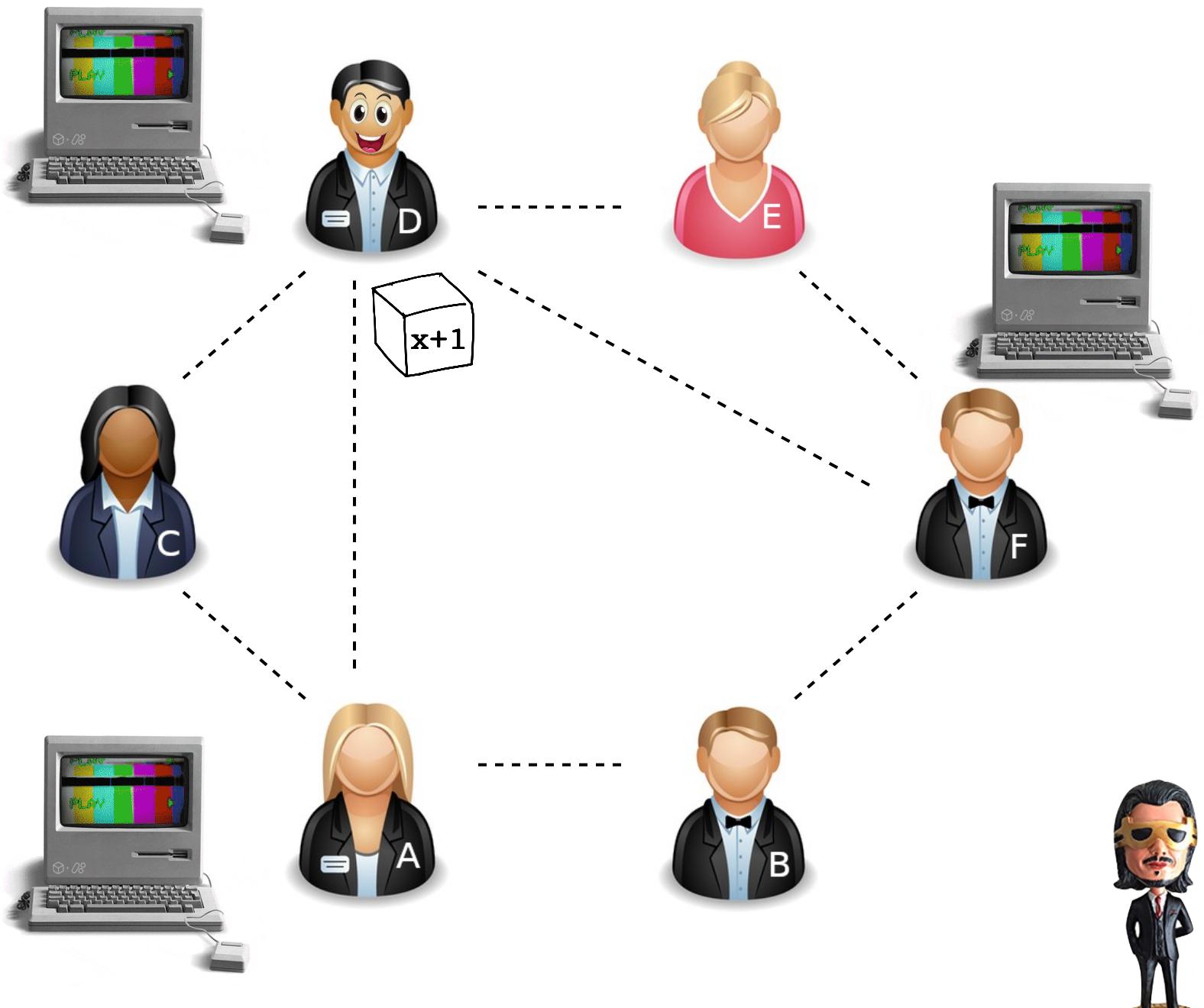
No me funcionó,
¡voy por otro!

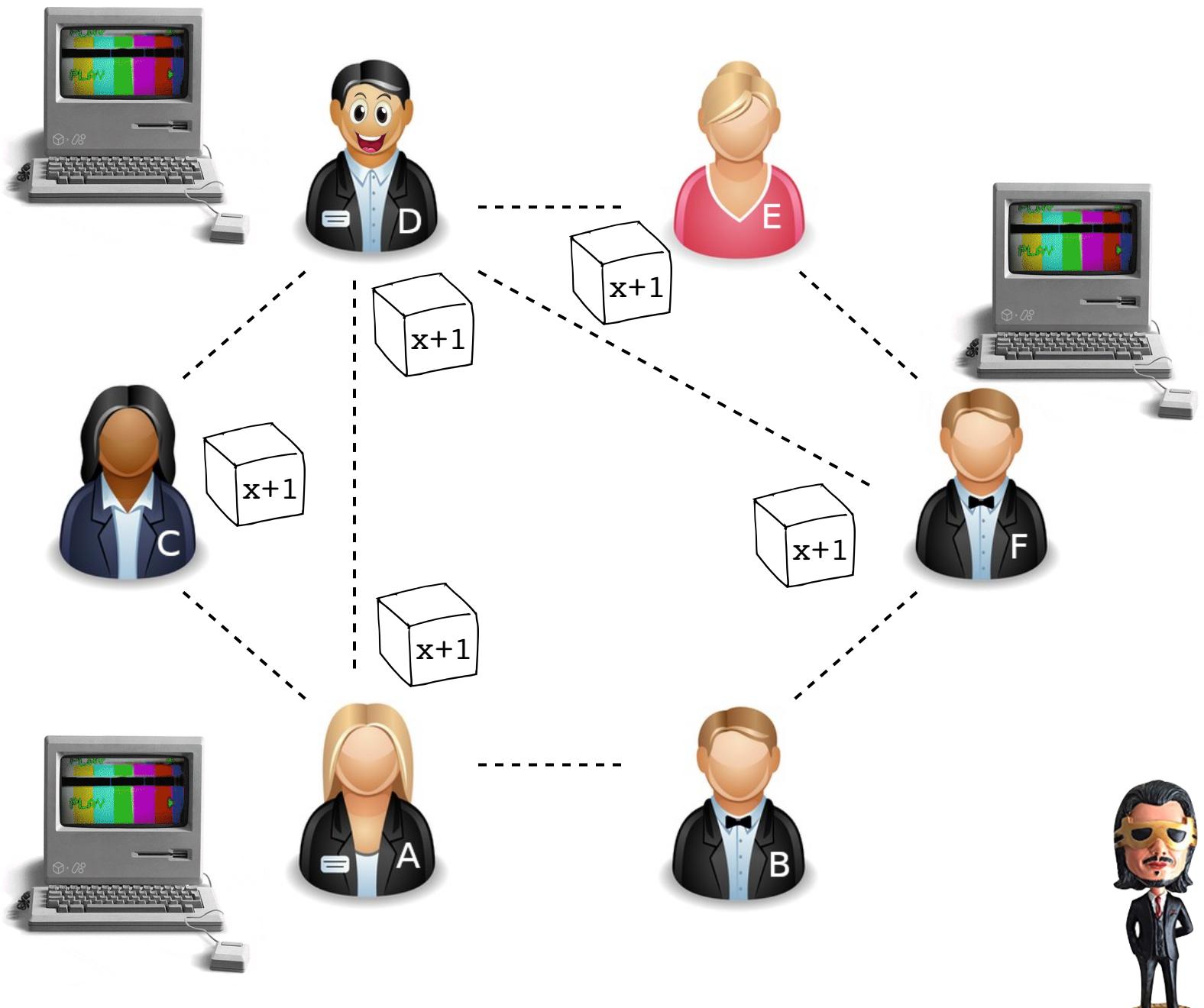


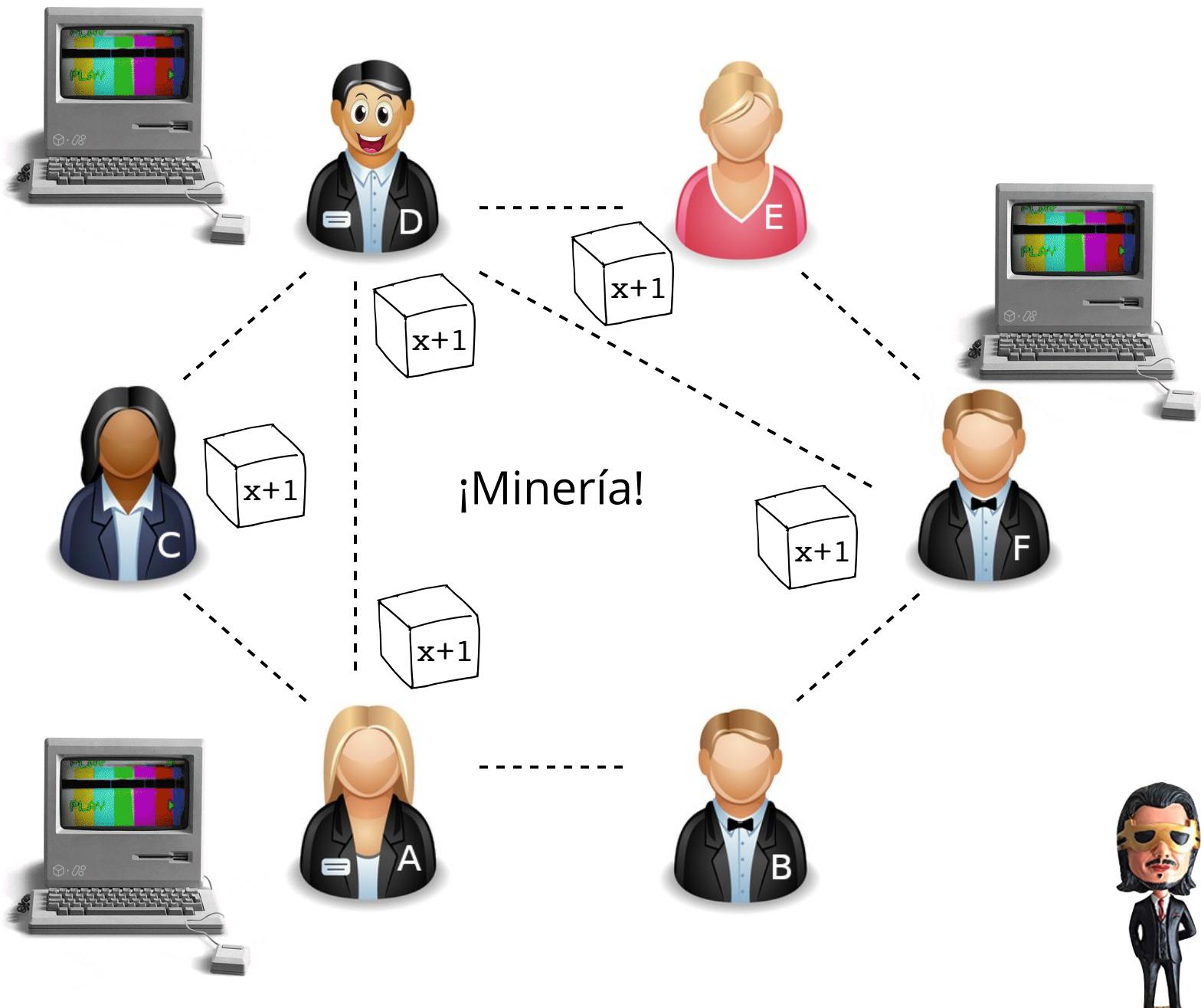
$h(x+1)$

00000000000000000000000010000111001001000111
101010100010111100000111110111000011100010
1100101010110101110001110000110101001111101
0011000111111100010010010100111000011100
0110010010011110001101110110011110100010110
000111100000010110010100001000001111011

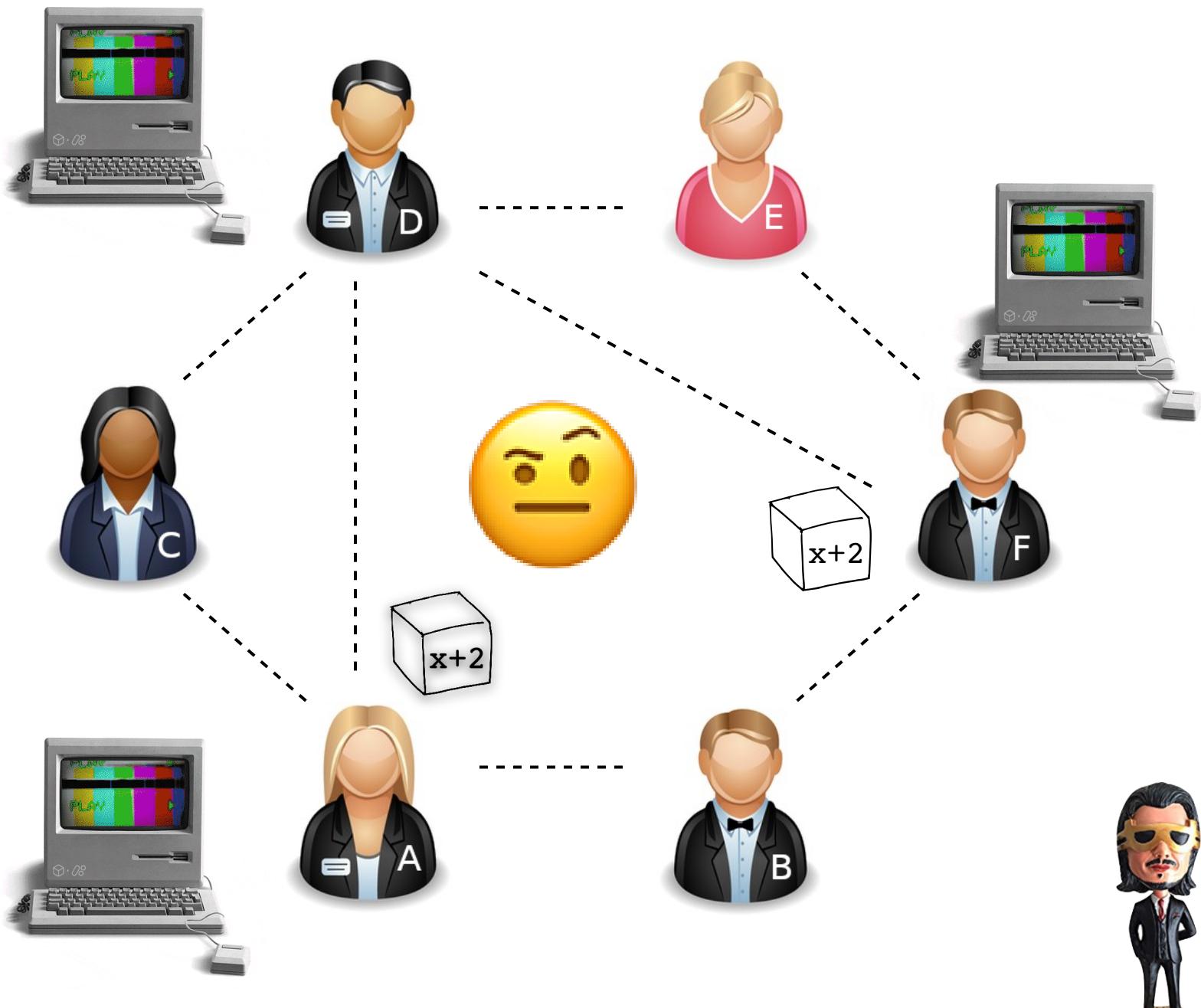






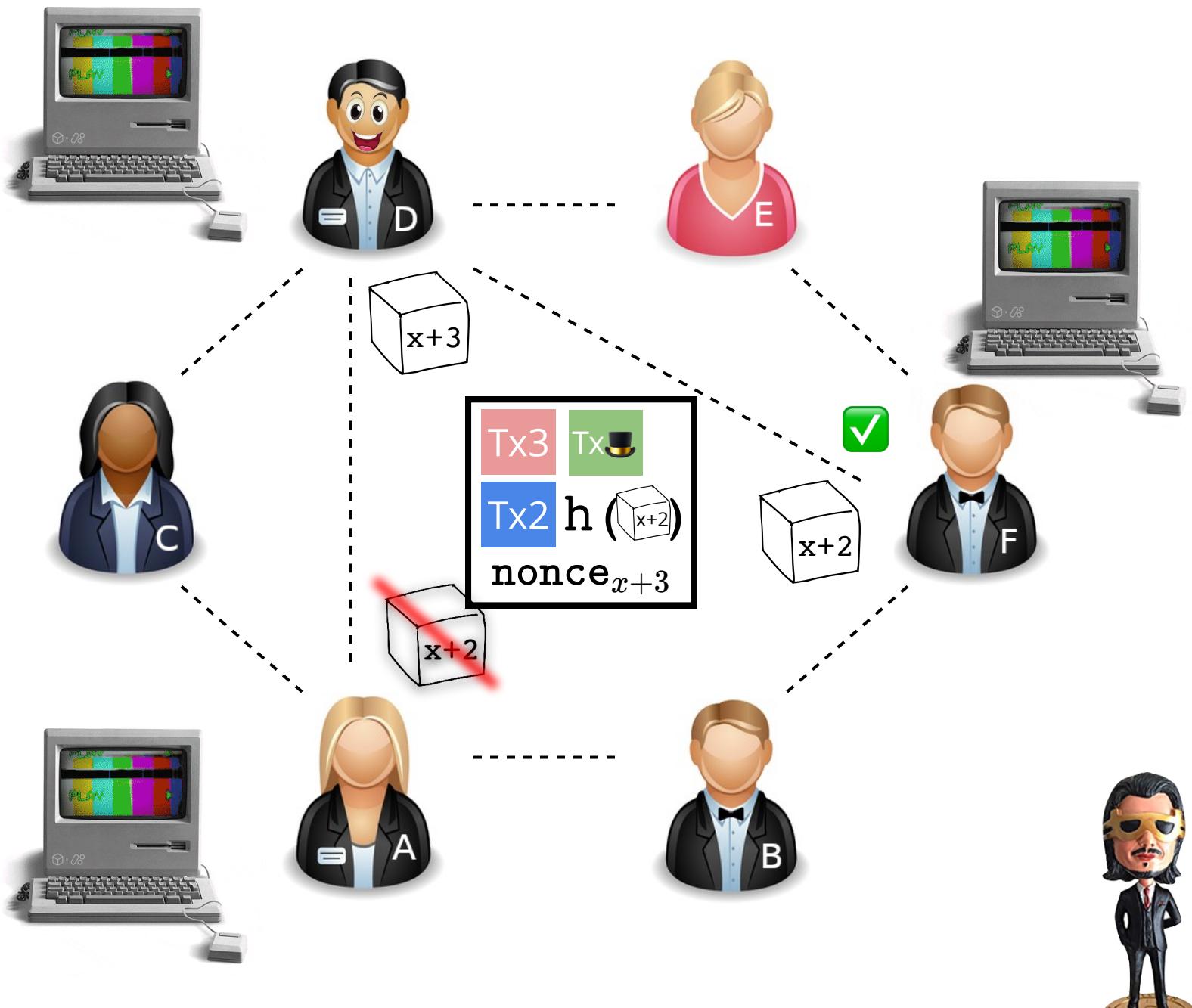


Fin del repaso

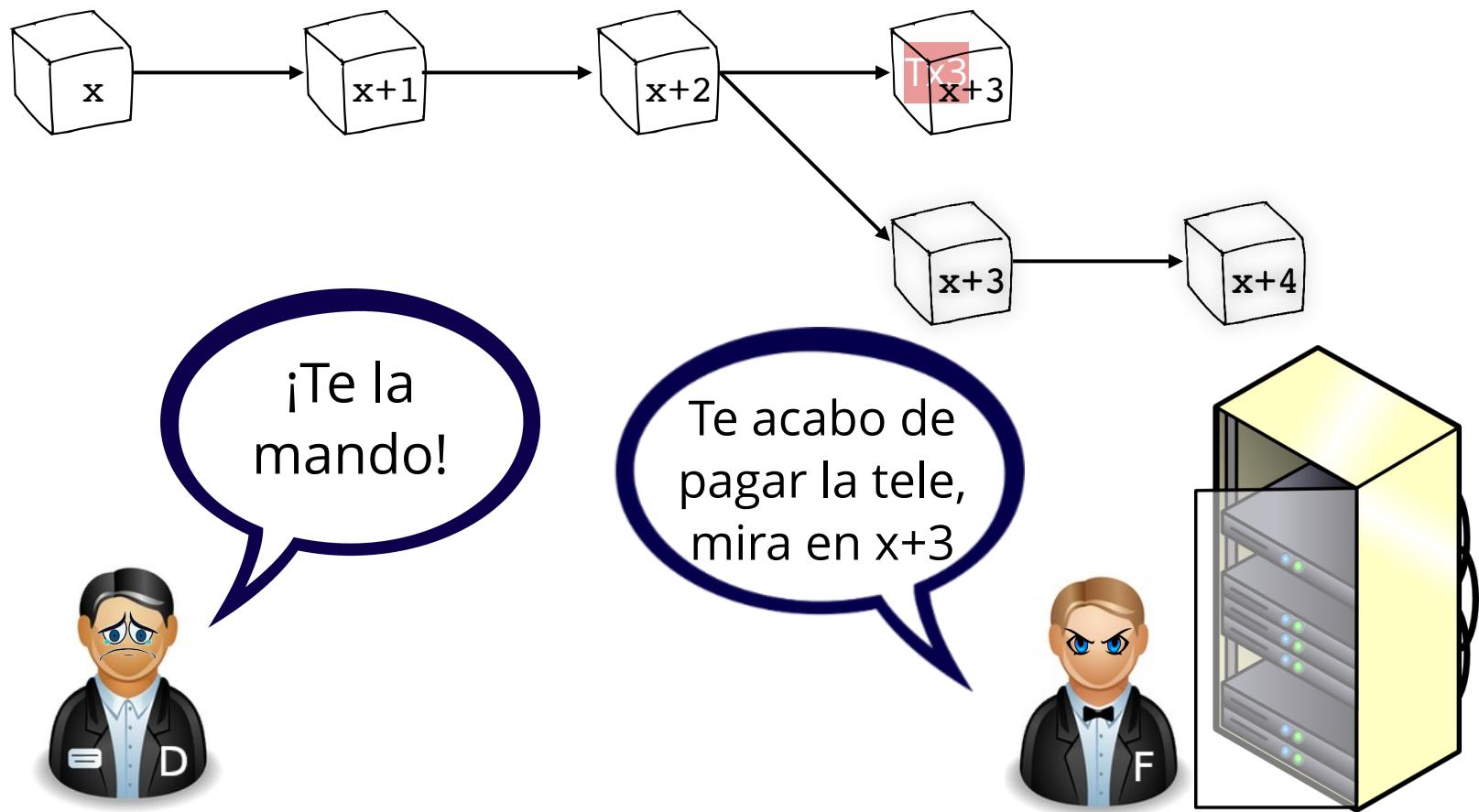


Mantengan los dos y sigan
trabajando sobre el quieran...
Cuando salga otro, sigan la
"cadena" más larga





51% Attack



Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshi@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

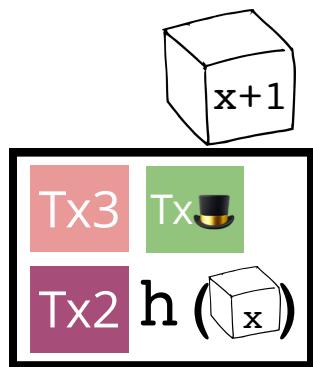
Bitcoin en la práctica

Bloques

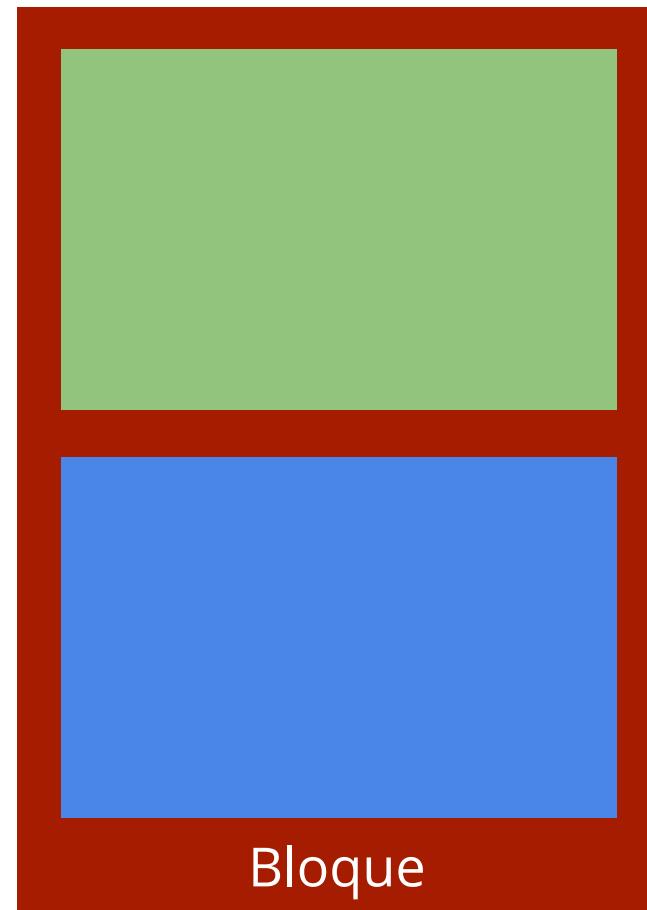
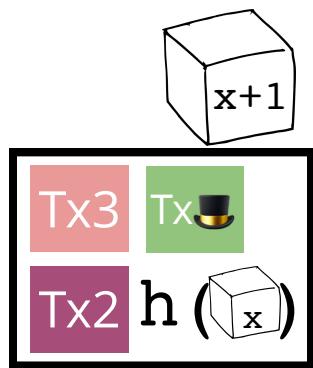
Blockchain de Bitcoin

Un bloque

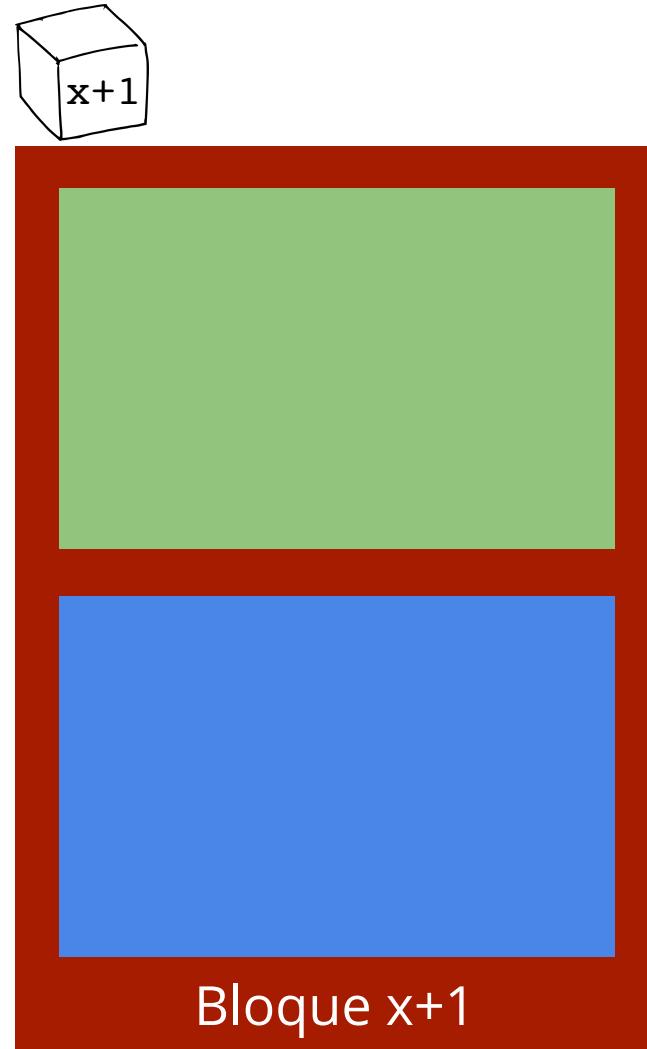
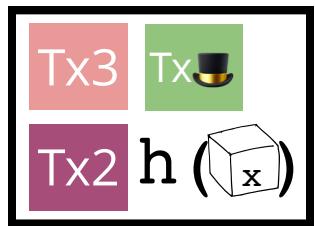
Un bloque



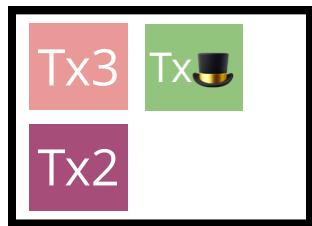
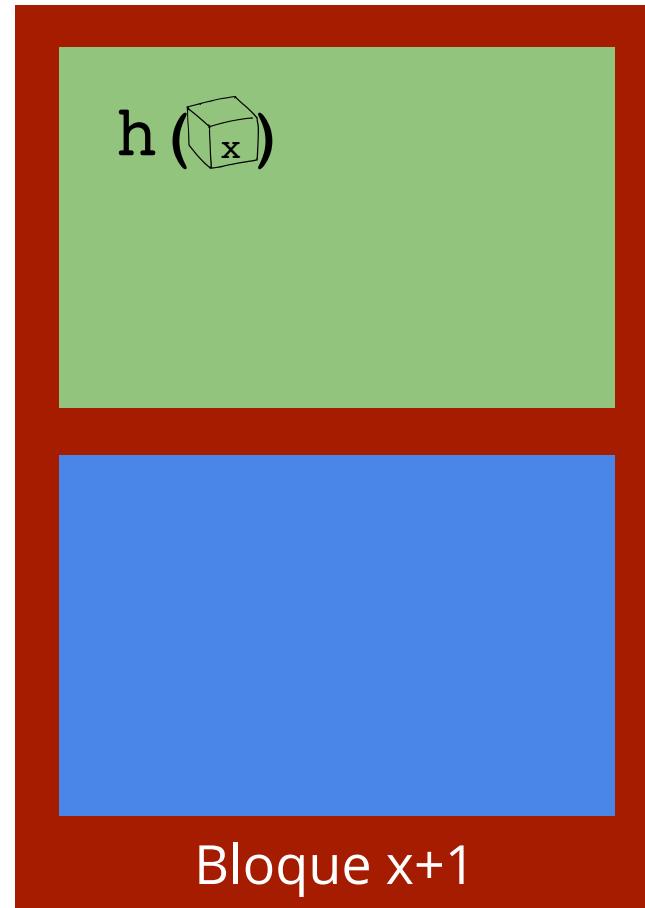
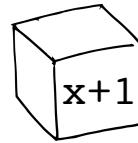
Un bloc



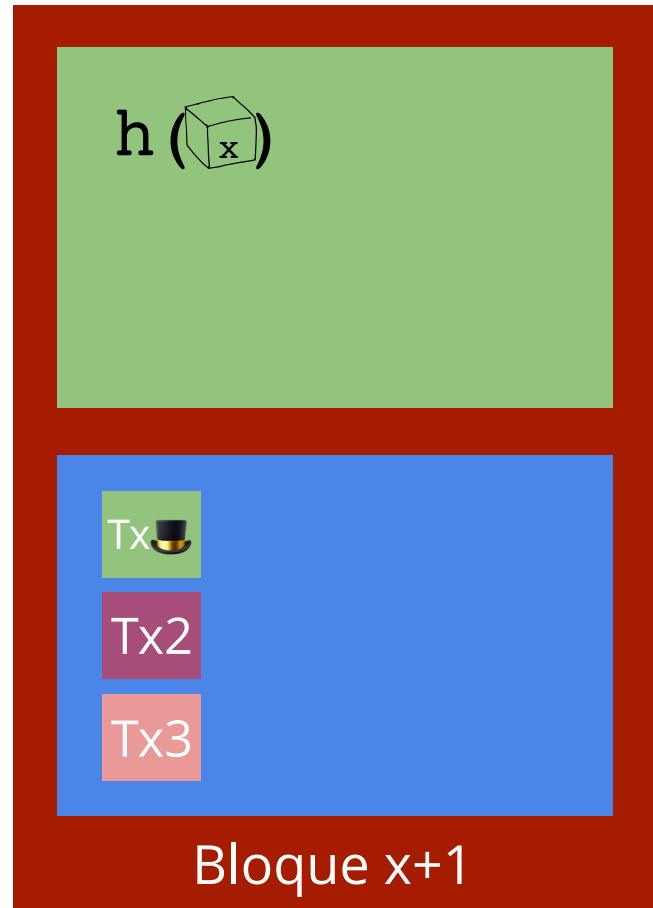
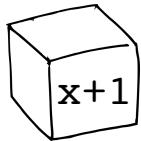
Un bloc



Un bloque



Un bloc

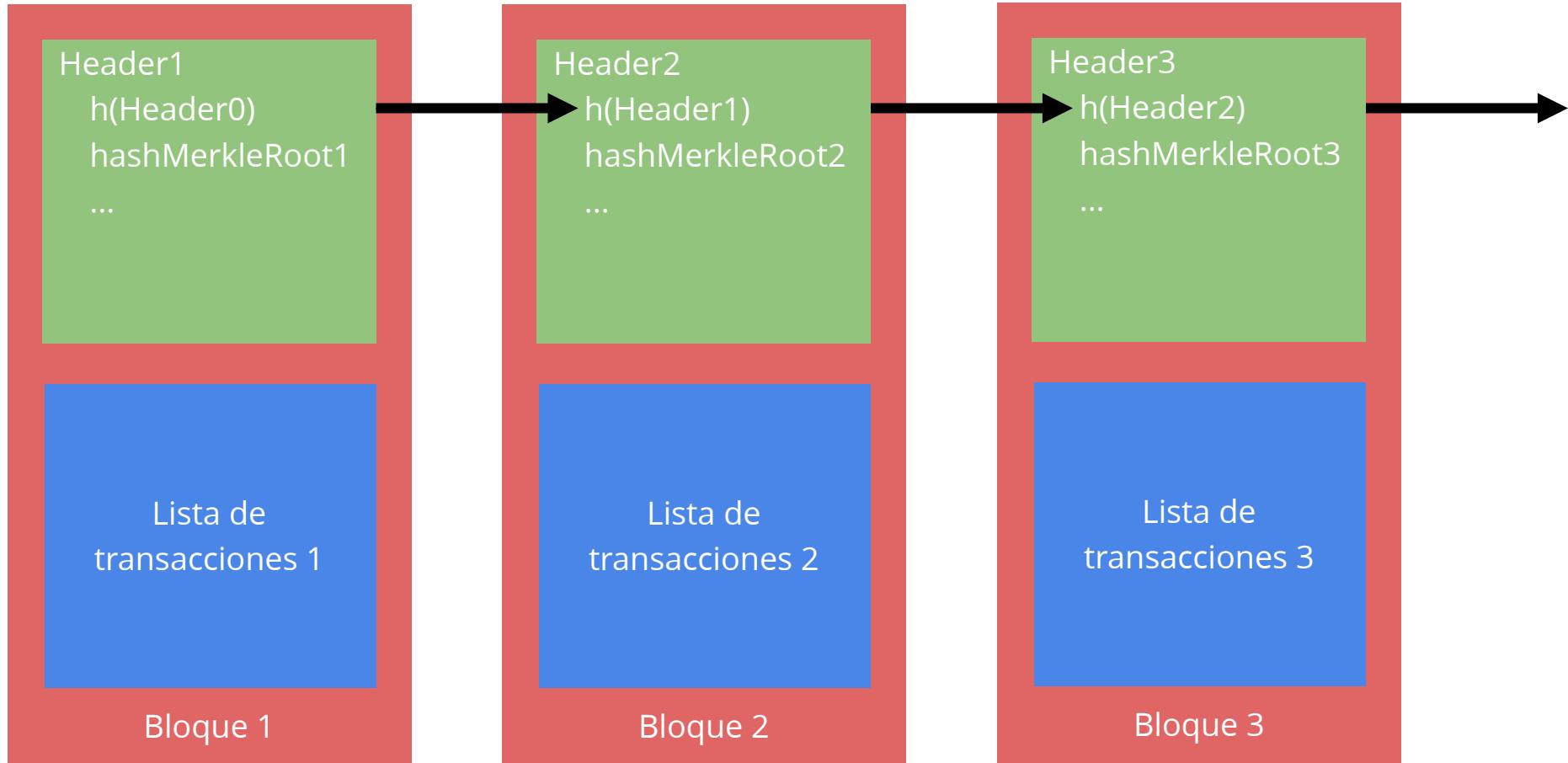


Header

SHA-256(SHA-256(...))

version	indica la versión de las reglas de validación de un bloque que deben ser usadas
hashPrevBlock	hash del header del bloque anterior
hashMerkleRoot	raíz del árbol de Merkle para las transacciones del bloque
time	Unix timestamp que indica cuándo fue generado el bloque
bits	dificultad asociada a generar un bloque
nonce	número de 32 bits

El blockchain de Bitcoin



¿Por qué necesitamos árboles de Merkle?

Necesitamos verificar que una transacción es parte de un bloque

- hashPrevBlock no incluye a las transacciones del bloque

Podríamos incluir en el header: $h(\text{Tx1} \hat{\wedge} \text{Tx2} \hat{\wedge} \text{Tx3})$

¿Por qué esto no es una buena idea?

Arboles de Merkle

Tx1

Tx2

Tx3

Tx4

Tx5

Tx6

Arboles de Merkle

$h(Tx1)$

$h(Tx2)$

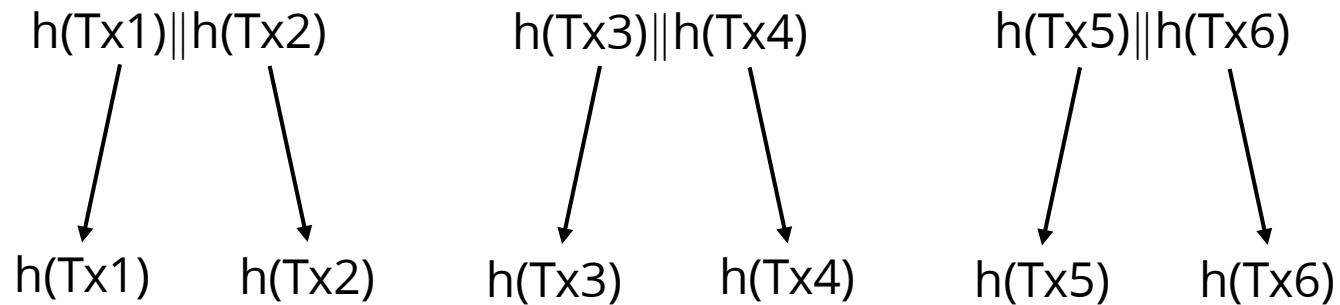
$h(Tx3)$

$h(Tx4)$

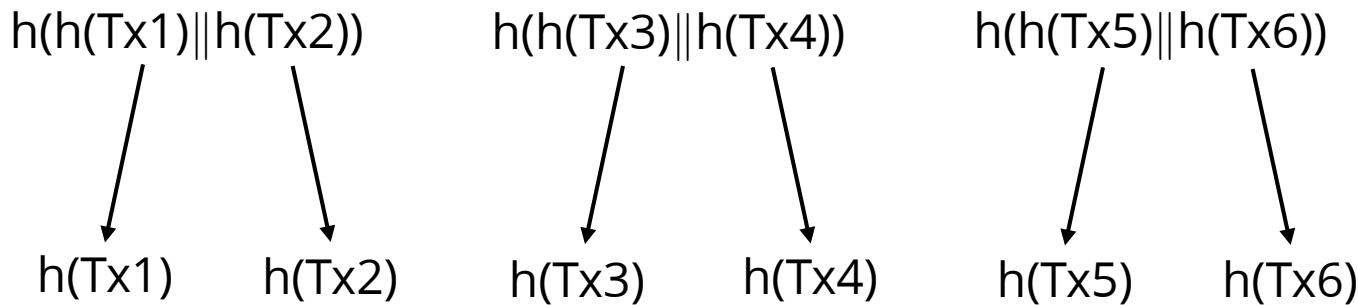
$h(Tx5)$

$h(Tx6)$

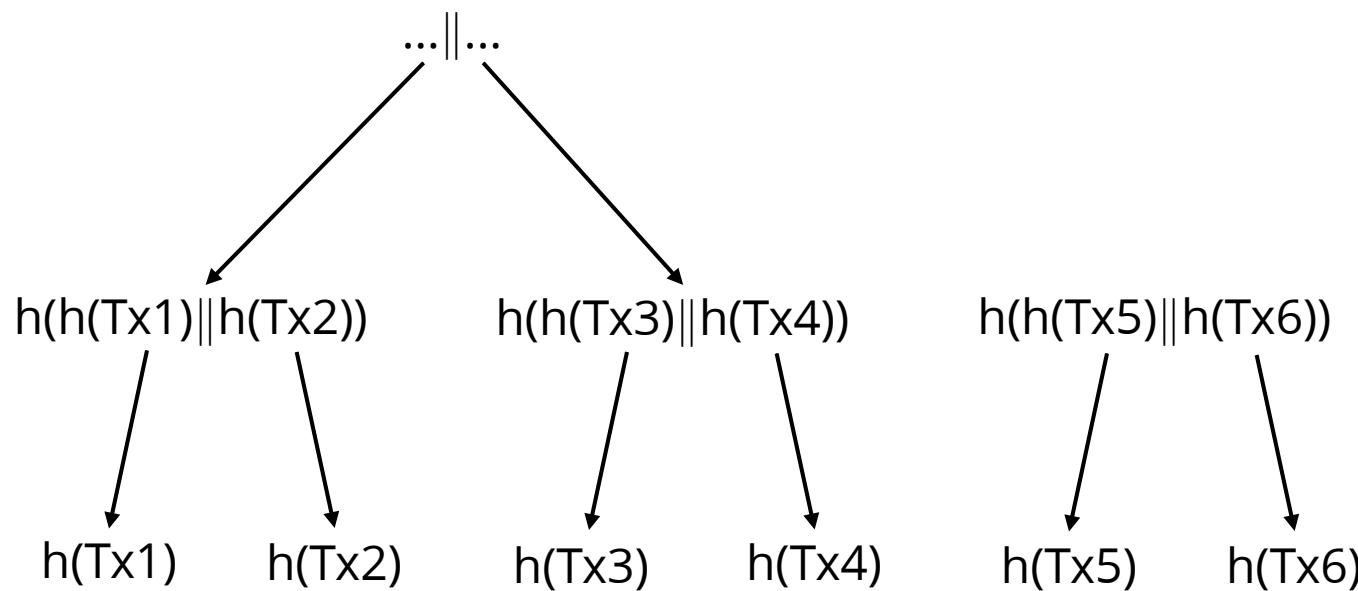
Arboles de Merkle



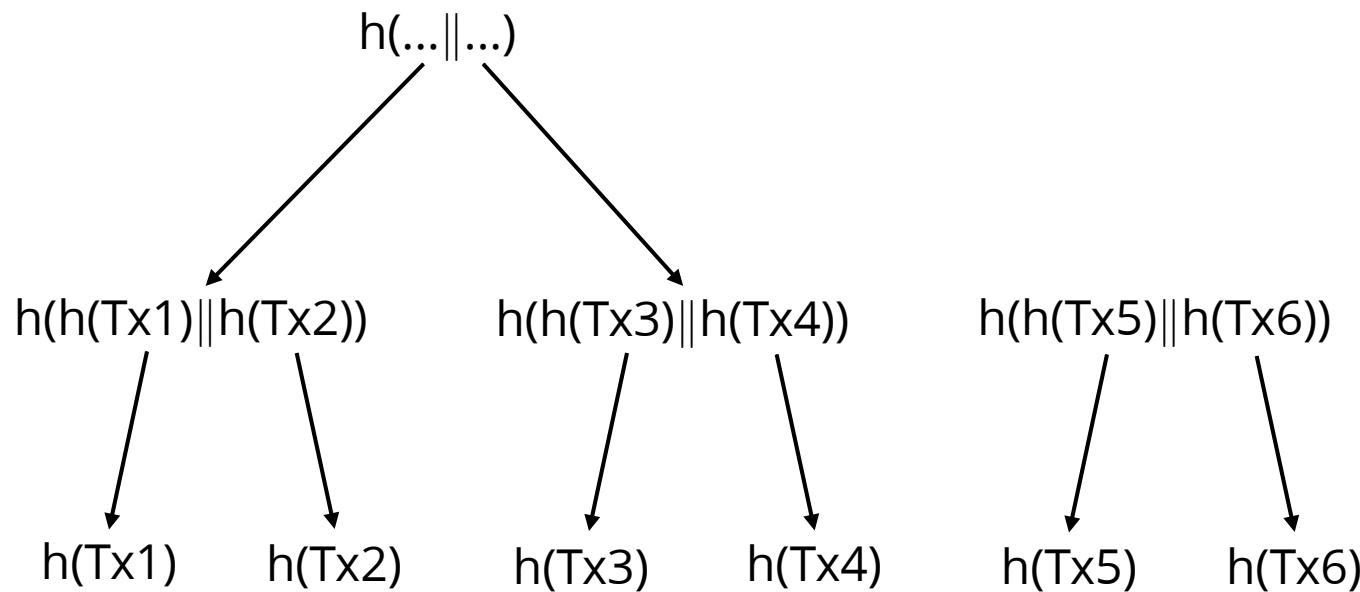
Arboles de Merkle



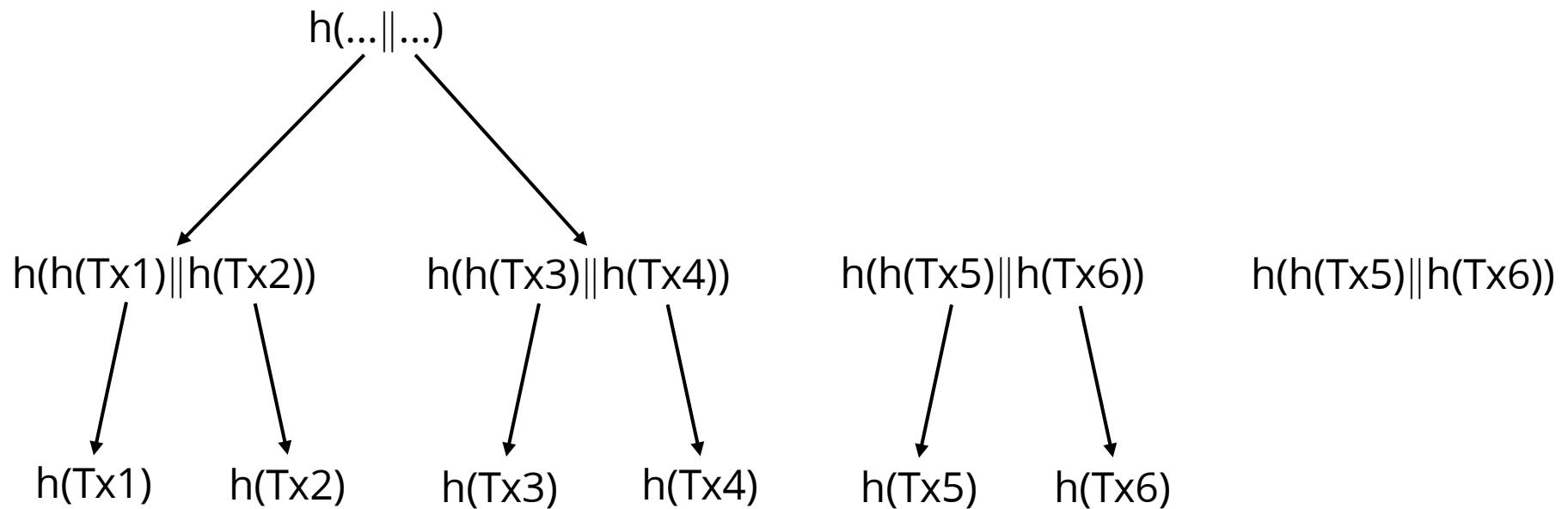
Arboles de Merkle



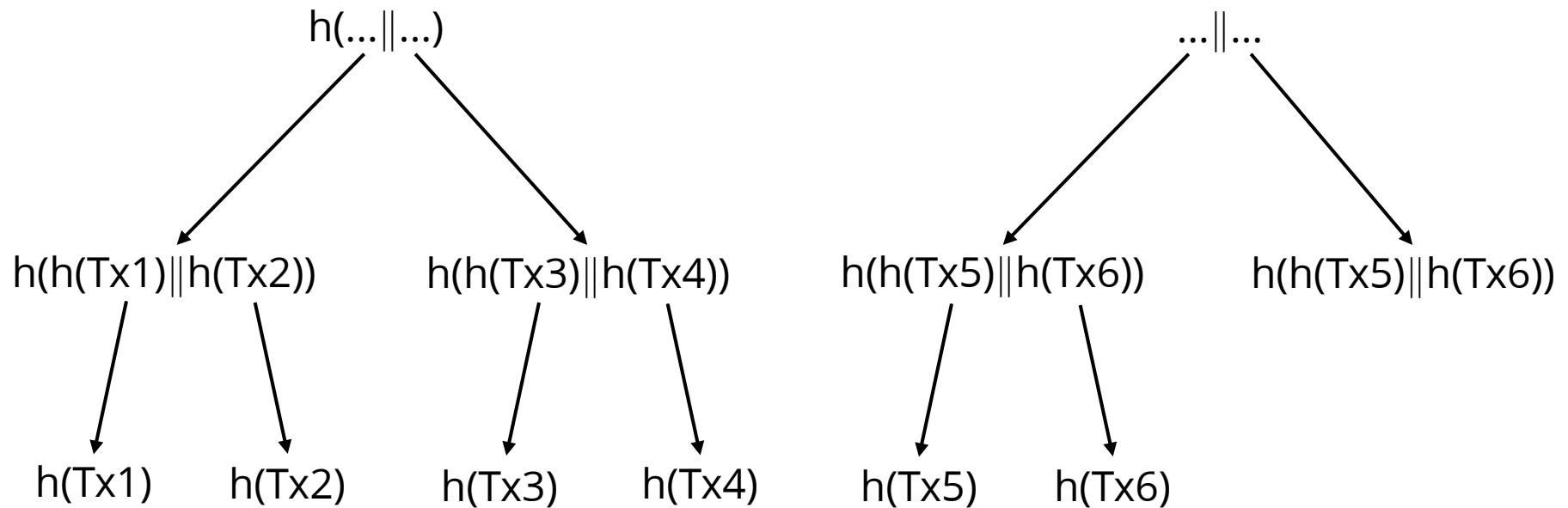
Arboles de Merkle



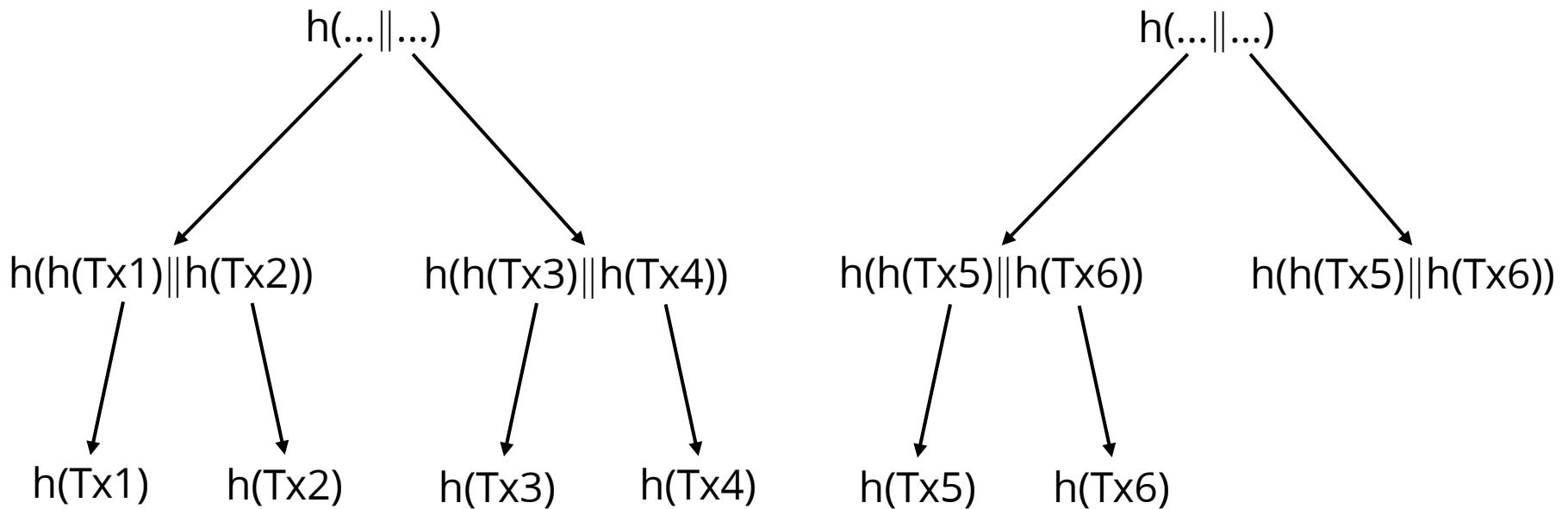
Arboles de Merkle



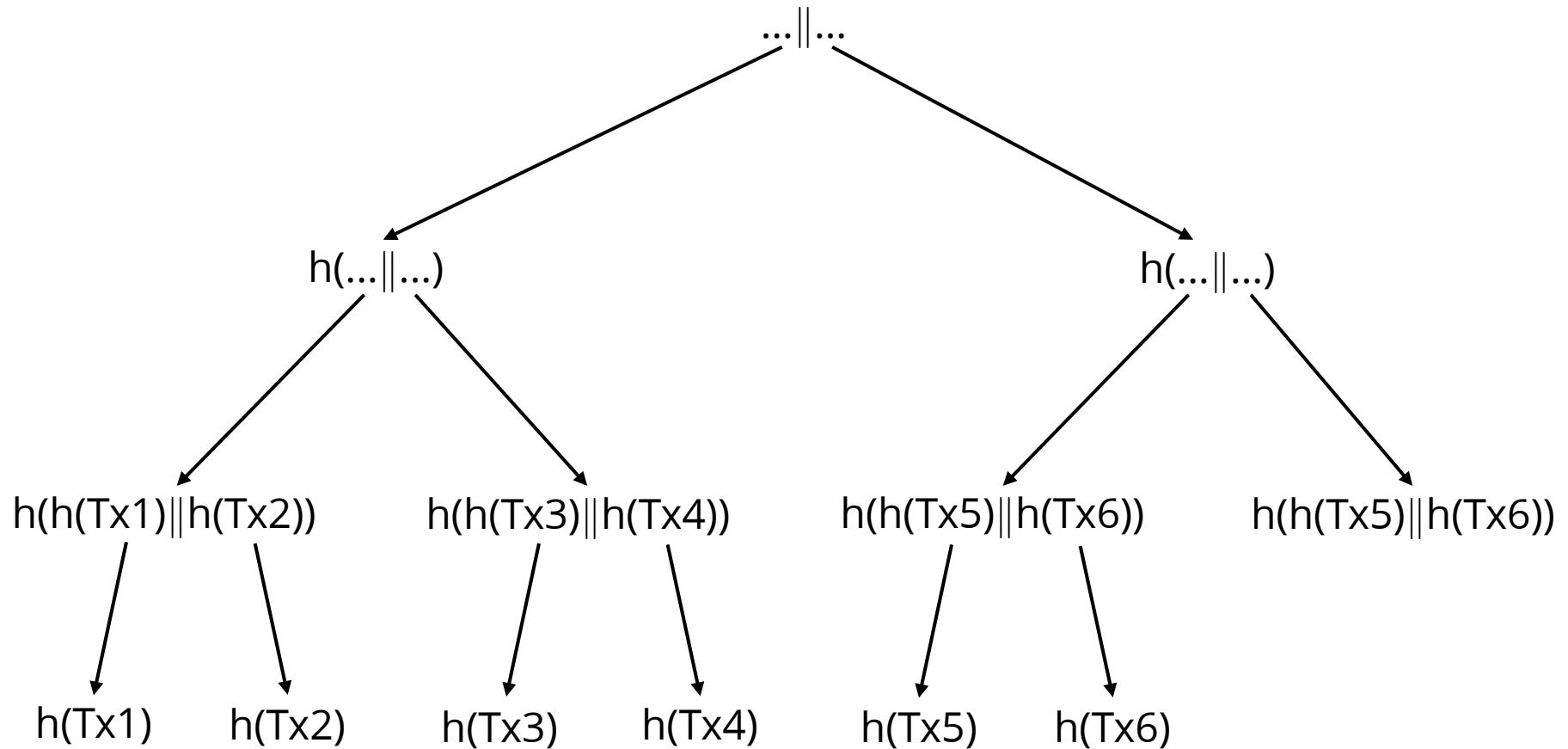
Arboles de Merkle



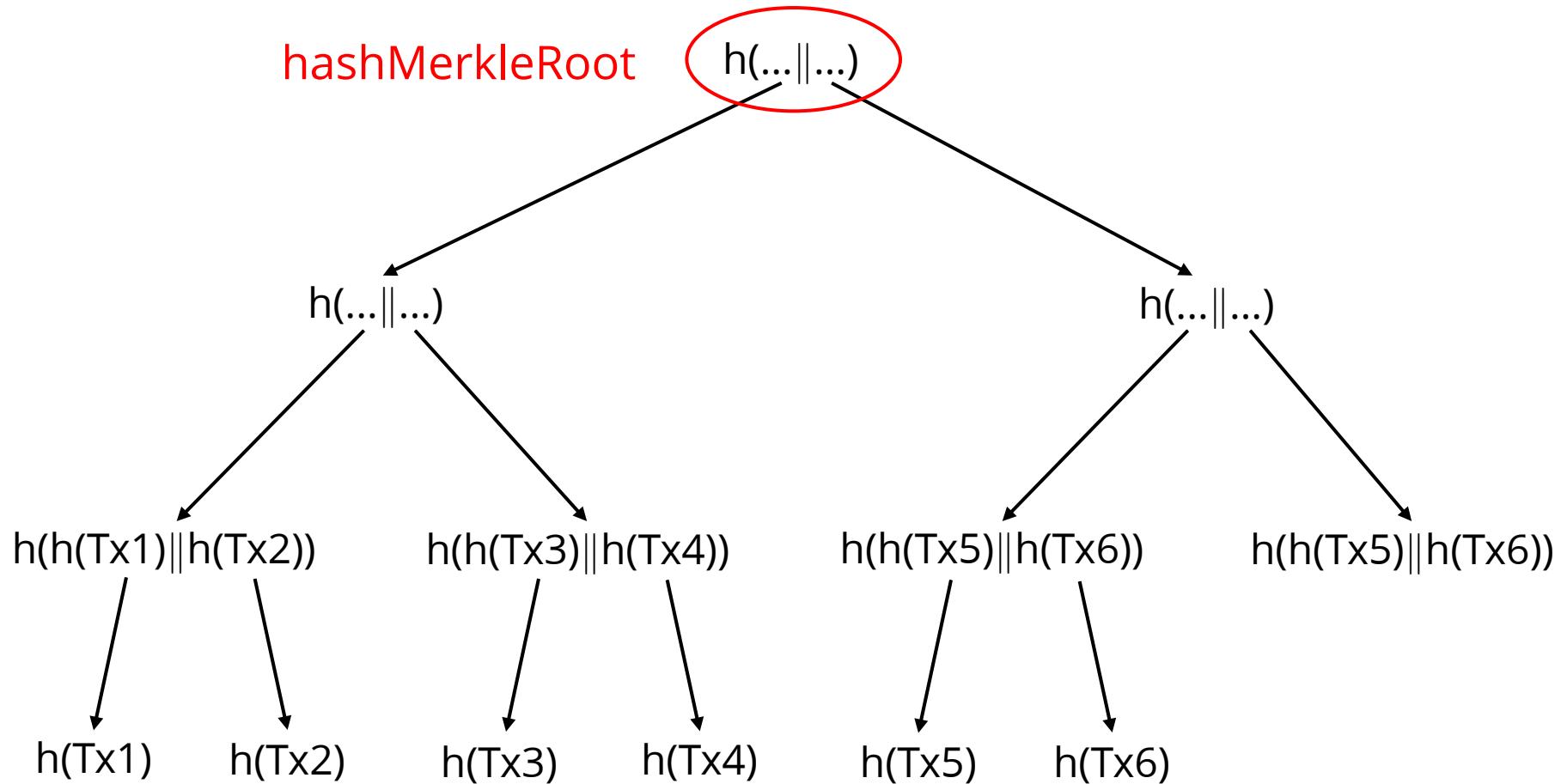
Arboles de Merkle



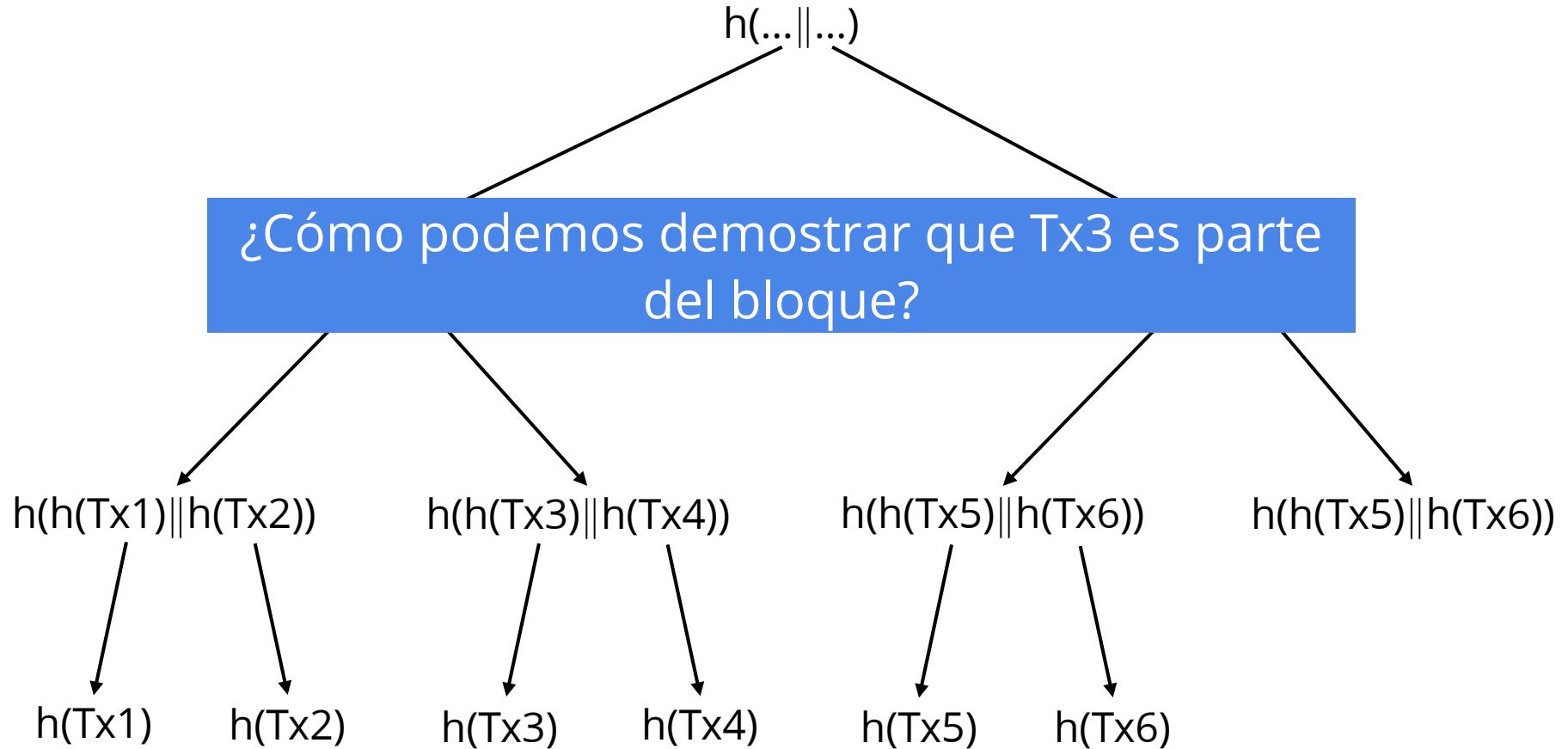
Arboles de Merkle



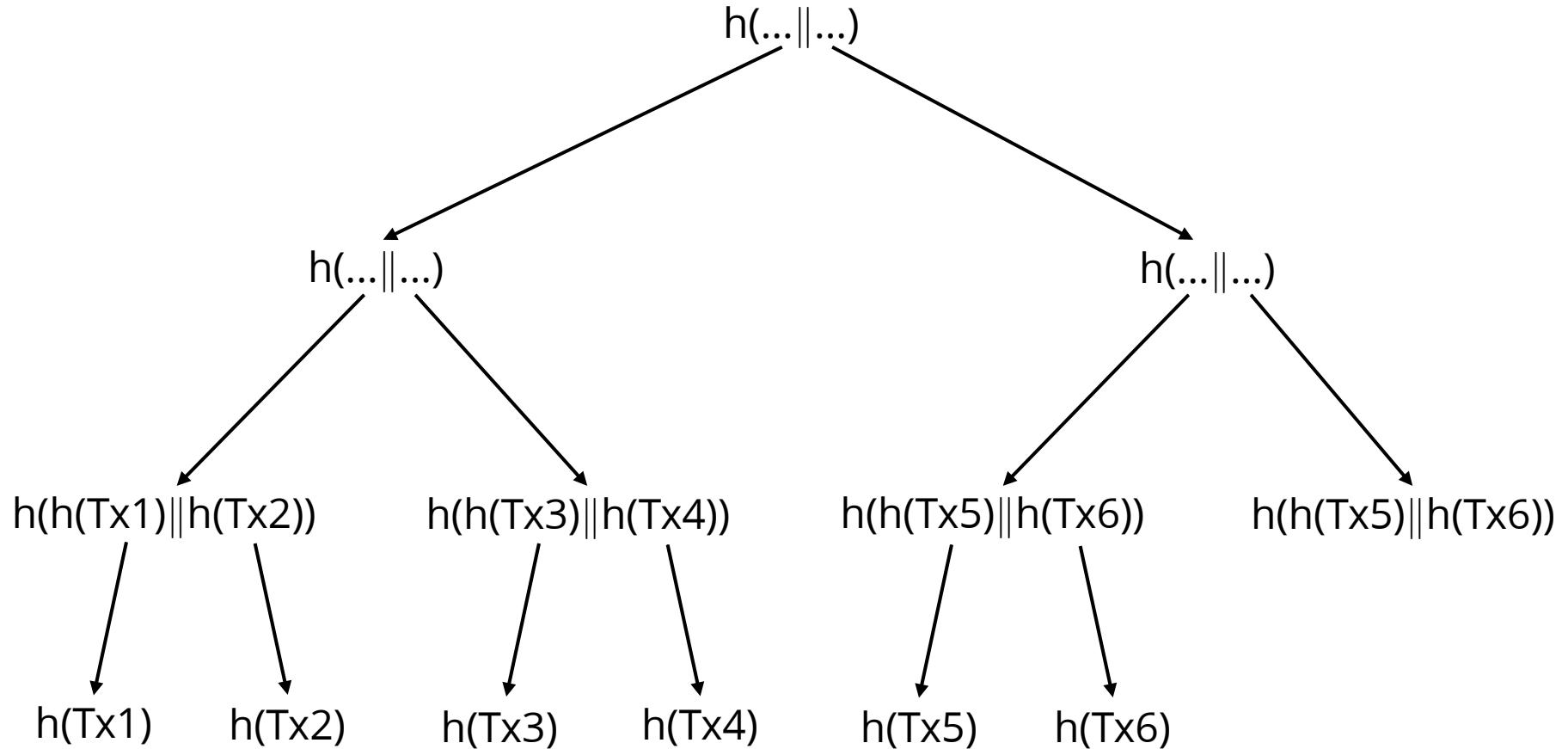
Arboles de Merkle



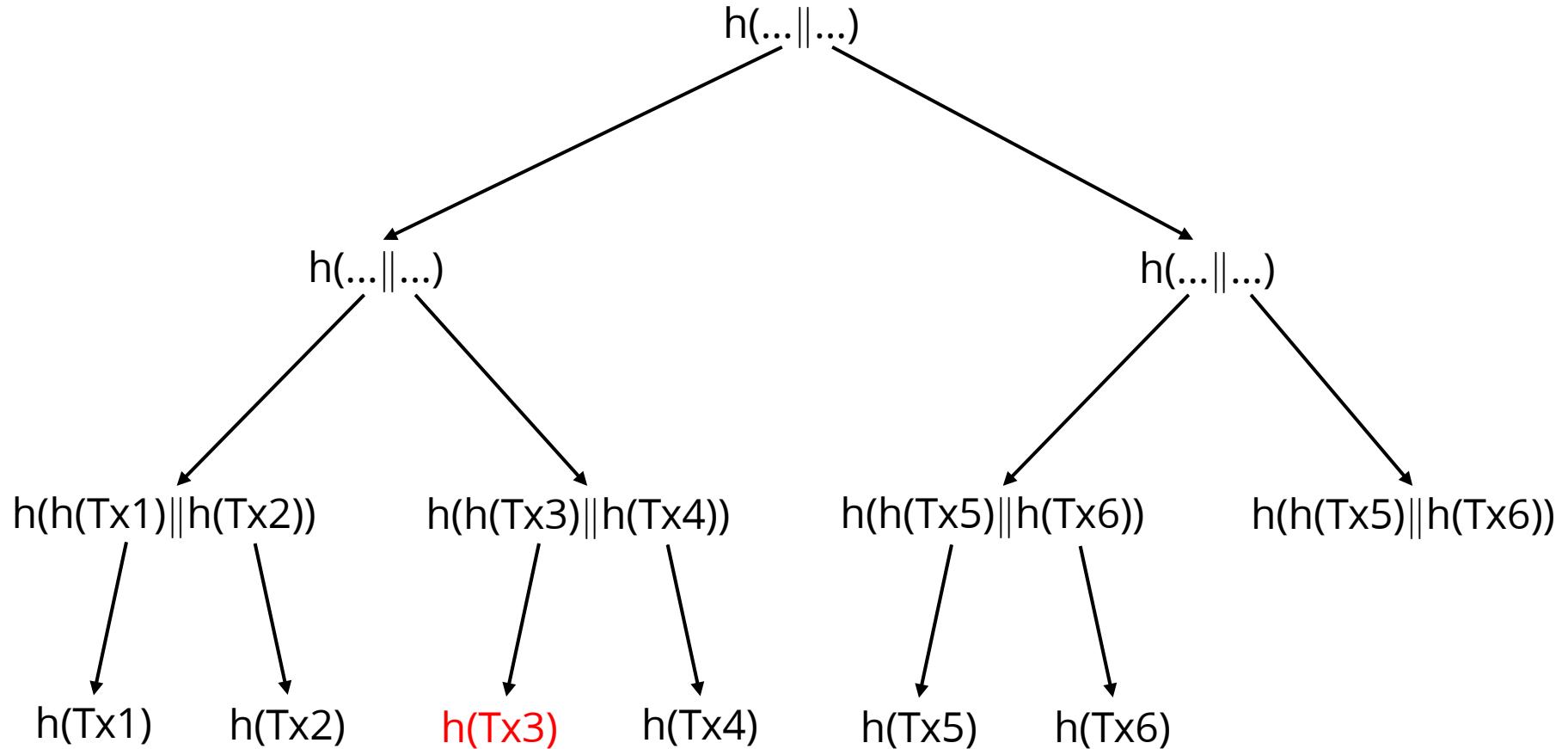
¿Qué ventajas tienen los árboles de Merkle?



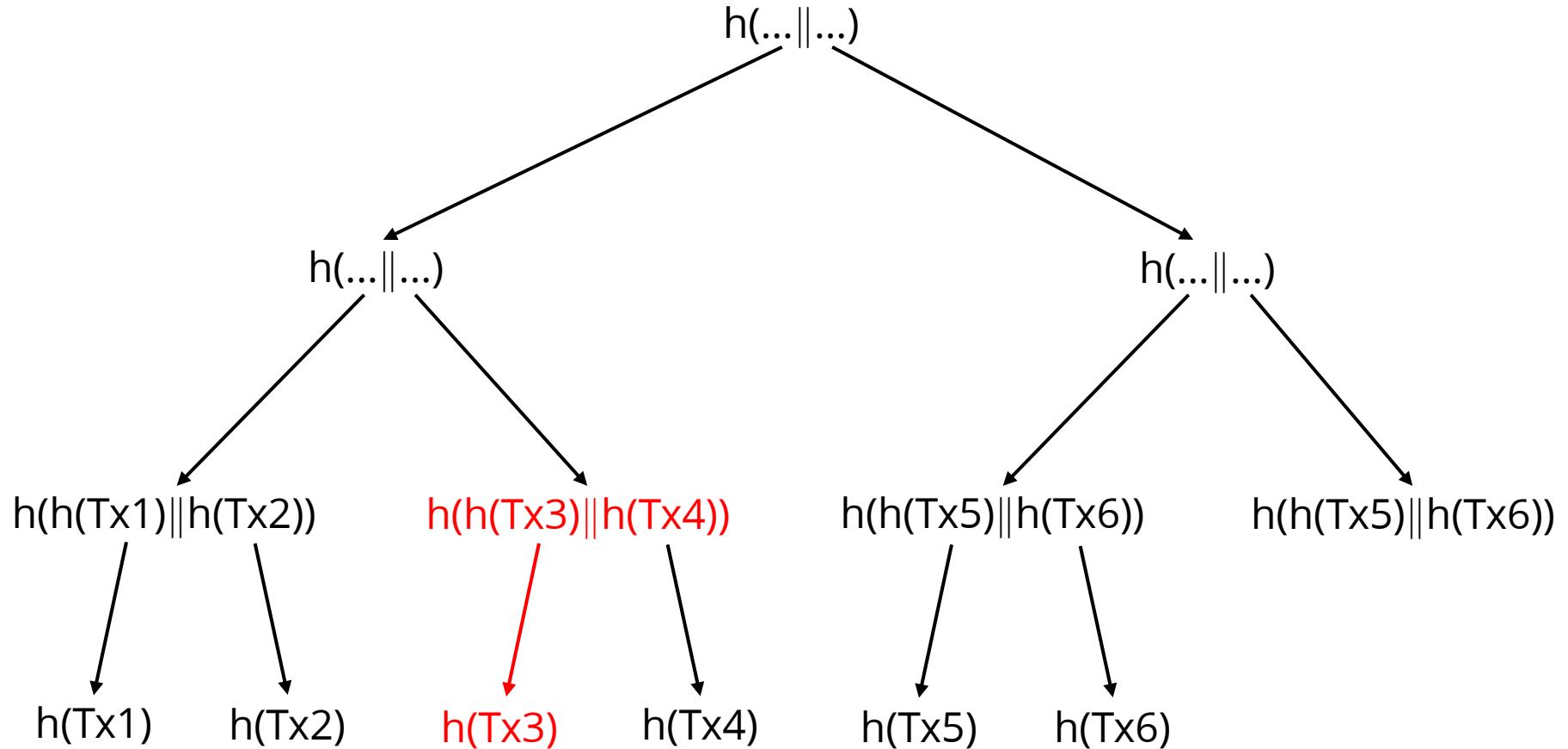
¿Qué ventajas tienen los árboles de Merkle?



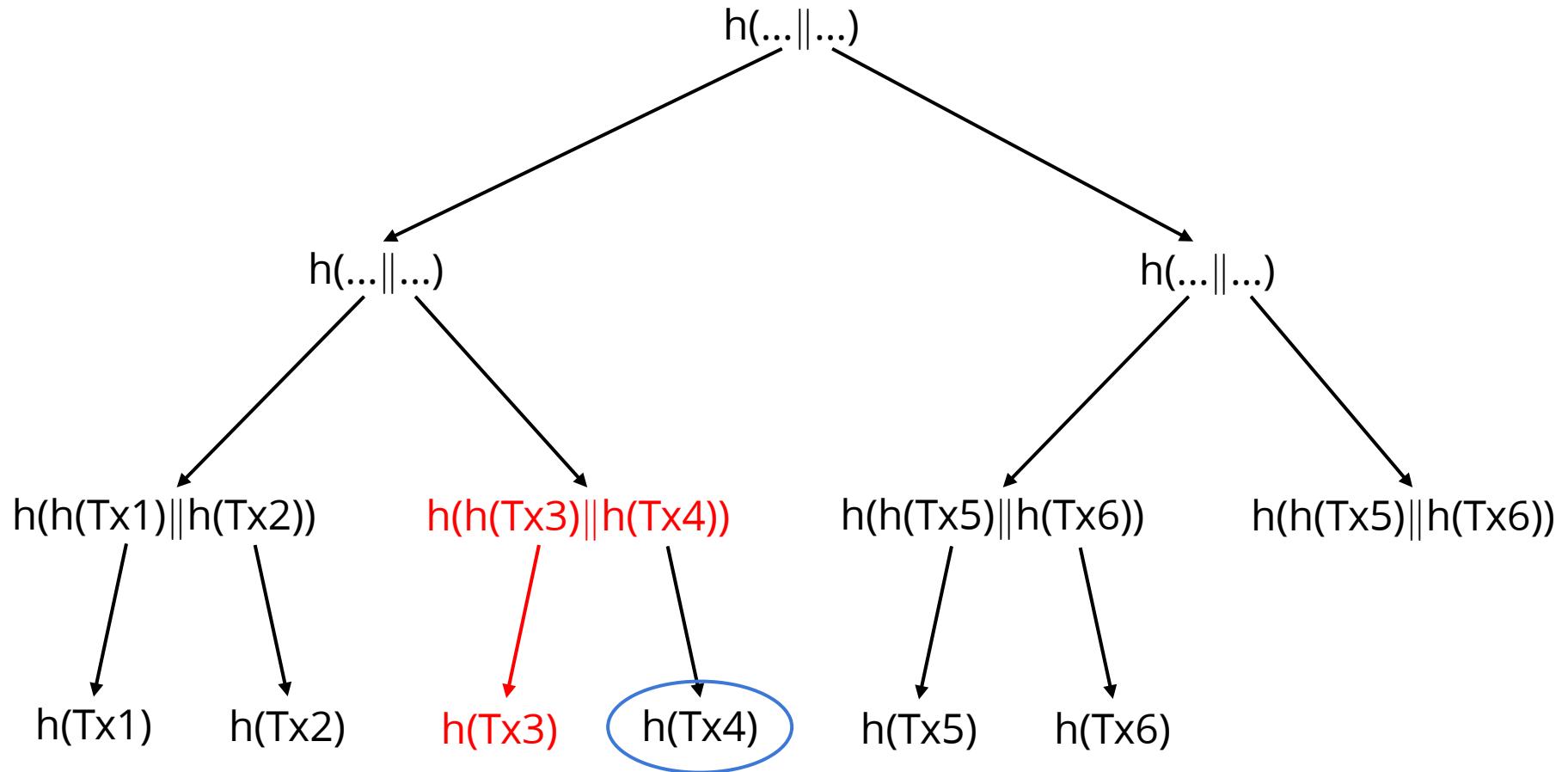
¿Qué ventajas tienen los árboles de Merkle?



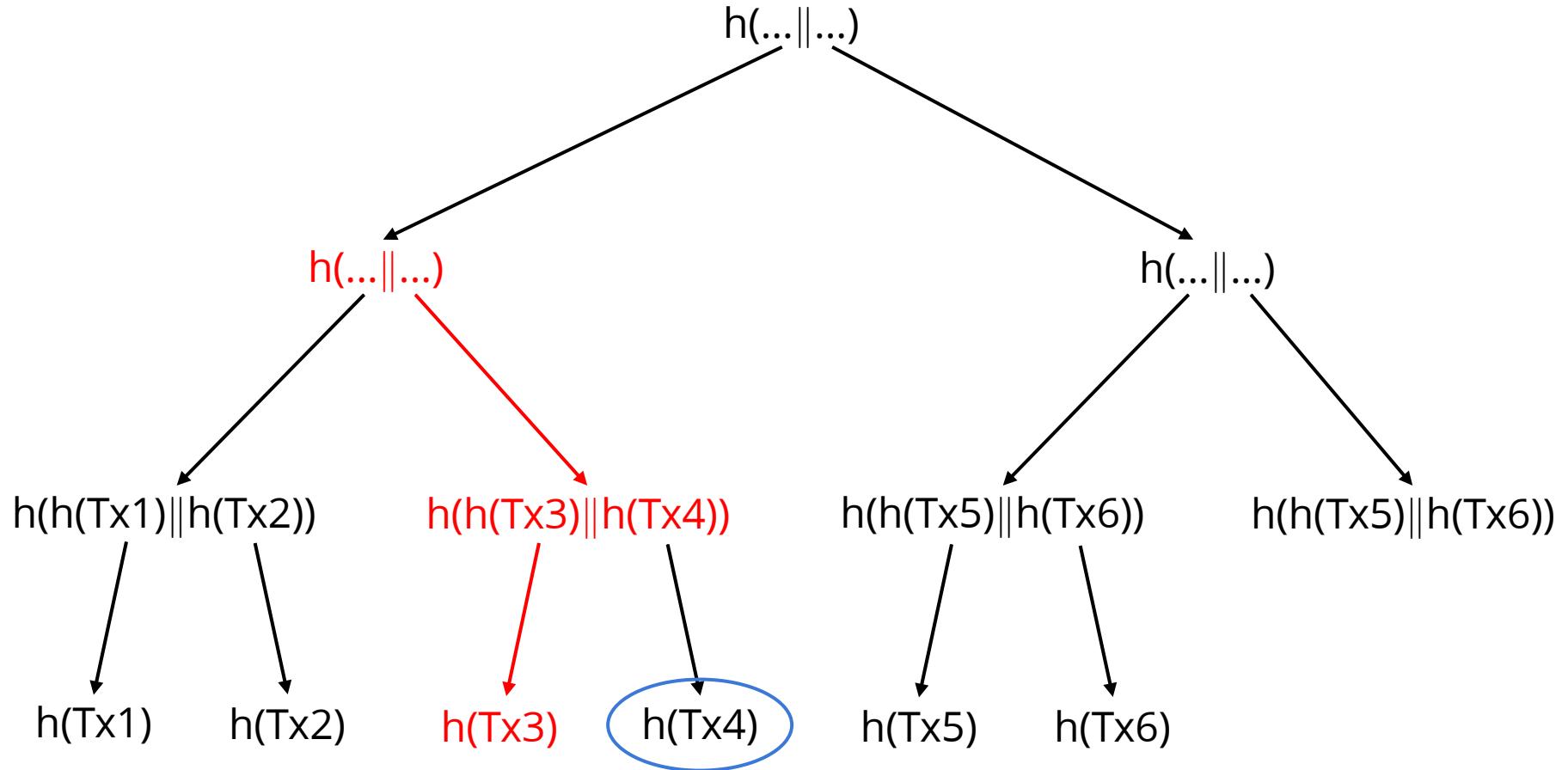
¿Qué ventajas tienen los árboles de Merkle?



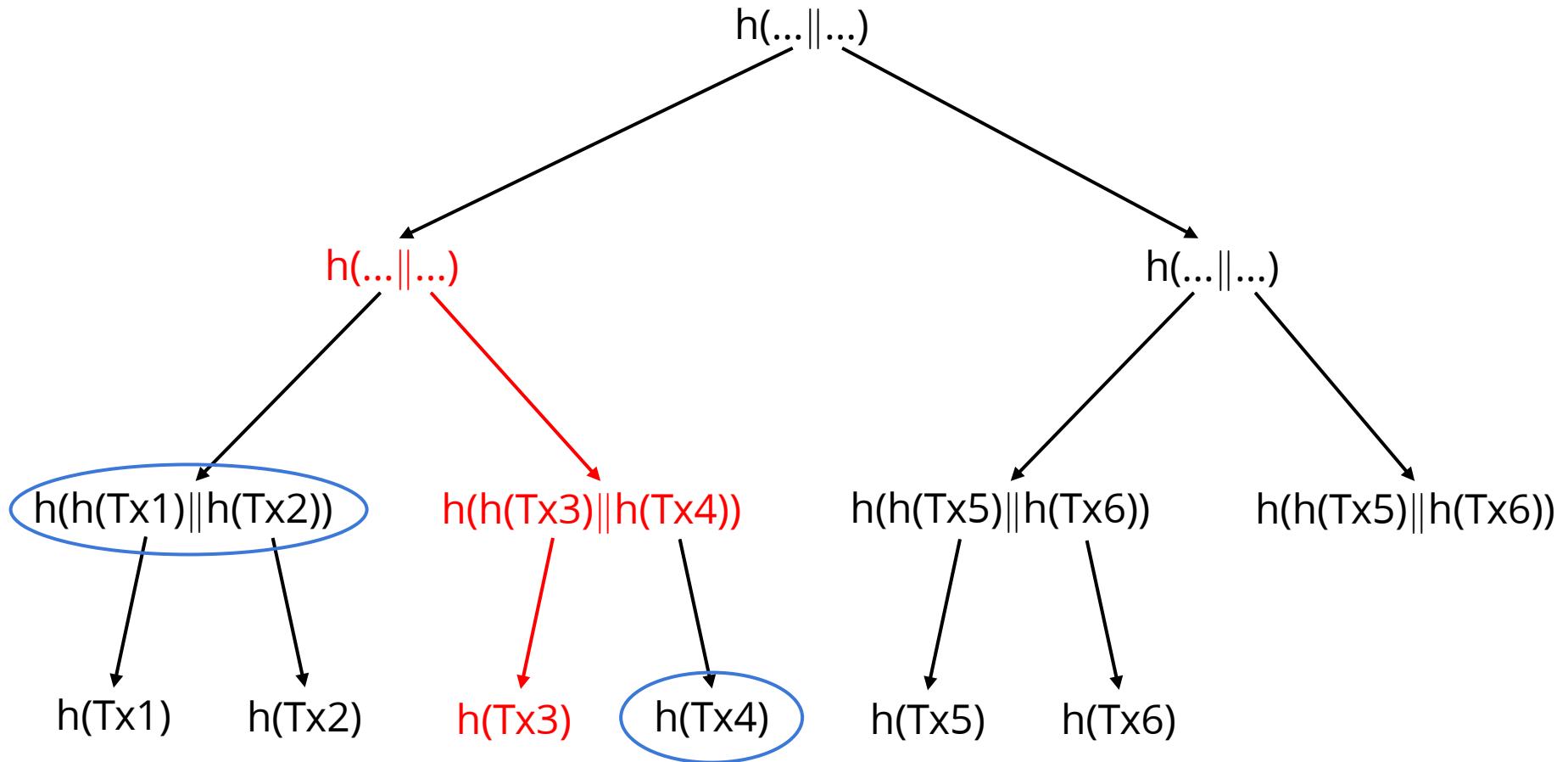
¿Qué ventajas tienen los árboles de Merkle?



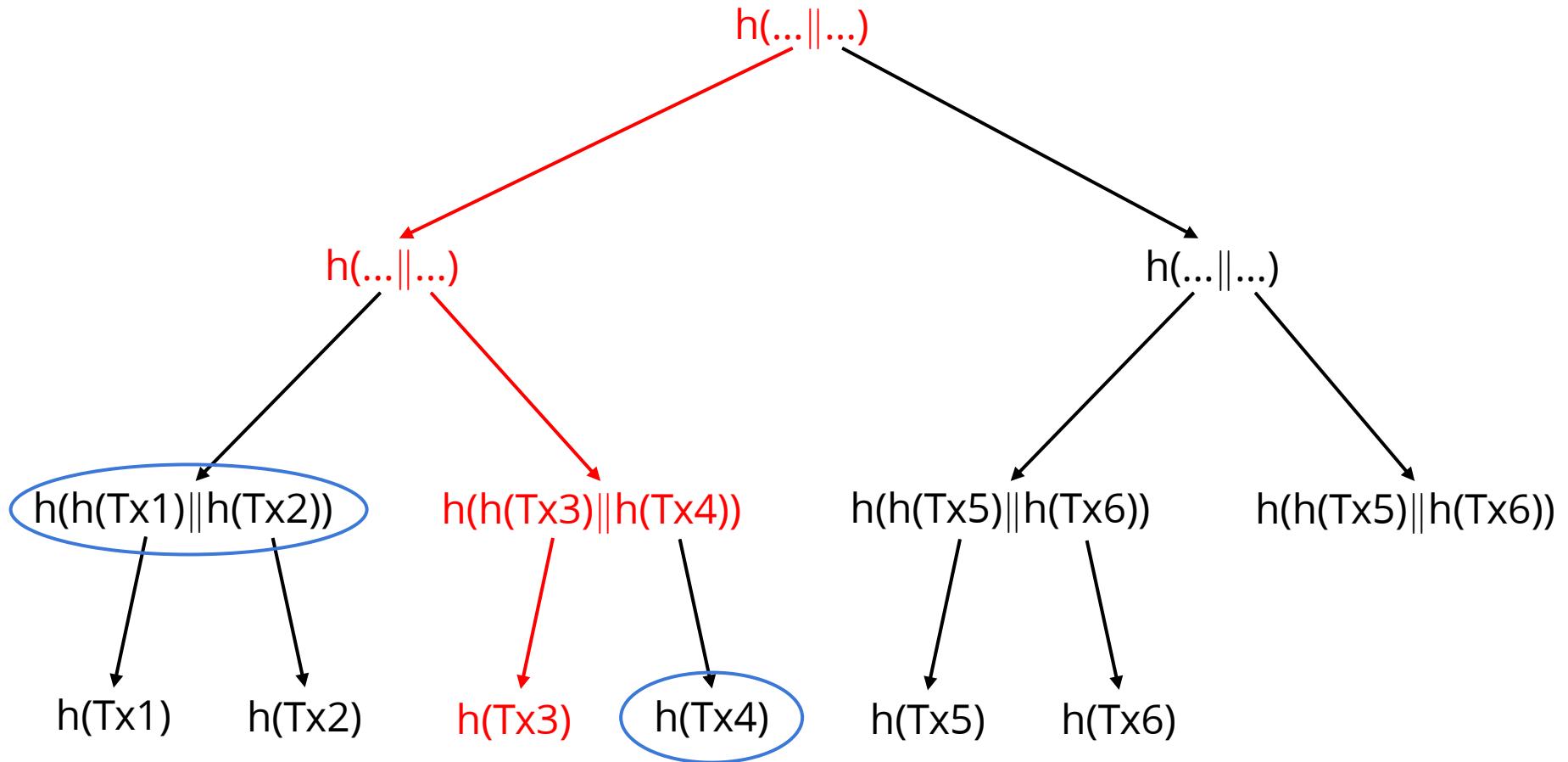
¿Qué ventajas tienen los árboles de Merkle?



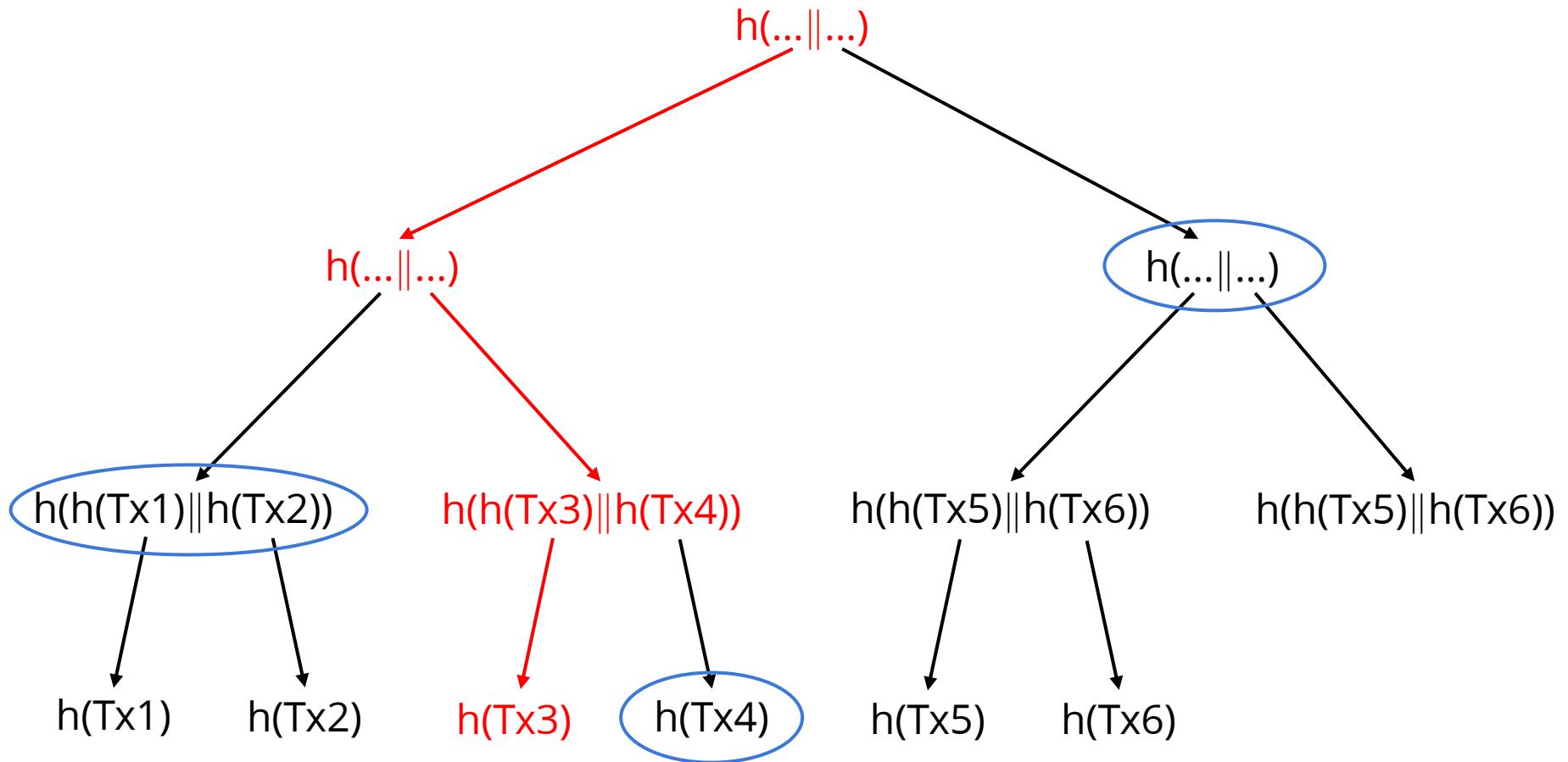
¿Qué ventajas tienen los árboles de Merkle?



¿Qué ventajas tienen los árboles de Merkle?



¿Qué ventajas tienen los árboles de Merkle?



Los que nos falta de header

version	indica la versión de las reglas de validación de un bloque que deben ser usadas
hashPrevBlock	hash del header del bloque anterior
hashMerkleRoot	raíz del árbol de Merkle para las transacciones del bloque
time	Unix timestamp que indica cuándo fue generado el bloque
bits	dificultad asociada a generar un bloque
nonce	número de 32 bits

Minería

Transacciones

Identificador de una transacción:

80975cddebaa93aa21a6477c0d050685d6820fa1068a
2731db0f39b535cbd369

Podemos ver la [codificación en hexadecimal](#) de esta transacción.

Y podemos ver la [traducción a JSON](#) de esta codificación.

JSON de una transacción

```
1  {
2      "version": 1,
3      "hash": "80975cddeb93aa21a6477c0d050685d6820fa1...",
4      "txid": "80975cddeb93aa21a6477c0d050685d6820fa1...",
5      "ins": [ ... ],
6      "outs": [ ... ]
7 }
```

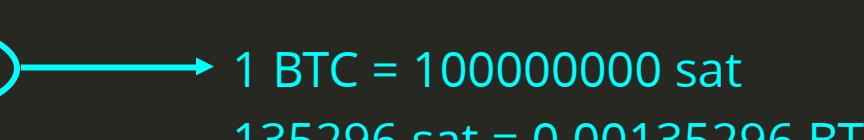
Entrada de una transacción

```
1 "ins": [
2   {
3     "n": 98,
4     "script": {
5       "asm": "OP_0 304402207e3e1158831eca394e472e43e..."
6     },
7     "txid": "08a1266ced5ef064741bd4bc51c1202456f2250..."
8   },
9   ...
10 ]
```

"txid":
"08a1266ced5ef064741bd4bc51c1202456f22509ae030231
860d6e9bef4acd5e"

Salida de una transacción

```
1 "outputs": [
2   ...
3   {
4     "n": 4,
5     "script": {
6       "addresses": [
7         "16BYtvVCunkZKvVyGMvD3BpRXnPUzTy4gF"
8       ],
9       "asm": "OP_DUP OP_HASH160 38d769cf2899983022b..."
10      },
11      "value": 135296
12    }
13 ]
```



1 BTC = 100000000 sat
135296 sat = 0.00135296 BTC

Cobrando una transacción

```
1 "txid": "80975cddebaa93aa21a6477c0d050685d6820fa1...",  
2 "outputs": [  
3     {  
4         "n": 4,  
5         "script": {  
6             "addresses": [  
7                 "16BYtvVCunkZKvVyGMvD3BpRXnPUzTy4gF"  
8             ],  
9             "asm": "OP_DUP OP_HASH160 38d769cf2899983022b..."  
10        },  
11        "value": 135296  
12    }
```

```
1 "txid": "a6e5da282a754881bd5abb6edf407d93c7d7ee7d1061c6...",  
2 "ins": [  
3     {  
4         "n": 4,  
5         "script": {  
6             "asm": "3045022100915cd28e731376443c6afff1e70ca88..."  
7         },  
8         "txid": "80975cddebaa93aa21a6477c0d050685d6820fa106...",  
9     }
```

Cobrando una transacción

```
1 "txid": "80975cddebba93aa21a6477c0d050685d6820fa1...",  
2 "outputs": [  
3   {  
4     "n": 4,  
5     "script": {  
6       "addresses": [  
7         "16BYtvVCunkZKvVyGMvD3BpRXnPUzTy4gF"  
8       ],  
9       "asm": "OP_DUP OP_HASH160 38d769cf2899983022b..."  
10     },  
11     "value": 135296  
12   }]
```

```
1 "txid": "a6e5da282a754881bd5abb6edf407d93c7d7ee7d1061c6...",  
2 "ins": [  
3   {  
4     "n": 4,  
5     "script": {  
6       "asm": "3045022100915cd28e731376443c6afff1e70ca88..."  
7     },  
8     "txid": "80975cddebba93aa21a6477c0d050685d6820fa106...",  
9   }]
```

Cobrando una transacción

```
1 "txid": "80975cddebbaa93aa21a6477c0d050685d6820fa1...",  
2 "outputs": [  
3     {  
4         "n": 4,  
5         "script": {  
6             "addresses": [  
7                 "16BYtvVCunkZKvVyGMvD3BpRXnPUzTy4gF"  
8             ],  
9             "asm": "OP_DUP OP_HASH160 38d769cf2899983022b..."  
10        },  
11        "value": 135296  
12    }
```

```
1 "txid": "a6e5da282a754881bd5abb6edf407d93c7d7ee7d1061c6...",  
2 "ins": [  
3     {  
4         "n": 4,  
5         "script": {  
6             "asm": "3045022100915cd28e731376443c6afff1e70ca88..."  
7         },  
8         "txid": "80975cddebbaa93aa21a6477c0d050685d6820fa106...",  
9     }
```

Cobrando una transacción

```
"asm":  
"OP_DUP OP_HASH160  
38d769cf2899983022b5611ab4d35bf7907dae20  
OP_EQUALVERIFY OP_CHECKSIG"
```

```
"asm":  
"3045022100915cd28e731376443c6afff1e70ca88d67bc01  
372aba91b3b2cb43581e5c5a53022021118f47188e96bdc82  
876f3344550c452bf2c3f19f49e078c887785dbfb2f6b01  
031165c872d4e0c43204d239ecf1d77eae0e691a9b4d5945f  
3d1304f0b7163f54c"
```

Cobrando una transacción

```
"asm":  
"OP_DUP OP_HASH160  
38d769cf2899983022b5611ab4d35bf7907dae20  
OP_EQUALVERIFY OP_CHECKSIG"
```

Cobrando una transacción

```
"asm":  
"OP_DUP OP_HASH160  
38d769cf2899983022b5611ab4d35bf7907dae20  
OP_EQUALVERIFY OP_CHECKSIG"
```



Programa en el lenguaje Script de Bitcoin

Cobrando una transacción

```
"asm":  
"3045022100915cd28e731376443c6afff1e70ca88d67bc01  
372aba91b3b2cb43581e5c5a53022021118f47188e96bdc82  
876f3344550c452bf2c3f19f49e078c887785dbfb2f6b01  
031165c872d4e0c43204d239ecf1d77eae0e691a9b4d5945f  
3d1304f0b7163f54c"
```

Cobrando una transacción

```
"asm":  
"3045022100915cd28e731376443c6afff1e70ca88d67bc01  
372aba91b3b2cb43581e5c5a53022021118f47188e96bdc82  
876f3344550c452bf2c3f19f49e078c887785dbfb2f6b01  
031165c872d4e0c43204d239ecf1d77eae0e691a9b4d5945f  
3d1304f0b7163f54c"
```



Firma digital

Cobrando una transacción

"asm":
"3045022100915cd28e731376443c6afff1e70ca88d67bc01
372aba91b3b2cb43581e5c5a53022021118f47188e96bdc82
876f3344550c452bf2c3f19f49e078c887785dbfb2f6b01
031165c872d4e0c43204d239ecf1d77eae0e691a9b4d5945f
3d1304f0b7163f54c"

Clave pública

Firma digital

El lenguaje Script

```
"asm":  
"OP_DUP OP_HASH160  
38d769cf2899983022b5611ab4d35bf7907dae20  
OP_EQUALVERIFY OP_CHECKSIG"  
  
"asm":  
"3045022100915cd28e731376443c6afff1e70ca88d67bc01  
372aba91b3b2cb43581e5c5a53022021118f47188e96bdc82  
876f3344550c452bf2c3f19f49e078c887785dbfb2f6b01  
031165c872d4e0c43204d239ecf1d77eae0e691a9b4d5945f  
3d1304f0b7163f54c"
```

El lenguaje Script

3045022100915cd28e731376443c6afff1e70ca88d67bc013
72aba91b3b2cb43581e5c5a53022021118f47188e96bdc828
76f3344550c452bf2c3f19f49e078c887785dbfb2f6b01
031165c872d4e0c43204d239ecf1d77eae0e691a9b4d5945f
3d1304f0b7163f54c

OP_DUP

OP_HASH160

38d769cf2899983022b5611ab4d35bf7907dae20

OP_EQUALVERIFY

OP_CHECKSIG

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

3045022100...

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

3045022100...

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

3045022100...

031165c872...

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

3045022100...

031165c872...

Ejecutando Script

3045022100...

031165c872...

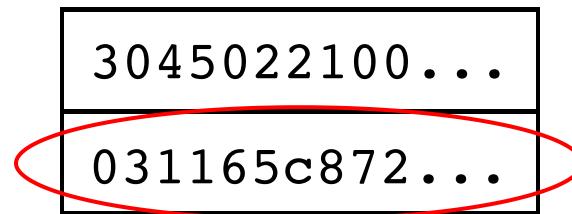
OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG



Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

3045022100...

031165c872...

031165c872...

Ejecutando Script

3045022100...

031165c872...

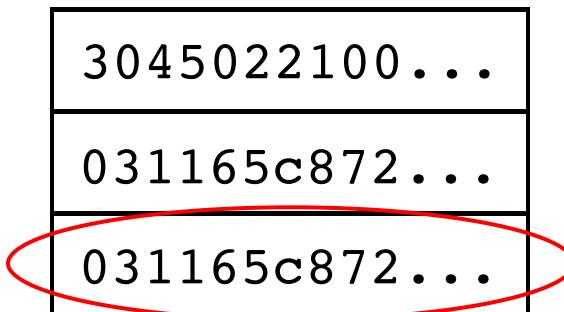
OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG



$$\text{HASH160}(x) = \text{RIPEMD-160}(\text{SHA-256}(x))$$

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

3045022100...

031165c872...

38d769cf28...

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

3045022100...

031165c872...

38d769cf28...

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

3045022100...

031165c872...

38d769cf28...

38d769cf28...

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

3045022100...
031165c872...
38d769cf28...
38d769cf28...

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

3045022100...

031165c872...

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

3045022100...
031165c872...

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

3045022100...
031165c872...

Verifica si "3045022100..." es una firma válida de la **transacción actual** dada la clave pública "031165c872..."

Ejecutando Script

Verifica si "3045022100..." es una firma válida de **h()**
dada la clave pública "031165c872..."

```
1 "txid": "a6e5da282a754881bd5abb6edf407d93c7d7ee7d1061c6...",  
2 "ins": [  
3   {  
4     "n": 4,  
5     "script": {  
6       "asm": "3045022100915cd28e731376443c6afff1e70ca88..."  
7     },  
8     "txid": "80975cddebaa93aa21a6477c0d050685d6820fa106...",  
9   }
```

Ejecutando Script

Verifica si "3045022100..." es una firma válida de **h()**
dada la clave pública "031165c872..."

Se firma la transacción actual completa, lo cual incluye todas las entradas y salidas

- Se firma el hash de un string que incluye las entradas sin las firmas (que van a ser calculadas) y las salidas completas

Ejecutando Script

Verifica si "3045022100..." es una firma válida de **h()**
dada la clave pública "031165c872..."

```
1 "txid": "a6e5da282a754881bd5abb6edf407d93c7d7ee7d1061c6...",  
2 "ins": [  
3   {  
4     "n": 4,  
5     "script": {  
6       "asm": "3045022100915cd28e731376443c6afff1e70ca88..."  
7     },  
8     "txid": "80975cddebaa93aa21a6477c0d050685d6820fa106...",  
9   }
```

Ejecutando Script

Verifica si "3045022100..." es una firma válida de **h()**
dada la clave pública "031165c872..."

```
1 "txid": "a6e5da282a754881bd5abb6edf407d93c7d7ee7d1061c6...",  
2 "ins": [  
3   {  
4     "n": 4,  
5  
6  
7  
8     "txid": "80975cddebaa93aa21a6477c0d050685d6820fa106...",  
9   }]
```

Ejecutando Script

Verifica si "3045022100..." es una firma válida de **h()**
dada la clave pública "031165c872..."

```
1 "txid": "adefda22a774488bd5abbed407d93c7d7ew7d1001ed...",  
2 "txe": {  
3     "n": 4,  
4     "s": "0000000000000000000000000000000000000000000000000000000000000000",  
5     "txid": "80975eddeba93ea21a6477cd0556d56d820fa106...",  
6     "v": 1  
7 }
```

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

3045022100...

031165c872...

Ejecutando Script

3045022100...

031165c872...

OP_DUP

OP_HASH160

38d769cf28...

OP_EQUALVERIFY

OP_CHECKSIG

1

El programa es válido si:

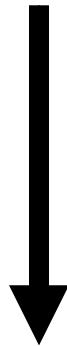
- Ninguna instrucción desencadena una falla
- El elemento superior del stack es distinto de cero al terminar la ejecución

¿Cómo se calcula una dirección de Bitcoin?

```
1 "outputs": [
2   ...
3   {
4     "n": 4,
5     "script": {
6       "addresses": [
7         "16BYtvVCunkZKvVyGMvD3BpRXnPUzTy4gF"
8       ],
9       "asm": "OP_DUP OP_HASH160 38d769cf2899983022b..."
10      },
11      "value": 135296
12    }
13  ]
```

¿Cómo se calcula una dirección de Bitcoin?

Clave pública del usuario: (x, y)



16BYtvVCunkZKvVyGMvD3BpRXnPUzTy4gF

Base 58

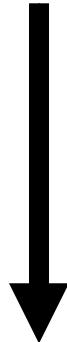
¿Cómo se calcula una dirección de Bitcoin?

031165c872d4e0c43204d239ecf1d77eae0e691a9b4d5945f3d1304
f0b7163f54c

¿Cómo se calcula una dirección de Bitcoin?

031165c872d4e0c43204d239ecf1d77eae0e691a9b4d5945f3d1304
f0b7163f54c

RIPEMD-160(SHA-256(...))



38d769cf2899983022b5611ab4d35bf7907dae20

¿Cómo se calcula una dirección de Bitcoin?

031165c872d4e0c43204d239ecf1d77eae0e691a9b4d5945f3d1304
f0b7163f54c

38d769cf2899983022b5611ab4d35bf7907dae20

¿Cómo se calcula una dirección de Bitcoin?

031165c872d4e0c43204d239ecf1d77eae0e691a9b4d5945f3d1304
f0b7163f54c

0038d769cf2899983022b5611ab4d35bf7907dae20

Network id

¿Cómo se calcula una dirección de Bitcoin?

031165c872d4e0c43204d239ecf1d77eae0e691a9b4d5945f3d1304
f0b7163f54c

0038d769cf2899983022b5611ab4d35bf7907dae20**eeccbce10**

¿Cómo se calcula una dirección de Bitcoin?

031165c872d4e0c43204d239ecf1d77eae0e691a9b4d5945f3d1304
f0b7163f54c

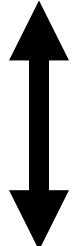
0038d769cf2899983022b5611ab4d35bf7907dae20**eeccbce10**

Checksum: SHA-256(SHA-256(0038d769cf2899983022b5611ab4d35bf7
907dae20)) = eecbce10...

¿Cómo se calcula una dirección de Bitcoin?

031165c872d4e0c43204d239ecf1d77eae0e691a9b4d5945f3d1304
f0b7163f54c

0038d769cf2899983022b5611ab4d35bf7907dae20eecbce10



16BYtvVCunkZKvVyGMvD3BpRXnPUzTy4gF

El pago a los mineros

Aún nos falta indicar cómo se ven las transacciones donde se crean bitcoin y se paga a los mineros

Estas son llamadas **coinbase transactions**

Coinbase transaction

Coinbase transaction

```
1 "outs": [
2   {
3     "n": 0,
4     "script": {
5       "addresses": [
6         "1CK6KHY6MHgYvmRQ4PAafKYDrglejbH1cE"
7       ],
8       "asm": "OP_DUP OP_HASH160 7c154ed1dc59609e3d2...",
9     },
10    "value": 1264124389
11  }
12 ]
```



12.5 BTC + pago por transacciones

La diferencia entre las entradas y las salidas de una transacción puede ser mayor que 0. Esta diferencia es considerada como pago para el minero

¿Cómo se calcula el pago de un minero?

Inicialmente el pago por colocar un bloque era de 50 BTC

Este pago se divide por 2 cada 210000 bloques, lo que se espera que ocurra cada cuatro años

- Esta división ha ocurrido 3 veces ya que estamos cerca del bloque 800000
- El pago actual de un minero es de 6.25 BTC

¿Cómo se calcula el pago de un minero?

La cantidad de Bitcoins es entonces:

$$\sum_{i=0}^{\infty} 210000 \cdot 50 \cdot \left(\frac{1}{2}\right)^i =$$

$$10500000 \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = 10500000 \cdot 2 = 21000000$$

Minería

Los ingredientes fundamentales

bits	dificultad asociada a generar un bloque
nonce	número de 32 bits

¿Cómo se representa la dificultad?

Bloque	Dificultad
530139	5077499034879.02
700000	18415156832118.24

$$\text{dificultad} = \frac{\text{valor objetivo máximo}}{\text{valor objetivo actual}}$$

¿Cómo se representa la dificultad?

Bloque	Dificultad
530139	5077499034879.02
700000	18415156832118.24

$$\text{valor objetivo actual} = \frac{\text{valor objetivo máximo}}{\text{dificultad}}$$

¿Cómo se representa la dificultad?

Bloque	Dificultad
530139	5077499034879.02
700000	18415156832118.24

Bloque 530139

valor objetivo actual =

ffff0000000000000000000000000000
000000000000000000000000
5077499034879.02

= 376f55fffffff6cc7dd90b5e2b9d74ed71b3
7ec4bcab13



hash del bloque = 1e6db880e65dd2582d42040d7d28d8b7b9dc
30da42134a

La dificultad se actualiza cada 2016 bloques

Se espera un bloque sea generado cada 10 minutos

- En 2 semanas se deberían generar 2016 bloques

Si en el bloque t se produjo un cambio de dificultad:

$$\text{dificultad}_{t+2016} = \text{dificultad}_{t+2015} \cdot \frac{1209600}{\text{time}_{t+2015} - \text{time}_t}$$

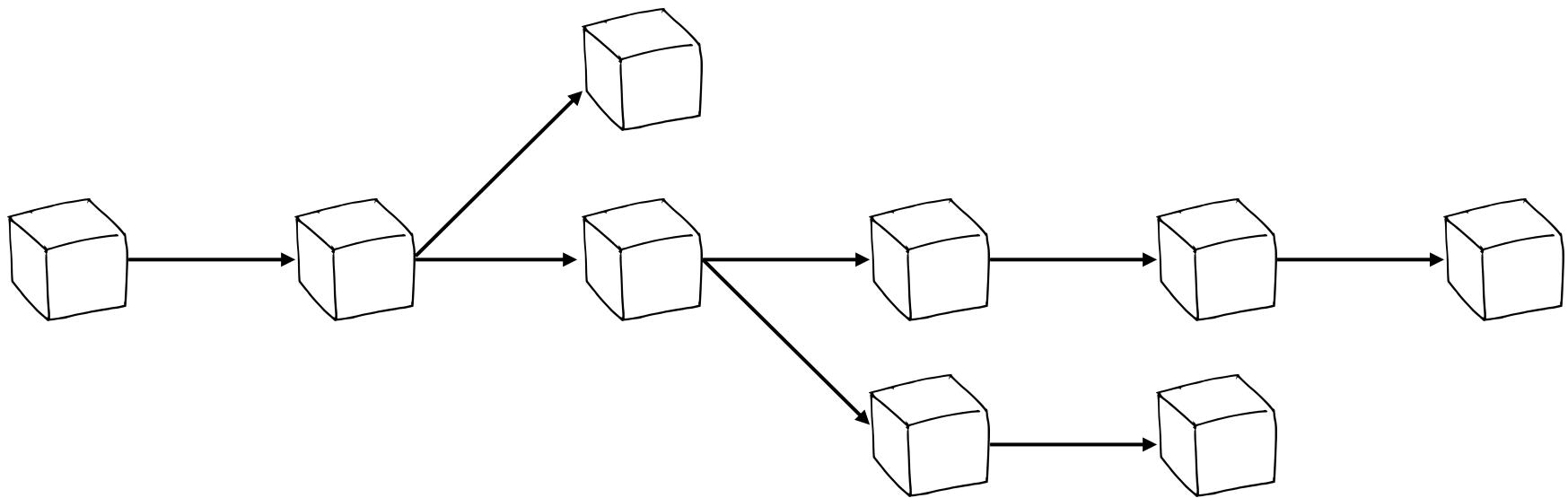
El protocolo

Vimos las reglas fundamentales que definen el protocolo de Bitcoin

- Hay muchos sitios donde puede obtener más información sobre este protocolo, por ejemplo [aquí](#)
- El código de Bitcoin es abierto y puede ser encontrado en este [repositorio](#)

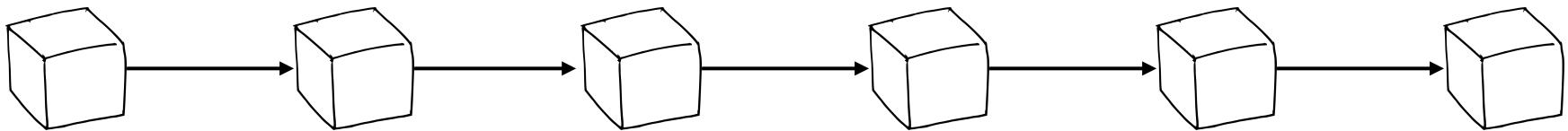
Tenemos que responder una última pregunta: ¿cómo se define el blockchain de Bitcoin?

El blockchain de Bitcoin



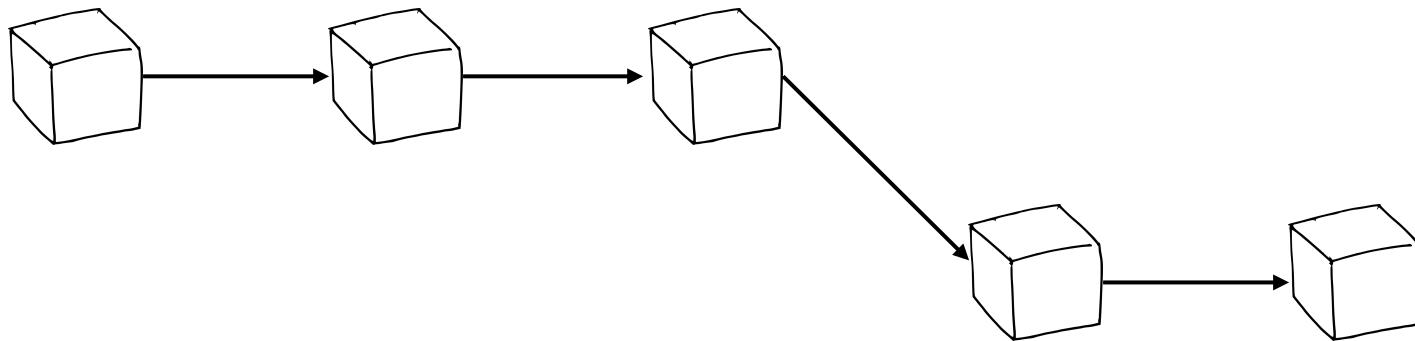
El blockchain es el **camino con el mayor trabajo acumulado**, lo cual es definido por la suma de las dificultades de los bloques del camino

El blockchain de Bitcoin



El blockchain es el **camino con el mayor trabajo acumulado**, lo cual es definido por la suma de las dificultades de los bloques del camino

El blockchain de Bitcoin



El blockchain es el **camino con el mayor trabajo acumulado**, lo cual es definido por la suma de las dificultades de los bloques del camino