Funciones de hash criptograficas

¿Cómo funciona Bitcoin?

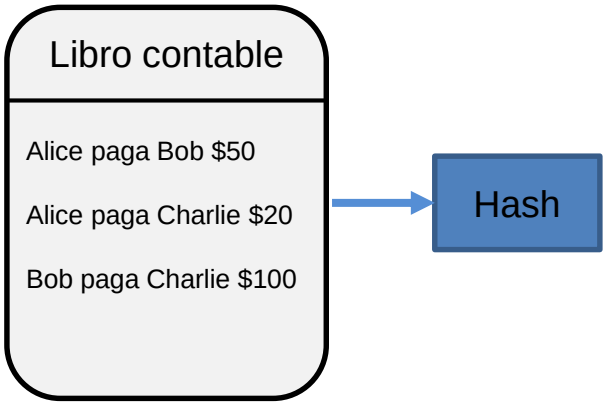
Que es una función de hash?

- Es una función
- Con ciertas propiedades (bucketing)

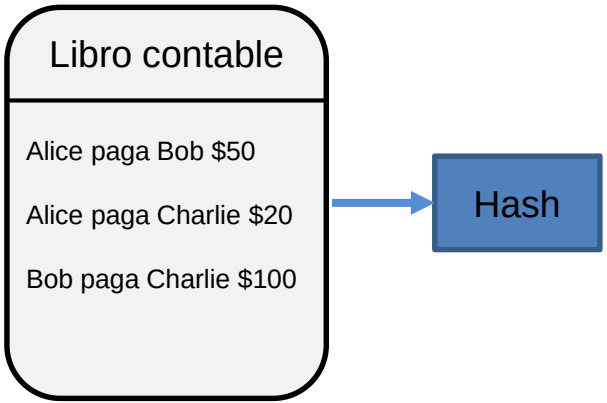
Que es una función de hash?

Función/algoritmo

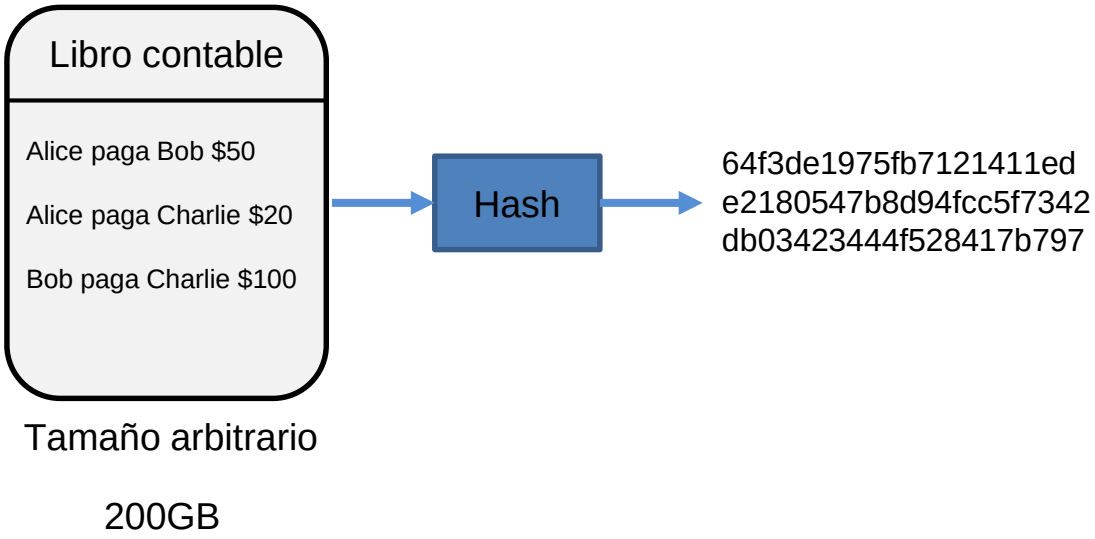
Que es una función de hash?



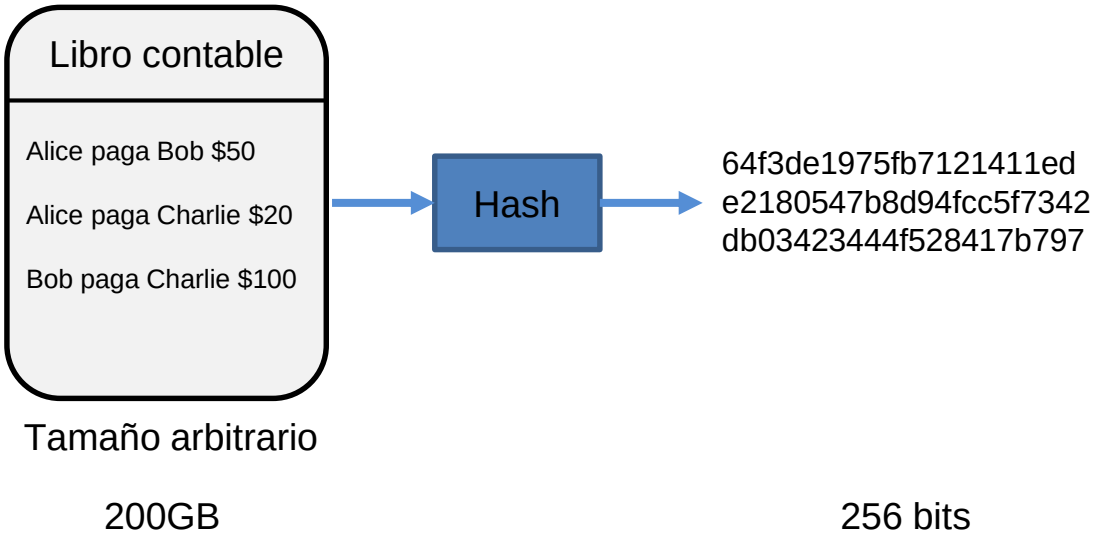
Que es una función de hash?



Que es una función de hash?

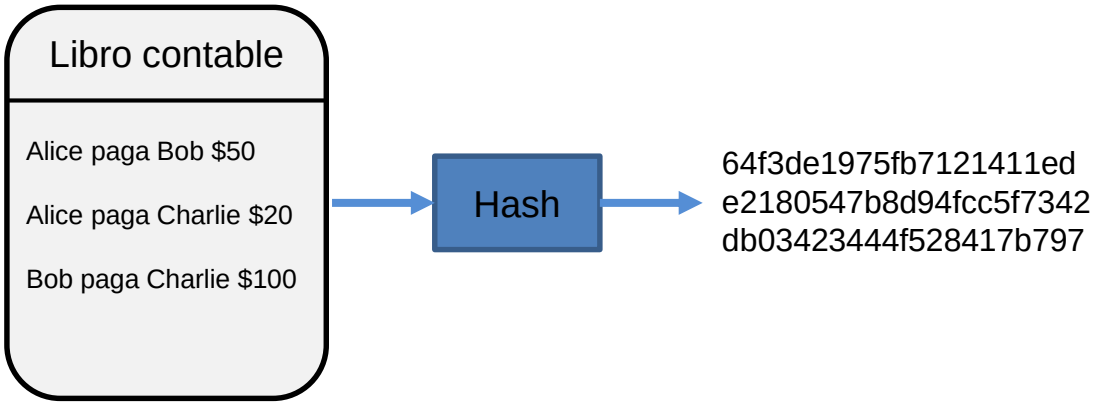


Que es una función de hash?



Que es una función de hash?

Computable eficientemente O(n)



Tamaño arbitrario

200GB

256 bits

Que es una función de hash?

Función con estas propiedades:

- Toma input de tamaño arbitrario
- Produce output de tamaño fijo
- Computa el output en O(n)

Que es función de hash criptográfica?

Función de hash con ciertas propiedades de seguridad:

- **Resistencia a colisiones**
- Hiding (función que esconde el input)
- Puzzle friendliness (para hacer el mineo)

Propiedad 1

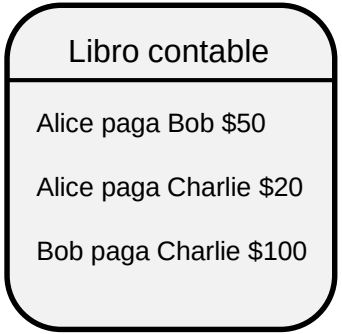
Resistencia a colisiones

Función de hash H es **resistente a colisiones** si:

- No es *factible encontrar* dos valores *x,y*
- $x \neq y$
- $H(x) = H(y)$

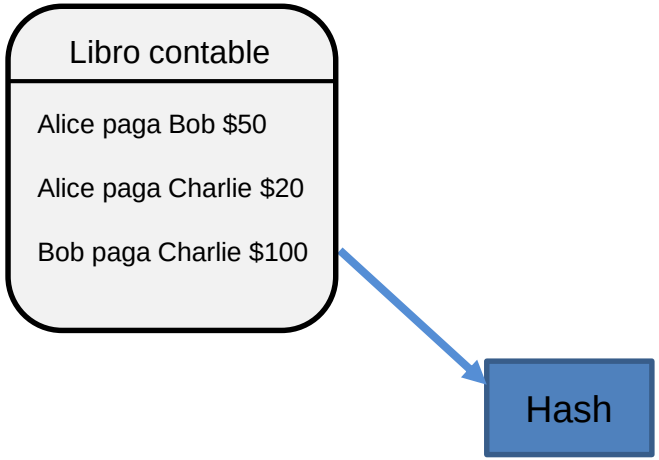
Propiedad 1

Resistencia a colisiones



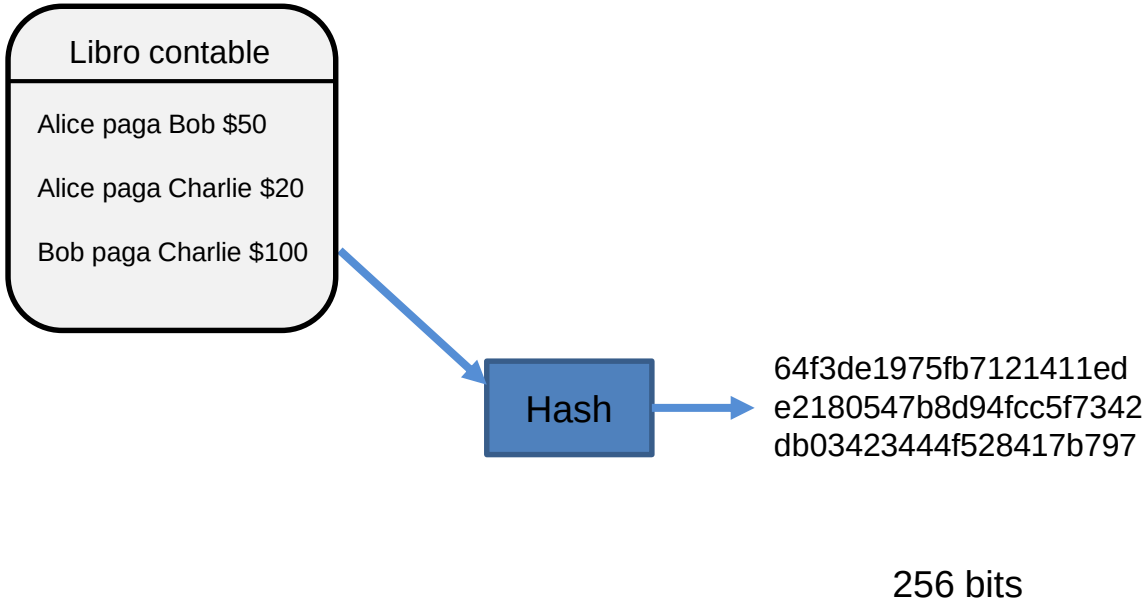
Propiedad 1

Resistencia a colisiones



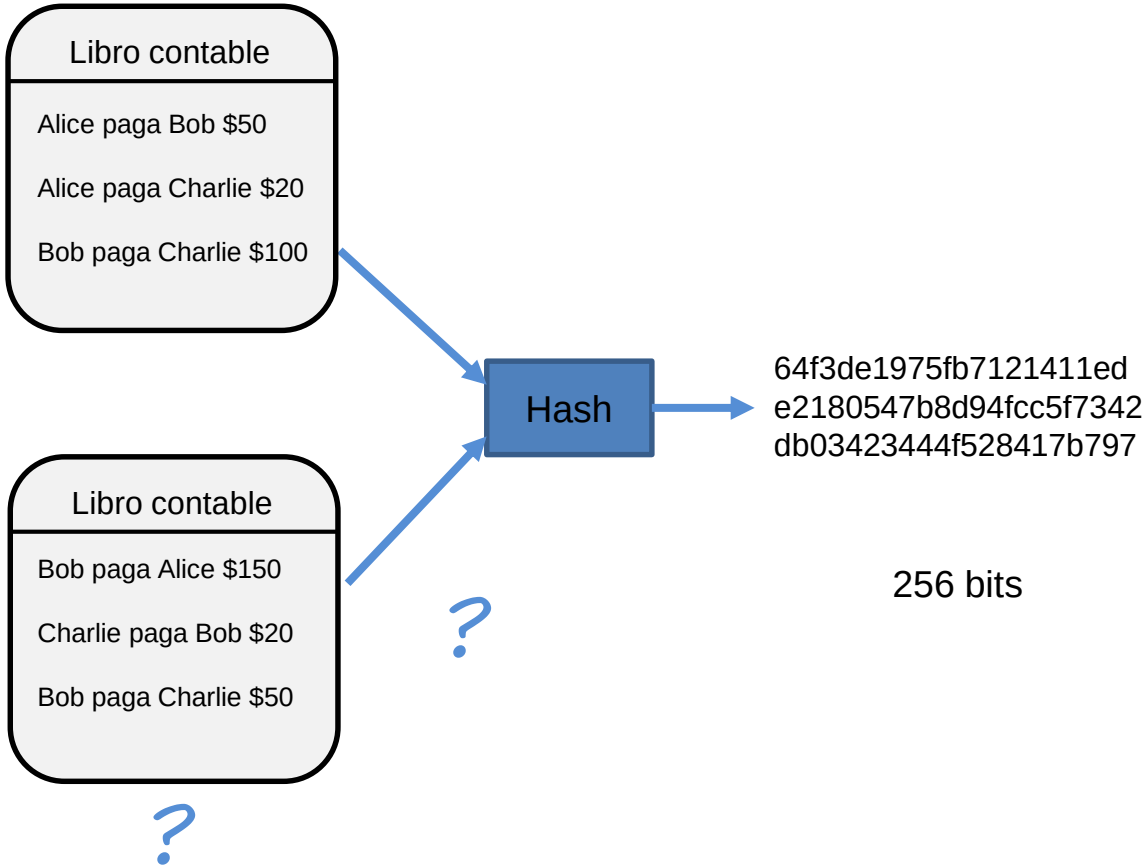
Propiedad 1

Resistencia a colisiones



Propiedad 1

Resistencia a colisiones



Propiedad 1

Resistencia a colisiones

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

Libro contable

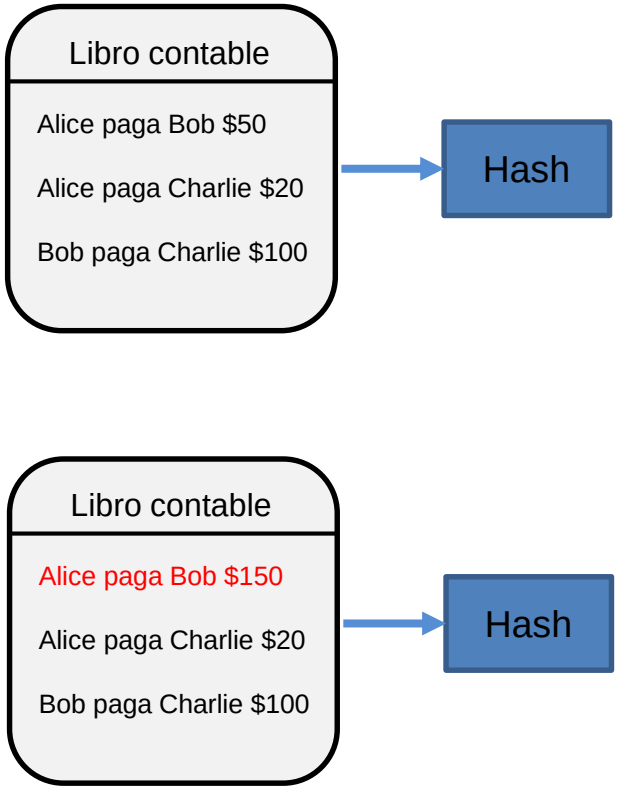
Alice paga Bob $150

Alice paga Charlie $20

Bob paga Charlie $100

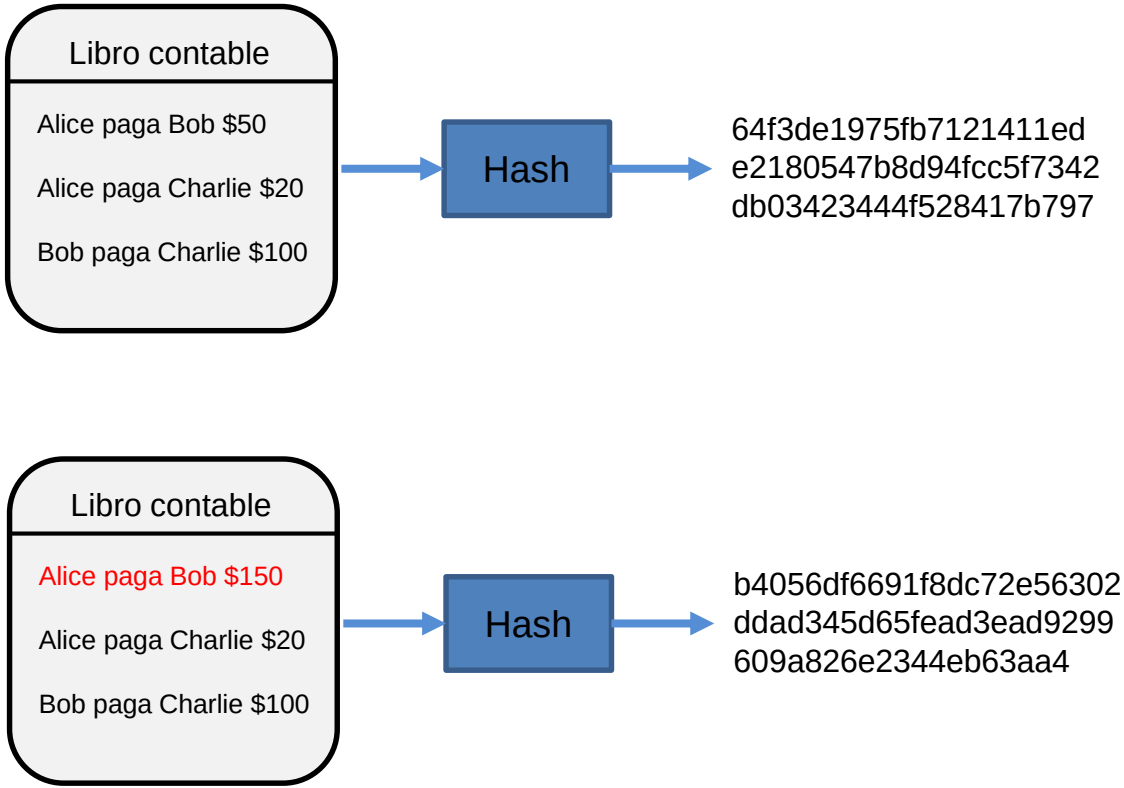
Propiedad 1

Resistencia a colisiones



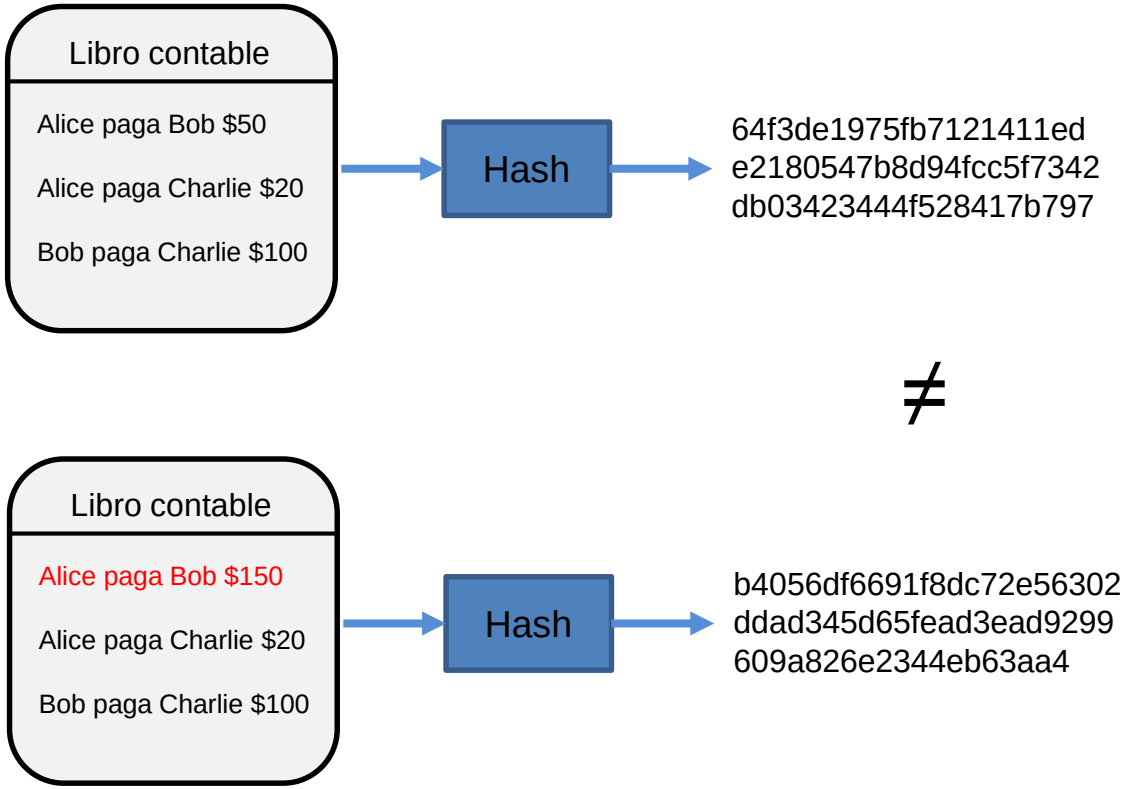
Propiedad 1

Resistencia a colisiones



Propiedad 1

Resistencia a colisiones



Propiedad 1

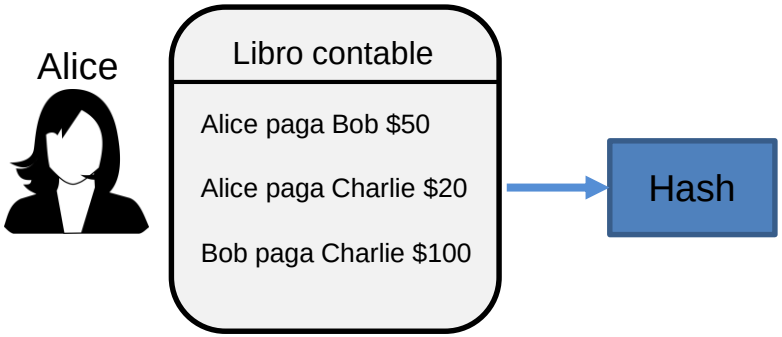
Resistencia a colisiones

Alice

Libro contable
Alice paga Bob $50
Alice paga Charlie $20
Bob paga Charlie $100

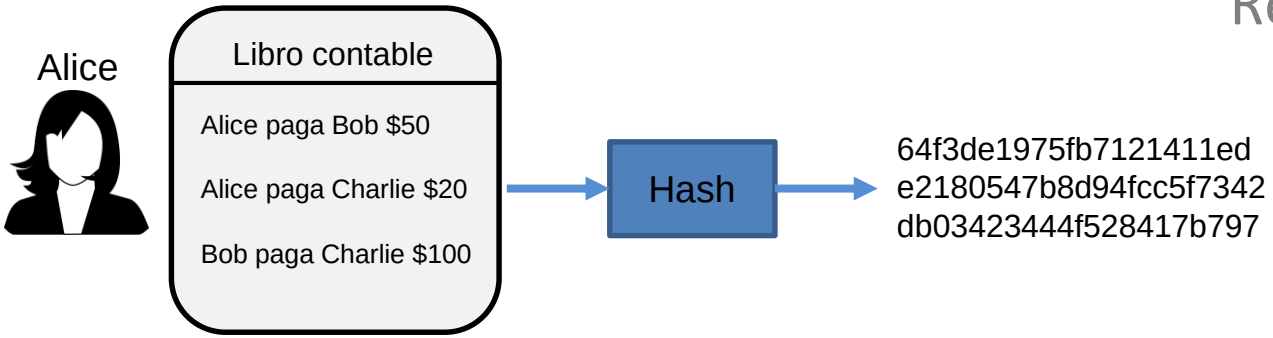
Propiedad 1

Resistencia a colisiones



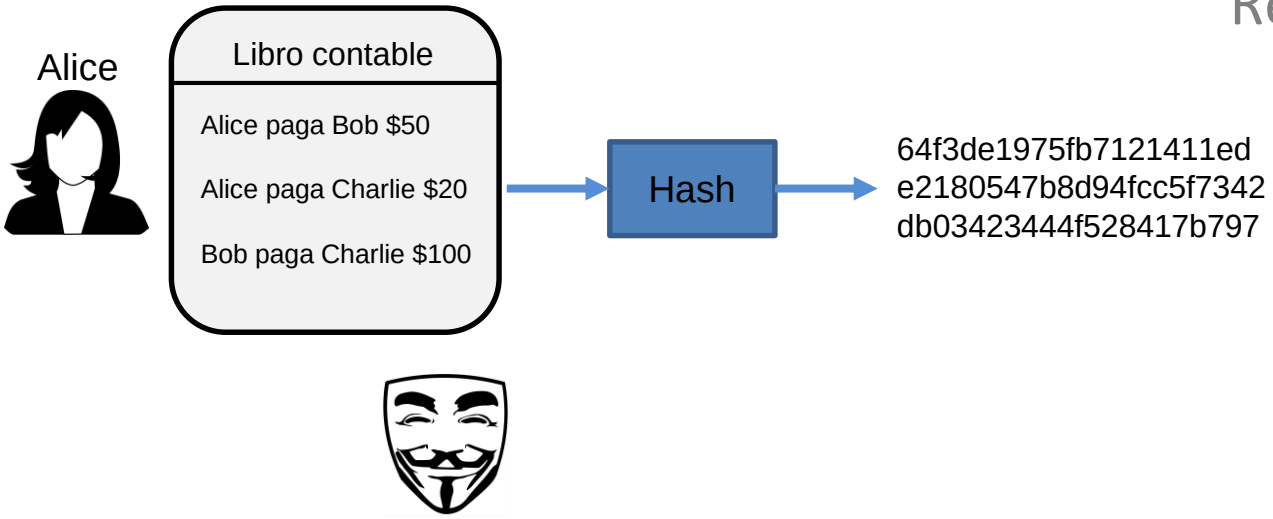
Propiedad 1

Resistencia a colisiones



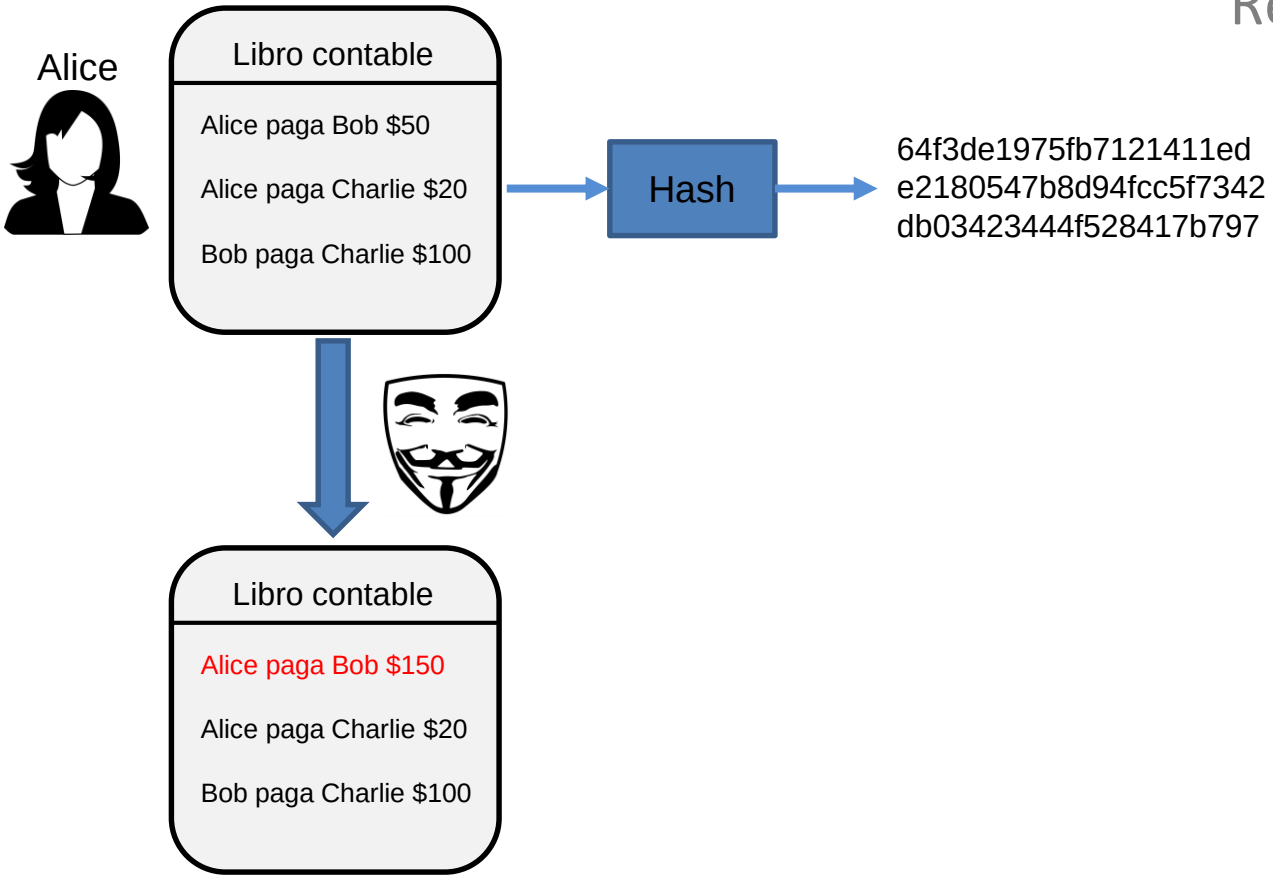
Propiedad 1

Resistencia a colisiones



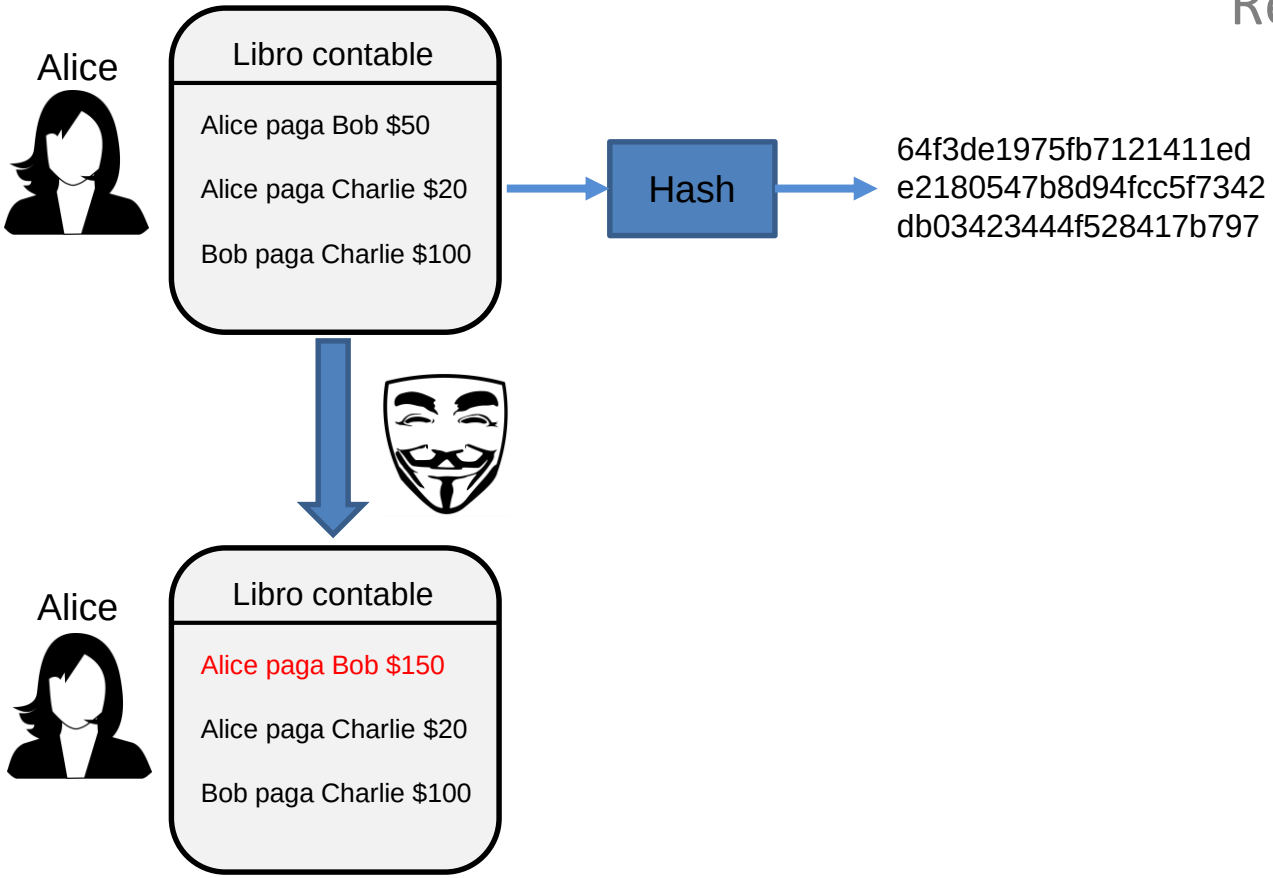
Propiedad 1

Resistencia a colisiones



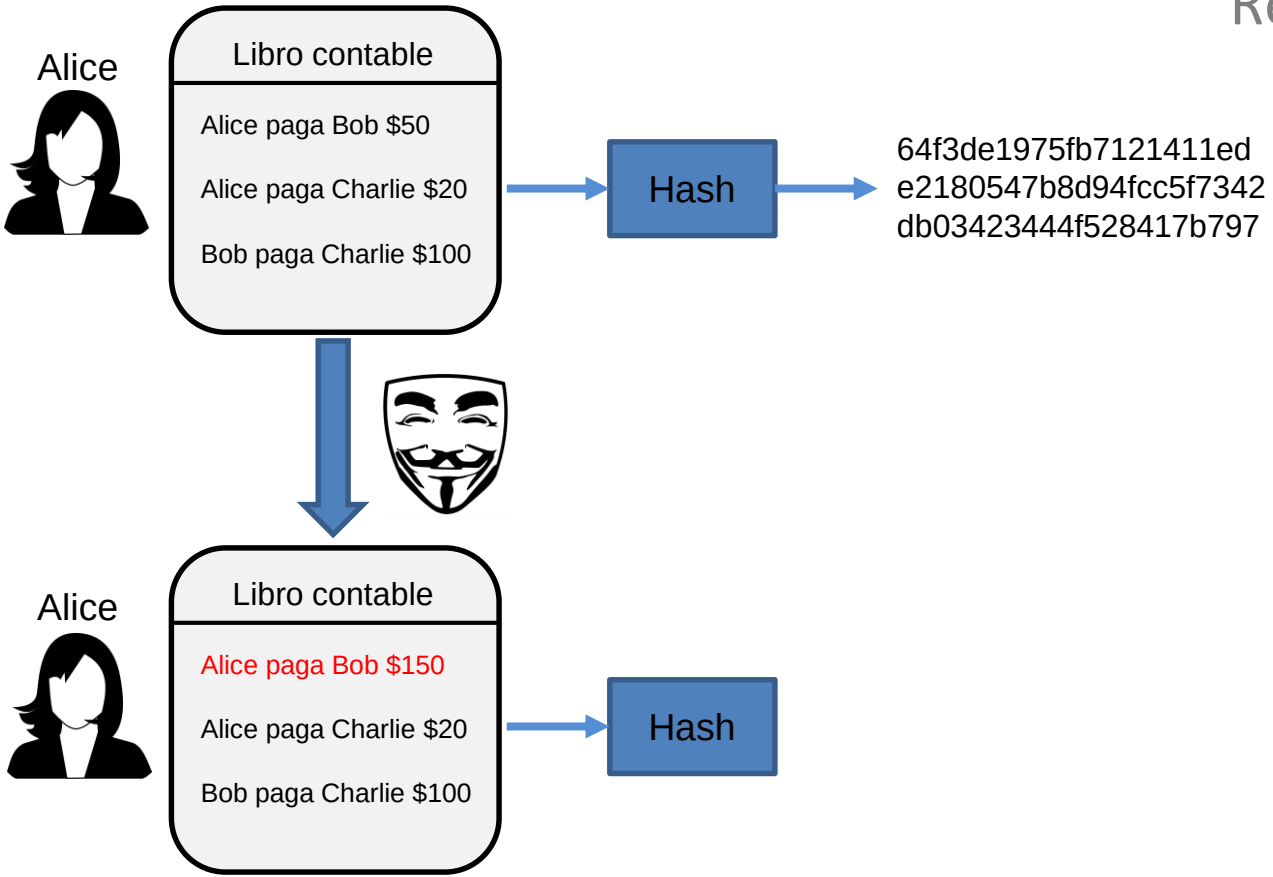
Propiedad 1

Resistencia a colisiones



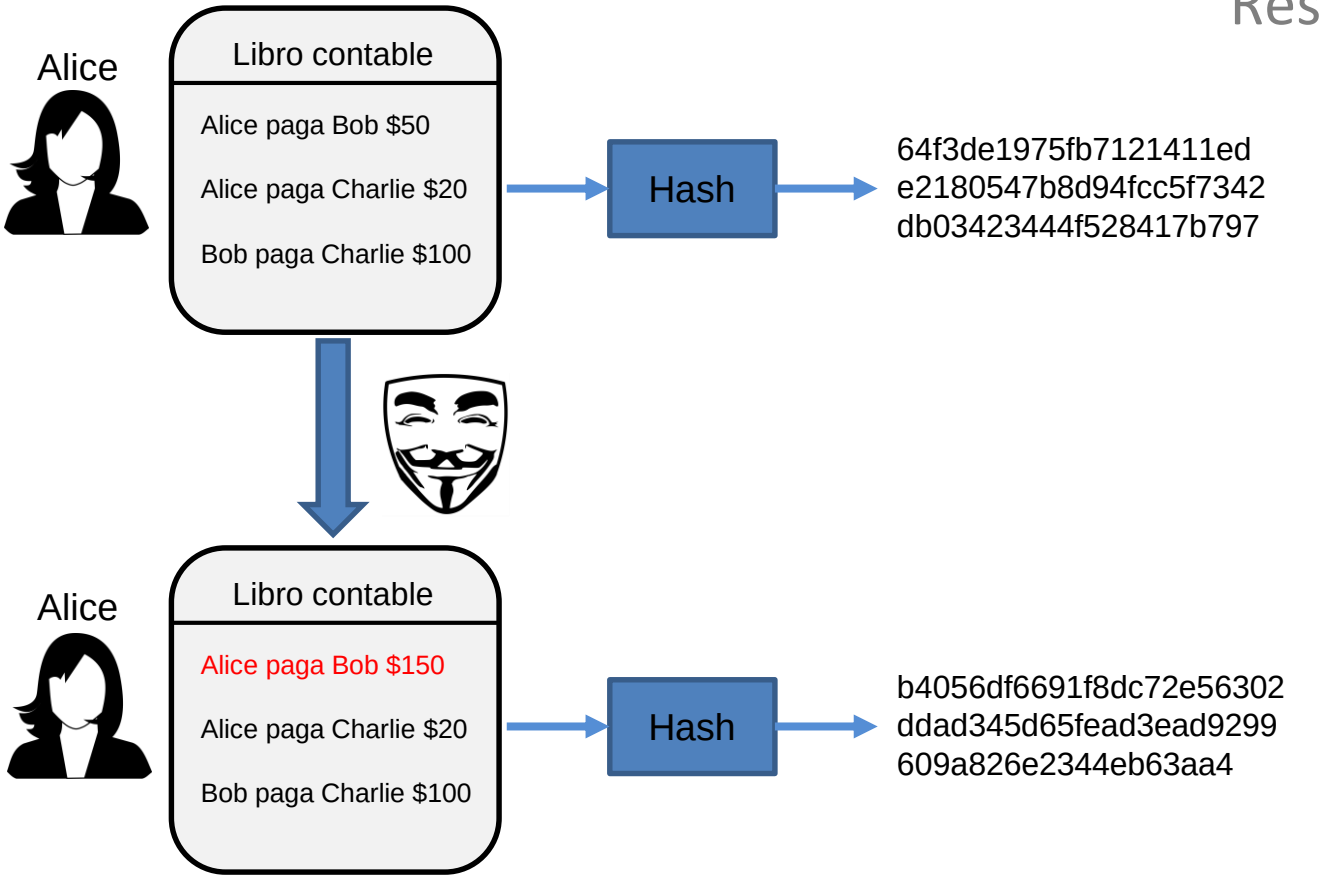
Propiedad 1

Resistencia a colisiones



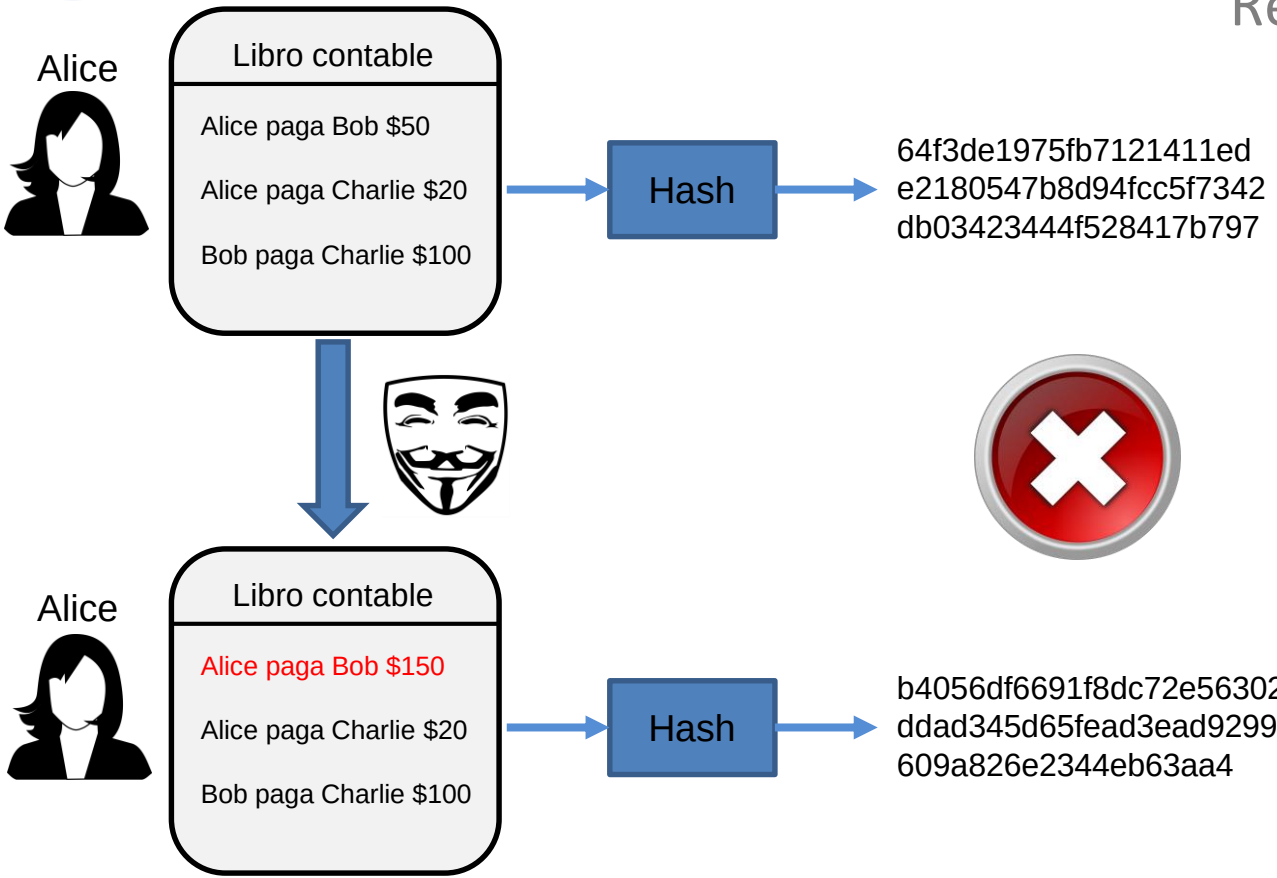
Propiedad 1

Resistencia a colisiones



Propiedad 1

Resistencia a colisiones



Problema 1 de ePeso

Alice

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

Bob

Charlie

Problema 1 de ePeso

Libro contable puede pesar mucho

Libro contable

Alice

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

Bob

Charlie

Problema 1 de ePeso

Libro contable puede pesar mucho

Libro contable

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

Alice

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

Bob

Charlie

Libro contable

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

Problema 1 de ePeso

Libro contable puede pesar mucho

Libro contable

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

Alice

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

Bob

Libro contable

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

Charlie

Libro contable

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

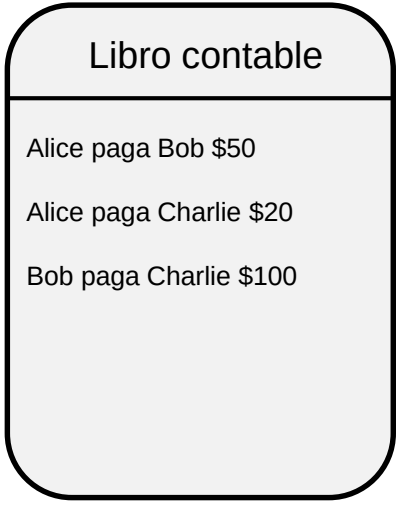
Bob paga Charlie $100

Problema 1 de ePeso

Libro contable puede pesar mucho

Libro contable

Alice



200GB

Bob

Libro contable

Charlie

Libro contable

Problema 1 de ePeso

Libro contable puede pesar mucho

Libro contable

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

200GB

Alice

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

200GB

Bob

Libro contable

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

200GB

Charlie

200GB

Libro contable

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

Libro contable

Problema 1 de ePeso

Libro contable puede pesar mucho

64f3de1975fb7121411ed
e2180547b8d94fcc5f7342
db03423444f528417b797

Alice

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

200GB

Bob

64f3de1975fb7121411ed
e2180547b8d94fcc5f7342
db03423444f528417b797

Charlie

64f3de1975fb7121411ed
e2180547b8d94fcc5f7342
db03423444f528417b797

Problema 1 de ePeso

Libro contable puede pesar mucho

256 bits

64f3de1975fb7121411ed
e2180547b8d94fcc5f7342
db03423444f528417b797

Alice

Libro contable

Alice paga Bob $50

Alice paga Charlie $20

Bob paga Charlie $100

200GB

Bob

64f3de1975fb7121411ed
e2180547b8d94fcc5f7342
db03423444f528417b797

256 bits

Charlie

256 bits

64f3de1975fb7121411ed
e2180547b8d94fcc5f7342
db03423444f528417b797

Propiedad 1

Resistencia a colisiones

Output de H es 256 bits. Como encontrar colisiones?

- Computar hashes de 0 hasta $2^{256} + 1$

Pregunta: cuanto se demora esto computando 1000000 hashes por segundo?

Propiedad 1

Resistencia a colisiones

Usando paradoja de cumpleaños:

- Vamos a reducir el número a $2^{130} + 1$
- Para tener la probabilidad 99.8 (pizarra)
- Es mejor?

Bitcoin: 50.000.000 Th/sec y todavía no encontraron una colisión:

- Tiempo esperado ~ 10^{11} años
- Edad de universo ~ 1.3 x 10^{10} años

Que es función de hash criptográfica?

Función de hash con ciertas propiedades de seguridad:

- Resistencia a colisiones
- **Hiding (función que esconde el input)**
- Puzzle friendliness (para hacer el mineo)

Propiedad 2

Hiding

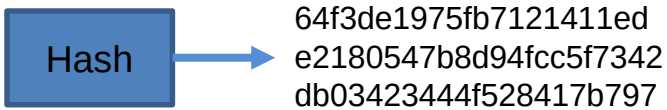
Función de hash H es **hiding** si:

- Dado $y = H(x)$ (pero no el x)
- No es factible encontrar el x

Propiedad 2

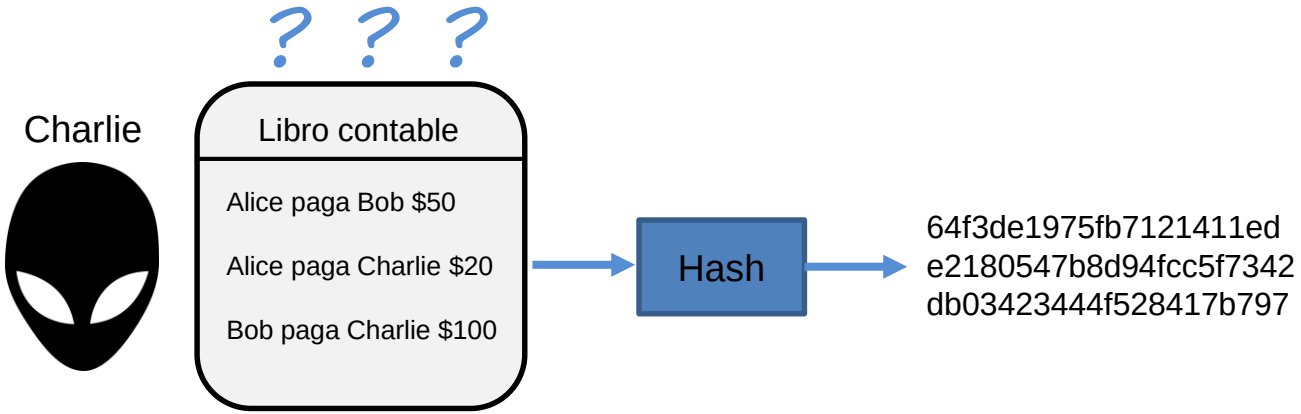
Esconder el input

Charlie



Propiedad 2

Esconder el input



Propiedad 2

Hiding

Función de hash *H* es **hiding** si:

- Dado *y = H(x)* (pero no el *x*)
- No es factible encontrar el *x*

Si el conjunto de entradas no es grande no funciona (coin flip)!!!

Propiedad 2

Hiding

Función de hash *H* es **hiding** si:

- Dado $y = H(r \,||\, x)$
- *r se toma aleatoriamente de un conjunto grande*
- No es factible encontrar el *x*

r en bitcoin se toma desde $[0,\ldots,2^{256}]$

Uso de hiding: un compromiso

Queremos simular poner un sobre sellado en la mesa:

- Escribo un mensaje
- Lo sello en un sobre
- Lo pongo en la mesa para que todos lo miran
- Despues si abro el sobre todos ven mi mensaje

Protocolo de compromiso digital

Dos algoritmos:

- *com := commit(msg,nonce)*
- *verify(com,msg,nonce)* returns *true/false*

nonce aleatorio (siempre)

Protocolo de compromiso digital

Propiedades de dos algoritmos:

- **Hiding**: dado *com* no es factible encontrar *msg*
- **Binding**: no es factible encontrar *(msg,nonce)* y *(msg',nonce')* t.q. *commit(msg,nonce) == commit(msg',nonce')*

Protocolo de compromiso digital

commit(msg,nonce) := H(msg || nonce):

- **Hiding**: hiding de *H*
- **Binding**: resistencia colisiones de *H*

Que es función de hash criptográfica?

Función de hash con ciertas propiedades de seguridad:

- Resistencia a colisiones
- Hiding (función que esconde el input)
- **Puzzle friendliness (para hacer el mineo)**

Propiedad 3

Puzzle friendliness

Función de hash *H* es **puzzle friendly** si:

- Dado *n*-bit output *y* de *H*
- Y dado un *k* (de un conjunto grande, uniforme distr.)
- No es factible encontrar *x* con $H(k \,||\, x) = y$
- En tiempo menos de $O(2^n)$

Propiedad 3

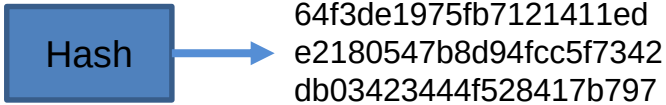
Puzzle friendliness

Charlie

Propiedad 3

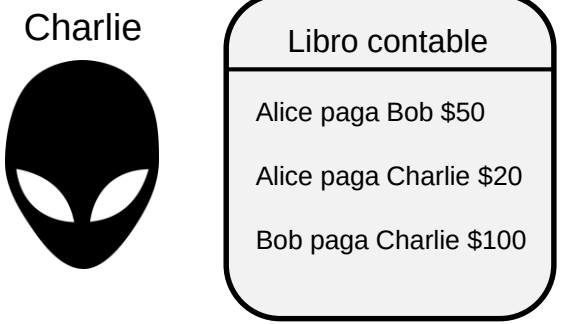
Puzzle friendliness

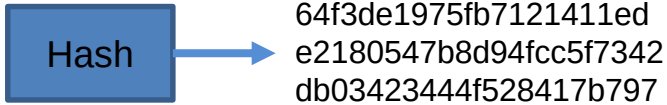
Charlie



Propiedad 3

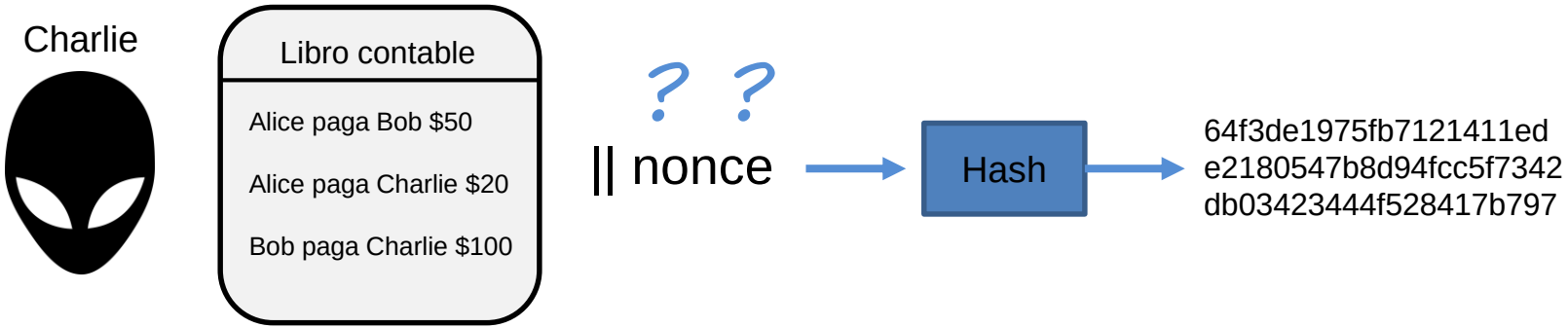
Puzzle friendliness





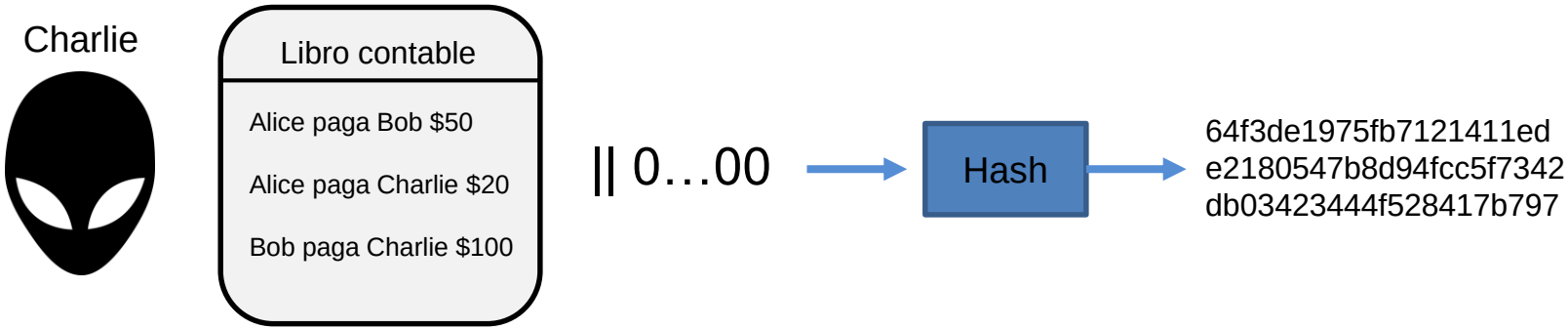
Propiedad 3

Puzzle friendliness



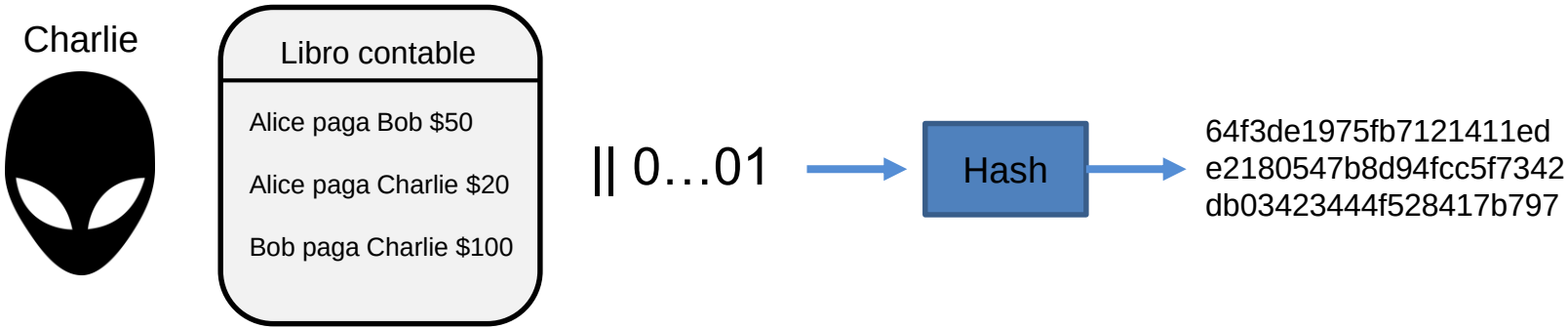
Propiedad 3

Puzzle friendliness



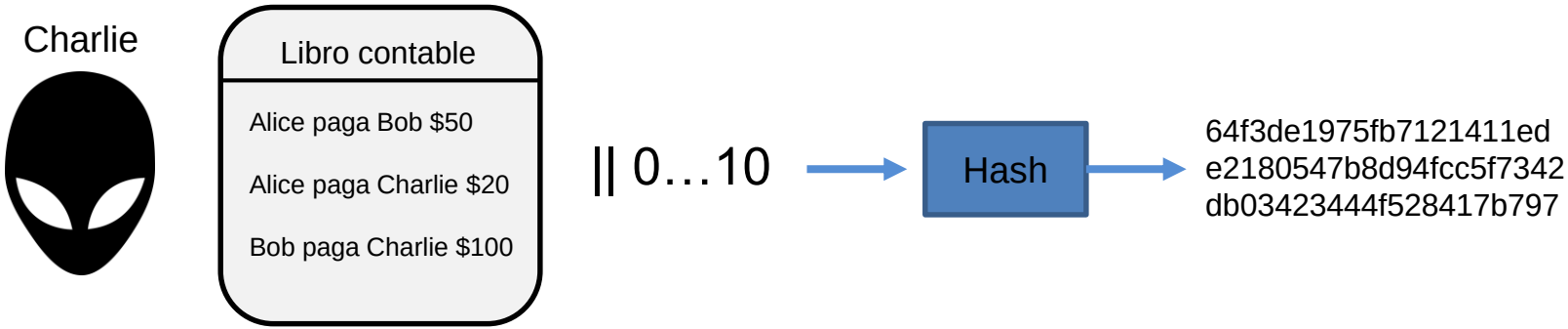
Propiedad 3

Puzzle friendliness



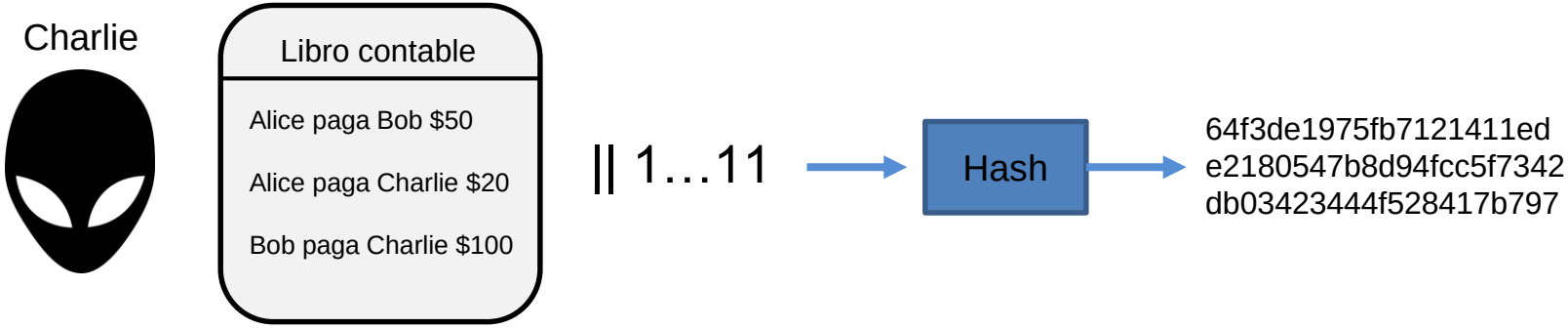
Propiedad 3

Puzzle friendliness



Propiedad 3

Puzzle friendliness



Mining

Search puzzle

Un **search puzzle** consiste de:

- Función de hash criptografica *H*
- Un puzzle *ID*
- Un conjunto de metas *Y*

Una **solución para el search puzzle** es un valor x t.q

- *H(ID || x)* pertenece a *Y*

Mining

Search puzzle

Una **solución para el search puzzle** es un valor x t.q

- *H(ID || x)* pertenece a *Y*

Puzzle friendly: cualquier *x* es igualmente probable para resolver el puzzle

Mining

Search puzzle

Dificultad del search puzzle: tamaño de Y

- Y = [0,…,2^{256}] en un H con 256 bits es trivial
- Y = {y} dificultad máxima
- Todos los tamaños intermedios

Así se ajusta la dificultad de mineo en Bitcoin

Función de hash en Bitcoin

SHA-256

NIST/NSA standard

Propiedad general de funciones de hash:

- Input de tamaño fijo – *m*
- Output de tamaño fijo – *n*
- Transformación Merkle-Dagmår

Función de hash en Bitcoin

SHA-256

Transformación Merkle-Dagmår:

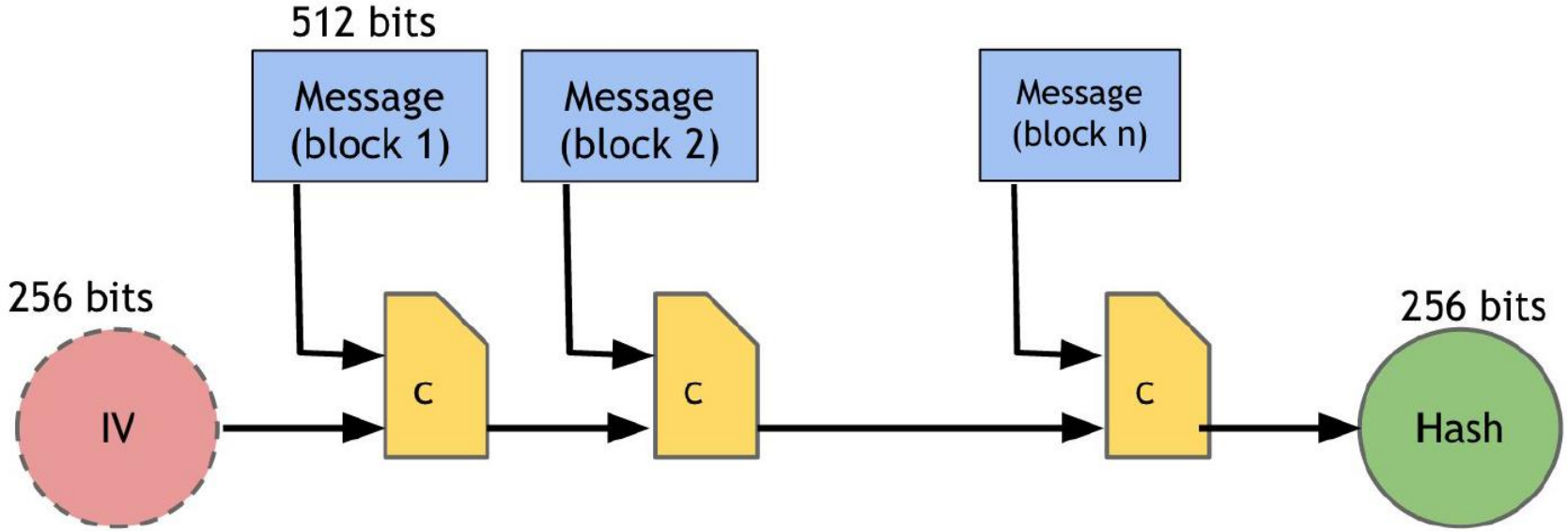
- Dividir input en bloques de tamaño m-n
- Procesar uno por uno con el output previo
- Usar un vector de inicialización (IV)

Caso de SHA-256:

- Input de tamaño 768 bits
- Bloques de tamaño 512 bits
- Output 256 bits

Función de hash en Bitcoin

SHA-256



Ejercicios

SHA-256

(asumiendo Python 3+)

(https://www.python.org/)

Necesitamos: hashlib (ya debería estar)

Funciones de hash en Python

Computando un hash

hash.py

```python
import hashlib

# normal sha256
def hash(message):
    # will return bytes of the created sha256 object
    return hashlib.sha256(message).digest()


# double sha256
def hash256(message):
    '''two rounds of sha256'''
    # will return bytes of the created sha256 object
    return hashlib.sha256(hashlib.sha256(message).digest()).digest()

```

Funciones de hash en Python

Computando un hash

hash_test.py

```python
from hash import hash

data1 = b'Criptomonedas'
hash1 = hash(data1)
print("Bytes: ",hash1)
hash1_hex = hash1.hex()
print("Hex:   ",hash1_hex)

data2 = b'criptomonedas'
hash2 = hash(data2)
print("Bytes: ",hash2)
hash2_hex = hash2.hex()
print("Hex:   ",hash2_hex)
```

```
Bytes:  b'd\xf3\xde\x19u\xfbq!A\x1e\xde!\x80T{\x8d\x94\xfc\xc5\xf74-\xb04#DOR\x84\x17\xb7\x97'
Hex:    64f3de1975fb7121411ede2180547b8d94fcc5f7342db03423444f528417b797
Bytes:  b'\xbb\xfe3R\xd8U\xca\x12&\xba\xdd%p\xe2\x84\xfat\xafwx\xd5c\xc5Iw\xdf8uc\x82J\xb4'
Hex:    bbfe3352d855ca1226badd2570e284fa74af7778d563c54977df387563824ab4
```

Ejercicios

SHA-256

Programar módulo de minar:

- Input: puzzle *ID* (como un SHA-256 hash)
- Input: conjunto de metas (dado como *y* en $[0,\ldots,2^{256}]$)
- Meta: $\{ z \text{ en } [0,\ldots,2^{256}] : z < y \}$ (como chequear esto rápido?)
- Módulo busca el *nonce* t.q. $SHA256(ID \,||\, nonce) < y$

Pensar en distintos modos de buscar el nonce!!!

Cual será su mejor estrategia si están usando solo su computador?

Ejercicios

En Bitcoin

Minar en Bitcoin usa los siguientes parámetros:

- Input: puzzle *ID* (como un SHA-256 hash)
- Input: conjunto de metas (dado como *y* en $[0,\ldots,2^{256}-1]$)
- Meta: $\{ z \text{ en } [0,\ldots,2^{256}-1] : z < y \}$ (como chequear esto rápido?)
- Módulo busca el *nonce* t.q. $SHA256(ID \,||\, nonce) < y$
- Pero *nonce* de Bitcoin pertenece a $[0,\ldots,2^{32}-1]$

Hay puzzles sin solución!!!

Como lo resuelven en Bitcoin???

Ejercicios

Para tener estrategias

Usemos los siguientes parámetros:

- Input: puzzle *ID* (como un SHA-256 hash)
- Input: conjunto de metas (dado como *y* en $[0,\ldots,2^{256}-1]$)
- Meta: $\{ z \text{ en } [0,\ldots,2^{256}-1] : z < y \}$
- Módulo busca el *nonce* t.q. $SHA256(ID \,||\, nonce) < y$
- Pero *nonce* pertenece a $[0,\ldots,2^{23}-1]$

Detectar cuando no hay solución!!!! (o estimarlo)

Ejercicios

Posibles estrategias

nonce pertenece a *[0,…,2^{23} -1] (cca 40 segundos en mi compu):*

- Minar en el orden creciente
- Minar en el orden decreciente
- Minar aleatoriamente

Probemos los tres con distintas metas!!!

Ejercicios

SHA-256

Mining pools:

- Dividir el espacio de busqueda entre los participantes
- Cada miembro mina cierto rango de inputs

Competencia entre pools para distintas metas:

- Jugar con la dificultad
- *nonce* en $[0,\ldots,2^{22}-1]$