Transaction mechanism of Bitcoin

How Bitcoin works?

Content

We will define two (centralized) cryptocurrencies:

- **Goofycoin (a bad one)**
- Scroogecoin (a not so bad one, but still a bad one)

Goofycoin

Participants

Goofy

Goofycoin

Participants

Goofy

Alice

Bob

Charlie

Goofycoin

Participants

Goofy

(SK_G, PK_G)

Alice

(SK_A, PK_A)

Bob

(SK_B, PK_B)

Charlie

(SK_C, PK_C)

Goofycoin

Participants

Goofy

(SK_G, PK_G)

Alice

(SK_A, PK_A)

Bob

(SK_B, PK_B)

Charlie

(SK_C, PK_C)

No need to register

We only need to know PK_G

Goofycoin

Three rules

1. Creating a goofycoin
2. Transfering a goofycoin
3. Verifying that a goofycoin is valid

Rule 1

Creating a goofycoin

(SK_G, PK_G)

Rule 1

Creating a goofycoin

(SK_G, PK_G)

CreateCoin [UniqueCoinID]

Rule 1

Creating a goofycoin

(SK_G, PK_G)

Signed SK_G
CreateCoin [UniqueCoinID]

Rule 1

Creating a goofycoin

(SK_G, PK_G)

Signed SK_G
CreateCoin [UniqueCoinID]

Goofy is the owner of every new goofycoin!!!

Rule 1

Creating a goofycoin

(SK_G, PK_G)

CreateCoin CoinID(27)

Rule 1

Creating a goofycoin

(SK_G, PK_G)

Signature of "CreateCoin CoinID(27)" with SK_G
CreateCoin CoinID(27)

Rule 1

Creating a goofycoin

(SK_G, PK_G)

Signature of "CreateCoin CoinID(27)" with SK_G
CreateCoin CoinID(27)

Goofy ows goofycoin CoinID(27)

Rule 2

Transfers

(SK_A, PK_A)

(SK_G, PK_G)

Rule 2

Transfers

(SK_A, PK_A)

(SK_G, PK_G)

Signed SK_G
CreateCoin CoinID(27)

Rule 2

Transfers

(SK_A, PK_A)

(SK_G, PK_G)

Coin27

Signed SK_G
CreateCoin CoinID(27)

Rule 2

Transfers

(SK_A, PK_A)

Pay to PK_A : Hp(Coin27)

(SK_G, PK_G)

Coin27

Signed SK_G
CreateCoin CoinID(27)

Rule 2

Transfers

(SK_A, PK_A)

(SK_G, PK_G)

Pay to PK_A : Hp(Coin27)

Coin27

Signed SK_G

CreateCoin CoinID(27)

Rule 2

Transfers

(SK_A, PK_A)

(SK_G, PK_G)

Signed SK_G

Pay to PK_A : Hp(Coin27)

Coin27

Signed SK_G

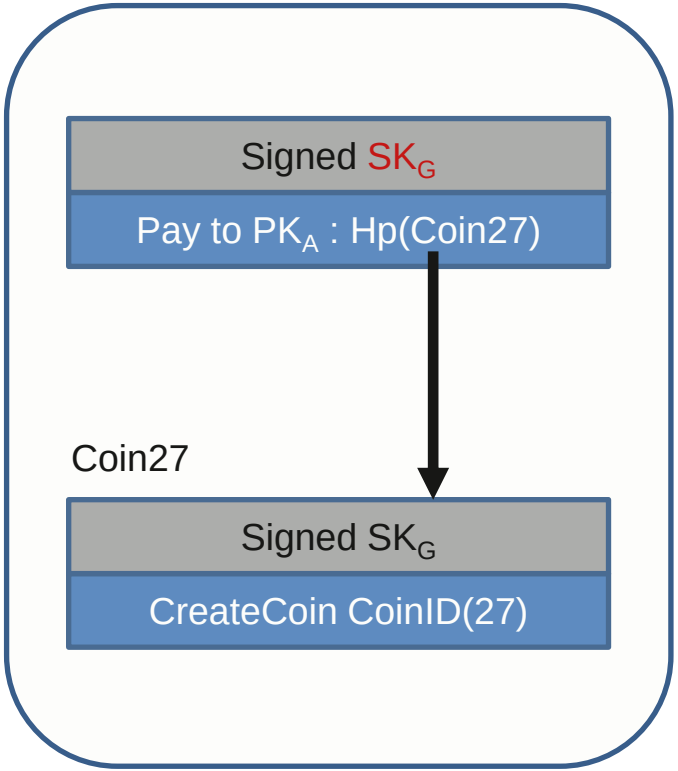
CreateCoin CoinID(27)

Rule 2

Transfers

(SK_A, PK_A)

(SK_G, PK_G)



Alice is the owner of the goofycoin with ID 27

Rule 2

Transfers

(SK_A, PK_A)

Rule 2

Transfers

(SK_A, PK_A)

Coin27[1]

Signed SK_G
Pay to PK_A : Hp(Coin27)

Coin27

Signed SK_G
CreateCoin CoinID(27)

Rule 2

Transfers

(SK_C, PK_C)

(SK_A, PK_A)

Coin27[1]

Signed SK_G
Pay to PK_A : Hp(Coin27)

Coin27

Signed SK_G
CreateCoin CoinID(27)

Rule 2

Transfers

(SK_C, PK_C)

(SK_A, PK_A)

Pay to PK_C : Hp(Coin27[1])

Coin27[1]

Signed SK_G

Pay to PK_A : Hp(Coin27)

Coin27

Signed SK_G

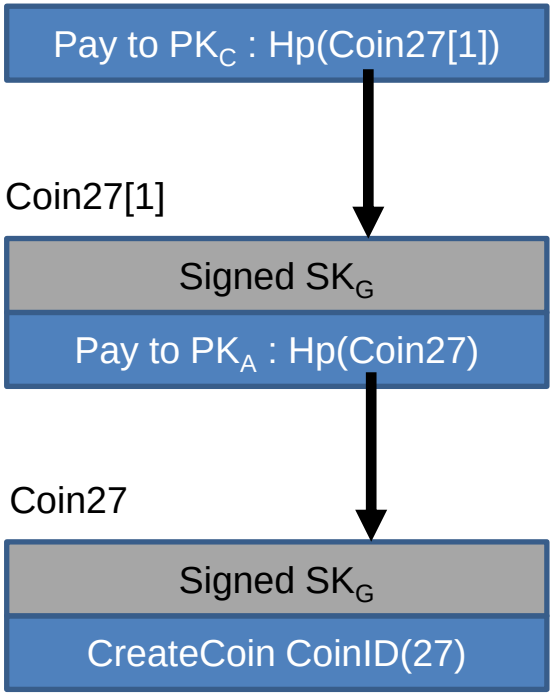
CreateCoin CoinID(27)

Rule 2

Transfers

(SK_C, PK_C)

(SK_A, PK_A)



Rule 2

Transfers

(SK_C, PK_C)

(SK_A, PK_A)

Signed SK_A

Pay to PK_C : Hp(Coin27[1])

Coin27[1]

Signed SK_G

Pay to PK_A : Hp(Coin27)

Coin27

Signed SK_G

CreateCoin CoinID(27)

Rule 2

Transfers

(SK_C, PK_C)

(SK_A, PK_A)

Coin27[2]

Signed SK_A
Pay to PK_C : Hp(Coin27[1])

Coin27[1]

Signed SK_G
Pay to PK_A : Hp(Coin27)

Coin27

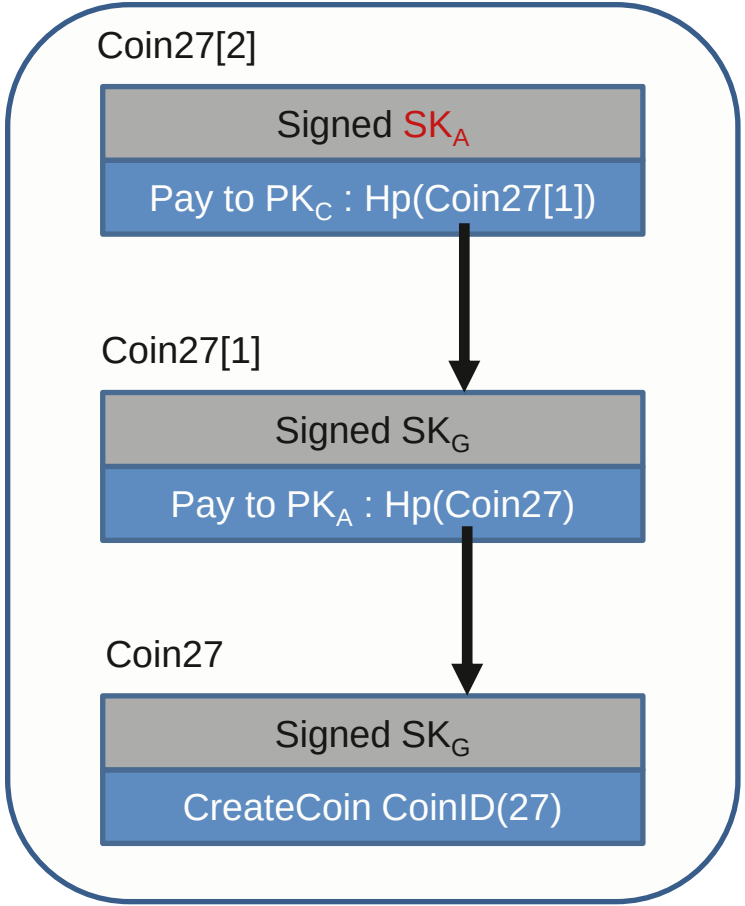
Signed SK_G
CreateCoin CoinID(27)

Rule 2

Transfers

(SK_C, PK_C)

(SK_A, PK_A)



Rule 3

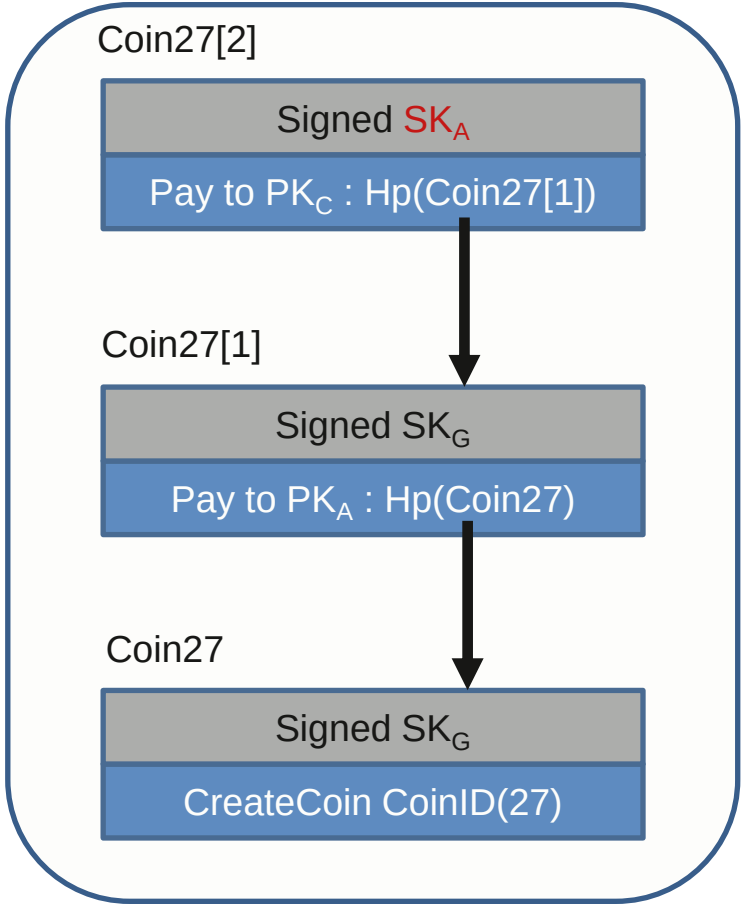
Verification

(SK_C, PK_C)

Rule 3

Verification

(SK_C, PK_C)



Rule 3

Verification

(SK_C, PK_C)

Coin27[2]

Signed SK_A

Pay to PK_C : Hp(Coin27[1])

Coin27[1]

Signed SK_G

Pay to PK_A : Hp(Coin27)

Coin27

Signed SK_G

CreateCoin CoinID(27)

Rule 3

Verification

(SK_C, PK_C)

Coin27[2]

Signed SK_A

Pay to PK_C : Hp(Coin27[1])

Coin27[1]

Signed SK_G

Pay to PK_A : Hp(Coin27)

Coin27

Signed SK_G

CreateCoin CoinID(27)

Rule 3

Verification

(SK_C, PK_C)

Coin27[2]

Signed SK_A

Pay to PK_C : Hp(Coin27[1])

Coin27[1]

Signed SK_G

Pay to PK_A : Hp(Coin27)

Coin27

Signed SK_G

CreateCoin CoinID(27)

Rule 3

Verification

(SK_C, PK_C)

Coin27[2]

Signed SK_A

Pay to PK_C : Hp(Coin27[1])

Coin27[1]

Signed SK_G

Pay to PK_A : Hp(Coin27)

Coin27

Signed SK_G

CreateCoin CoinID(27)

Rule 3

Verification

(SK_C, PK_C)

Coin27[2]

Signed SK_A

Pay to PK_C : Hp(Coin27[1])

Coin27[1]

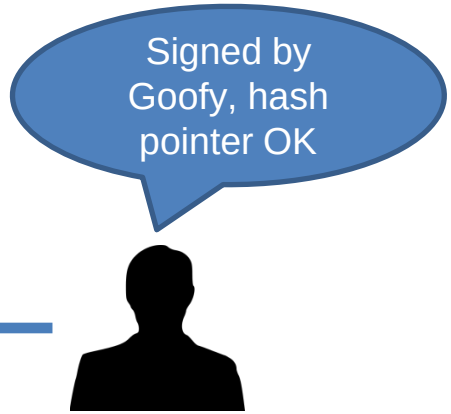
Signed SK_G

Pay to PK_A : Hp(Coin27)

Coin27

Signed SK_G

CreateCoin CoinID(27)



Rule 3

Verification

(SK_C, PK_C)

Coin27[2]

Signed SK_A

Pay to PK_C : Hp(Coin27[1])

Coin27[1]

Signed SK_G

Pay to PK_A : Hp(Coin27)

Coin27

Signed SK_G

CreateCoin CoinID(27)

Rule 3

Verification

(SK_C, PK_C)

Coin27[2]

Signed SK_A

Pay to PK_C : Hp(Coin27[1])

Coin27[1]

Signed SK_G

Pay to PK_A : Hp(Coin27)

Coin27

Signed SK_G

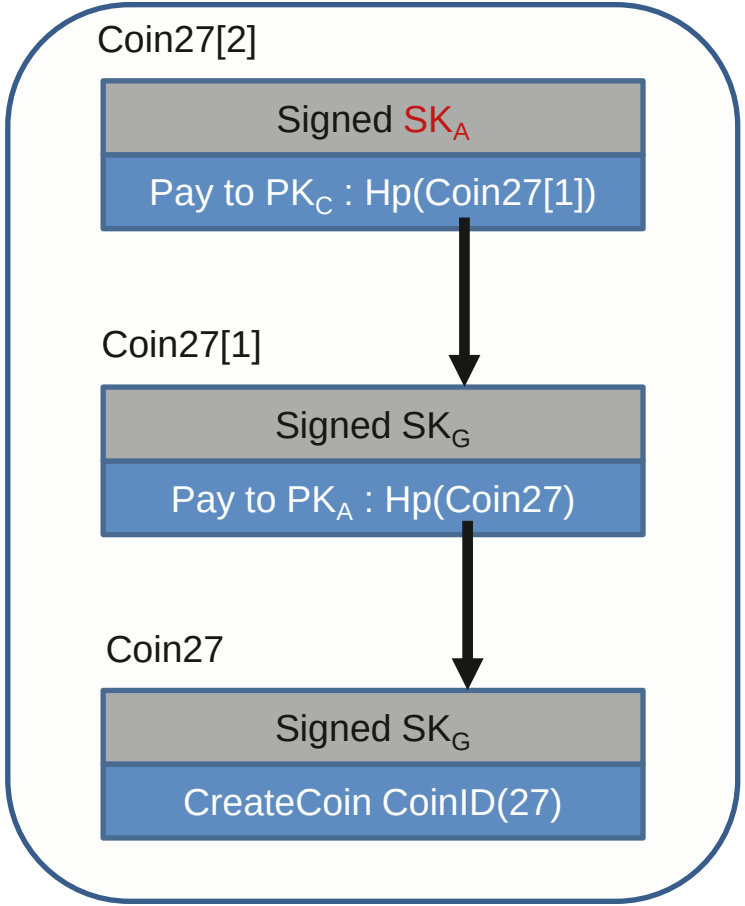
CreateCoin CoinID(27)

Signed by Alice, Hp OK

Rule 3

Verification

(SK_C, PK_C)



Goofycoin

Three rules

1. Goofy creates all goofycoins
2. Transfering goofycoins
3. Verifying that a goofycoin is valid

An issue with Goofycoin

Double spending attack

(SK_A, PK_A)

An issue with Goofycoin

Double spending attack

(SK_A, PK_A)

Coin27[1]

Signed SK_G
Pay to PK_A : Hp(Coin27)

Coin27

Signed SK_G
CreateCoin CoinID(27)

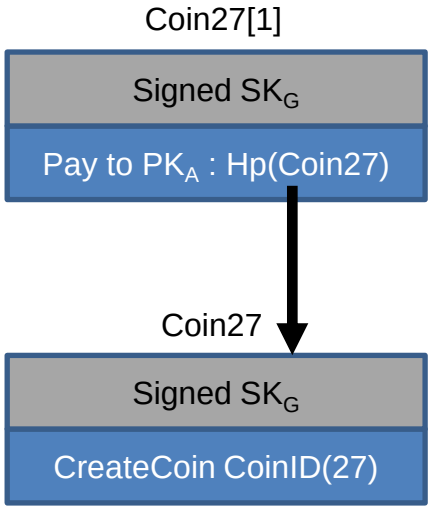
An issue with Goofycoin

Double spending attack

(SK_C, PK_C)

(SK_B, PK_B)

(SK_A, PK_A)



An issue with Goofycoin

Double spending attack

(SK_C, PK_C)

Coin27[2]

Signed SK_A
Pay to PK_C : Hp(Coin27[1])

(SK_B, PK_B)

Coin27[1]

Signed SK_G
Pay to PK_A : Hp(Coin27)

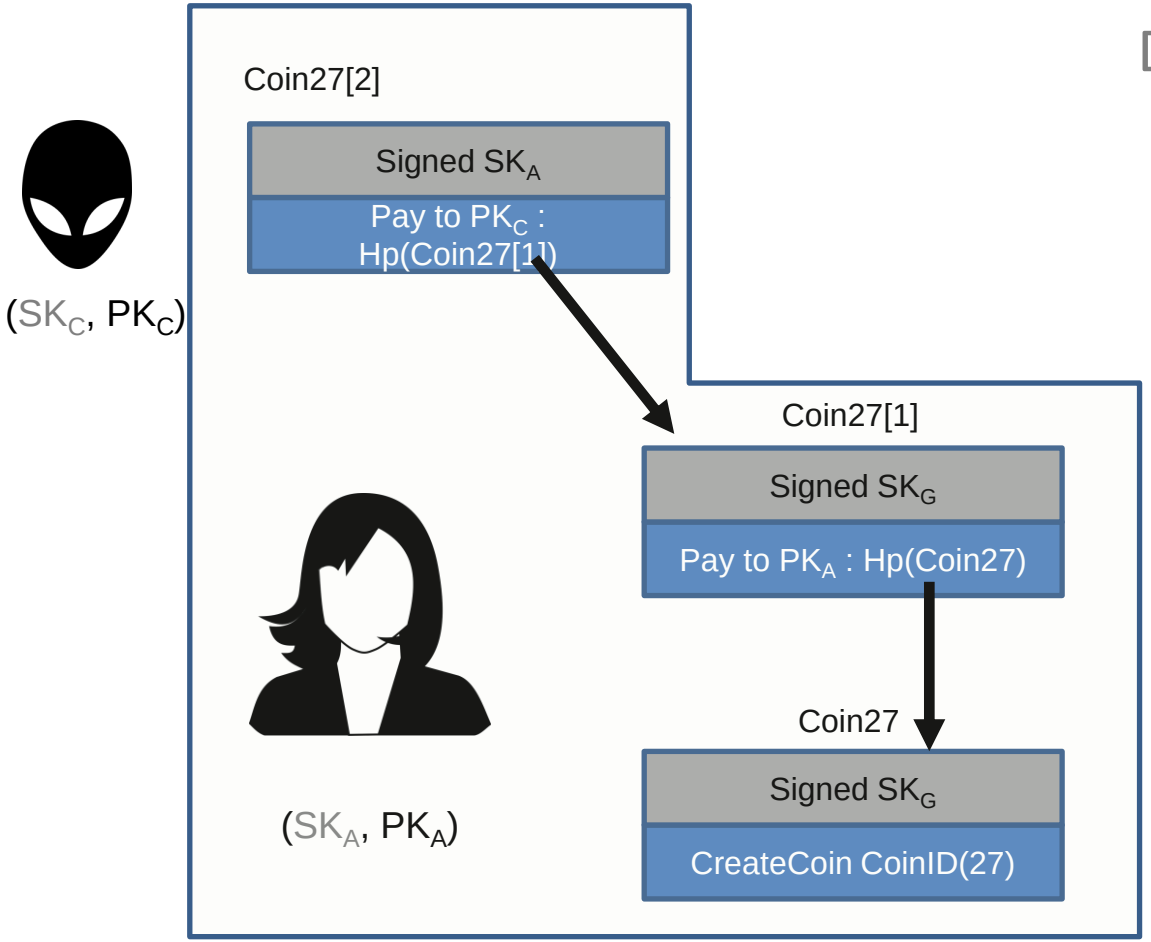
(SK_A, PK_A)

Coin27

Signed SK_G
CreateCoin CoinID(27)

An issue with Goofycoin

Double spending attack



(SK_B, PK_B)

An issue with Goofycoin

Double spending attack

(SK_C, PK_C)

Coin27[2]

Signed SK_A
Pay to PK_C : Hp(Coin27[1])

(SK_B, PK_B)

Coin27[1]

Signed SK_G
Pay to PK_A : Hp(Coin27)

(SK_A, PK_A)

Coin27

Signed SK_G
CreateCoin CoinID(27)

An issue with Goofycoin

Double spending attack

(SK_C, PK_C)

Coin27[2]

Signed SK_A
Pay to PK_C : Hp(Coin27[1])

Coin27[2]

Signed SK_A
Pay to PK_B : Hp(Coin27[1])

(SK_B, PK_B)

Coin27[1]

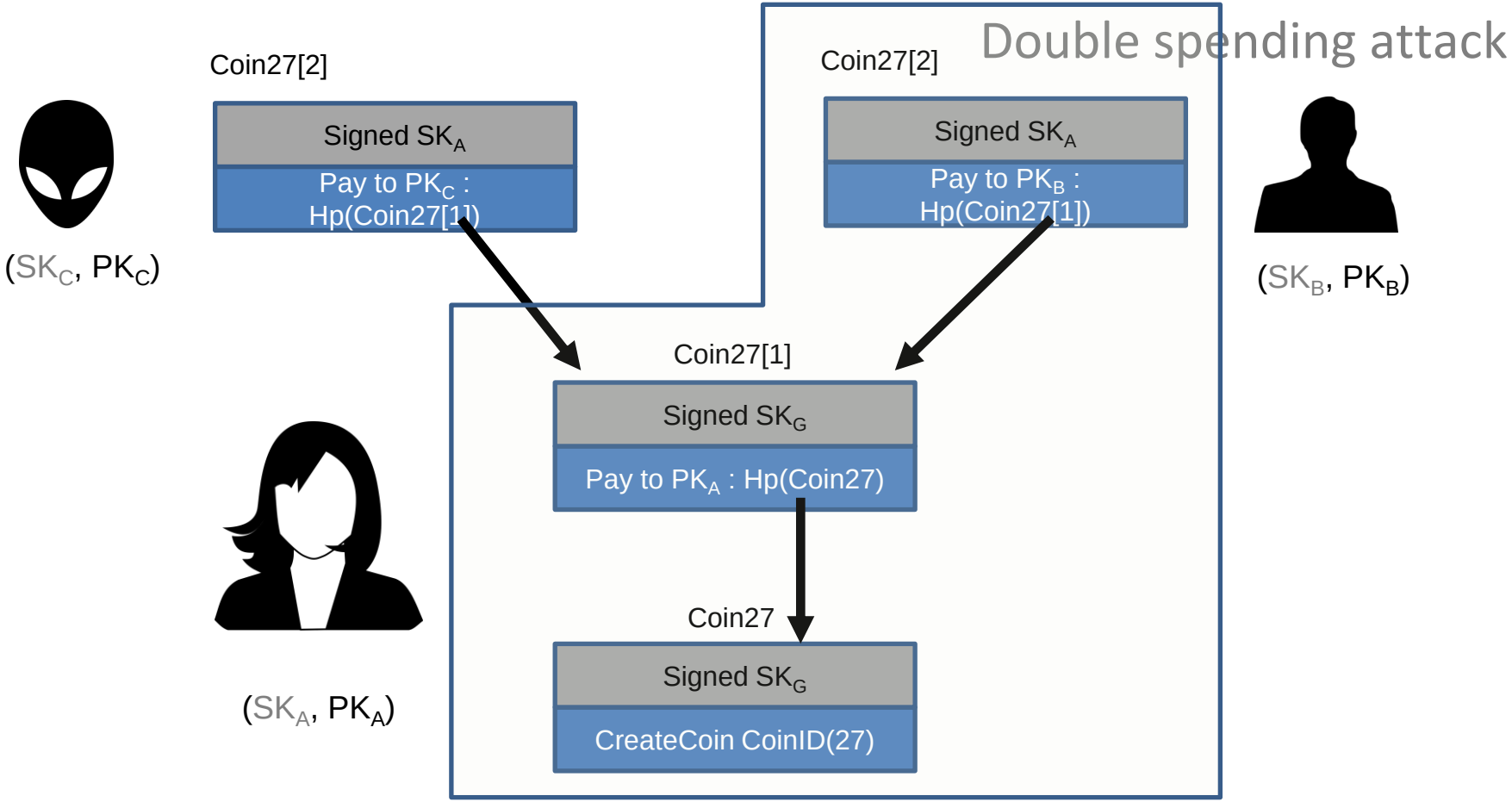
Signed SK_G
Pay to PK_A : Hp(Coin27)

(SK_A, PK_A)

Coin27

Signed SK_G
CreateCoin CoinID(27)

An issue with Goofycoin



An issue with Goofycoin

Double spending attack

(SK_C, PK_C)

Coin27[2]

Signed SK_A
Pay to PK_C : Hp(Coin27[1])

Coin27[2]

Signed SK_A
Pay to PK_B : Hp(Coin27[1])

(SK_B, PK_B)

Coin27[1]

Signed SK_G
Pay to PK_A : Hp(Coin27)

(SK_A, PK_A)

Coin27

Signed SK_G
CreateCoin CoinID(27)

An issue with Goofycoin

Double spending attack

(SK_C, PK_C)

Coin27[2]

Signed SK_A
Pay to PK_C : Hp(Coin27[1])

Coin27[2]

Signed SK_A
Pay to PK_B : Hp(Coin27[1])

(SK_B, PK_B)

Coin27[1]

Signed SK_G
Pay to PK_A : Hp(Coin27)

(SK_A, PK_A)

Coin27

Signed SK_G
CreateCoin CoinID(27)

How to solve this???

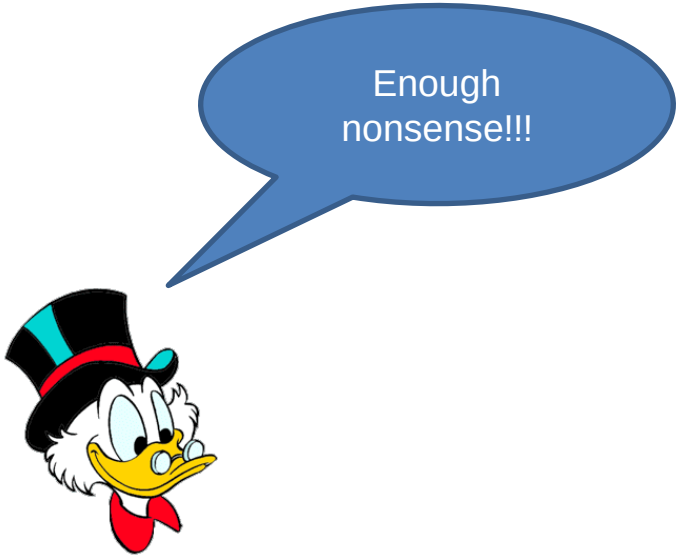
An interesting observation

Bitcoin uses the same model of transactions as Goofycoin:

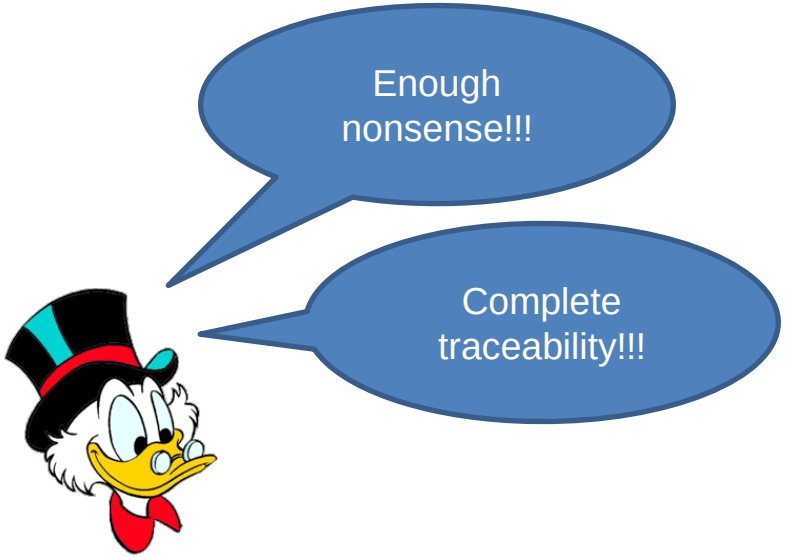
- List of hash pointers until the point the Bitcoin was created
- Obviously, we need to solve the double spending issue

Scroogecoin

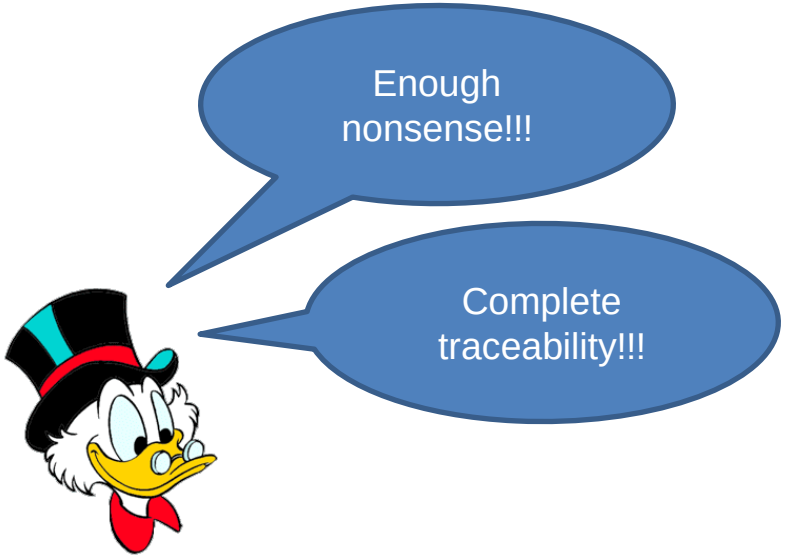
Scroogecoin



Scroogecoin



Scroogecoin



Ledger

Scrooge creates $70 for Alice

Scrooge creates $50 for Bob

Alice pays Bob $50

Alice pays Charlie $20

Bob pays Charlie $100

Charlie pays Alice $120

…

Scroogecoin

Protection against double spending

Scrooge publishes all the thansaction in a public Ledger:

- Once realized, the transaction remains in the Ledger forever (with an ID)
- To protect the Ledger against future changes, we use a blockchain

With Blockchain:

- One can verify if the money had been spent in the past

Scroogecoin

The general structure

Scroogecoin

The general structure

Scroogecoin

The general structure

transID: 73
transaction data

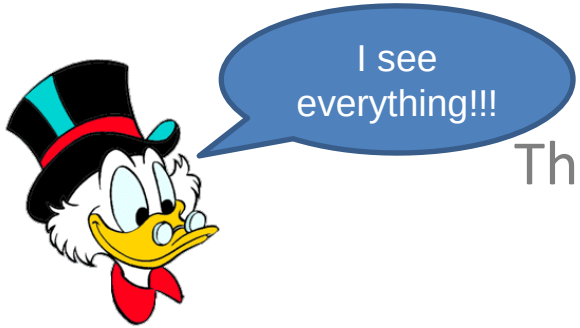
Scroogecoin

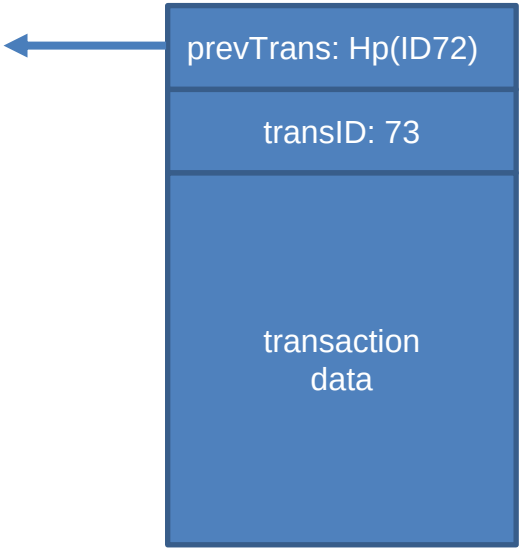
The general structure

prevTrans: Hp(ID72)
transID: 73
transaction data

Scroogecoin

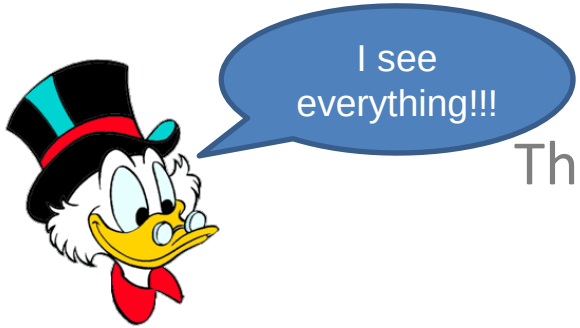
The general structure



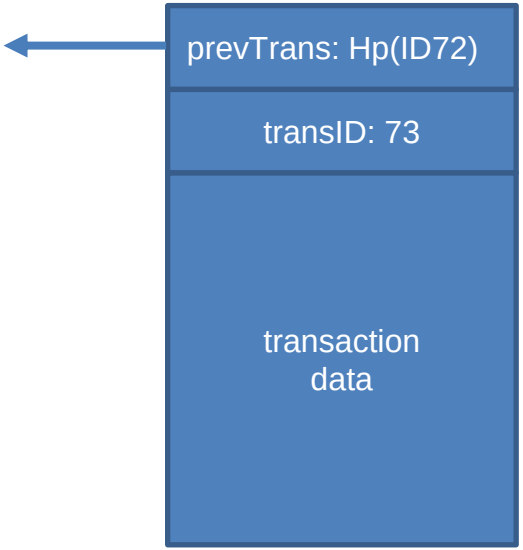


Scroogecoin

The general structure

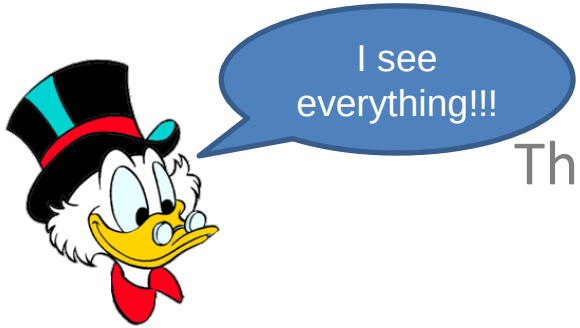


ID73

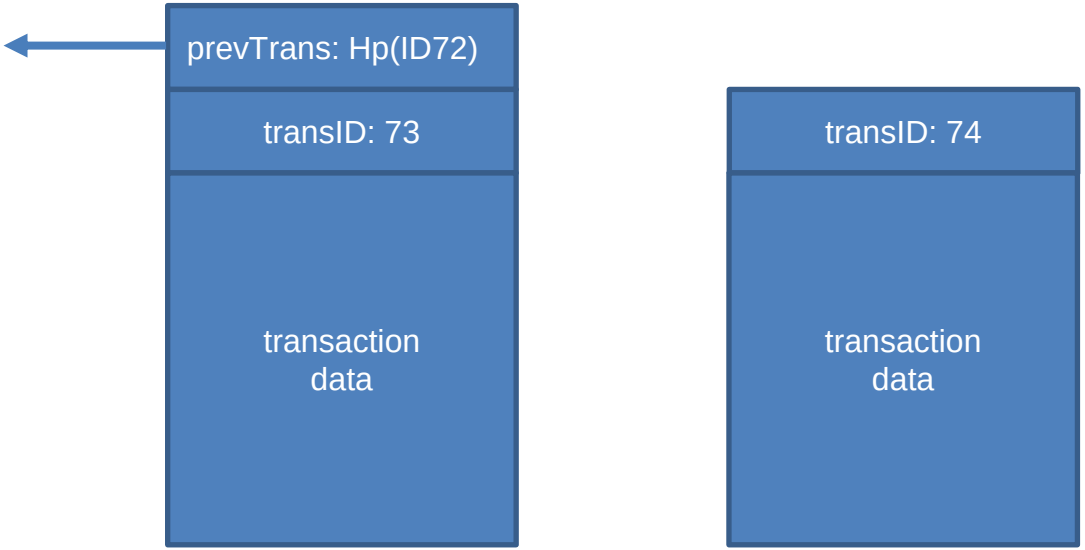


Scroogecoin

The general structure

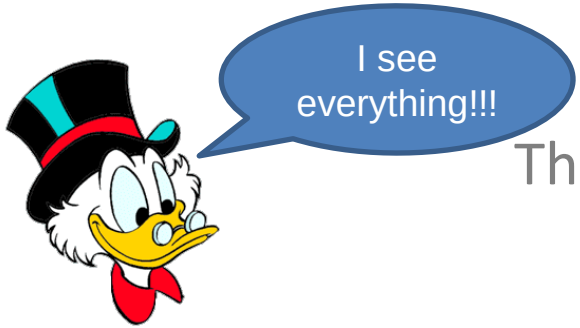


ID73

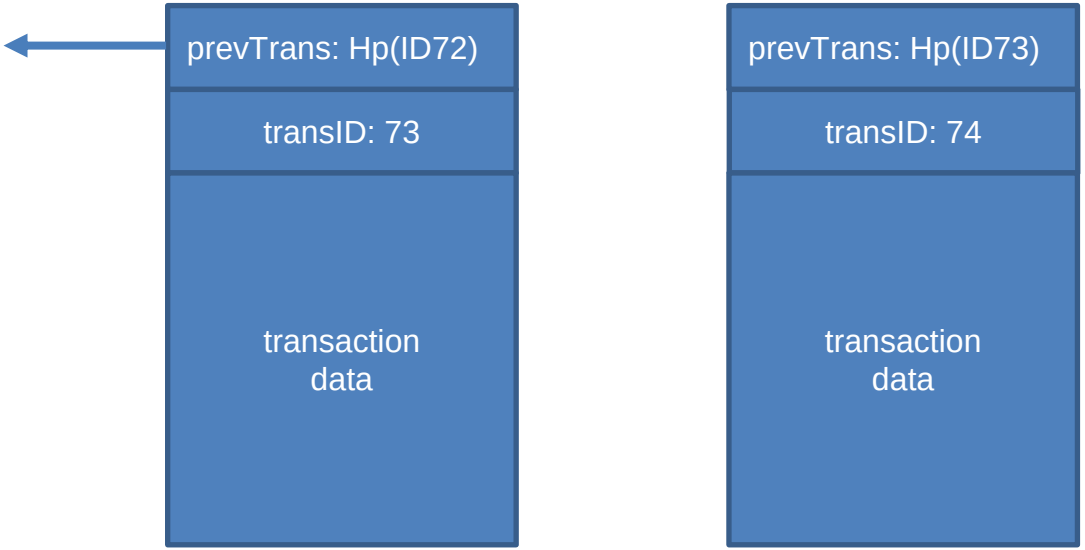


Scroogecoin

The general structure

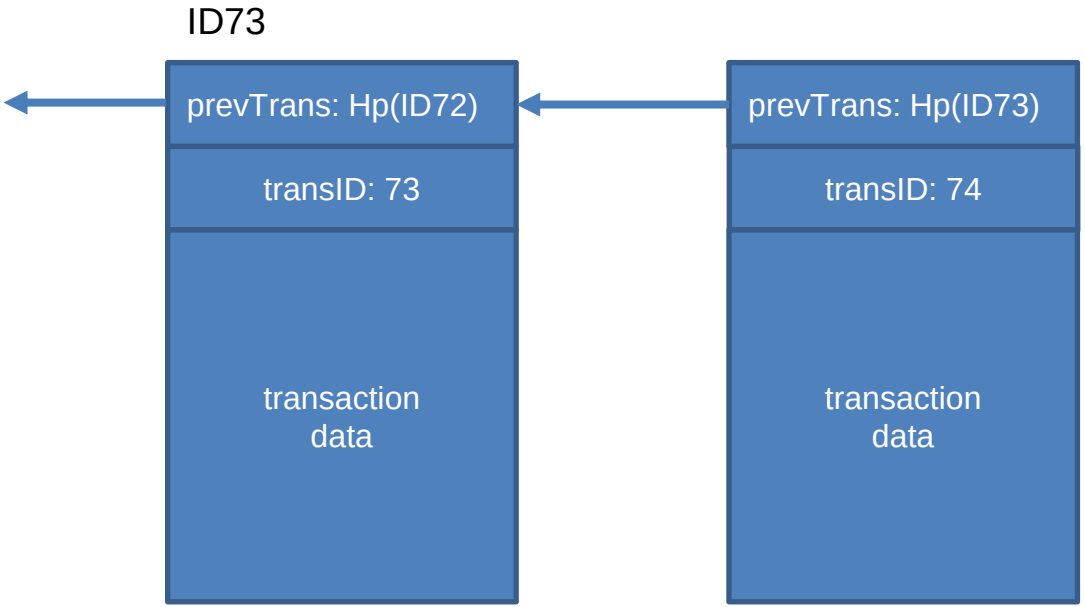


ID73



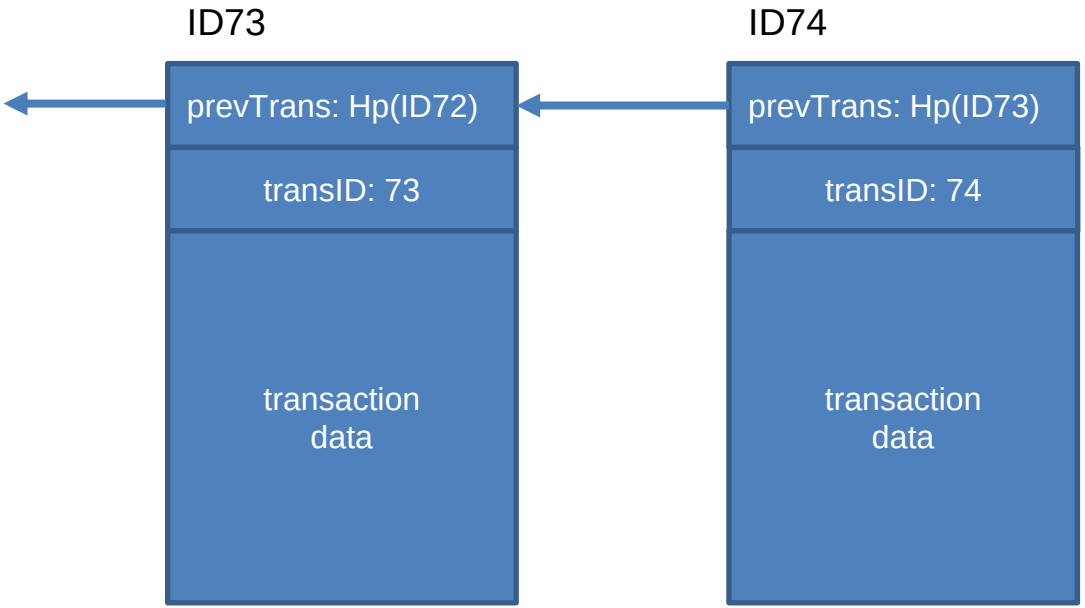
Scroogecoin

The general structure



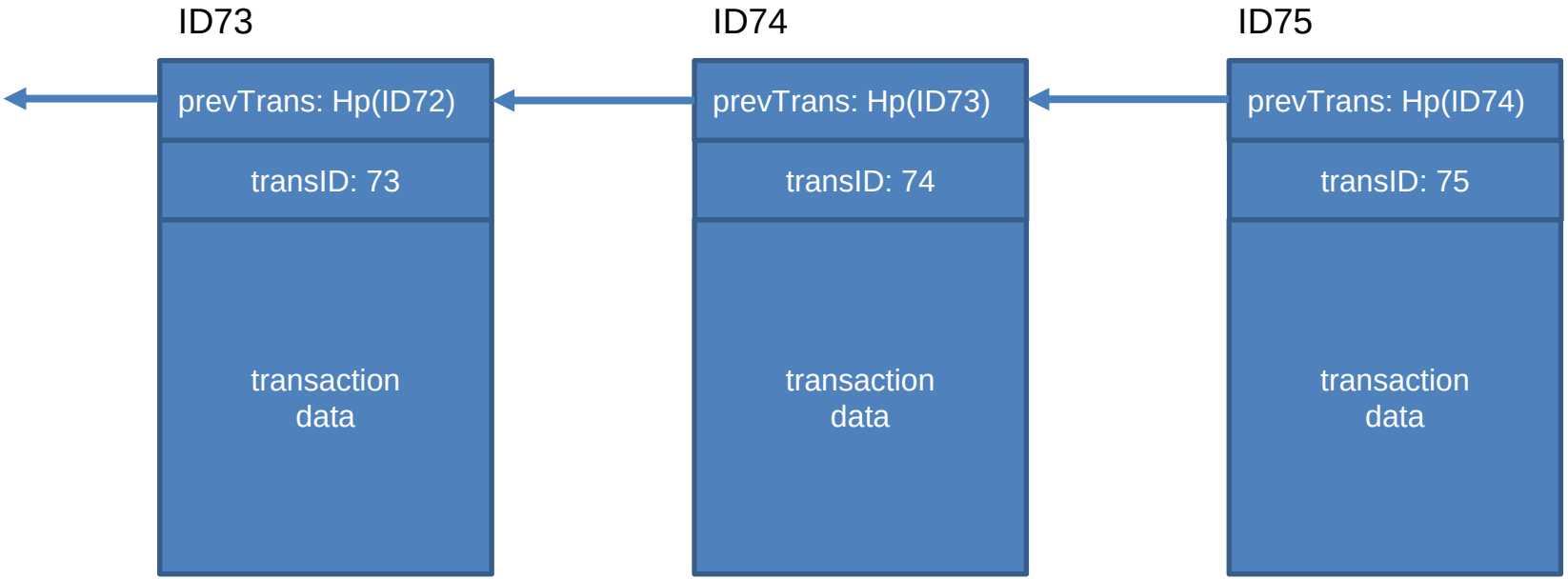
Scroogecoin

The general structure



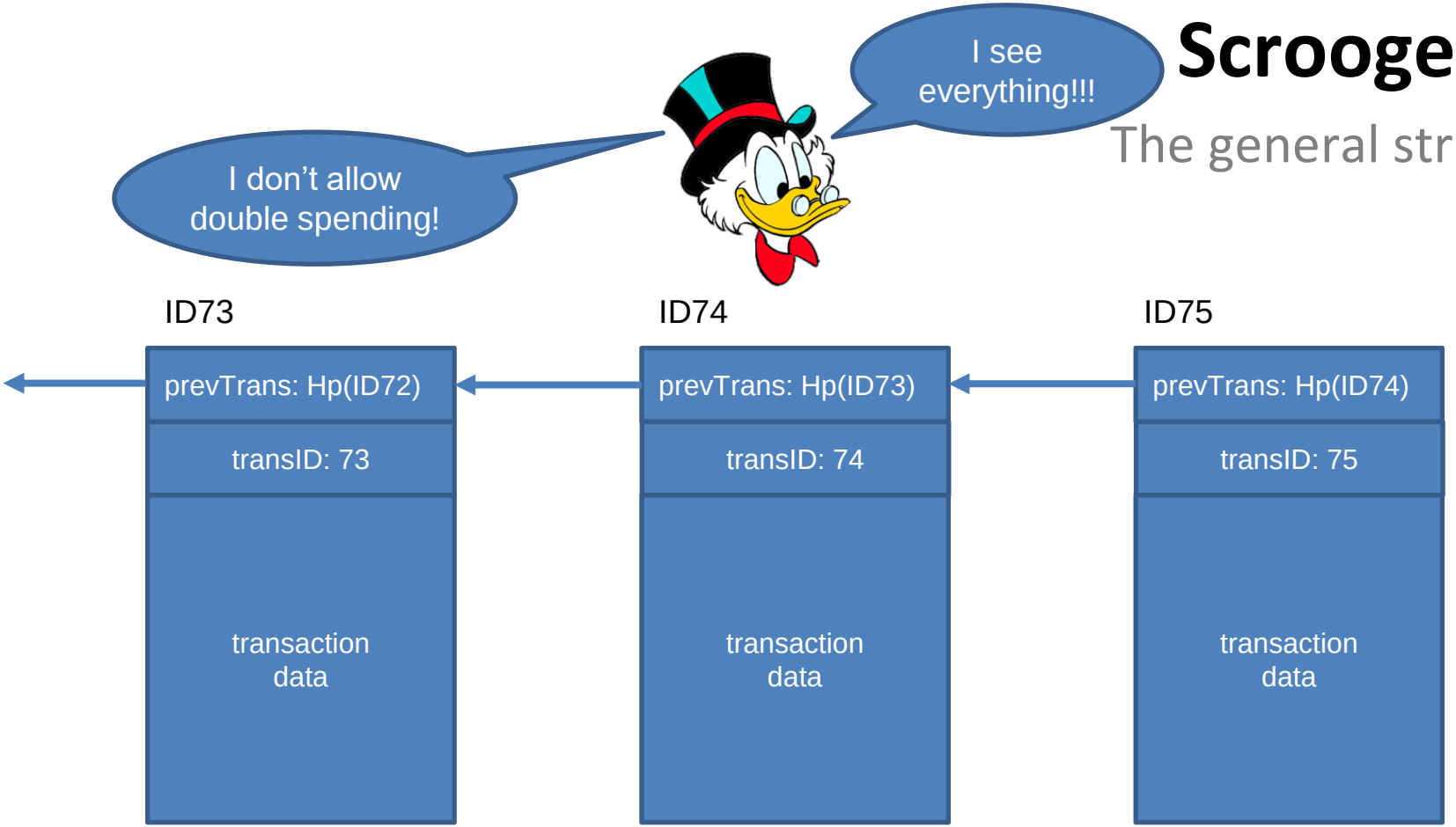
Scroogecoin

The general structure



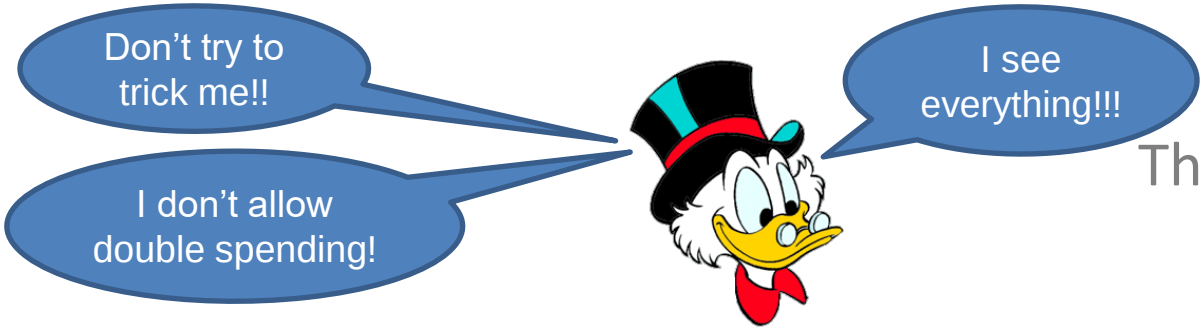
Scroogecoin

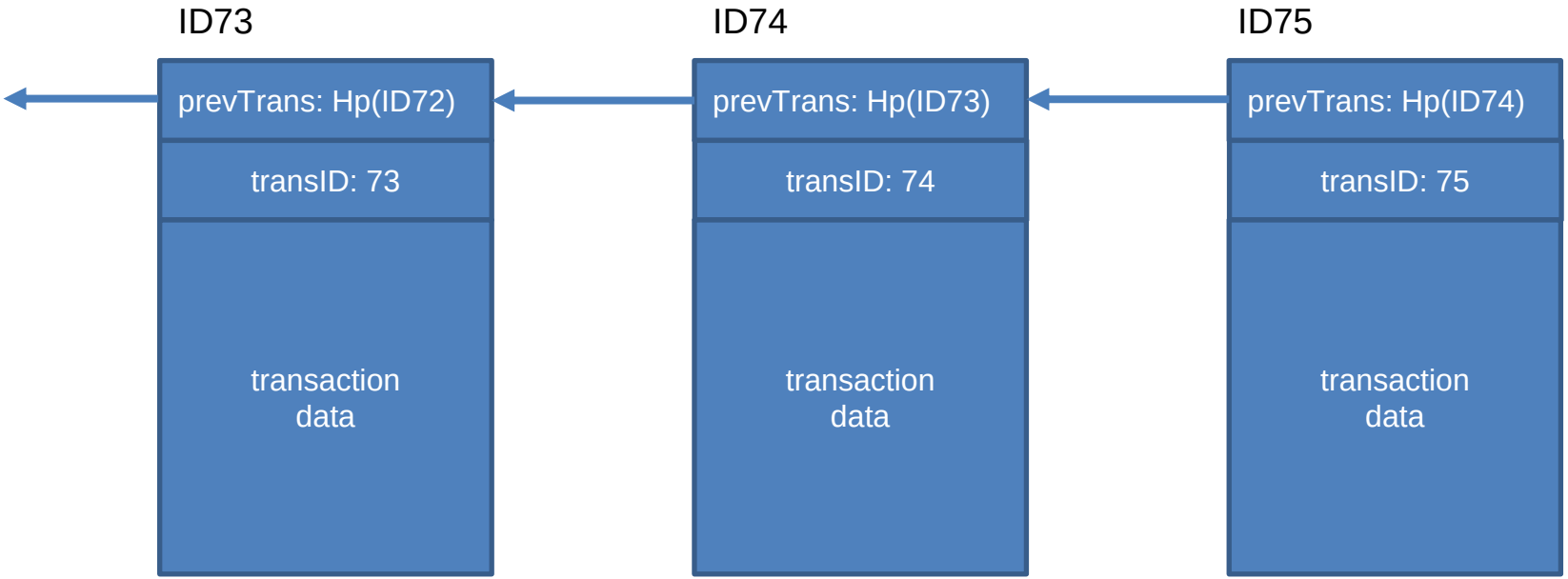
The general structure



Scroogecoin

The general structure





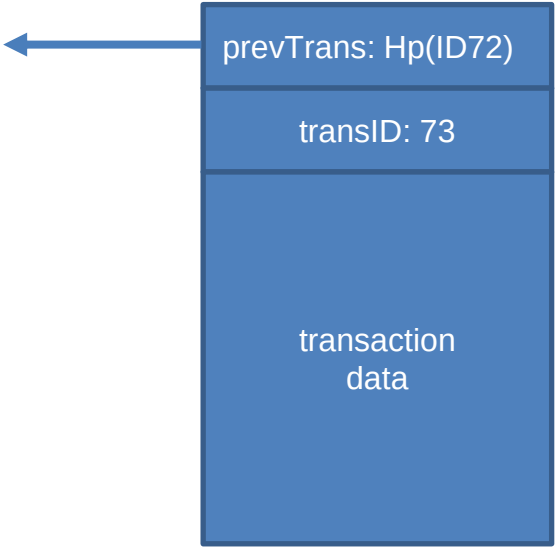
Important

Scrooge signs the blocks

Important

Scrooge signs the blocks

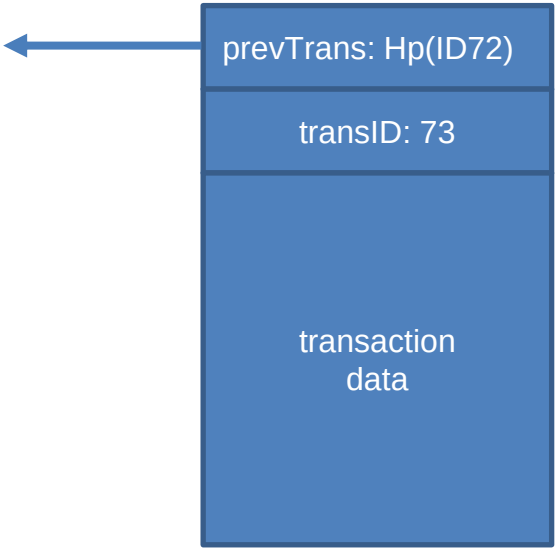
ID73



Important

Scrooge signs the blocks

ID73



Scroogecoin

Rules:

- A transaction is valid only if it appears in the Blockchain block signed by Scrooge
- Scrooge will not include transactions that attempt to double spend
- Anyone can verify that a transaction is valid using the Blockchain

Why do we need the Blockchain?

Is it enough if Scrooge just signs the transactions?

Scroogecoin

Rules:

- A transaction is valid only if it appears in the Blockchain block signed by Scrooge
- Scrooge will not include transactions that attempt to double spend
- Anyone can verify that a transaction is valid using the Blockchain

Why do we need the Blockchain?

Is it enough if Scrooge just signs the transactions?

- With the Blockchain, we can check if Scrooge ever tries to change a transaction in the future (if we have any block with his signature)

Scroogecoin

Almost identical to Goofycoin

Two types of transactions:

- CreateCoins (only Scrooge)
- PayCoins (anyone)
- Each transaction has an unique ID (its position in the Ledger)

Scrooge publishes the Ledger of the entire economy:

- Scrooge will not add a transaction that attempts to double spend
- We will use a Blockchain to make sure Scroog does not modify the transactions

Scroogecoin

CreateCoins transaction

Scroogecoin

CreateCoins transaction

transID: 73	type:CreateCoins

Scroogecoin

CreateCoins transaction

transID: 73	type:CreateCoins
coins created	

Scroogecoin

CreateCoins transaction

transID: 73		type:CreateCoins
coins created		
num	value	recipient

Scroogecoin

CreateCoins transaction

transID: 73		type:CreateCoins
coins created		
num	value	recipient
0	3.2	0xf4…

Scroogecoin

CreateCoins transaction

transID: 73		type:CreateCoins
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…

Scroogecoin

CreateCoins transaction

transID: 73		type:CreateCoins
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…

Scroogecoin

CreateCoins transaction

transID: 73		type:CreateCoins
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…
Signature by $\mathbf{SK_{Scrooge}}$		

Scroogecoin

CreateCoins transaction

transID: 73		type:CreateCoins
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…
Signature by $\mathbf{SK_{Scrooge}}$		

Creating multiple scroogecoins!!!

Scroogecoin

CreateCoins transaction

transID: 73		type:CreateCoins
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…
Signature by $\mathbf{SK_{Scrooge}}$		

For diferent recipients!!!

Scroogecoin

CreateCoins transaction

transID: 73		type:CreateCoins
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…
Signature by $\mathbf{SK_{Scrooge}}$		

With different values!!!

Scroogecoin

CreateCoins transaction

transID: 73		type:CreateCoins
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…
Signature by $\mathbf{SK_{Scrooge}}$		

Signing the transaction

Scroogecoin

CreateCoins transaction

transID: 73		type:CreateCoins
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…
Signature by $\mathbf{SK_{Scrooge}}$		

coinID 73(0)

Scroogecoin

CreateCoins transaction

transID: 73		type:CreateCoins
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…
Signature by $\mathbf{SK_{Scrooge}}$		

coinID 73(0)

coinID 73(1)

Scroogecoin

CreateCoins transaction

transID: 73		type:CreateCoins
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…
Signature by $\mathbf{SK_{Scrooge}}$		

coinID 73(0)

coinID 73(1)

coinID 73(2)

Scroogecoin

CreateCoins transaction

CreateCoins transaction:

- **Valid if signed by Scrooge**
- Scrooge can create any amount of Scroogecoins in this transaction
- Scrooge can create more than one Scroogecoin at the same time
- Every Scroogecoin created can have a different recipient (different from Scrooge)

Scroogecoin

PayCoins transaction

transID: 74	type: PayCoins

Scroogecoin

PayCoins transaction

transID: 74	type: PayCoins
coins consumed	

Scroogecoin

PayCoins transaction

transID: 74	type: PayCoins
coins consumed	
num	consumed coinID

Scroogecoin

PayCoins transaction

transID: 74	type: PayCoins
coins consumed	
num	consumed coinID
0	coinID 73(1)
1	coinID 73(2)

Scroogecoin

PayCoins transaction

transID: 74	type: PayCoins
coins consumed	
num	consumed coinID
0	coinID 73(1)
1	coinID 73(2)
coins created	

Scroogecoin

PayCoins transaction

transID: 74		type: PayCoins
coins consumed		
num	consumed coinID	
0	coinID 73(1)	
1	coinID 73(2)	
coins created		
num	value	recipient

Scroogecoin

PayCoins transaction

transID: 74		type: PayCoins
coins consumed		
num	consumed coinID	
0	coinID 73(1)	
1	coinID 73(2)	
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…

Scroogecoin

PayCoins transaction

transID: 74		type: PayCoins
coins consumed		
num	consumed coinID	
0	coinID 73(1)	
1	coinID 73(2)	
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…

Owned by Alice

Scroogecoin

PayCoins transaction

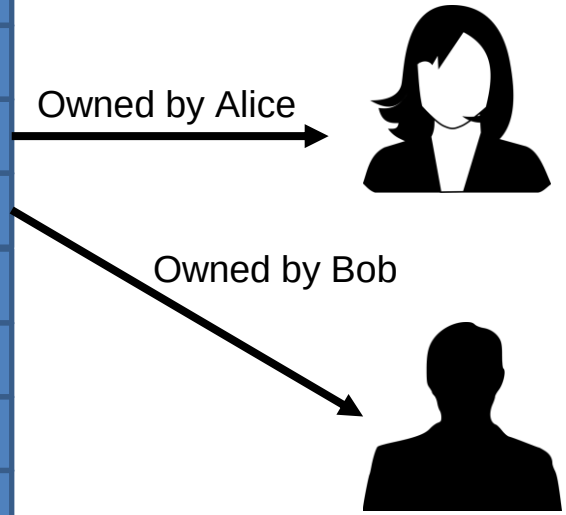
transID: 74		type: PayCoins
coins consumed		
num	consumed coinID	
0	coinID 73(1)	
1	coinID 73(2)	
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…
Signature by $\mathbf{SK_{Alice}}$		

Owned by Alice

Scroogecoin

PayCoins transaction

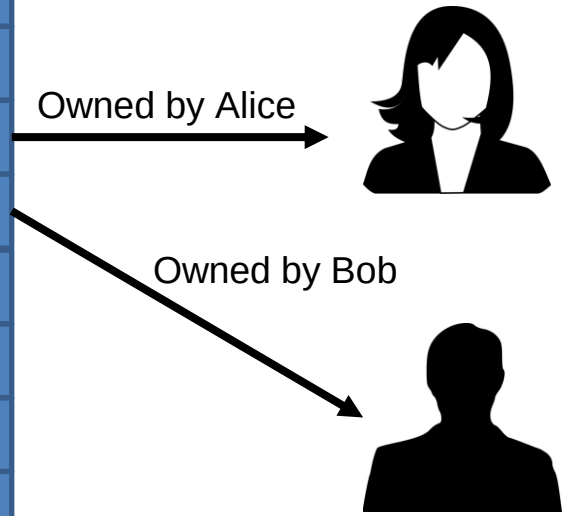
transID: 74		type: PayCoins
coins consumed		
num	consumed coinID	
0	coinID 73(1)	
1	coinID 73(2)	
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…
Signature by $\mathbf{SK_{Alice}}$		



Scroogecoin

PayCoins transaction

transID: 74		type: PayCoins
coins consumed		
num	consumed coinID	
0	coinID 73(1)	
1	coinID 73(2)	
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…
Signature by $\mathbf{SK_{Alice}}$		
Signature by $\mathbf{SK_{Bob}}$		



Scroogecoin

PayCoins transaction

PayCoins transaction:

- Consumes valid Scroogecoins (created previously in some transaction output)
- Consumes Scroogecoins that were not previously spend (doble spending)
- The total value of consumed coins is equal to the value of created coins
- All coin owners signed the *entire* transaction

If these conditions are met:

- Scrooge will accept the transaction PayCoins
- And publish it in the Blockchain (signing the hash pointer)

Scroogecoin

Transactions

The two types of transactions in Scroogecoin:

- Both create new Scroogecoins
- Consume zero or more Scroogecoins
- Only valid if published in the Blockchain signed by Scrooge

Scroogecoin

One Scroogecoin(of any value) is immutable:

- It is created precisely once
- It is consumed (spent, destroyed) only once
- After that it exists only as a proof of validity of other Scroogecoins

Scroogecoin

PayCoins transaction

transID: 74		type: PayCoins
coins consumed		
num	consumed coinID	
0	coinID 73(1)	
1	coinID 73(2)	
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…
Signature by $\mathbf{SK_{Alice}}$		
Signature by $\mathbf{SK_{Bob}}$		

Once spent; these coins are useless

Scroogecoin

PayCoins transaction

transID: 74		type: PayCoins
coins consumed		
num	consumed coinID	
0	coinID 73(1)	
1	coinID 73(2)	
coins created		
num	value	recipient
0	3.2	0xf4…
1	1.7	0xa1…
2	4.6	0x55…
Signature by $\mathbf{SK_{Alice}}$		
Signature by $\mathbf{SK_{Bob}}$		

Once spent; these coins are useless

This prevents double spending attack!!!

Scroogecoin

How to divide a Scroogecoin?

- Create a transaction that pays ***me*** two Scroogecoins of desired value

Scroogecoin

What does Scroogecoin achieve?

- Avoids double spending
- The transactions are immutable (once done they stay forever in the blockchain)
- It is not necessary to register in the system (digital signatures)
- No one can falsify a transaction (unless they have our private key)
- No one can steal our Scroogecoins (without our private key)
- No one can spend the money they don’t have

One problem with Scroogecoin

Scrooge:

- A completely centralized system

Scrooge has a lot of power:

- He can not falsify transactions (blockchain)
- He can create any amount of money for himself
- He can prohibit certain people he dislikes to participate in the system
- He can demand each transaction to pay some money to him
- He can get bored and stop publishing new transactions

What is Bitcoin?

Bitcoin = Scroogecoin without Scrooge

What is Bitcoin?

Our next task:

- Remove Scrooge
- **Decentralization**

What is Bitcoin?

Our next task:

- Remove Scrooge
- **Decentralization**

But before this: we need to understand Scroogecoin well

- The transaction mechanism of Bitcoin is almost identical to Scroogecoin

UTXO

Unspent transaction output

I can spend a Scroogecoin in PayCoins when:

- The Scroogecoin is valid
- The Scroogecoin was not yet spent

To verify that a Scroogecoin is valid:

- Scrooge maintains a pool of UTXOs
- When a new transaction arrives, Scrooge checks if the inputs are in the UTXO pool

UTXO

Unspent transaction output

In reality (Bitcoin):

- Each block contains more than one transaction
- Scrooge receives many transactions to include in the next block
- Makes it crucial to maintain a pool of UTXOs
- To check if the inputs are valid

If there are many transactions in a block:

- One can refer to the other (if correctly ordered) and spend its outputs
- Two transactions can try and spend the same Scroogecoin
- Which one will we include?

UTXO

UTXO pool

transID: 74		type: PayCoins
num	consumed coinID	
0	coinID 73(1)	
coins created		
num	value	recipient
0	3.2	0xf4…
Signature by $\mathbf{SK_{Alice}}$		

UTXO

UTXO pool

transID: 74		type: PayCoins
num	consumed coinID	
0	coinID 73(1)	
coins created		
num	value	recipient
0	3.2	0xf4…
Signature by $\mathbf{SK}_{\mathbf{Alice}}$		

Review the entire blockchain to check that the coin exists and was not yet spent

UTXO

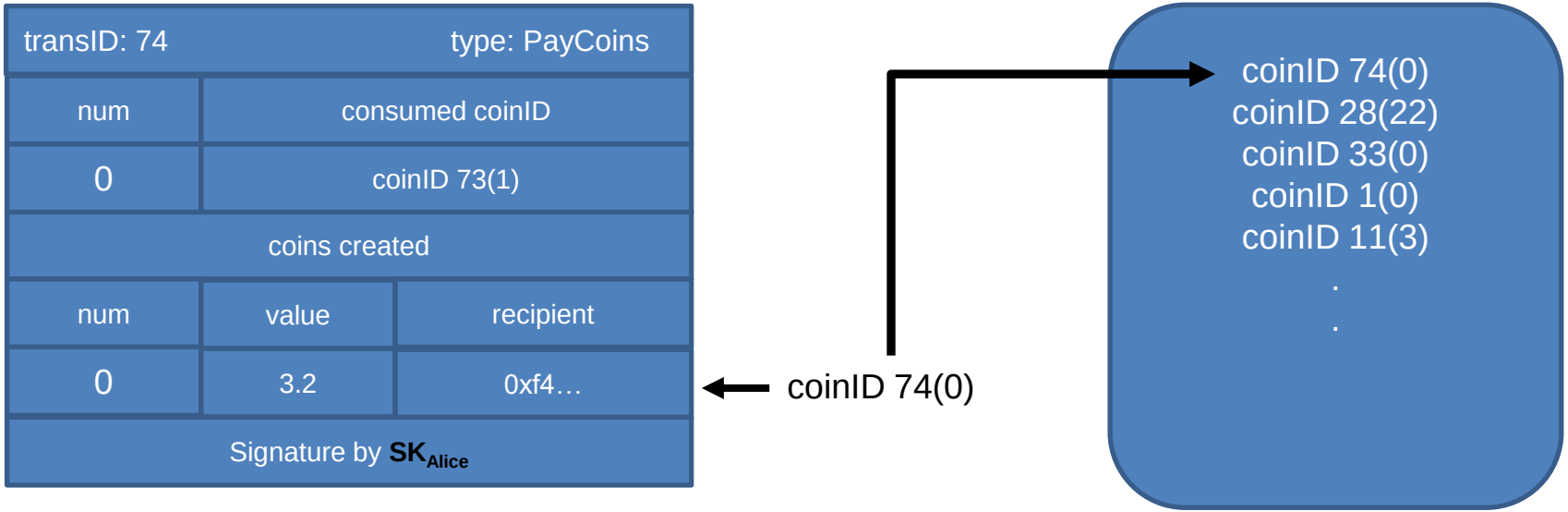
UTXO pool

transID: 74		type: PayCoins
num	consumed coinID	
0	coinID 73(1)	
coins created		
num	value	recipient
0	3.2	0xf4…
Signature by $\mathbf{SK}_{\mathbf{Alice}}$		

← coinID 74(0)

UTXO

UTXO pool



Practice time!!!

Implementing Scroogecoin

An UTXO pool:

coinID 74(0)
coinID 28(22)
coinID 33(0)
coinID 1(0)
coinID 11(3)
.
.
.

transID: 74		type: PayCoins
num	consumed coinID	
0	coinID 74(0)	
coins created		
num	value	recipient
0	3.2	0xf4…
Signature by SK_{Alice}		

Practice time!!!

Implementing Scroogecoin

An UTXO pool:

coinID 74(0)
coinID 28(22)
coinID 33(0)
coinID 1(0)
coinID 11(3)
.
.
.

transID: 74		type: PayCoins
num	consumed coinID	
0	coinID 74(0)	
coins created		
num	value	recipient
0	3.2	0xf4…
Signature by SK_{Alice}		

Check in the UTXO pool

Practice time!!!

Implementing Scroogecoin

An UTXO pool:

coinID 74(0)
coinID 28(22)
coinID 33(0)
coinID 1(0)
coinID 11(3)
.
.
.

transID: 74		type: PayCoins
num	consumed coinID	
0	coinID 74(0)	
coins created		
num	value	recipient
0	3.2	0xf4…
Signature by SK_{Alice}		

Check in the UTXO pool

If OK, check the rest

References

Narayanan et. al., chapter 1.5