Lesson 1

Bitcoin overview

Joseph Bonneau

This lecture

- Crypto background
  - hash functions
  - digital signatures
- Intro to cryptocurrencies
  - basic ledger-based cryptocurrency
  - sybils and 51% attacks

Lecture 1.1:

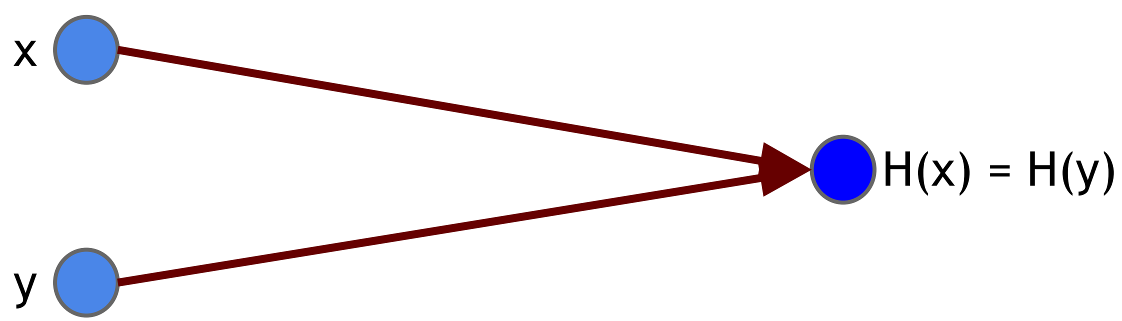
Cryptographic Hash Functions

- Hash function:
  - Deterministic function H: $\{0,1\}^* \rightarrow \{0,1\}^k$
  - Accepts ~any string as input
  - fixed-size output (we’ll use k=256 bits)
  - efficiently computable
- Security properties:
  - collision-free
  - one-way
  - puzzle-friendly (we’ll define this more later)

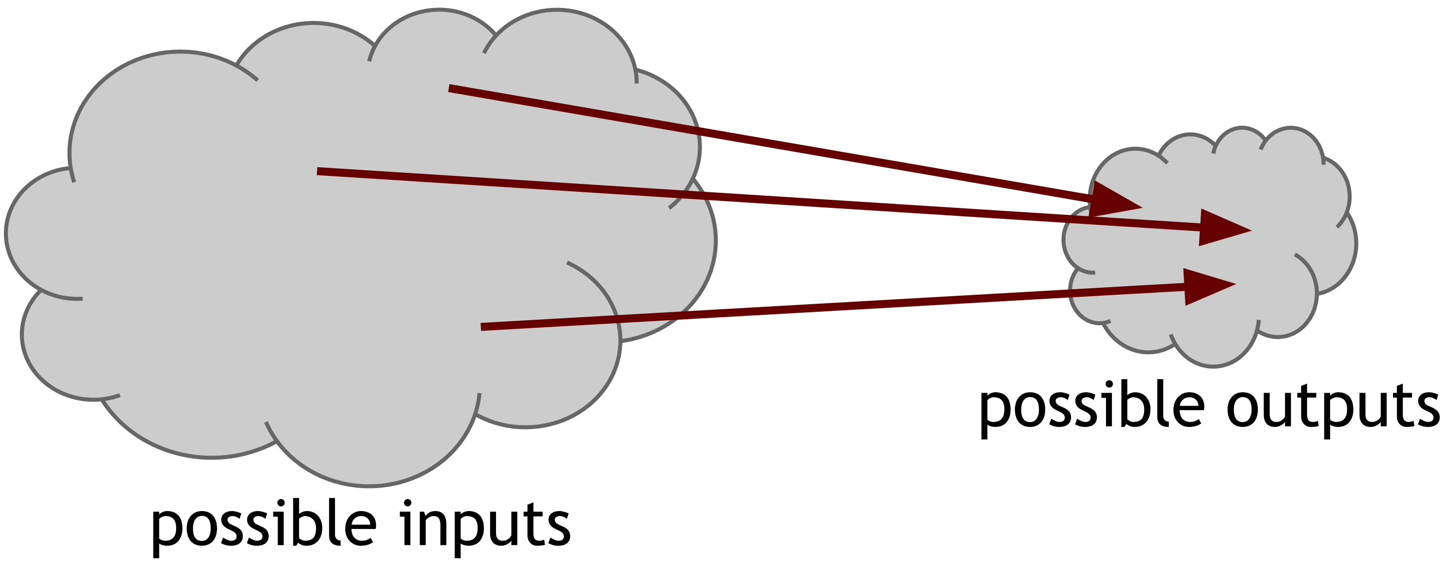
Hash property 1: Collision-free

Nobody can find x and y such that

x != y and H(x)=H(y)



Collisions exist ...



... but can anyone find them?

Birthday attack on any 256-bit hash **H**:

1. try 2^{130} randomly chosen inputs
2. >99.8% chance that two of them will collide

This works no matter what **H** is

… but it takes too long to matter

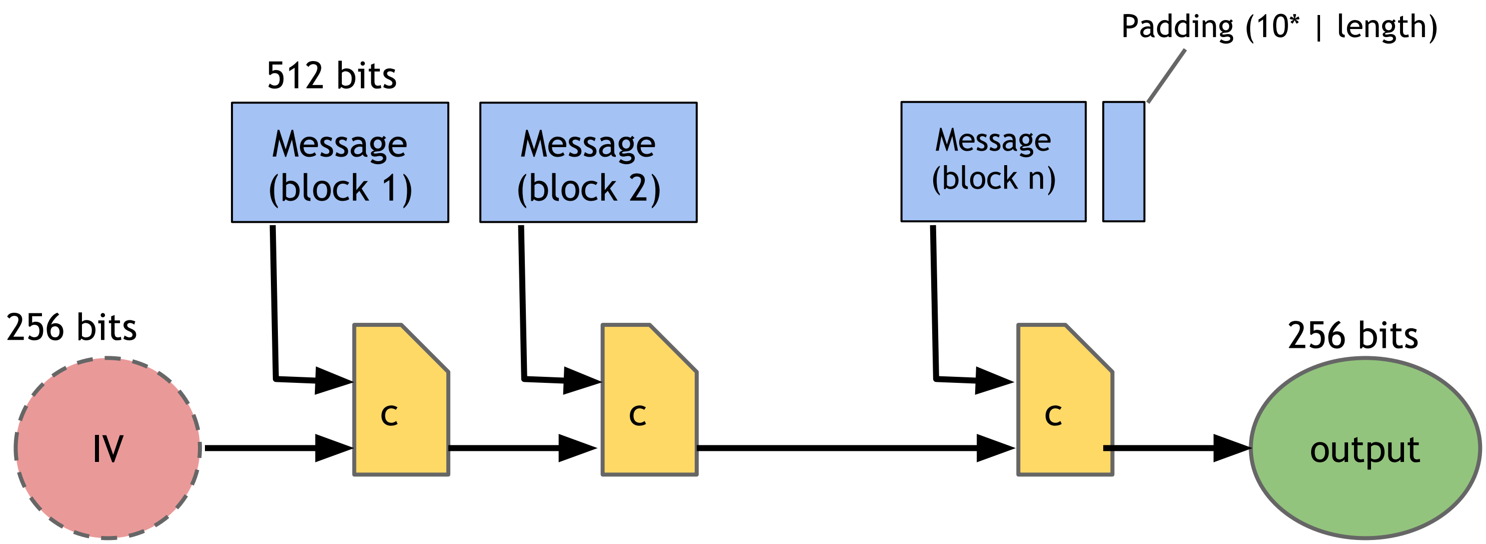
There are faster ways to find collisions for some H

- MD5 (collisions found)
- SHA-1 (near-collisions found)

Others are currently *collision-resistant*:

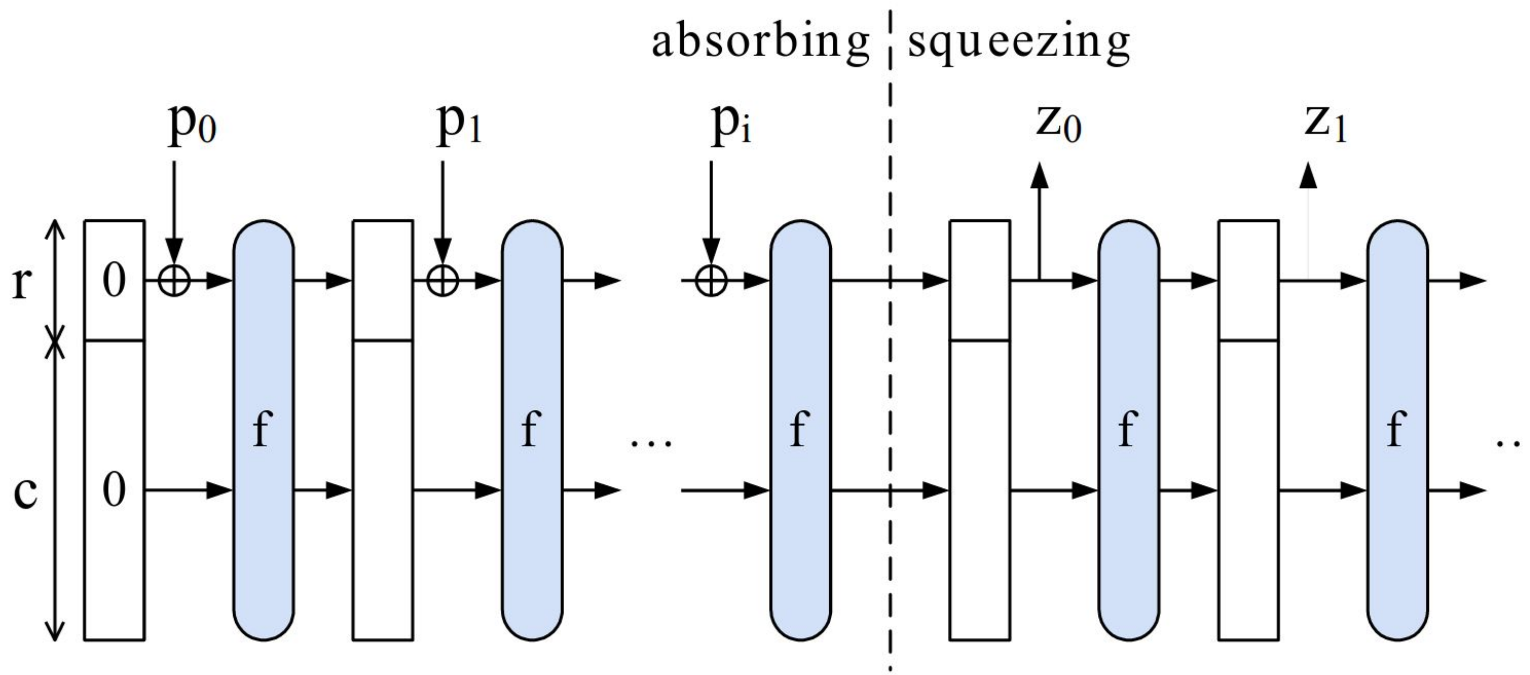
- SHA-256 (used heavily Bitcoin and others)
- SHA-3 (used in Ethereum)

Merkle-Dåmgard construction (SHA-256)



Theorem: If c is collision-free, then the hash is collision-resistant

Sponge construction (SHA-3)



Theorem: If *f* is a PRP, then the hash is collision-resistant

Application: Hash as message digest

If we know $H(x) = H(y)$ we assume that $x = y$.

Instead of storing x, store $H(x)$

Can fetch x from untrusted source and verify $H(x)$

Hash property #2: one-wayness

We want something like this:

“Given H(x), it is infeasible to find x”

But this breaks down if we know information about x:

H(“heads”)

H(“tails”)

easy to find x!

Hash property 2’: Hiding

If r is chosen from a probability distribution that has *high min-entropy*, then given H(*r* | *x*), it is infeasible to find *x*.

commit(*x*) := H(*r* | *x*)
verify(*com*, *r*, *x*) := H(*r* | *x*) == *com*

High min-entropy means that the distribution has no particular value with probability above some low limit

Lecture 1.2:

Hash pointers and authenticated data structures

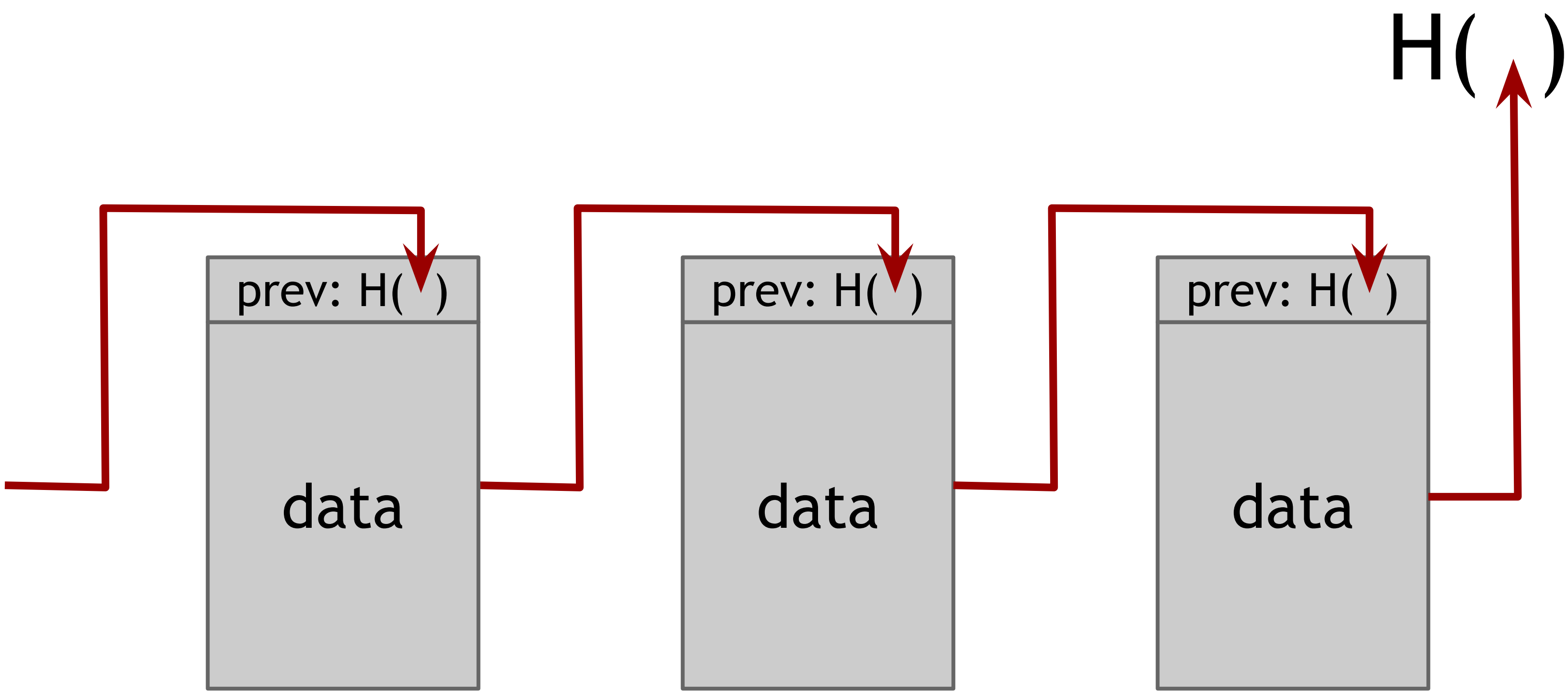
Key idea:

1. Take any pointer-based data structure
2. Replace pointers with cryptographic hashes

We now have an *authenticated data structure*

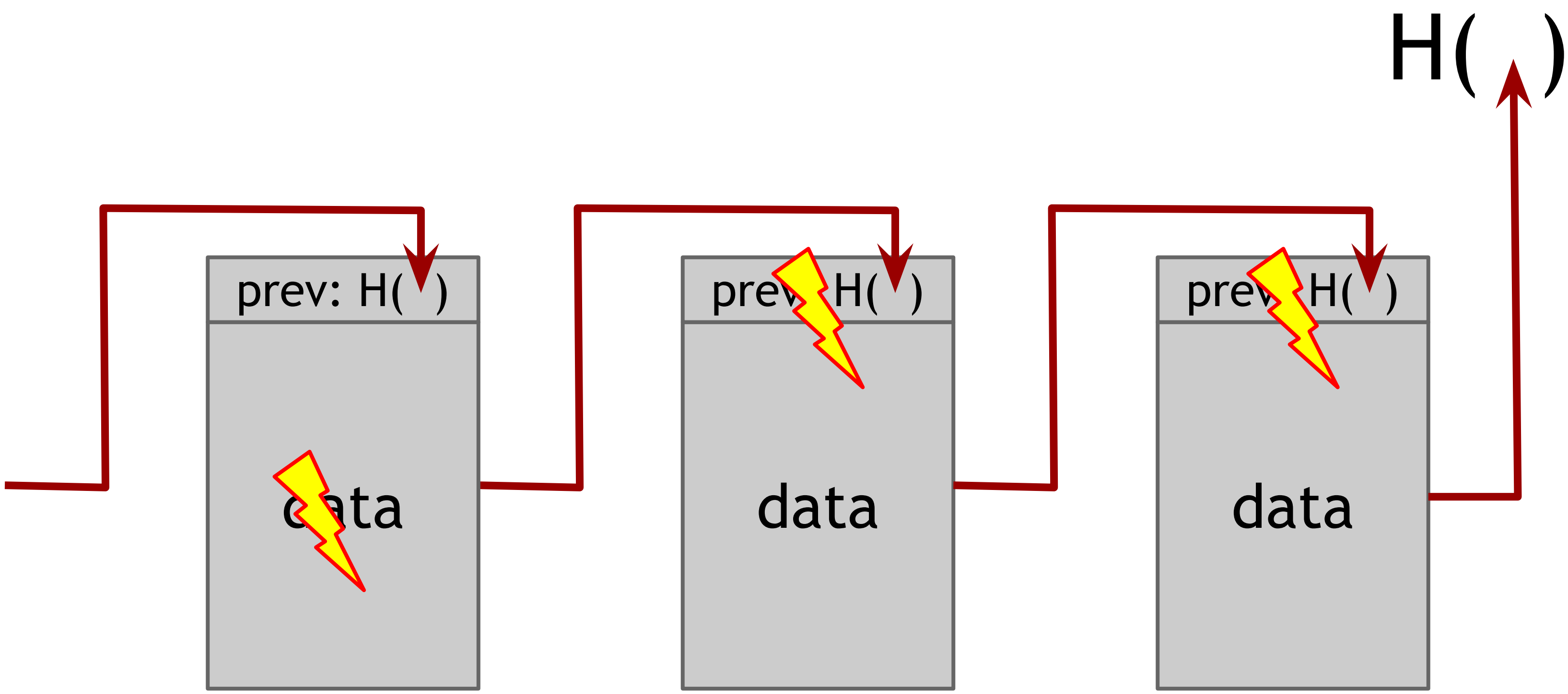
Hash pointers

Blockchain: Linked list with hash pointers



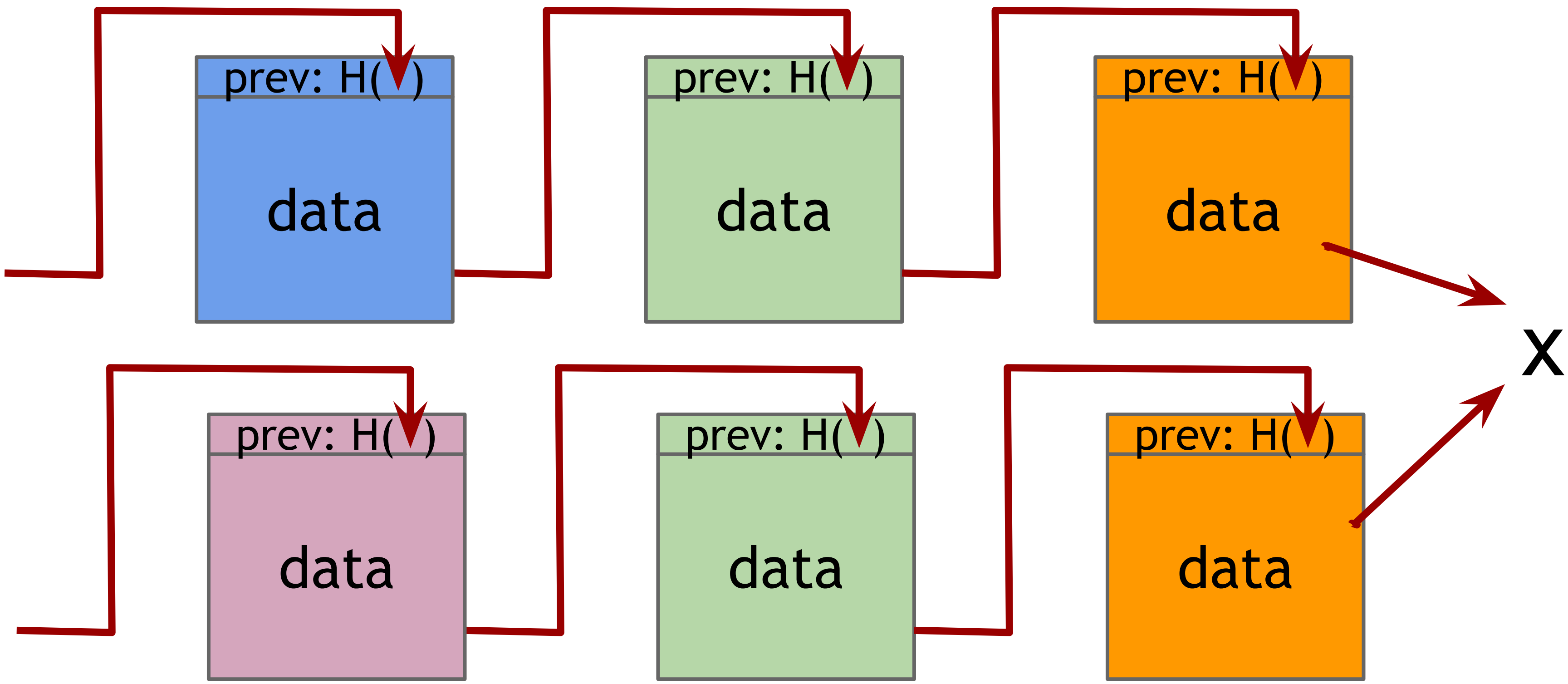
use case: tamper-evident log

Modifications to any block will propagate forever

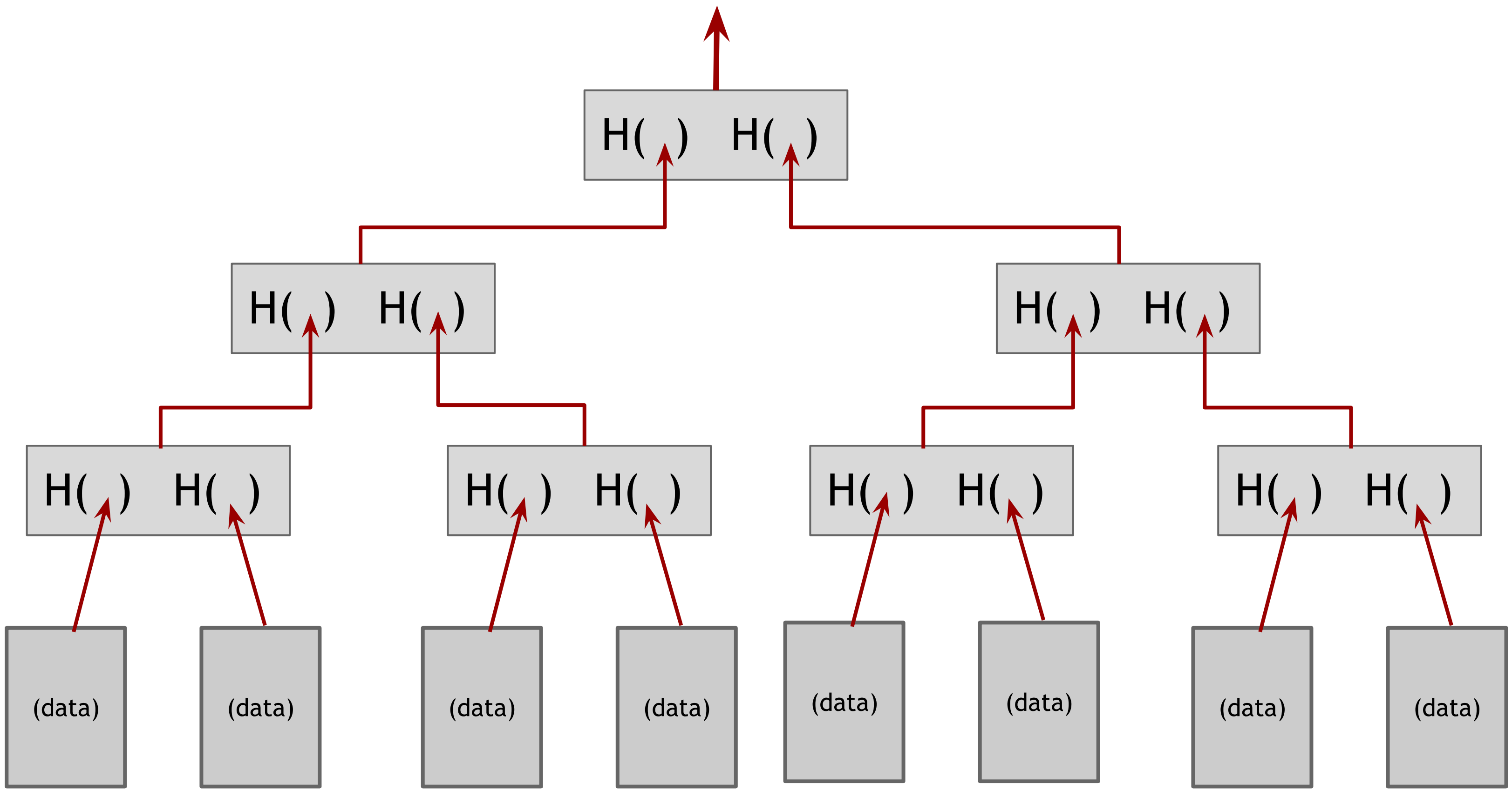


Theorem:

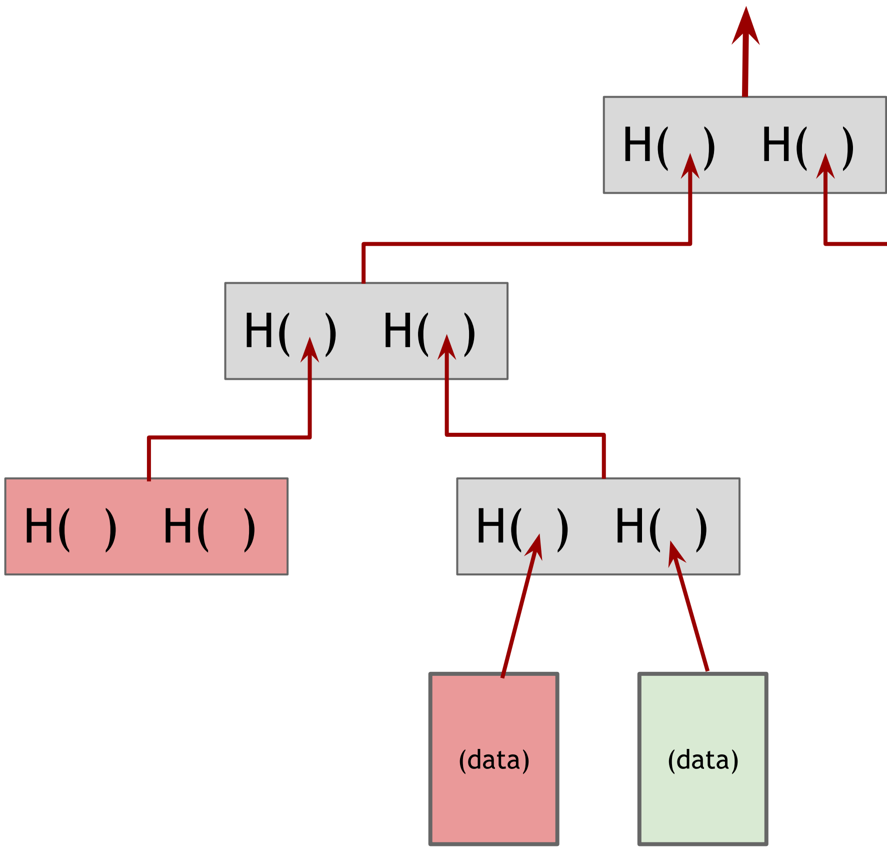
chains with same hash, different data → collision



Merkle tree: binary tree with hash pointers



proving membership in a Merkle tree



show O(log n) neighbors

Comparison

	Blockchain	Merkle tree
Abstraction	list	set
Commitment size	O(1)	O(1)
Append	O(1)	O(lg n)
Update	O(n)	O(lg n)
Membership proof	O(n)	O(lg n)

Can we do better?

Patricia tree/radix tree/trie

- Hash-pointer version of a radix trie
- Implements a $\{0,1\}^* \rightarrow \{0,1\}^*$ map
- O(lg n) proofs, storage

Used in Ethereum, not Bitcoin...

Generalizing the concept

can use hash pointers in any pointer-based DAG

General libraries exist (GPADS)

Lecture 1.3:

Digital Signatures

Digital signatures 101

(sk, pk) := genKey(keysize)

sk: secret signing key

pk: public verification key

sig := sign(sk, message)

isValid := verify(pk, message, sig)

can be randomized algorithms

Requirements for signatures

correctness

sk, pk = genKey(keysize) ⟶

verify(pk, message, sign(sk, message)) == true

unforgeability (EUF-CMA security)

adversary given pk

adaptively may query sign(m_i) oracle

cannot output a valid signature pair (σ, m’) for any new message m’

Bitcoin uses ECDSA

- Elliptic Curve Digital Signature Algorithm
- curve used is `secp256k1`
- set of points $(x,y) \in F_p \times F_p \mid y^2 = x^3 + 7$
- $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$
- Forms a group E, $|E| = q \approx p \approx 2^{256}$

	range	format	size (bits)
sk	$\mathbf{Z}_q$	random	256
pk	E	sk · G	512/257*
m	$\mathbf{Z}_q$	H(message)	256
sig	$\mathbf{Z}_q \times \mathbf{Z}_q$	(r, s)	512

The airing of ECDSA grievances

Problem	Remedies
re-using randomness leaks sk	use PRF(m) as randomness (or use BLS)
malleable	normalization (or use BLS)
not threshold friendly	complex SMPC, EC-Schnorr, BLS, RSA
not quantum safe	Hash-based sigs, lattice-based crypto

Useful convention public key == identity

- Anybody can get an identity with genKey
    - Collisions statistically negligible
- To “speak” as pk, sign using sk
- Keys are *pseudonyms*

Addresses in Bitcoin

- Address = H(pk)   (usually)

- Hashed, converted to base56:

1BvBMSEYstWetqTFn5Au4m4GFg7xJaNVN2

1JBonneauruSSoYm6rH7XFZc6Hcy98zRZz

Lecture 1.4:

Simple cryptocurrencies

Obvious approach

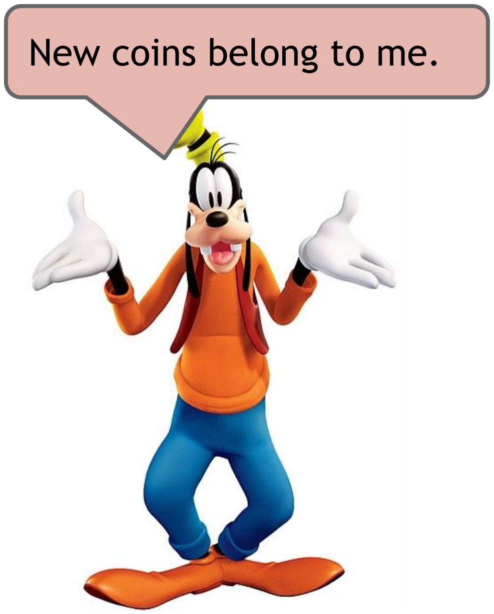
1. Use public keys as addresses
2. Sign to authorize transfer to new address

New coins created [somehow]

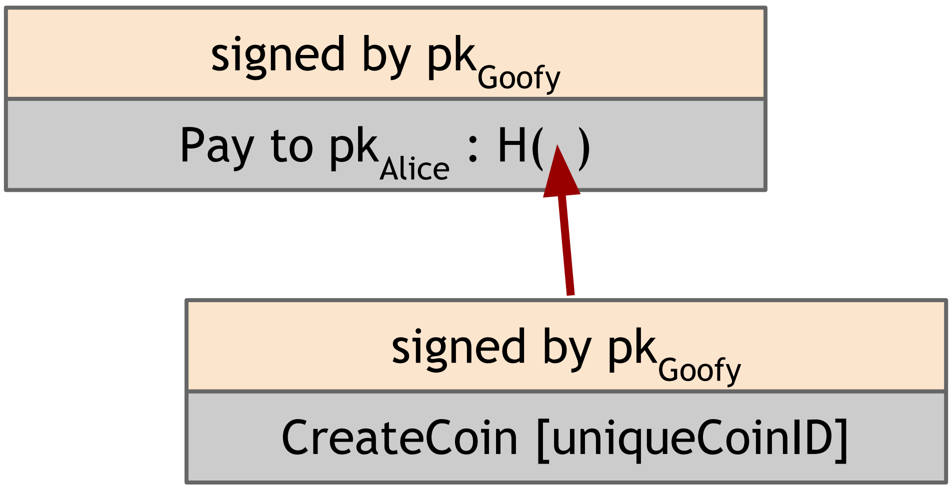
GoofyCoin

Goofy can create new coins

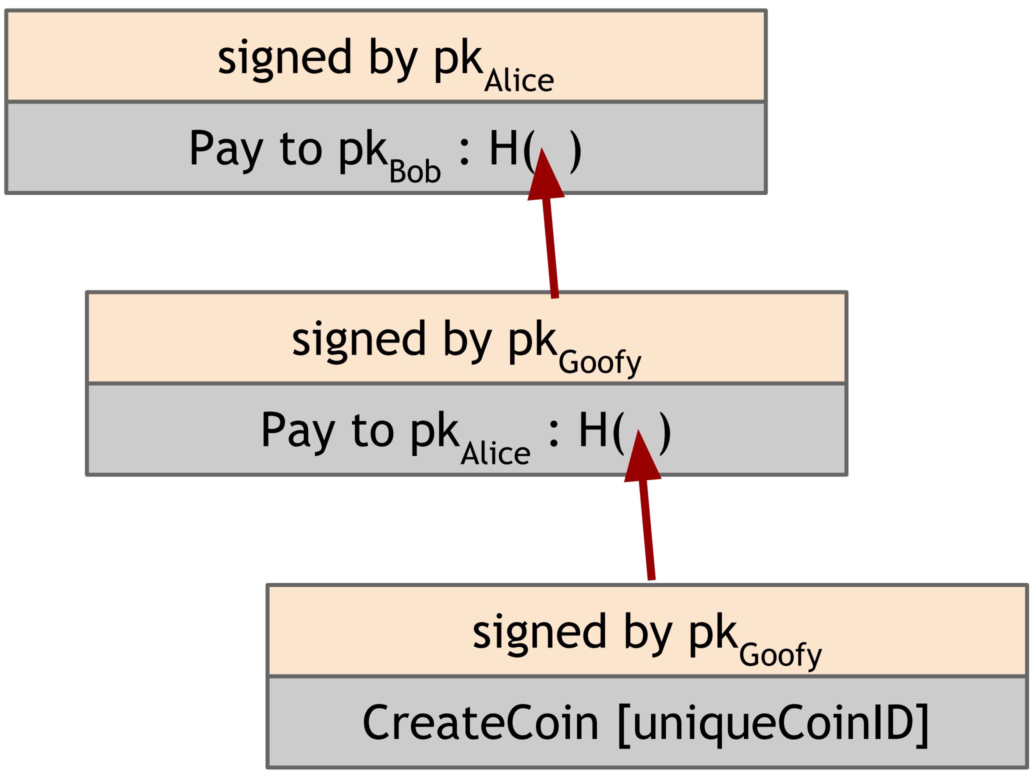
signed by pk_{Goofy}
CreateCoin [uniqueCoinID]

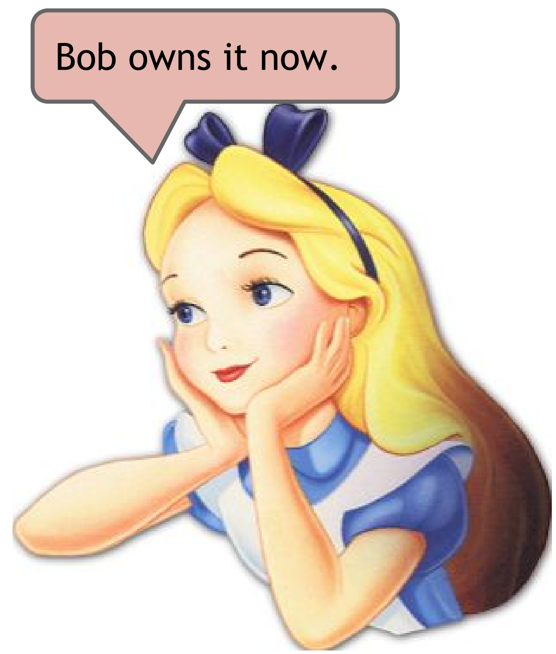


A coin’s owner can spend it.

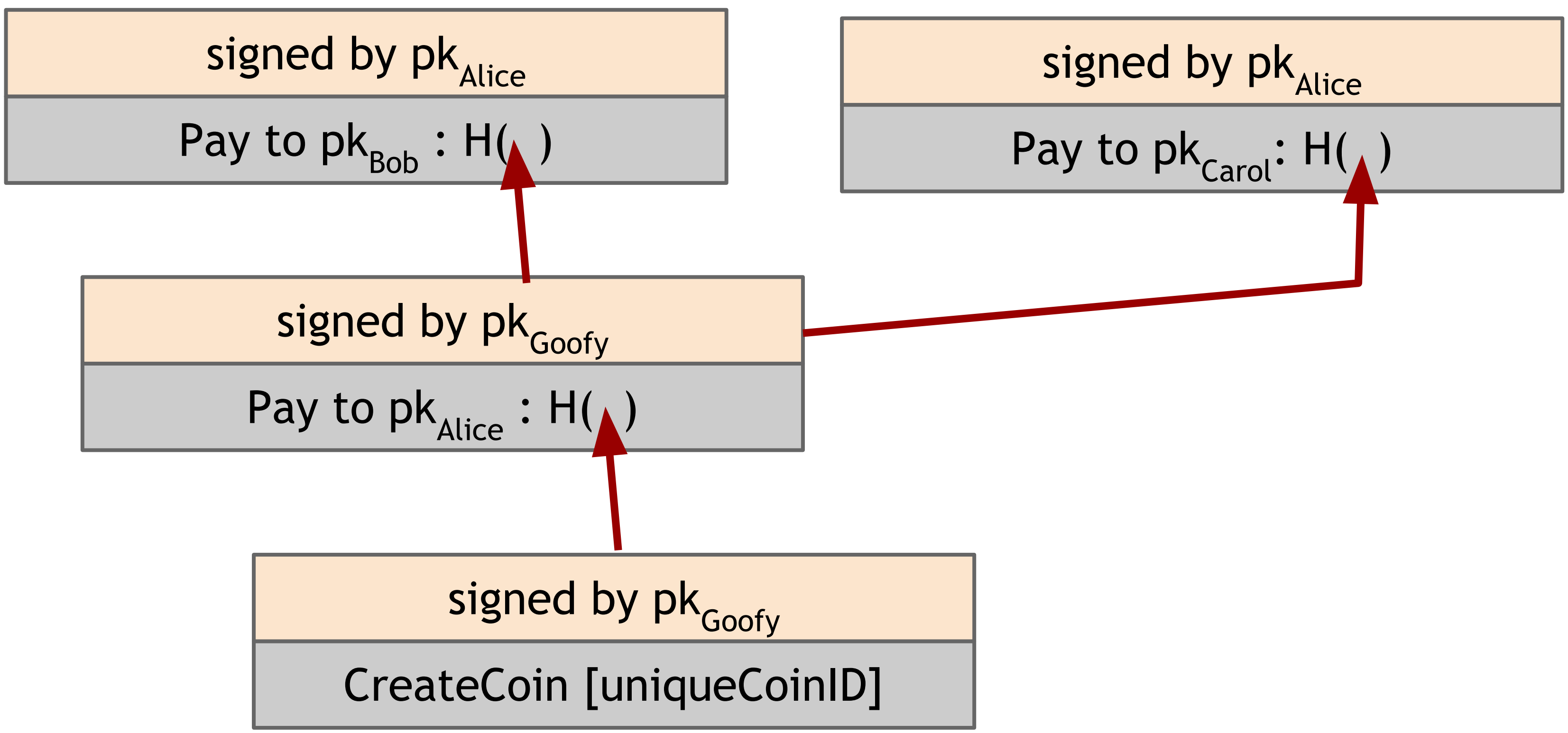


The recipient can pass on the coin again.

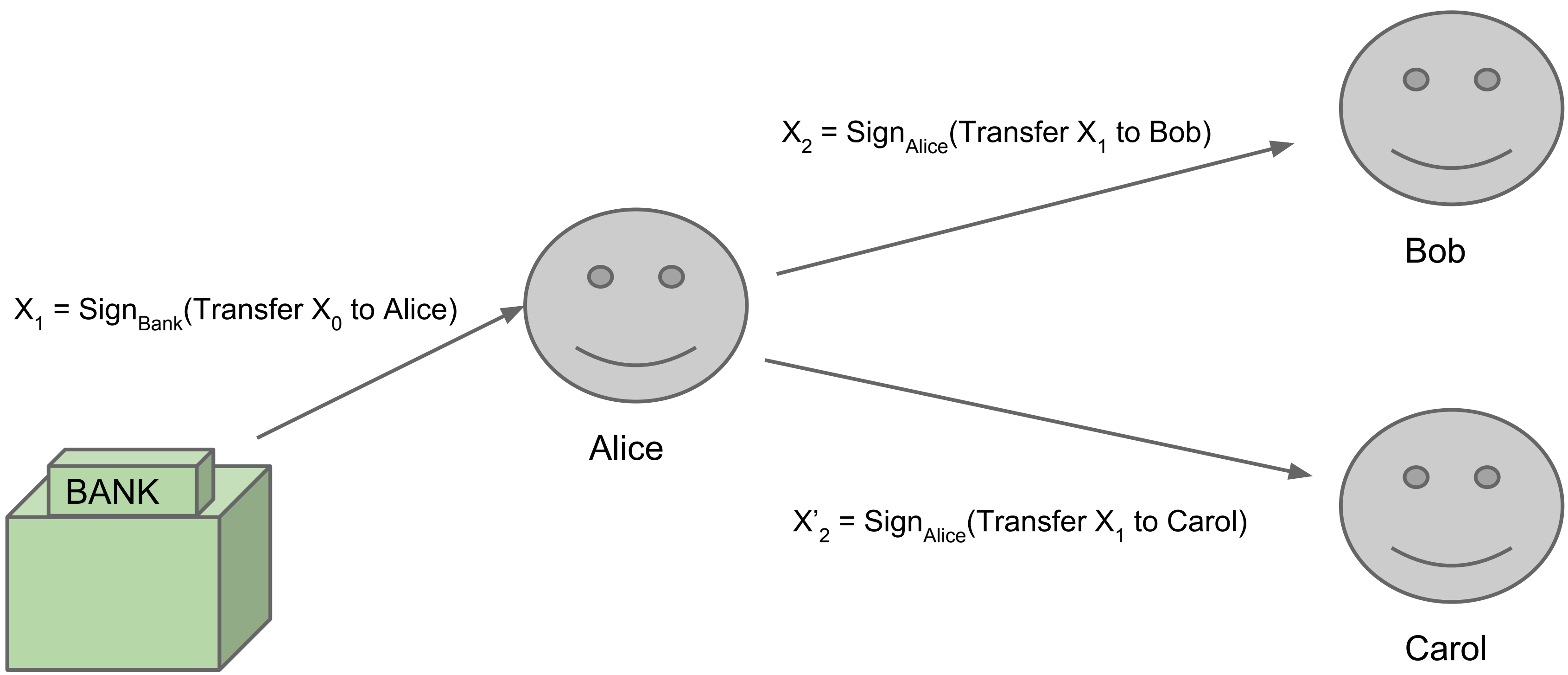




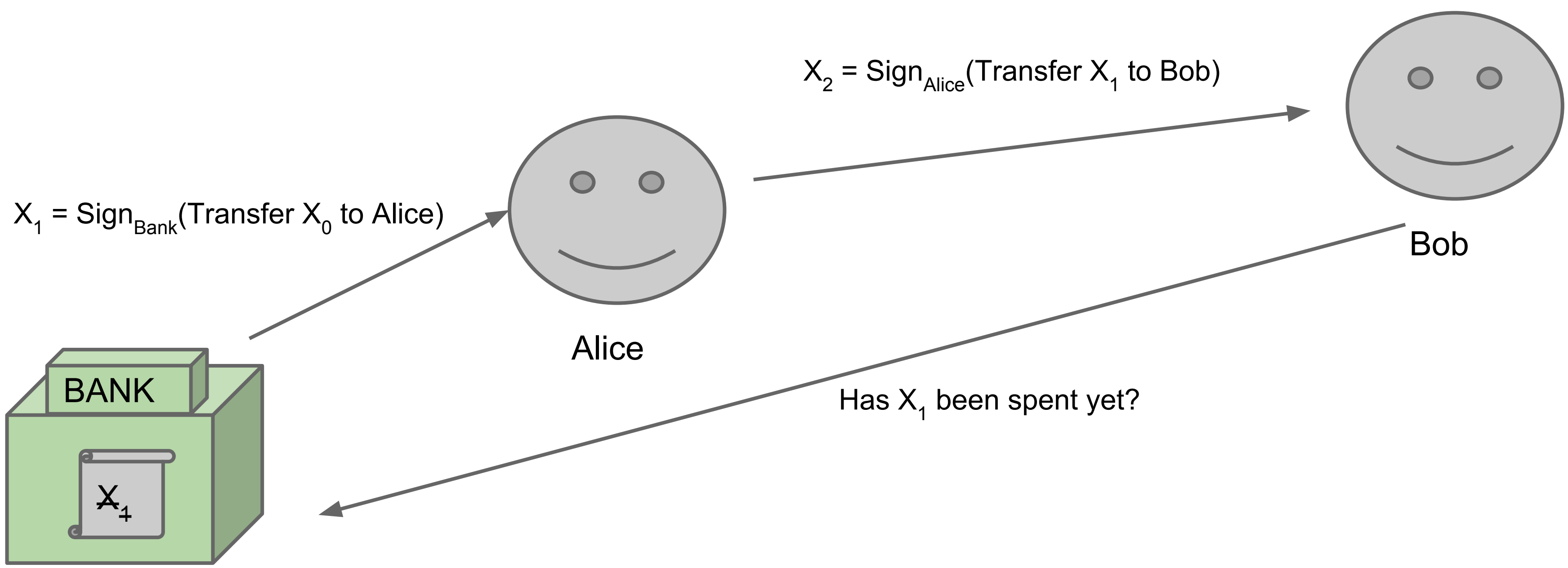
double-spending attack



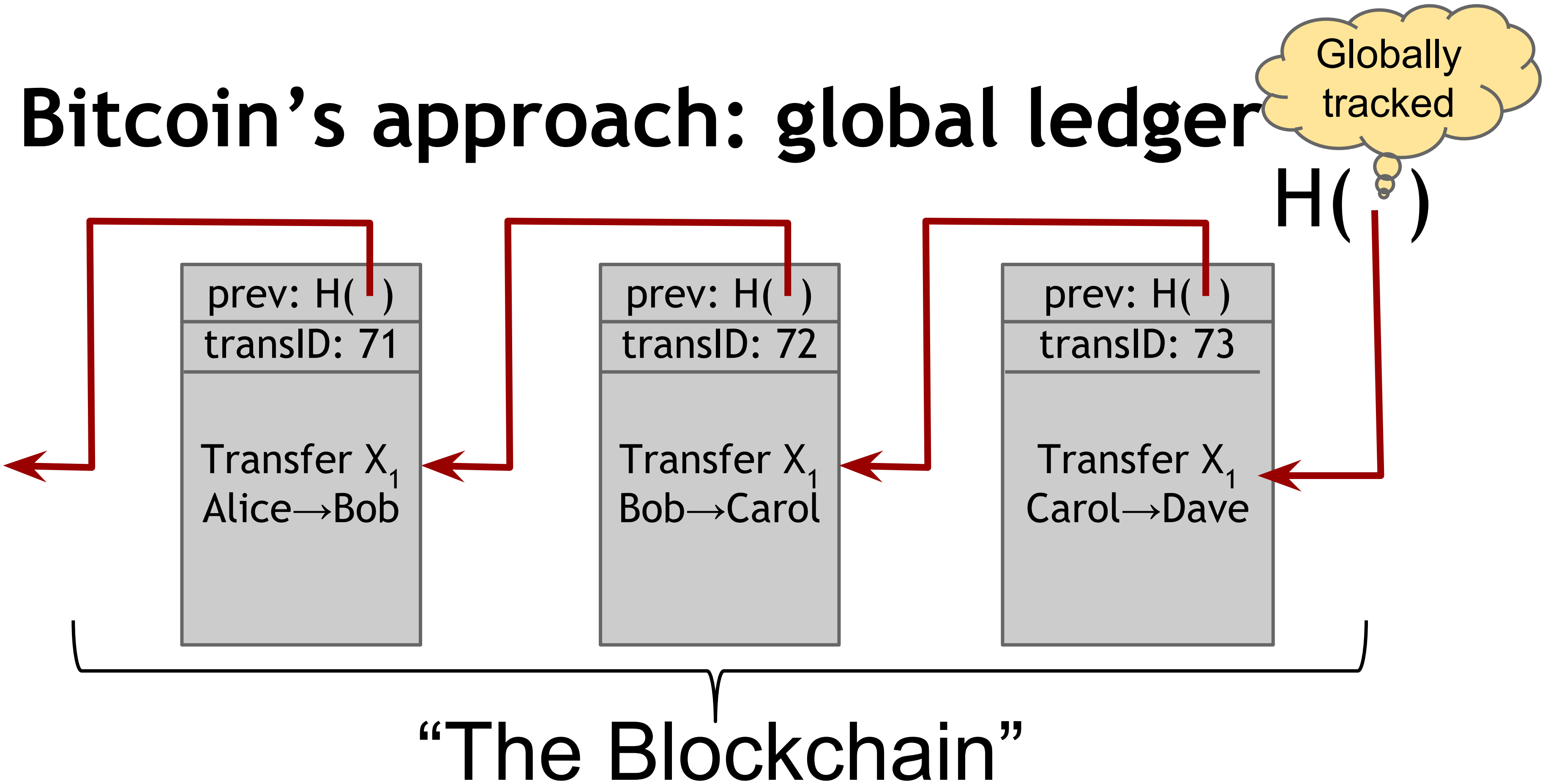
Double-spends must be prevented



Traditional approach: talk to the issuer



Bitcoin's approach: global ledger



Lecture 1.5:

Transaction semantics

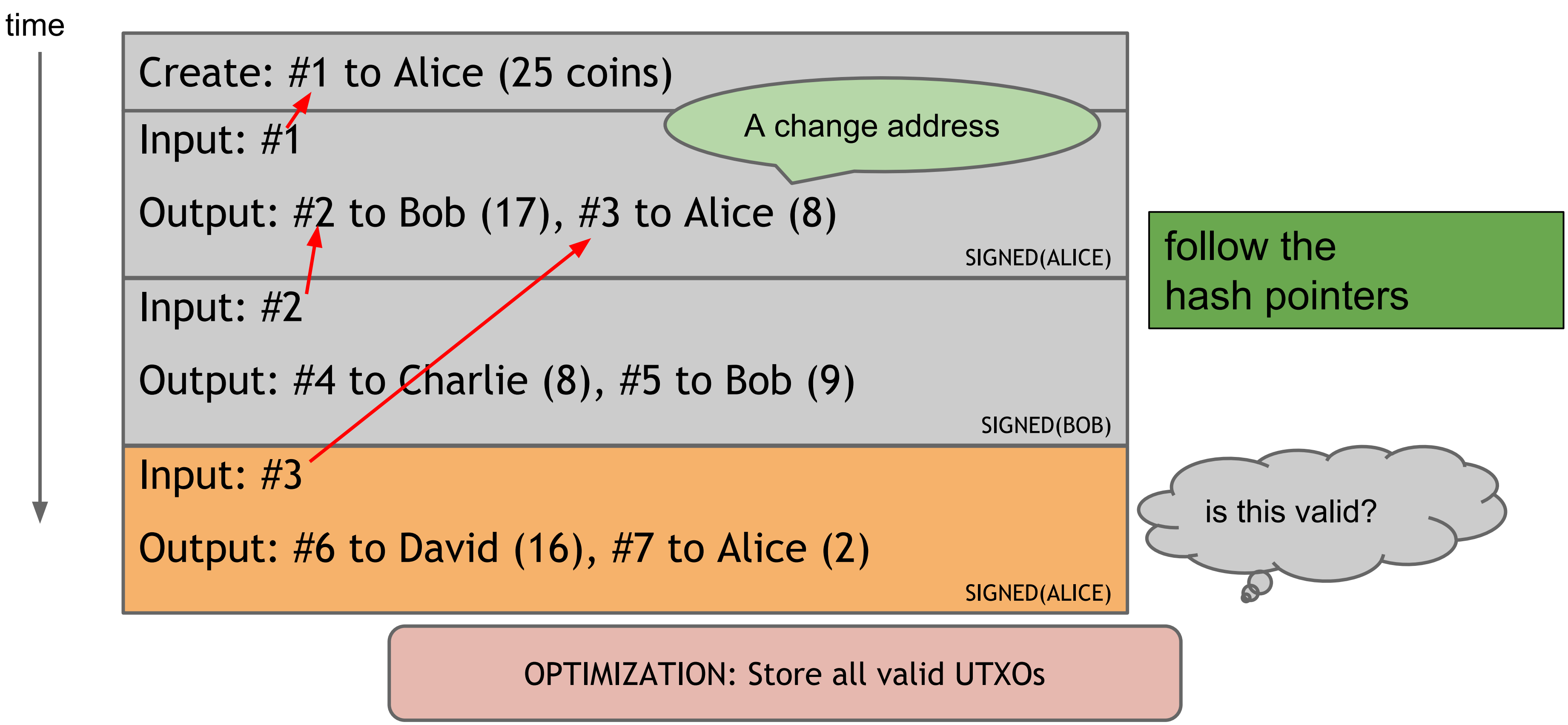
Bitcoins are *immutable*

“Coins” aren’t transferred, subdivided, or combined

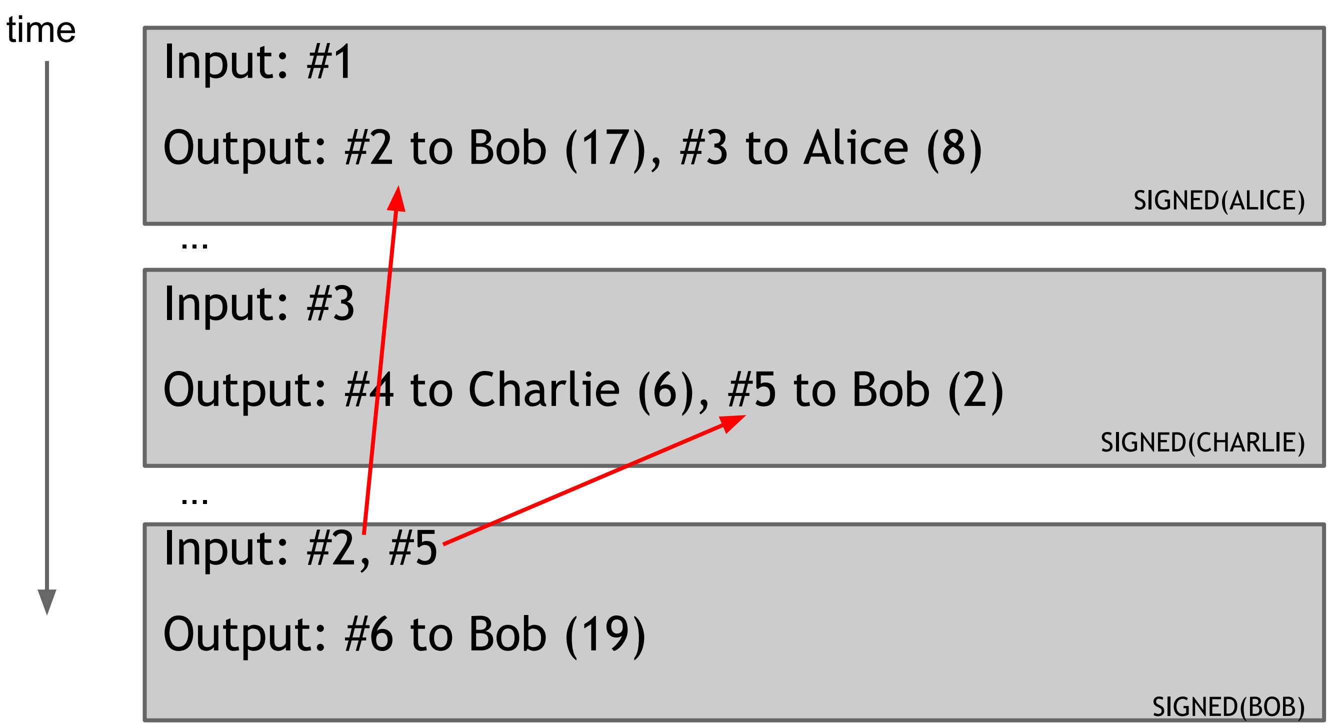
Transactions destroy old “coins”, create new ones

- easily replicate division via *change addresses*

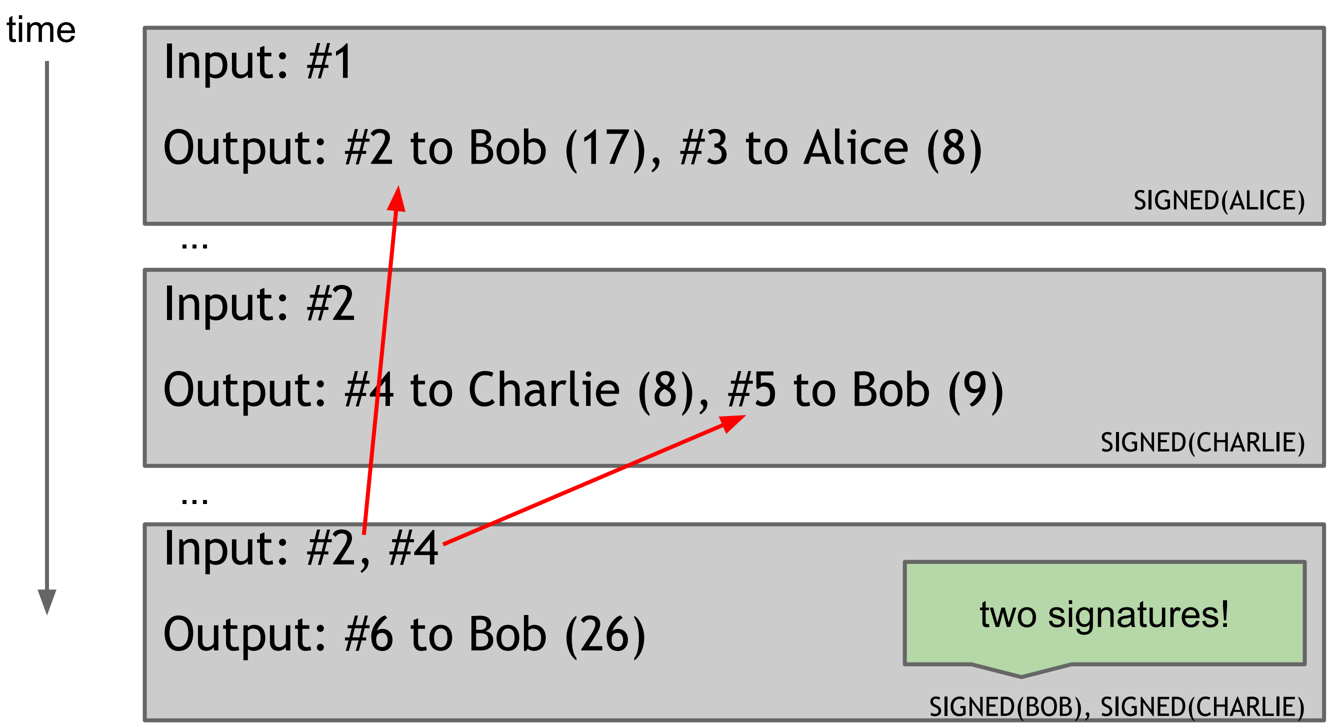
A transaction-based ledger (Bitcoin)



Merging value



Joint payments



A real Bitcoin transaction

metadata

input(s)

output(s)


```
{
    "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
    "ver":1,
    "vin_sz":2,
    "vout_sz":1,
    "lock_time":0,
    "size":404,
    "in":[
     {
       "prev_out":{
         "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
         "n":0
       },
         "scriptSig":"30440....3f3a4ce81"
     },
     {
       "prev_out":{
         "hash":"7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",
         "n":0
       },
       "scriptSig":"304602210....3f3a4ce81"
     }
    ],
    "out":[
     {
       "value":"10.12287097",
       "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"
     }
    ]
}
```

Transaction inputs ransaction inputs

previous transaction

signature

(more inputs)

```
"in":[
  {
    "prev_out":{
      "hash":"3be4...80260",
      "n":0
    },
"scriptSig":"30440....3f3a4ce81"
  },
...
],
```

Transaction outputs

```
"out":[
    {
      "value":"10.12287097",
      "scriptPubKey":"OP_DUP OP_HASH160 69e...3d42e
OP_EQUALVERIFY OP_CHECKSIG"
    },
    ...
]
```

output value

output address

(more outputs)

Why are addresses a script??

Output “addresses” are really *scripts*

```
OP_DUP
OP_HASH160
69e02e18...
OP_EQUALVERIFY OP_CHECKSIG
```

Input “addresses” are *also* scripts

scriptSig

```
30440220...
0467d2c9...
```

scriptPubKey

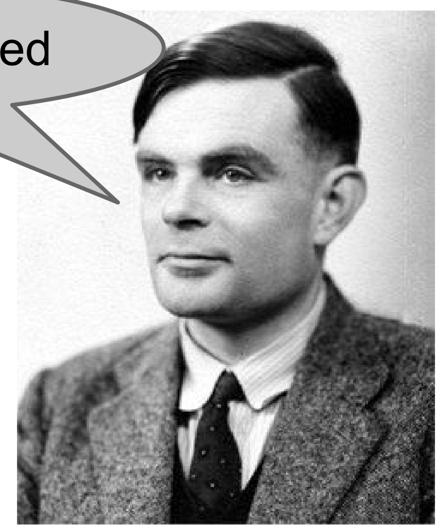
```
OP_DUP
OP_HASH160
69e02e18...
OP_EQUALVERIFY OP_CHECKSIG
```

TO VERIFY: Concatenated script must execute completely with no errors

Bitcoin scripting language (“Script”)

Design goals

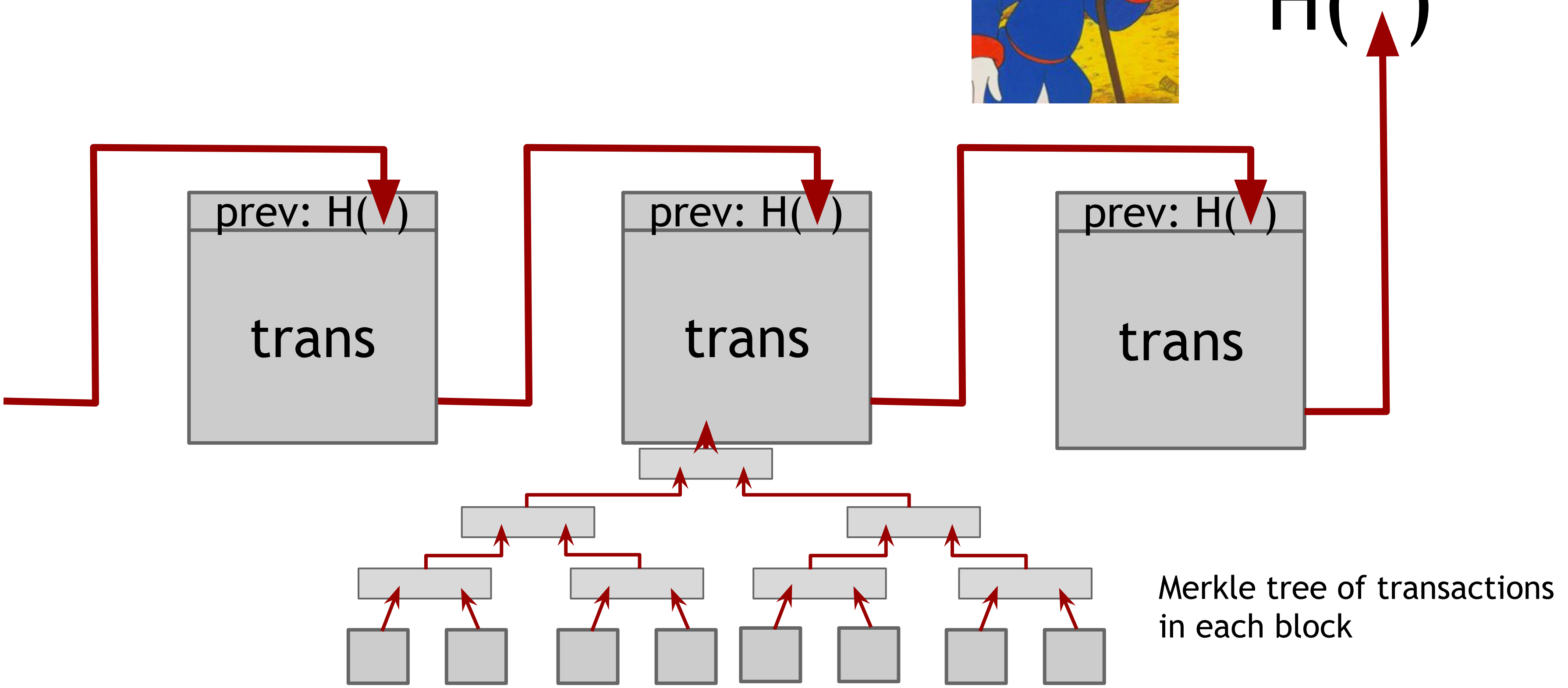
- Built for Bitcoin (inspired by Forth)
- Stack-based
- Simple, finite
- No looping
- Support for cryptography
  - MULTISIG addresses

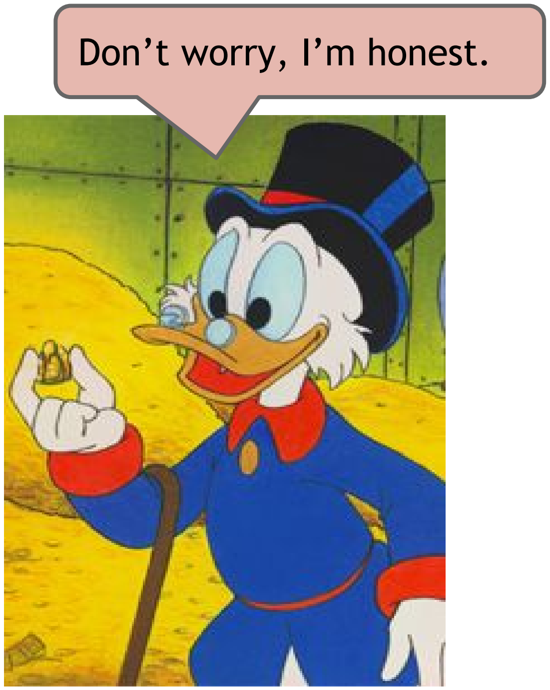


Lecture 1.6:

Centralized ledger (ScroogeCoin)

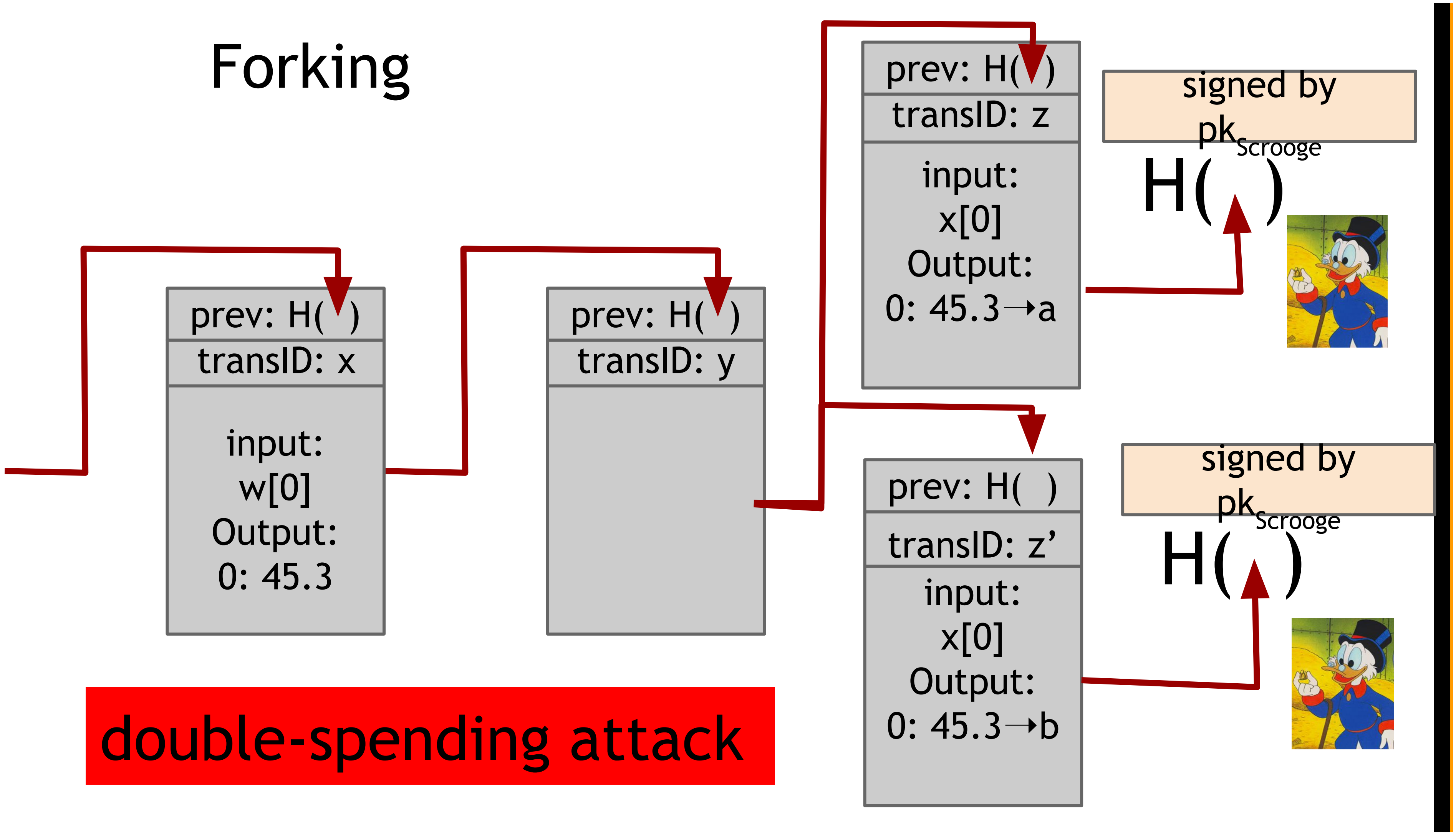
Scrooge publishes ledger of all transactions
(a blockchain, signed by Scrooge)





What if Scrooge is malicious?

Forking



double-spending attack

Other Scrooge problems

- Blacklist addresses
- Demand transaction fees
- Go offline
- Get hacked

Decentralization

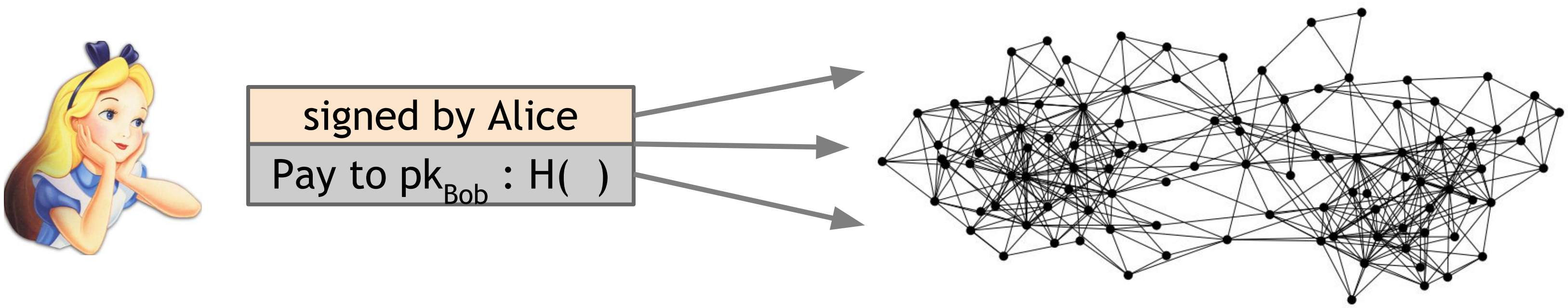
Can we avoid vulnerability to misbehavior by one entity?

Lecture 1.7:

Decentralized ledger: Bitcoin

Bitcoin is a peer-to-peer system

When Alice wants to pay Bob:
she broadcasts the transaction to all Bitcoin nodes



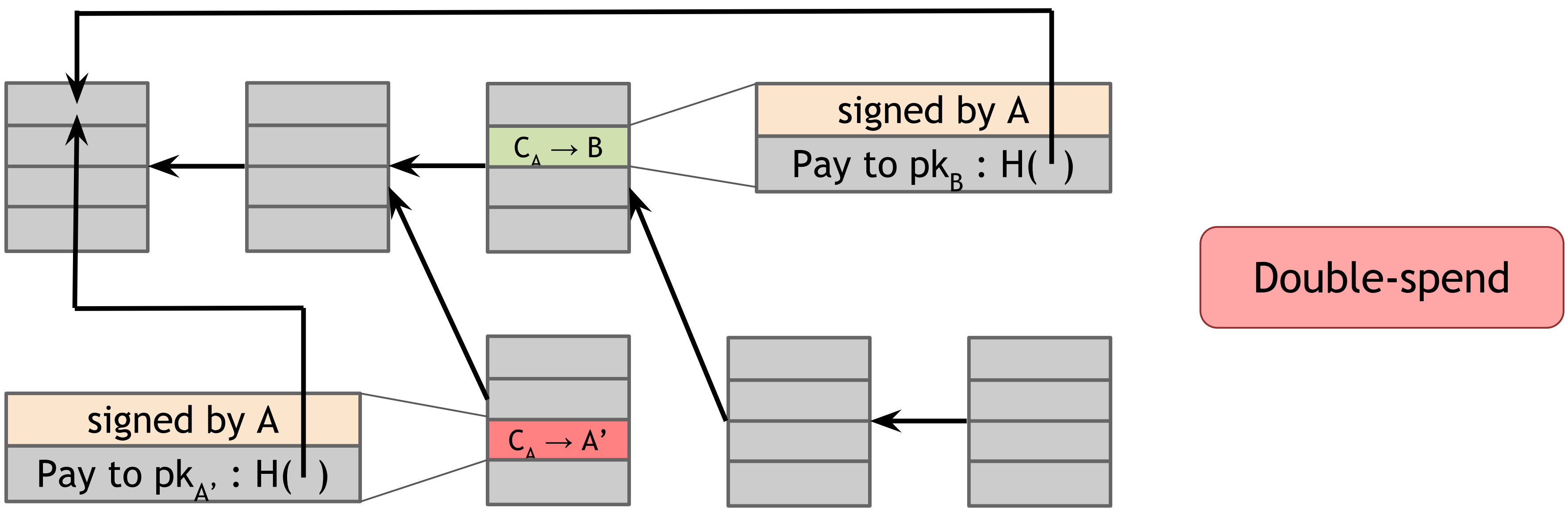
All nodes must agree on a sequence of transactions

Bitcoin consensus (simplified)

1. Transactions are broadcast to all nodes
2. In each round a random* node signs a block of new transactions, including the hash of the previous block
3. Other nodes accept the block if all transactions are valid
4. Invalid blocks are ignored, next node repeats this block
5. Longest chain is considered canonical

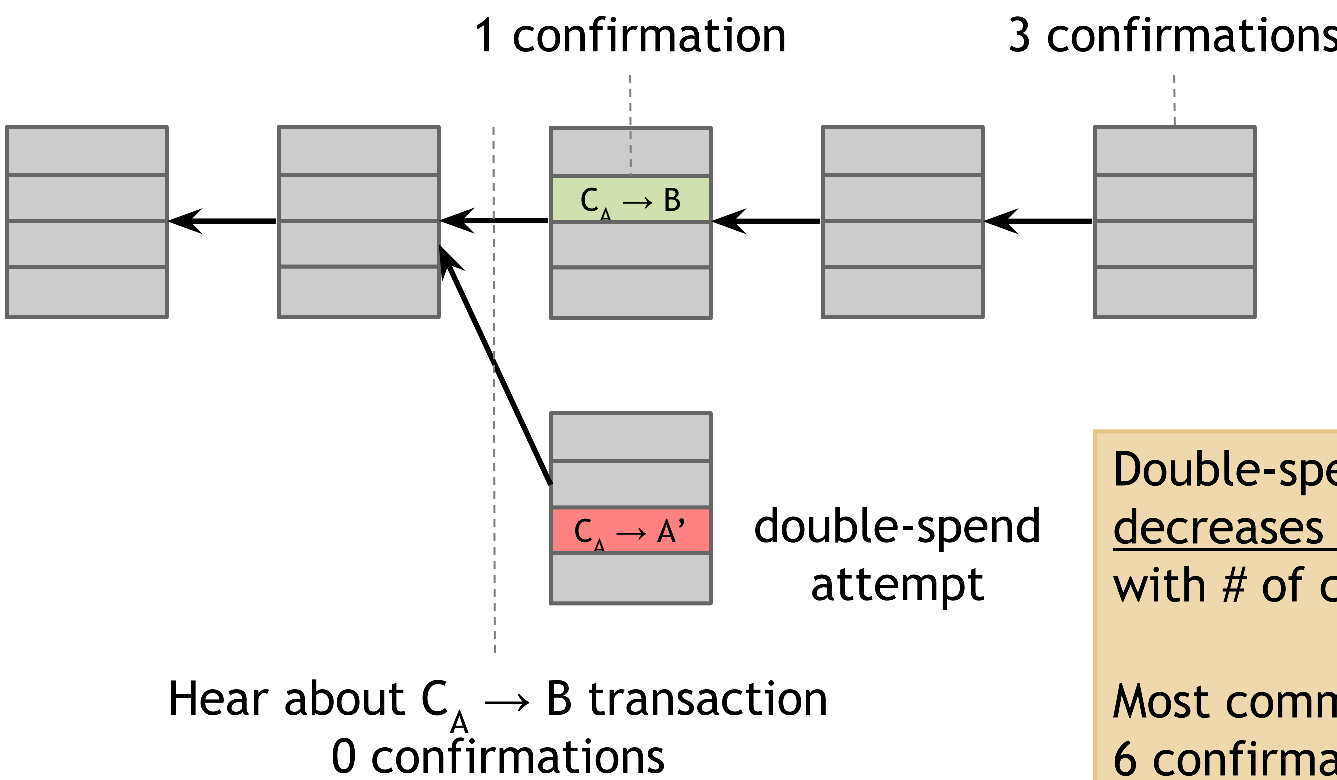
Leads to a valid canonical chain with “honest majority”

What can a malicious node do?



Honest nodes will extend the longest valid branch

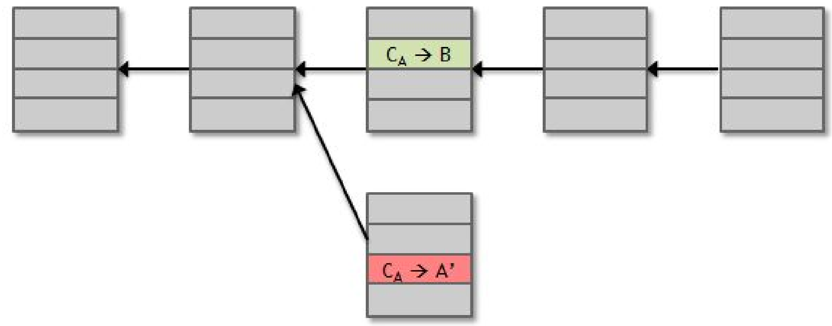
From a merchant's point of view



Double-spend probability <u>decreases exponentially</u> with # of confirmations

Most common heuristic:
6 confirmations

Basic properties

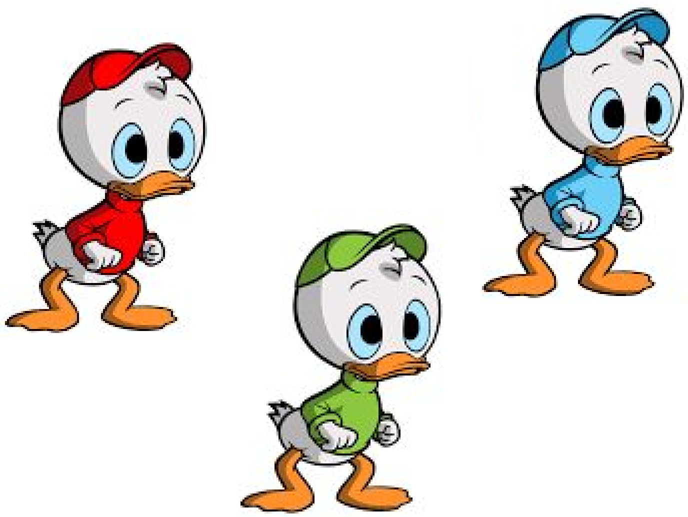


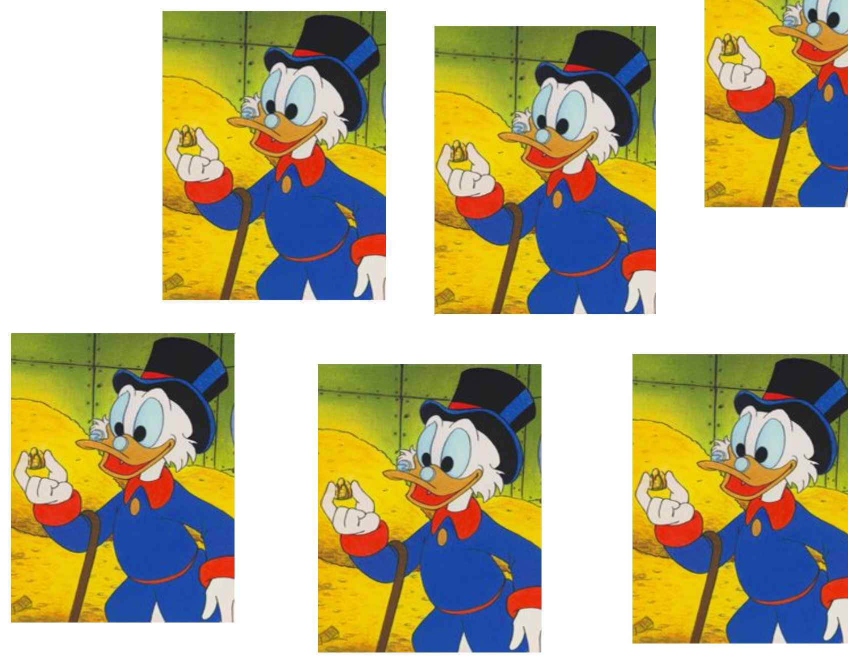
Protection against invalid transactions is cryptographic

Protection against double-spending relies on consensus

You're never 100% sure a transaction is in the blockchain

Honest majority of whom?





Recall: addresses can be freely created

Solution: “vote” by CPU power

Bitcoin mining puzzle:

Given previous block *prev*, new block *curr*:

Find *n* such that $H(prev|curr|n) < 2^{256-d}$

d is a difficulty parameter

First solution wins

SHA-256 is “puzzle-friendly”

Optimization-free

No better strategy than trying random nonces

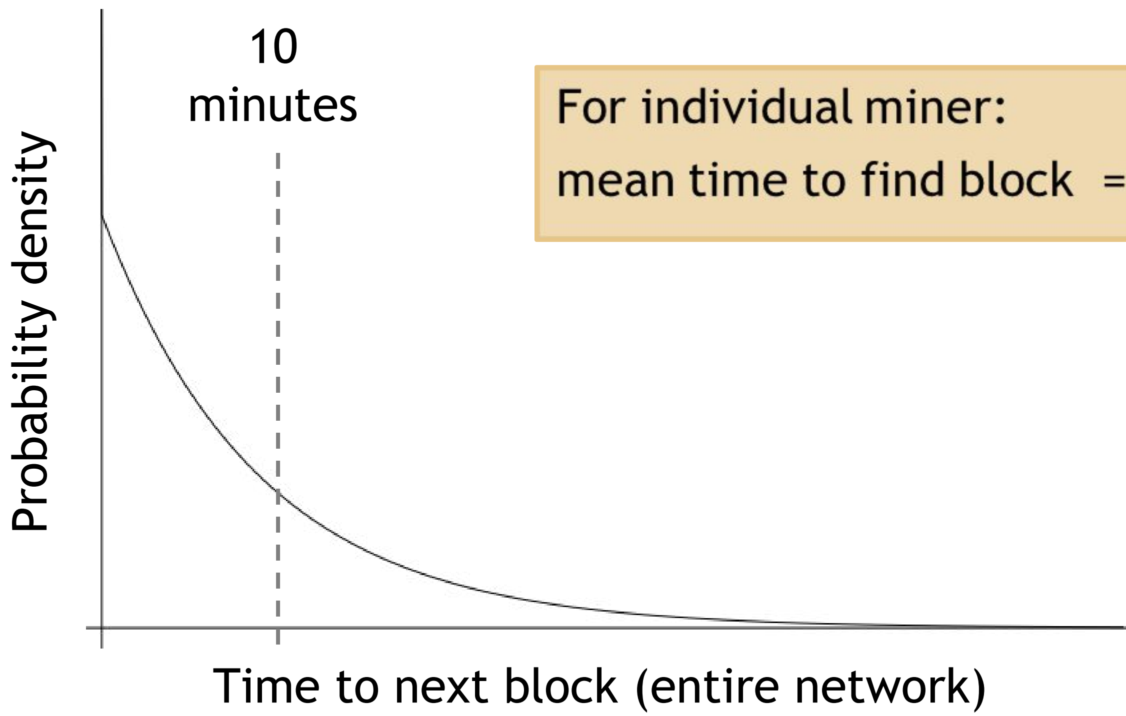
Progress-free

You don’t get any closer the more work you do

Parameterizable

Easy to adjust difficulty

Time to solution is probabilistic



For individual miner:

$$\text{mean time to find block} = \frac{\text{10 minutes}}{\text{fraction of hash power}}$$

Miners are rewarded for solutions

Creator of block gets to

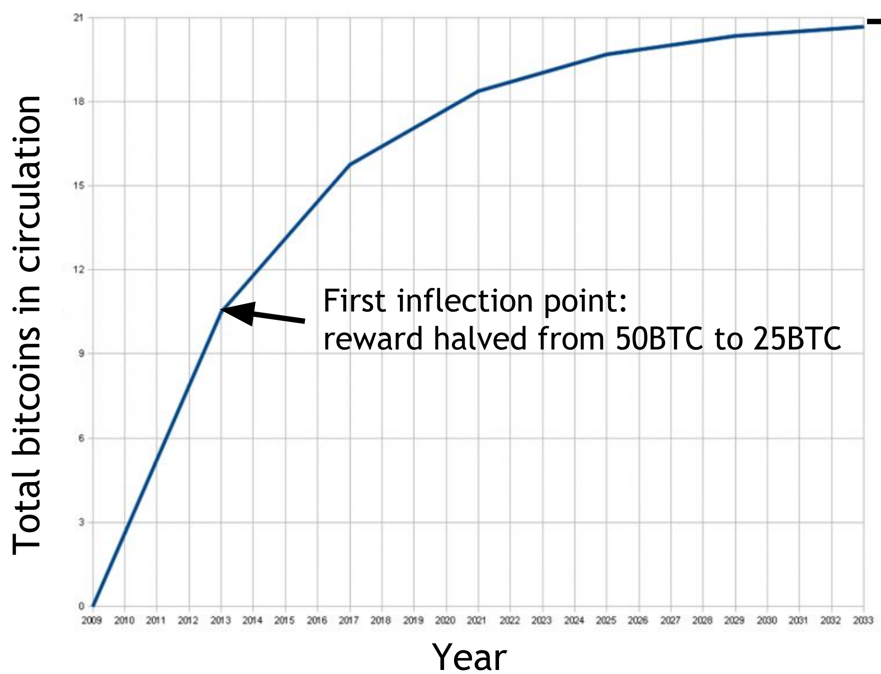
- include <u>special coin-creation transaction</u> in the block
- choose recipient address of this transaction

"Block reward" currently 25 BTC, halves every 4 years

Transaction fees also kept

Rewarded only if block is on eventual consensus branch!

There’s a finite supply of bitcoins



Total supply: 21 million

Block reward is how
new bitcoins are created

Runs out in 2040. No new
bitcoins unless rules change

Recap

Bitcoins created by special mining transactions

Bitcoins owned by public keys (addresses)

Bitcoin transfers authorized by digital signatures

Blockchain records all transfers, prevents double spends

Miners extend blockchain by solving proof of work

Miners rewarded by creating new bitcoins

Claims about Bitcoin

"Solves Byzantine agreement"

FALSE

"Secure if 51% of hash power is honest"

Depends on definition of "secure"

"Secure if everybody follows their incentives"

Nobody really knows

"Really interesting"

Hopefully

For more high-level background

Bitcoins and cryptocurrency technologies.

Narayanan, Bonneau, Felten, Miller, Goldfeder