Which Abstractions for the Blockchain Technology?

Emmanuelle Anceaume
CNRS / IRISA
emmanuelle.anceaume@irisa.fr
http://people.irisa.fr/Emmanuelle.Anceaume/

2008 : The Bitcoin white paper

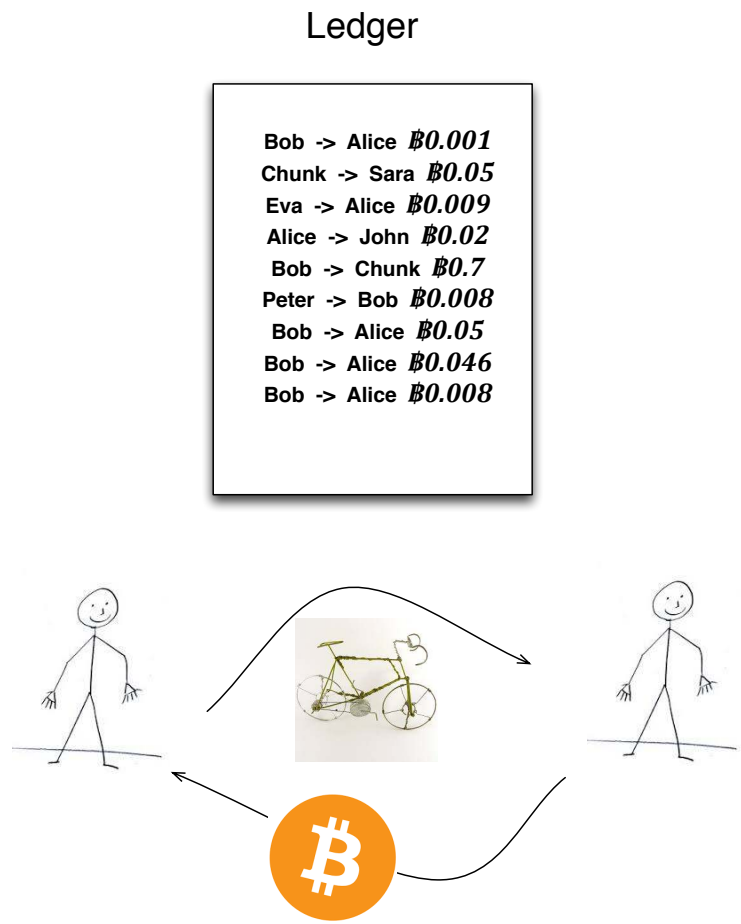
Bitcoin: A Peer-to-Peer Electronic Cash System

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.

An overview of Bitcoin

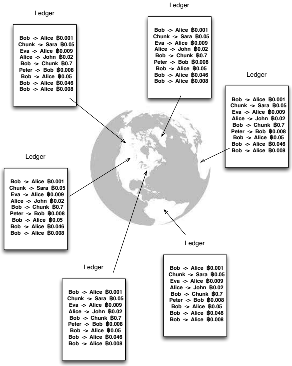
- Bitcoin is a cryptocurrency and payment system
  - It allows users to anonymously exchange goods against digital currency
- All the valid transactions are recorded in a public ledger, the blockchain
  - Allows anyone to audit and check the integrity of all the transactions



Public ledgers

No trustworthy centralized control.
Everyone

- maintains its own copy of the ledger
- disseminates all the transactions
- can read all the transactions



Achieving

- consistency and availability of the ledger

While preventing

- transaction censorchip and counterfeiting (i.e., double-spending attacks)

Content of this talk

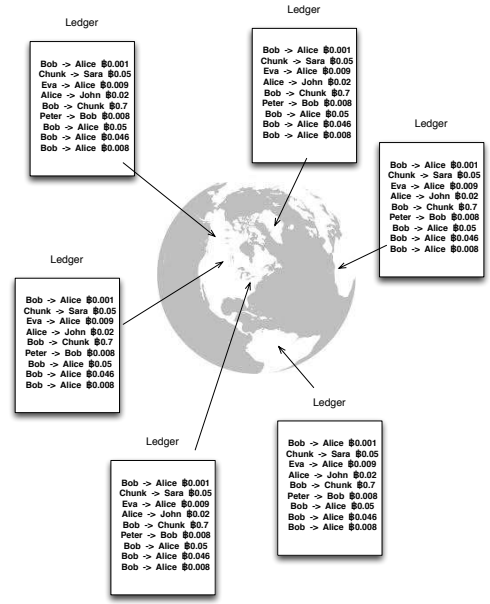
- The addressed problem
  - Construction of a cryptocurrency and payment system with no trusted third party
- Crypto Abstractions
  - hash functions, digital signatures, hash pointers, Merkle trees
  - Application : Bitcoin transactions
- Abstractions for a distributed cryptocurrency system
  - Communication primitive
  - Application : communication in Bitcoin
  - Incentive mechanisms
  - Application : minting process in Bitcoin
  - Nakamoto Consensus
  - Application : Construction of an immutable chain of blocks
- Conclusion

Which Abstractions ?

Digital currencies

- A string of « 0 » and « 1 »
- No central bank to prevent double spending attacks
- Solution: rely on cryptography

Crypto primitives



1. Abstraction 1 : cryptography

- cryptographic hash functions
- digital signatures
- Merkle tree

1.1 Hash functions

A hash function is an algorithm that allows to compute a fingerprint of fixed size from data of arbitrary size

$$\begin{aligned} H : 0,1^* &\rightarrow 0,1^n \\ M &\mapsto H(M) \end{aligned}$$

- rather than manipulating data of arbitrary size, a fingerprint is associated to each data which makes operation easier

1.1 Hash functions

A hash function satisfies the following properties

- The input space is the set of strings of arbitrarily length
  - « hello world » and « hellohellohello world » are perfectly fine inputs
- The output space is the set of strings of fixed length
  - H(« hello world ») = 000223
  - H(« hellohellohello world ») = 130554
- H is deterministic
- H is efficiently computable
  - Given a string s of length n the complexity to compute $H(s)$ is $O(n)$

In addition to these properties, crypto-hash functions have additional requirements

1.2 Properties of cryptographic hash functions

- Collision resistance

  It must be difficult to find two inputs x and x' such that
  $$H(x) = H(x')$$

- Second pre-image resistance

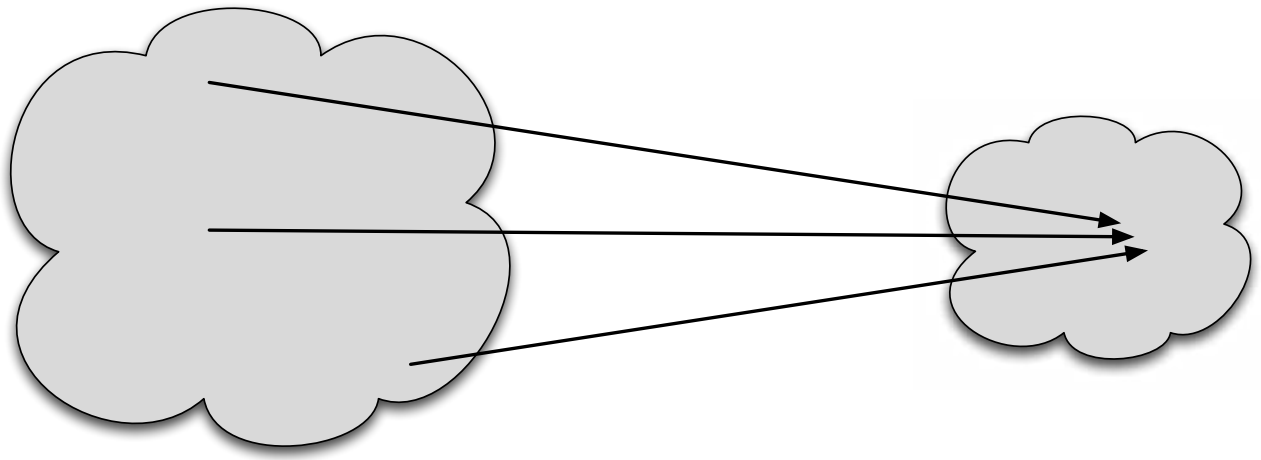
  Given an input x, it must be difficult to find an input value $x' \neq x$ such that $H(x') = H(x)$

- Pre-image resistance

  Given z, it must be difficult to find an input value x such that
  $$H(x) = z$$

Collision resistance

collisions do exist



possible inputs

possible outputs

Image source: *Bitcoin and Cryptocurrency Technologies*.

but can anyone find them ?

Collision resistance

Find two inputs x and x' such that $H(x) = H(x')$

Generic attack (i.e., a technique capable of attacking any hash function)

- Choose $2^{n/2}$ random messages
- Compute the hashed values and store them
- Find one pair (x, x') such that $H(x) = H(x')$

If a computer calculates $10,000$ hashes/s

- it would take 10^{27} years to output 2^{128} hashes, and
- thus 10^{27} years to produce a collision with probability $1/2$

Astronomical number of computations ! !
So far no hash functions have been proven to be collision resistant

Collision resistance

Collision resistant hash functions allows us

- to identify data by its hashed value (i.e digest, fingerprint)
  - if $H(x) = H(y)$ then it is safe to assume that $x = y$
- Bitcoin :
  - to identify transactions
  - to make blocks resistant to tampering (modifying a single bit changes the fingerprint)

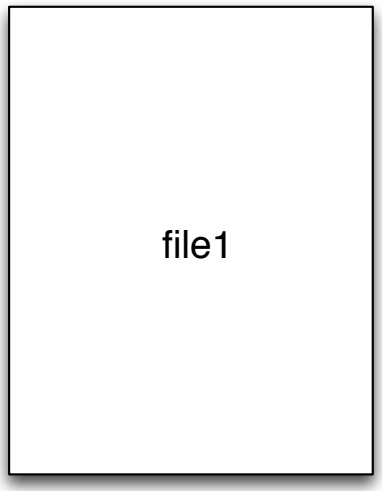
Second-preimage resistance

Given an input x, it is difficult to find an input value $x' \neq x$ such that $H(x') = H(x)$

Generic Attack : probabilistic search

- Given x and its hashed value $H(x)$ (n bits value)
- Randomly choose x_i and compute $z_i = H(x_i)$
- Proba$(z_i = H(x)) = 1/2^n$
- Thus after having chosen 2^n inputs one can find a pre-image $x_i \neq x$ such that $H(x_i) = H(x)$

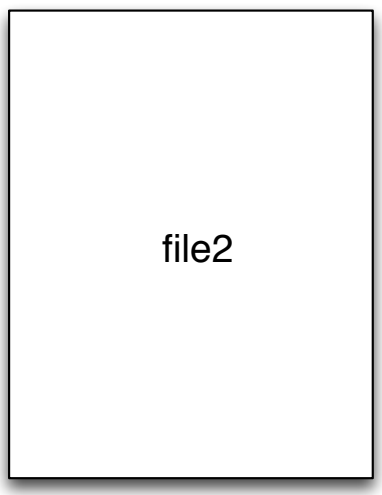
File integrity



Assume you have file 1

H(file1) = 3214 5670 ab67 0123 8760
2123 34BF 0A23

(256 bits)



Is it possible to build another file, file 2, such that both files have the same fingerprint?

Preimage resistance

Given z, find an input value x such that $H(x) = z$

Generic Attack : probabilistic search

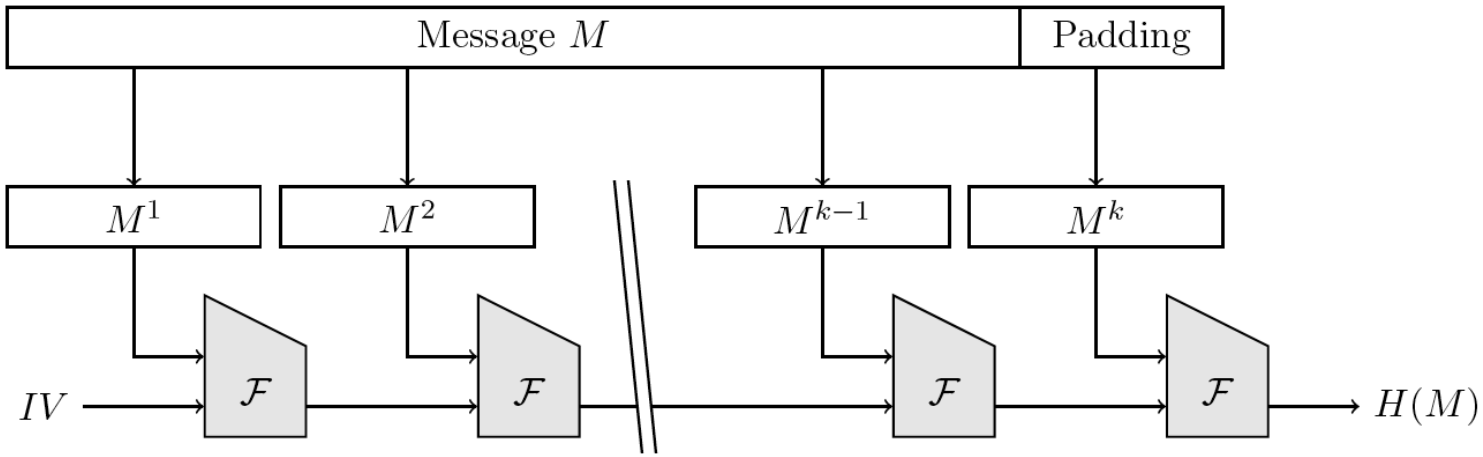
- Given a hashed value z
- Randomly choose x_i and compute $z_i = H(x_i)$
- Proba$(z_i = z) = 1/2^n$
- Thus after having chosen 2^n inputs one can find a pre-image x_i such that $H(x_i) = z$

Password storage

- In your machine, passwords are not stored. Only their hashed values are stored
- When you want to authenticate, the login pg computes the hashed value, which is compared with the one stored in /etc/passwd

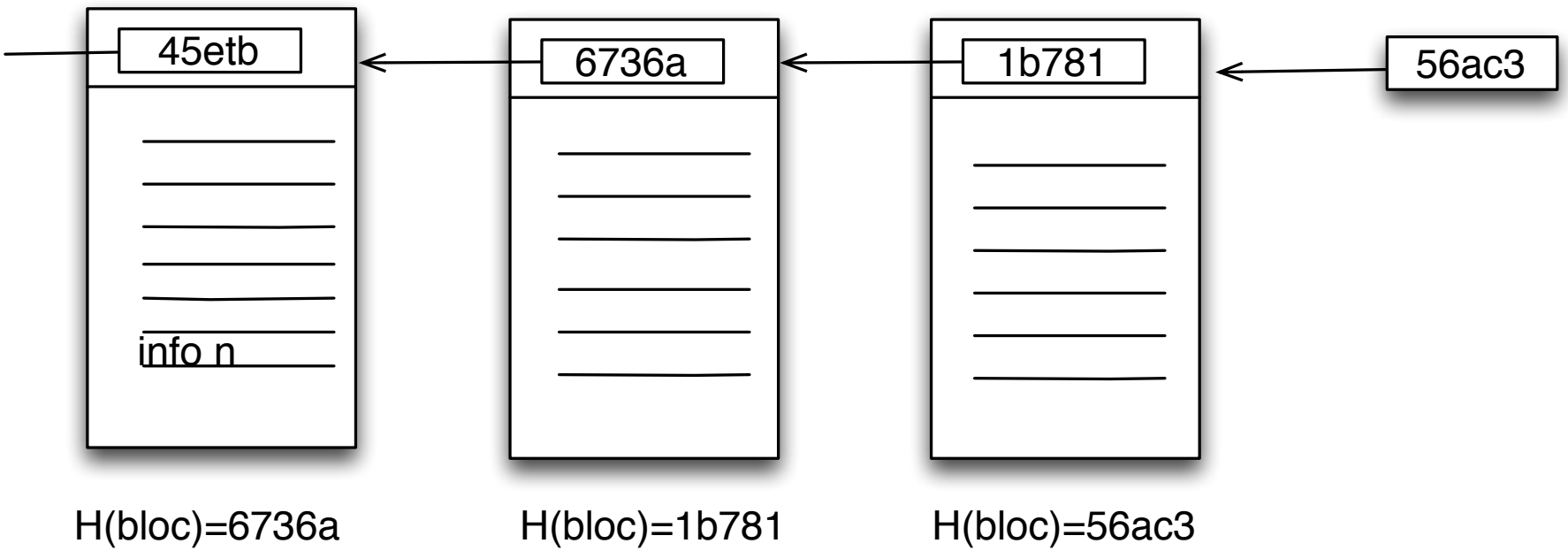
Property : Given the hashed value y it must be difficult to find x such that $H(x) = H(password) = y$

Merkle-Damgard construction



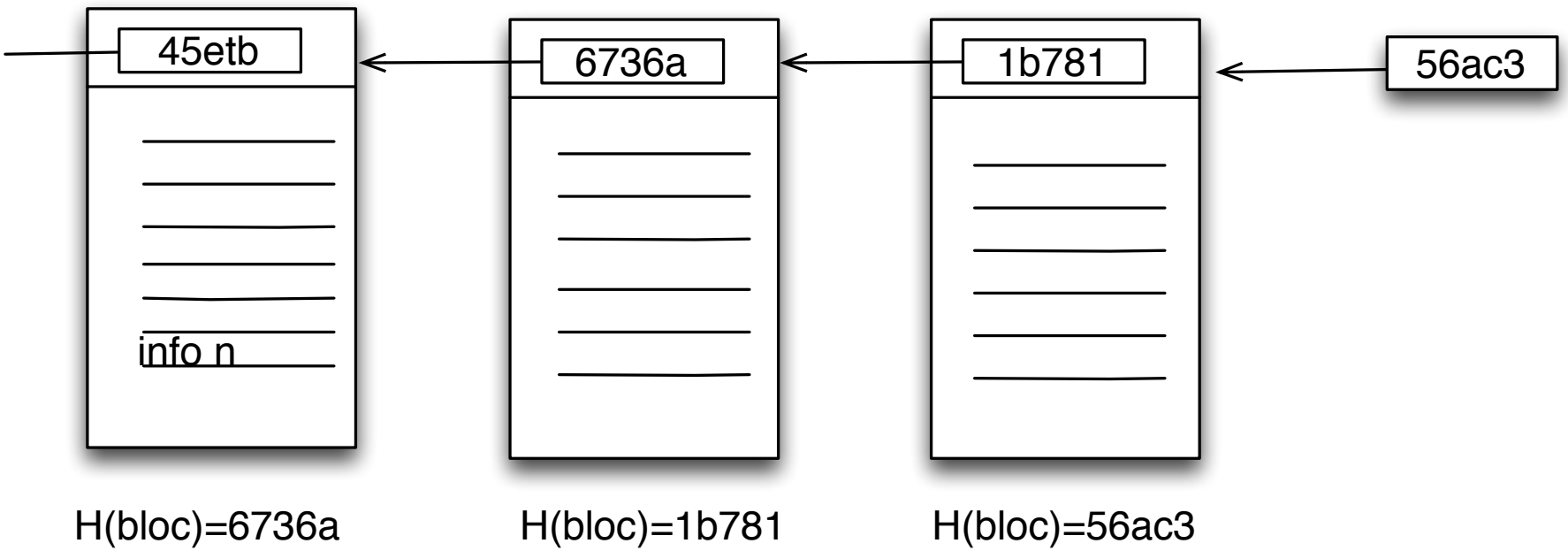
1.2 Hash pointers

A hash pointer is the cryptographic hash value of the pointed information



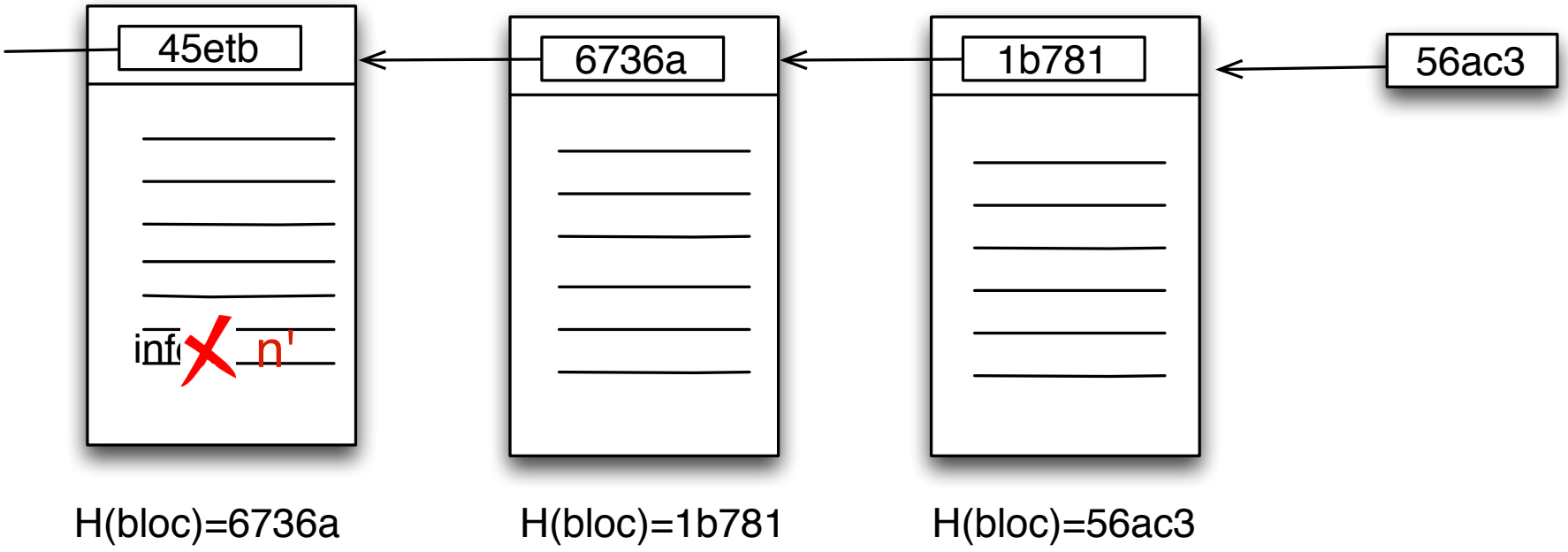
1.2 Hash pointers

Hash pointers allows the construction of a log data structure that allows the detection of any manipulation



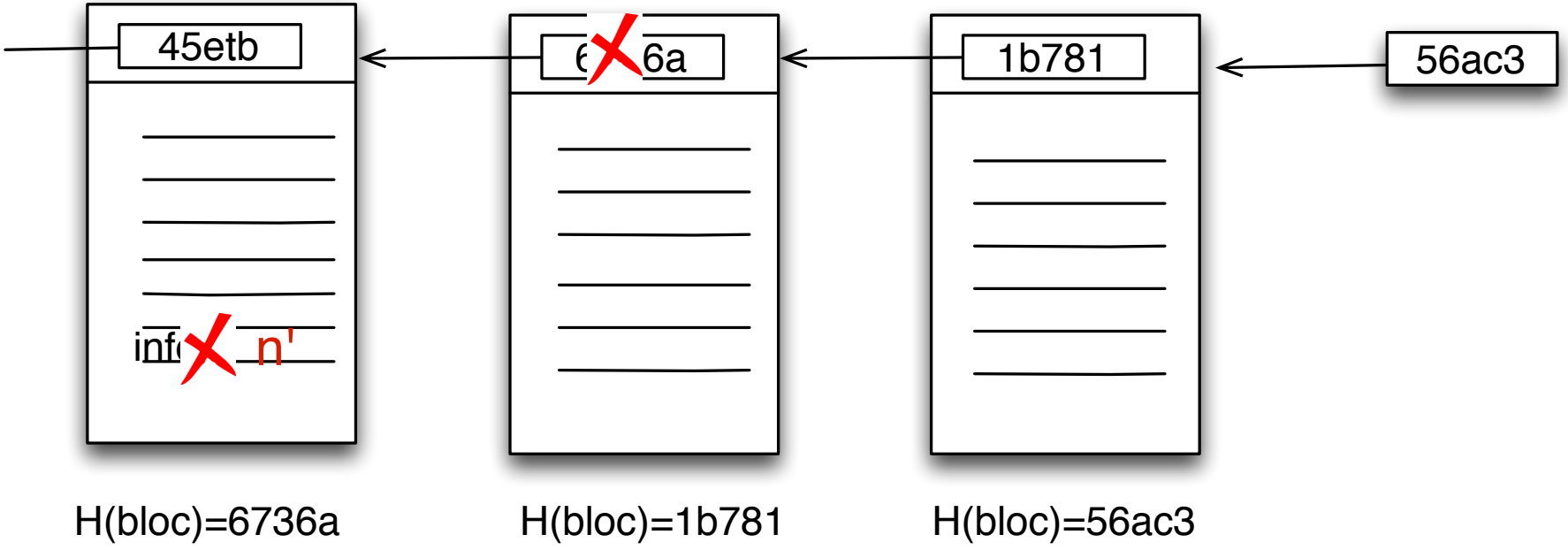
1.2 Hash pointers

Hash pointers allows the construction of a log data structure that allows the detection of any manipulations



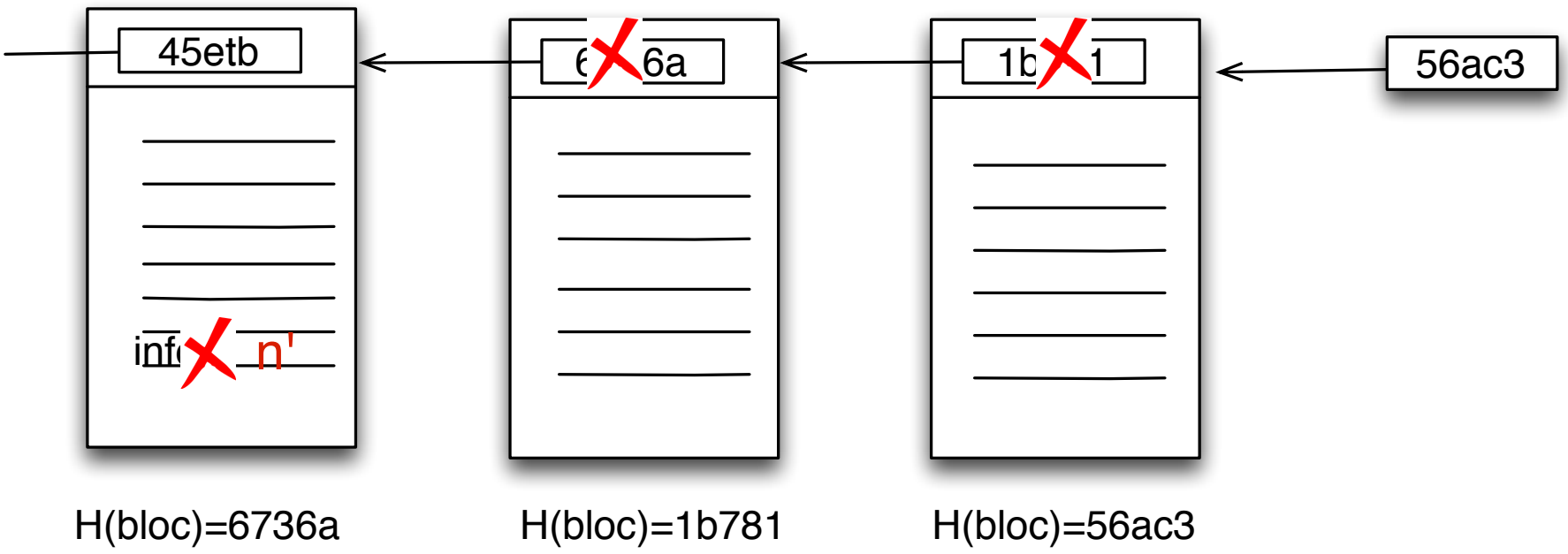
1.2 Hash pointers

Hash pointers allows the construction of a log data structure that allows the detection of any manipulations



1.2 Hash pointers

Hash pointers allows the construction of a log data structure that allows the detection of any manipulations



✓ By only keeping the hash pointer of the head of the data structure, we have a tamper-evident hash of a possibly very long list

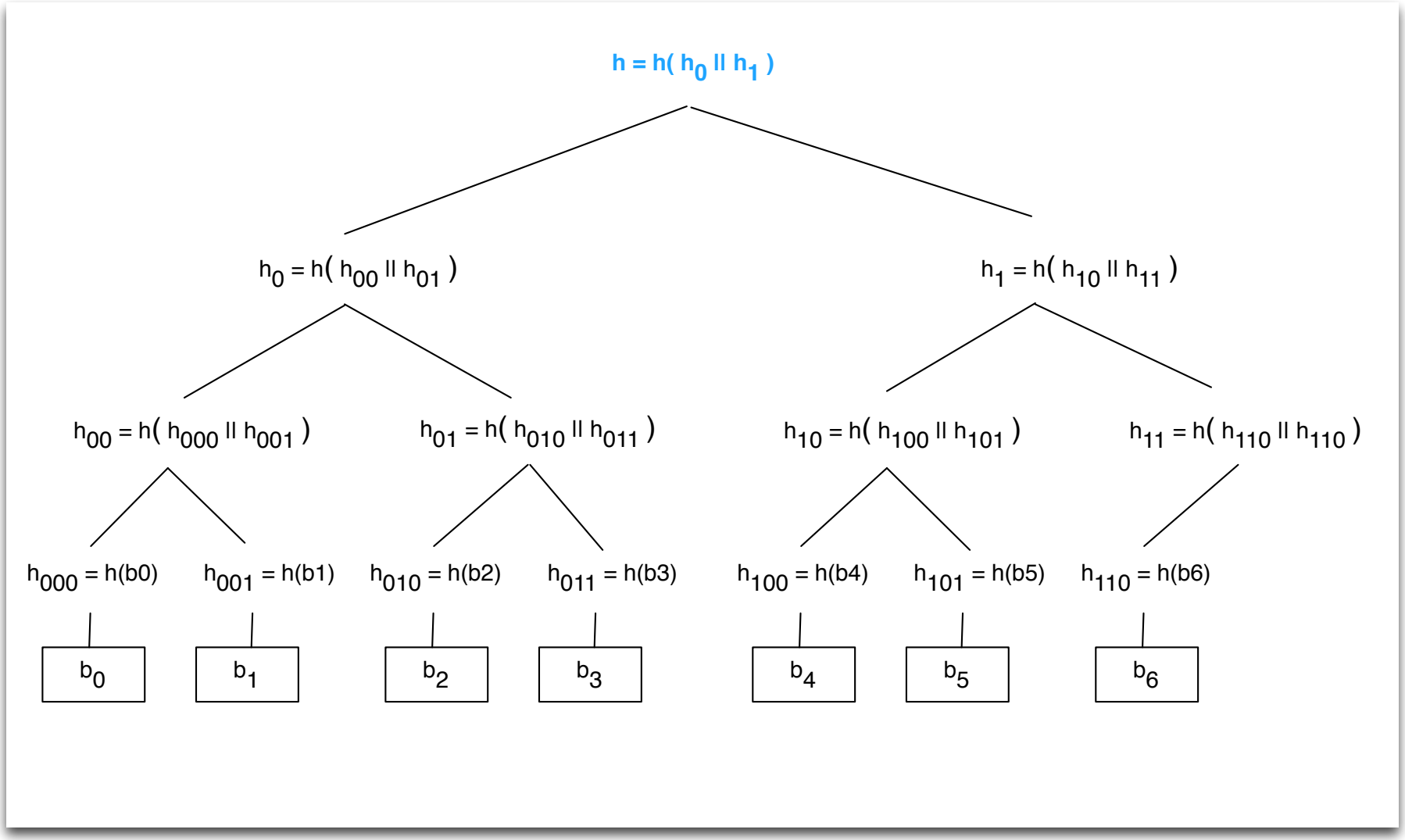
1.3 Hash tree : Merkle Tree

A Merkle tree[1] is a tree of hashes

- Leaves of the tree are data blocks
- Nodes are the hashes of their children
- Root of tree is the fingerprint of the tree

1. Merkle, R. C. (1988). "A Digital Signature Based on a Conventional Encryption Function". Advances in Cryptology - CRYPTO '87.

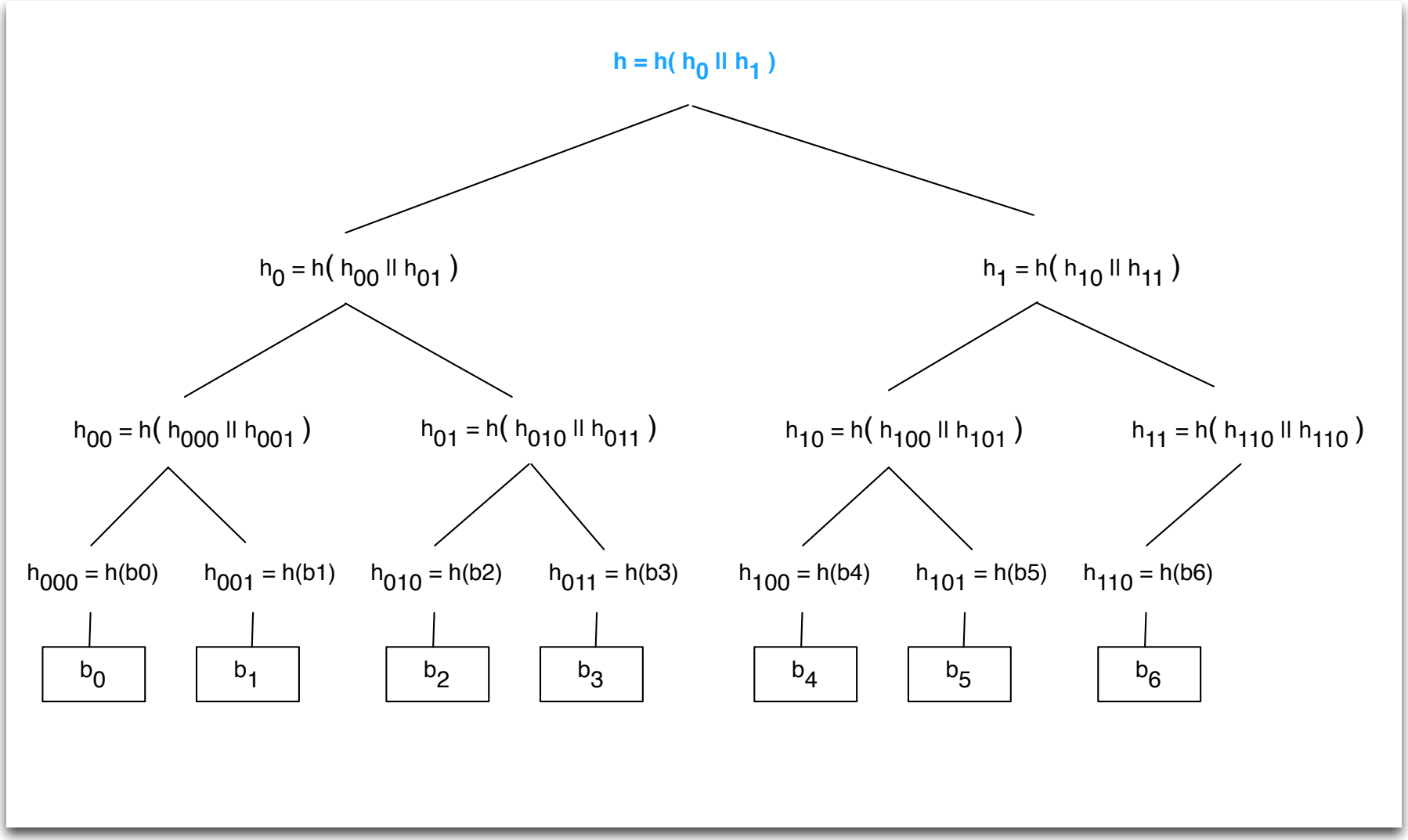
1.3 Hash tree : Merkle Tree



1.3 Hash tree : Merkle Tree

- ✓ Checking the integrity of the n data blocks of the tree
  - easy due to collision resistance property of crypto. hash functions
- ✓ Data blocks membership
  - checked with $\log n$ pieces of information and in $\log n$ operations

1.3 Hash tree : Merkle Tree



1.4 Digital signature primitive

A digital signature is just like a signature on a document

- Only the creator of the document can sign, but anyone can verify it
- Signature is tied to a particular document

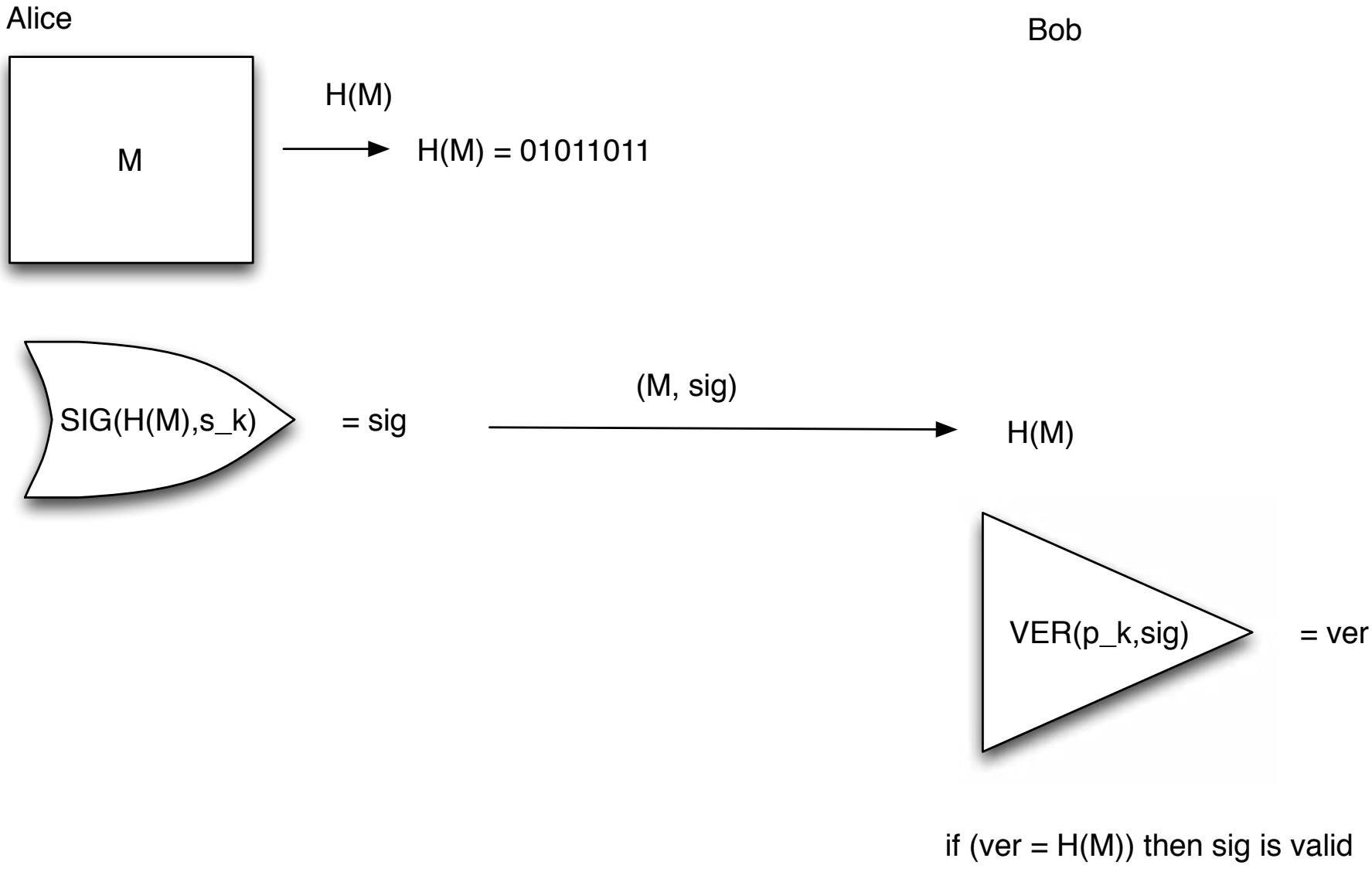
How can we build such a digital signature ?

1.4 Digital signature

Three functions :

- (s_k, p_k) := generateKeys(keysize)
  - s_k : private signing key
  - p_k : public verification key
- sig := sign(s_k, H(message))
- ver := verify(p_k, sig)

1.4 Digital signature



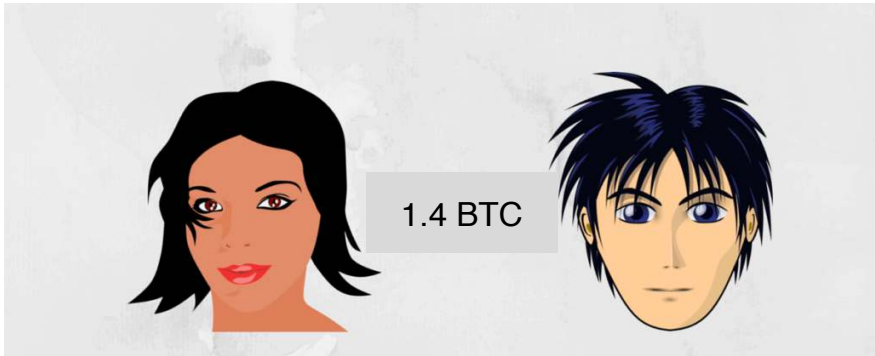
Using verification public key as an identity

Idea [2]

- If you see (m,sig) such that the signature verifies under p_k (i.e. *verify*(p_k, *sig*)= *true*) then one can see p_k as a party saying statements by signing them

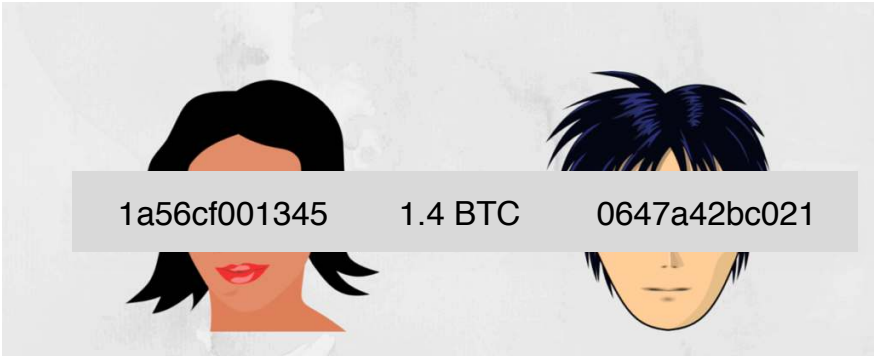
2. D. Chaum, « Untraceable Electronic Mail, Return Addresses, and Digital Pseudonums », Communications of the ACM 24(2), pp :84–90,1981

No identities



- Alice and Bob want to transfer values
- But Alice does not want to use neither her identity nor Bob's one in the transaction

Using verification public key as identities



- If you see a msg such that the signature verifies under p_k (i.e. *verify*(p_k, *msg*, *sig*)= *true*) then on can see p_k as a party saying statements by signing them
- To speak on behalf of p_k one must know s_k
- So there is an identity in the system such that only a single one can speak for it [3]

3. D. Chaum, « Untraceable Electronic Mail, Return Addresses, and Digital Pseudonums », Communications of the ACM 24(2), pp :84–90,1981

Using Verification public key as identities

- ✓ By looking at public keys as identities you can generate as many identities as you want !
- ✓ No central authority in charge of registering them !

Using verification public key as an identity

What about privacy ?

- no relationships between p_k based identities and real identities
- by using the same p_k (identity) an adversary can infer some relationships based on the activity of p_k

Using the « public key as identity » principle in crypto-payment systems

- Most of the crypto-currencies use this principle to handle accounts
- An account is a pair (p_k, s_k) and an amount of coins

First detour to Bitcoin : Transactions

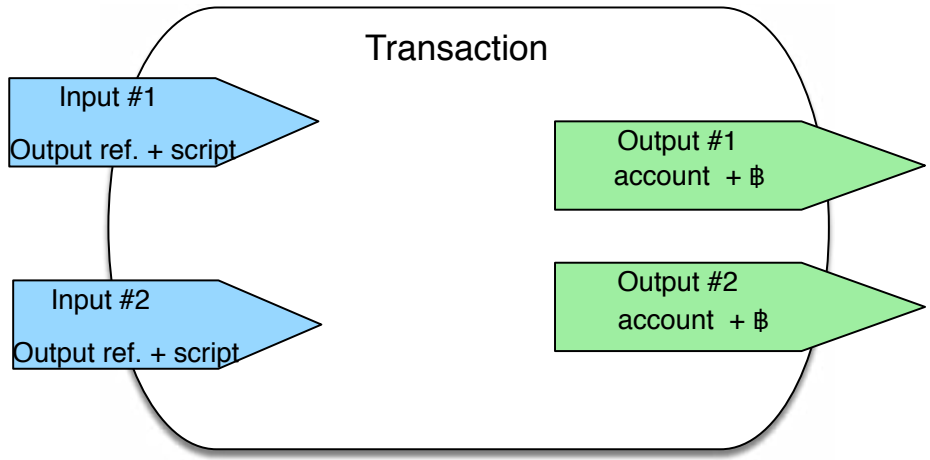
- Transactions : state of the crypto-currency system
  - data structure that allows Alice to transfer bitcoins to Bob and Charly
  - does not contain any confidential information
  - verifiable by anyone $\rightarrow$ no trusted third-party !

Transaction : account

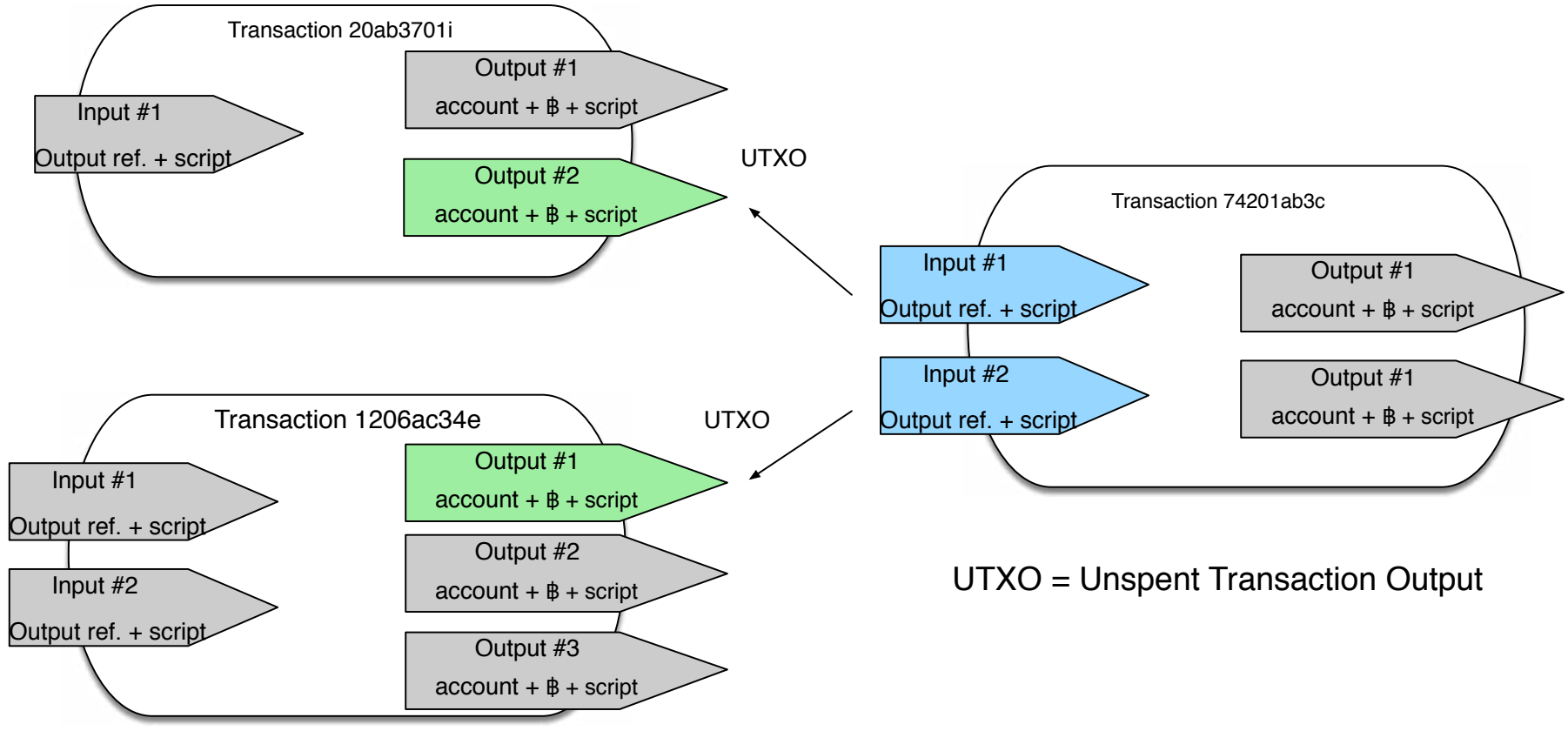
- An account is a « one shot object »
  - an account is a <(private key,public key), amount of bitcoins>
  - relies on the *public key as identity* principle
  - debited once !
  - each time you are the recipient of a transaction, it's safer if you create a new account (privacy reasons)

Transaction : structure

- Alice's account (input address)
- Bob's and Charly's accounts (output address)
- possibly some change
- transactions fees



Relying on past transactions to create new ones

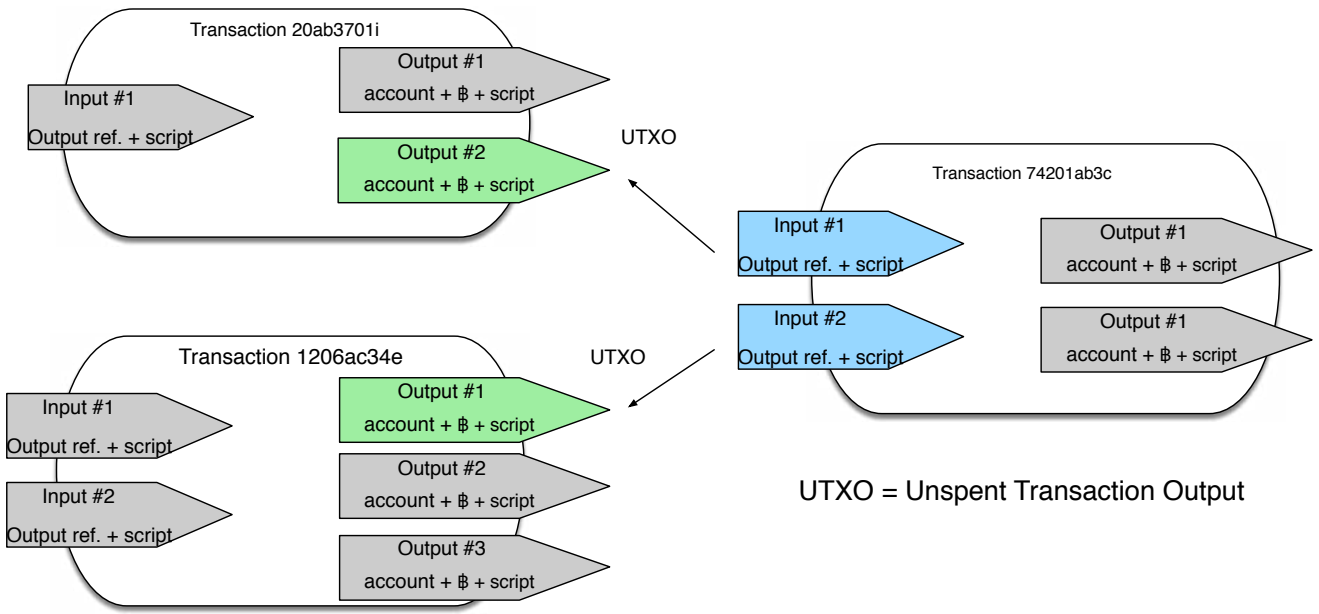


UTXO = Unspent Transaction Output

Do not forget Tx fees, i.e., ฿ input > ฿ output

Transaction : script

- Bitcoin relies on a (limited) script language
- to lock an output, the script provides a challenge
  - *i.e.*, fingerprint of the recipient public key, $H(p_k)$
- to unlock an input, the script provides the solution of the challenge
  - *i.e.*, public key p_k together with s_k's signature



UTXO = Unspent Transaction Output

Do not forget Tx fees, i.e., ฿ input > ฿ output

Script language

Decoded:

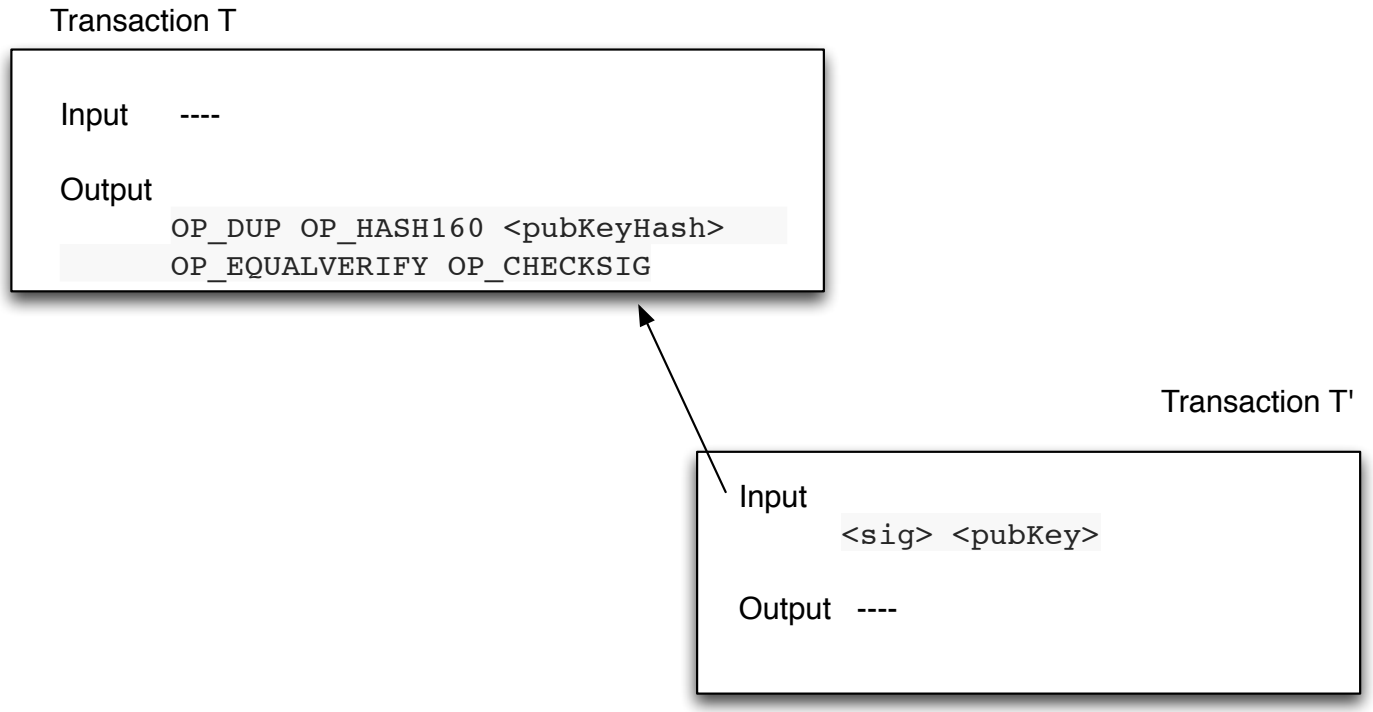
- Header: ver=1, vin.size=1, vout.size=2, nLockTime=438069
- Inputs:
  - ID: 4983442ccd814372b8c8614bf81942ba17c6e3470f21897524a63caf54aa54ba
  - Index: 1 (input value: 0.030 980 35 BTC)
  - scriptSig:
    - Signature: 304402201e0b0555330b9ba6dc689aeebecf0464391a882c41a6650ab66f803179860a1802207d1b46c45d37e8a88fee49c3e02adb9cd3ccb4bb96ca313135e00a5c01f71a6b[ALL]
    - Key: 02374f390070a14763707fe9310a73eaf2b2221734d0ff0a0684078571e2a12e9e
  - nSeq: 4294967294

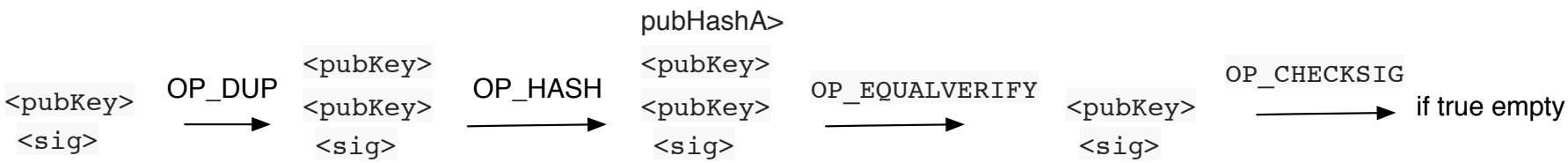
Script language

Decoded (ctd):

- Outputs:
  - n: 0
    - value: 0.016 858 15 BTC
    - ScriptPubKey: `OP_DUP OP_HASH160` 3d5b9da23ff21a211f101ee2adec37d6b797db7c `OP_EQUALVERIFY OP_CHECKSIG`
  - n: 1
    - value: 0.010 000 00 BTC
    - ScriptPubKey: `OP_DUP OP_HASH160` 4ce03f31d4bdbc2932f14cea99f4d96edcdbef0c `OP_EQUALVERIFY OP_CHECKSIG`

Script language





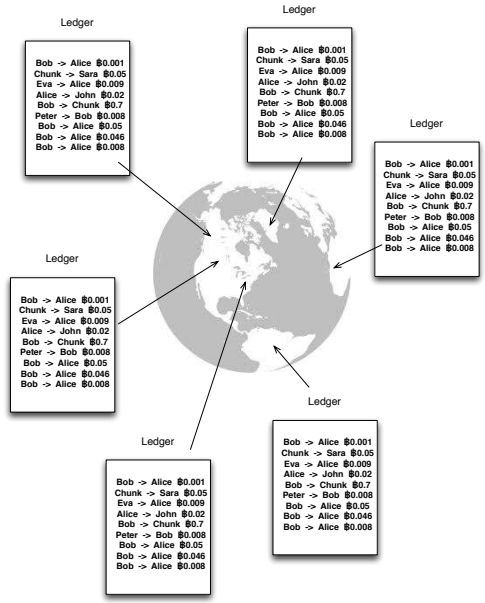
Which Abstractions ?

Crypto primitives

Hash function
Digital signature
Merkle tree

Broadcast primitive

Dissemination of information in the open Bicoin network

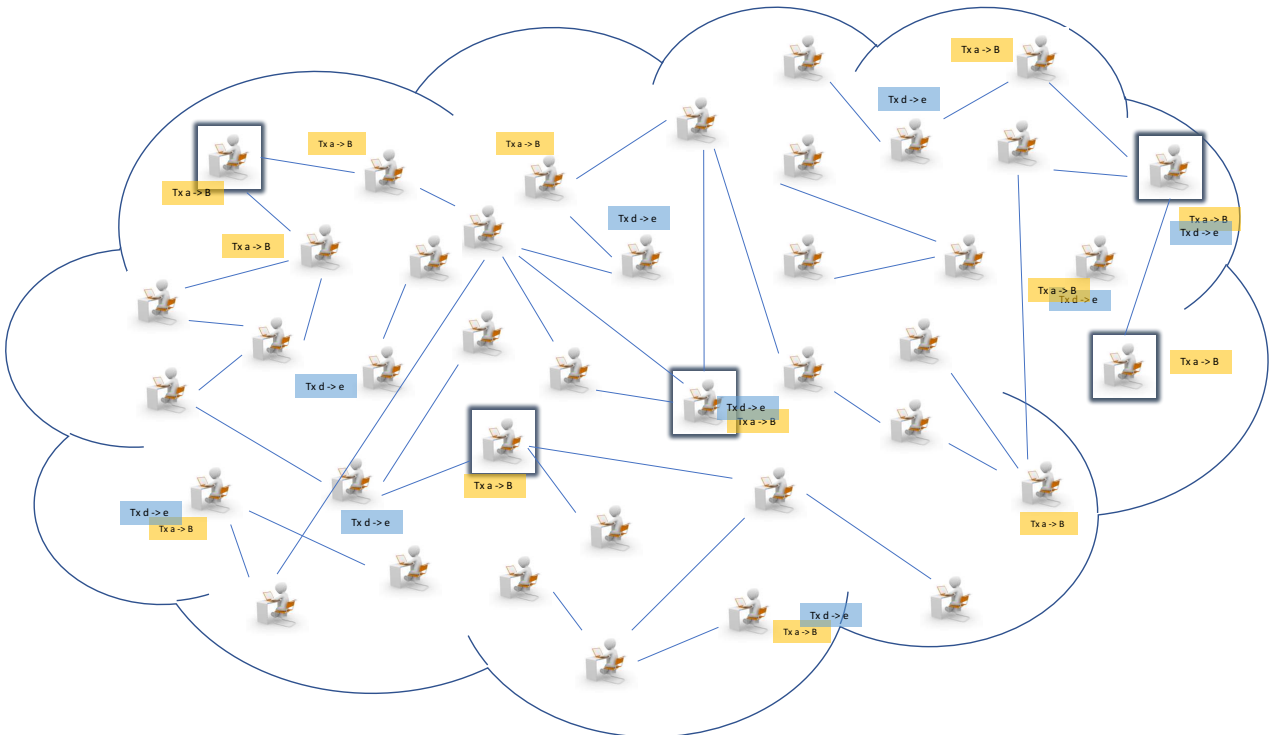


Abstraction 2 : Communication primitive

- Size : set of thousands or millions of nodes $\mathcal{N}$
- Dynamic : nodes can join and leave the system at will
- Randomness is fundamental
- Any individual in the system should have the same probability to be chosen as a neighbor of any other individual

Bitcoin Network

- Peer-to-peer network[d]
- Topology formed through a randomized process
- No coordinating entity
- Broadcast is based on flooding/gossiping neighbors' link



4. Gnutella (2000), GIA (2003), Cyclon (2005)

Abstraction 2 : properties

1. Fully decentralized
2. Uniformity
3. Low latency [4]

$$\frac{\text{msg. transmission time}}{\text{block time interval}} \ll 1$$

✓ Allows to keep the probability of fork small (i.e. 10^{-3})

✓ PoW mechanism allows this ratio to continuously hold

4. J. Garay and A. Kiayias, The Bitcoin Backbone Protocol : Analysis and Application, Eurocrypt 2015

Which Abstractions?

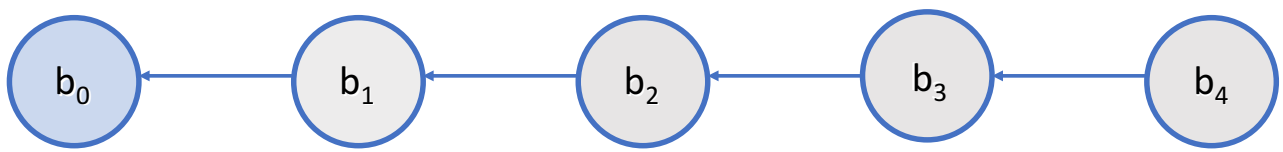
Hash function
Digital signature
Merkle tree

Dissemination of information in the open Bicoin network

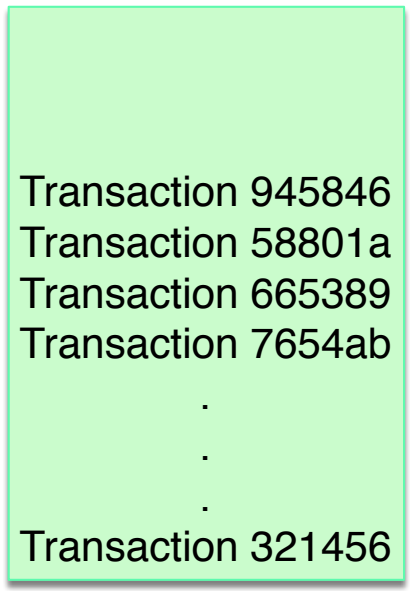
Persistent ordering of all the transactions ever created with no trusted party

Abstraction 3 : Agreement on the content of the ledger

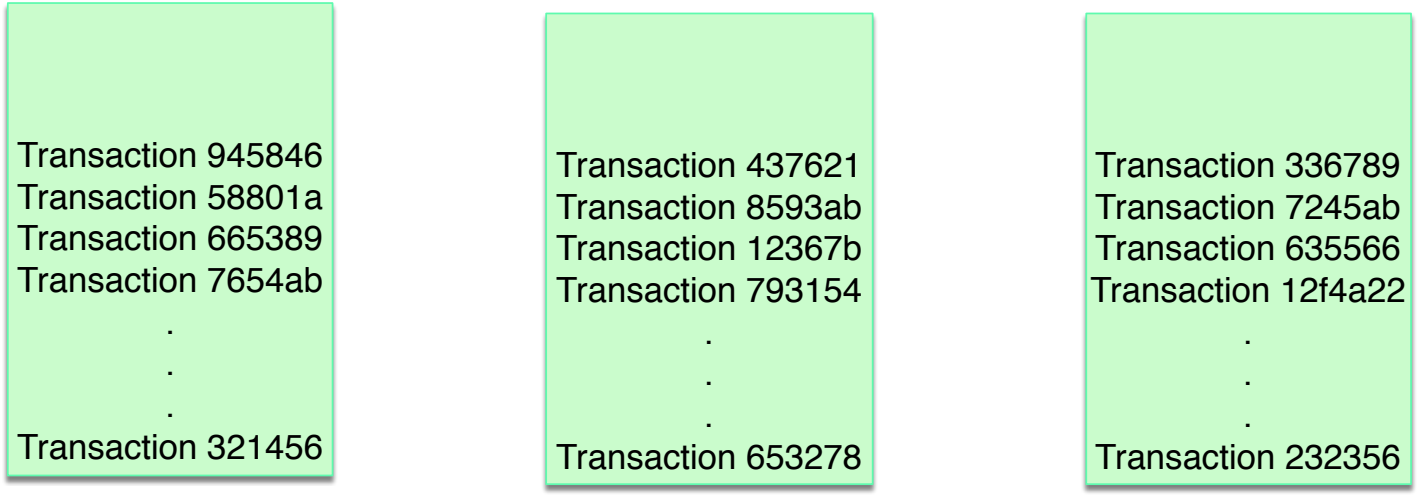
A publicly, immutable, and ordered log of transactions



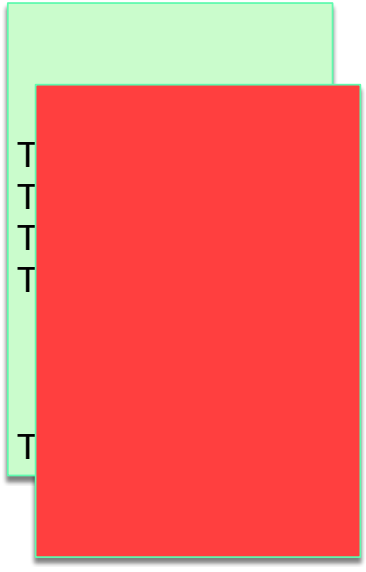
Block of transactions



Blocks of transactions



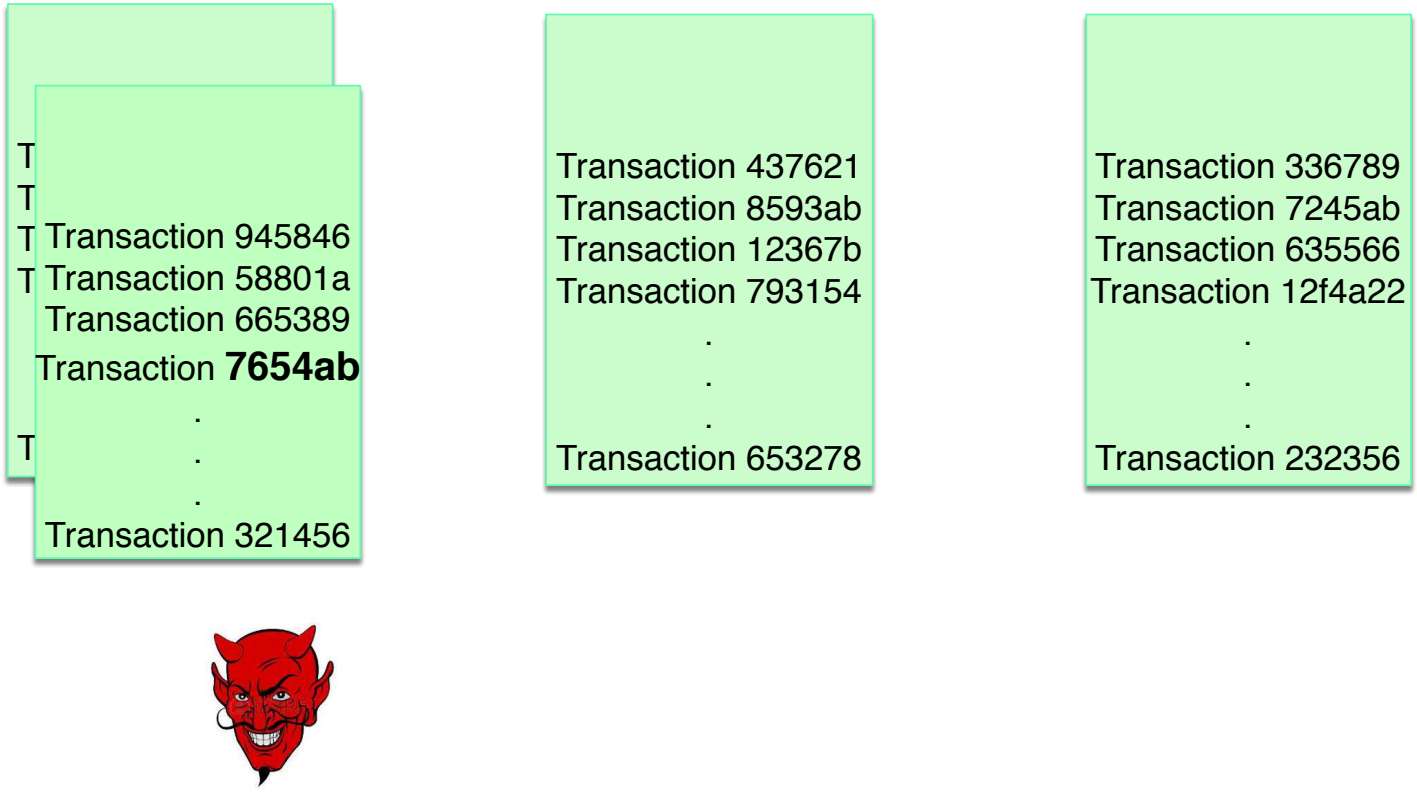
Resistant to attacks



Transaction 437621
Transaction 8593ab
Transaction 12367b
Transaction 793154
.
.
.
Transaction 653278

Transaction 336789
Transaction 7245ab
Transaction 635566
Transaction 12f4a22
.
.
.
Transaction 232356

Resistant to attacks



A chain of blocks : the blockchain

A publicly, immutable, and totally ordered log of transactions

- Why digital signatures are not sufficient ?
- How can we securely link blocks ?
- How can we prevent transactions in a block from being manipulated ?
- Who is allowed to create blocks ?
- Who is in charge of checking that blocks are correctly created ?

A chain of blocks : the blockchain

A publicly, immutable, and totally ordered log of transactions

- Why digital signatures are not sufficient ?
  - ✗ do not prevent double-spending attacks
- How can we securely link blocks ?
- How can we prevent transactions in a block from being manipulated ?
- Who is allowed to create blocks ?
- Who is in charge of checking that blocks are correctly created ?

A chain of blocks : the blockchain

A publicly, immutable, and totally ordered log of transactions

- Why digital signatures are not sufficient ?
  - ✗ do not prevent double-spending attacks
- How can we securely link blocks ?
  - ✓ by linking them with cryptographic fingerprints
- How can we prevent transactions in a block from being manipulated ?
- Who is allowed to create blocks ?
- Who is in charge of checking that blocks are correctly created ?

A chain of blocks : the blockchain

A publicly, immutable, and totally ordered log of transactions

- Why digital signatures are not sufficient ?
  - ✓ do not prevent double-spending attacks
- How can we securely link blocks ?
  - ✓ by linking them with cryptographic fingerprints
- How can we prevent transactions in a block from being manipulated ?
  - ✓ efficient cryptographic fingerprint of the transactions
- Who is allowed to create blocks ?

- Who is in charge of checking that blocks are correctly created ?

A chain of blocks : the blockchain

A publicly, immutable, and totally ordered log of transactions

- Why digital signatures are not sufficient ?
  - ✗ do not prevent double-spending attacks
- How can we securely link blocks ?
  - ✓ by linking them with cryptographic fingerprints
- How can we prevent transactions in a block from being manipulated ?
  - ✓ efficient cryptographic fingerprint of the transactions
- Who is allowed to create blocks ?
  - ✓ any node in the network capable of solving a given challenge
- Who is in charge of checking that blocks are correctly created ?

A chain of blocks : the blockchain

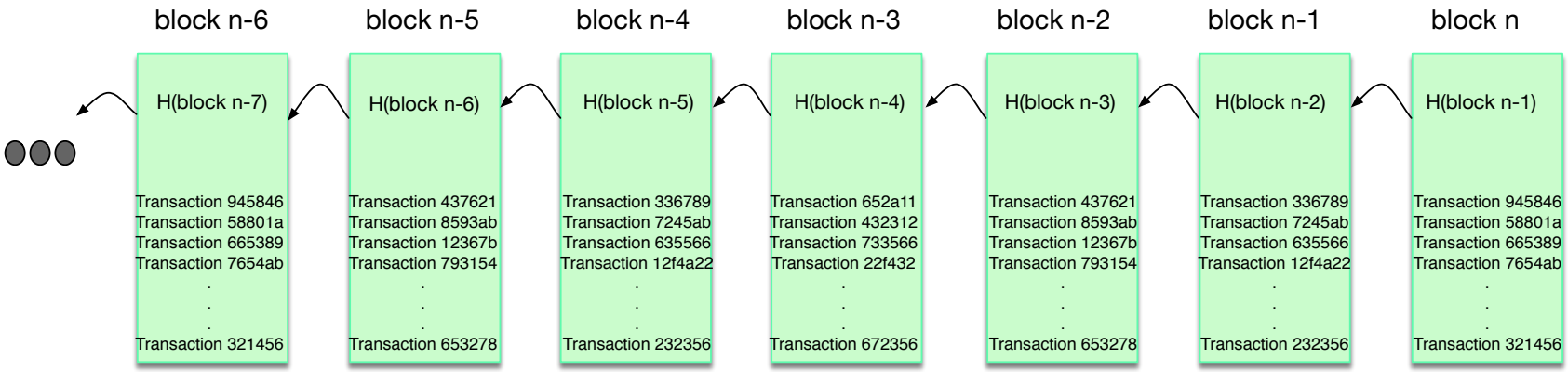
A publicly, immutable, and totally ordered log of transactions

- Why digital signatures are not sufficient ?
  - ✗ do not prevent double-spending attacks
- How can we securely link blocks ?
  - ✓ by linking them with cryptographic fingerprints
- How can we prevent transactions in a block from being manipulated ?
  - ✓ efficient cryptographic fingerprint of the transactions
- Who is allowed to create blocks ?
  - ✓ any node in the network capable of solving a given challenge
- Who is in charge of checking that blocks are correctly created ?
  - ✓ everyone ! !

How can we securely link blocks ?

Cryptographic Hash functions

- allows you to compute a fixed-size fingerprint from any type of data
- deterministic function
- efficiently computable
- practically impossible to invert (« one-way function »)

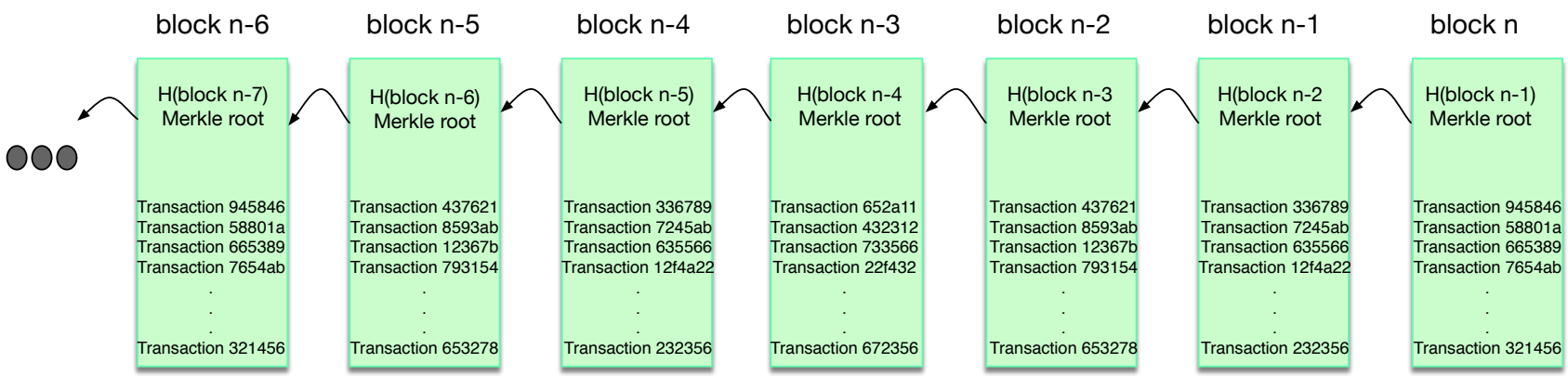


How can we prevent transactions of a block from being manipulated ?

Merkle tree

A Merkle tree [5] is a tree

- Leaves of the tree are data blocks
- Nodes are the cryptographic hashes of their children
- Root of tree is the fingerprint of the tree



Nakamoto Consensus protocol : Adversary model [6]

The Computational Threshold Adversary (CTA) model

- The adversary controls a minority of the total amount of computational power
- The CTA model is also called the permissionless model
  - There is no membership protocol
  - Incorporates some degree of synchrony

The Stake Threshold Adversary (STA) model

- The adversary controls a minority stake in some limited resource, i.e., the crypto-currency
- May allow to punish malicious parties via some stake deposit

6. Ittai Abraham and Dahlia Malkhi, « The blockchain consensus layer and BFT », The distributed Computing Column, 2017

Nakamoto Consensus protocol

The goal of the Consensus Nakamoto protocol is to implement a blockchain replication protocol[7]

- (Uniqueness) There is a unique chain of blocks (i.e., the blockchain)
- (Liveness) The blockchain is constantly growing
- (Safety) The deeper a block is buried in the blockchain the harder it is to abort it
- (Fairness) The proportion of blocks of non-faulty parties in the blockchain is proportional to their computation power

7. J. Garay and A. Kiayias, The Bitcoin Backbone Protocol : Analysis and Application, Eurocrypt 2015

Nakamoto Consensus protocol

Two ingredients : Leader Election Oracle + Heavier Fork Strategy

1. A leader election oracle implemented by the PoW mechanism
   - Each party is elected independently
   - The probability of electing each party is proportional to its relative computational power

$\rightarrow$ Synchronize the creation of blocks

$\rightarrow$ Prevent Sybil attacks

$\rightarrow$ Incentivize correct behavior through currency creation

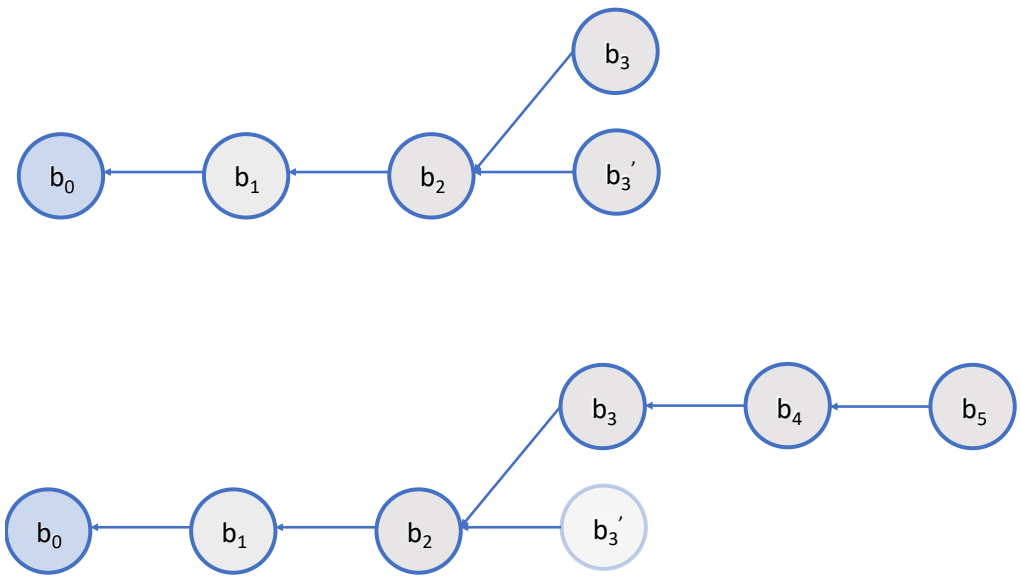
Nakamoto Consensus protocol

Two ingredients : Leader Election Oracle + Heavier Fork Strategy

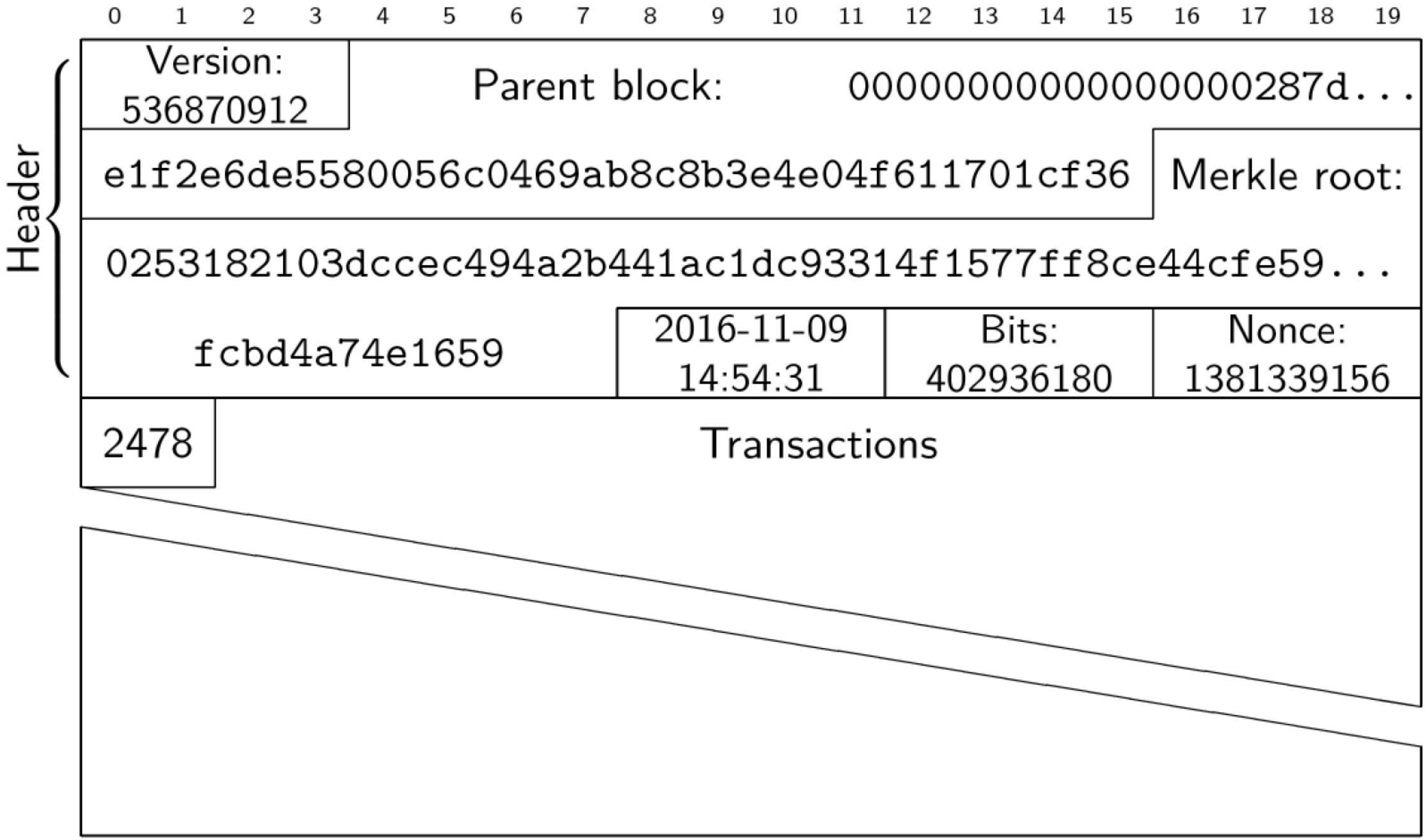
2 The simple and local Heavier Fork Strategy
- Select the chain which required the greatest among of work
- Random nature of the PoW + Non-faulty leaders always use the chain which required the greatest among of work

→ One chain will be extended more than the others

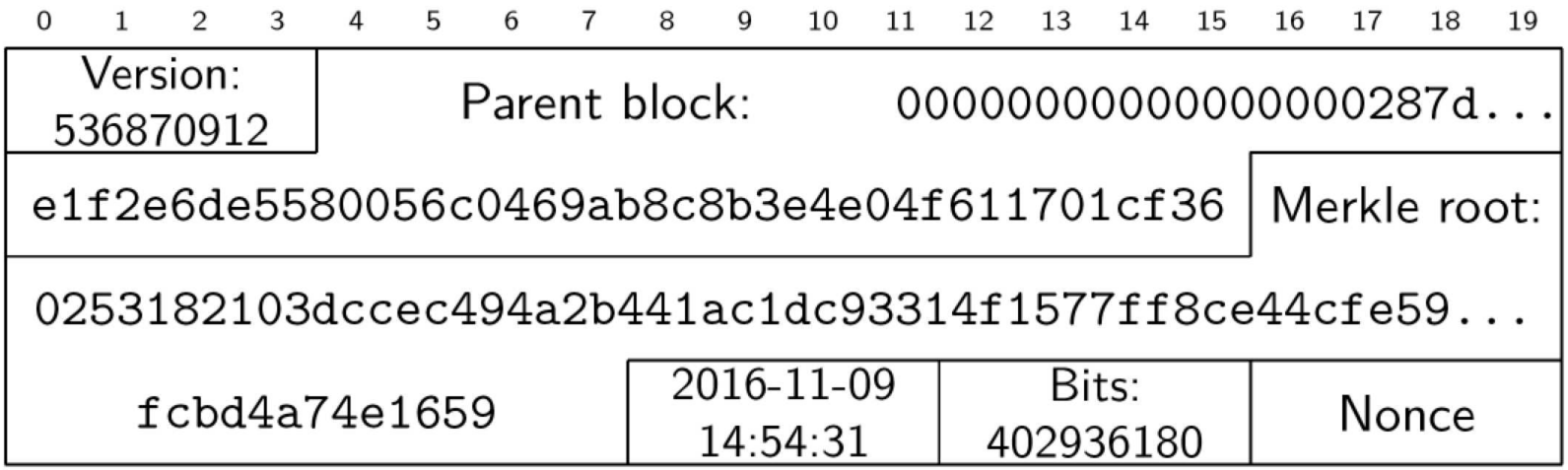
→ This is the blockchain !



Blockchain Proof-of-Work



Blockchain Proof-of-Work

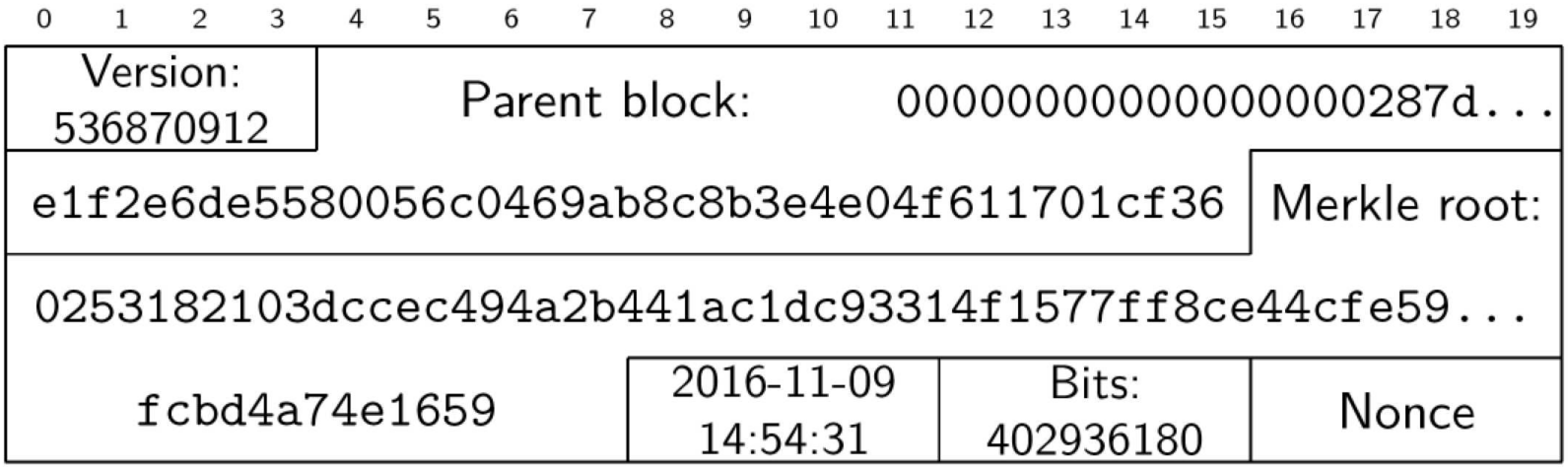


Goal: $\mathrm{SHA256} \circ \mathrm{SHA256}(\mathrm{header}) \leqslant \mathrm{target}$

Every 2016 blocks:

$$\mathrm{target} \leftarrow \mathrm{target} * \max(\frac{1}{4}, \min(4, \frac{\mathrm{real}}{\mathrm{expected}}))$$

- Comes down to compute a hash partial inversion
  - find pre-image for a partially specified hash function output



Goal: $\mathrm{SHA256} \circ \mathrm{SHA256}(\mathrm{header}) \leqslant \mathrm{target}$

Every 2016 blocks:

$$\mathrm{target} \leftarrow \mathrm{target} * \max(\frac{1}{4}, \min(4, \frac{\mathrm{real}}{\mathrm{expected}}))$$

Each party tries to provide a pow, and the proba p that one try is successful is

$$p = D/2^{\kappa},$$

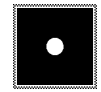
with $\{0,1\}^{\kappa}$ the range of $H(.)$, and D the difficulty.

Proof-of-Work

string=HelloWorld !, nonce=0, difficulty=000

Proof-of-Work

string=HelloWorld !, nonce=0, difficulty=000



```
HelloWorld!0:3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb6
```

Proof-of-Work

string=HelloWorld !, nonce=1, difficulty=000

```
HelloWorld!0:3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb6
HelloWorld!1:b5690c48c2d0a09481186aaa99e4e090901ff2ac4d572e6706dfd30eef
```

Proof-of-Work

string=HelloWorld !, nonce=2, difficulty=000

```
HelloWorld!0:3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb6
HelloWorld!1:b5690c48c2d0a09481186aaa99e4e090901ff2ac4d572e6706dfd30eef
HelloWorld!2:5b6fd9c27fcb54ca23404d9428f081b7c9280ba6370e33a6a20b16f40c
```

Proof-of-Work

string=HelloWorld !, nonce=3, difficulty=000

```
HelloWorld!0:3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb6
HelloWorld!1:b5690c48c2d0a09481186aaa99e4e090901ff2ac4d572e6706dfd30eef
HelloWorld!2:5b6fd9c27fcb54ca23404d9428f081b7c9280ba6370e33a6a20b16f40c
HelloWorld!3:9c5d769416aa0ca894abf22bd17bd30fbb6959291423ae1903a9f86a1f
....
```

Proof-of-Work

string=HelloWorld !, nonce=94, difficulty=000

```
HelloWorld!0:3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb6
HelloWorld!1:b5690c48c2d0a09481186aaa99e4e090901ff2ac4d572e6706dfd30eef
HelloWorld!2:5b6fd9c27fcb54ca23404d9428f081b7c9280ba6370e33a6a20b16f40c
HelloWorld!3:9c5d769416aa0ca894abf22bd17bd30fbb6959291423ae1903a9f86a1f
....
HelloWorld!94:7090a0e5d88cff635e42ea33fcd6091a058e9cdd58ab8cd5c21c1c704
```

Proof-of-Work

string=HelloWorld !, nonce=95, difficulty=000

```
HelloWorld!0:3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb6
HelloWorld!1:b5690c48c2d0a09481186aaa99e4e090901ff2ac4d572e6706dfd30eef
HelloWorld!2:5b6fd9c27fcb54ca23404d9428f081b7c9280ba6370e33a6a20b16f40c
HelloWorld!3:9c5d769416aa0ca894abf22bd17bd30fbb6959291423ae1903a9f86a1f
....
HelloWorld!94:7090a0e5d88cff635e42ea33fcd6091a058e9cdd58ab8cd5c21c1c704
HelloWorld!95:b74f3b2cf1061895f880a99d1d0249a8cedf223d3ed061150548aa621
HelloWorld!96:447ca2fa886965af084808d22116edde4383cbaa16fd1fbcf3db61421
```

Proof-of-Work

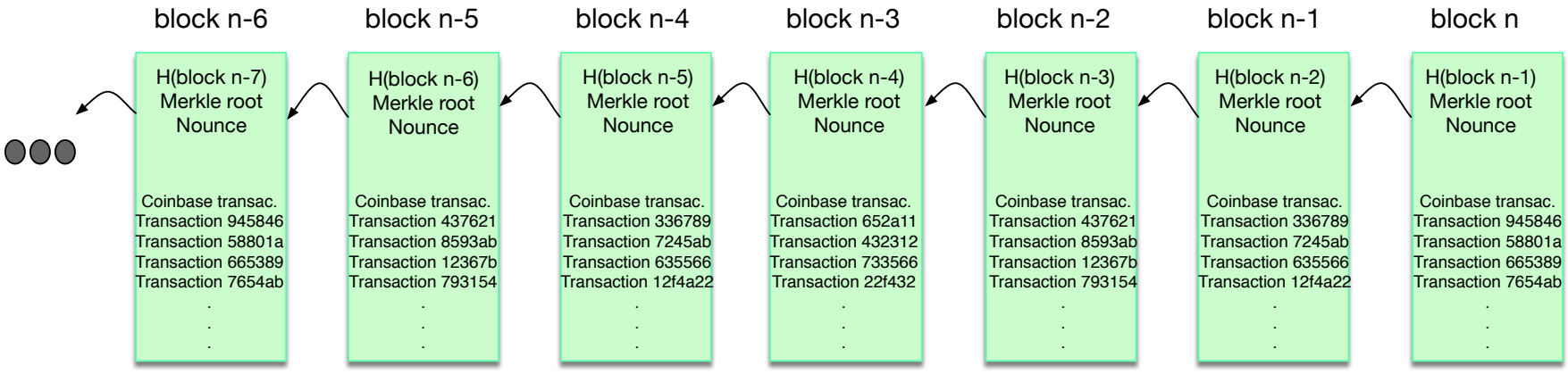
string=HelloWorld !, nonce=97, difficulty=000,

```
HelloWorld!0:3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb6
HelloWorld!1:b5690c48c2d0a09481186aaa99e4e090901ff2ac4d572e6706dfd30eef
HelloWorld!2:5b6fd9c27fcb54ca23404d9428f081b7c9280ba6370e33a6a20b16f40c
HelloWorld!3:9c5d769416aa0ca894abf22bd17bd30fbb6959291423ae1903a9f86a1f
....
HelloWorld!94:7090a0e5d88cff635e42ea33fcd6091a058e9cdd58ab8cd5c21c1c704
HelloWorld!95:b74f3b2cf1061895f880a99d1d0249a8cedf223d3ed061150548aa621
HelloWorld!96:447ca2fa886965af084808d22116edde4383cbaa16fd1fbcf3db61421
HelloWorld!97:000ba61ca46d1d317684925a0ef070e30193ff5fa6124aff76f513d96
```

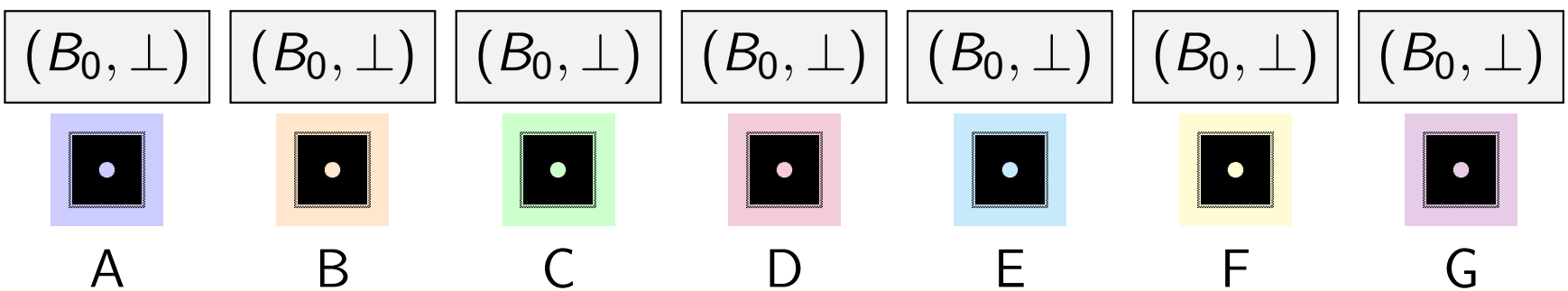
Blockchain Proof-of-Work

Properties

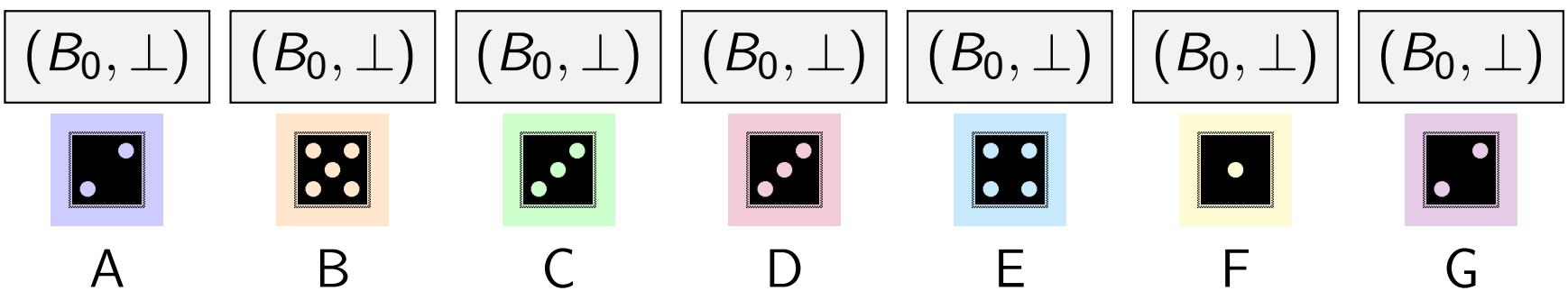
- Algorithmically computable
- Difficult to solve but quickly verified
- Difficulty proportional to computation power
  - average generation time : 10 minutes
  - proba that one try is successful is proportional to the difficulty and the range of $H(.)$
- Memoryless process



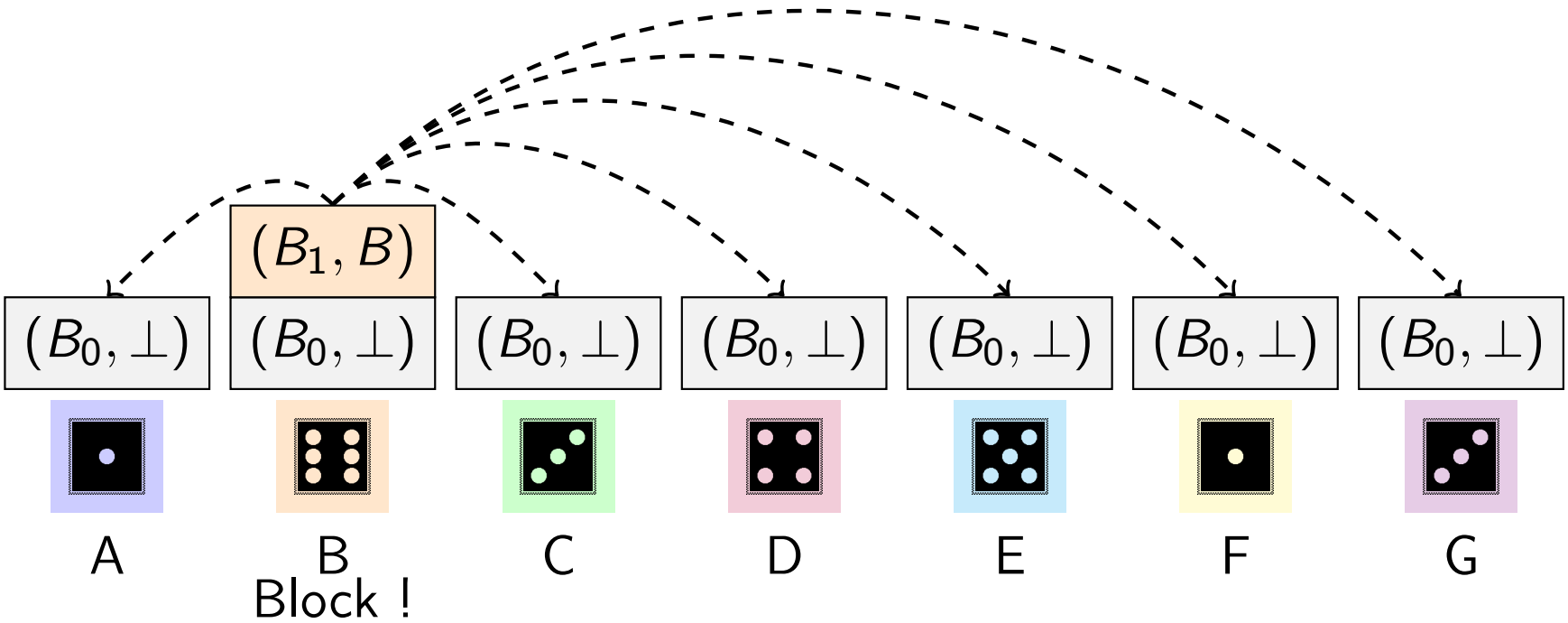
Blockchain construction



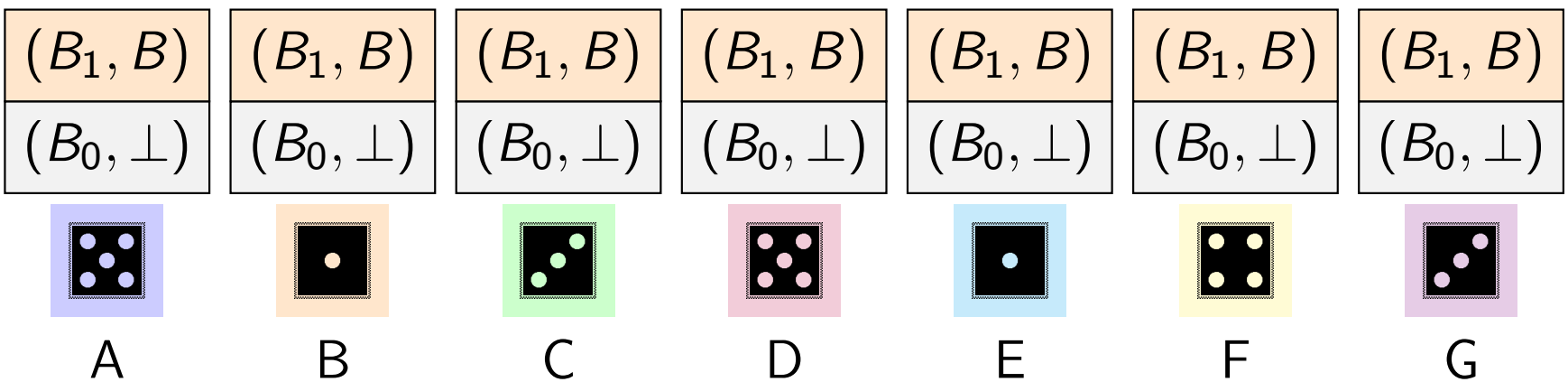
Blockchain construction



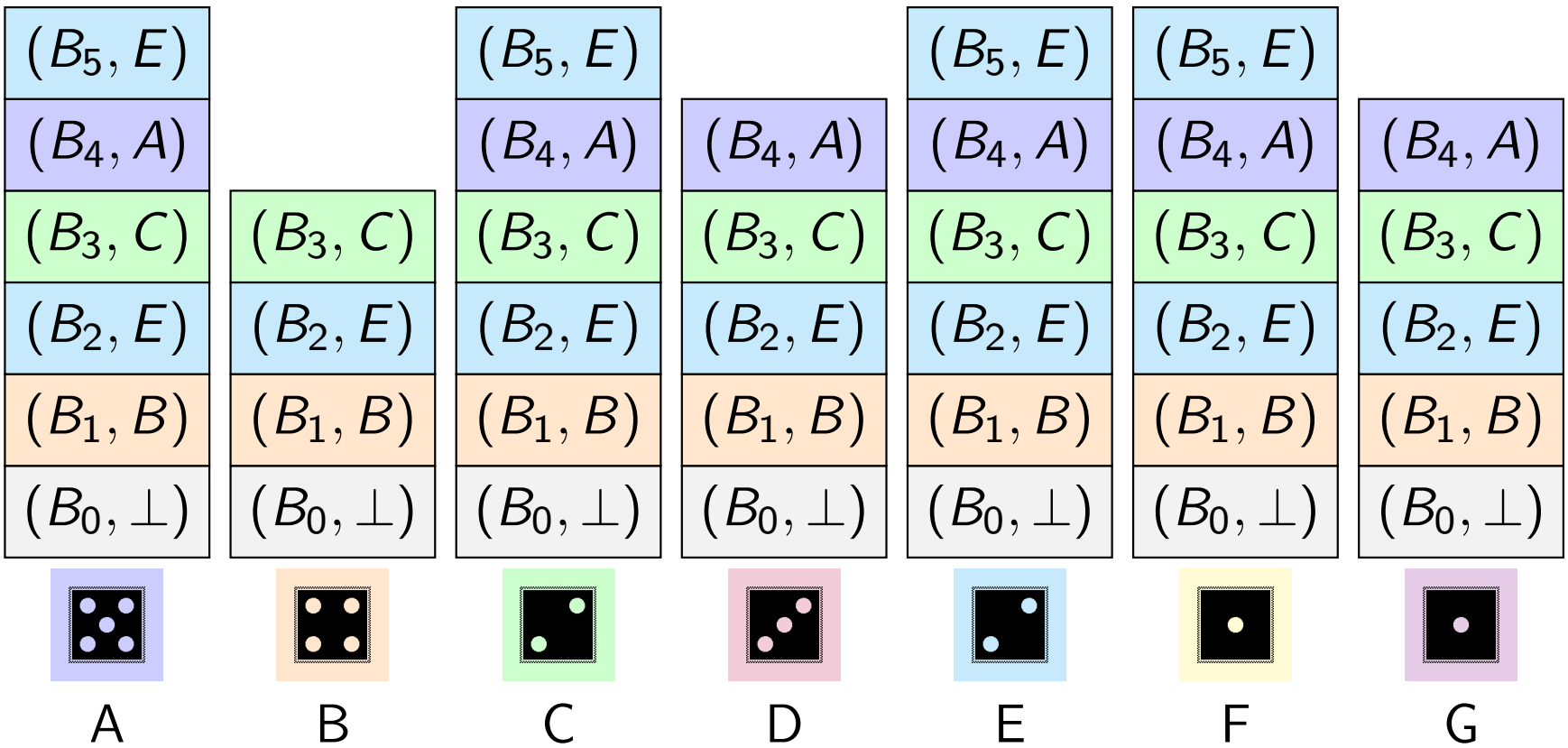
Blockchain construction



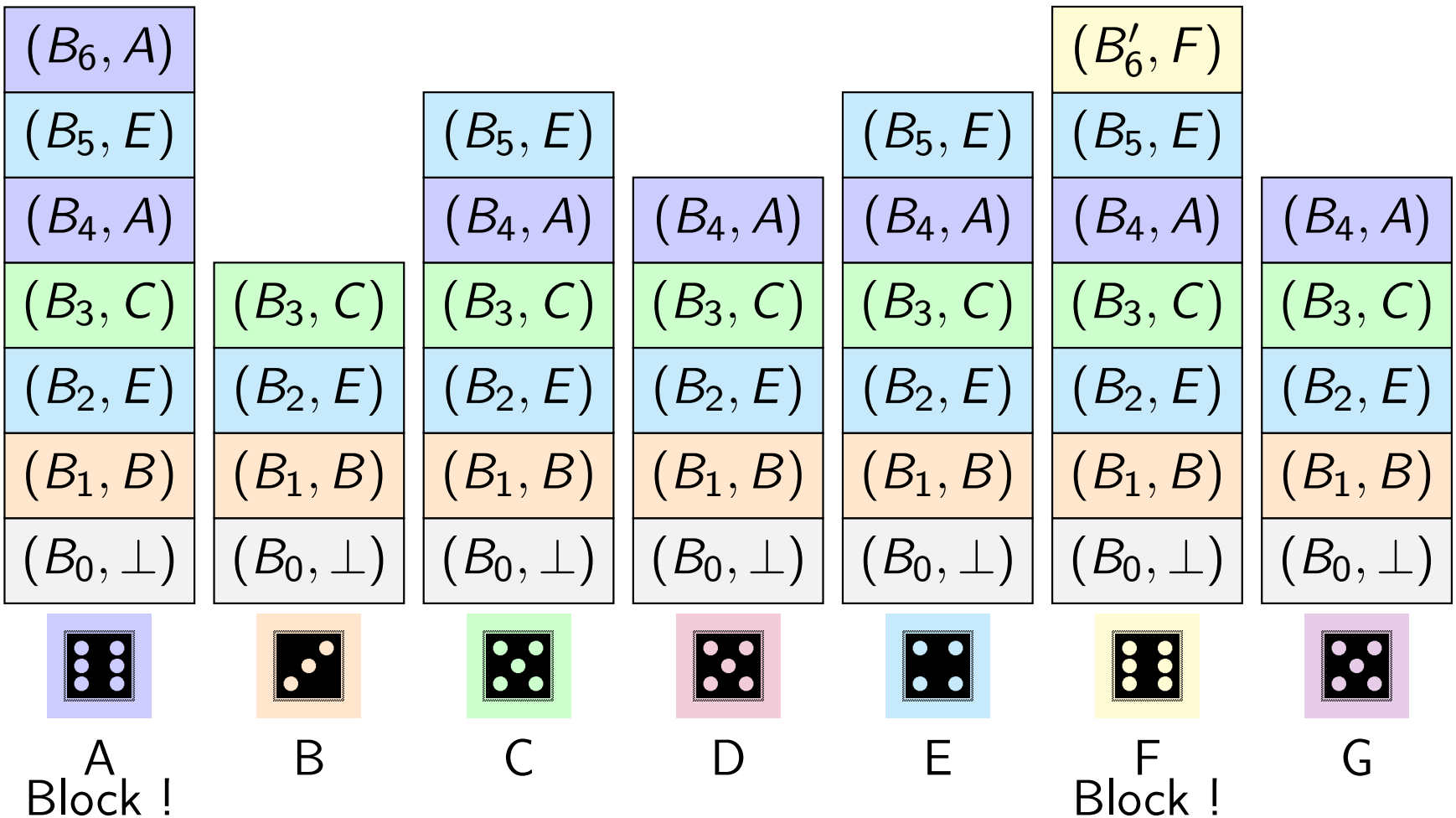
Blockchain construction



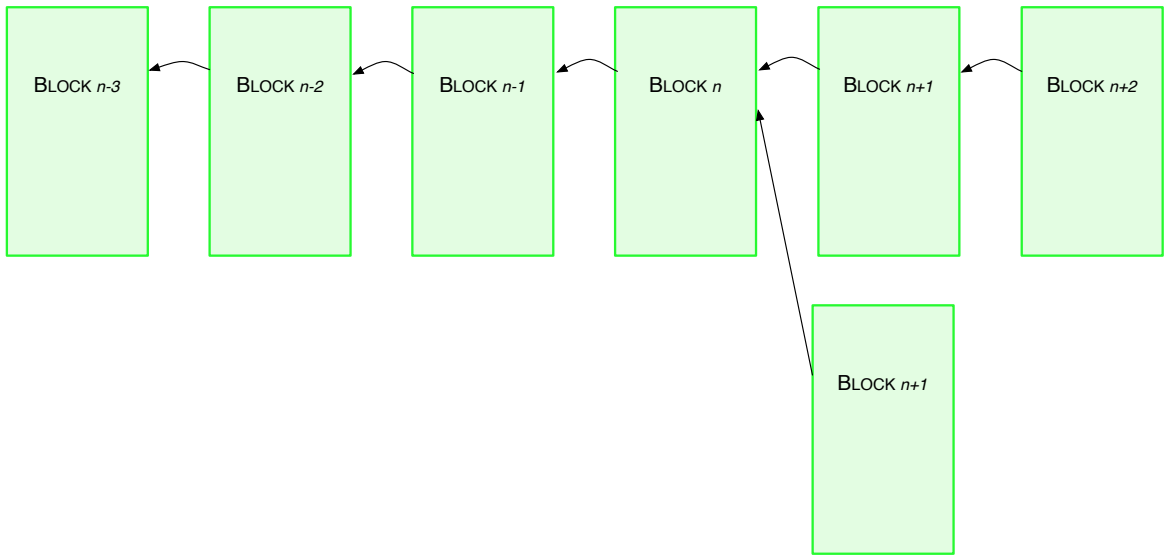
Blockchain construction



Conflict : Transient inconsistencies



Blockchain fork

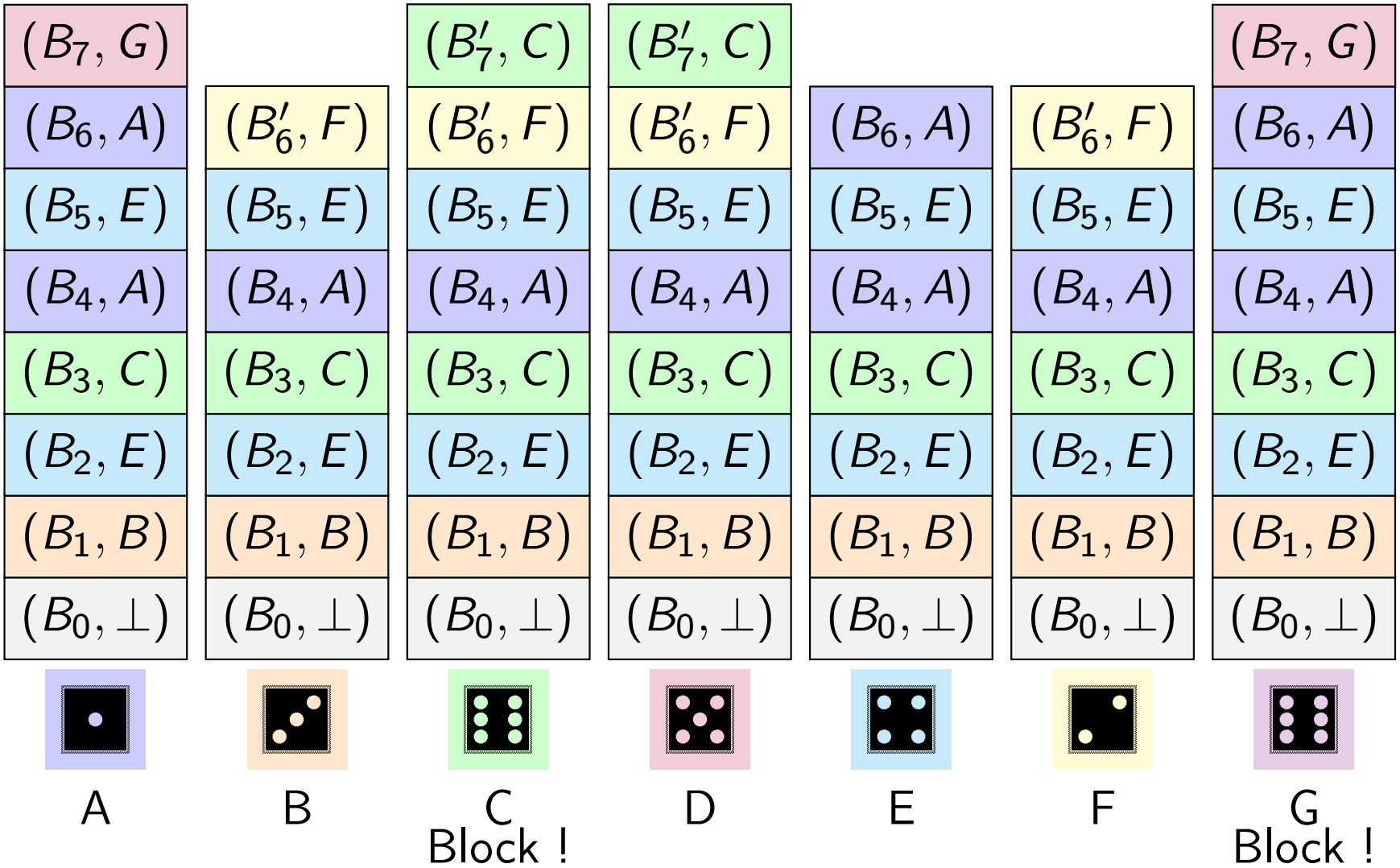


Nakamoto conflict chain rule

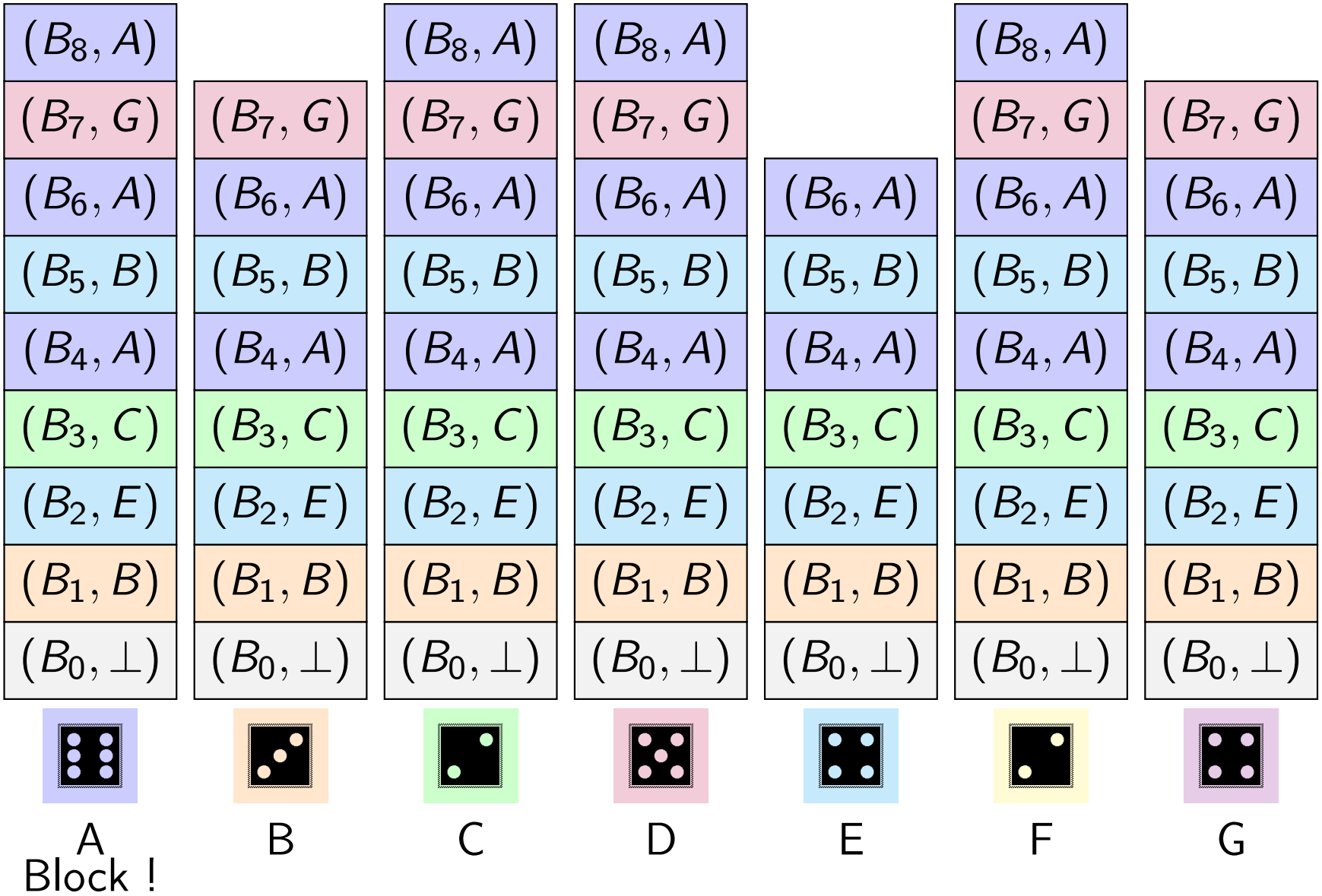
Mine on the longest branch !

- Proba. that a block is removed from the blockchain decreases exponentially with the number k of blocks appended to it.
- k = confirmation level
- When $k \geq 6$, this proba is very small

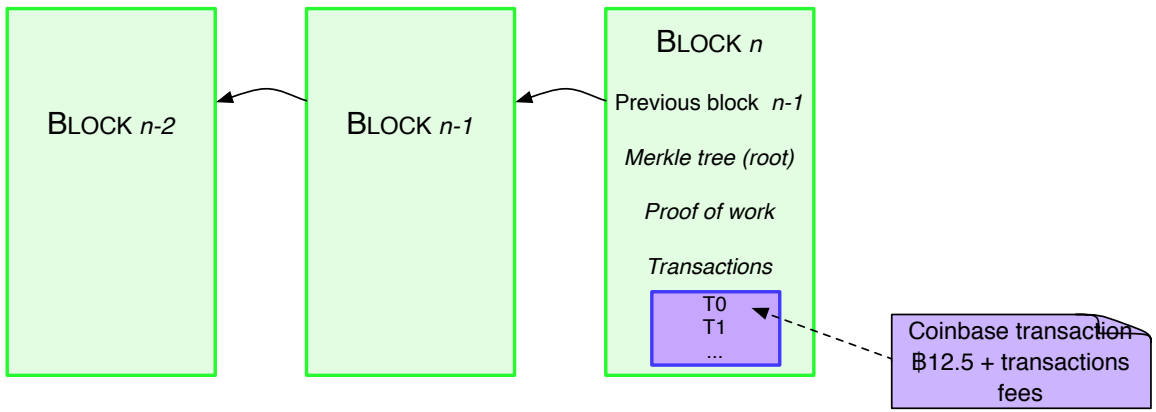
Conflict : Transient inconsistencies



Conflict : Transient inconsistencies



Abstraction 4 : Minting money to incentivize correct behavior



- Nodes are working (racing) to solve the computational puzzle
- This is in exchange for monetary rewards
  - Coinbase transaction = 12.5 bitcoins + transaction fees
  - Started at 50 bitcoins and is halved every $210,000$ blocks
- This incentivizes miners to only work on valid blocks
  - « valid block » : for miners, this means blocks for which the majority will accept to build upon

Summary

- The bitcoin system relies on a smart combination of abstractions to build a payment system and a crypto-currency
  - in a completely distributed system (i.e., no trusted third party)
  - large-scale and open to everyone
  - incentive to honestly behave (through financial awards)
- Performance issues : several transactions per second
- Environmental issues : proof-of-work

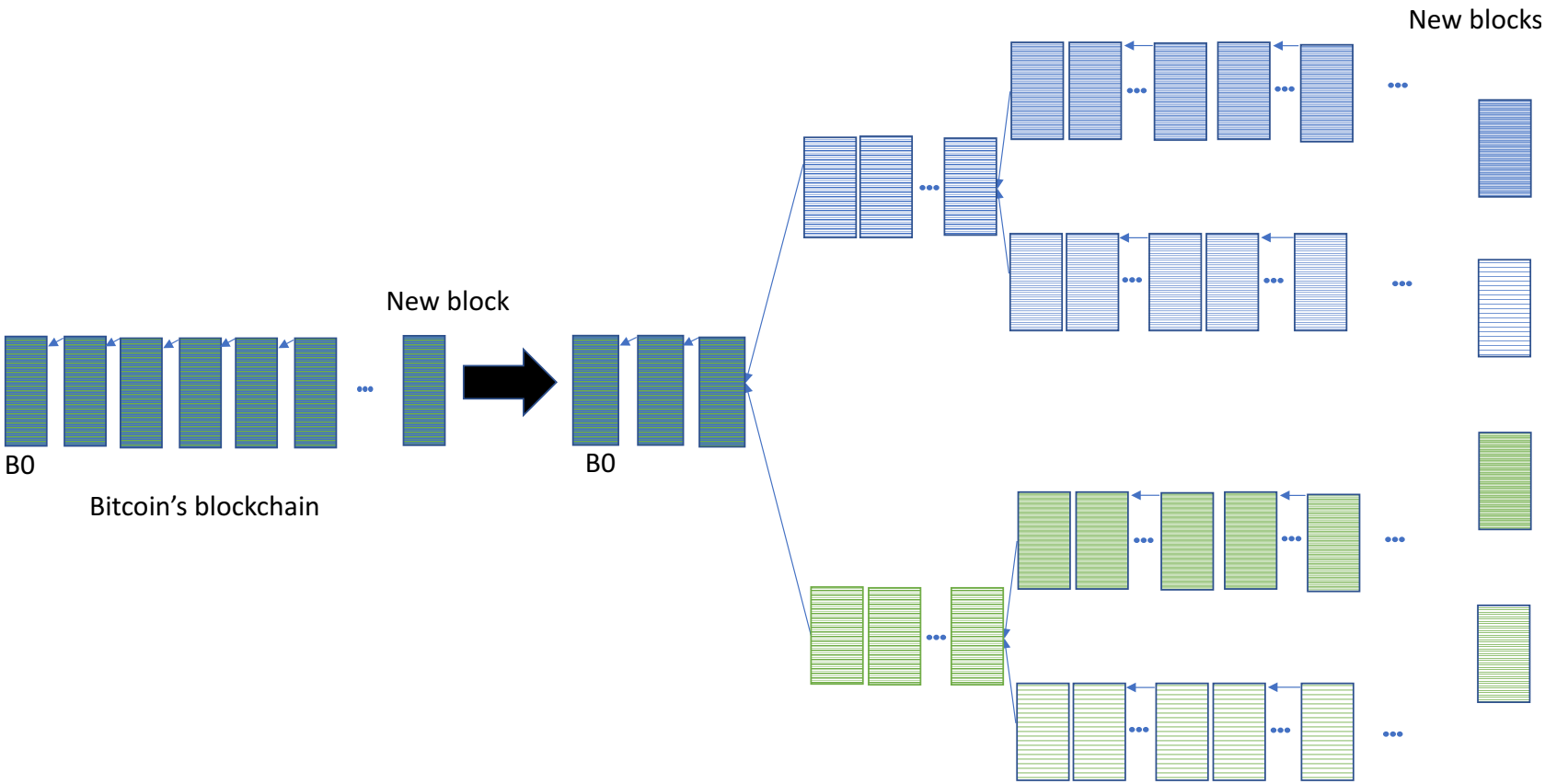
Electing a leader in a permissionless setting

- Proof-of-work
  - ✓ Works in practice and this is true since 2009
  - ✓ Security analysis
  - ✗ Environmental issues
  - ✗ Diminishing returns
  - ✗ Distinction between miners and crypto-currency holders (i.e., miners control what is inside blocks)
- Proof-of-work with useful computation, proof-of-space, proof-of-storage, ...
  - ✗ Physical resources
- Proof-of-stake
  - ✓ Abstract but limited resource : coin itself[8]
  - ✓ All the required information is already in the blockchain
  - ✗ Numerous attacks

8. A. Durand, E. Anceaume, R. Ludinard, « StakeCube : Combining sharding and proof-of-stake to build fork-free secure permissionless distributed ledgers », Int'l Conference on Networked Systems, 2019

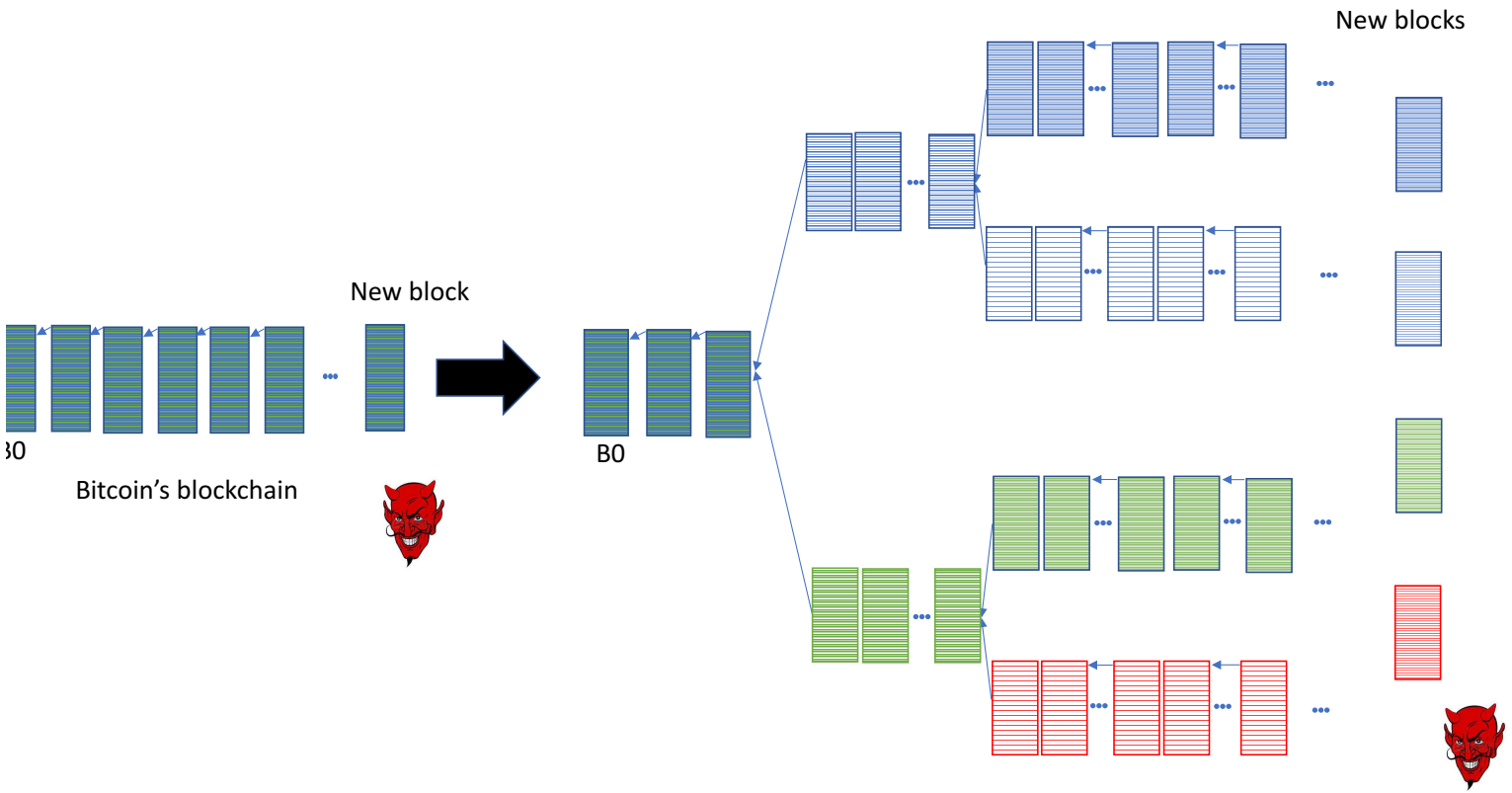
Self-Adapting to the actual number of transactions

1. A ledger with several parallel (but not conflicting) chains
2. Valid (and not conflicting) blocks should be mined in parallel



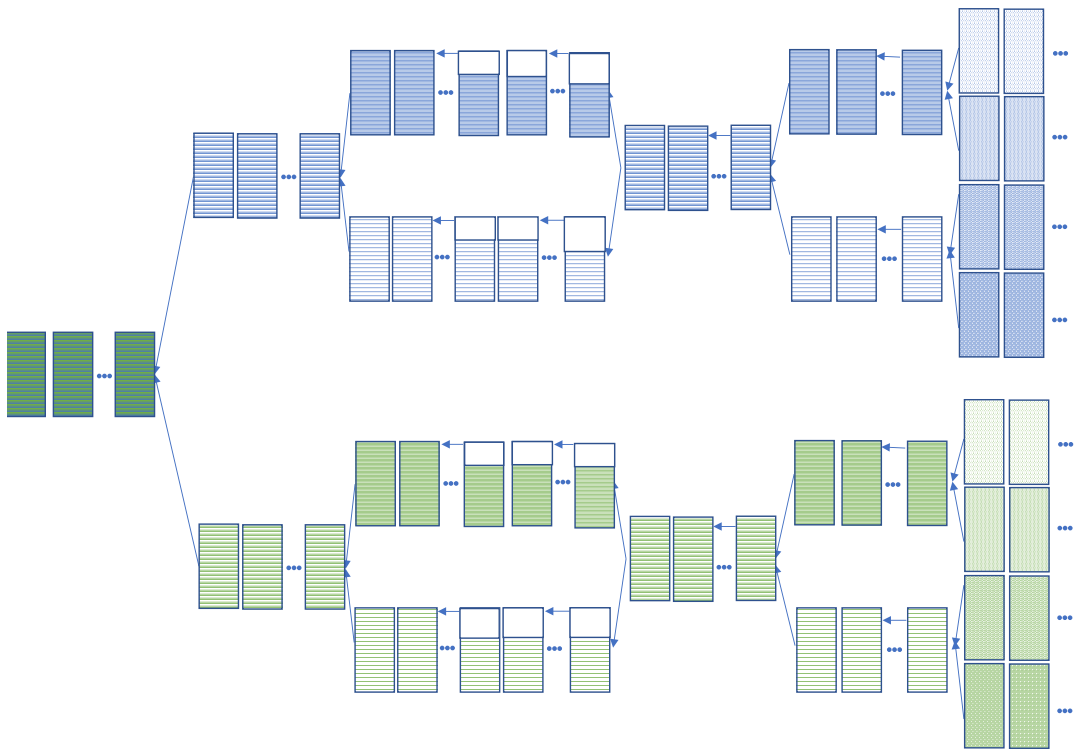
Self-Adapting to the actual number of transactions

1. A ledger with several parallel (but not conflicting) chains
2. Valid (and not conflicting) blocks should be mined in parallel
3. Miners should not be able to predict the chain of the ledger to which their blocks will be appended
   - Cannot devote their computational power to a specific chain



Self-Adapting to the actual number of transactions

1. A ledger with several parallel (but not conflicting) chains
2. Valid (and not conflicting) blocks should be mined in parallel
3. Miners should not be able to predict the chain of the ledger to which their blocks will be appended
4. Overloaded chains should dynamically split and underloaded ones should dynamically merge



⚠ All these features should be verifiable by anyone at any time[9]

9. E. Anceaume, A. Guellier, R. Ludinard, B. Sericola, Sycomore : a Permissionless Distributed Ledger that self-adapts to Transactions Demand, IEEE NCA, 2018

Permissionless and others

Other ways of building distributed ledgers

- Consortium blockchains
  - Assume a well-known set of entities capable of the read, write, and execute operations
  - Do not necessarily require to solve a challenge to write a new block
  - Rely on Byzantine agreement algorithms
- Private blockchains
  - Assume the presence of a trusted third party in charge of the read, write, and execute operations [10]
  - Do not need to solve a challenge to write a new block
  - Do not need Byzantine agreement algorithms

10. Haber and Stornetta, How to timestamp a digital document, J. of cryptology, 3(2), pp 99–111, 1991

Thank you for your attention
Any questions ?