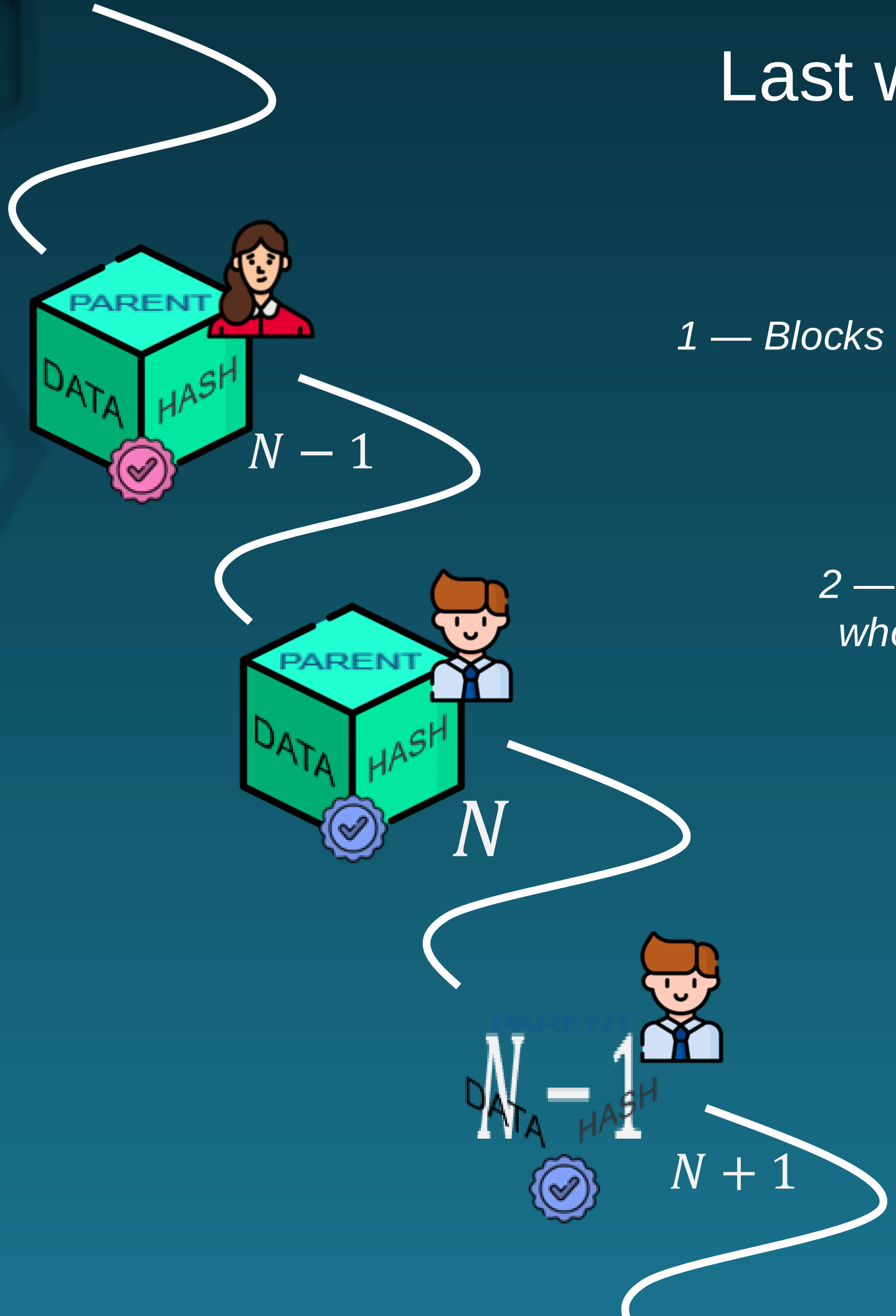
Blockchain and Applications

Chapter 3

Consensus algorithms

Last week



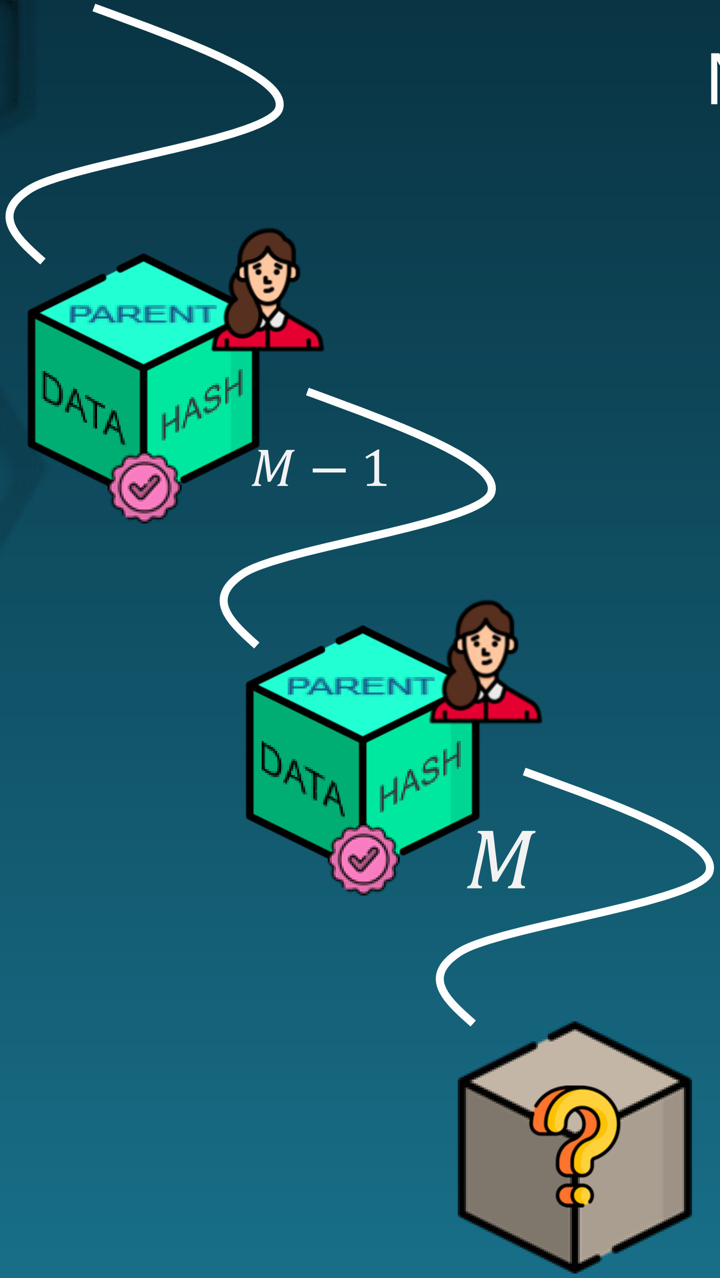
1 — Blocks are certificates that contain certificates

2 — If any certificate is tampered with, the whole blockchain (starting the block that changed) is corrupted

3 — All blocks following the altered block must be signed again

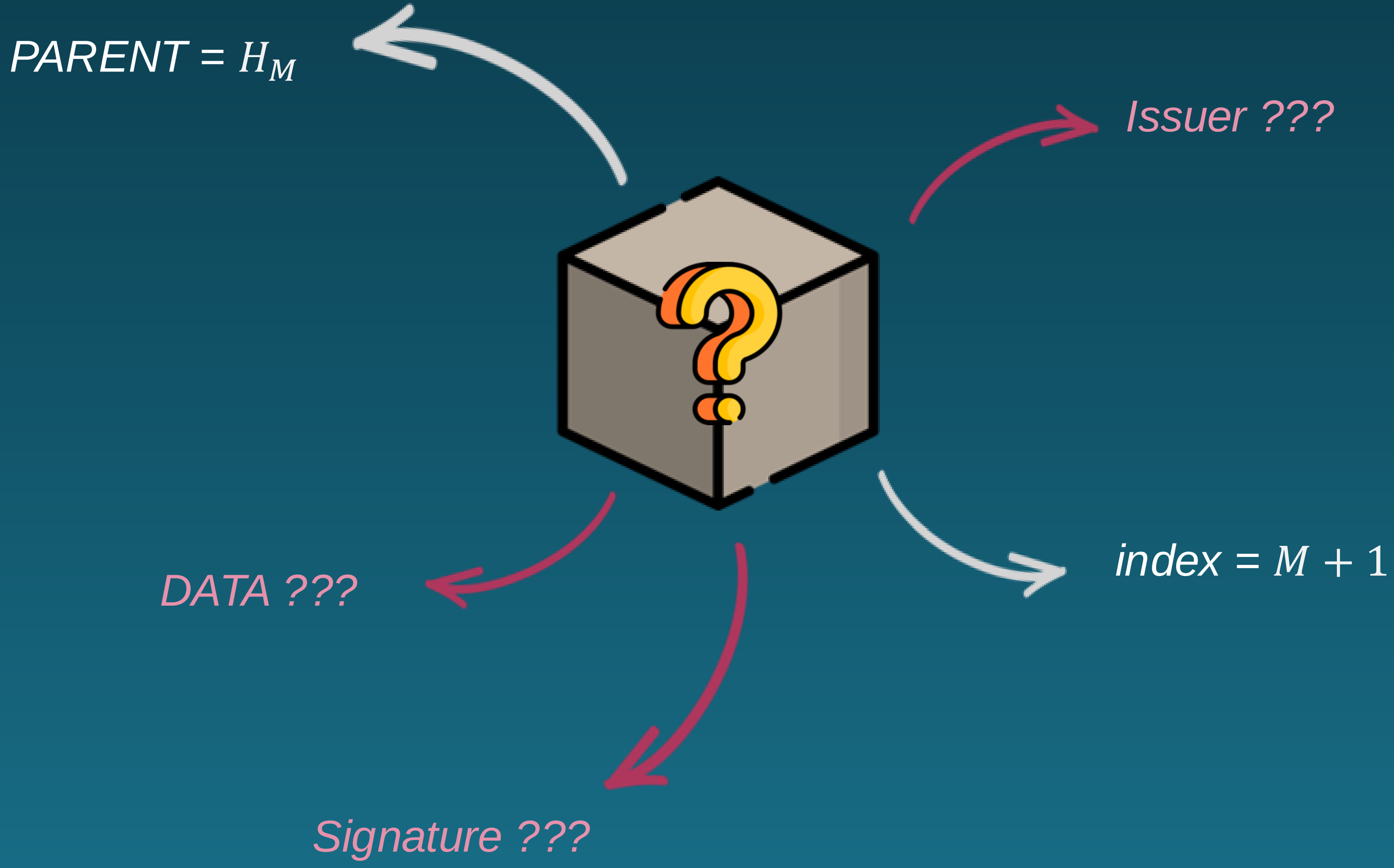
ULTIMATE SECURITY : Data inside a blockchain is IMMUTABLE

Next block ?



Question : How do we add blocks to the blockchain ?

Next block ?



PARENT = H_M

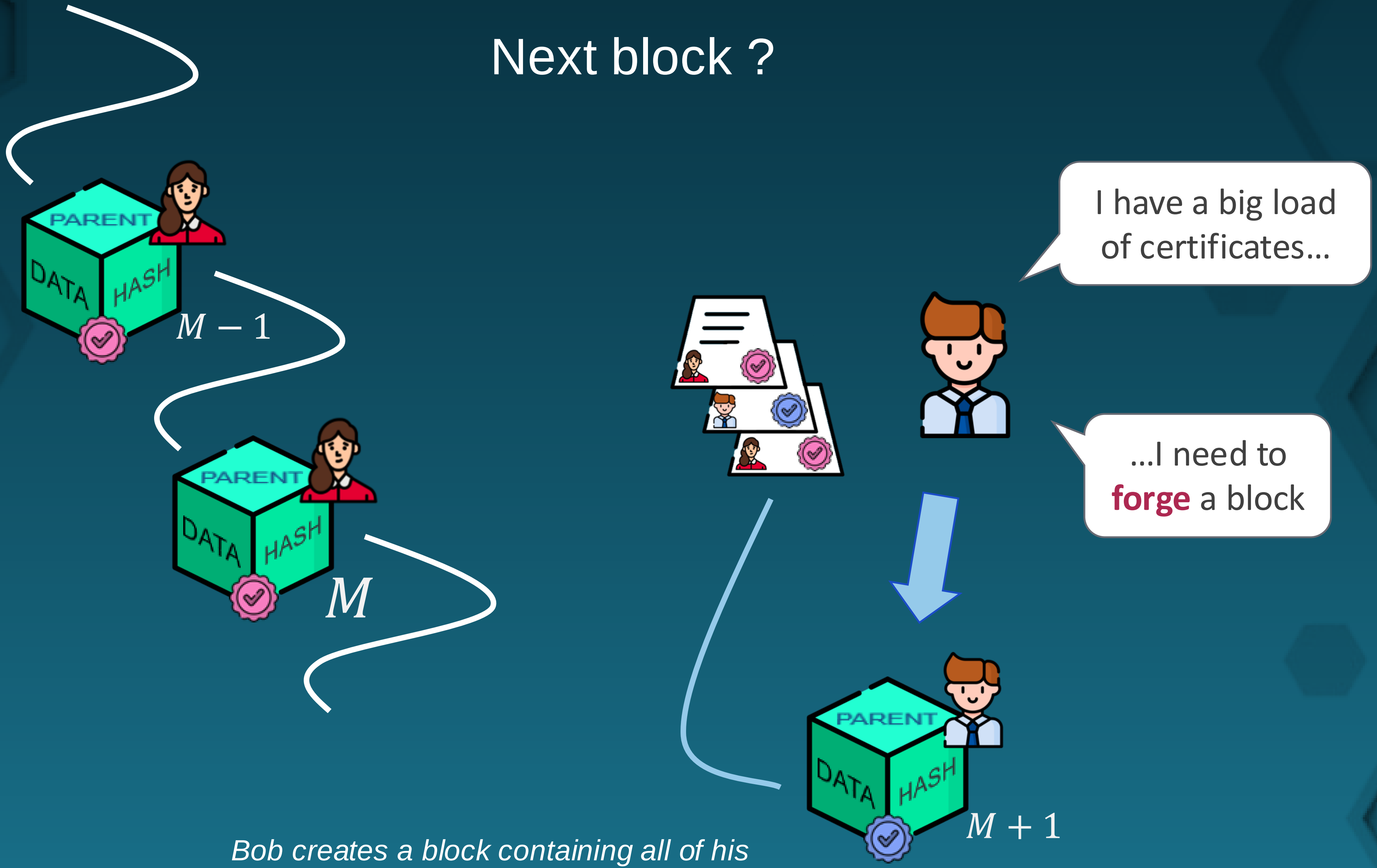
Issuer ???

DATA ???

index = $M + 1$

Signature ???

Next block ?

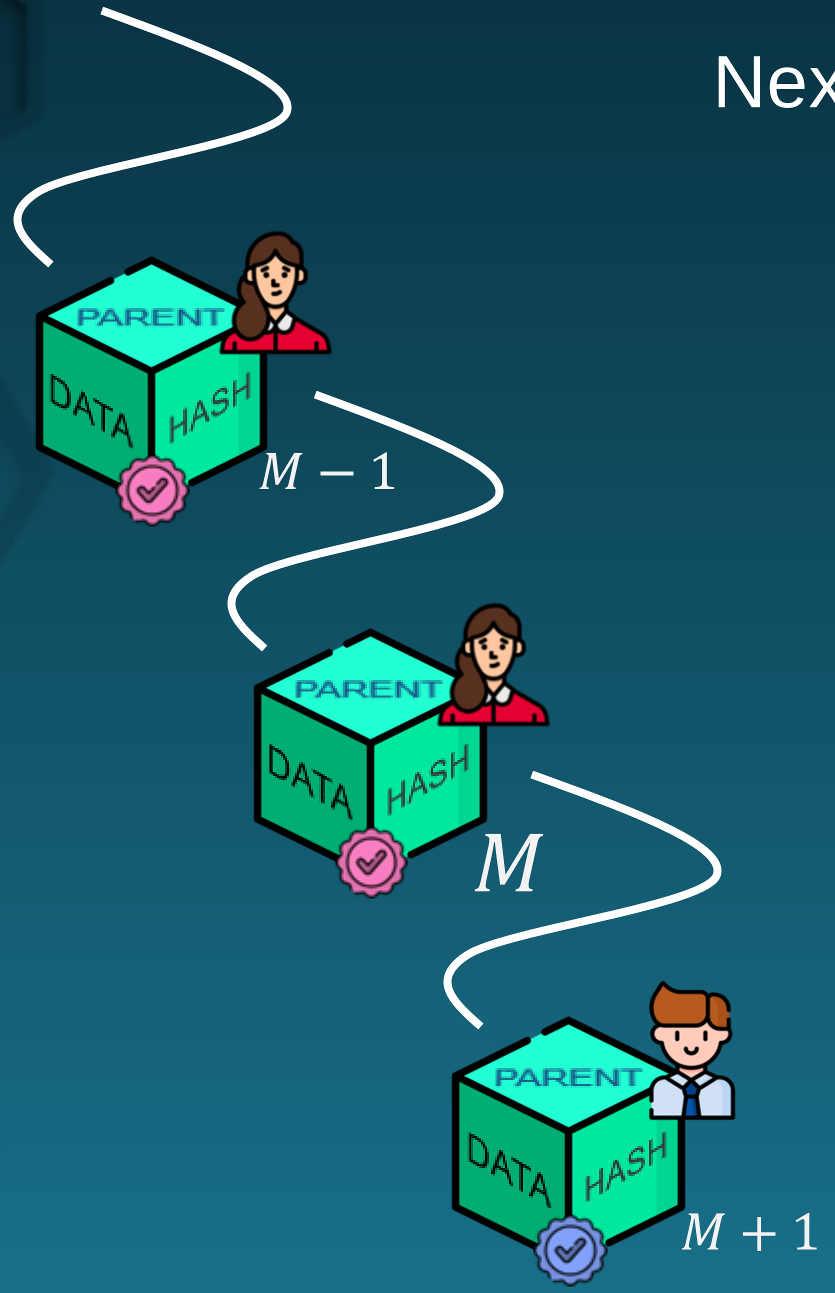


I have a big load of certificates…

…I need to **forge** a block

Bob creates a block containing all of his certificates, and adds it to the blockchain

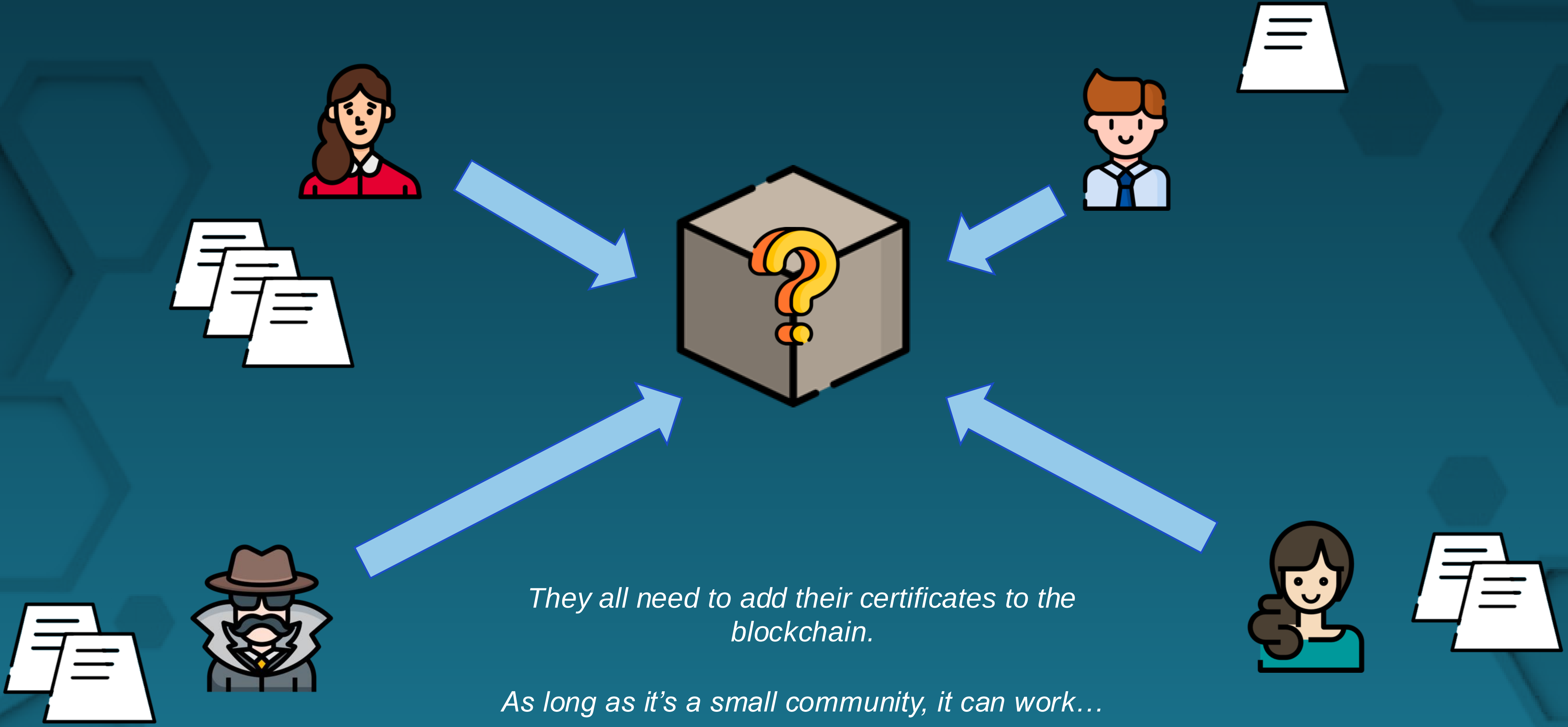
Next block ?



The blockchain is 100% valid !

… But is this a good system ?

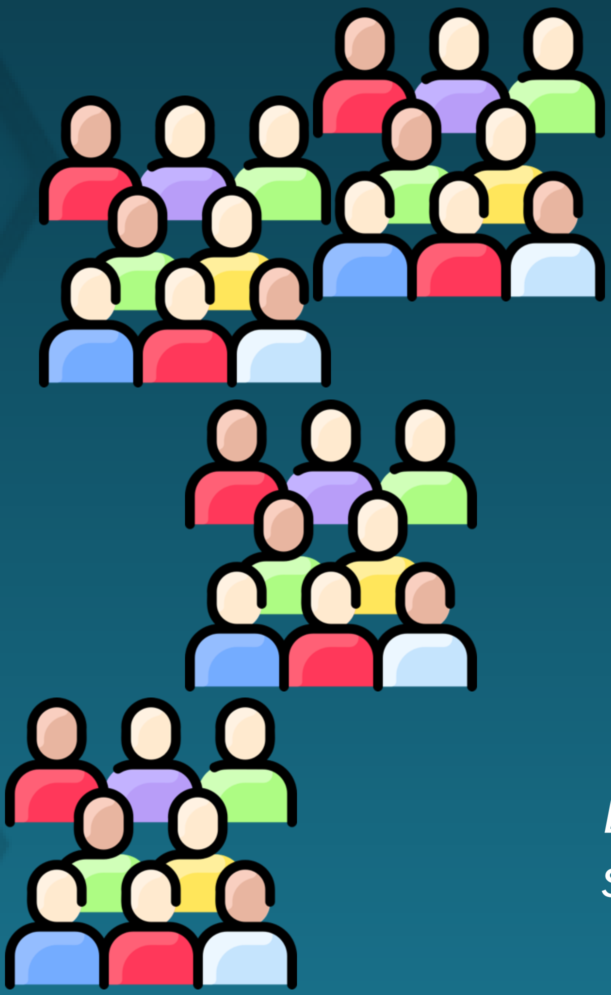
Problems — Anarchy

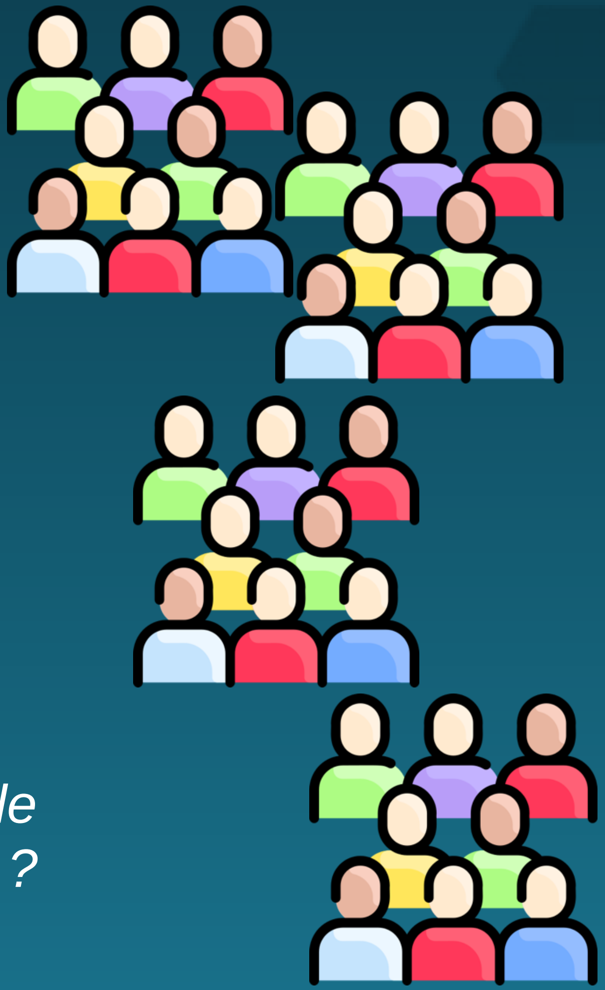


They all need to add their certificates to the blockchain.

As long as it's a small community, it can work…

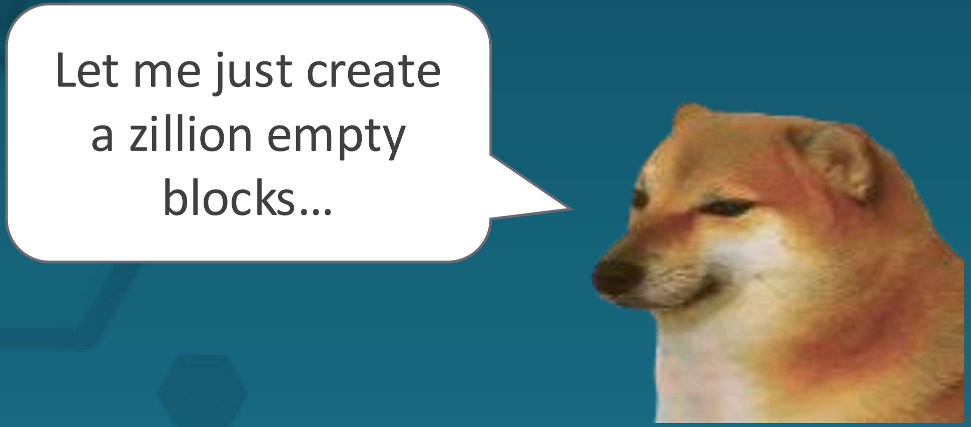
Problems — Anarchy

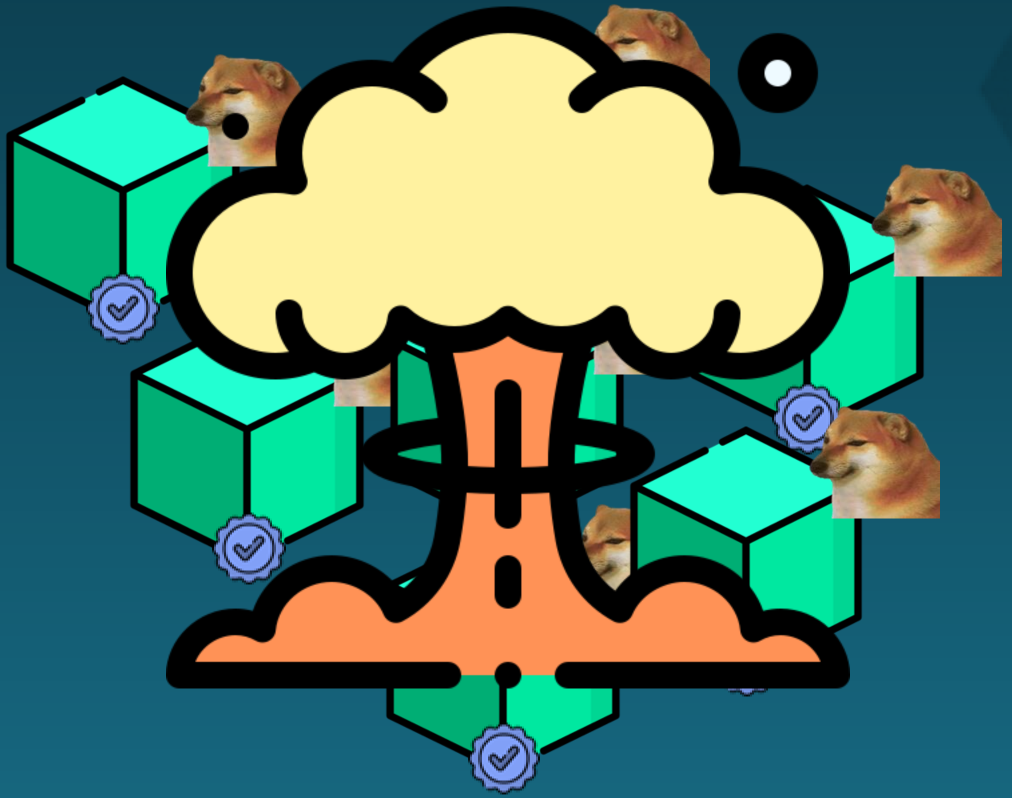




But what if millions of people share the same blockchain ?

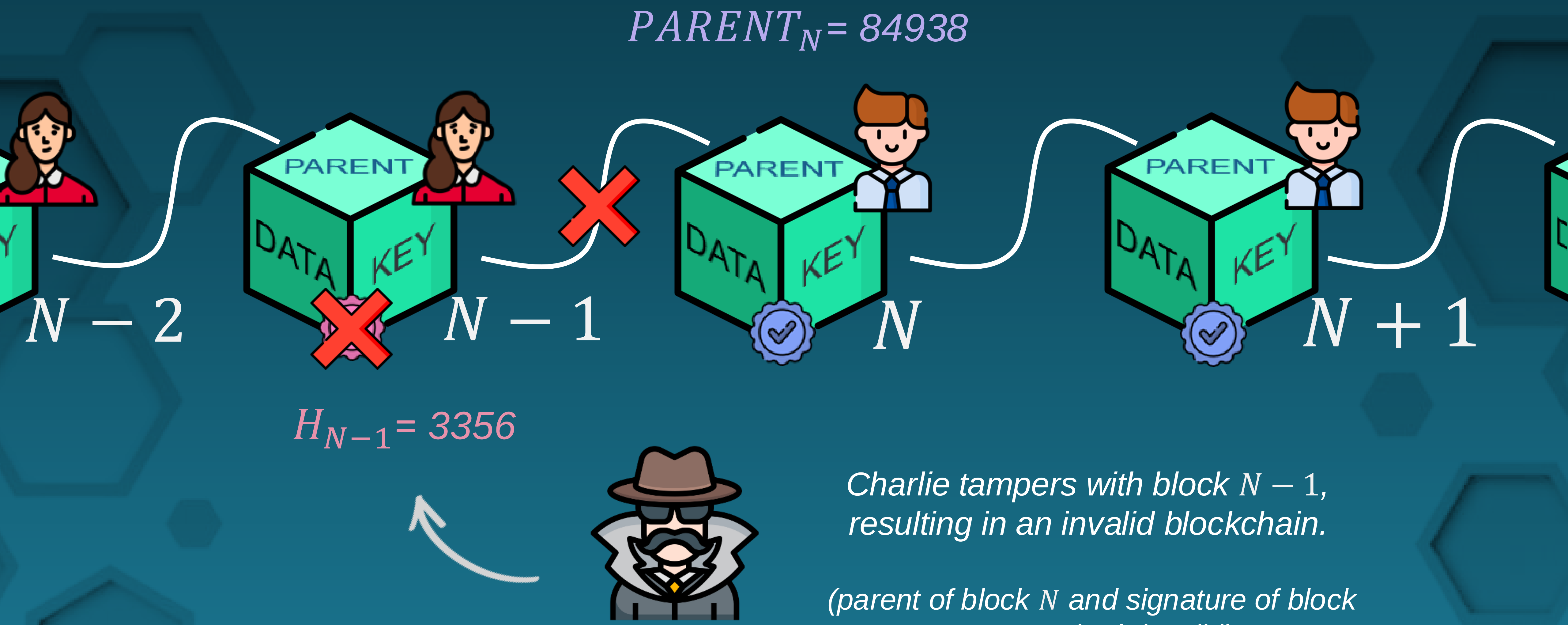
Problems — Nihilism





If anyone could just add zillions of blocks, the server would crash…

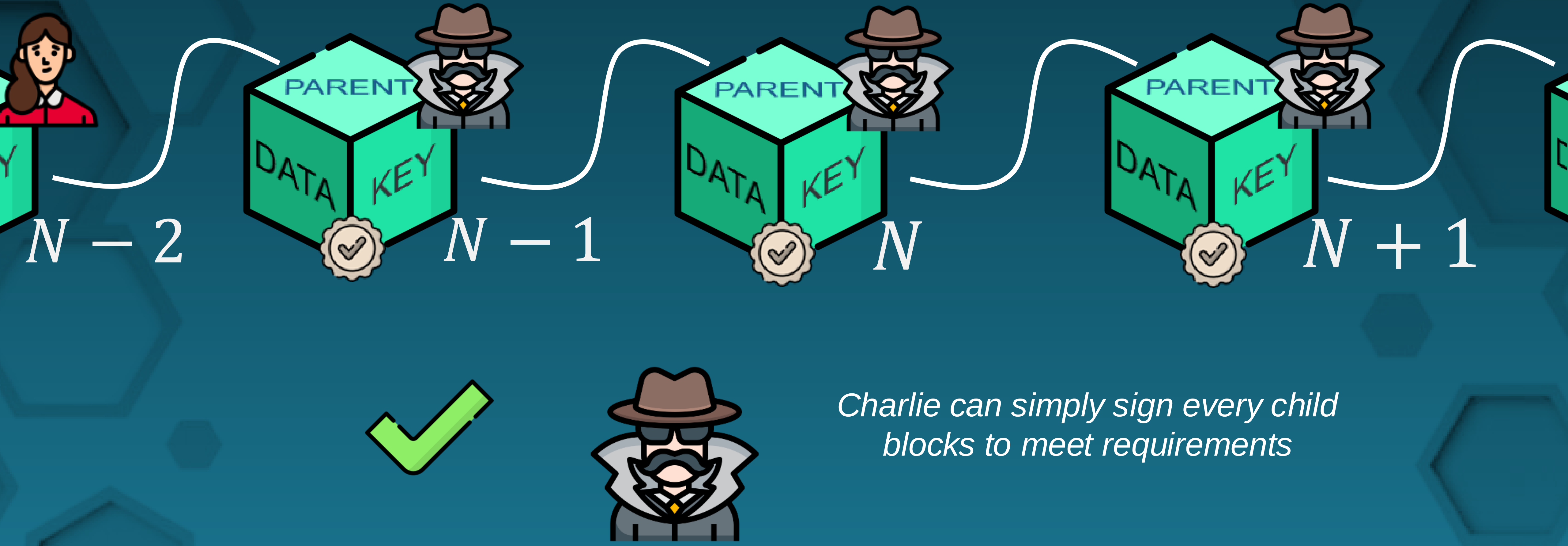
Problems — Mutability



Charlie tampers with block $N-1$, resulting in an invalid blockchain.

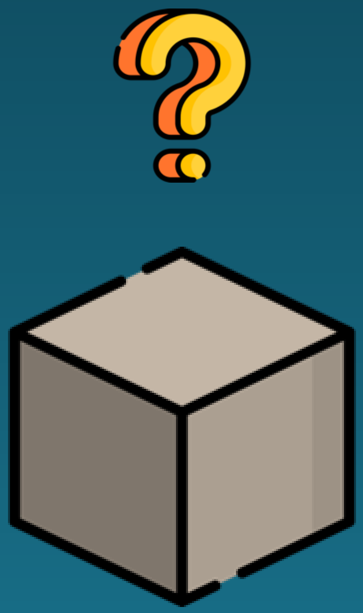
(parent of block N and signature of block $N-1$ are both invalid)

Problems — Mutability



Charlie can simply sign every child blocks to meet requirements

Forger control



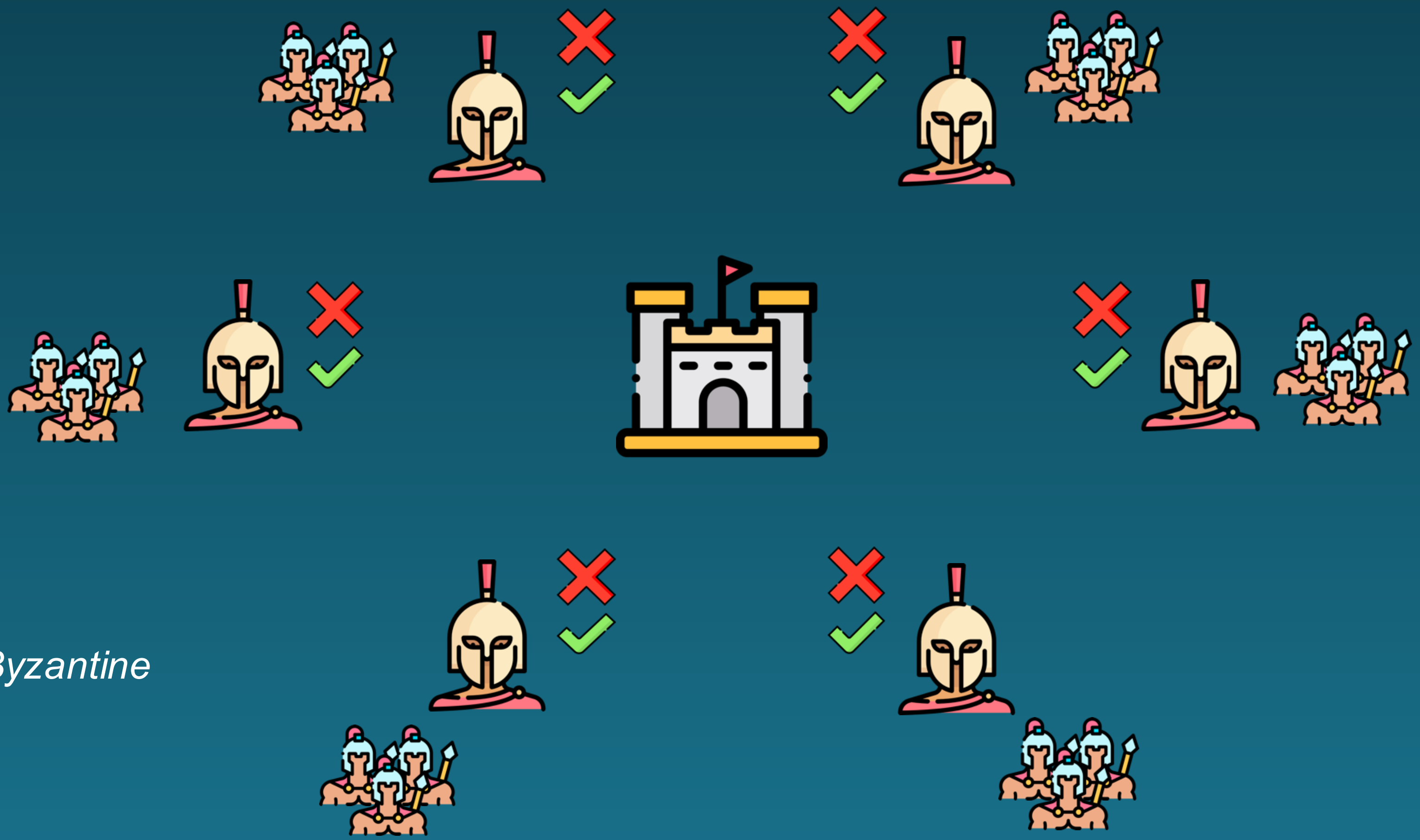
We need to control who the next forger is…

Problems — Consensus

But how can we have everyone agree on a forger when everyone wants to be one ?...

We need a consensus algorithm

Problems — Consensus



Isn't it just like the Byzantine generals ?

1st — Proof-of-Work

Satoshi Nakamoto

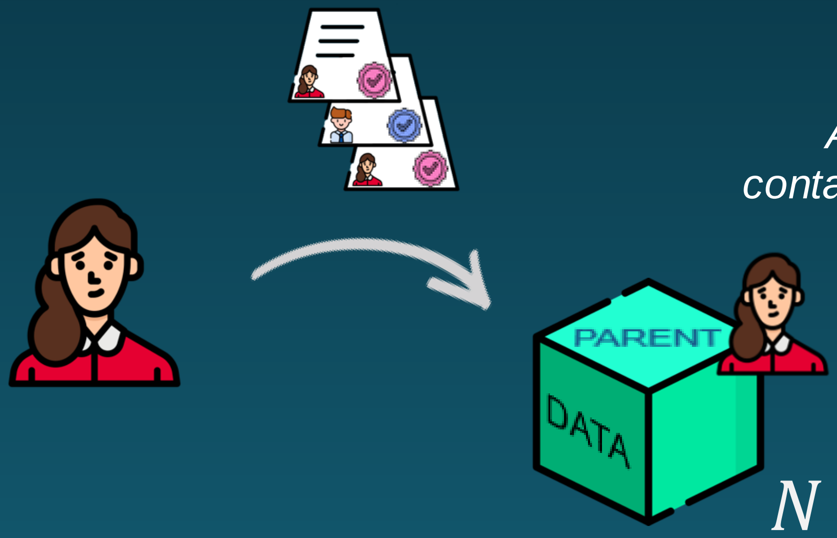
Bitcoin mining

—

Proof-of-work

2008

Proof-of-Work



Alice prepares a block containing all of her certificates

Block parent

Block index

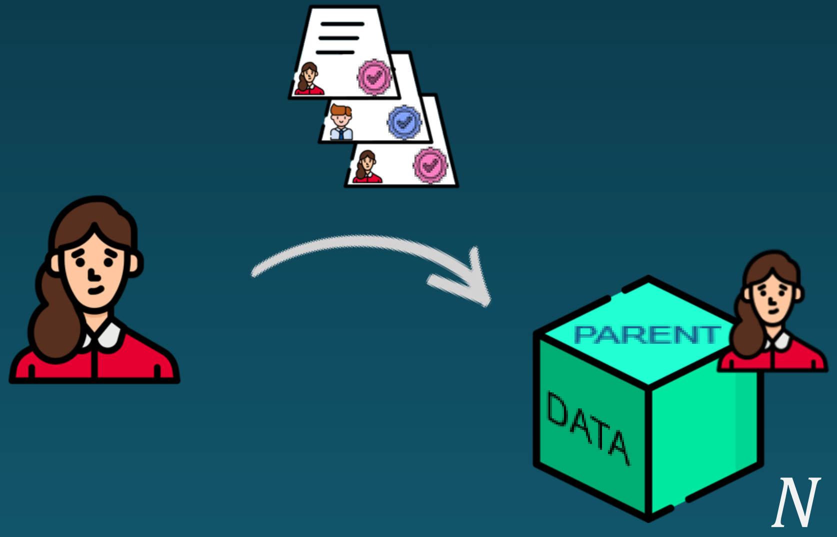
Certificates

Issuer (Alice)

$$H_N = ab566f4e0 \ldots$$

Her block already has a hash

Proof-of-Work



Block parent

Block index

Certificates

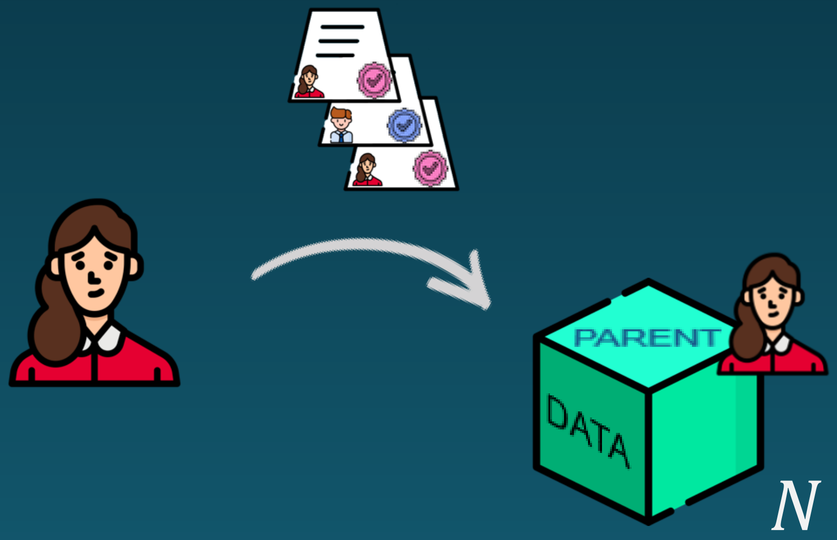
Issuer (Alice)

Nonce : 4327

$$H_N = 59d005313 \ldots$$

By adding a useless data (nonce), the hash of the block changes

Proof-of-Work



Block parent

Block index

Certificates

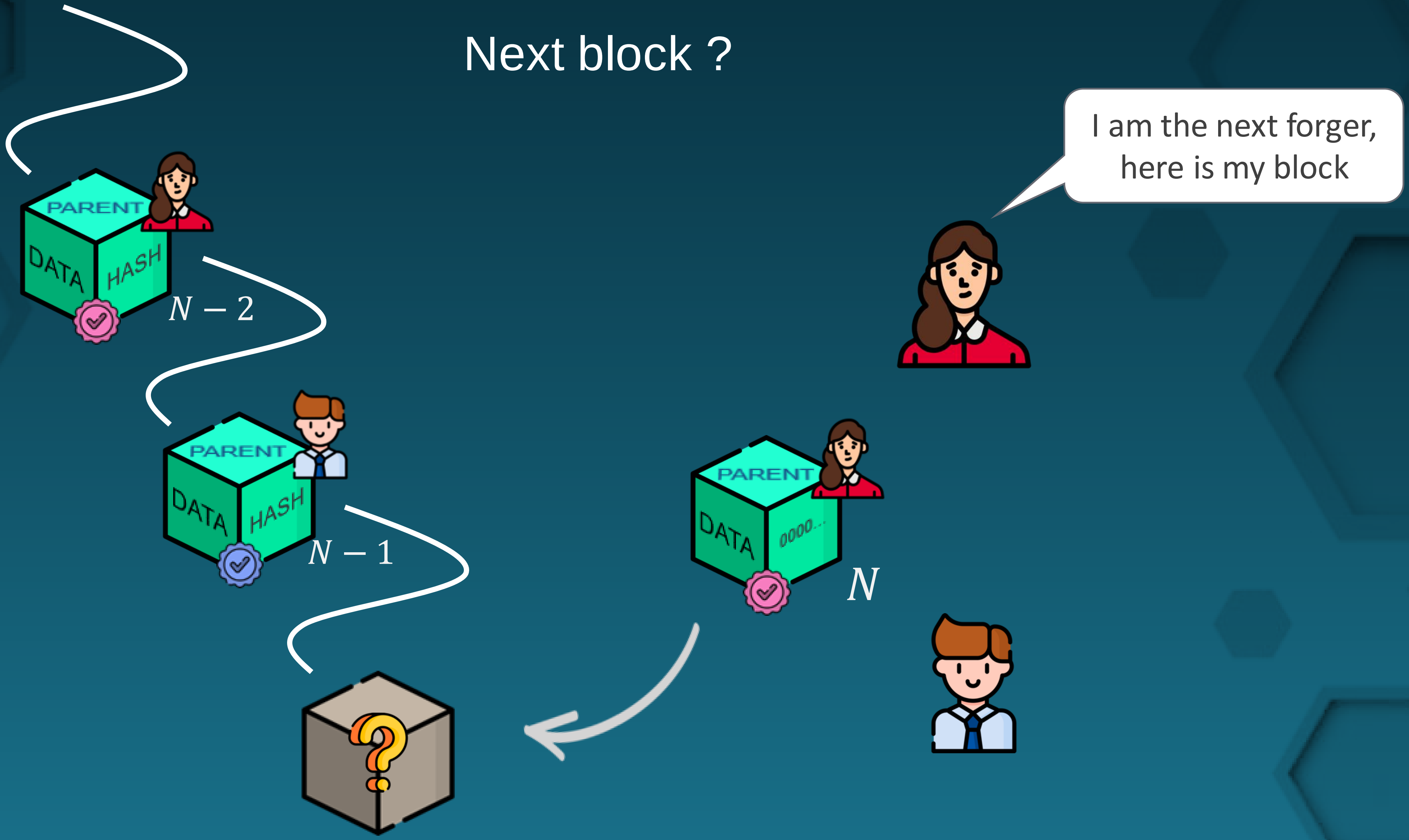
Issuer (Alice)

Nonce : 99706

$$H_N = 0000efc67\ldots$$

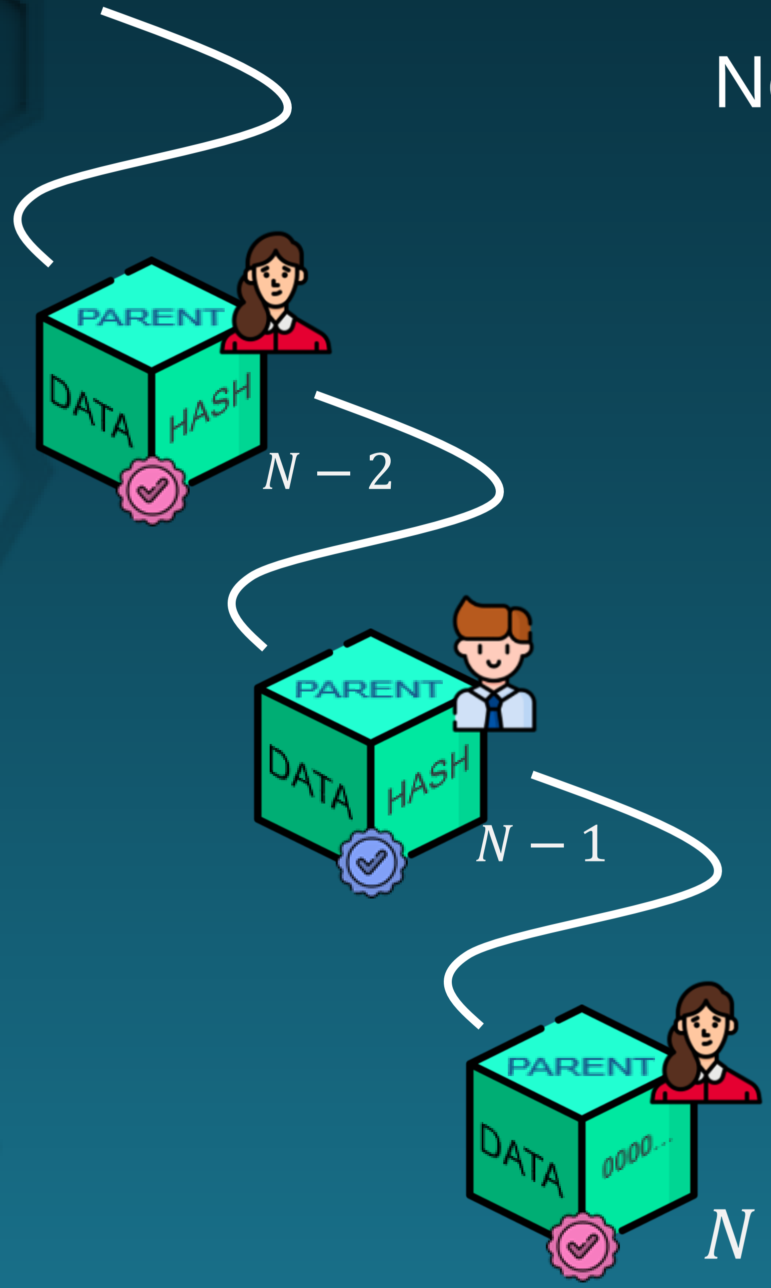
At one point she finds a nonce such that the block's hash starts with K zeros

Next block ?

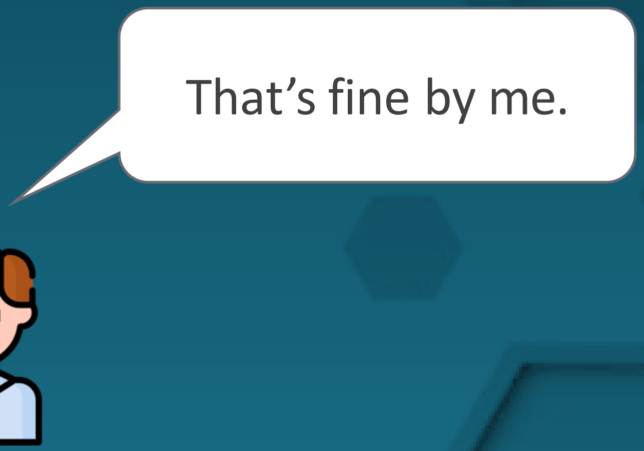


I am the next forger,
here is my block
PARENT
DATA
HASH
$N-2$
PARENT
DATA
HASH
$N-1$
PARENT
DATA
0000...
N

Next block ?



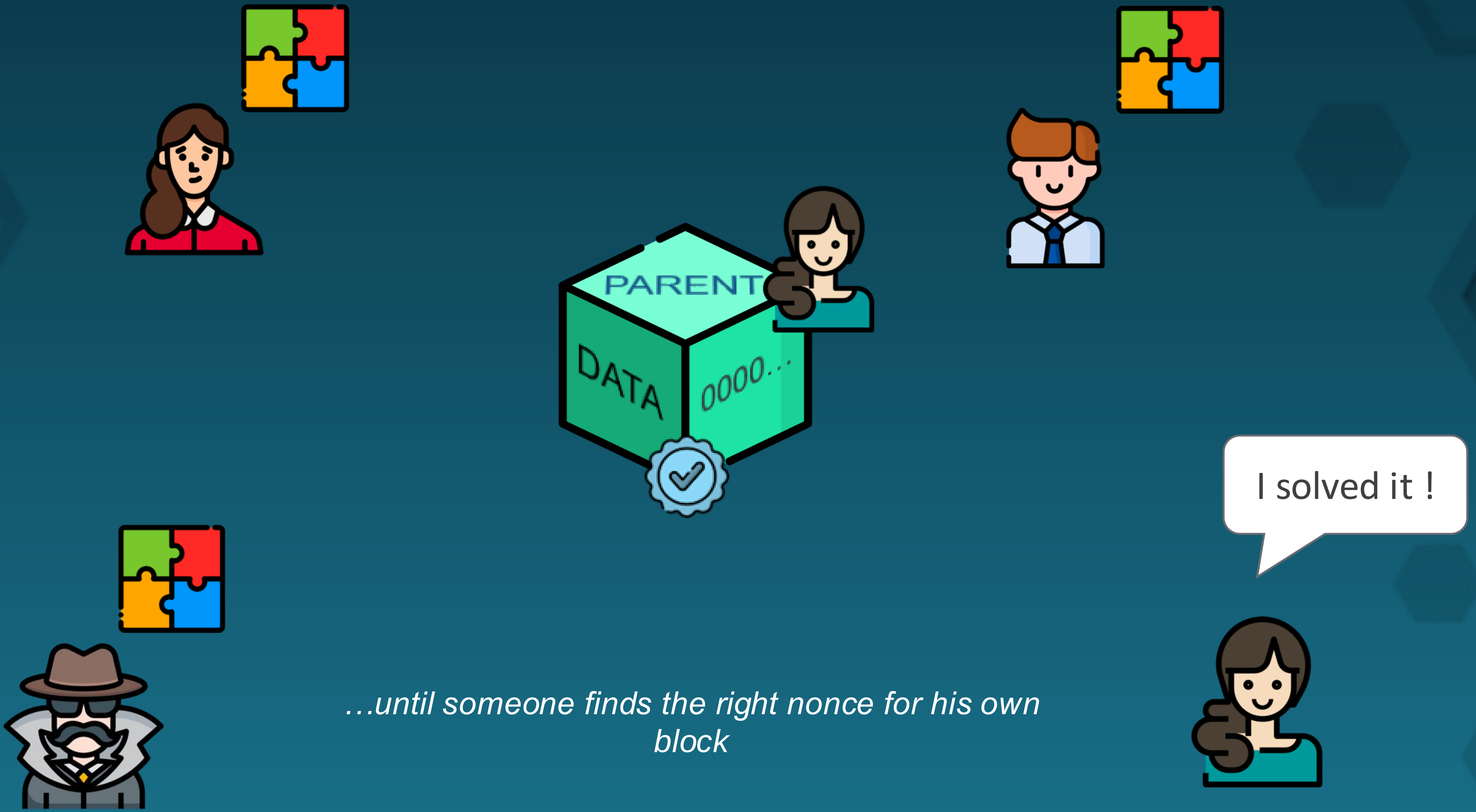
Bob can check that by hashing Alice's block's payload (including the nonce), he gets a hash that starts with K zeros



Consensus

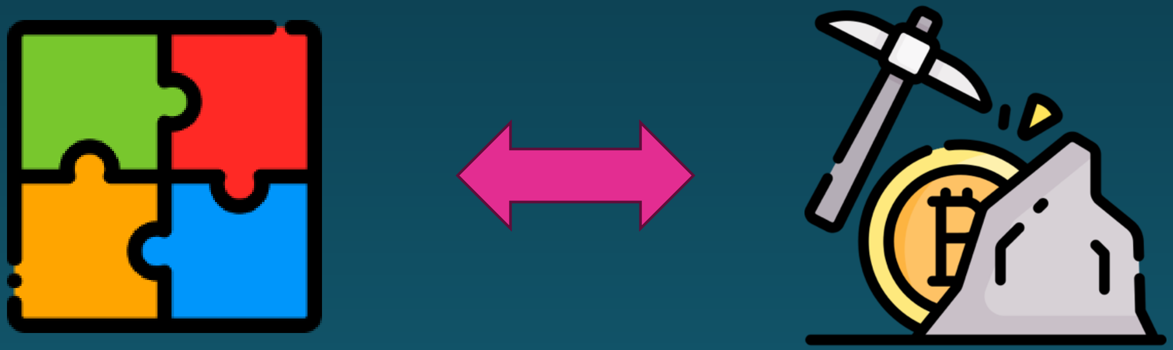
Everyone solves the hashing puzzle on his side…

Consensus



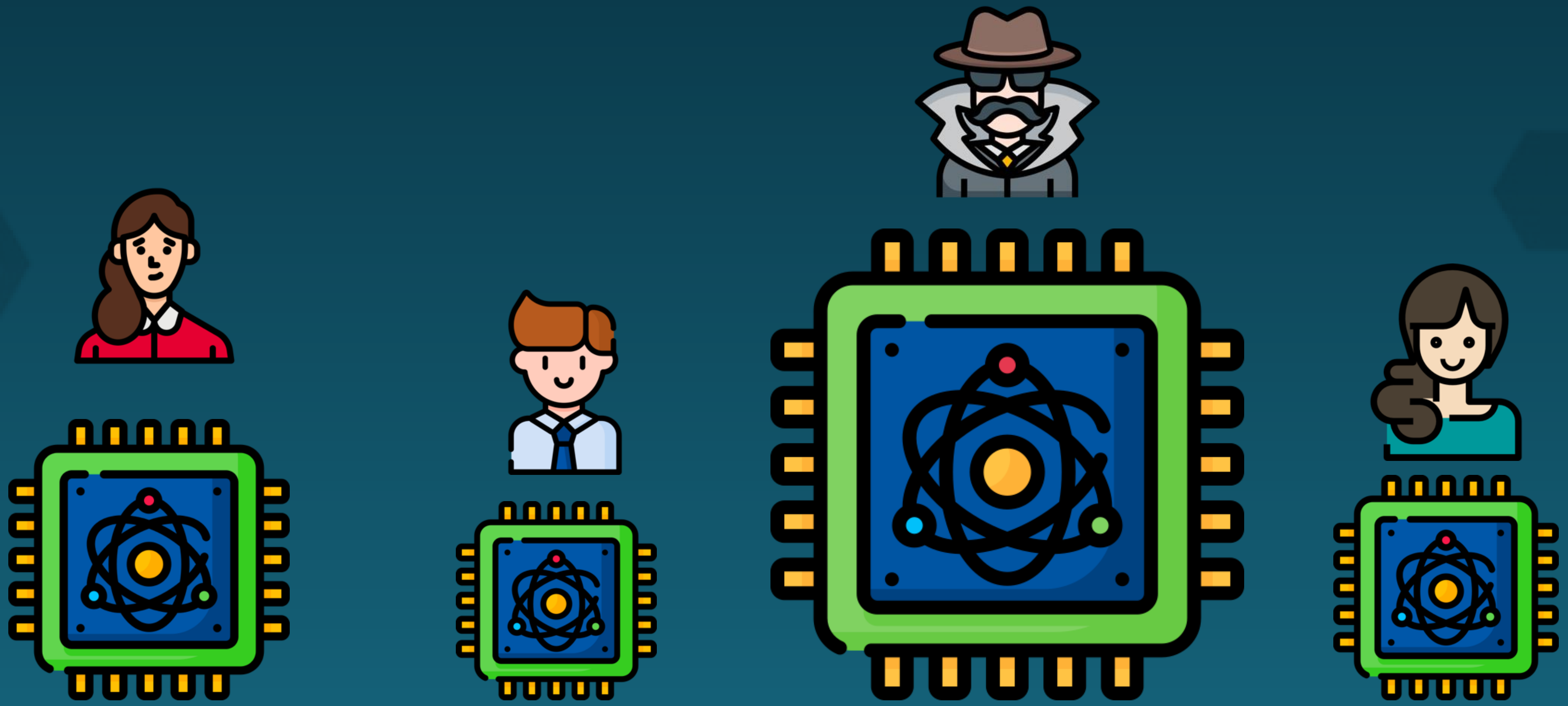
…until someone finds the right nonce for his own block

Bitcoin mining



Solving the puzzle = mining

Probabilities to forge



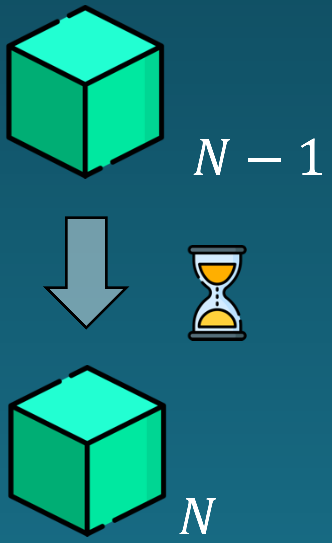
Each contributor has a probability to forge that is proportionnal to its computing power (hash rate)

Mining time

Satoshi Nakamoto

$$\mathrm{H} = \underbrace{\mathbf{0000}}_{}a7cd\ \ldots$$

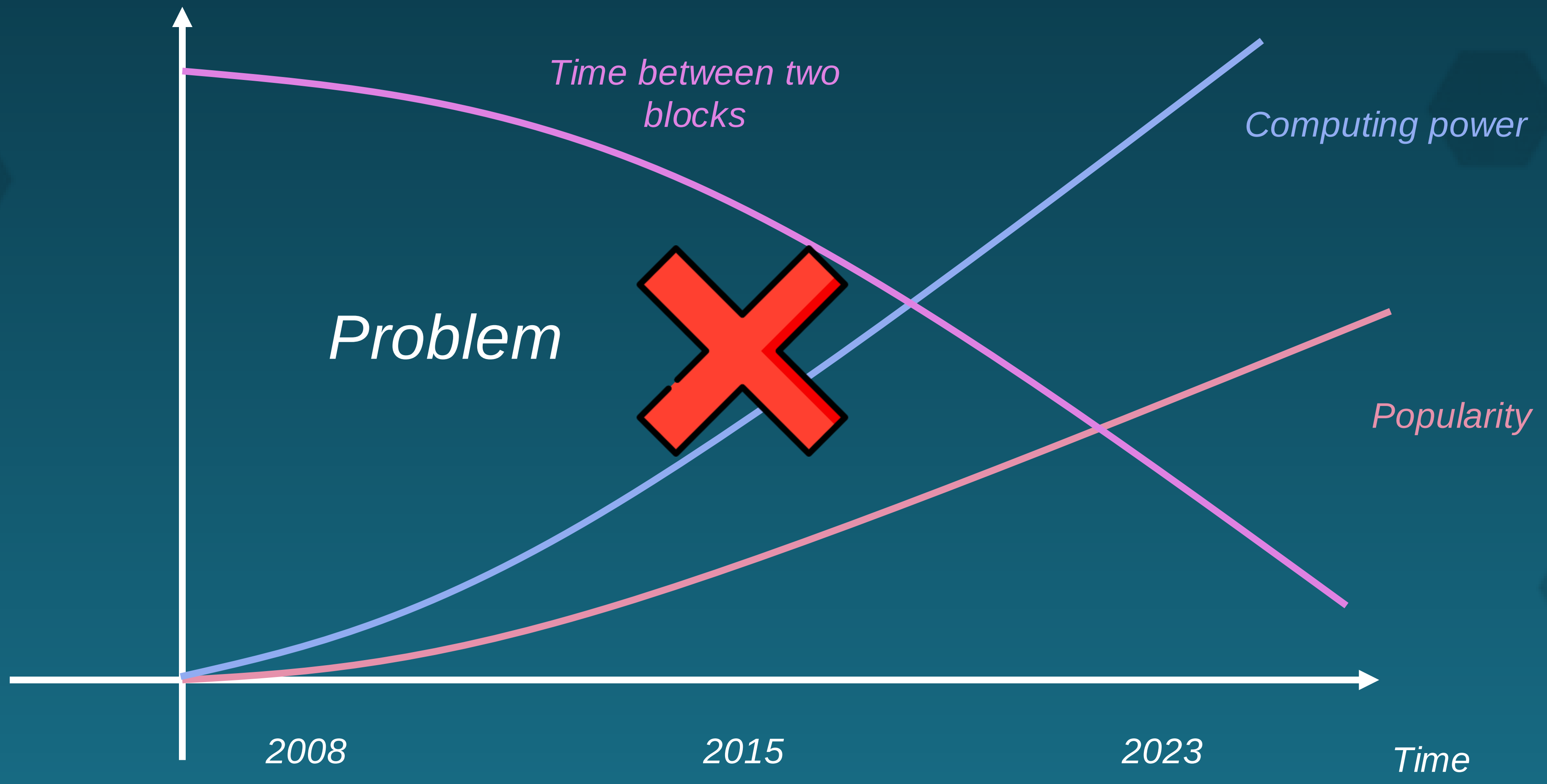
K zeros



Time between two blocks is a function of K and total computing power

Wanted 10 minutes

Mining time

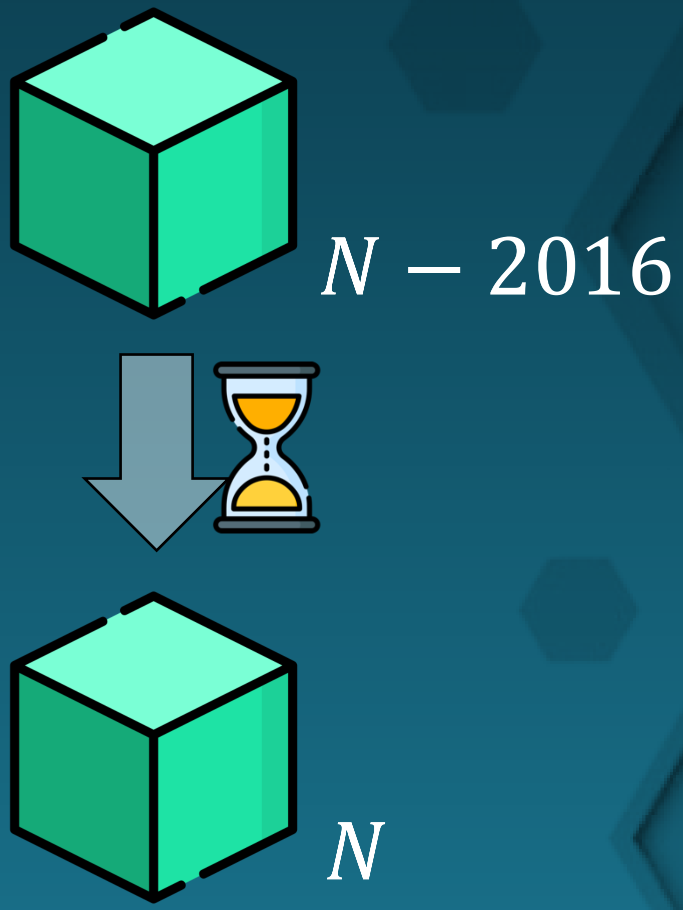

Time between two blocks
Computing power
Problem
Popularity
2008
2015
2023
Time

Mining time

$\mathbf{000}823bd \ldots$

$\mathbf{0000}a7cd \ldots$

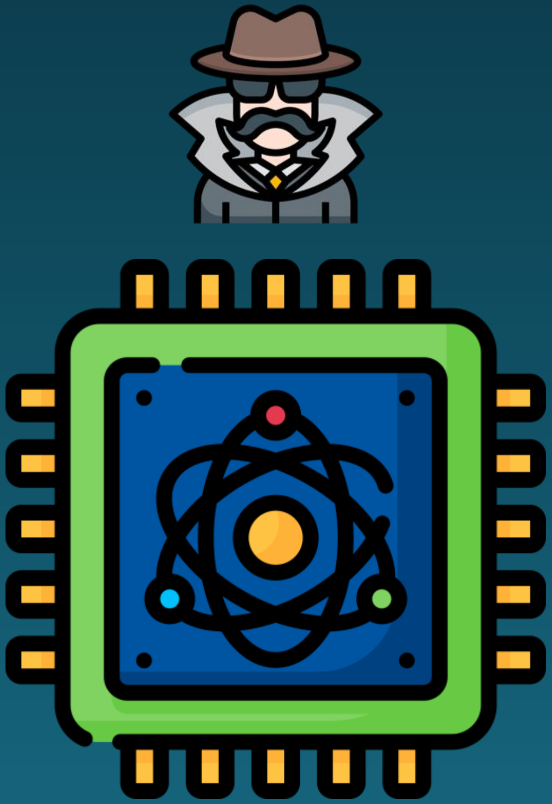
$\mathbf{00000}33b \ldots$



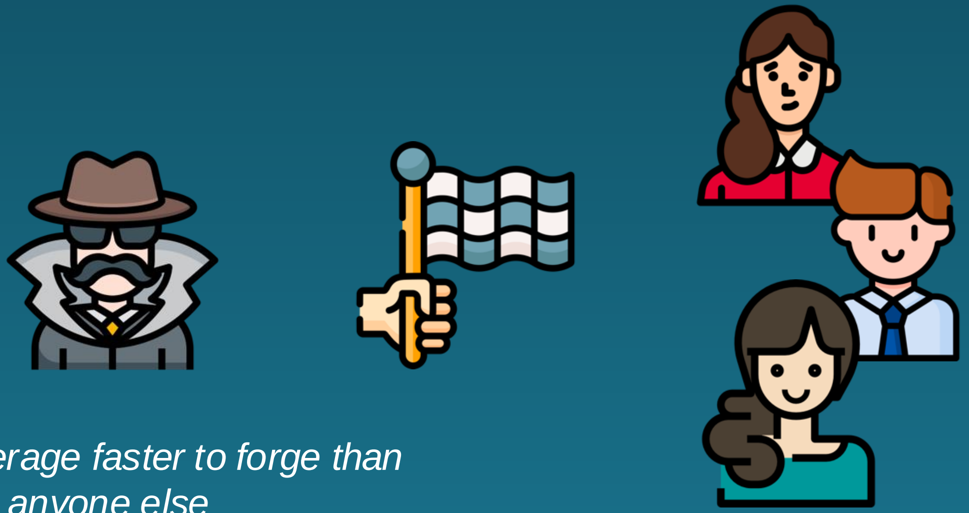
K is calculated using time needed for last 2016 blocks

51% attack

What happens when someone holds 51% of the total computing power of the blockchain ?

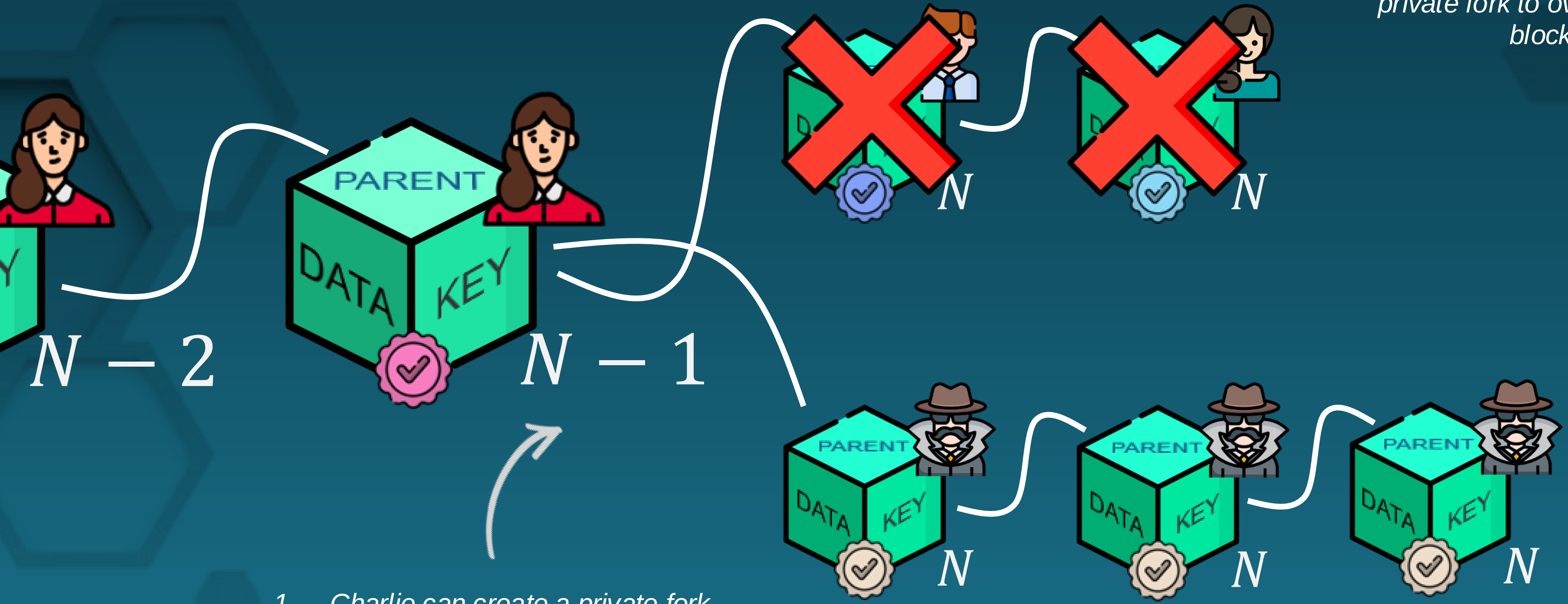


51%



He is in average faster to forge than anyone else

51% attack



3 — Since we keep the longest blockchain, Charlie can publish his private fork to override the actual blockchain

1 — Charlie can create a private fork for the blockchain

2 — Charlie is faster than anyone else to create new blocks

51% attack

Bitcoin Gold — Vertcoin — 2018

Ethereum Classic — 2019

For bitcoin : 260 EH/s

260.000.000.000.000.000.000 hashes per seconds

Roughly half of Sweden’s annual electricity consumption

Proof-of-Work

Perks

- Very simple
- Does not need parties to agree
- Adapts to computing power and popularity
- New people can join the train at the same ”point”

Proof-of-Work

Downsides

- Energy consumption +++
- Somehow vulnerable to 51% attacks
- Beneficial to people with great purchasing power
- Irrelevant for private companies

Consensus Algorithms

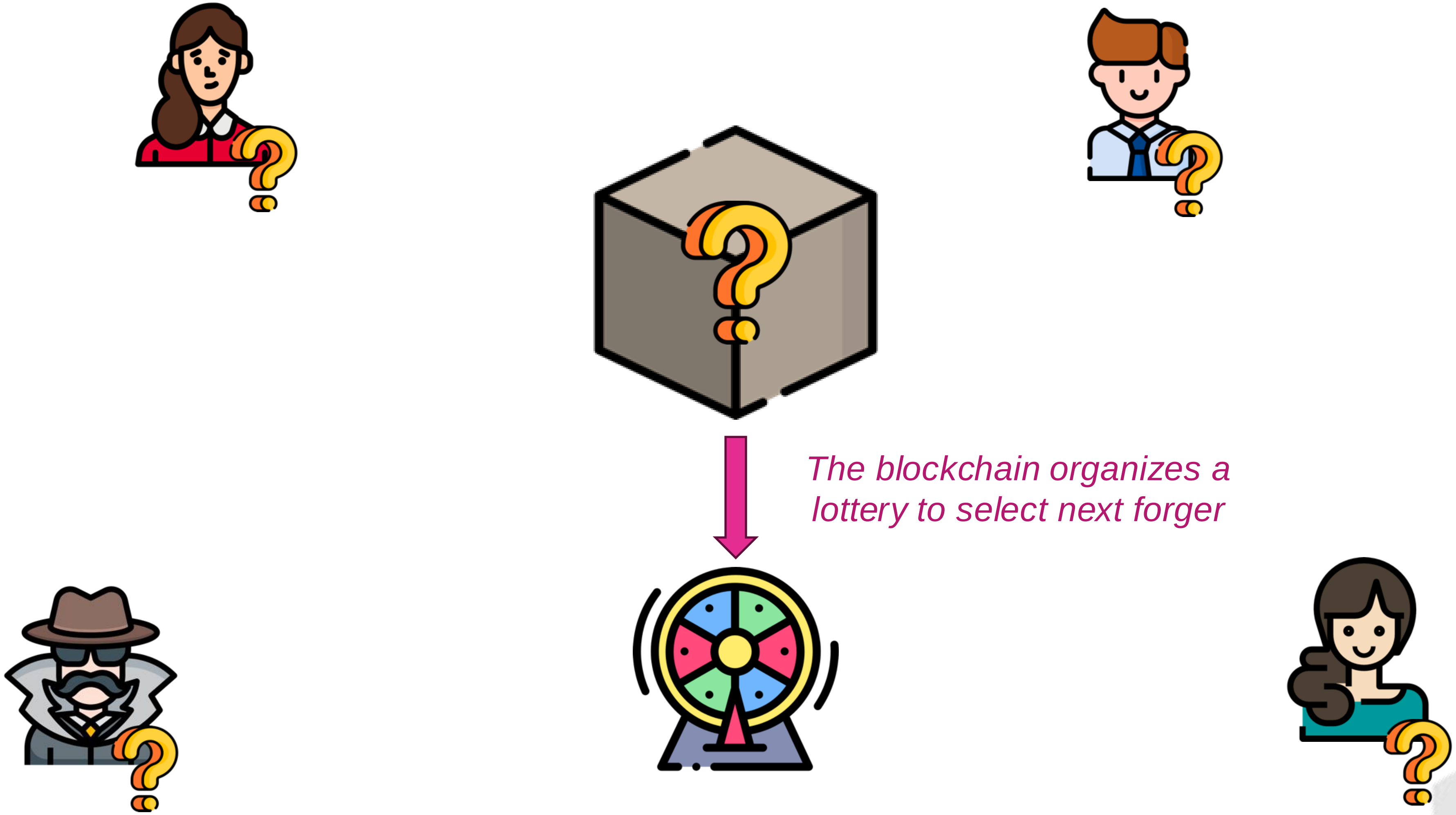
The most common

- Proof-of-Work
- Proof-of-Stake
- Delegated Proof-of-Stake
- Proof-of-Burn
- Proof-of-Authority
- Proof-of-Time

Currently the best for purely decentralized blockchains

Proof-of-Stake (2012)

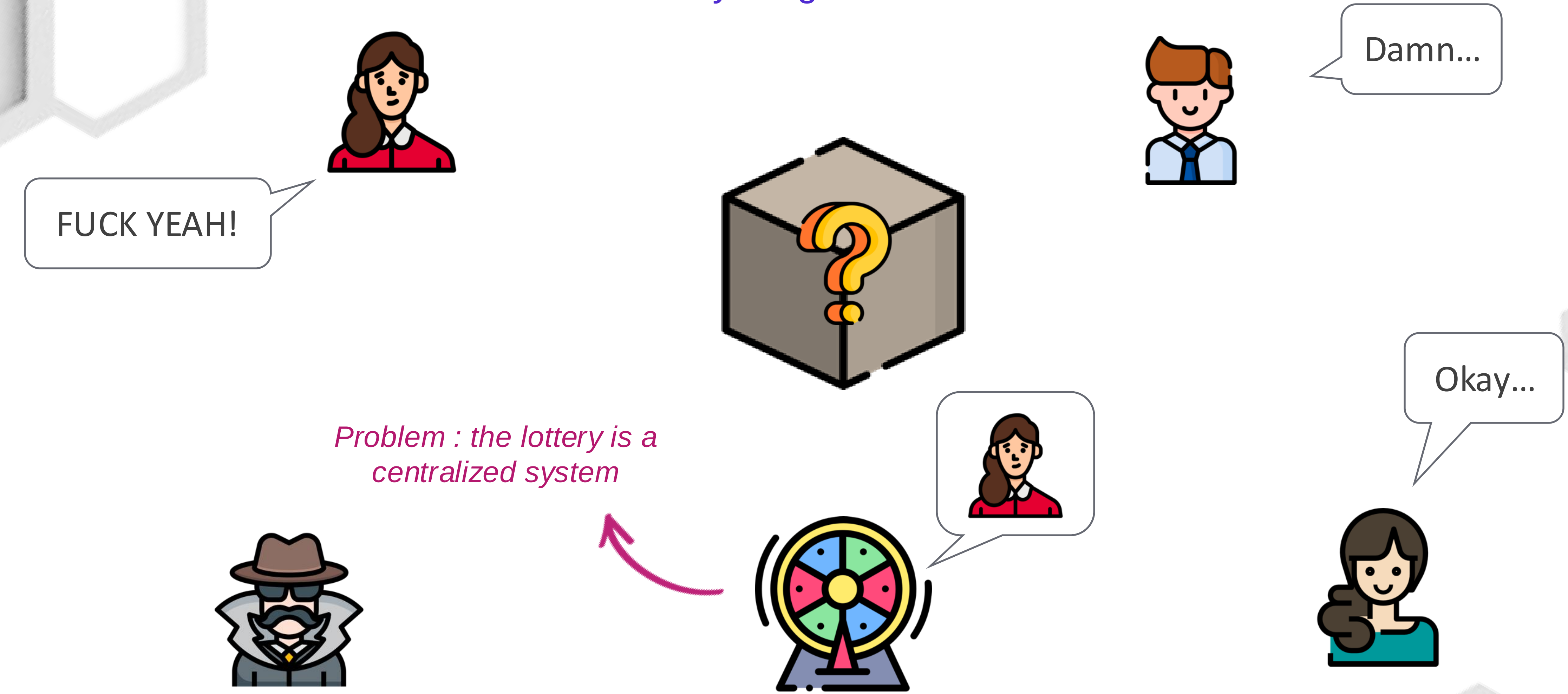
Sunny King et Scott Nadal



The blockchain organizes a lottery to select next forger

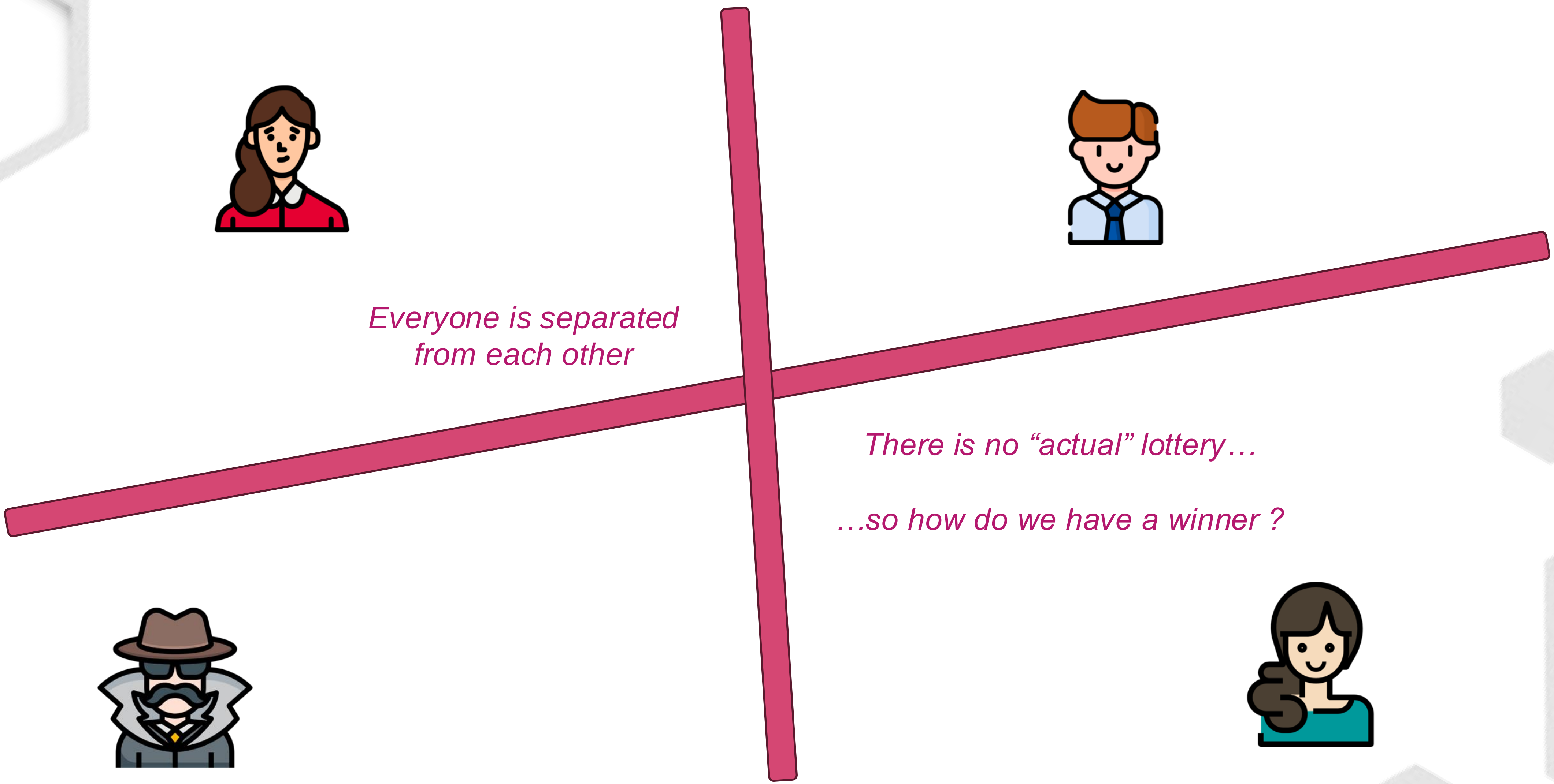
Proof-of-Stake (2012)

Sunny King et Scott Nadal



Problem : the lottery is a centralized system

Proof-of-Stake — In reality

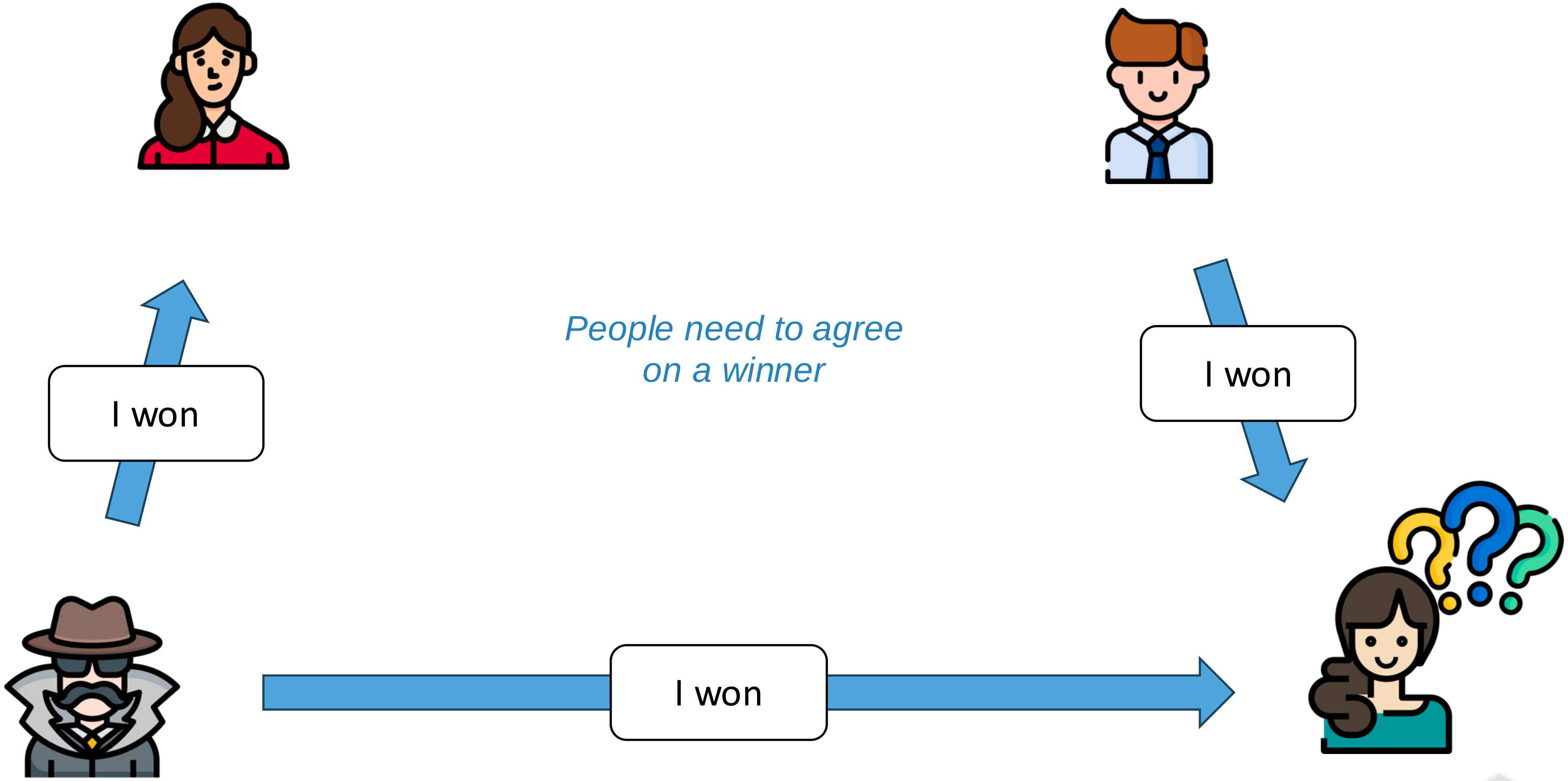


Everyone is separated from each other

There is no “actual” lottery…

…so how do we have a winner ?

Byzantin generals problem



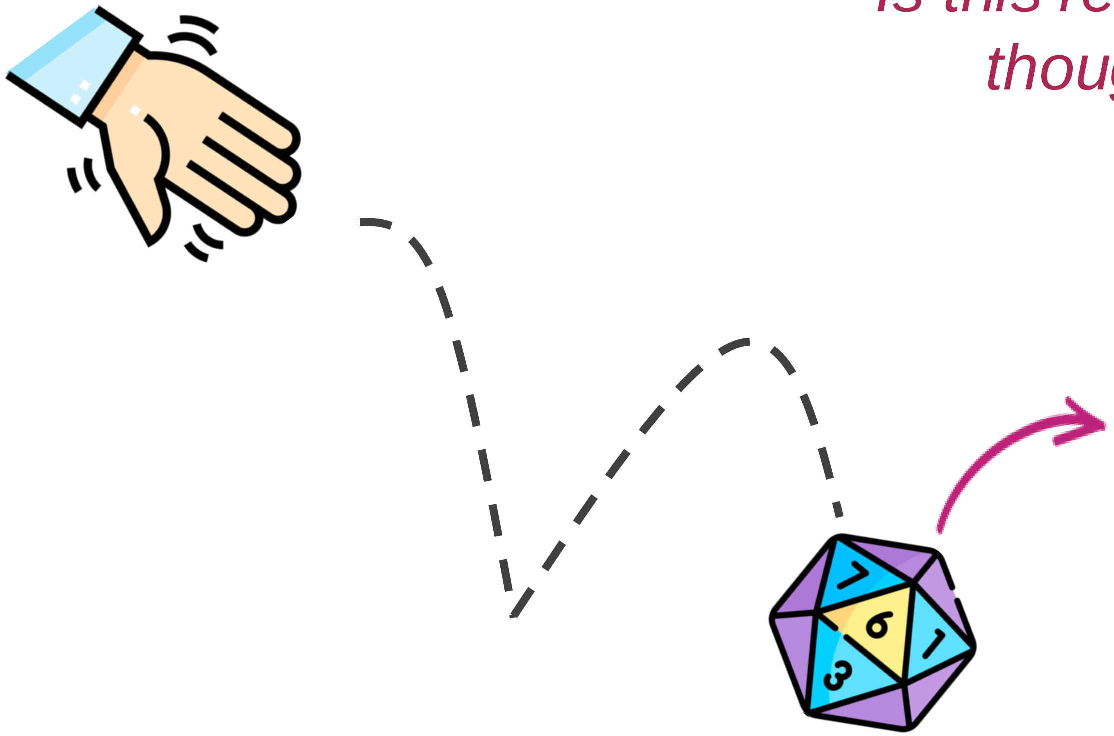
People need to agree on a winner

“Consensus” algorithms

We call them “consensus algorithms” but there really is no consensus after all

Deterministicity

Is this really luck though ?

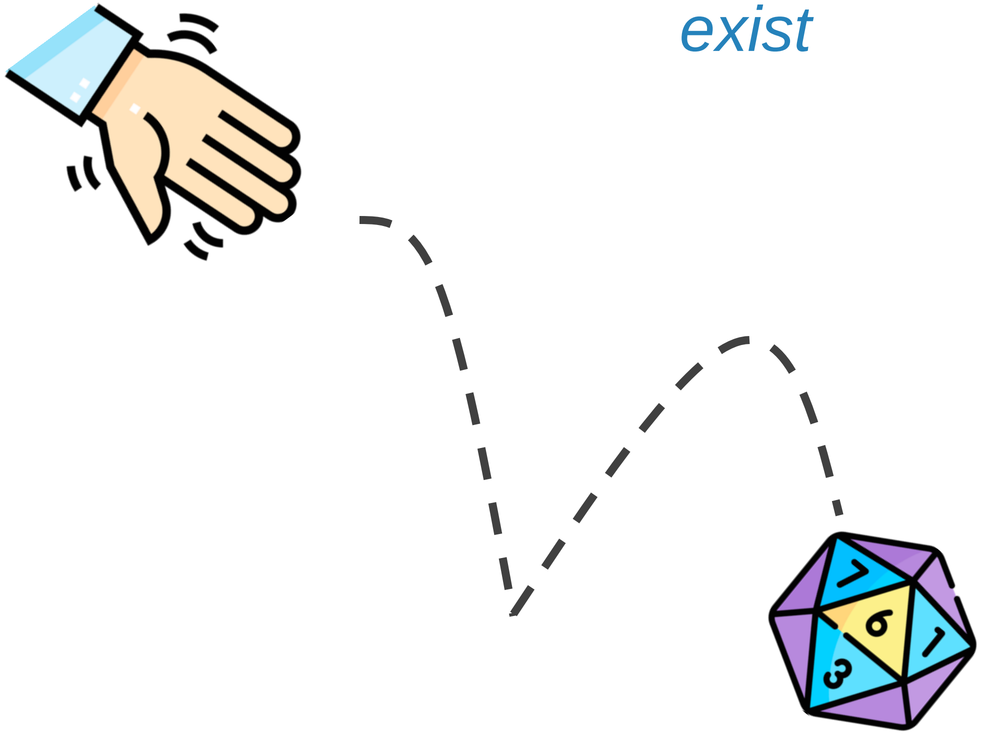


Result : 20

LUCKY YOU

Deterministicity

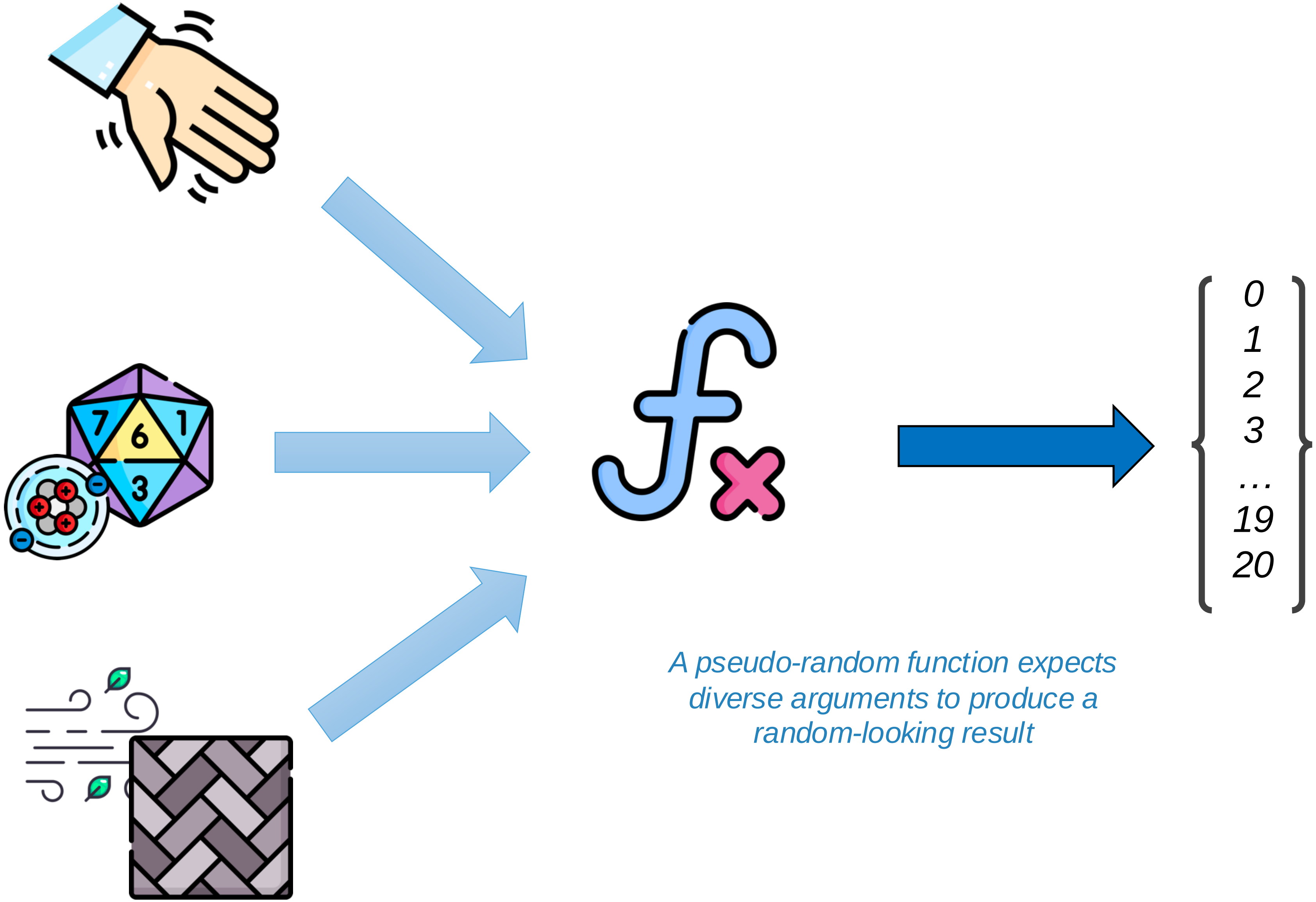
Random does not exist



Air pressure
Hand gesture
Ground angle
Atoms in the dice
Earth’s magnetism
Quantum phy
…

Result : 20

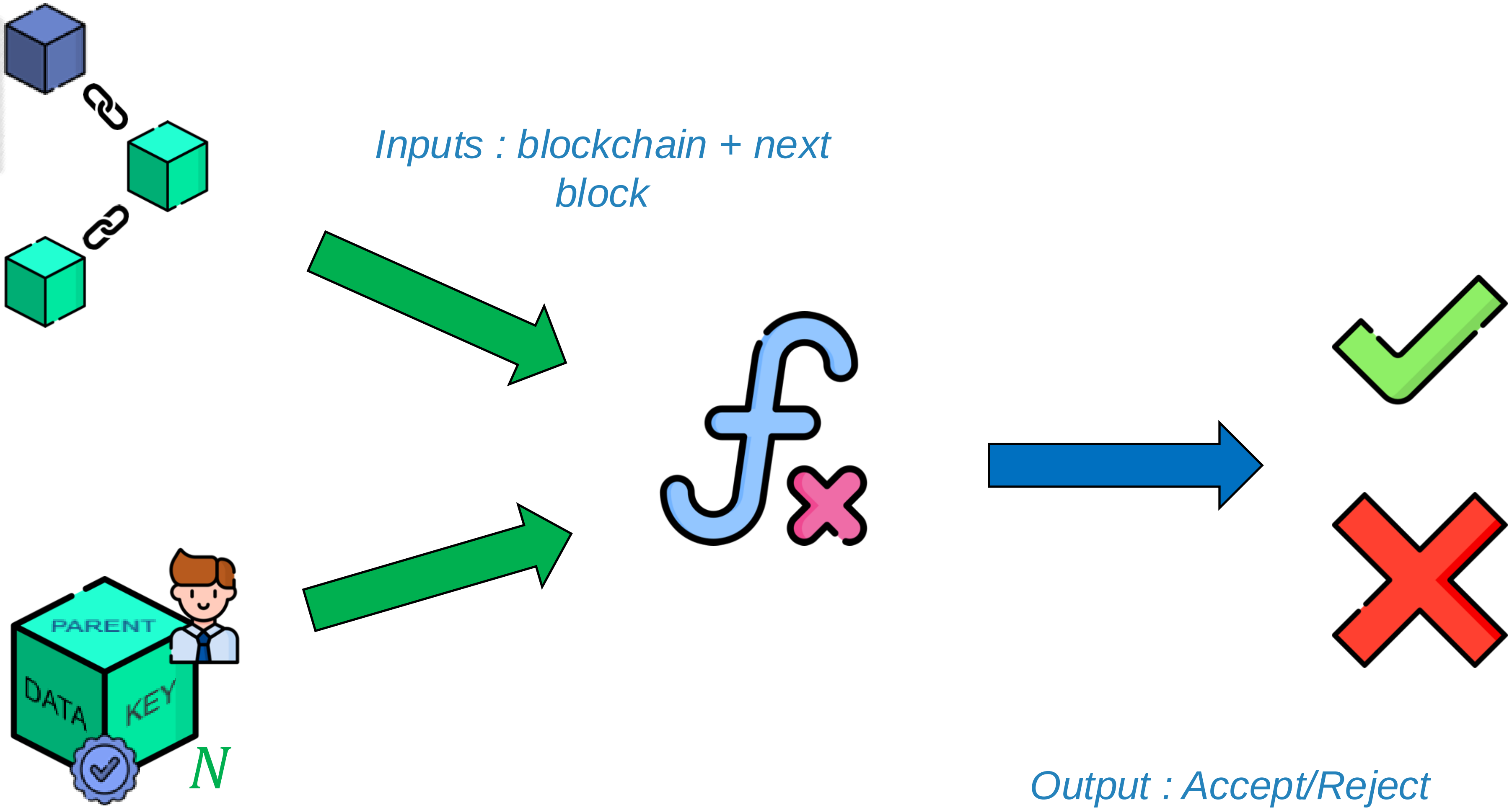
Pseudo-random function



$$\left\{\begin{array}{c}0\\1\\2\\3\\\ldots\\19\\20\end{array}\right\}$$

A pseudo-random function expects diverse arguments to produce a random-looking result

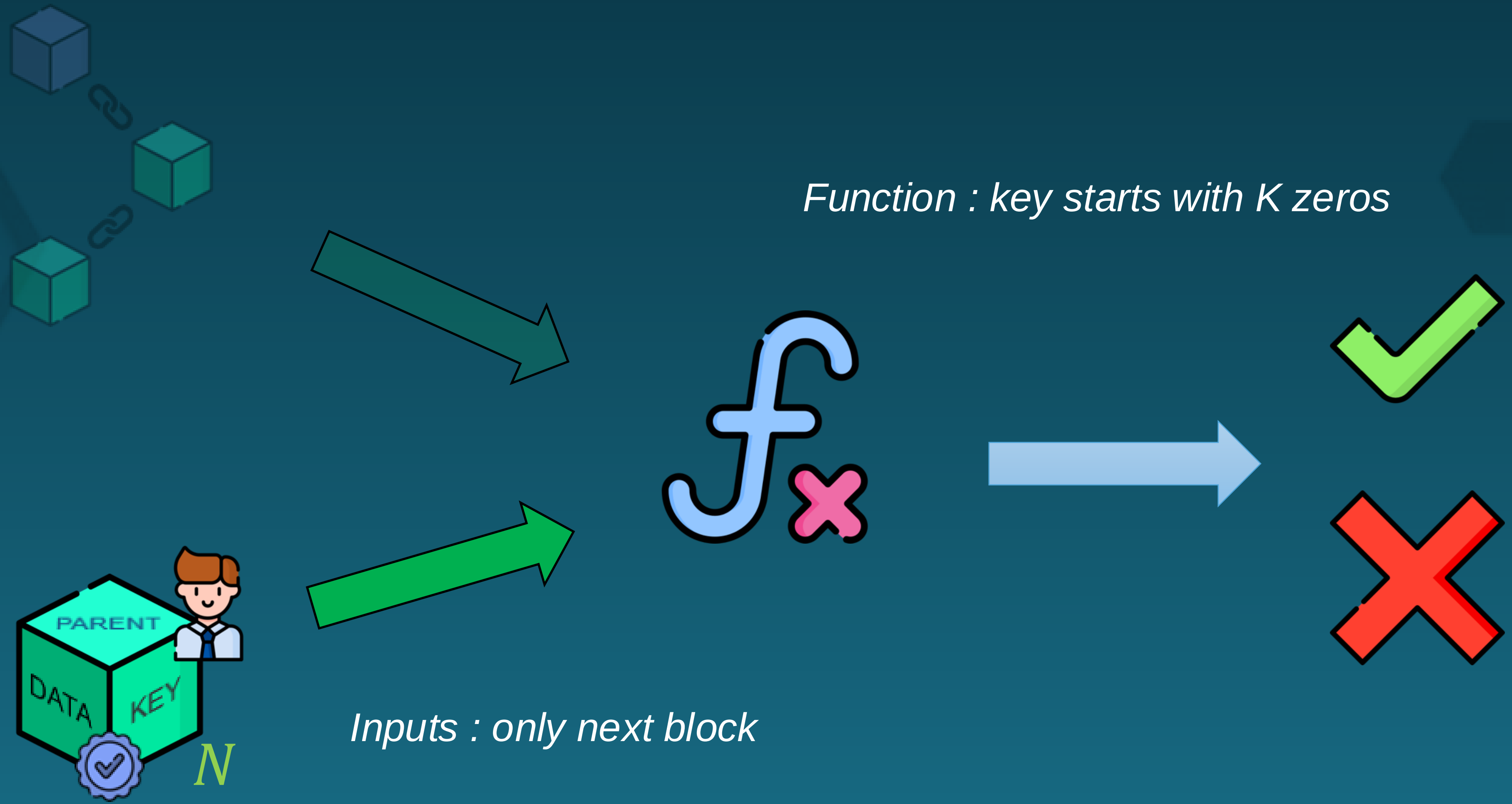
“Consensus” algorithm



Inputs : blockchain + next block

Output : Accept/Reject

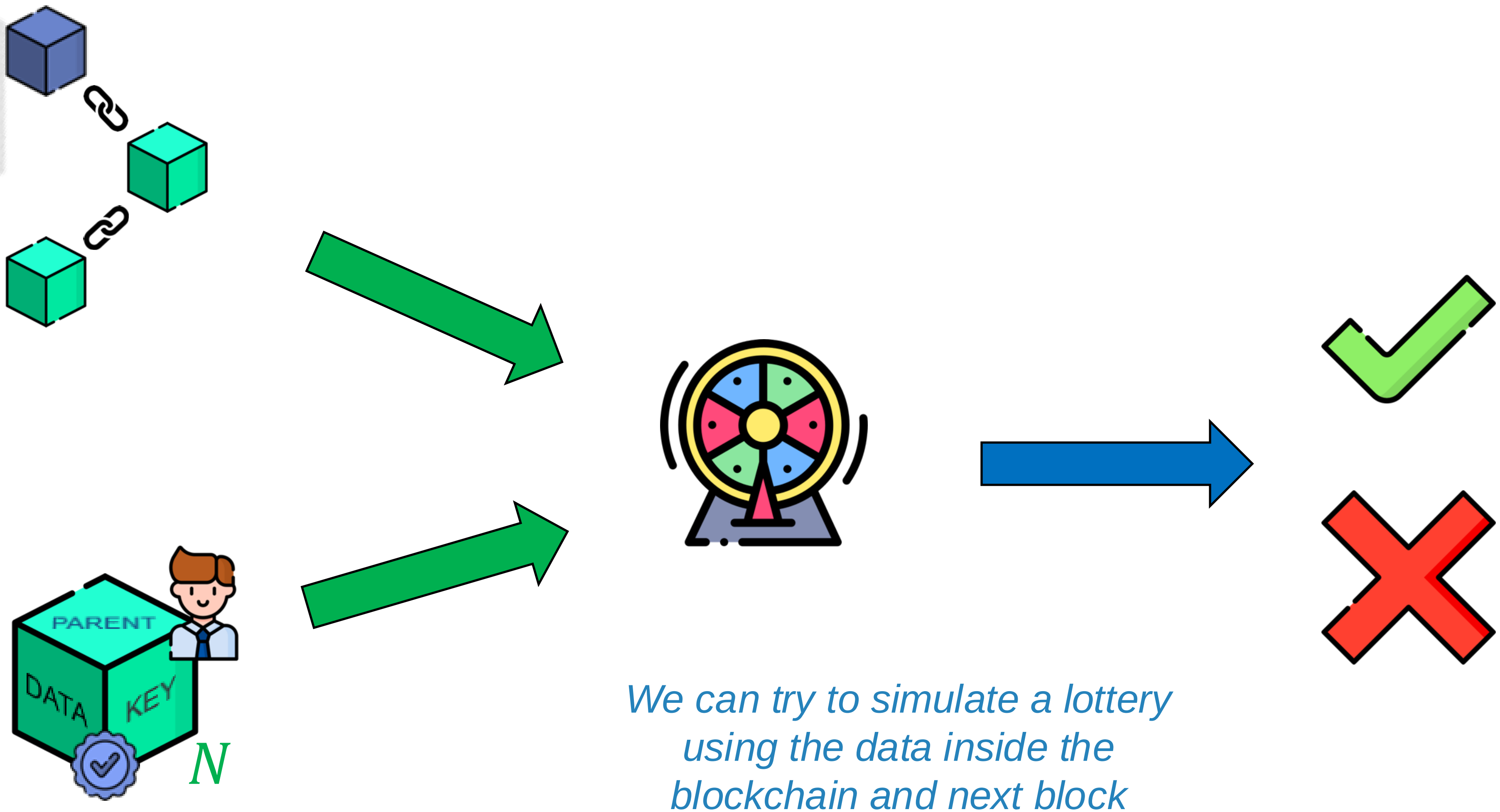
Example : Proof-of-Work



Function : key starts with K zeros

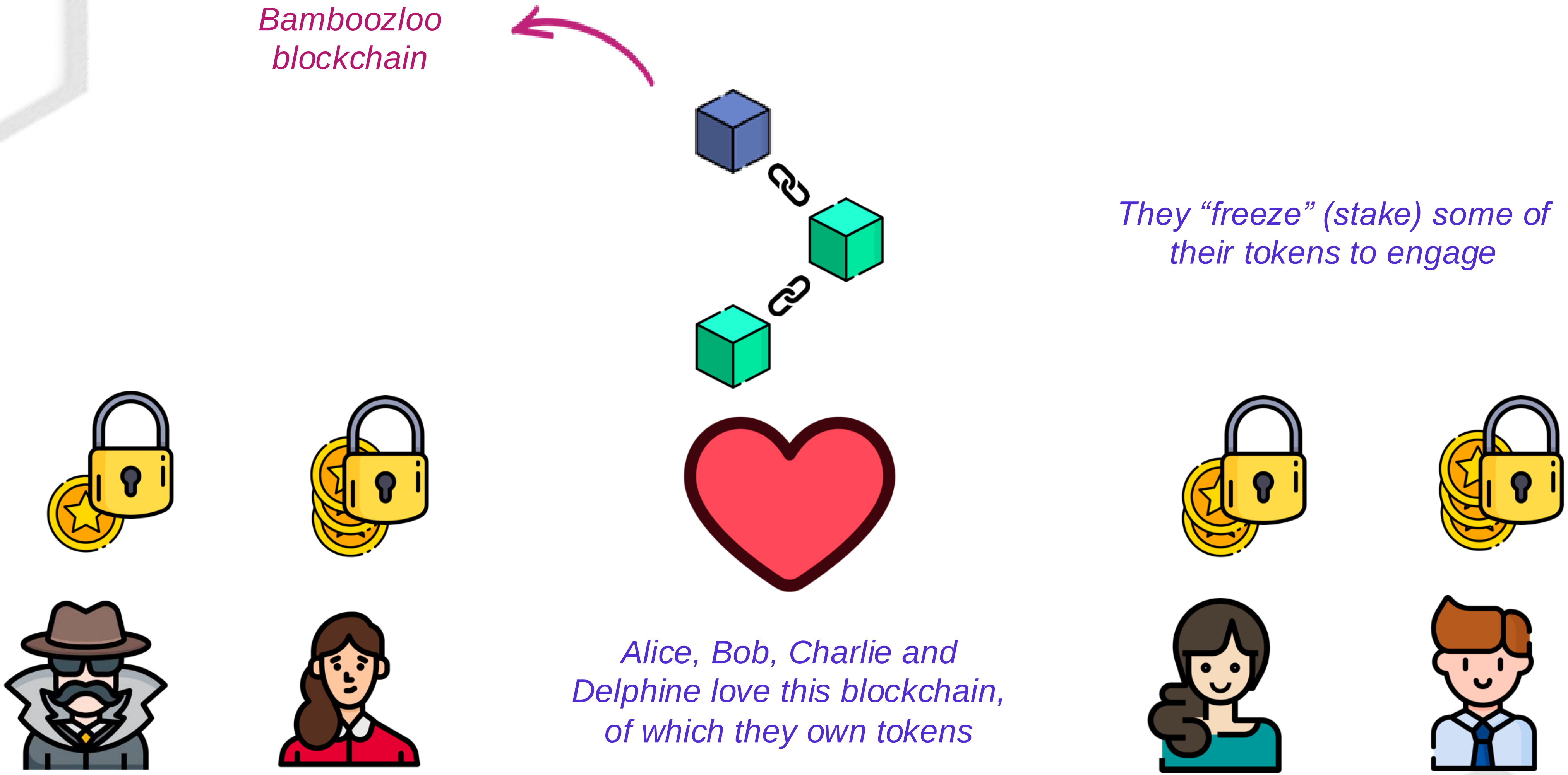
Inputs : only next block

Proof-of-stake



We can try to simulate a lottery using the data inside the blockchain and next block

Proof-of-stake



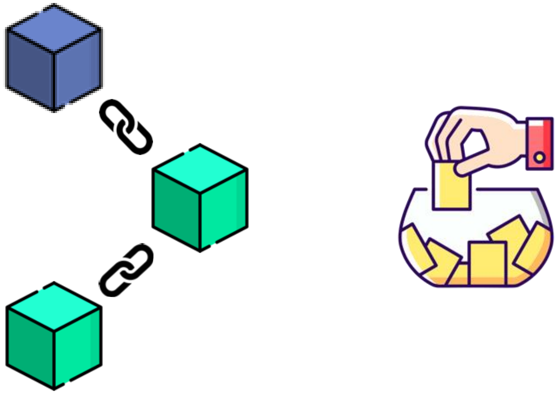
Bamboozloo blockchain

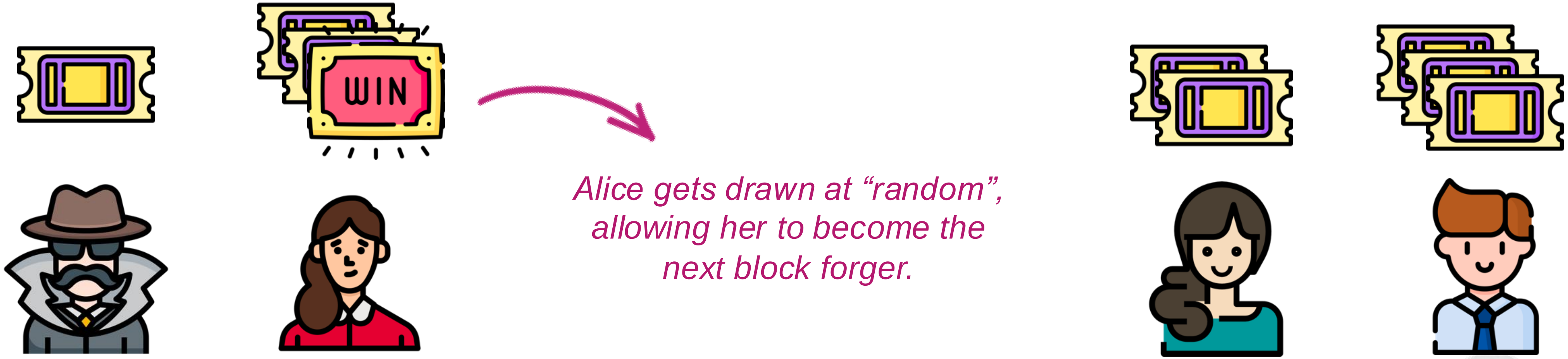
They “freeze” (stake) some of their tokens to engage

Alice, Bob, Charlie and Delphine love this blockchain, of which they own tokens

Proof-of-stake

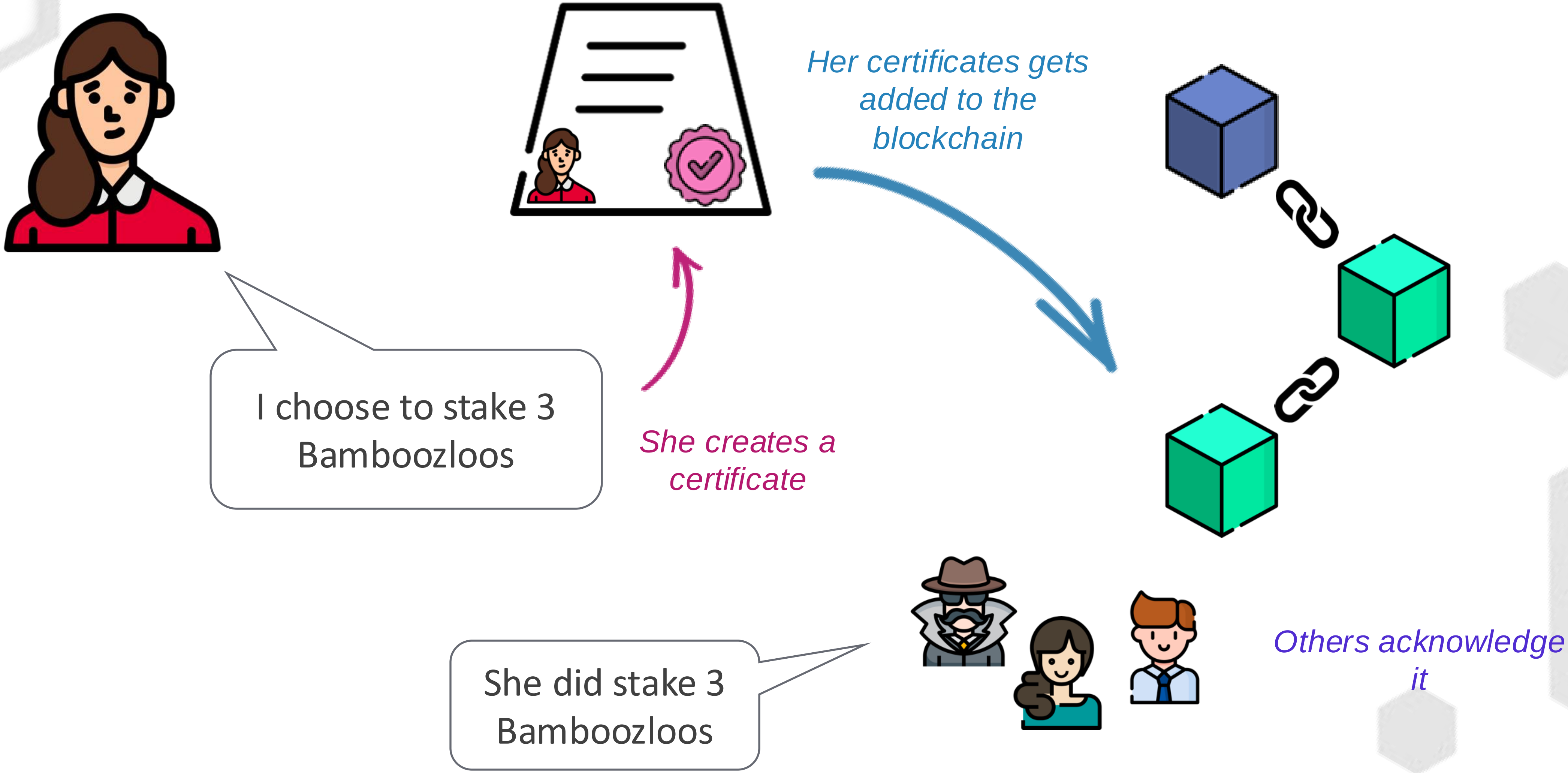
For each of their staked token, they get a lottery ticket to be selected as next forger





Alice gets drawn at “random”, allowing her to become the next block forger.

Staking



I choose to stake 3 Bamboozloos

She creates a certificate

Her certificates gets added to the blockchain

Others acknowledge it

She did stake 3 Bamboozloos

Tickets

Alice gets 3 tickets

Each ticket has information…

Owner : Alice

Latest block hash : -4273784

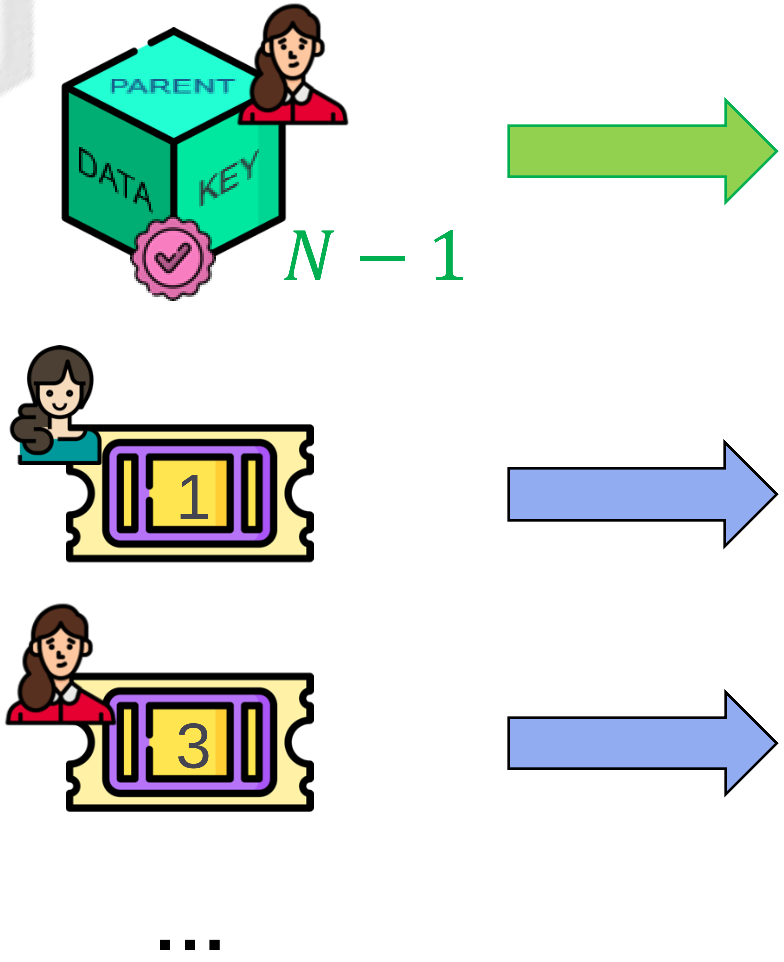
Ticket number : 1

$$H = 2278364$$

…that can be scrambled into a hash

Lottery

We look for the ticket whose hash is closest to the latest block hash



$N - 1$

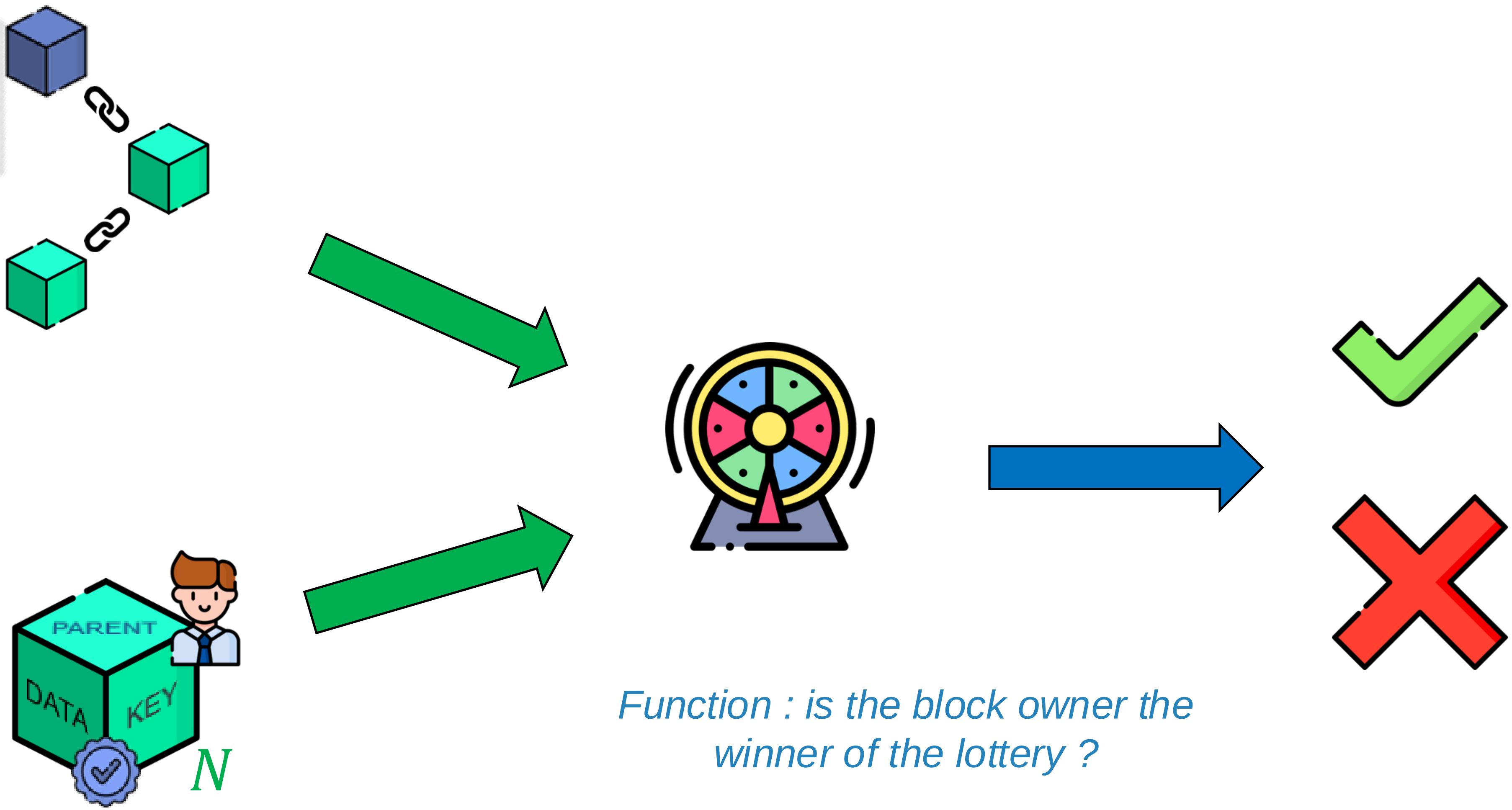
$$H_{N-1} = -654$$

$$H = 357462$$

$$H = -1792$$

WINNER !

Proof-of-stake



Function : is the block owner the winner of the lottery ?

(comparing tickets with latest block hash of the blockchain)

Proof-of-Stake

Perks

- Does not consume energy
- Fair
- Incentives people to engage into the blockchain
- 51% attack requires to own more than half the total market capitalization (and accept to lose it)

Proof-of-Stake

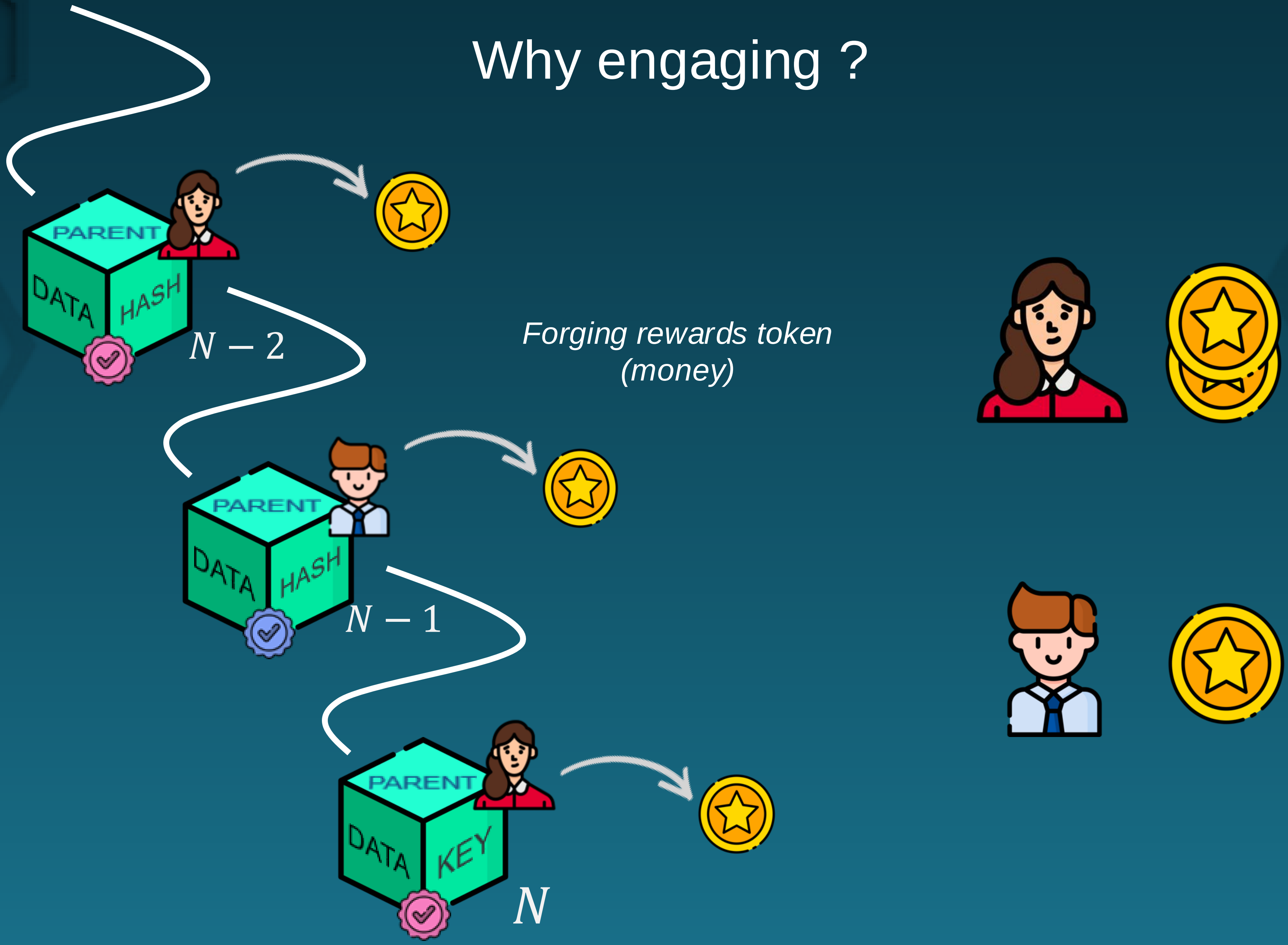
Downsides

- Rich-getting-richer

Can be addressed using Delegated Proof-of-Stake

- Requires a decent tokenomic

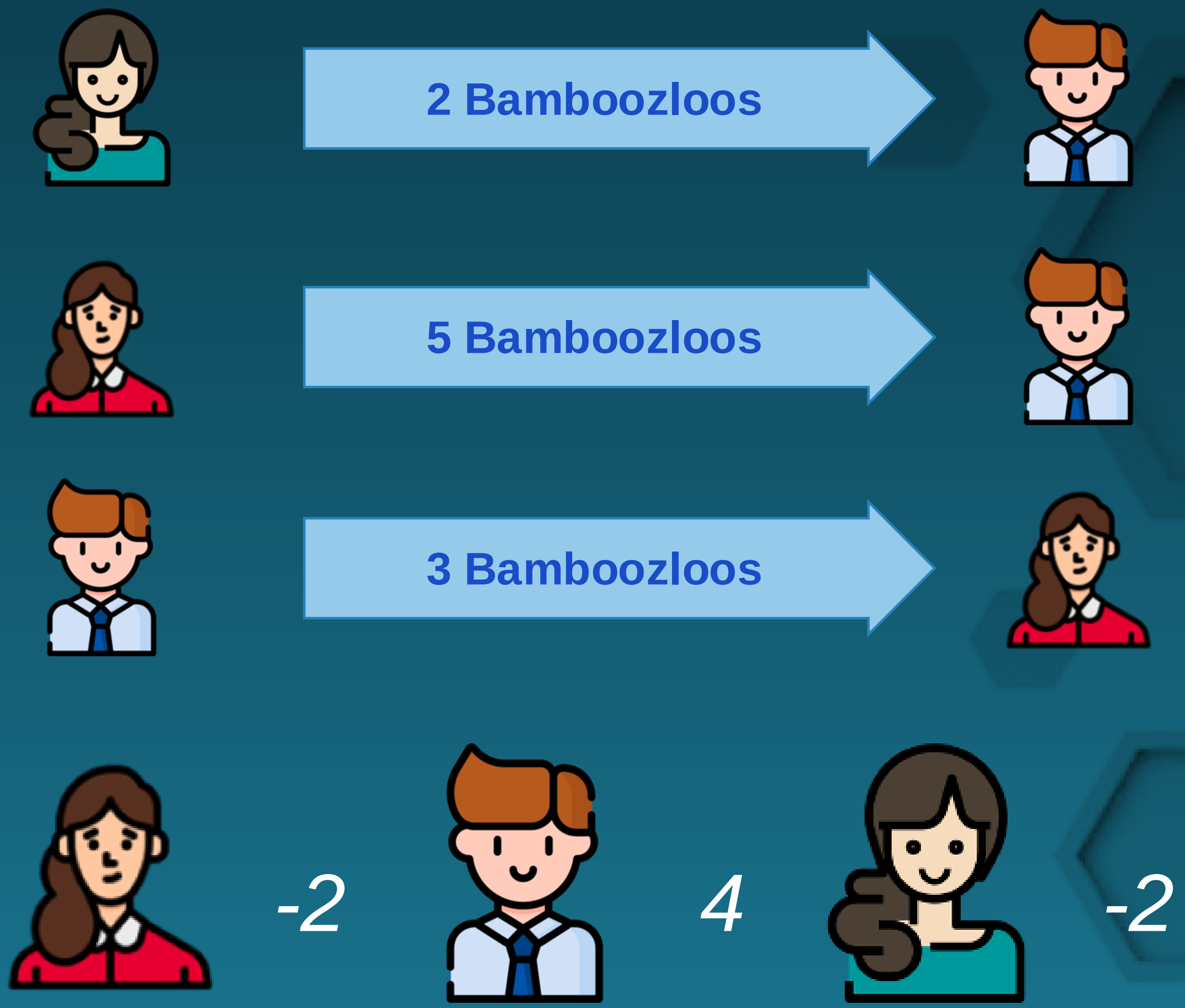
Why engaging ?



Forging rewards token (money)

Back to Bamboozloos

Is it a good idea to allow debts ?



2 Bamboozloos

5 Bamboozloos

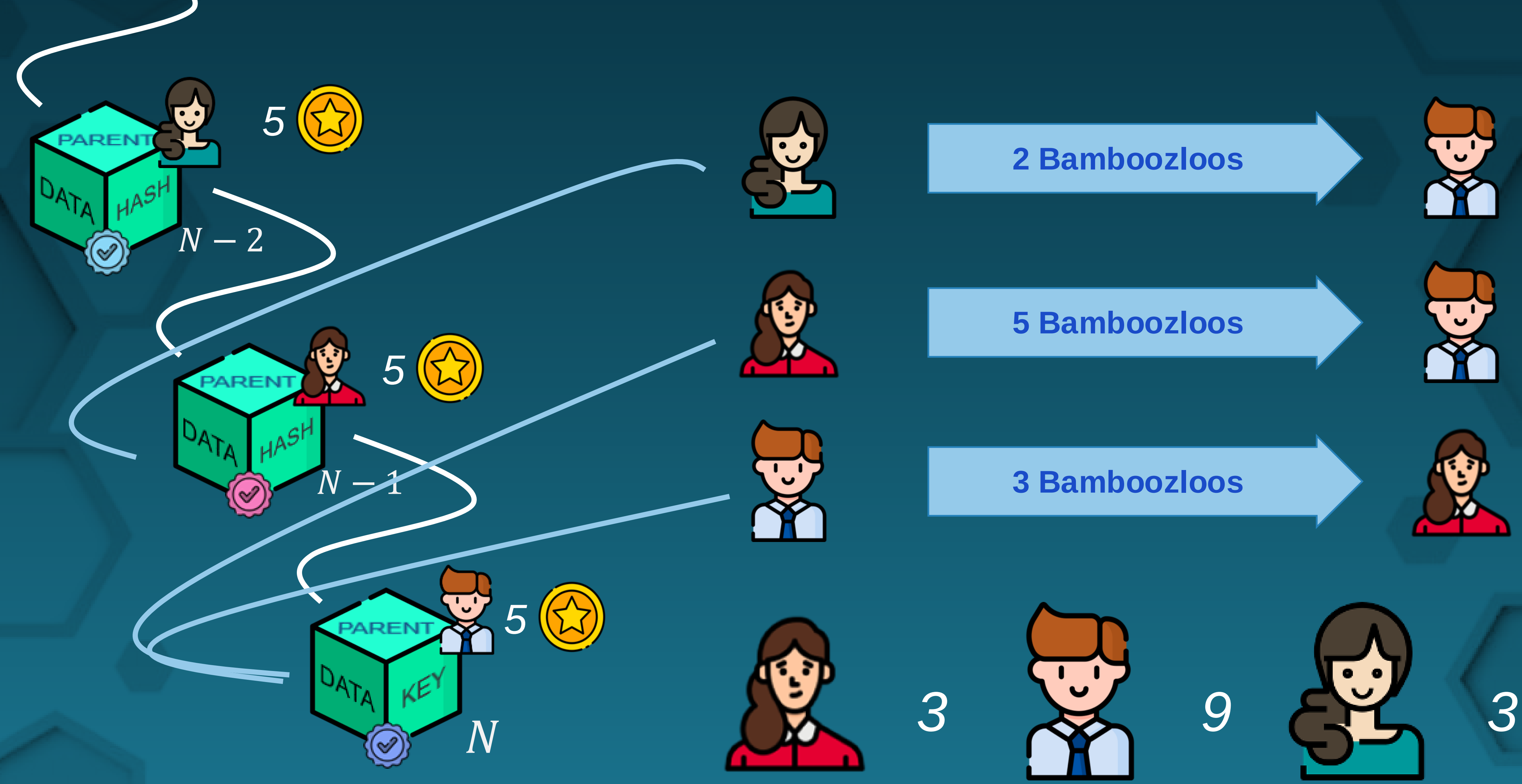
3 Bamboozloos

-2

4

-2

Back to Bamboozloos



2 Bamboozloos

5 Bamboozloos

3 Bamboozloos

Consensus Algorithms

The most common

- Proof-of-Work
- Proof-of-Stake
- Delegated Proof-of-Stake
- Proof-of-Burn
- Proof-of-Authority
- Proof-of-(Elapsed-)Time

Solana — Proof-of-History

Blockchain et Applications

Quiz 3

Algorithmes de consensus