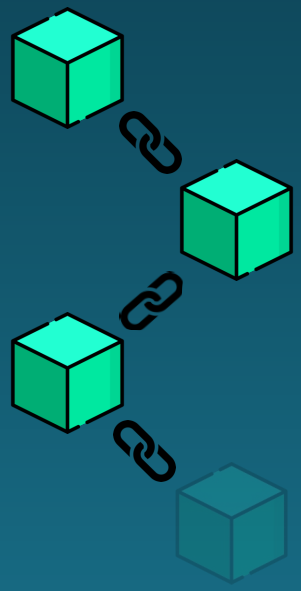
Blockchain and Applications

Chapter 2

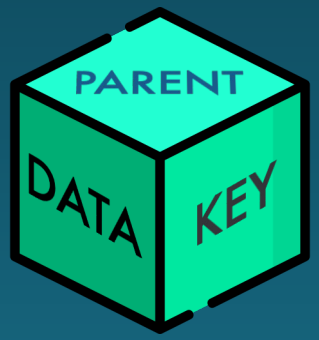
Cryptography and Certificates

Reminder of last week

Chain of blocks



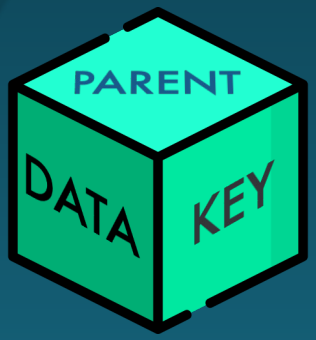
Blocks



Ledger

Name	Data	Signature
Alice	…	
Alice	…	
Bob	…	

This week – what's in the block ?



Conceptually
(Lecture)

Concretely
(Lecture + TD 2)

Certificates

Certificates

Alice wants to declare something

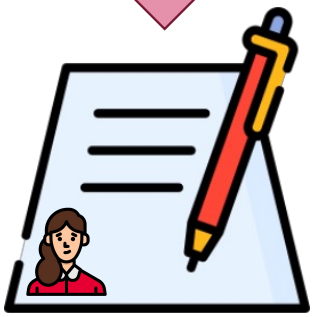
Covfefe

Statement

Certificates

Covfefe

She writes that statement down

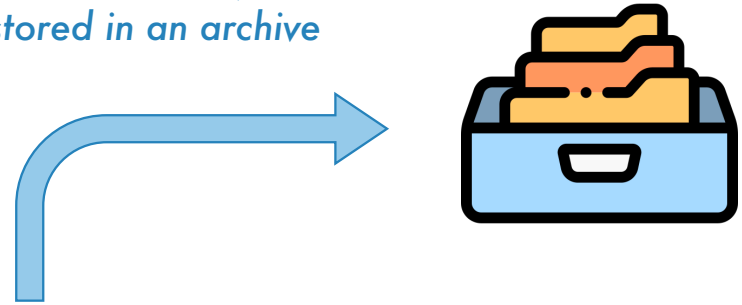


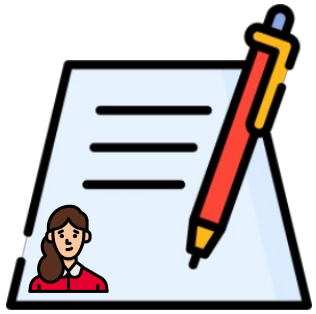
Certificate

She adds her name

Certificates

Once it's done, it is stored in an archive

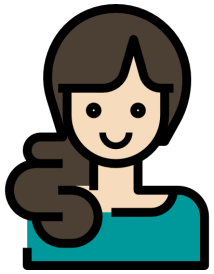


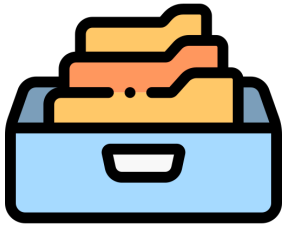


Certificate

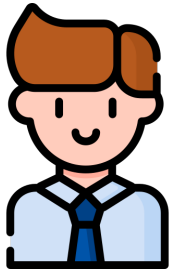
Certificates

Did Alice just say "Covfefe" ?

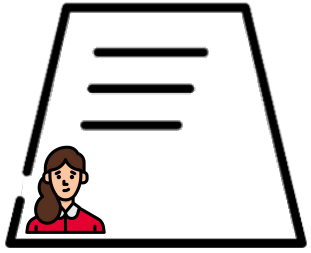




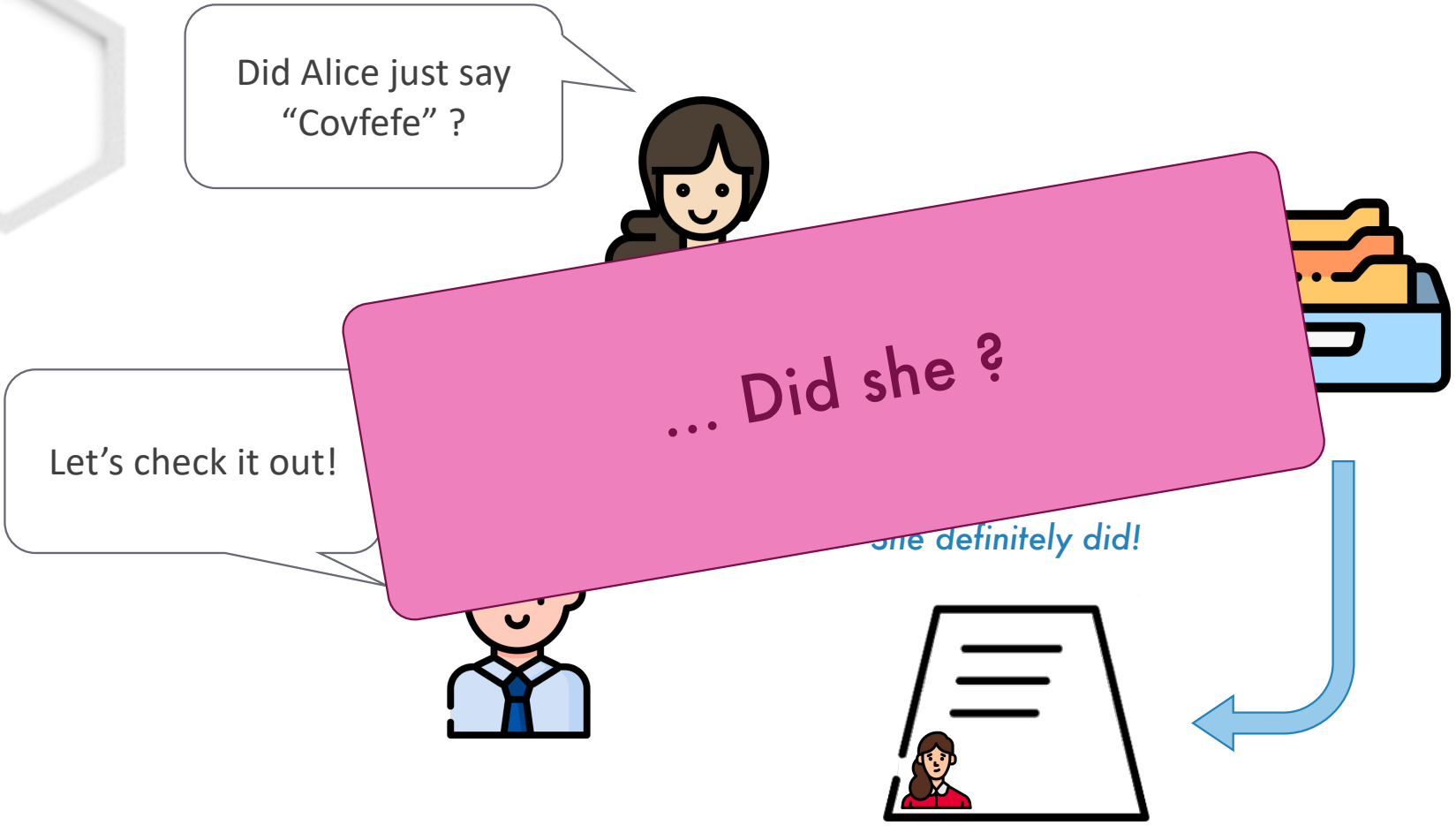
Let's check it out!



She definitely did!



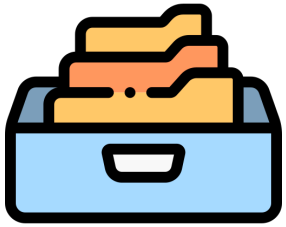
Certificates

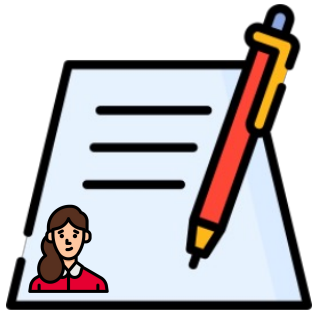


Certificates

I love eating pasta with Nutella

Straight to the archives



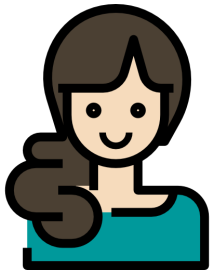


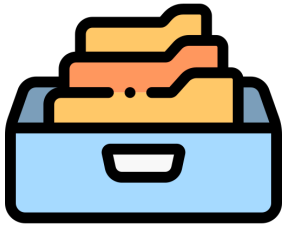
Charlie adds Alice's name

"Certificate"

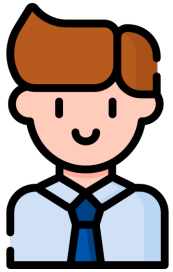
Certificates

Ew… Gross!

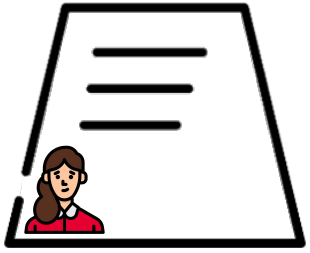




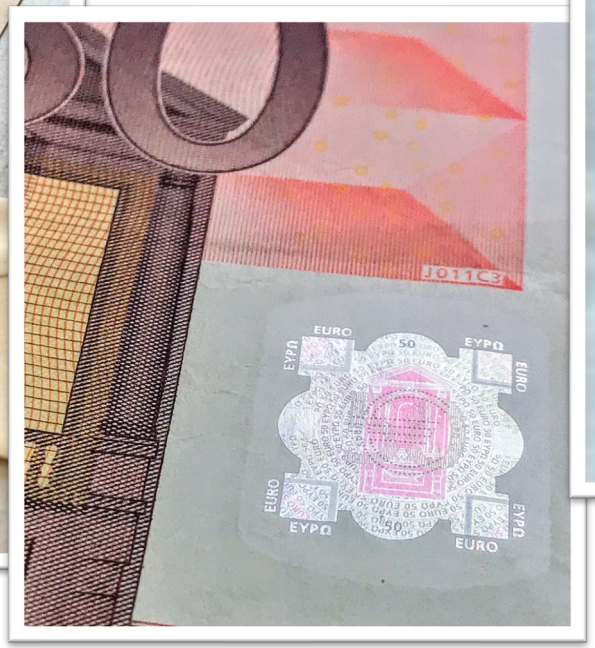
I see Alice likes her pasta with Nutella, what a freak!



She definitely did say such a thing



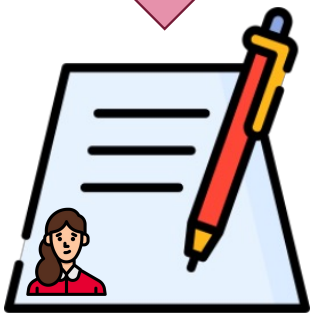
Certificate authenticity



How do we legitimate a certificate ?

Covfefe

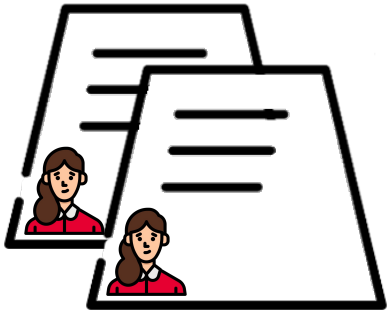
Alice creates a certificate with her statement



She still adds her name

How do we legitimate a certificate ?

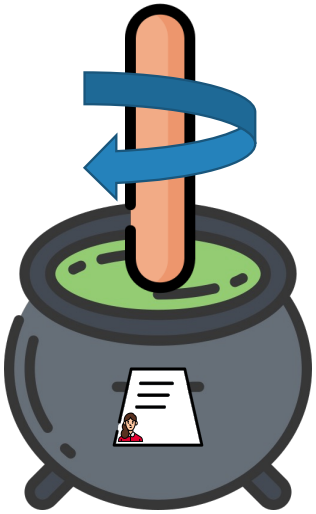
She proceeds to make a copy of it...

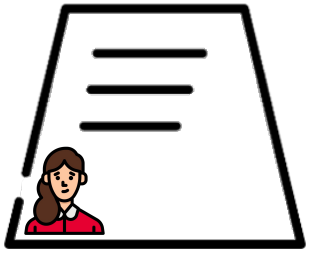


...and shoves it into a cauldron

How do we legitimate a certificate ?

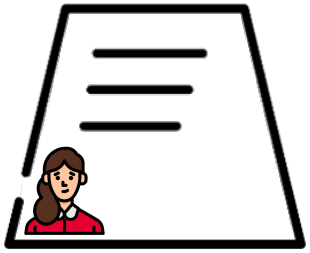
A big nice stir is required to make it homogeneous



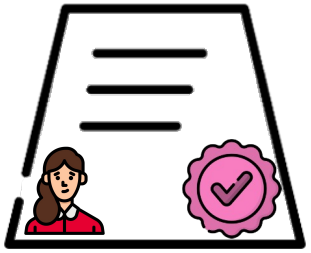


How do we legitimate a certificate ?

She then casually adds a drop of her blood to the mixture



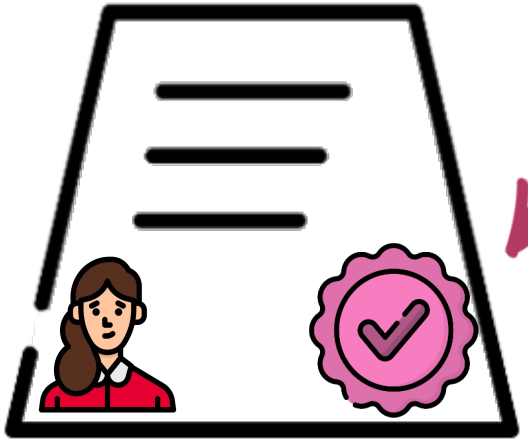
How do we legitimate a certificate ?



Finally she uses the mixture to create a dry seal

The certificate is now authentic

How do we legitimate a certificate ?

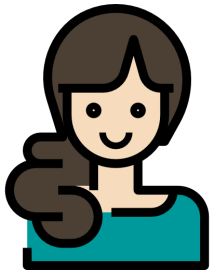


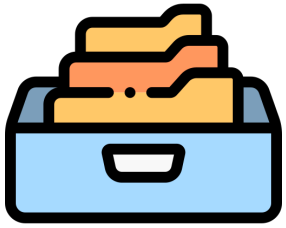
This is the signature of Alice.

As long as noone obtains a drop of her blood, noone will be able to produce an authentic seal.

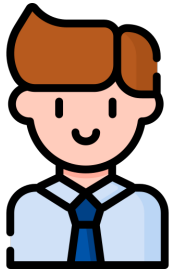
Certificate authenticity

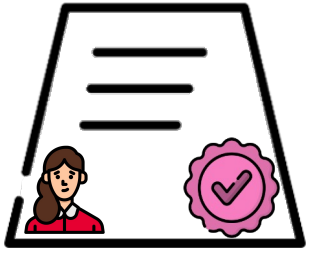
Did Alice just say “Covfefe” ?



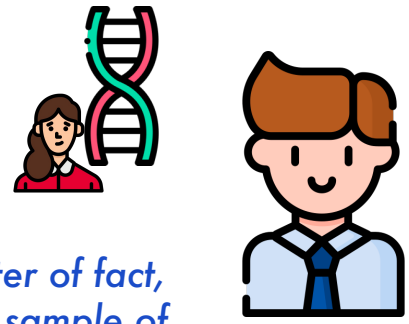


Well… There sure is a seal on her statement

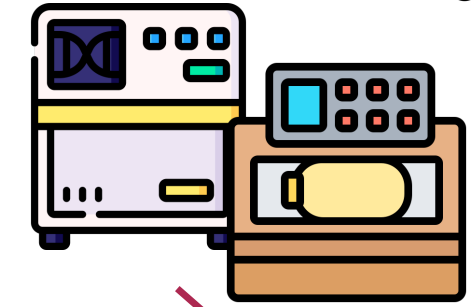




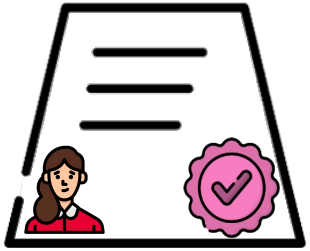
How do we check for authenticity



1. As a matter of fact, Bob owns a sample of Alice's DNA



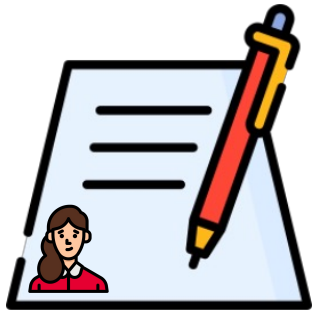
2. Bob runs a sample of the seal through DNA matching process



3. The test is a success : Alice did issue this certificate

Can we counterfeit a certificate ?

I love eating pasta with Nutella

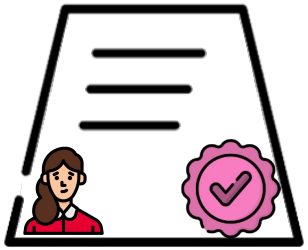


Charlie adds again Alice's name

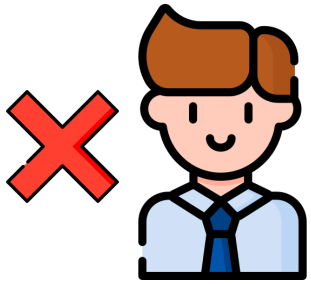
"Certificate"

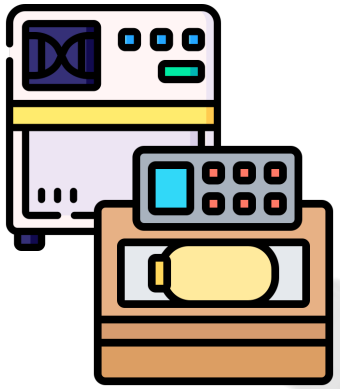
Can we counterfeit a certificate ?

He then adds a drop of HIS OWN blood

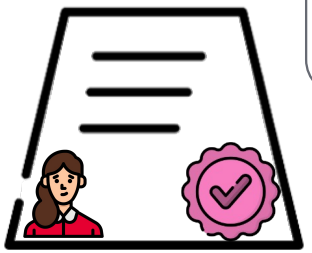


Can we counterfeit a certificate ?



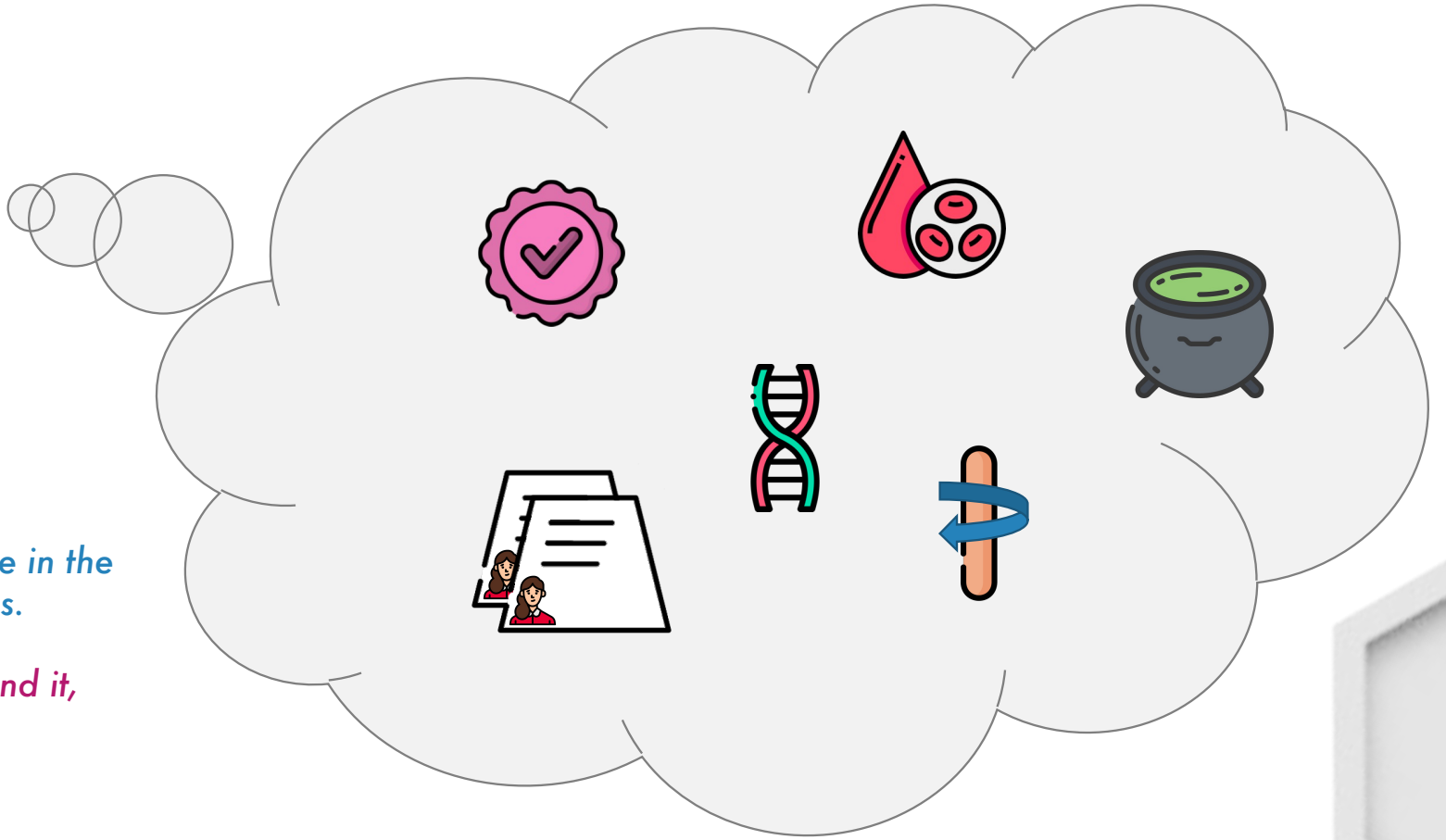


Wait… She does not like pasta with Nutella !



Can we counterfeit a certificate ?

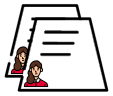
Takeaway



Different actors intervene in the authenticity process.

But what is really behind it, technically ?

Analogy

	Public Key	Represents the public identity of an individual
	Private Key	An equivalent of a password, that helps sign a certificate
	Payload	Contains all the relevant data, that needs to be trusted
	Hashing	One-way mathematical function that collapses data into a unique object
	Signature	Authenticates the identity of the issuer

Hexadecimal representation

Base 10 : $1\ 3\ 9\ 7\ 2\ 9\ 8\ 0$

$0\ to\ 9 \qquad 0\ to\ 9$

Base 16 : $D\ 5\ 3\ 5\ F\ 4$

$0\ to\ 16 \qquad 0\ to\ 16$

$0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$

$a(10)\ b(11)\ c(12)$

$d(13)\ e(14)\ f(15)$

Hexadecimal representation

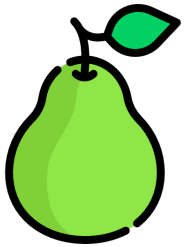
Base 10 : 1 3 9 7 2 9 8 0

$$
\begin{aligned}
1 * \quad & 1\,0\,0\,0\,0\,0\,0\,0 \\
+3 * \quad & 1\,0\,0\,0\,0\,0\,0 \\
+9 * \quad & 1\,0\,0\,0\,0\,0 \\
+7 * \quad & 1\,0\,0\,0\,0 \\
+2 * \quad & 1\,0\,0\,0 \\
+9 * \quad & 1\,0\,0 \\
+8 * \quad & 1\,0 \\
+0 * \quad & 1
\end{aligned}
$$

Base 16 : D 5 3 5 F 4

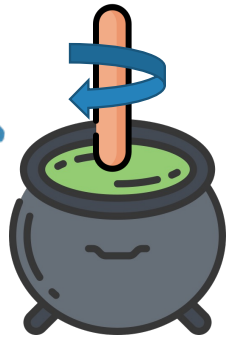
$$
\begin{aligned}
+13 * \quad & 1\,0\,4\,8\,5\,7\,6 \\
+5 * \quad & 6\,5\,5\,3\,6 \\
+3 * \quad & 4\,0\,9\,6 \\
+5 * \quad & 2\,5\,6 \\
+15 * \quad & 1\,6 \\
+4 * \quad & 1
\end{aligned}
$$

Hashing



A pear

An explicit representation of the pear (color, size, ...)



We give a good stir
=
We use a mathematical function to shake the bits

$9ce0255a$
$8bcdd32e$
$9c6ab17$
...
-1022893756

A hash of the pear (string, int, whatever)

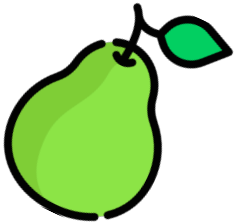
Hashing (in short)

Literally ANYTHING

$9ce0255a8bcdd32$

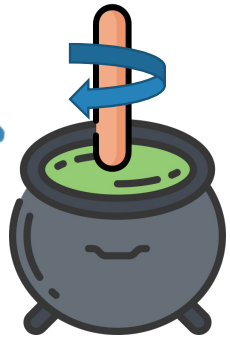
Why is it so good ?

Hashing – Very sensitive



A rotated pear

An almost identical representation



We give a good stir
=
We use a mathematical function to shake the bits

$ff48973e2$
$0b00df92$
$3e06fac1$
...

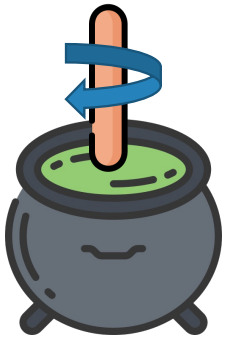
478274

A very different hash

Hashing – One way

No pear !

We can't have all atoms of the pear in our representation

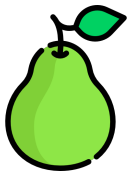


We cannot "unstir" the mixture

$9ce0255a$
$8bcdd32e$
$9c6ab17$
...
-1022893756

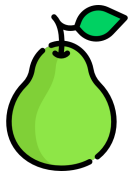
A random hash

Hashing – Fast comparison

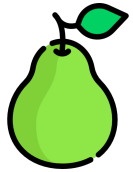


−1022893756

478274



−1022893756

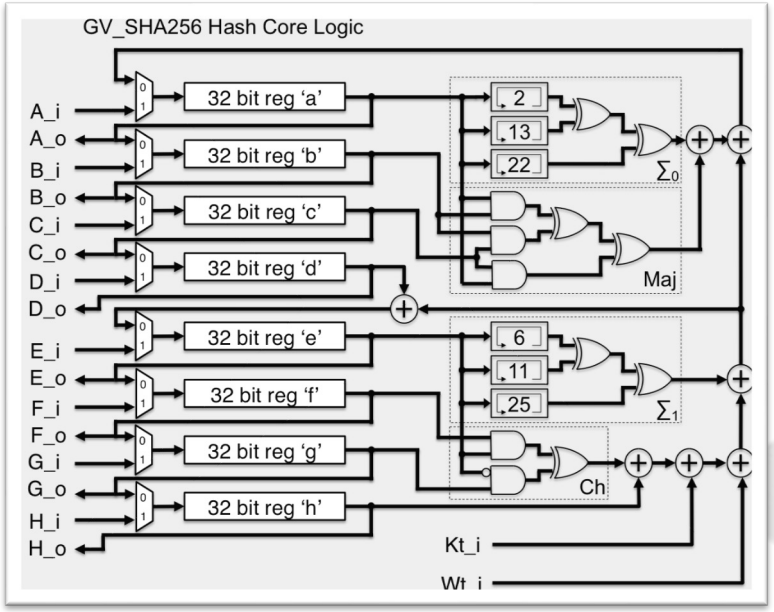


−1022893756

Hashing – Algorithms

```python
# Fast hash using XOR operator

def simple_hash(inputString):
    output = 0
    for character in inputString:
        output += ord(character)
        output ^= (output * 31)
        output %= 2 ** 32
    return output
```



TD 1

SHA-256

Private / Public keys

Private key

Public key

Linked by a mathematical process

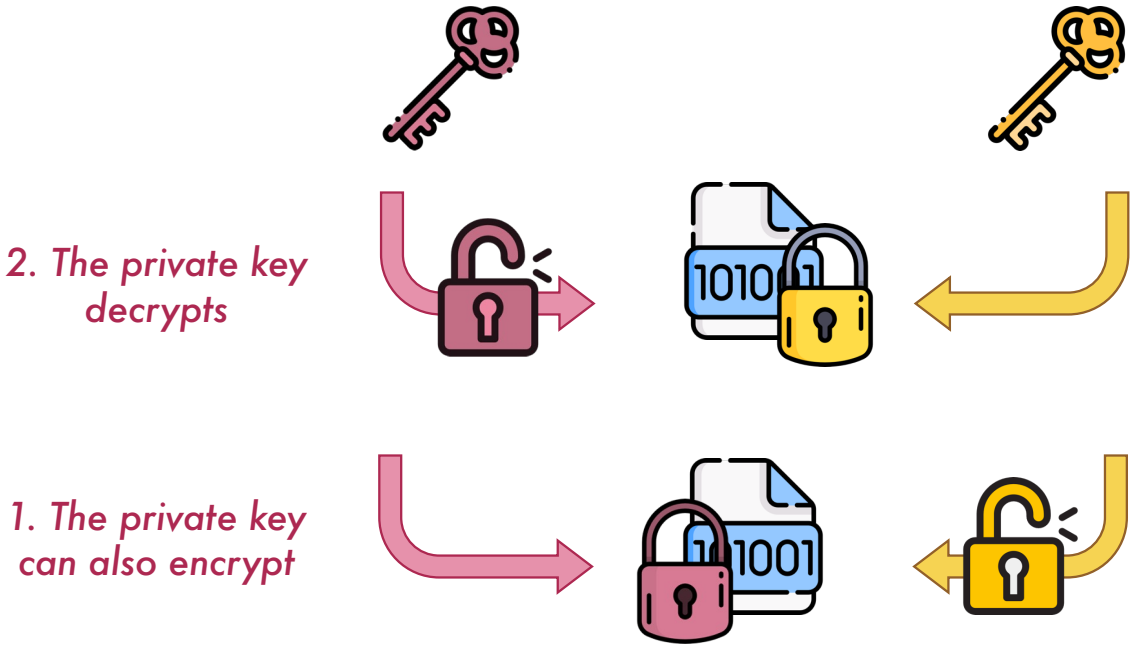
Should remain hidden

Can be shared

The public key can be obtained from the private key

The private key CANNOT be obtained from the public key

Private / Public keys



RSA (1978)

Ron Rivest, Adi Shamir et Leonard Adleman

- p and q two very large prime numbers
- calculate $n = p * q$
- calculate $\phi(n) = (p-1) * (q-1)$

Euler's totient function

- find e prime with $\phi(n)$
- find d inverse of e modulo $\phi(n)$

$(d, \phi(n))$

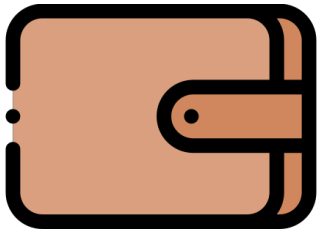
Private key

(e, n)

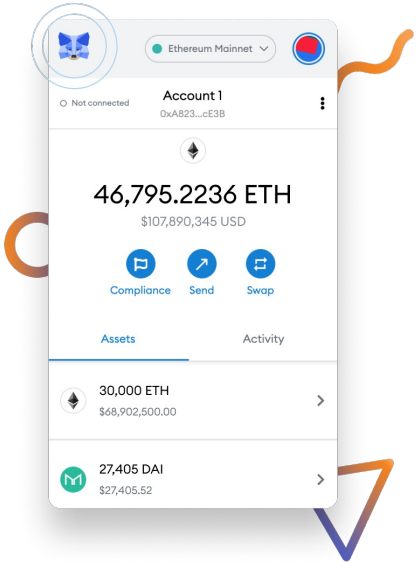
Public key

Wallet

A "wallet" is just a location that stores public/private key pairs



LEDGER



Signature (encrypt)

$H = 93827489$

$(d, \phi(n))$

(e, n)

$S = H^d mod\ n$

Freshly produced signature

Signature (decrypt)

$H = 93827489$

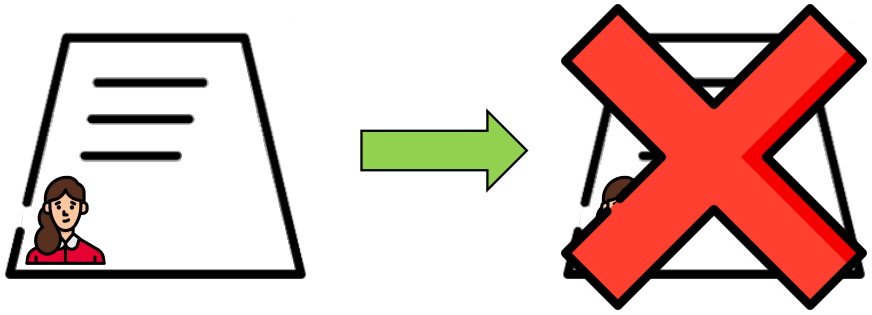
$(d, \phi(n))$

(e, n)

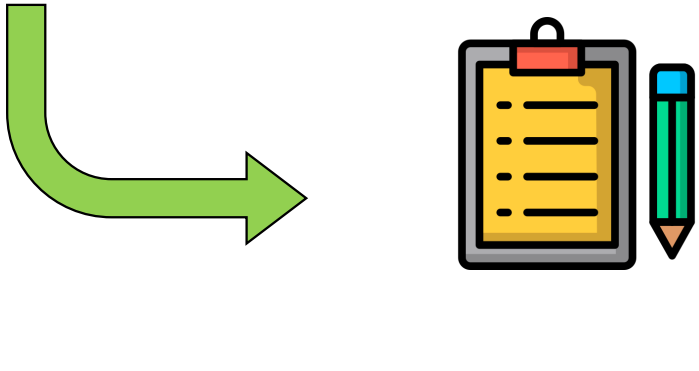
$$H == S^e \bmod n$$

The public key belongs to the owner of the private key that signed this hash

Payload



A copy of the certificate

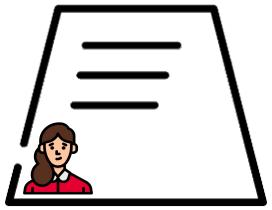


A summary of the key features of the certificate

Issuer, Date of creation, important data, ...

Summary

Alice's certificate



Certificate's payload

Certificate's hash

$H = 93827489$

Alice

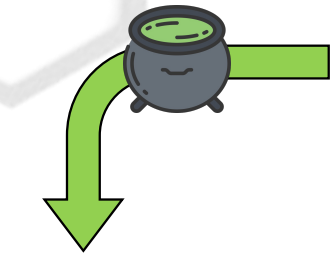
$(d, \phi(n))$

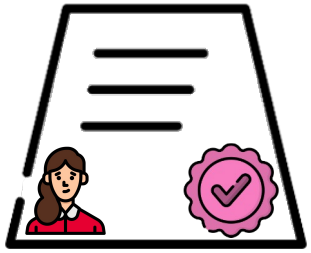
Alice's private key

$S = H^d mod\ n$

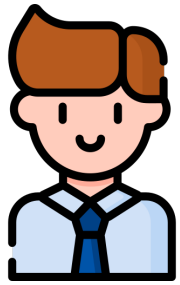
Certificate's signature

Summary





Bob



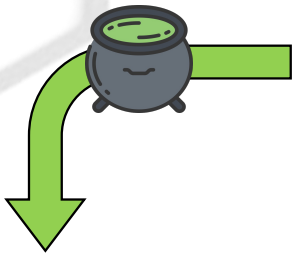
$H = 93827489$

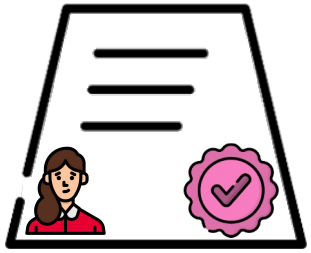
(e, n)

$H == S^e mod\ n$

Certificate is authentic

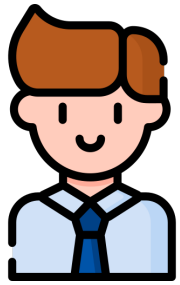
Summary – Key point





Charlie changes one data in the certificate

Bob



$H = -748779$

The hash changes

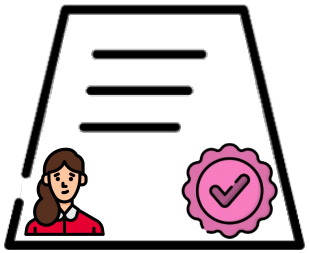
(e, n)

$H \neq S^e mod\ n$

Certificate is no longer authentic

In reality – blockchains

What's inside a certificate



Issuer : `4dd04826f82a33b…` — *Public key*

Date : `178093038494` — *Timestamp*

Data : `{`

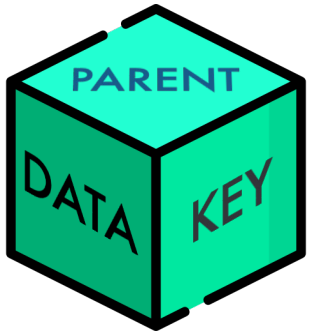
`…` — *Anything*

`}`

Signature : `c56e345aaa900bc…`

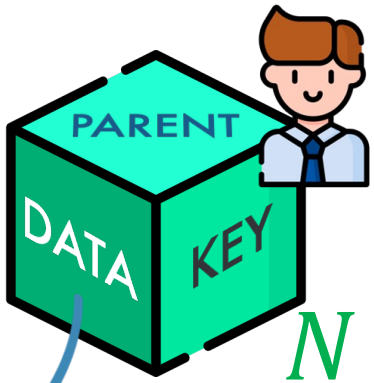
Blocks

What is a block ?



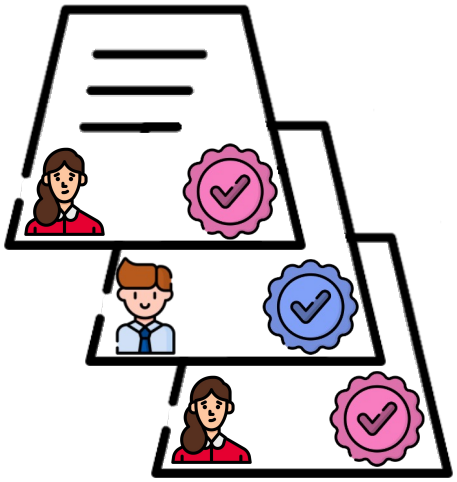
What is a block ? – Data

The block also has an owner (forger)...



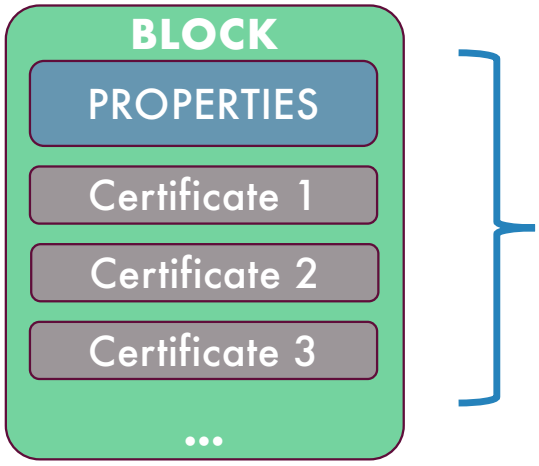
...and an index

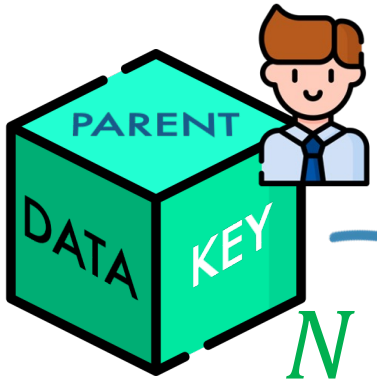
The data is nothing else than a bunch of certificates



And other stuff (date of creation, nonce, ...)

What is a block ? – Key





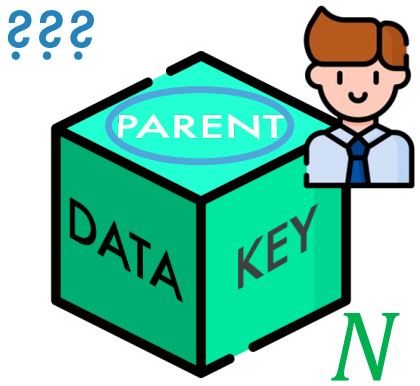
N

The key is just the block's hash...

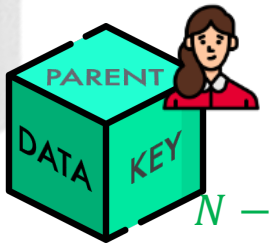
$H = -7200548$

...that is obtained using the payload technique

What is a block ? – Parent



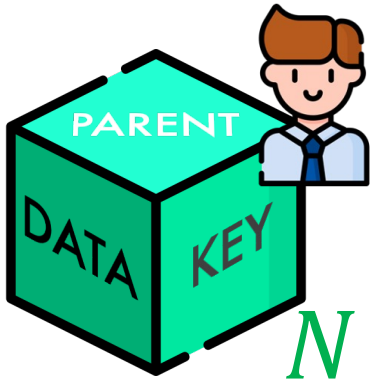
What is a block ? – Parent



$N-1$

$H = 8764327$

We store the parent's hash



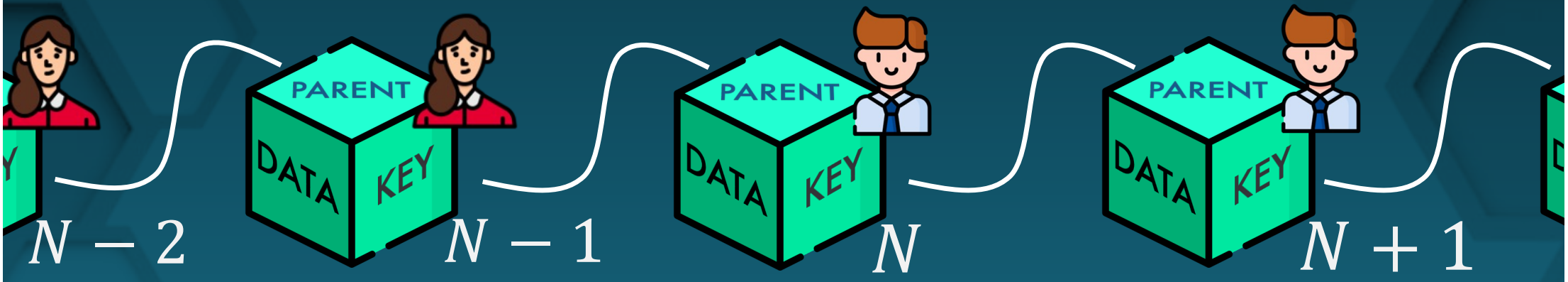
N

Let's see shy it matters...

Blockchains

Blockchains

$PARENT_N$= 84938

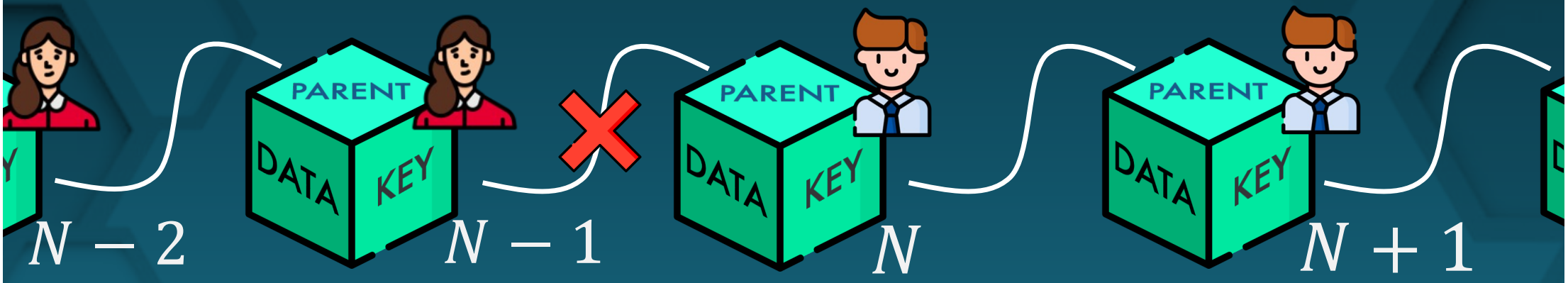


H_{N-1}= 84938

Blockchains

Parent points to nothing

$PARENT_N$= 84938



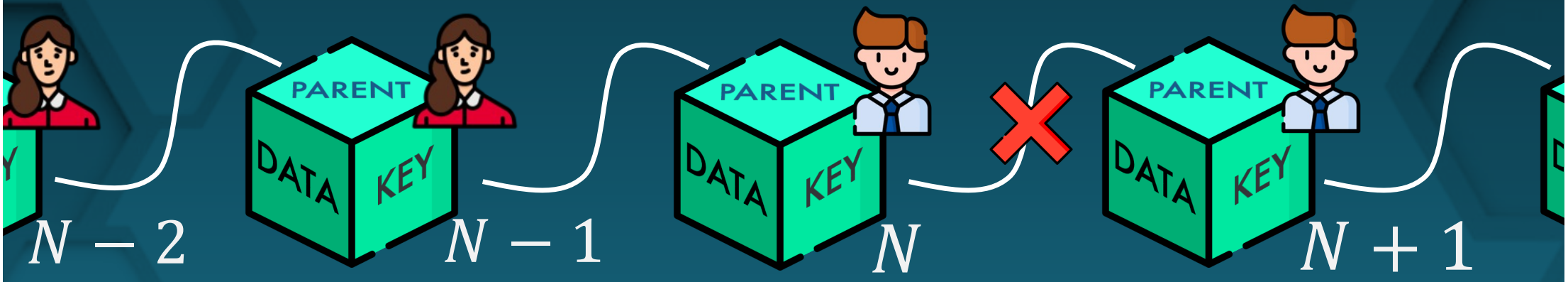
H_{N-1}= 3356

Charlie tampers with block N-1, thus changing its hash

Blockchains

$PARENT_N = 3356$

But block N+1 then points to nothing too



$H_{N-1} = 3356$

Charlie must also change block N to meet parent's hash

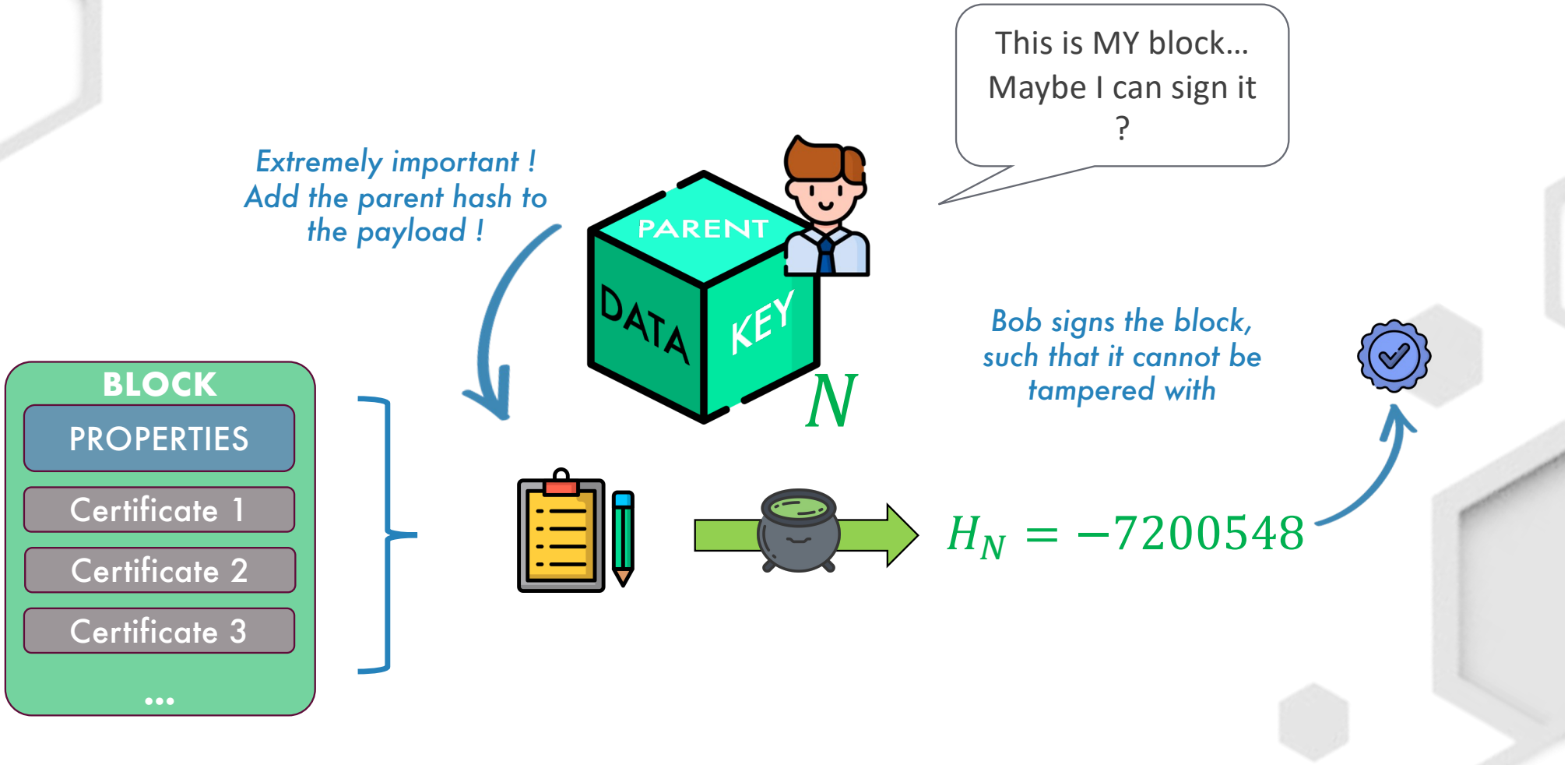
Blockchains

So I just need to change all following blocks to have a healthy blockchain again...

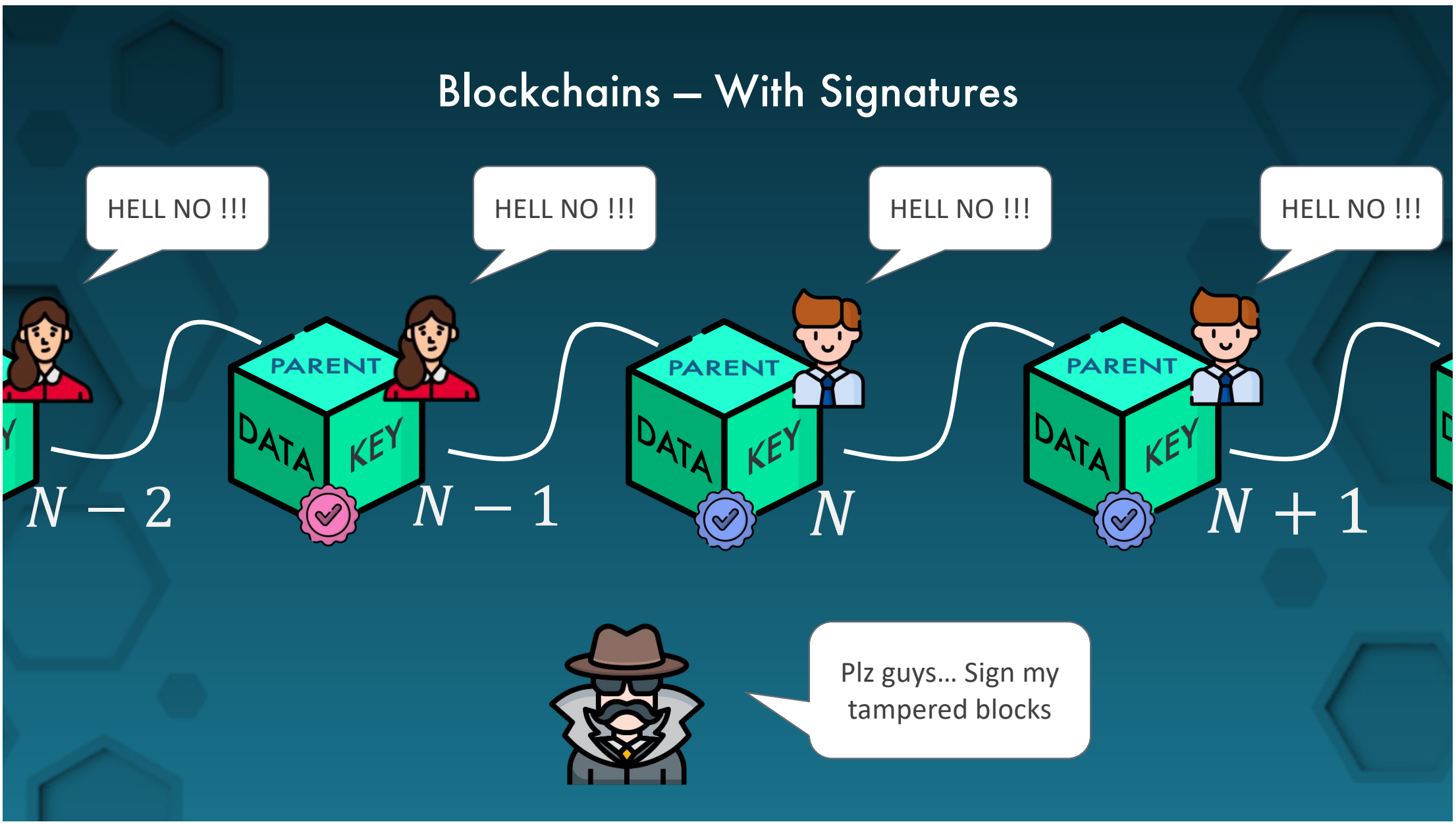
...Right ?

Well YES... But no !

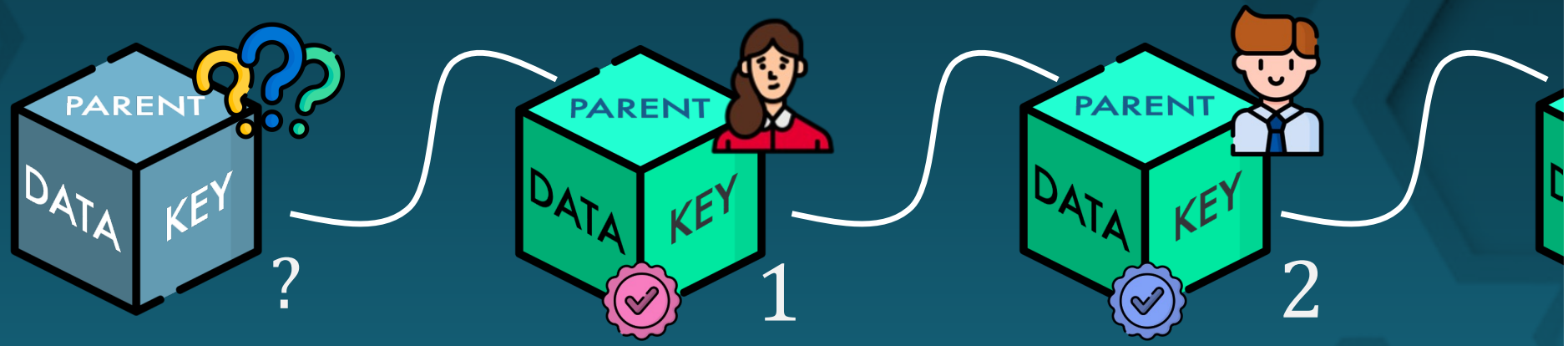
What is a block ? – Signature



Blockchains – With Signatures



Blockchains – Genesis Block



Digital identity

(e, n)

(e', n')

Alice and Bob both have their own wallet, which is like their digital identity

Digital identity

What it actually looks like...

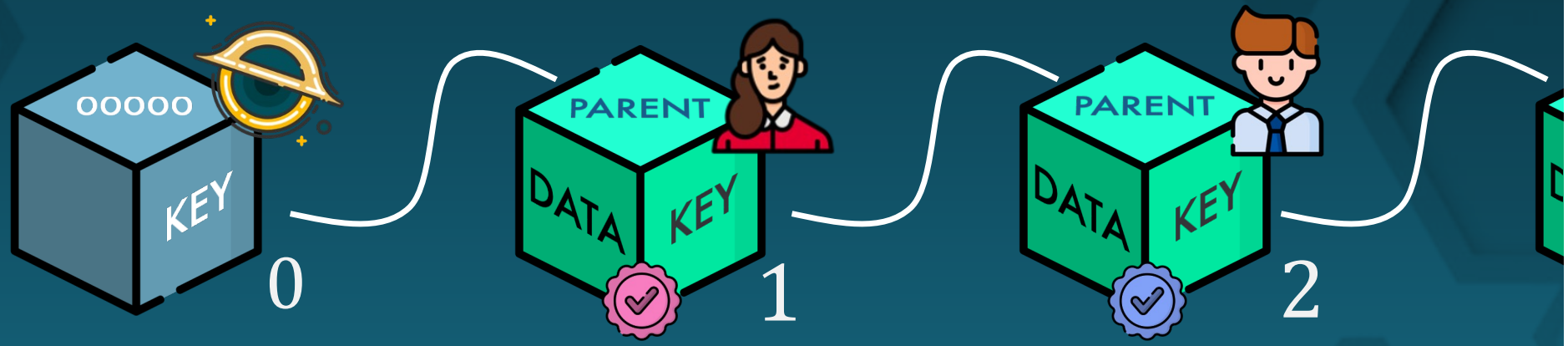
$e63ac0c281ed24cc576cdfc012...$

$4f15ce6e5f459a56274788d22...$

$0000000000000000000000000...$

The Black Hole owns the "zero" key

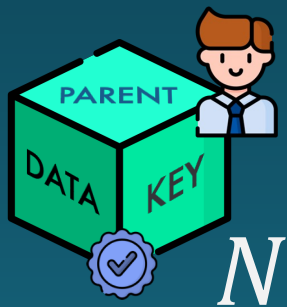
Blockchains – Genesis Block

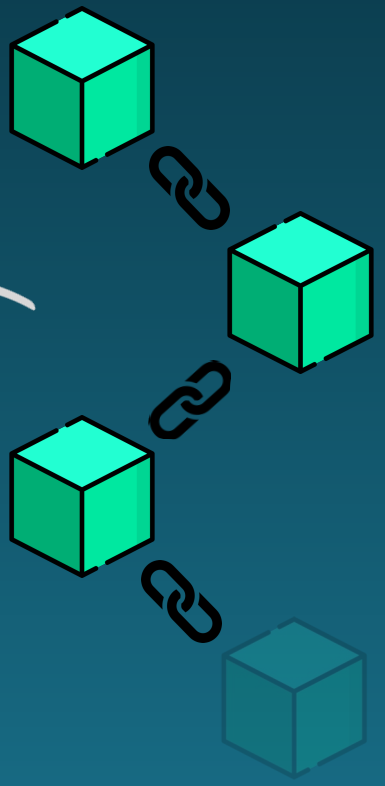


Arbitrarily, the black hole owns the genesis block

Checking for corruption

Alice checks blocks one by one





Signature is valid

Parent is valid

No corrupted certificates

...

Security

Is it 100% safe ??

Security

No conceptual flaw

Code is a weakness

Bitcoin Inflation bug 2010

Practical example

Cryptocurrencies

Transactions

I want to give Bob 2 Bamboozloos

WTF is even that ???

Transactions

Delphine creates a certificate for this transaction, and signs it

Issuer : Delphine

Recipient : Bob

Amount : 2

Thanks… I guess ?

Cryptocurrencies

2 Bamboozloos

5 Bamboozloos

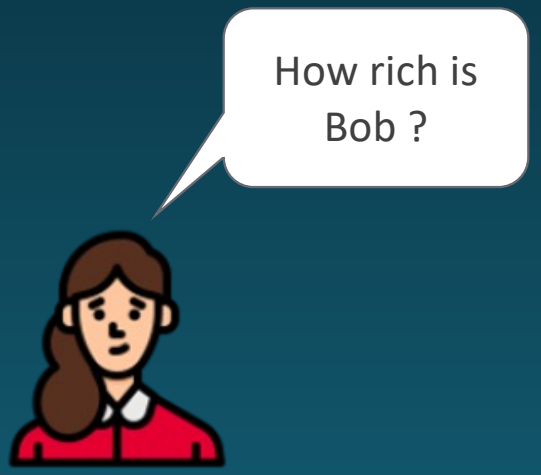
3 Bamboozloos

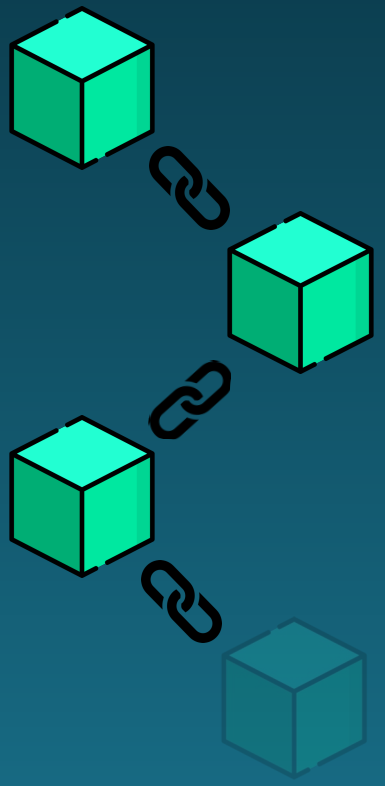
-2

4

-2

Balance





Alice checks blocks one by one and sums all transactions involving Bob to calculate his Bamboozloo balance

Transaction Details

Overview

Transaction Hash:	*0x6f9fb01022150eca37db772fc9b9a2444612bd7344b7e66b1690d8fe1c91deea*
Status:	Pending
Block:	(*Pending*)
Time Last Seen:	00 days 00 hr 00 min 25 secs ago (Apr-14-2021 11:59:17 PM)
Estimated Confirmation Duration:	< 45 secs | Gas Tracker
Pending Txn Queue: Beta	0% 100%
From:	0xab41b92c6d43e4b8a670b75879f5bb809646602e
To:	0xa33c8f687250f748cdade9a514b9293dbf49c4a9
Value:	0.037290222 Ether ($90.67)
Max Txn Cost/Fee:	0.002961 Ether ($7.20)
Gas Price:	0.000000141 Ether (141 Gwei)

Weakness

Security

Byte storage limit caused an overflow bug, allowing someone to earn 184,467,440,737 bitcoins

Code is a weakness

Bitcoin Inflation bug 2010

Blockchain et Applications

Quiz 2

Structure de donnée