Scaling Blockchains: from Bitcoin to the Lightning Network

Scheme for Promotion of Academic and Research Collaboration

Davide Patti
davide.patti@dieei.unict.it

Outline

- Physical VS Information Transfers
- Hash Functions, Proof-of-work, and Mining
- **Asymmetric Cryptography**
- Inside Blocks: Transactions
- Tools & Demo: Electrum
- Scaling Blockchains: The Trilemma
- Scaling to upper layers: The Lightning Network
- Open Research Topics and Challenges

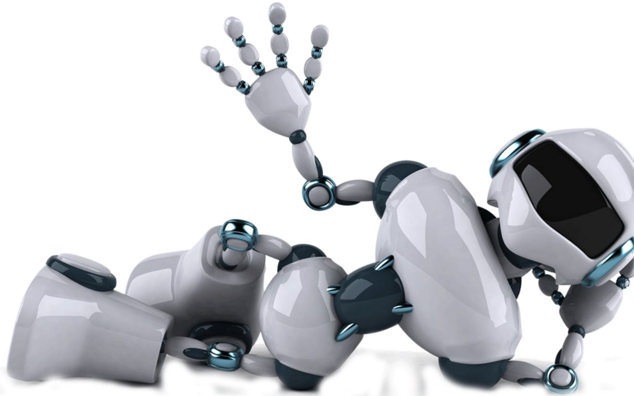
Transactions at Layer 1

Bob wants to send X bitcoins to Alice

1) How do we ensure that only Alice can receive the value?

2) How can Alice be sure that is Bob that wrote the transaction?

Notice: Human-like names are used only for clarity. They could also be non-human, smart objects, Artificial Intelligence entities etc...

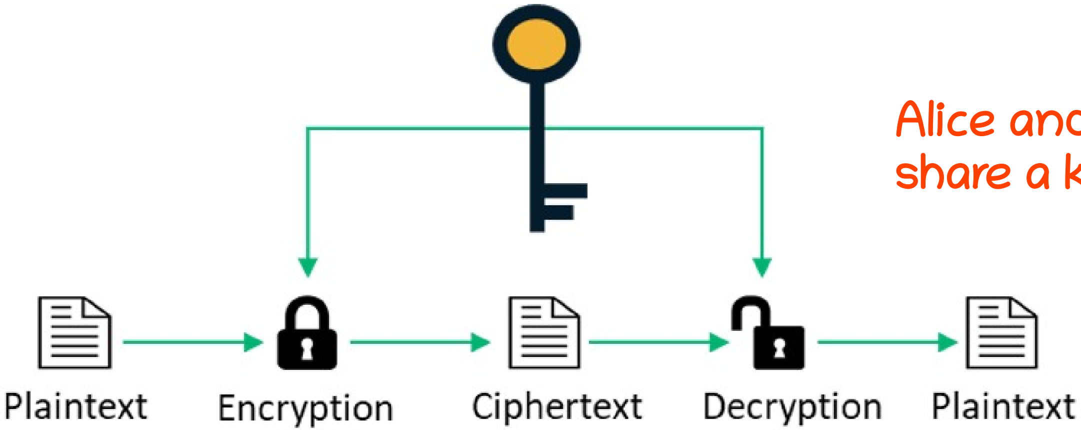


Symmetric Encryption

Each pair of people wanting to exchange messages, must share a secret key

Alice and Bob encrypt messages using the same key

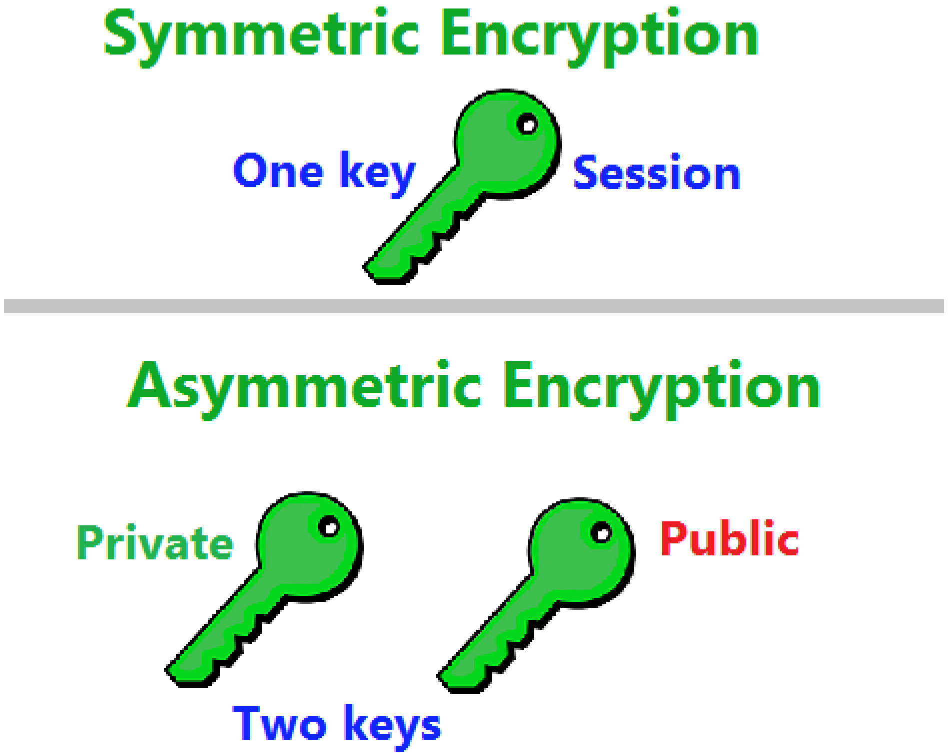
- **PROBLEM 1:** If Alice choose a key, must communicate it to Bob (possible leak)
- **PROBLEM 2:** If Alice wants to send messages to other people, must create a key **for each new person**, in theory, everyone in the world!



Alice and Bob MUST share a key !

Asymmetric Encryption

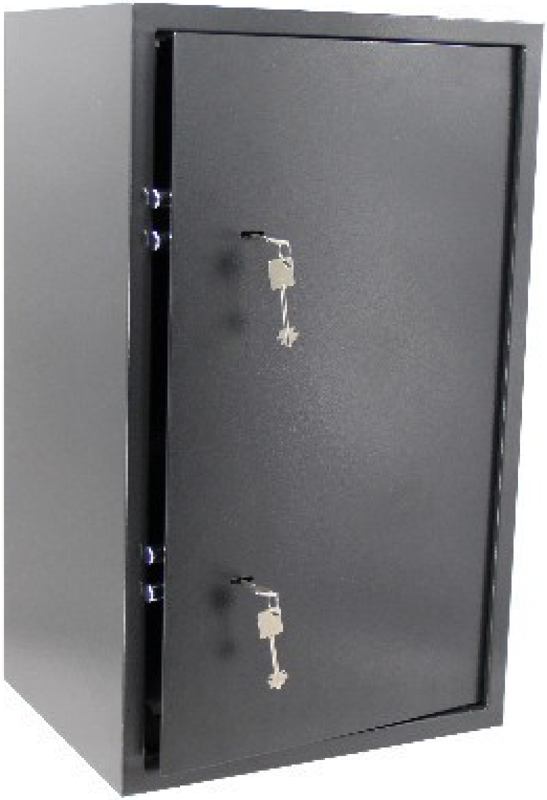
- THERE IS NOT a new key for each couple of people communicating
- **Each user** has two keys:
  - **A public key**, given to everybody. Everyone knows the public key of all other people.
  - **A private key**, kept secret, never shared, never transmitted.



Asymmetric Cryptography

- Imagine a box with two locks:
  - One for the private key
  - One for the public key
- ...and a particular mechanism:

  If you lock with one key, you will able to open the box only using the other key
- Anyone can get the public key of a given user, only the private key is NEVER shared

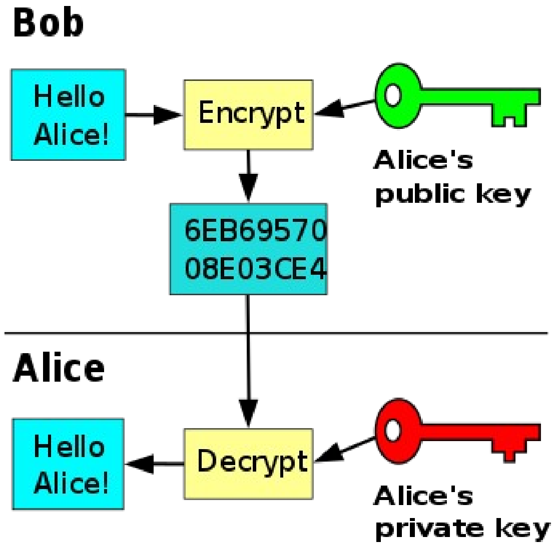


Q1: How do we ensure that only Alice will read the transaction from Bob?

- Bob puts the transaction data inside the box, and closes it with the **Alice's public key**
- Now, it can only be opened with **Alices's private key**
- Only Alice has the private key to open that box.

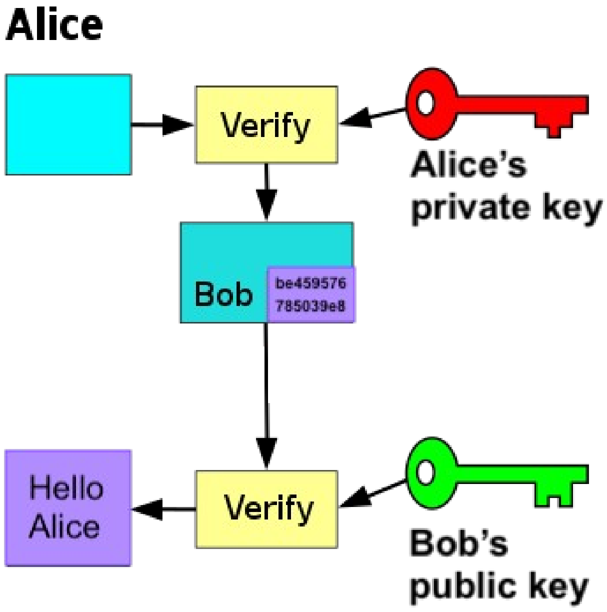
→ **<u>Question 1 solved: Bob created a message that only Alice can read</u>**

Notice: This still not ensure that it was Bob to send it



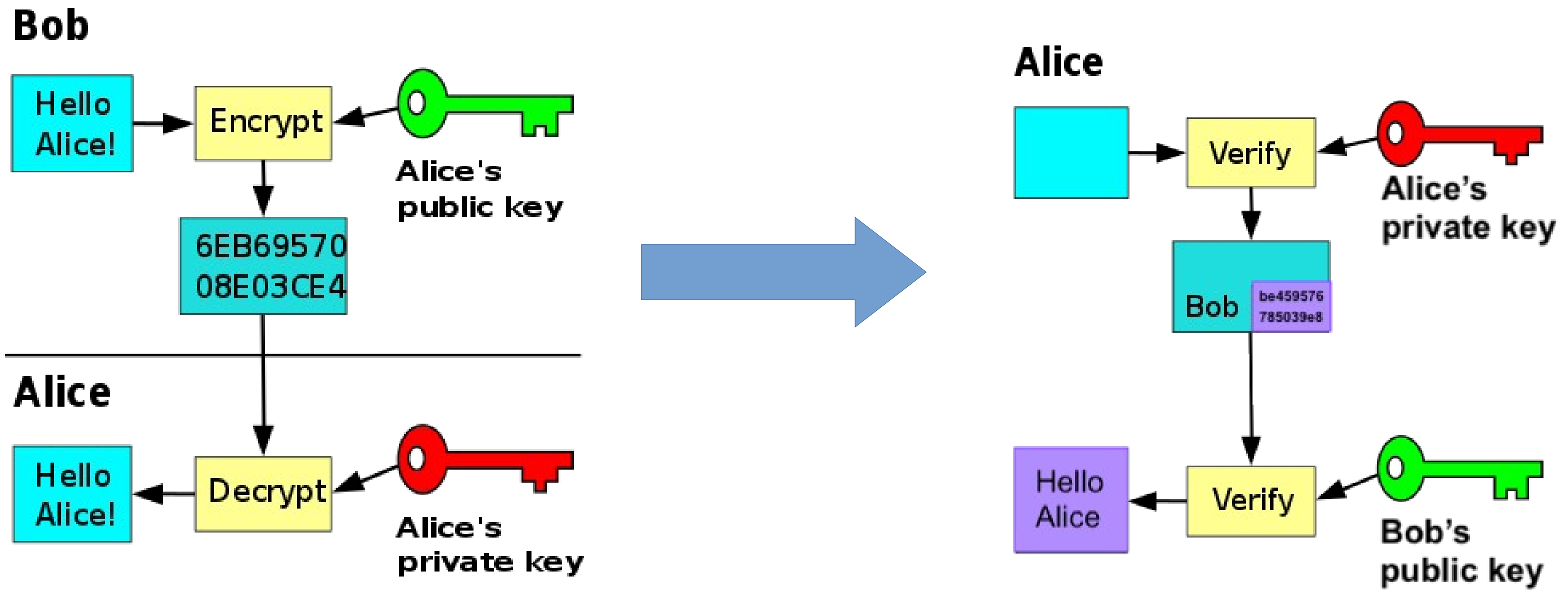
Q2: How can Alice be sure that it was really Bob that wrote that transaction?

- When Alice opens the box with her **Alice private key**, she finds **another box** inside
- But Alice, **like everyone**, has the **Bob's public key**, so he try to open the box she just found.
- If it opens, it means that it was really Bob that sent it, since **only Bob could use the Bob private key to close it.**



Mission Accomplished: Bob sent a message that only Alice can read, **without sharing any secret**

The **external ALICE BOX** could be opened only by Alice with her private key, and the **internal BOB BOX** could have been closed only by Bob with his private key



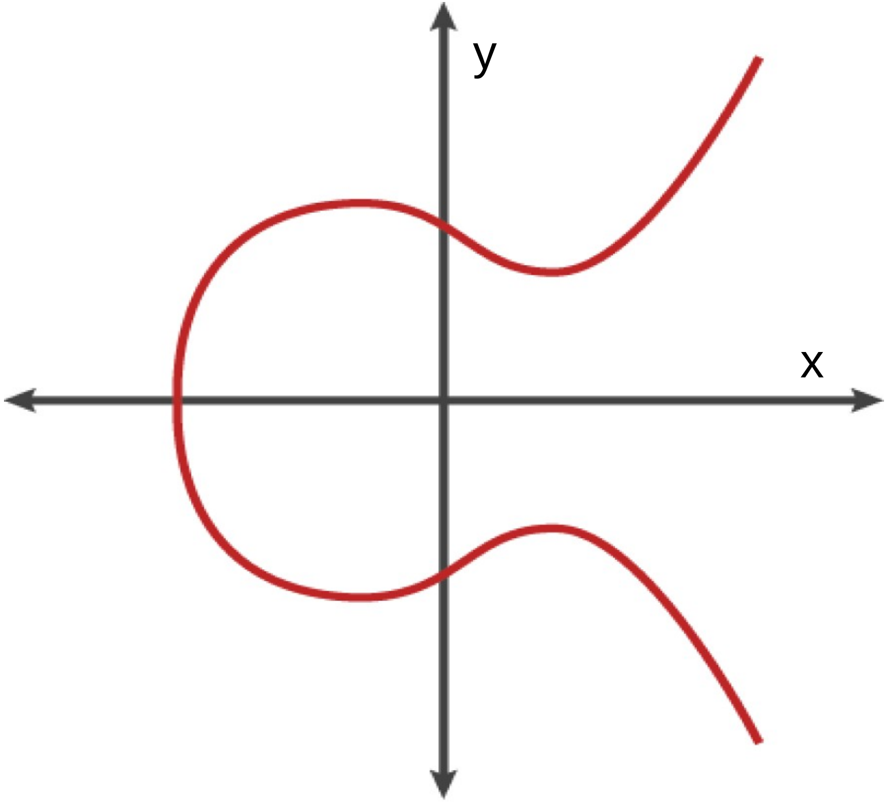
How are those Magic Boxes implemented digitally?

- How can **public key** and **private key** be related without being both revealed?
- In other words: how can I demonstrate to have the **private key** that unlocks was has been sent to my **public key**?
- **But most important: what are those keys?**

It's Time for Fun Math

- Let's consider a number: **n**
- Let's consider a constant number **G** which is the same for everyone and known by everyone
- In traditional math, if I have "**n**" and **G**, it's easy to compute:
  - **P** = **n** * **G**
- Also in the opposite direction:
  - if I give you the result **P**, and ask you to guess which is "**n**", it's very easy: **n** = **P**/**G**
- Thus, in traditional math, the equation **P** = **n** * **G** is reversible

Elliptic Curve Math (secp256k1)

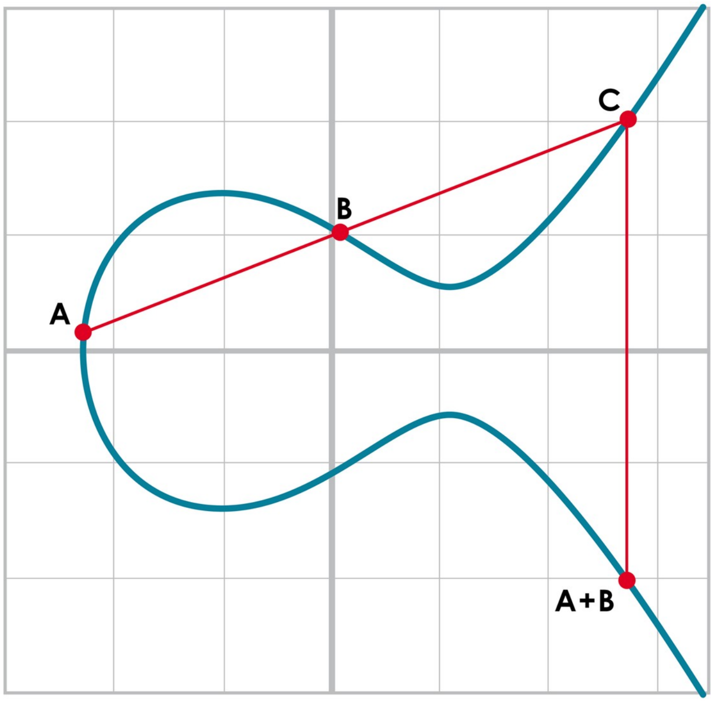


$$y^2 = x^3 + 7$$

Let’s define a new type of Addition

Define an addition operation like this:

- Given two points A and B, trace a line passing by A and B
- The result is the symmetric of the intersection C on the other side of the plane

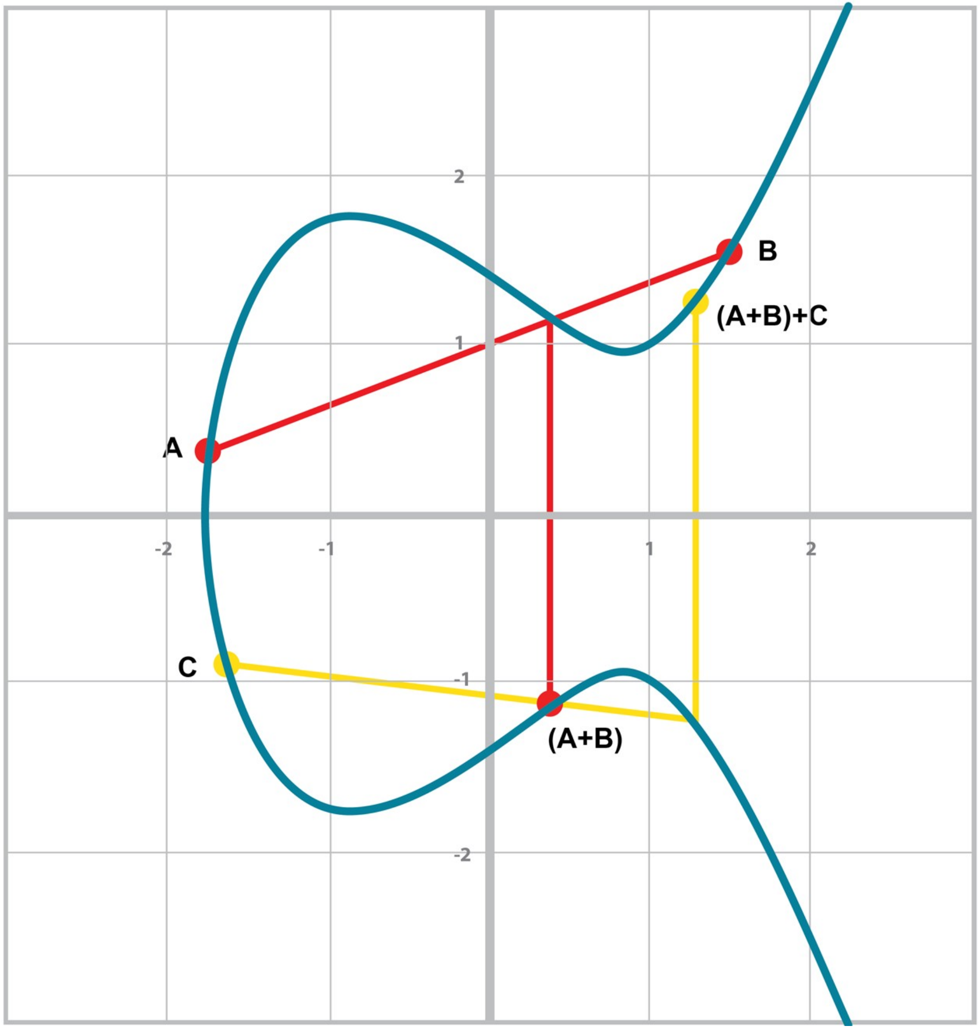


Fun fact: in a so defined addtion, all the traditional known properties still remain!
(commutative, associative, etc…)

Try by yourself:

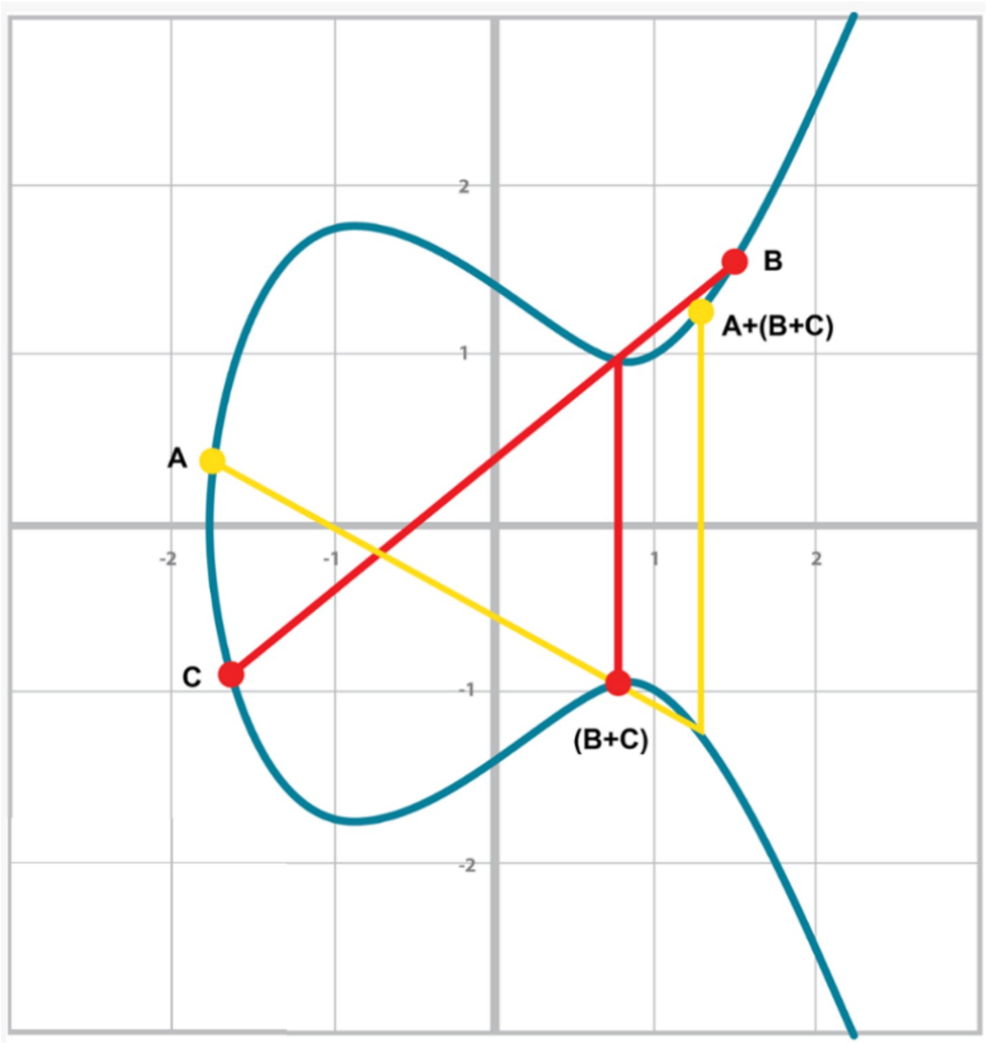
(A+B) + C = A + (B+C) ?

Start with (A+B) then add C



…now do A + (B+C)

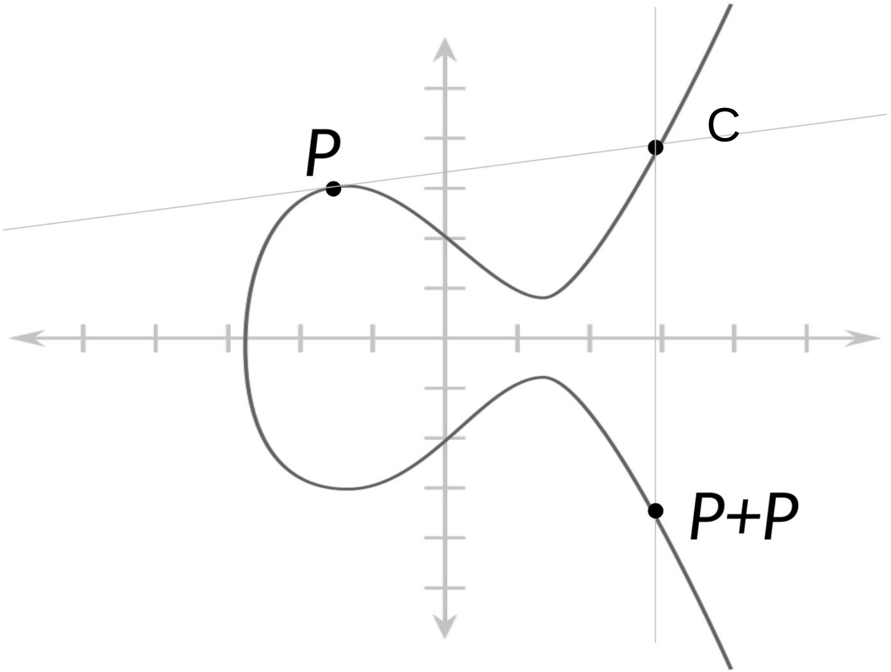
it’s the same point as in the previous slide!



Adding a Point to itself

A is the same as B, they are both in the “P” position, so it’s like:

- plotting the tangent to the curve passing by P
- going to the symmetric point of the intersection C, as usual

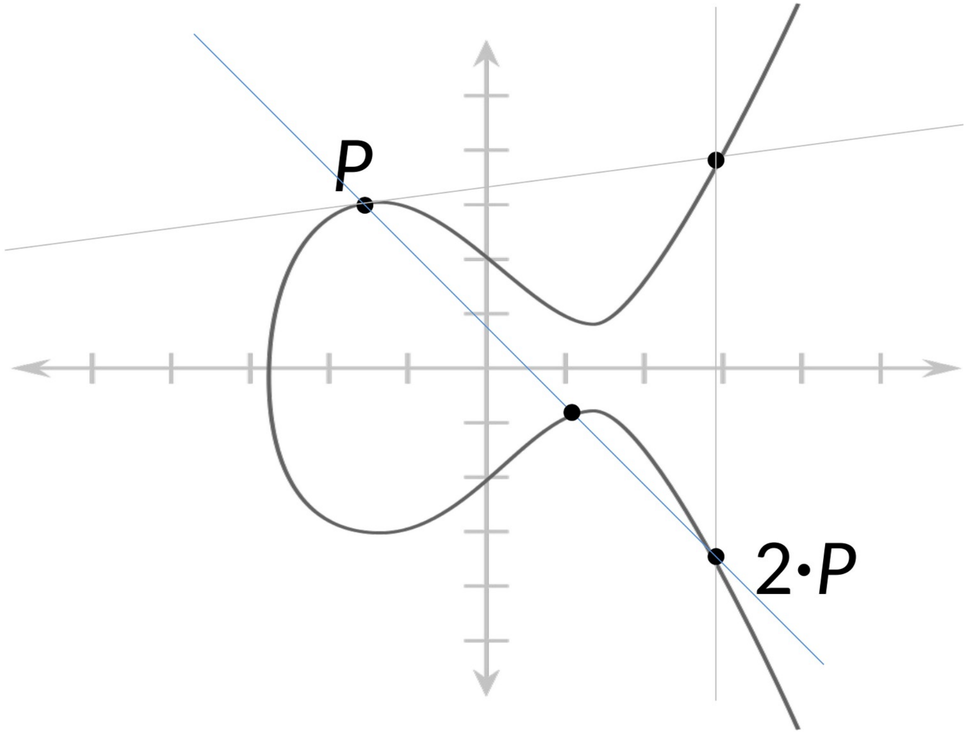


Scalar Multiplication

What does it mean to start from a point P and multiply it by n?

For example, with n = 6:

6* P = P+P+P+P+P+P

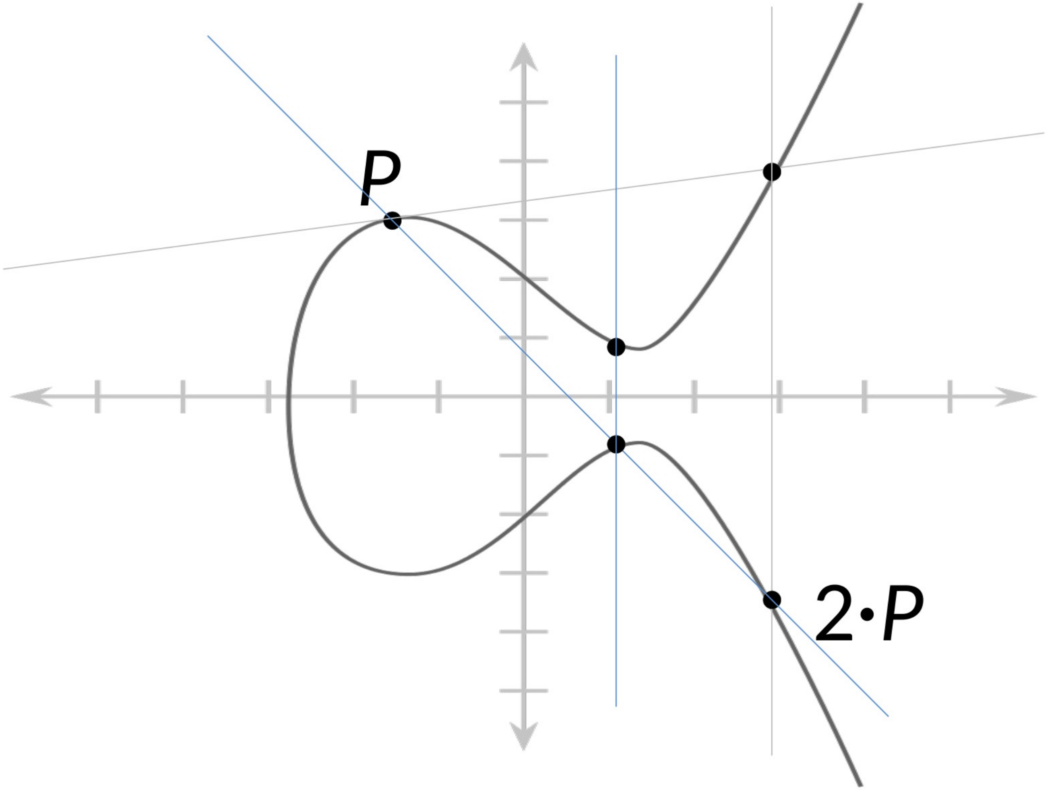


Scalar Multiplication

What does it mean to start from a point P and multiply it by n?

For example, with n = 6:

6* P = P+P+P+P+P+P

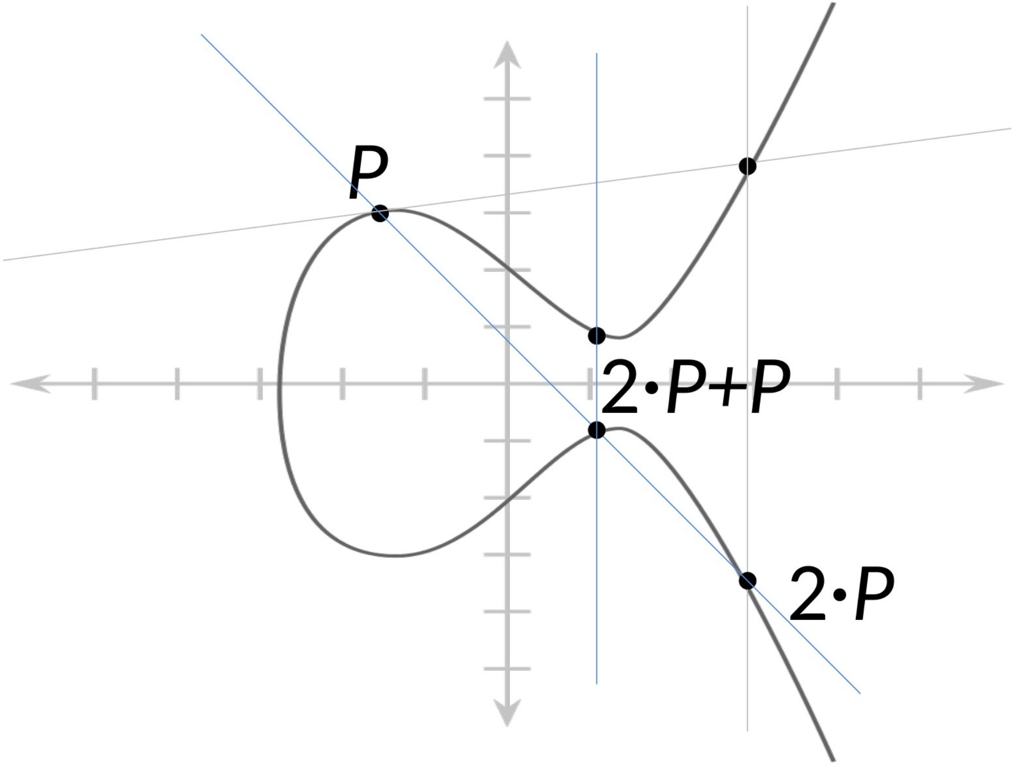


Scalar Multiplication

What does it mean to start from a point P and multiply by n?

For example, with n = 6:

6* P = P+P+P+P+P+P



Scalar Multiplication

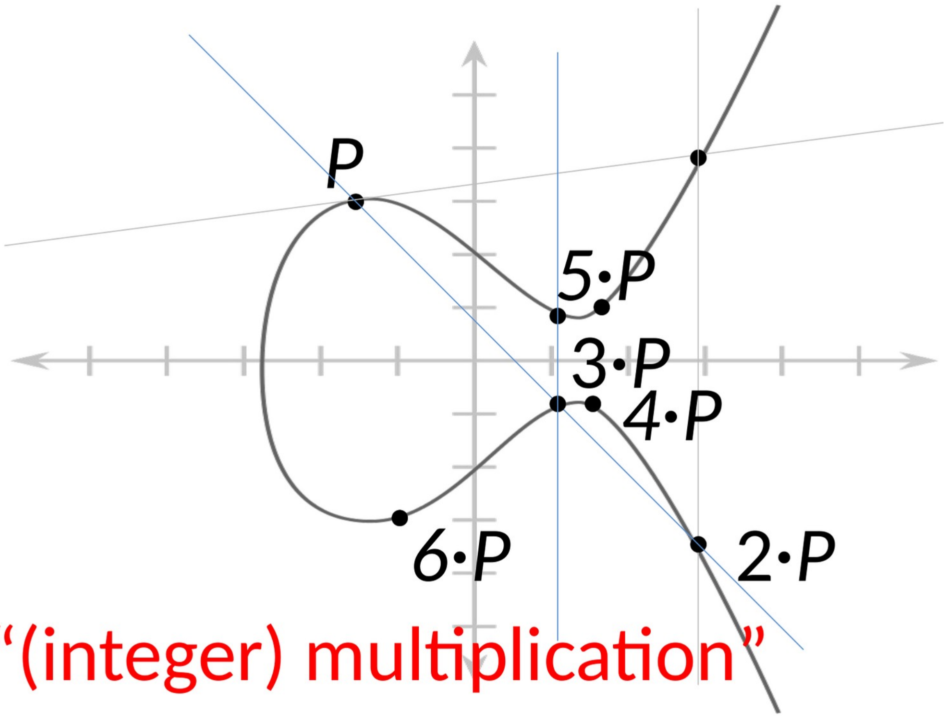
What does it mean to start from

point P and multiply it by n?

For example, with n = 6:

6* P = P+P+P+P+P+P

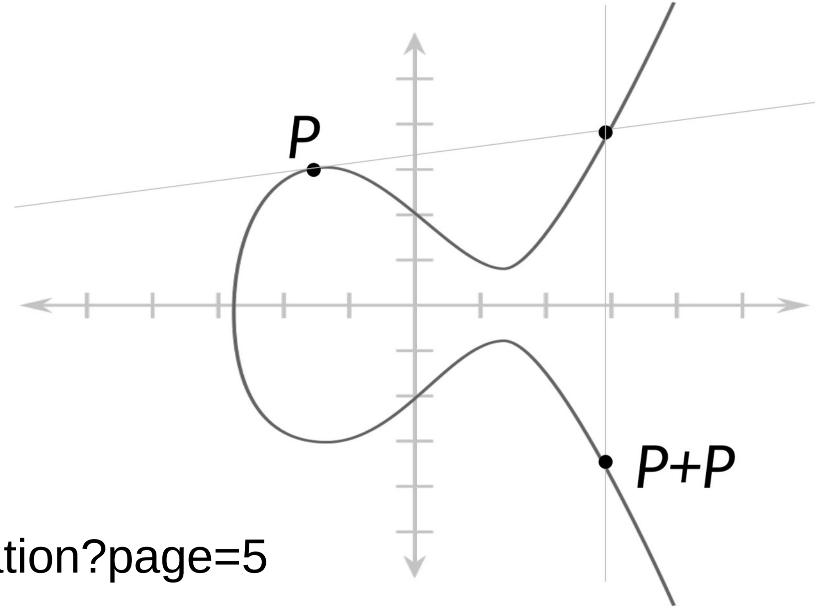
Notice: every step it's easy, but it seems to follow strange path, like a bouncing ball...



Scalar Multiplications shortcuts

Scalar multiplication can be exponentially reduced by using the geometric properties of Elliptic Curve Math (eg, doubling using the tangent)

To go from a point P to 24·P :
P → 2·P
2·P → 3·P
3·P → 6·P
6·P → 12·P
12·P → 24·P

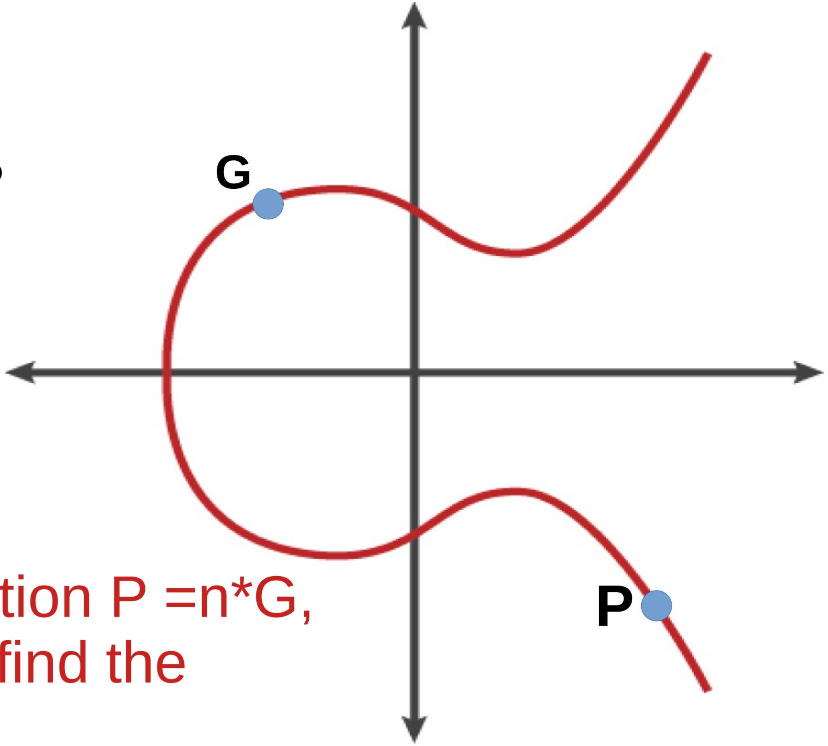


DEMO: try it!
https://www.bitcoinsimulator.tk/explanation?page=5

Inverting of Scalar Multiplication

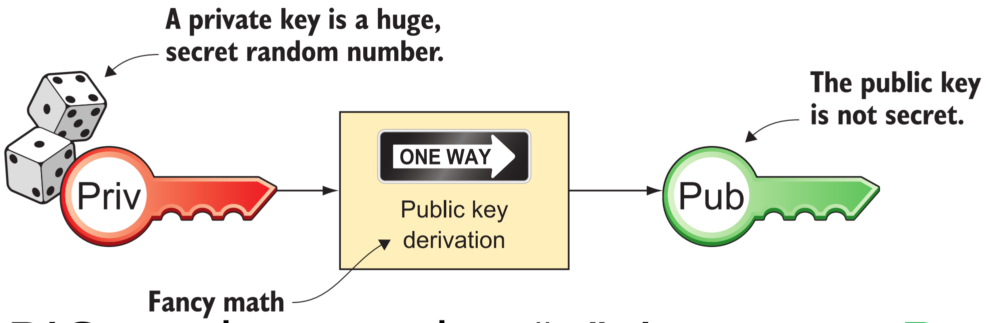
Given a final point **P** and a starting point
Can you guess “n” such that P=n*G ?

the only way to find the number n is to
try P, 2·P, 3·P, etc.



There is no way to reverse the multiplication P =n*G, **except testing every number** until you find the correct n

Private/Public Key Generation

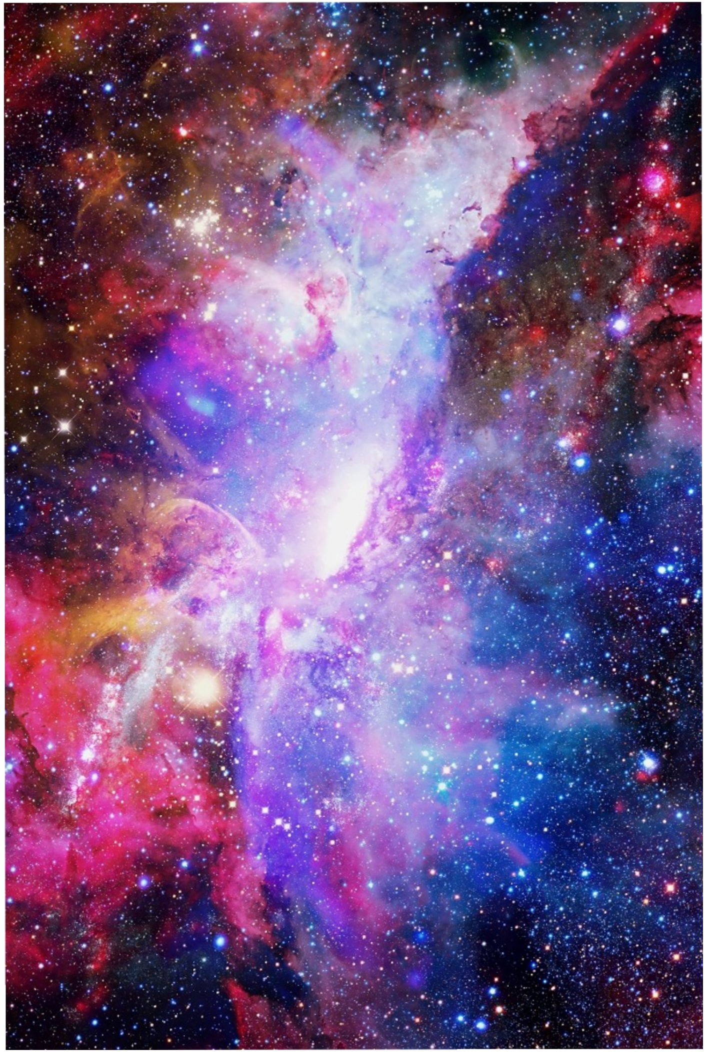


- Starting from a BIG random number "**n**", I compute **P** = **n*G**
- The coordinates of the resulting point **P** will be my **Public Key**
- I'm the only one who knows the "**n**" number, this will be my **Private Key.** Thus, **I'm the only one able to decompose P as n*G**
- All the numbers are 256 bits, so for the other people it is impossible to test every potential "n" to check if **n*G = P**

How Big is 2^256?

To give you an idea, here are some relative scales: **2^256 is about 10^77**

- Number of atoms in the universe ~ 10^80
- A trillion (10^12) computers doing a trillion computations every trillionth (10^(–12)) of a second for a trillion years is still less than 10^56 computations.
- Think of finding a private key this way: **there are as many possible private keys in Bitcoin as there are atoms in a billion galaxies.**



Secp256k1 Generation Point

Gx =

0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798

Gy =

0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8

Owning is an Abstraction

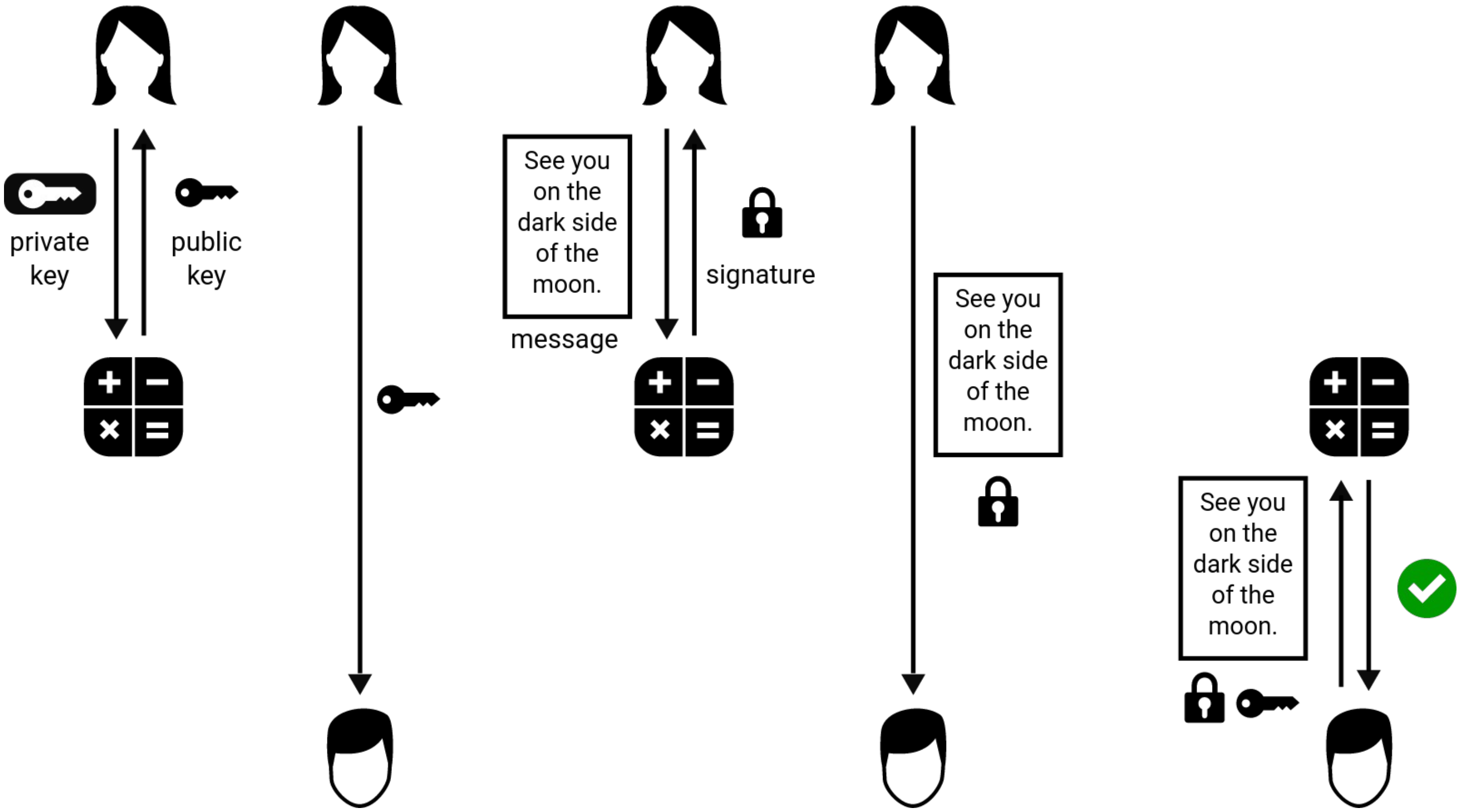
Since a destination **public key P** is always known, and the **G** point is fixed to a constant...

=>

...owning bitcoin means being in possession of a number n of 256 bits so that P=n*G, where P is your public key

You NEVER trasmit **n**:you only prove that you know it

How can Alice demostrate to have here private key?



Schnorr Signatures

n : Alice private key
P=**n*G** : Alice public key
m: message to sign
R = **r***G (**r** is a number, chosen randomly every time)

Alice wants to demostrate that some message **m** has been written by her, the owner of the private key **n**

Defining the “challenge”

Let’s consider the hash of these concatenated values:
$\mathbf{e}$=Hash(R,P,m)

Bob already knows **P** and **m**. **R** will be communicated by Alice before starting.

So the hash value “e” is a way to fix the “challenge”, so that none of them can be changed later

Proof “s” Calculation

Alice secretly calculates **s**=**r**+**e***n** and gives “**s**” to Bob

Notice:
Bob cannot understand how “s” has been constructed, because is missing **r** and **n**!

Verification of the Proof "s"

- Bob computes s***G**
- ...if it was true that Alice created **s=r+e*n** then s***G** = **G***(**r**+**e***n**)** = **R+e*P**
- **Important:** Bob has **P R** and **e**
- So, if s***G** = **R**+**e***P** ,** then Bob can be sure that the sender had the private key **n**

Why also include the random “r” ?

Proof will be:

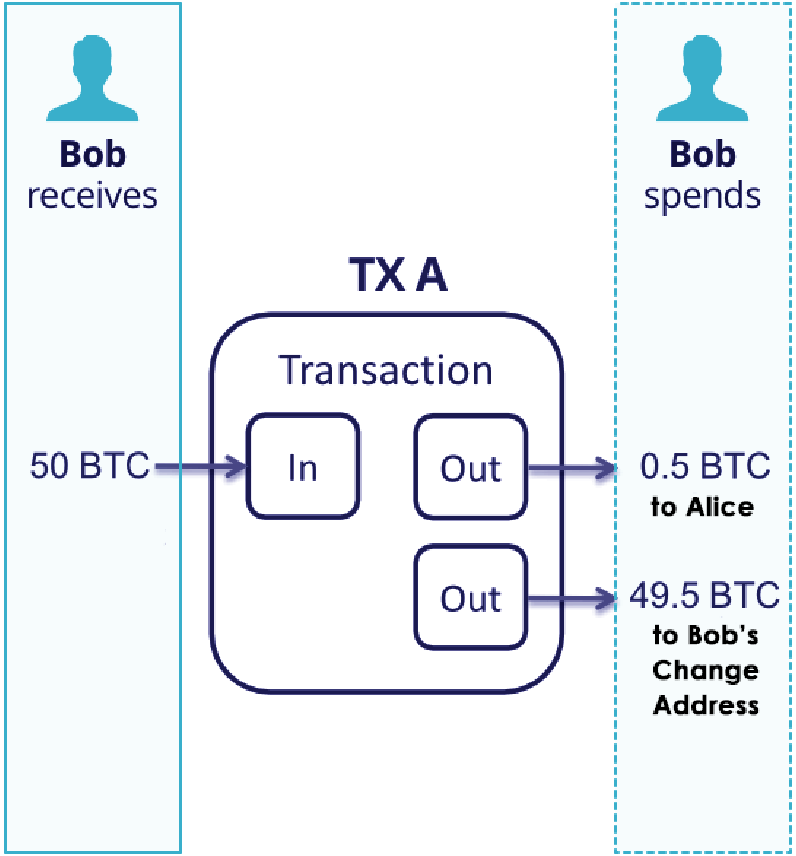
s***G** = **G*e*n** = **e*P**

...but from the proof you get **n=s/e**

https://popeller.io/schnorr-basics

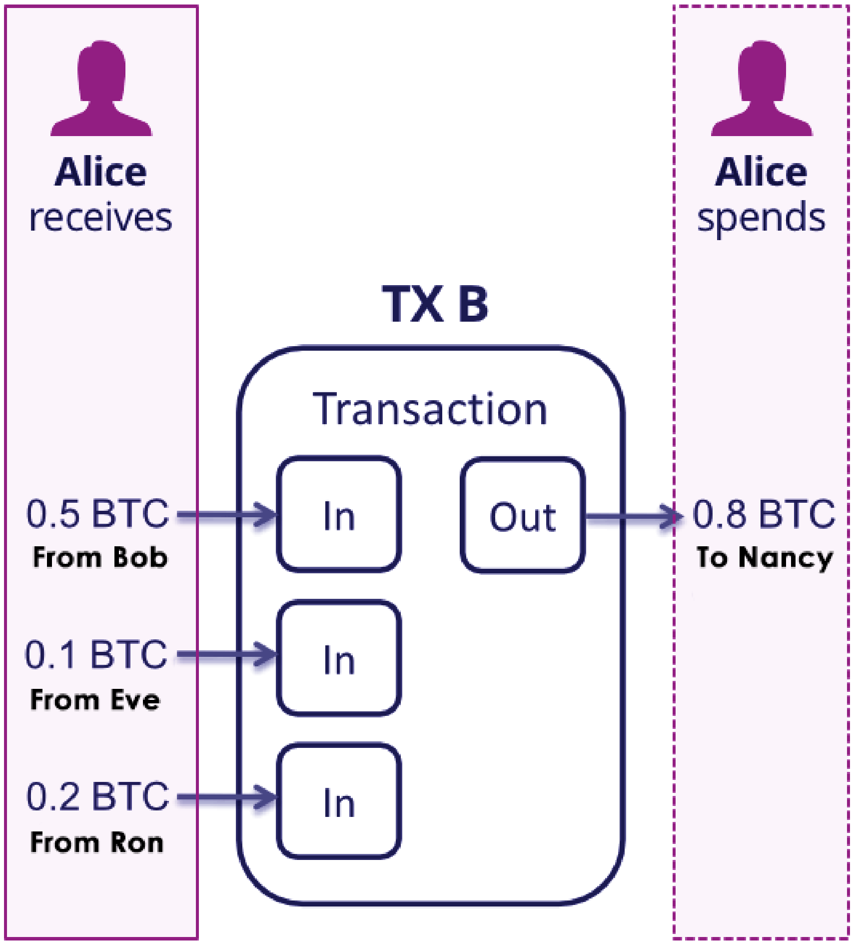
Playstation3 hack:

https://arstechnica.com/gaming/2010/12/ps3-hacked-through-poor-implementation-of-cryptography/

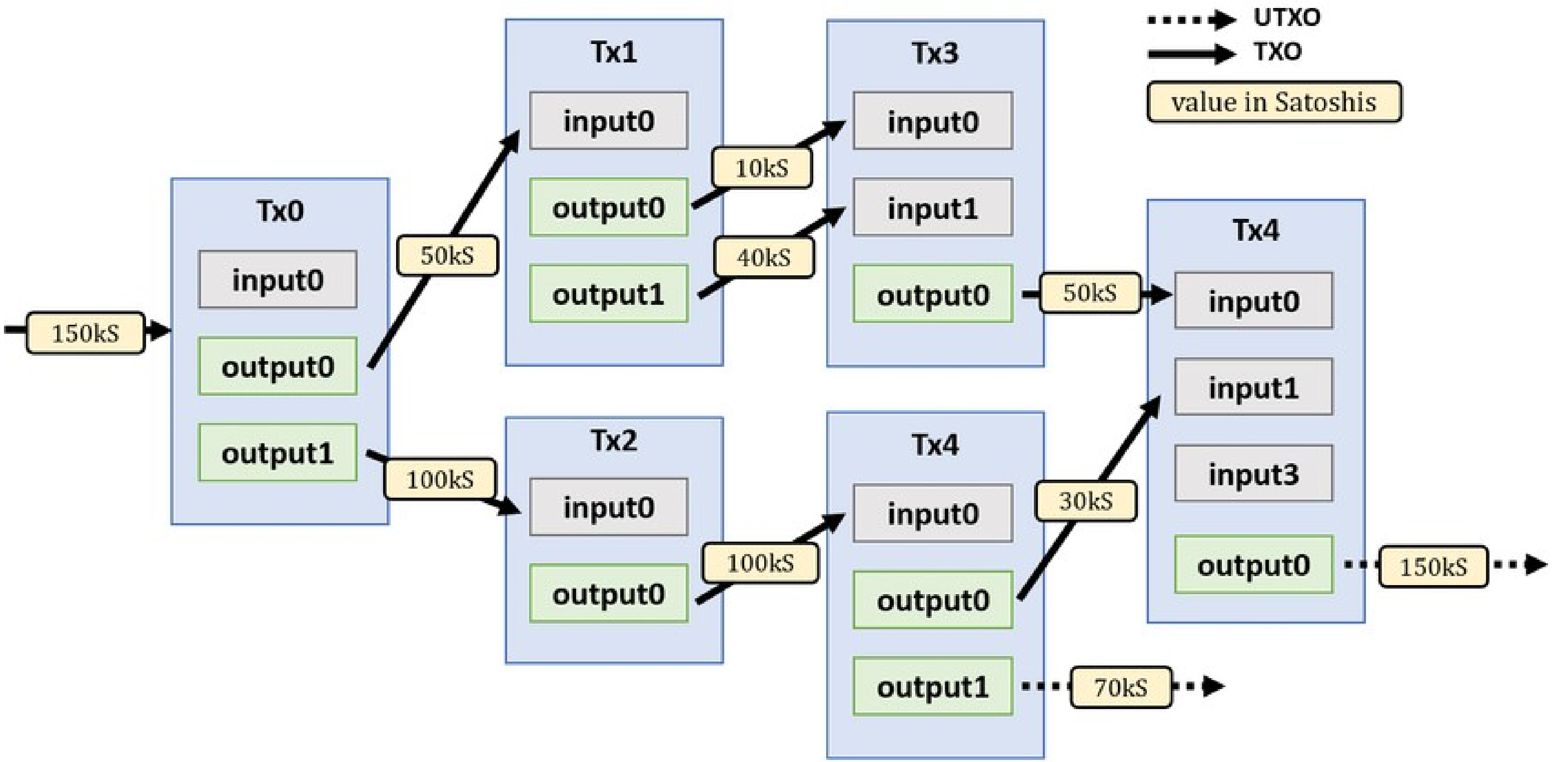


In each TX, the owner of a public key sign with his/her private key specifying the Owner's public key

- A TX can spend several inputs
- In this example, Bob, Eve and Ron in some past txs, used the public key of Alice as destination
- Alice can aggregate different previous outputs, even from different Txs and use them a inputs



Also, multiple outpus (public keys) can be used as destination: in Tx0, some entity sign a 150k sat to create two outputs, that will be spent in future blocks Tx1 and Tx2



What Does it Means “Owning”?

There is nothing as: *“having bitcoins in some place”*

There is no account, no storage point, no registration

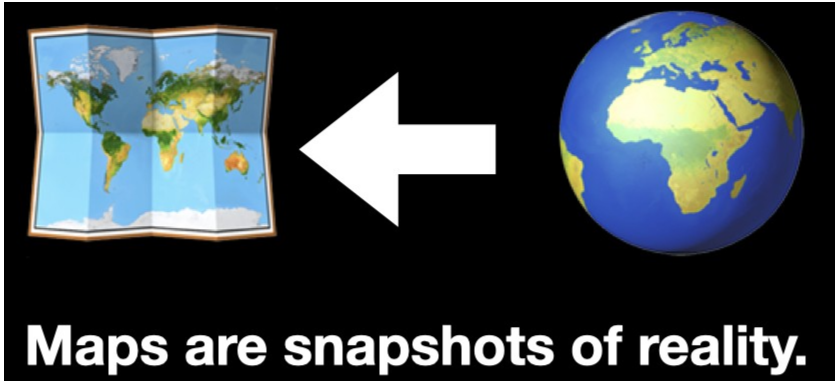
- “owning” is an abstraction that means only 2 things:

1) Some other entities, in some previous transactions, used their own private keys to using your **public key P** as destination output

2) Since only you know the private key **“n”** corresponding to **P**, **only you will be able to demostrate** that **n*G** = **P**, unlocking them in some future transactions

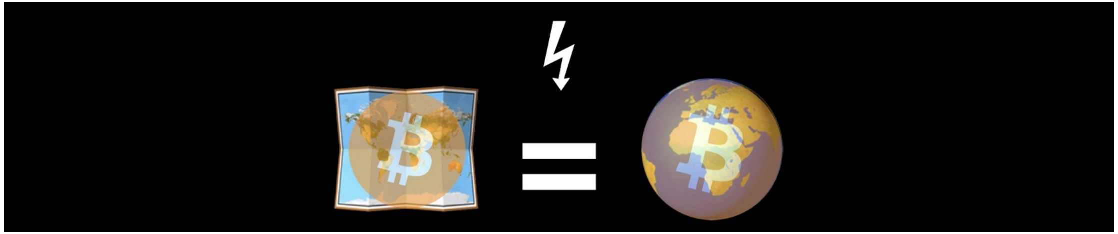
Information vs Physical Transfers

Information: The Map, NOT the Territory



Direct Physical Transfers: no problem, apple atoms are both reality and representation
Physical Transfers with Ledger(Maps): we need some trusted entityt the updates information on Maps → some problems
Pure Information: no external physical reality, LOT of problems!!

In Bitcoin the Map IS the Territory



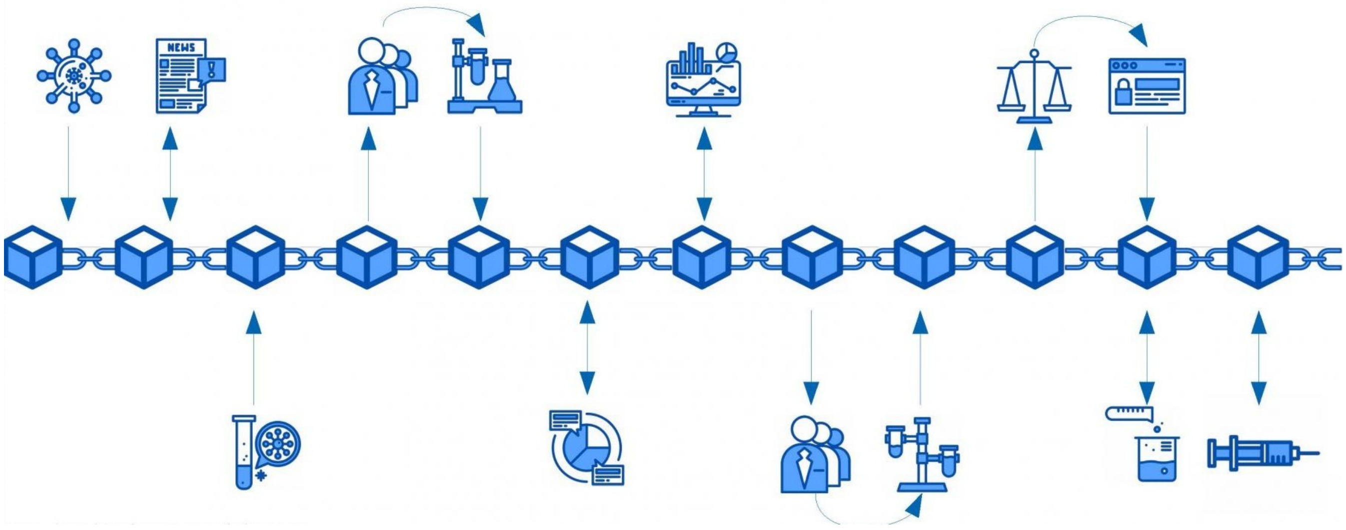
- A transaction **NOT ONLY** “describes” a transfer from A to B it’s **ALSO** the “reality” of the transfer
- **It is the immutable proof that the capability of solving some the math problem has been moved from entity A to B.**

Impossible Mission?

1)**We must guarantee the order of events**

2)**Ensure that sender and receiver are the correct ones**

→ Entities not trusting each other agree on some "digital reality"



FUD Moment...

“...banning the blockchain!”

“banning the possession of... a number?

FUD Moment…

“You can follow the history of each public key address…”

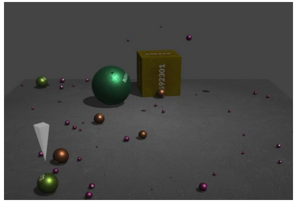
…Wait! wasn’t it the perfect tool for hidden Illegal activity?!?

Fancy Visualization Tools

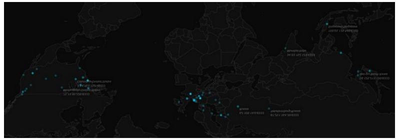
https://blocks.wizb.it/#

https://mempool.space/

https://privacypros.io/tools/bitbonkers/



https://bitnodes.io/nodes/live-map/



http://www.bitlisten.com/

Self-custody: Brain Wallets

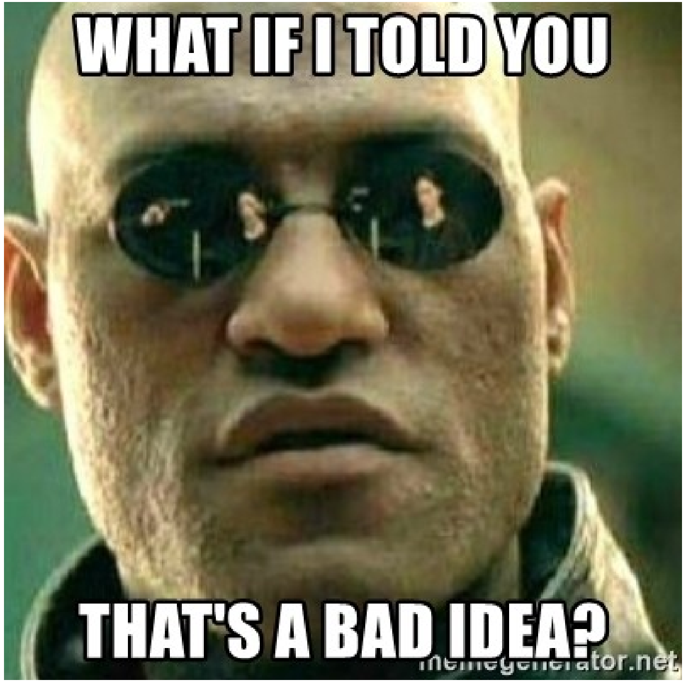
- **"I hate writing 256 bits!"** why don't choose some secret password that **I will always remember**, calculate the SHA256 hash, and then use the resulting 256 bits number as private key n !
- The corresponding public key will be: P=n*G
- Everybody will use P to send me transactions, but nobody will be able to know "n" to unlock them!

Brain Wallets

- **Human brain is not good a good source of entropy :)**
  - An attacker specifically focused on you could try even millions of words combinations related to you, e.g.:

“I love <wife/husband/cat>”

- Also, quotes like beginning of books, songs, etc are very easily checked in any moment



- **Experiment:** private key chosen from hashing parts of famous books/songs
- Four of the sweeps occurred after 22 blocks
- The first one took a few seconds
- All the funds were swept away within a day

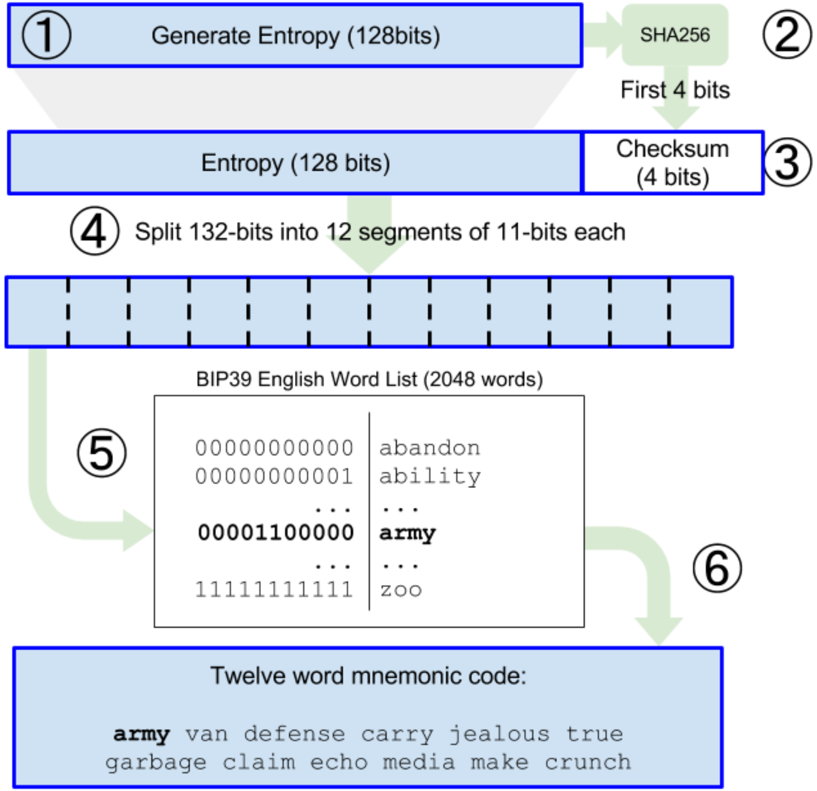
https://blog.bitmex.com/call-me-ishmael/

Passphrase	Source	Sha256
Call me Ishmael	"Moby-Dick" by Herman Melville	a88910233e176ef4489b52d686f326d7ff9ccff686065a44cbd3665384508ad6
It is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want of a wife	"Pride and Prejudice" by Jane Austen	be09c4df6444afa6adff8098c0cf273c3e9fef04a1a8e20de8218eca0bec383d
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair	"A Tale of Two Cities" by Charles Dickens	e051a4337000945d99a46ac1b56244106f732535c11c22c229c6d620ab47199f
In the beginning God created the heaven and the earth	The King James version of the Bible	f153b22c61d6013bf2d7aa5a3fe75327187636131e9022dab1304333751a7301
The answer, my friend, is blowin' in the wind	"Blowin' in the Wind" by Bob Dylan	aeac73098d2b9a29ba47c4893c2d0b6fbbba487d21b1b3a57a55ee11c7a6a476

- The speed of the redemption of the funds clearly indicates that people have **servers up online 24/7 scanning the blockchain**
- These servers are likely to have **pre-generated many hundreds of thousands of Bitcoin addresses**, using text from thousands of published works, music, books, academic papers, magazines, blogs, tweets and other media and then stored these in a database.

Solution: BIP39 Standard

- **Opposite approach of Brainwallet:** generate a random **private key n**, an convert the bits into a set of words
- Segments of 11 bits can be used as index of a 2048 words vocabulary
- BIP39 Standard: These words have been specifically chosen so that cannot be confused, even if handwritten



DEMO: Try BIP39

https://learnmeabitcoin.com/technical/mnemonic

“Well...Really Good tip, I will use BIP39... Thank You!”

“...But I had another great idea:

I will split my 12 secret words into two sets of 6 words each, putting them in two different locations...

Then, if one location is discovered there are still 6 words missing!”


THAT IS...
A GREAT IDEA!!

….ehm…. Actually NOT!

Key Security is not Linear

- 12 word mnemonic offers 128 bits of entropy
- Each bit has an exponential effect on entropy
- Cutting 12 words in half is cutting your entropy down to 64 bits each, which is not as secure anymore.

- This is the difference:

2^64 = 18446744073709551616
2^128 =340282366920938463463374607431768211456

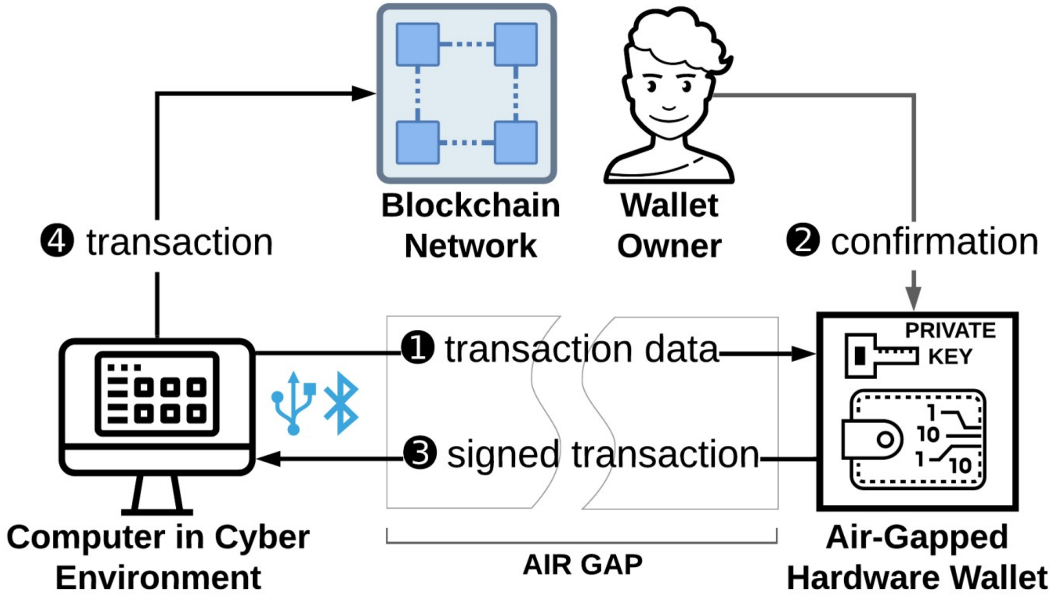
E.g., assuming 1ns per test, it's like 584 years vs 10^20 centuries

Common Sense Suggestions

- BIP39 seed words are not a password to be entered any moment, but **just a backup** of the private key n, don't need to access to them every time!
- Just create 2 or 3 copies of the seed word list on paper and put in different locations
- **NEVER type the words in web sites**
- **Whenever you are asked to enter the seed words: probably a fishing attack**

Cold Storage: Hardware Wallets

- BIP39 Seed words only entered into device to initialize the **private key n**
- The private key stays in a physically separate offline hardware, encrypted in a Secure Element chip



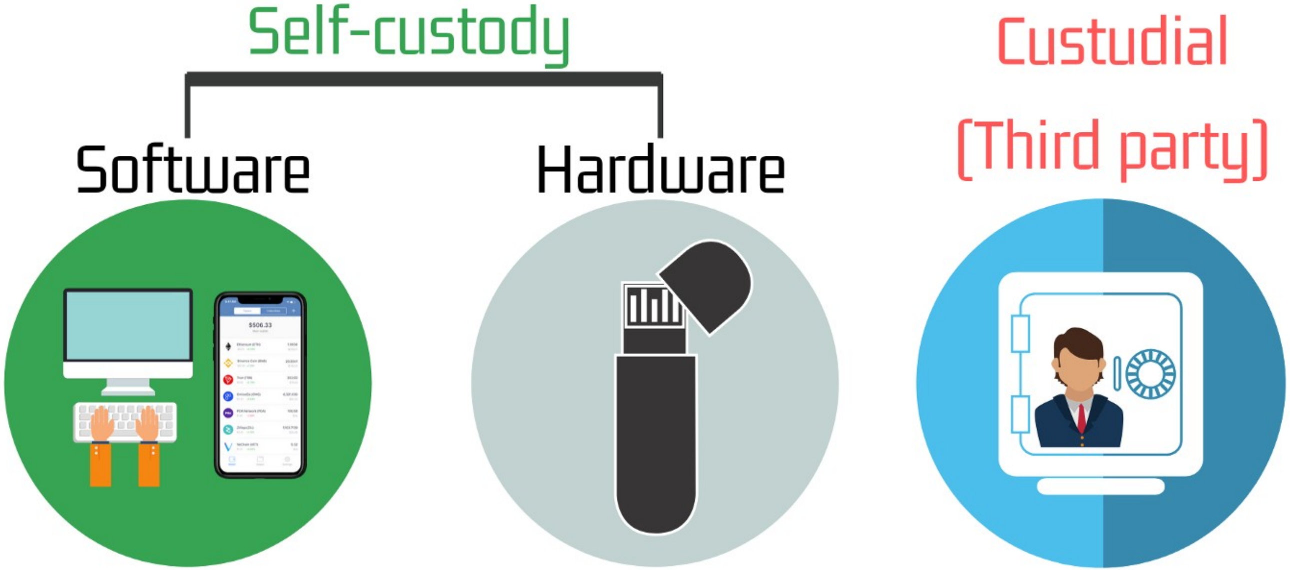
Small cheap hardware devices with simple and verified hardware

Opensource+Openhardware:

- https://github.com/coldcard/firmware

Custodial vs Self-Custodial

- Who owns the private key?
- Custodial: the private key is managed by a third party



User Security Trade-offs

	Self-Custody	Third-Party Custody
Hot	BRD, EXODUS *Inexpensive, moderately secure* Atomic Wallet	BINANCE, coinbase *Convenient, easy, most at-risk* OKEX
Cold	TREZOR *Hardware-based, more secure, most responsibility* Ledger	BitGo *Convenient, most secure, pricey, designed for institutions* coinbase Custody

What are Wallets?

- An user could have multiple private/public keys in possession
- Thus, from the perspective of the user, the "balance" is the sum of all his/her unspent outputs
- A Wallet is a software that collects all private/public keys for a given user
- A Wallet is only meant to improve the user-experience: blockchain knows nothing about "wallets", there only transactions

DEMO: Using a Wallet on Testnet

Electrum wallet (or another): https://electrum.org/#download

Run normally at least once, then try the testnet:

- On MacOS terminal:
  - `open –n /Applications/Electrum.app --args –testnet`
- On Windows terminal:
  - `electrum-VERSION_HERE –testnet`

  – or equivalent, depending on your executable name
- On Ubuntu/Linux (from Applications directory in home)
  - `./electrum-VERSION_HERE --testnet`

DEMO: Testnet Faucets

- You can get some tBTC to play with from a testnet bitcoin faucet, which gives out free tBTC on demand.

  Here are a few testnet faucets:

- https://coinfaucet.eu/en/btc-testnet
- https://testnet-faucet.mempool.co
- https://bitcoinfaucet.uo1.net
- https://testnet.help/en/btcfaucet/testnet