Blockchain on Enki

## **Metadata**

**Name:** Blockchain

**Description:** The underlying technology of Bitcoin, the blockchain will be the motor of future distributed networks.

**Courses:** Fundamentals

## **Sequencing**

**Fundamentals:**

* The Blockchain
* The Bitcoin Blockchain
* Building the Ledger
* Mining
* Wallets
* Bitcoin Anonymity
* Altcoins
* Consensus Algorithms
* Forking
* Cryptoeconomics
* Ethereum I
* Solidity
* Ethereum II
* Ethereum on the Web

## **Lessons**

The following lessons will be served in the order of their appearance

### The Blockchain

**Description:** Discover what the blockchain technology and how it can be used.

**Insight 1:** What Is a Blockchain?

**First Question:**

Identify the first application of a blockchain:

???

\* peer-to-peer payment system

\* domain name registration

\* distributed database of public records

\* proof-of-concept for cryptographic mechanisms

**Explanation:**

A \*\*blockchain\*\* (a literal chain of blocks) is a peer-to-peer network which is looking to build and maintain a sequence of information bundles. For the information to be added to the sequence, it has to be verified first by trust-worthy nodes.

![blockchain](%3Csvg%20width%3D%22100%25%22%20height%3D%22auto%22%20viewBox%3D%220%200%20320%20221%22%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%3E%3Ctitle%3EGroup%207%3C%2Ftitle%3E%3Cg%20fill%3D%22none%22%20fill-rule%3D%22evenodd%22%3E%3Crect%20fill%3D%22%23FFF%22%20fill-rule%3D%22nonzero%22%20width%3D%22320%22%20height%3D%22221%22%20rx%3D%229%22%2F%3E%3Cg%20fill%3D%22%23414143%22%3E%3Cpath%20d%3D%22M69%2031l17%209.59V59l-17-9.59zM87.06%2020L104%2029.527%2087.06%2039%2070%2029.527zM88%2059l17-9.59V31l-17%209.59z%22%2F%3E%3C%2Fg%3E%3Cpath%20d%3D%22M112.5%2040.25h22M160%2083.5V65M160%20154v-18.5%22%20stroke%3D%22%23414143%22%20stroke-width%3D%223%22%20stroke-linecap%3D%22square%22%2F%3E%3Cg%20fill%3D%22%23414143%22%3E%3Cpath%20d%3D%22M142%20101l17%209.59V129l-17-9.59zM160.06%2090L177%2099.527%20160.06%20109%20143%2099.527zM161%20129l17-9.59V101l-17%209.59z%22%2F%3E%3C%2Fg%3E%3Cg%20fill%3D%22%23414143%22%3E%3Cpath%20d%3D%22M142%20170l17%209.59V198l-17-9.59zM160.06%20159l16.94%209.527L160.06%20178%20143%20168.527zM161%20198l17-9.59V170l-17%209.59z%22%2F%3E%3C%2Fg%3E%3Cpath%20d%3D%22M186.5%20110.25h22%22%20stroke%3D%22%23414143%22%20stroke-width%3D%223%22%20stroke-linecap%3D%22square%22%2F%3E%3Cg%20fill%3D%22%23414143%22%3E%3Cpath%20d%3D%22M215%20101l17%209.59V129l-17-9.59zM233.06%2090L250%2099.527%20233.06%20109%20216%2099.527zM234%20129l17-9.59V101l-17%209.59z%22%2F%3E%3C%2Fg%3E%3Cg%20fill%3D%22%23414143%22%3E%3Cpath%20d%3D%22M142%2031l17%209.59V59l-17-9.59zM160.06%2020L177%2029.527%20160.06%2039%20143%2029.527zM161%2059l17-9.59V31l-17%209.59z%22%2F%3E%3C%2Fg%3E%3C%2Fg%3E%3C%2Fsvg%3E)

The main goal of blockchains is to give back to individuals the control over their own data, in contrast to having huge, private companies taking advantage of that. The power of the network comes from its \*\*decentralization\*\*, \*\*high availability\*\* and its \*\*tamper resistance\*\*, given enough nodes are part of the network. Every blockchain is using cryptography concepts, such as asymmetric cryptography and hash functions, to establish, support and maintain properties like decentralization, data integrity and anonymity.

Theoretical blockchains go back to the early 90's, when multiple different projects were aiming to creat fully digitized, decentralized and securitized payment systems[1]. Most of them never had the chance to see the light of day, and those who had were unsuccessful.

However, in January 2009, the first public blockchain was released - the Bitcoin blockchain. It was created by a person/group of people that signed as Satoshi Nakamoto. We will cover Bitcoin blockchain in a future workout so we won't dive into details right now.

What is important to point out though is that, although limited in functionality, Bitcoin worked as proof-of-concept for more complex blockchains. The peer-to-peer payment system pionereed by Bitcoin allowed forward thinkers to extend the concept to incorporate applications like crowdfunding, domain registration, prediction markets, voting, etc.

Having an individual blockchain for each every project isn't wanted, nor scalable. The most complex project so far, Ethereum, is a blockchain that works as a framework for developing and sharing any kind of application by writing code in its Turing-complete programming language, Solidity.

Blockchains had to start by doing one thing (and doing it well). Consider the following list of concepts one has to familiarize with to develop and release a blockchain:

\* Data structures: linked lists, hash maps and graphs (merkle trees)

\* Cryptography: asymmetric (public-key) cryptography, hash functions

\* Distributed systems: consensus, consistency and sharding

\* Networking: peer-to-peer protocols, the packet model, routing

\* Economy: game theory, macroeconomics

Another proof of the worthiness and capabilities of this technology comes from Estonia: the small european country started managing its registers, such as national health, judicial, security and commercial code systems on their own blockchain. The system has been live since 2012 and their initiative is only moving forward.

**Second question:**

A blockchain is a ??? network.

\* distributed

\* centralized

\* free

**Insight 2:** Blockchain Components

**First Question:**

Identify the item that `is not` part of a blockchain:

???

\* Central database

\* Set of transactions

\* Miner

\* Cryptography

\* Reward

**Explanation:**

A blockchain is a type of network. Just like any other network, it has to follow a \*\*protocol\*\*. \*\*Nodes\*\* are required to acknowledge, verify and append information. This time, however, information is not converging towards a single source of truth (e.g. central database), but is brodcasted between nodes. This property is called \*\*decentralization\*\*.

After a certain period of time (dependant on internet speed and latency) and a number of comfirmation from other verifying nodes, the set of data (\*\*block\*\*) is confirmed as part of the blockchain.

A distinction can also be made between nodes: there are \*\*full nodes\*\* and \*\*mining nodes\*\*. In short, mining nodes, or simply \*miners\*, are the ones that verify and bundle the valid, available[1] transactions into a new block. They also make sure to include a reference to the previous block in the chain. Full nodes are the ones that receive the bundled blocks from miners and verify its \*integrity\*. If everything is in order, they then append the block to their copy of the blockchain and send the block and a confirmation message forward.

To prevent nodes for turning against the network, they \*are rewarded\* whenever they find a valid block - but more on this later.

The high-level components of the blockchain can be enumerated as follows:

\* `Protocol`: The set of rules on which the network runs

\* `Full Node`: A node that validates transactions and blocks

\* `Miner`: A node that gathers transactions and bundles them up in a block

\* `Block`: A set of transactions available to the miner at some point in time

\* `User`: The human component behind other participants in the network

**Second question:**

What is the name of blockchain participants that verify whole blocks validity?

???

\* Full nodes

\* Miners

\* Traders

\* Verifiers

**Insight 3:** Blocks

**First Question:**

What function is used to check if the content of any block has been altered?

???

Identify a property of the above function:

???

\* Hash function

\* One-way function

\* Public-key function

\* Reductor function

\* Bijective function

\* Reversible function

**Explanation:**

\*\*Blocks\*\* are the structural component of the blockchain. They exist, no matter the problem the blockchain solves. The piece of information that goes in a block is usually called \*transaction\*.

In a payment system, a transaction will contain a sender, a recipient and a transaction value. In a domain name registration service, a buyer, the domain bought and the value of the domain will most likely be recorded. The data in a block that stores decentralized applications will most likely be the code. In other words, any application will have its own transaction type.

We've got data that goes in the block, now we need the link between them. We also need a way to ensure that once a block is in a blockchain it cannot be modified - such that a reference to the position of the block in the blockchain would not suffice. This is where \*\*cryptographic hash functions\*\* come into play.

### Hash functions

Informally, a \*\*hash function H\*\* is a function that compresses an input of any size to a string of fixed length. The length of the output depends on the function specifications, but the most common you will see are either `256`, `384`, or `512` bits long.

They are one-way functions: the compression algorithm is fairly simple and always yields the same result, but the chance of finding the input based on the hash string is close to nil. The current standard is `FIPS-202` (a.k.a. Secure Hash Algorithm 3 or simply SHA-3).

Consider the following examples:

```bash

string: Enki is great!

SHA3-256: 8b9580615ff7c00180ceae471033b8ced0

092ea79540db552e6f559af40d53fa

string: Enki is great

SHA3-256: 5e8f454b534ec3eccf0a431d960b67a07a

cdf21d722a4d9fd1f4f3b35fe67507

string: Blockchain is the future

SHA3-256: 9dc3d2f91fc8a9417a8dfd4c510f4690db

d3424012ebdedb98d46c55395194e8

```

As you can clearly see, removing the `!` from the first string yields a completly different hash - this ensures that even the smallest change made to a block will be reflected in its hash.

The input doesn't necessarily have to be a string since the function is modyfing the underlying representation, which is always in bits.

### Hashing block data

If any piece of data can be represented as an unique string of fixed length, what if, to build block N, we we are to hash the content of block N-1 and block N-1's reference to block N-2? By including the content, any modification done to it would change the hash. By also including the reference to block N-2, we ensure that every block before cannot be tampered with.

We would then be able to give a somewhat formal representation of what is hashed:

```bash

// '|' = concatenation of strings

ref(n) = H(header(n) | content(n)

| ref(n-1))

```

This reference is also called the \*\*address\*\* of the block. Depending on the blockchain, other variables might find their way into the hash function in order for the network to validate it. We'll talk about how exactly it happens using the Bitcoin and Ethereum blockchains as examples in future workouts.

But for now, consider a network that accepts any block that contains the previous block reference:

[Image here with what goes in a block]

It's worth mentioning about the \*\*genesis block\*\* or the first block of the blockchain: as there's no previous, the reference will be equal to `0`.

In the "learn more" section you can find a link to an interactive website that allows you to create and append blocks to a simplified version of the Bitcoin blockchain.

**Second question:**

Identify the input to the hash function that yields the address of a new, generic block:

???

\* Previous block's reference and current's content and header

\* Genesis block's reference and previous' reference

\* Account transactions and the reference to the previous block

### The Bitcoin Blockchain

**Description:** The first public blockchain - Bitcoin.

**Insight 1:** Bitcoin

**First Question:**

Identify the two cryptographic mechanisms that combined produce a `digital signature`:

???

\* Public key and private key

\* Public key and UTXO

\* Private key and full name

\* A photo of your signature and an encryption algorithm

**Explanation:**

\*\*Bitcoin\*\* was the first public blockchain which works as a ledger for tracking bitcoin tokens[1] transactions happening between users. A token is a "e-currency based on cryptographic proof"[2]. The first block was mined on the 3rd of January 2009.

It was all the idea of one (or a team), Satoshi Nakamoto. But no one knows who Satoshi Nakamoto really is. There had been speculations over the years, but none of them were confirmed.

Some say this worked in favor of Bitcoin: by using a pseudonym, Satoshi Nakamoto made sure the spotlight was pointed to the technology rather than to the person behind it. Slowly, but surely, bitcoin price raised year by year, attracting more and more enthusiasts.

But the skyrocketing price is not the reason Bitcoin is great. Yes, it is fairly limited in functionality and slow, but without it there wouldn't have been any blockchains. Like any pioneer, Bitcoin survived through numerous scandals, regulatory commissions and heists in a long way to prove it is worth adopting.

A Bitcoin block contains a list of \*\*transactions\*\* between network participants. You don't necessarily have to be an active participant, but you do need a Bitcoin address. A \*\*Bitcoin address\*\* is the public half of the \*\*digital signature\*\* used to sign outgoing transactions. As long as you have a \*\*public key\*\*, anyone can send you bitcoins. If you also have the \*\*private key\*\*, you can send bitcoins to others.

A Bitcoin transaction looks something like this:

```bash

Satoshi Nakamoto sends 10 bitcoins

to Hal Finney

```

What might seem strange is that no absolute balance is kept on the blockchain. This model is called \*\*UTXO\*\*, or \*Unspent Transaction Output\*.

**Second question:**

What is the transaction model used by the Bitcoin blockchain?

???

\* UTXO model

\* Account model

\* A combination of UTXO and account models

\* A simple transaction

**Insight 2:** The Bitcoin Blockchain

**First Question:**

What is the maximum number of transactions per second the Bitcoin network can process?

???

\* 7

\* 3

\* 15

\* 5

**Explanation:**

In the previous insight, we talked about people sending bitcoins around. But where do bitcoins come from?

Unlike real world currencies, who are issued by a country's central bank to meet a financial goal, bitcoins come in a limited supply of 21 millions. They are \*\*created by the newtork\*\* to \*\*reward\*\* the miner who appended a valid block to the blockchain.

In a sense, they are created from thin air. But the steady rate at which they are created and the feature of having a limited supply make them a desirable asset. The \*\*block reward\*\* in 2009 was 50 bitcoins and it has been halving ever since every 210,000 blocks.

A new block is discovered roughly every 10 minutes, so the block reward remains constant for about 4 years. The reward will be 12.5 bitcoins until late June 2020.

So, for the first users to be able to transact bitcoins, they had to mine them first. In the early days, it was still possible to do it on your PC using the CPU. Nowadays, unless you have entire rooms filled with dedicated hardware (ASICs or application-specific integrated circuits), there's almost no chance of finding a block by yourself.

But the extra dedicated computer power isn't helping with the network throughput[1], which is somewhere between 3 and 7 transactions processed per second. It is, however, helping with the security of the network: more people verifying transactions means a larger price to attack the network.

After enough bitcoins had been mined, there needed to be a way to exchange them for fiat currencies ($/€/¥). As such, cryptocurrency exchanges were created specifically for that. People could buy and sell bitcoins to one another and even trade them for profit.

Bitcoin became a pseudo-anonymous way to donate to Wikipedia, Wikileaks and others. Vendors also started to accept bitcoin as a payment method, so you could spend your bitcoins or pizza or magazine subscriptions. But the highly volatile price and the network's throughput limitation meant that the actual value of the arriving bitcoins might not reflect the real value of the items.

**Second question:**

Which of the following is not a way in which you can obtain bitcoins?

???

\* Listening to the Bitcoin blockchain

\* Selling products on the Internet

\* Exchanging other currencies for bitcoins

\* Buying bitcoins from ATMs

\* Mining

**Insight 3:** The Bitcoin Block

**First Question:**

What is the consensus algorithm implemented in the Bitcoin blockchain?

???

\* Proof of work

\* Proof of stake

\* Proof of burn

\* Proof of authority

**Explanation:**

The process of validating a Bitcoin block is called \*\*proof of work\*\* (PoW). We'll talk more about PoW and other \*consensus algorithms\* in a future workout, but, for now, you should know that it describes a process in which miners try to solve a cryptographic puzzle.

The content of a Bitcoin block is a list of transactions available to the miner at a certain point in time.

If you remember, the simplest way to compute a block's address is to hash the previous block reference, and new block's content and header. Bitcoins makes everything harder by introducing in the equation the miner's identity and the concept of a \*\*nonce\*\*, or a random number that has to be added to the hash input such that the hash meets a certain criteria. All the puzzle is is finding the right nonce that makes the hash meets the network requirement.

Any byte changed in the input changes the output drastically: by incrementing the nonce, a new hash is obtained. The block is \*valid\* if the resulting hash starts with more than predetermined number of \*0\*'s. The hash is always 64 characters (64 bytes) long and the number of \*0\*'s is the difficulty. The difficulty is set by the network and it depends on the processing power (this is also called hashing power) the network has.

The more 0's the hash has to start with, the more difficult the computation is. The first block mined had \*difficulty 1\* and it needed to begin with at least 8 \*0\*'s. The last block discovered at the time of writing this had a difficulty of \*3 trillion\*, or roughly 18 \*0\*'s. This difficulty is equivalent to a hash rate of 22,500 PH/s (PetaHashes/s), or 22 million trillion hashes per second.

In the learn more section, you should find a link to an online block explorer where you can play around.

**Second question:**

In the context of Bitcoin mining, what is the component a miner is searching for?

???

\* The nonce

\* The transactions

\* The previous block's hash

\* The hash

### Building the Ledger

**Description:** How is the Bitcoin blockchain built?

**Insight 1:** The Ledger

**First Question:**

Identify a valid blockchain operation from the list below:

???

\* Append data

\* Modify data

\* Remove data

\* Update data

**Explanation:**

The ledger, or better said "the transaction-based ledger", is a blockchain's \*log of transactions\*. As with any other ledger, the transacted object must be \*\*uniquely identifiable\*\*, under one form or another. Bitcoin, like most of the other blockchains, are gravitating around their only asset - the cryptocoin. Hence, the Bitcoin ledger will record bitcoin transactions.

Other types of ledgers include an account-based ledger, used by the Ethereum blockchain, or a combination of the two, used by the QTUM blockchain.

### Why are most blockchains built around coins?

Digital coins are easier to track. We cannot necessarily link one coin to an actual person, but we can track their movement. Digital coins (similar to any other currency) are commonly accepted as a mean to evaluate services, items, shares, etc. and it is in developers interest to also release a token.

Before money, it was something like this:

![barter cricle](%3Csvg%20width%3D%22100%25%22%20height%3D%22auto%22%20viewBox%3D%220%200%20320%20248%22%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%3E%3Ctitle%3EGroup%204%3C%2Ftitle%3E%3Cg%20fill%3D%22none%22%20fill-rule%3D%22evenodd%22%3E%3Crect%20fill%3D%22%23FFF%22%20fill-rule%3D%22nonzero%22%20width%3D%22320%22%20height%3D%22248%22%20rx%3D%229%22%2F%3E%3Cg%20transform%3D%22translate%2828%2069%29%22%3E%3Ccircle%20stroke%3D%22%230058DE%22%20stroke-width%3D%225%22%20cx%3D%2238.5%22%20cy%3D%2238.5%22%20r%3D%2238.5%22%2F%3E%3Ctext%20font-family%3D%22Roboto-Bold%2C%20Roboto%22%20font-size%3D%2226%22%20font-weight%3D%22bold%22%20fill%3D%22%23000%22%3E%3Ctspan%20x%3D%2231%22%20y%3D%2229%22%3EX%3C%2Ftspan%3E%3C%2Ftext%3E%3Cpath%20d%3D%22M19.599%2032.25h37.14%22%20stroke%3D%22%238CBAFF%22%20stroke-width%3D%22.5%22%20stroke-linecap%3D%22square%22%2F%3E%3Ctext%20font-family%3D%22Roboto-Light%2C%20Roboto%22%20font-size%3D%2214%22%20font-weight%3D%22300%22%3E%3Ctspan%20x%3D%2219.062%22%20y%3D%2248%22%20fill%3D%22%23000%22%3EHas%3C%2Ftspan%3E%20%3Ctspan%20x%3D%2243.521%22%20y%3D%2248%22%20font-family%3D%22Roboto-Regular%2C%20Roboto%22%20font-weight%3D%22normal%22%20fill%3D%22%23000%22%3E%20%3C%2Ftspan%3E%20%3Ctspan%20x%3D%2247.048%22%20y%3D%2248%22%20font-family%3D%22Roboto-Medium%2C%20Roboto%22%20font-weight%3D%22400%22%20fill%3D%22%23649AEB%22%3EA%3C%2Ftspan%3E%20%20%3Ctspan%20x%3D%2212%22%20y%3D%2261%22%20fill%3D%22%23000%22%3EWants%3C%2Ftspan%3E%20%3Ctspan%20x%3D%2251.067%22%20y%3D%2261%22%20font-family%3D%22Roboto-Regular%2C%20Roboto%22%20font-weight%3D%22normal%22%20fill%3D%22%23000%22%3E%20%3C%2Ftspan%3E%20%3Ctspan%20x%3D%2254.595%22%20y%3D%2261%22%20font-family%3D%22Roboto-Medium%2C%20Roboto%22%20font-weight%3D%22400%22%20fill%3D%22%236198EE%22%3EB%3C%2Ftspan%3E%3C%2Ftext%3E%3C%2Fg%3E%3Cg%20transform%3D%22translate%28219%2021%29%22%3E%3Ccircle%20stroke%3D%22%230058DE%22%20stroke-width%3D%225%22%20cx%3D%2238.5%22%20cy%3D%2238.5%22%20r%3D%2238.5%22%2F%3E%3Ctext%20font-family%3D%22Roboto-Bold%2C%20Roboto%22%20font-size%3D%2226%22%20font-weight%3D%22bold%22%20fill%3D%22%23000%22%3E%3Ctspan%20x%3D%2231%22%20y%3D%2229%22%3EY%3C%2Ftspan%3E%3C%2Ftext%3E%3Cpath%20d%3D%22M19.599%2032.25h37.14%22%20stroke%3D%22%238CBAFF%22%20stroke-width%3D%22.5%22%20stroke-linecap%3D%22square%22%2F%3E%3Ctext%20font-family%3D%22Roboto-Light%2C%20Roboto%22%20font-size%3D%2214%22%20font-weight%3D%22300%22%3E%3Ctspan%20x%3D%2219.458%22%20y%3D%2248%22%20fill%3D%22%23000%22%3EHas%3C%2Ftspan%3E%20%3Ctspan%20x%3D%2243.917%22%20y%3D%2248%22%20font-family%3D%22Roboto-Regular%2C%20Roboto%22%20font-weight%3D%22normal%22%20fill%3D%22%23000%22%3E%20%3C%2Ftspan%3E%20%3Ctspan%20x%3D%2247.444%22%20y%3D%2248%22%20font-family%3D%22Roboto-Medium%2C%20Roboto%22%20font-weight%3D%22400%22%20fill%3D%22%23649AEB%22%3EB%3C%2Ftspan%3E%20%20%3Ctspan%20x%3D%2212%22%20y%3D%2261%22%20fill%3D%22%23000%22%3EWants%3C%2Ftspan%3E%20%3Ctspan%20x%3D%2251.067%22%20y%3D%2261%22%20font-family%3D%22Roboto-Regular%2C%20Roboto%22%20font-weight%3D%22normal%22%20fill%3D%22%23000%22%3E%20%3C%2Ftspan%3E%20%3Ctspan%20x%3D%2254.595%22%20y%3D%2261%22%20font-family%3D%22Roboto-Medium%2C%20Roboto%22%20font-weight%3D%22400%22%20fill%3D%22%236198EE%22%3EC%3C%2Ftspan%3E%3C%2Ftext%3E%3C%2Fg%3E%3Cg%20transform%3D%22translate%28142%20146%29%22%3E%3Ccircle%20stroke%3D%22%230058DE%22%20stroke-width%3D%225%22%20cx%3D%2238.5%22%20cy%3D%2238.5%22%20r%3D%2238.5%22%2F%3E%3Ctext%20font-family%3D%22Roboto-Bold%2C%20Roboto%22%20font-size%3D%2226%22%20font-weight%3D%22bold%22%20fill%3D%22%23000%22%3E%3Ctspan%20x%3D%2231%22%20y%3D%2229%22%3EZ%3C%2Ftspan%3E%3C%2Ftext%3E%3Cpath%20d%3D%22M19.599%2032.25h37.14%22%20stroke%3D%22%238CBAFF%22%20stroke-width%3D%22.5%22%20stroke-linecap%3D%22square%22%2F%3E%3Ctext%20font-family%3D%22Roboto-Light%2C%20Roboto%22%20font-size%3D%2214%22%20font-weight%3D%22300%22%3E%3Ctspan%20x%3D%2219.393%22%20y%3D%2248%22%20fill%3D%22%23000%22%3EHas%3C%2Ftspan%3E%20%3Ctspan%20x%3D%2243.852%22%20y%3D%2248%22%20font-family%3D%22Roboto-Regular%2C%20Roboto%22%20font-weight%3D%22normal%22%20fill%3D%22%23000%22%3E%20%3C%2Ftspan%3E%20%3Ctspan%20x%3D%2247.379%22%20y%3D%2248%22%20font-family%3D%22Roboto-Medium%2C%20Roboto%22%20font-weight%3D%22400%22%20fill%3D%22%23649AEB%22%3EC%3C%2Ftspan%3E%20%20%3Ctspan%20x%3D%2212%22%20y%3D%2261%22%20fill%3D%22%23000%22%3EWants%3C%2Ftspan%3E%20%3Ctspan%20x%3D%2251.067%22%20y%3D%2261%22%20font-family%3D%22Roboto-Regular%2C%20Roboto%22%20font-weight%3D%22normal%22%20fill%3D%22%23000%22%3E%20%3C%2Ftspan%3E%20%3Ctspan%20x%3D%2254.595%22%20y%3D%2261%22%20font-family%3D%22Roboto-Medium%2C%20Roboto%22%20font-weight%3D%22400%22%20fill%3D%22%236198EE%22%3EA%3C%2Ftspan%3E%3C%2Ftext%3E%3C%2Fg%3E%3Cpath%20d%3D%22M214.09%2069.338l-12.064-2.392%202.912%2010.608%209.151-8.216zM111.684%2098.175l92.664-25.437.675-.185-.37-1.35-.675.185-92.664%2025.437-.675.185.37%201.35.675-.185zM205.148%20146l10.832-5.825-8.993-6.335-1.839%2012.16zm30.78-44.903l-25.463%2036.14-.403.573%201.144.806.403-.572%2025.463-36.14.403-.573-1.144-.806-.403.572zM103.998%20136.151l5.898%2010.792%206.274-9.035-12.172-1.757zm36.901%2024.774l-28.1-19.515-.576-.399-.798%201.15.575.4%2028.1%2019.514.576.4.798-1.15-.575-.4z%22%20fill%3D%22%230058DE%22%20fill-rule%3D%22nonzero%22%2F%3E%3C%2Fg%3E%3C%2Fsvg%3E)

Since we now have a non-perishable intermediary, there's no need to count of the others to need what we have to offer when we need what they have to offer.

Having a mean to reward well behaved participants in the network, more people are incentivized to join. As more people join the network, the more secure and trustworthy the network becomes.

This way, blockchains are able to self-sustain solely based on the adoption rate.

### The Bitcoin ledger

The ledger is built to be \*\*append-only\*\*. If any other type of update operation is to take place (removal or modification), it usually means that the blockchain's security and integrity are compromised.

Unlike physical ledgers, if a transaction ever took place, there is no way of hiding it. Once happened and confirmed, the details of the transaction will be available to anyone connected to the network. This is one of the reasons the blockchain technology became so popular.

**Second question:**

In the context of coin-based blockchains, the ledger is

???

\* transaction-based

\* account-based

\* property-based

\* storage-based

**Insight 2:** The UTXO Model

**First Question:**

What does UTXO stand for?

???

\* Unspent transaction output

\* Unspent trade output

\* Unspent transaction set

\* Useful transaction set

**Explanation:**

The more scalable way of tracking account balances on the blockchain is the \*\*UTXO\*\* model. To send coins from one account to another, the sender must have an \*unspent transaction output\* recevied previously.

Any transaction requires an UTXO as input and outputs another UTXO. The only transaction that doesn't require an UTXO as input is the block reward transaction: the network creates the bitcoins and outputs a unspent transaction that grants the miner a number of bitcoins.

One important aspect of UTXO model is that every unspent output must be fully consumend in case even the smallest amount of bitcoins is accessed. Consider the following example:

```bash

# Block 1

The network gives Alice 50 bitcoins

# Block 2

Alice gives Bob 10 bitcoins

Alice gives Alice 40 bitcoins

```

Alice now has an UTXO that has a value of 40 bitcoins and Bob has an UTXO of 10 bitcoins. The transaction of sending bitcoin to yourself in order to fully consume an UTXO is called \*\*change address\*\*.

Because UTXOs are used in their entirety, verification process can ignore those that have already been used as input in other transactions. This is why UTXO model is considered to be more scalable than the account model.

**Second question:**

What is a change address transaction?

???

\* A transaction fully consuming an UTXO

\* A transaction send to someone else

\* A transaction between account of the same person

**Insight 3:** Consensus

**First Question:**

What is the type of cryptography used for authentication on a distributed network?

???

What is the outcome of reaching a consensus?

???

\* Public-key cryptography

\* A new block is added

\* RSA

\* The network achieves decentralization

\* New tokens are created

\* Symmertrical cryptography

**Explanation:**

Consider the following scenario: you and your friends want to go out coffee but you are having a hard time deciding where. You all have loyalty cards for different coffee shops (say you receive a free 6th coffee), so everyone is incentivize to voice their opinion.

You could split, buy your desired coffee and meet later - but you won't be able to chit-chat over coffee anymore. Being all equal, you must reach a \*\*consensus\*\*. What if you race 100m and the winner picks the place? What if you promise to pay for their coffee if your friends don't enjoy it? What if one of the option is Starbucks, but you would rather drink a real coffee? How about agreeing to meet every week for a coffee at a different place?

This is also the case for blockchain: there has to be a consensus about what are the conditions a block should meet such that the network accepts it. Consensus algorithms are needed to prevent \*double-spending\*. There isn't a singular \*consensus algorithm\* and they are rather case-specific. We will cover the most popular ones later.

**Second question:**

What problem is solved by waiting for consensus before adding new blocks?

???

\* Double-spend

\* 51% attack

\* Sybil attack

\* Replay attack

**Insight 4:** Verifying the Ledger

**First Question:**

Given the image above, identify the block that is checked when verifying block 4?

???

\* Block 2

\* Block 1

\* Block 3

\* Block 4

**Explanation:**

For the sake of simplicity, we'll tackle the main steps of the verification process of Bitcoin ledger. For more complex operations, such as working with multi-signature accounts, storing records or running code, the verification process will require more variables to be considered.

To append a new block to a transaction-based blockchain, there are verifications conducted on two levels:

\* Transaction level verifications

\* Check if there are enough founds to complete the transaction

\* Check for double-spend

\* Block level verification

\* Check block references

### Transaction verification and the double-spending problem

First and foremost, the blockchain must verify if the coin sender has enough coins to complete the transactions: this is done by going through all previous transactions. If Alice has enough coins, the first part of the verification process is done.

Now, the network has to be sure that Alice doesn't try spend the same coins that were supposed to get to Bob on another transaction to someone else. This is known as the double-spending problem.

In a centralized environment, i.e. bank, there is a single source of truth: the main database. If you would ever check your balance using internet banking, you will most likely see two values - the total balance and the available balance. The available balance is the total balance minus any holds that haven't cleared the account yet.

By keeping track of a second balance, the bank knows for sure you won't be able to spend the same $10 you had used to buy groceries on coffee the next day.

In an decentralized environment, it takes a while for transaction confirmations to move between nodes. Consider this scenario: Alice is living in the UK and buys something for 10 tokens from Bob. The transaction is valid by itself and is brodcasted from the UK:

```bash

Alice pays Bob 10 coins

```

Because it takes a while for the transaction to be added to a block and then to reach the nodes in Australia, Alice could also brodcast another transaction from a server in Australia that said:

```bash

Alice pays Alice1 10 coins

```

By the nature of blockchain, if Alice had only 10 coins and both transactions were to be processed at the same time, one would go through and the other would be dropped. If Bob was to confirm the payment before the network actually did it, the transaction dropped could be the one sending him the coins.

This is usually dealt with by waiting for a number of new blocks (5-6) to be appended after the block containing the said transaction to "confirm" it. The longer the new chain, the more trust-worthy it is consider to be.

### Block verification

This verification ensures that everybody has exactly the same history: the hash of the previous block must be correct and match the end of the currently longest chain. Ideally, the lookup should be done up to the genesis block. However, this isn't scalable.

In most cases, the network designates a checkpoint block, that is used as a definite source of truth.

**Second question:**

What is the problem solved by verifying all previous transactions?

???

\* Double-spending

\* Identity tracing

\* Bottlenecking

\* Theft

**Insight 5:** Malicious Nodes

**First Question:**

What is it that keeps nodes from going bad?

???

\* Monetary incentives

\* Good faith

\* Exclusion from blockchain

\* Higher pick rate

**Explanation:**

The double-spend attack isn't the only threat to a blockchain and is far from being the most destructive. \*\*Malicious nodes\*\* are an innevitable property of open, distributed networks. Some join trying to distrupt the network, others to steal assets. Taking control of the network is also possible, but comes at a great cost. What follows is a succinct list of possible attacks:

\* 51% attack

\* Sibyl attack

\* Replay attack

### 51% attacks

A single entity controlling more than half of the network resources is also controlling the network as a whole. Blocks can be rewritten, accounts can be emptied, transactions can be ignored. This is probably the most destructive attack that could happen.

Possible in theory, it is very hard to be brought into practice: the size of the network is proportional with the difficulty of pulling of such an attack. But once this happens, the attack has a 100% success rate.

### Sybil attacks

An attacker fills the network with nodes whose goal isn't to control the network per se, but rather help the execution of other attacks. If there are malicious nodes in the network, they can ignore certain transactions, allowing double-spending attacks to happen and even isolate people from the network altogether (DoS attack).

### Replay attacks

A subtype of man-in-the-middle attack, where the transaction packet sent over the network is intercepted and replayed after a certain delay. With no prevention in place, an account can be drained by repeatedly sending copies of same transaction to the nodes.

### Conclusion

To prevent nodes from trying to interfere with the network, the blockchains rewards (\*block reward\*) the miners proposing a valid block, that ends up in the blockchain.

**Second question:**

Identify a blockchain attack that, although unlikely, has 100% rate of success.

???

\* 51% network control

\* Sybil attack

\* Double-spend attack

\* DDoS

### Mining

**Description:** All about finding that next block

**Insight 1:** Mining

**First Question:**

Choose from the following list the case in which successful mining doesn't yield any reward:

???

\* There is a longer branch being built on.

\* Double-spend attack.

\* Successfuly mining blocks always yields rewards.

\* The nonce is 0.

**Explanation:**

\*\*Mining\*\* is part of the process of securing the network. Miners race one another in solving a computationally-difficult, cryptographic puzzle that would entitle them to a reward. We can only talk about \*mining\* in the context of proof-of-work consensus protocols.

Bitcoin is a blockchain that implements such a consensus algorithm. Miners race to find the right \*\*nonce\*\* that would make the the double SHA-256 hash start with a fixed number or more of `0`s. The number of 0s is called the \*mining difficulty\*.

They are incentivized to do this through a block reward - the first miner to propose a valid block receives some coins. To make sure a block to be valid, the miners have to check the transactions they include. By doing so, they protect the network.

There is also the concept of \*side branches\*. Side branches occur when two or more nodes find a block approximately at the same time. If some of the transactions are shared, a single branch will be merged into the main chain while the others are ignored.

Nodes are coded to prefer the longest chains, the ones that required the most computation power to create. In a sense, the lottery is not only about finding the right nonce, but also convincing other miners to work on top of your block.

**Second question:**

What effect does mining have on the structure of the blockchain?

???

\* The blockchain grows.

\* The miner receives a block reward.

\* The mined block contains the block reward transaction.

\* The last block is skipped.

**Insight 2:** How Can One Mine?

**First Question:**

In the context of Bitcoin, which piece of hardware is the most effective for mining?

???

\* ASIC

\* GPU

\* CPU

\* FPGA

**Explanation:**

Bitcoin's puzzle is computationally-difficult. That means that you need processing power to compute the double hash effectively. Back in 2009, mining using your PC's CPU was still possible. The reward at that time was bigger (50 BTC), but its USD equivalent was a lot less (under 10 cents/BTC).

People quickly realised that GPUs perfomed 50-100x times better than CPUs. But GPUs are better at handling memory-hard puzzles and specialized hardware started coming out on the market: FPGAs, or field-programmable gate arrays.

The increase in Bitcoin price made mining a highly profitable field. Many companies invested in finding new hardware that would give people the edge over others. As such, a new industry that produced ASICs (application-specific integrated circuits) emerged.

The most expensive ASIC has a hash rate of `13.5 TH/s` and it sells for $3,000. Latest generation CPUs can perform at most couple hundreds hashes per second (200-300 H/s), which means the ASIC is \*40 trillion times\* faster.

The problem is ASICs take a long time to ship. Some people even think that the producers use them to mine bitcoins before sending them to users. And by the time they reach the customers, they won't be as effective as they might have been because the network's processing power is only increasing over time.

**Second question:**

In the context of Bitcoin, what is the name of the variable used to solve the challenge?

???

\* Nonce

\* Hash

\* Backlink

\* Rock

**Insight 3:** Mining Pools

**First Question:**

Identify a pro and a con of mining in a pool:

pro: ???

con: ???

\* steadier income

\* downtime independent of miner

\* higher uptime

\* high reward

\* no fees

**Explanation:**

Bitcoin mining has become too difficult for anyone to have a chance of finding a block by themselves, even with the most expensive ASICs. The CPU solo mining feature was even removed from the main Bitcoin client in 2016. Luckly, the Bitcoin protocol supports pooling.

A \*\*pool\*\* is a distributed network of computers that combines the processing power of all members to increase the chance of finding the nonce. Because the block must contain some of the miner's information, the block is mined in the name of the pool.

On `blockchain.info` you can see the last blocks mined and who relayed them. Most of the times, you will find pool names - \*ViaBTC\*, \*BTC.com\* or \*F2Pool\*.

So users give up the (extremely small) chance of getting the full reward for themselves for a more steady income. If the pool is to find a block, the reward is distributed to members following a set of rules.

One of the more popular ways of splitting the reward left after the pool manager had taken their fee is to count \*mining shares\*. A mining share is a hash proposed by a miner that doesn't quite meet the network requirement (say 15 0's) but it is close to it (12-14 0's).

Also, because the rework is split based on the work done, the miner who found the right nonce won't get any extra shares - so if another miner had more shares, they will recieve a bigger cut.

Mining pools can also swap between mined coins to maximize the profit. This implies that the two puzzles require the same computations.

But mining in a pool has its own downsides: if the pools goes offline, the miners won't produce anything either. There's also a chance that the pool manager keeps the entire reward for himself.

**Second question:**

For most cryptocurrencies, solo mining stops being profitable after the first few months. To increase the chance of payouts, it is recommended to join ???

\* a mining pool

\* a mining tree

\* a congregation

\* a mine

**Insight 4:** Energy Consumption and Ecology

**First Question:**

The three things you need for setting up a mining data center are:

???, not to miss out on blocks

???, to pay the minimum price to run the rig

???, to help with cooling

\* good network

\* cheap power

\* cool climate

**Explanation:**

High processing power doesn't come cheap. The same $3,000 piece of hardware eats up 1,320 kW, which is around 31.5 kW/h running non-stop every day. Depending on the cost of household electricity, this might range from $1.48 a day in Ukraine to $5.67 a day in California, US.

One of the ofter overlooked cost is the \*cooling cost\*. The ASIC's built-in fan works for the individual piece of hardware in a temperate climate, stacking them may need external cooling.

The price per kW/h decreases in case of industrial use (can reach up to 50% discount). Entire data centers have been built in the past years in countries with \*cheap electricity\*, \*cold weather\* and \*fast internet\*, to the point where individual mining can only be done for fun.

As of early 2018, in percentage of the world's electricity consumption, the entire Bitcoin network is estimated to consume `0.24%`. Bangladesh is the closest country in terms of energy consumption.

**Second question:**

Other than electricity consumed for making the chip and running it, what else does consume the most energy during mining?

???

\* Cooling

\* Distributing work

\* Hashing

\* Collecting shares

### Wallets

**Description:** All about blockchain identity and addresses.

**Insight 1:** Identity

**First Question:**

none

**Explanation:**

The 'key' behind any blockchain is the cryptographic protocol on which it runs. Asymmetrical cryptography, or public-key cryptography, is a system that uses a pair of two keys for authentication and authorization: a public key and a private key. The pair of keys is \*your address on the blockchain\*.

The transaction example in one of the first workouts was:

```bash

Alice sends Bob 10 coins

```

But this isn't really how transactions look like in the real world. An actual Bitcoin address looks like this:

```bash

1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa

```

The private key is randomly generated using a set of rules. The public key is derived from the private key. Their purpose is given away by their name: the public keys are made available to everyone, while the private keys should be kept secret at all times.

Without going into too much detail, consider this simple example:

```bash

Randomly generated private key (8 chars):

aNsjti!k

```

The private key is always 8 chars long. As the algorithm to compute the public key, we concatenate odd index characters in the private key (starting at index 0):

```bash

Derived public key (4 chars):

Njik

```

In the above example, the key can be brute-forced easily. In the real world, computing the private key from the public key should be infeasible.

On the blockchain, someone's identity is their public key. The private key is the proof of identity. By themselves, neither key can be used to authorize transactions. Combining them, however, yields what is called a "digital signature". The digital signature authorizes the blockchain to record actions as being taken by the signing user.

There are two algorithms, one for signing and one for verifying, that make use of the public and private keys:

```bash

Signature generation S

S = (message, private key)

-> message signature

Signature verification V

V = (message, public key, message signature)

-> outputs true or false

```

The actual protocol for computing it is public for transparency reasons. You can find a link attached to the Bitcoin wiki about the process.

## Practice

What is the type of cryptography used for authentication on a distributed network?

???

\* Public-key cryptography

\* A new block is added

\* RSA

\* The network achieves decentralization

\* New tokens are created

\* Symmertrical cryptography

**Second question:**

none

**Insight 2:** Wallets

**First Question:**

What does a cryptocurrency wallet contain?

???

\* A list of public and private key pairs

\* A list of account ID and balance pairs

\* A QR code

\* A list of coin IDs

**Explanation:**

Each Bitcoin address has two keys: a public and a private one. You only share the public one and keep the private one secret.

A (digital) \*\*wallet\*\* is a piece of software that can listen to and interact with the network. In case it is also connected to the network, it can handle transactions and display the sum of UTXOs that had as recipient \*an address generated by the wallet\*.

Probably one of the bigest misconception people have about wallets is that your wallet is your balance on the blockchain. A cryptowallet is rather a collection of identities on the blockchain, each with its own UTXOs. Because of that, in general, but also in particular for Bitcoin, it is also recommended to never use an address more than once (we'll cover the reasons why soon).

Usually, the main blockchain client has an integrated wallet as well (the possibility of generating private keys and computing public ones). Another feature that you must be looking for when choosing a wallet is its ability to encrypt the private keys.

The encryption is necessary because the private key would be otherwise saved in plain text - not necessarily something you want if you have invested a large sum of money into the coin.

There are also paper wallets. They are usually a printed page that contains 2 QR codes, that carry the public key and the private key, respectively.

Physical wallets are pieces of hardware not connected to the internet that can run wallet applications. They usually require purchashing the actual device, but can generate new addresses, a feature the paper wallet doesn't have.

**Second question:**

What do you need to obtain an address' balance for a particular cryptocoin?

???

\* The public key and a blockchain connection.

\* The private key and a blockchain connection.

\* The public key.

\* The public/private key pair.

**Insight 3:** Keeping Your Wallet Secure

**First Question:**

Which type of wallets is inherently more secure?

???

\* Offline wallets

\* Online wallets

\* Paper wallets

\* Mobile wallets

**Explanation:**

### Are cryptowallets secure?

They are generally secure, up to a degree. While the cryptographic mechanisms are secure for now, the insecurity is given by the type of wallet you're using: online, mobile, paper, hardware, etc.

It is more likely for an online wallet to be stolen from a server than an offline wallet to be stolen from you. Even the simpler paper wallets work great long-term, if you take care of them right.

Remember that this is all a game of keeping your private key secure: if you lose it or share it, all your money are gone. Similarly, if someone fools you into sending them your money, there's not much you can do to recover them.

### Are cryptowallets anonymous?

Unless you are storing an anonymous coin like XMR (monero), best you could hope for is pseudo-anonymity. While an wallet doesn't store any personal information, the transactions are there on the blockchain. As it will be discussed in the next workout, there are some ways in which you could trace someone's identity. But unless you are dealing with very large sums, there's no need to worry too much.

**Second question:**

What is the level of anonymity of most wallets?

???

\* Similar to one of the coin

\* Anonymous

\* Public

### Bitcoin Anonymity

**Description:** Is Bitcoin really anonymous?

**Insight 1:** Anonymity

**First Question:**

What is anonimity?

???

What is privacy?

???

\* The condition in which the real identity of the person is unknown.

\* The condition in which the actions of a person are kept secret.

\* The condition in which both the identity and actions of a person are kept secret.

**Explanation:**

There's a saying that goes "anonymity loves company". If you are the only one to buy a laptop on a certain day, then a transaction is surely yours. But if 10 different people made the same purchase, yours suddenly becomes harder to trace.

The same is true for any blockchain. The more people using the network, the harder (but no impossible) it is to track individual addresses. Notice that we are not necessarily talking about identifying the person (which is also possible), but rather linking transactions together.

Regarding Bitcoin, we can talk about a \*pseudoanonymity\*. Say you were the recipient of two transactions in

the past, two UTXOs of 1 and 2 bitcoins, respectively. Now, if you are looking to sell more than 2 bitcoins at the same time, you would have to consume both UTXOs.

Unless you are the owner of the addresses, you wouldn't be able to digital sign the transaction combining them. Hence, we can guess that they belong to the same user/wallet. Your identity wasn't revealed, but your transactions can now be tracked.

**Second question:**

What is the level of anonymity provided by the Bitcoin network?

???

\* pseudoanonymity

\* complete anonymity

\* no anonymity

**Insight 2:** How to De-Anonymize Bitcoin

**First Question:**

What is a property that is `not` required for one's Bitcoin activity to be unlinkable?

It should be hard to ???

\* look up previous out transactions of the same user

\* link together addresses of the same user

\* link different transactions made by the same user

\* link the sender to its recipient

**Explanation:**

In the previous insight, we mentioned a specific way of linking transactions to an online identity: joint payments. Most of the times, however, a single redeemed UTXO is enough to track someone if the transacted value is specific enough.

A bitcoin transaction can represent up to 8 digits after the decimal points (`0.000000001 bitcoin` is called a `satoshi`), so the distribution space is large. If someone is to know the exact value of your transaction and a timeframe in which you had intiated the transaction, there's a chance they could track your address. If we assume a uniform distribution over the `[1, 10^8)` interval, and know that the maximum number of transactions per second Bitcoin can handle is 7, it would probabilistically take around 14 years for a second transaction with the same exact decimals to appear. While, in reality, the distribution is not uniform, you can still be certain that, for a longer or shorter period of time, the transaction can be correctly identified.

And depending on where they found out about the exact value of the transaction, they can even know your real identity. By transacting with anyone in person, you are linking your identity to the address they received coins from. By donating bitcoins and sharing the donation confirmation to incentivize others to do the same, you are also revealing your blockchain identity.

Why should you care though? Well, chances are you won't have that many bitcoins anyway. If you made a good investment and have a large sum of money in your bank account, there's a slim chance of anyone knowing it unless you told someone. But your Bitcoin address is public and will always be public.

If you decide to buy more bitcoins because you belive it is the coin of the future, what stops others from trying to steal your private key? Attention to details and precaution go a long way. To make the whole tracking process harder, \*mixing services\*, or simply mixers, appeared.

**Second question:**

Which of the following practices reduces the chances (ever so slightly) of being de-anonymized?

???

\* Creating a new address for each transaction

\* Using the same service repeatedly

\* Using an online exchange

\* Combining multiple outputs into inputs for the same transaction

**Insight 3:** Mixing

**First Question:**

Why would a bitcoin holder use a mixer?

???

\* To reduce the chance of being tracked

\* To gamble

\* To meet other holders

\* The network demands so

**Explanation:**

A \*\*mixer\*\* (tumbler) is an intermediary service that makes sure the bitcoins you sent to your recipient aren't yours.

You can think of it as an intermediary to which multiple people send \*their bitcoins\* (as UTXO) and a recipient address and the mixer makes sure to send further \*other bitcoins\*. But `0.0023256` in would still be `0.0023256` out, even redeemed through another UTXO.

If someone tracks your address, they would be able to find out your recipient. To avoid that, mixers may take fixed payments of `0.01` bitcoins and would send the reminder back to one of your addresses later. The out payments are also released after a random interval of time.

Legality is another issue mixing services have to deal with. Because they can be used to laundry money, to protect themselves and their business, they may refuse to mix transactions of more than a certain threshold.

But remember that mixers are centralized. They are controlled by single entity has to power to store your bitcoins and never actually make the transaction forward. Due dilligence is needed when using such a service to ensure that it is reputable and trustworthy.

**Second question:**

Why would a bitcoin holder use a mixer?

???

\* To reduce the chance of being tracked

\* To gamble

\* To meet other holders

\* The network demands so

### Altcoins

**Description:** Altcoins and the reasons for their appearance.

**Insight 1:** Altcoins

**First Question:**

If an altcoin is created by modifying the Bitcoin source code, where can you interact with the altcoin, other than exchanges?

???

\* The altcoin's blockchain

\* The Bitcoin blockchain

\* The Ethereum blockchain

\* Its smart contract

**Explanation:**

Bitcoin hasn't been a hit from the beginning. It took a while for people to see for themselves why bitcoins are useful and get used to using them. But there were some people that didn't want to just get used to its limitations: pseudoanonimity, long transaction confirmation time, low transaction throughput, difficulty in obtaining competitive mining hardware and high associated costs, etc.

Bitcoin source code has always been open-source. For someone to start their own \*\*blockchain\*\*, all they had to do was to copy (\*fork\*) the source code, find all occurences of `Bitcoin`, `BTC`, and `bitcoin`, replace them with the new name of the coin and run the code.

This method, however, wouldn't change anything to how the system works. One of the first notable altcoins was launched in 2011: `Litecoin`. Litecoin blockchain is a fork of Bitcoin and, consequently, is very similar to it.

But similar doesn't mean the same. While both have their coins as the central asset of the blockchain, Litecoin was created to speed up the transactions. Litecoin's blocktime is 2.5 minutes, a quarter of Bitcoin's. To match the time, the maximum supply of litecoins is 4 times the max supply of bitcoins, up to a total of 84 millions.

In terms of consensus algorithm used, Litecoin is secured using proof of work, but this time with a memory-hard hashing algorithm named `Scrypt`, rather than plain \*SHA-256\*.

However, Litecoin hasn't managed to solve the anonymity problem Bitcoin had. This justified the creation of new altcoins like `Dash`, `Monero`, or `Zcash`, with a goal of making the users untraceable on the blockchain.

The above mentioned coins and multiple others all have their own merits of improving the decentralized, transaction-focused system that Bitcoin pioneered. But this was nothing new. The true game changer, the blockchain that allowed Turing-complete smart contracts to be deployed and run was `Ethereum`.

Suddenly, blockchain wasn't just about trading coins. New businesses started to follow the trend and release their own token. Ethereum made that process much easier: it has created a standard of tokens (ERC-20, although not the only one) that it can recognize and trade on the already secured \*\*main network\*\*.

`EOS`, `NEO` and `Qtum` are other networks that are meant to be competing with Ethereum on the same smart-contract market.

**Second question:**

If a token is created following an established token standard, will it need its own blockchain to secure the transactions?

???

What platform released a standardized token format?

???

\* No

\* Ethereum

\* Yes

\* Bitcoin

\* Litecoin

**Insight 2:** Why Is It Worth Having Multiple Coins?

**First Question:**

Identify one way of creating new cryptocurrencies:

???

\* Forking already established ones

\* It can't be improved

\* Adding new nodes to the network

\* Modifying the mining puzzle

**Explanation:**

Altcoins represent competition and competition is always good. It drives change and it allows people to find something they actually belive in.

It also means that you don't have to necessarily build everything from scratch anymore. You can use and build upon existing code. And since Ethereum, the process is faster and simpler.

The blockchain has that distributed, open-source culture to the point where "decentralization" becomes the antithesis of "secrecy". If a new blockchain-related company catches your eye, you should always be looking for their white-paper, in which you can usually find the reasoning behind and goal of the project.

By researching those that succeeded in a competitive market makes it easier to spot those projects that are only built to be a scam. As it was the case with BitConnect recently, the hype and the fear of missing out can lead to bad decission-making. A strong community is more likely to filter out potential harmful agents.

One of the biggest concerns right now is the rise of ICOs that receive hundreds of millions in crowdfunding campaigns only to die a quick death. So, if anything, helping the ecosystem through crownfunding is encouraged and may even pay off in the future, but always research the team and and read the whitepaper yourself.

**Second question:**

Blockchain-based projects, as any other start-ups, are more likely to fail than succeed. Therefore, is an altcoin guaranteed to be valuable in the future?

???

\* False

\* True

### Consensus Algorithms

**Description:** Ways of securing blockchains

**Insight 1:** Consensus Algorithms

**First Question:**

In case the node picked to append the next block is based on the computing power it has, what is the process called?

???

\* Proof of work

\* Proof of stake

\* Proof of concept

\* Proof of power

**Explanation:**

\*\*Consensus algorithms\*\* are the blockchain bodyguards. They prevent \*double spending\*. Some blockchains depend on participants investing in hardware (miners), while others on people investing in their token (stakers). Some depend on a third party organization (authority), while others on randomly selected groups of participants. So far, no algorithm that would work great in all cases has been found; they all have their pros and cons.

The blockchain needs them because, if no majority agrees on the state, there is no single source of truth,and hence, no blockchain. Furthermore, not all nodes in the networks are honest, such that consensus algorithms must account for and hold for adversarial condition. A list of proposed solutions to secure the blockchain is presented below:

\* Proof of work

\* Proof of stake

\* Delegated proof of stake

\* Other solutions to the Byzantine Generals Problem

Consensus has to be reached about blocks to be added to the blockchain. Usually, the system behind the blockchain has a method of incentivizing users to reach consensus. In a proof of work environment, the incentive is the block reward - a number of tokens generated by the blockchain out of "thin air" given to those that secured the blockchain.

In proof of stake, people vouch for blocks they think should be added to the blockchain by staking coins. Most of the times, the reward is the sum of transaction fees for that block or a considerably smaller block reward - similar to an interest rate rather than an one-time lottery ticket. Qtum and Peercoin both use proof of stake.

To have a chance of winning the block reward in proof of work, you have to be mining. A full node in the Bitcoin network does not receive anything. Qtum, on the other hand, allows everyone that holds coins in a wallet to stake and have a chance at winning the block reward.

**Second question:**

In the context of Bitcoin, what is the reward for running a full node?

???

\* There is no reward.

\* The node reward is equal to the block reward.

\* Exactly 10% of the block reward.

\* A small fee off of ever confirmed transaction.

**Insight 2:** Proof of Work (PoW)

**First Question:**

In the context of Bitcoin, what is the puzzle solved?

???

\* Finding a hash less than a target

\* Computing private keys

\* Reversing a hash

\* Finding a Cunningham chain

**Explanation:**

\*\*Proof of work\*\* is the original consensus algorithm for blockchains. It is based on the 'work harder, not smarter' principle: you can't trick the blockchain if honest miners are working harder than everyone else is.

PoW blockchains will always favor the longest chain of blocks, that is the chain that had the most work put into it. Assuming most miners work on the same chain, that will be the one to grow the fastest and be the most trustworthy. Any PoW blockchain is safe as long as a minimum of 50% of miners are honest.

The advantage of PoW algorithms is that they are straightforward: everybody solves the same puzzle; but finding an adequate puzzle is difficult. Some of the properties of a proper problem are:

\* Should be difficult to compute.

\* Should allow for parameterizable cost: the difficulty should grow or shrink relative to the total computing power in the network.

\* The solution should be trivial to verify as every node in the network will have to carry through with the process.

The most common puzzles are of two types: computational heavy (many operations) or memory heavy (big input data). One the one hand, the double SHA-256 function used in Bitcoin (and others) has to be computed for every nonce in a very large interval. On the other hand, in Ethereum, the complexity is given by the large data set (1GB) rather than a variable nonce.

PoW requires miners to invest in mining hardware: ASICs or GPU cards. This is an \*\*external investment\*\* where the retailers that sell the hardware have no direct involvement in the development of the blockchain (other than profiting off of it). On top of the already elevated overhead cost of electricity and the low transaction throughput, there are enough compelling reasons for blockchains to migrate from proof of work to proof of stake.

**Second question:**

Who does PoW scheme usually advantage?

???

\* Big players with lots of resources

\* Average miner

\* Those with over 1000 coins

\* No one in particular

**Insight 3:** Proof of Stake (PoS)

**First Question:**

What is the main difference between PoW and PoS?

???

\* No mining hardware needed

\* No verification needed

\* Miners verify

\* Miners are called stakeholders

**Explanation:**

\*\*Proof of stake\*\* is the new, cool kid on the block. The hard-hats miners are gone and have been replaced by \*\*minters\*\*. Minters stake, or freeze, some of their coins to have a chance of being selected to propose a new block that points to one of the previous blocks. The probablity of being selected is proportionate to their share in the total stake pool: 3 staked coins in a pool of 100 coins means a 3% chance of being selected to propose the next block.

The main advantage of PoS is that the verification can be done with \*no performance hit\* on an average PC. Moreover, by having to purchase coins instead of mining hardware to stake, the investment stays within the blockchain. In theory, acquiring 51% of the total stake pool is a lot easier to achieve than having 51% of the procesing power in a PoW blockchain. Even then, one is not incentivized to attack the blockchain because they risk losing their investment altogether because of the coin depreciation.

The downside is a particular node stance called \*nothing at stake\*. In chain-based proof of stake (more on this later), unless there is a penalty in place for voting for both branches in a potential fork, no consensus might be reached. The nodes do not even have to be malicious: voting on both branches to avoid losing the stake by bidding on the wrong one is acceptable from a self-centered, economical point of view.

```text

C D

| |

\ /

B

|

A

```

A validator can easily propose a new block to follow C and a new block to follow D because, compared to PoW, proposing the block has practically inexistent computational cost.

**Second question:**

Identify the weighting factor for choosing validators in a PoS environment:

???

\* The amount of coins one has.

\* The computing power one has.

\* The time since one's last transaction.

\* One's time spent mining.

**Insight 4:** Delegated Proof of Stake (DPoS)

**First Question:**

Why is DPoS more centralized than PoS?

???

\* Only a limited number of nodes can propose blocks

\* Only nodes with more tokens than average can propose blocks

\* On the contrary, PoS is more centralized

\* Because it is faster

**Explanation:**

Similar to how the election system works in a democracy, \*\*delegated proof of stake\*\* implies nodes voting delegates to validate blocks for them. At any time, there are at least 50-100 validators.

Validators are picked at random to push their block. The small number of delegates allows efficient distribution of block proposing tasks and validation. But validators are not in for life: if they miss the target repeatedly or propose invalid transaction, they are replaced using the same voting system.

Having nodes collaborate rather than compete for publishing blocks (specific to PoW, PoS), the block publishing times can get as low as 1 second. The trade-off is the relative centralization that occurs by having a few validators. Steemit and EOS are some examples of blockchain with high throughput that implement DPoS.

**Second question:**

What do most nodes in a DPoS environment vote for?

???

When does a new vote happen?

???

\* Block validators

\* When validators can't keep up the pace

\* Blocks

\* Every 2 hours

\* The price of the coin in USD

\* When the price of the coin drops by more than 15%

**Insight 5:** Byzantine Fault Tolerance (BFT)

**First Question:**

In general, which solution to the Byzantine Generals Problem is the fastest (first answer) and which is the slowest (second answer)?

???

???

\* Federative byzantine agreement

\* Proof of work

\* Proof of stake

\* Delegated proof of stake

\* Byzantine fault tolerance

**Explanation:**

A problem specific to distributed systems is the \*\*Byzantine Generals Problem\*\*. The problem states that a reliable computer system must handle malfunctioning components that give conflicting information to different parts of the system.

The problem is presented as follows:

A group of Byzantine generals and their divisions camped around a city. They can only communicate by messengers to come up and agree on a battle plan. To save the city, insurgents infiltrate the army and try to confuse the generals by posing as messengers. Moreover, some of the generals themselves may be traitors. In case the messages are oral (easily forged), the problem has been proven solvable if and only if \*more\* than two-thirds of the generals are loyal (1 general can confound 2 others). If the messages are unforgeable, the problem is solve for any number of generals.

\*\*Byzantine fault tolerance\*\* characterizes a system that tolerates the class of failures that may occur in the context of BGP. The members of this class are considered to be the most severe and difficult to deal with. All consensus algorithms mentioned previously are solutions to this problem.

Another solution is called \*\*Federated Byzantine Agreement\*\* (FBA). Stellar and Ripple are blockchain that implement this protocol. The main idea is that generals (pre-selected nodes) are responsible for their they own chains and focus on sorting messages to establish the single source of truth.

Ripple fundation select the validators themselves, while Stellar allows nodes to choose the validators to trust. So far, FBA is one of the fastest and most reliable classes of consensus algorithms.

**Second question:**

What is the problem that most consensus algorithm are trying to solve?

???

\* The Byzantine generals problem

\* The Byzantine fault tolerance Problem

\* The blockchain problem

\* Centralization problem

### Forking

**Description:** Forking blockchains

**Insight 1:** What Are Forks?

**First Question:**

What is a soft fork?

???

What is a hard fork?

???

\* Ignoring a state/block in the future

\* Modifying states that are considered to be immutable

\* Redistributing the balance of one account to others

**Explanation:**

On the blockchain, the majority dictates the current state of the chain. In theory, a blockchain is immutable. In practice, the immutability follows some constraints (more than 50% of nodes are honest). As such, if the network agrees, the state can be modified. The process is called \*\*forking\*\*. There are two types of forks: \*hard forks\* and \*soft forks\*.

A \*\*hard fork\*\* is a permanent divergence from a previous state of the blockchain. They usually occur in case of force majeure events, like reversing transactions, changing the consensus algorithm or enforcing new rules. Nodes also require an updated version of the blockchain client software that accounts for the changes to be able to follow the fork. A successful hard fork implies the majority of nodes upgrading and following the new path.

A \*\*soft fork\*\* is a less drastic divergence: it usually happens to yield past transactions invalid or simply freeze wallets. The nodes that agree to the fork check the new transactions against the new conditions. A soft fork is backwards compatible - as long as more than 50% of nodes update, the blocks that are valid for those will also be valid for the rest of the network. However, the nodes that haven't upgraded might still propose invalid blocks.

**Second question:**

Determine the truth value of the following sentence: A blockchain is immutable under any circumstances.

???

What is one case in which the ledger can be modified?

???

\* False

\* The network initiates a hard-fork

\* True

\* The network initiates a soft-fork

\* There is no such case

**Insight 2:** Forking to Create Altcoins

**First Question:**

Identify what `cannot` be changed in the source code of a fork for it to be considered as a valid fork:

???

\* Anything can be changed

\* The coin name

\* The mining puzzle

\* The genesis block

\* The estimated time between blocks

**Explanation:**

When referring to altcoins, \*\*forking\*\* has two different meanings. A source-code fork implies a new blockchain altogether. There are also hard-forks to create altcoins that share the initial blockchain's history up to the parent block.

Let's focus on Bitcoin forks: in a previous insight, Litecoin was mentioned. Litecoin is a source-code fork, which has a Block 0 that is different from Bitcoin's Block 0. Over the years, multiple Bitcoin source-code fork altcoins appeared and, most of the time, inherently different from Bitcoin.

Blockchains are under constant development. New transaction types are added, scaling improvements are proposed and bugs are fixed. Usually, the community has the same idea about the future of the blockchain. In the rare cases in which opinions are clashing, a resonable solution is to fork the network.

The first Bitcoin hard-fork happened in August 2017, giving birth to Bitcoin Cash. The fork initiators did not agree with the Bitcoin Core team on what should be done to solve Bitcoin's scaling problem. Consequently, at block 478558, every wallet on the main network that had BTC also had an 1:1 equivalent amount of BCH.

**Second question:**

What is it that is forked off a blockchain to create an inherently different one?

???

\* The source code

\* The state of the blockchain

\* The source code and everything up to block X

\* The nodes

**Insight 3:** Forking to Reverse Transactions

**First Question:**

What conditioned the Ethereum hard-fork?

???

\* The DAO heist

\* A 51% attack

\* Community divergences about scaling

\* Etheretum had only its source-code forked

**Explanation:**

Ethereum found itself in a rough spot after theDAO heist: the initial soft-fork that froze the thief's wallet was followed by a hard-fork that modified the smart contract and allowed investors to withdraw their ETH.

But a part of the community was not in agreement with the hard-fork. They strongly belived that the blockchain should be immutable unless attacked and no one should tamper with it, not even the majority.

As such, a part of the minority decided not to upgrade and continue building on the old chain under the name of \*Ethereum Classic\*.

**Second question:**

Which of the following statements describes Ethereum Classic?

???

\* Ethereum network pre DAO hard-fork

\* Ethereum network post DAO hard-fork

\* Ethereum network with a PoS consensus algorithm

\* A sham Ethereum network

### Cryptoeconomics

**Description:** Cryptography and economics.

**Insight 1:** How Do Cryptocurrencies Acquire Value?

**First Question:**

In general, what is it that gives value to items?

???

\* People find them valuable

\* Their price is low

\* They make a good investment

\* They are useful and long lasting

**Explanation:**

In general, how do things acquire value? The answer is straight-forward: they either serve a purpose/solve a problem (PC, cars, etc.) or enough people consider them to be valuable (currencies). The dollar/british pound/yen are useful because they can be used to buy food, a house or a car. Only humans and other living beings are intrinsically valuable since they can create value.

Money is a concept created by us. Unlike bartering, it is way simpler to trade a chicken for coins and subsequently trade the coins for some wheat. A piece of paper on which you write $10 has no value, nor the one on which you write $100. Objectively, the $10 or $100 inscribed on the banknote shouldn't make any difference. But because we recognize some banknotes as valuable, it is a difference worth $90.

That's money in general. With cryptocurrencies, the question is how much we value their attributes: global, intermediary-free transfers, trading speed, security, anonimity (in some cases) or the innovation they represent. Bitcoin can be transferred anywhere globally, in a secure manner and relatively fast. The number of products for which a bitcoin can be exchanged also influences its value. However, the Bitcoin blockchain usage is limited to being a ledger for its coin. Ways to extend the basic capabilities of Bitcoin are being researched but everything has to be adopted through community's consensus.

Ether, on the other hand, powers the Ethereum network's smart contract system. On top of what Bitcoin does, Ethereum can also execute code files in exchange for gas. Gas can only be bought with ether (or rather its smallest unit, wei). As such, ether is used to pay for every transaction in order for it to be registered on the network. So, as long as you need/value what the Ethereum network can do, ether will also be valuable for you.

**Second question:**

Objectively, which of the following two cryptocurrencies is more valuable?

???

\* Ether

\* Bitcoin

\* They are as valuable

**Insight 2:** Why Is the Price Fluctuating so Much?

**First Question:**

What is the main dirver of a cryptocoin price?

???

\* Traded volume

\* Circulating supply

\* Market cap

**Explanation:**

In terms of the actual price (a coin's value in a certain fiat currency), no one can predict it exactly: who would have hoped that a bitcoin would reach $20,000?

In total, cryptocurrencies' market cap approached 1 trillion in December 2017. Bitcoin's market cap never exceeded $320 billions. Bitcoin traded volume was way smaller: maybe $20 billions on all exchanges corroborated, during peaks. In other words, it would be easy for someone that owns a relevant % of the total number of coins to manipulate the price in either directions. If the tendency of the market is to drive the price down, it's called a bear market. A bull market is the opposite: prices are going up.

It's hard to have stability when most cryptocurrencies have no commodity or governmental back-up. Only lately had platforms started to appear (read ERC20 tokens vs Bitcoin forks) for which their coins can be exchanged for a service, hence creating their own small economy. This is similar to how forex markets (where currencies are traded for one another) represent the economy of the countries: USD as the economy of USA and GBP as the economy of Great Britain.

Moreover, there are no regulations: pump and dumps, inside trading or other (normally illegal) methods of price manipulation/prediction are not punished.

Of course, markets in general cannot be explained in 5 minutes and many other things can be said about how cryptocurrency markets work. Given the how broad and unpredictable the topic is, self-study is required in order to understand all the mechanisms.

**Second question:**

What is the metric that describes how valuable a is cryptocurrency?

???

\* Market cap

\* Circulating supply

\* Node investment

\* Volume

**Insight 3:** Cryptoeconomics

**First Question:**

What are the means through which long term holding of properties are encouraged?

???

\* economic incentives

\* economic punishments

\* cryptographic algorithms

**Explanation:**

Technically, \*\*cryptoeconomics\*\* is the study of incentive mechanisms that are applicable to a cryptographically secure and distributed environment. In other words, it refers to a system in which past messages are verified through cryptography, while economic incentives and punishments are used to encourage the long term holding of properties and reduce the malevolent behavior.

The resulting digital economies are governed by computer code. However, much like a regular economy, they still require definitions for policies, may them be fiscal, monetary or privacy-related. Moreover, they also need the platform maintainers, that would update the protocols overtime in response to changes or needs.

The subject is vast and a solution is far from being platform agnostic. If you want to research the topic more, bookmark the insight and you should receive a link to do so.

**Second question:**

What are the means through which malevolent behavior is prevented?

???

\* economic punishments

\* economic incentives

\* cryptographic algorithms

### Ethereum I

**Description:** Ethereum, the game changer

**Insight 1:** Ethereum and Ether

**First Question:**

What is the main purpose of the Ethereum Virtual Machine?

???

\* To run applications

\* To track digital coins ownership

\* To speed up digital transactions

\* To create and breed cryptokitties

**Explanation:**

\*\*Ethereum\*\* is the blockchain that changed how the game is played. Before its creation, most cryptocurrencies were the cryptographic equivalent of fiat currencies. Once it was launched, cryptocurrencies started to represent something: a token through which people could interact with a given platform.

Ethereum is a distributed supercomputer. Its main feature is the ability to store and run applications. It was created by \*Vitalik Buterin\*, back then only 21 years old. The code is open-source and can be found on GitHub.

Applications are run using blockchain's turing complete virtual machine: Ethereum Virtual Machine or EVM. Every node in the network will therefore run the code you deploy on the blockchain. How do you make sure people don't run malicious code or infinite loops?

Enter \*\*ether\*\*, Ethereum's token: make people pay for the time and memory used by the network to run their code or transfers. So if someone is trying to do something malicious, their transaction would be dropped but their transaction fee retained.

A transaction is paid for in \*gas\*, that can be bought using ether subdenominations. Usually, gas price is expressed in gweis (giga weis, 10^9 wei), where 1 ether is equal to 10^18 weis. Transaction fees then become the product of the gas price and the amount of gas used.

Until recently, Ethereum used \*Ethash\*, a memory-heavy PoW algorithm, to create blocks. Average block time is 14 seconds, compared to Bitcoin's 10 minute block time. Because PoW can end up consuming more electricity than some countries (see Bitcoin and its evergrowing thirst), Ethereum developers are trying to migrate to a PoS algorithm, called `Casper Protocol`. Until that is possible, a hybrid PoW/PoS algorithm is going to be implemented: every 49 regular blocks would still be generated using the PoW algorithm, while nodes would stake to append a special 50th block, called checkpoint block.

**Second question:**

What is the biggest expected change in the Ethereum blockchain protocol, that will effect how rewards are distributed?

???

\* Swapping PoW with PoS

\* Removing the preriodical block reward reduction

\* Further reducing the block time

\* Removing the gas cost of running DApps

**Insight 2:** What Does Ethereum Do Differently?

**First Question:**

Fill in the gaps such that the next sentence makes sense:

Bitcoin blockchain functionality is

limited to ???

while Ethereum blockchain can also

???

\* tracking a digital currency

\* run and store applications

\* solve world hunger

\* ASICs for solving PoW

\* use GPUs

**Explanation:**

Other than being distributed and having a PoW algorithm initially, Ethereum does pretty much everything else \*differently\* than Bitcoin.

### Account

Ethereum is using an \*\*account model\*\* aka balance model. You are probably familiar with the centralized version of this since it's employed by banks. The problem with a decentralized account model is its vulnerability to \*replay attacks\*: if you broadcast a signed transaction in which you pay Bob 0.5 ETH, Bob could re-broadcast the exact same transaction (being signed and everything) multiple times to withdraw all of your funds. To prevent these attacks, Ethereum accounts attach to transactions their \*sequence number\* (nonce). Once a nonce has been redemeed once, every other transaction that has a nonce less or equal that the last value redeemed is dropped.

### Transactions

Ethereum accepts more than transfer transactions. Given its capability of running code, Ethereum has special transactions used to interact with smart contracts: creation, calling, state querying.

Regardless of transaction type, 4 variables are required:

\* Gas price: higher gas price means a higher transaction fee and makes nodes more likely of including the transaction in the next block

\* Gas limit: depending on the resources used by the smart contract, the amount of gas used is determined; if the gas limit is excedeed (transaction costs more than you wanted to pay), the transaction is dropped but the transaction fee is not returned

\* Nonce: transaction sequence number

\* From: transaction signatory

### Solidity

\*\*Solidity\*\* is Ethereum's most popular programming language. All you have to do is throw some Solidity code (similar to JavaScript in syntax) into a `.sol` file and deploy the smart contract on the blockchain. There are a few tools you can use, but probably the easiest to set-up is `Remix`[1] compiler.

### Token Standards

Another feature of Ethereum is \*\*token standards\*\*: an interface for cryptocurrencies recognized by the Ethereum blockchain. The most notable token standard is ERC20. But why would you use the standard instead of releasing a blockchain of your own? Well, having to find people to mine/stake your blockchain and low adoption are the main reasons new blockchains fail.

Moreover, Ethereum became a decentralized crowfunding platform: you could launch your own project right away and ask people that owned ether to sponsor it in exchange for your tokens ("stock").

**Second question:**

Identify the programming language used to write smart contracts:

???

\* Solidity

\* Java

\* C++

\* Ethereum

**Insight 3:** What are Smart Contracts?

**First Question:**

What do you have to pay for to publish a smart contract on the blockchain?

???

\* Memory, disk and computing power

\* A fixed fee every month

\* Disk storage

\* Nothing

**Explanation:**

\*\*Smart contracts\*\* are Ethereum's code files. They would be useless without Ethereum Virtual Machine. EVM is a runtime environment for executing compiled smart contracts: EVM does not run Solidity directly, but rather runs the \*bytecode\* obtained by compiling `.sol` files[1].

Here's an example of a Solidity smart contract. We'll cover in-depth what each keyword means later on:

```javascript

pragma solidity ^0.4.23;

contract MyAwesomeToken {

mapping (address => uint) public

ownerToBalance;

function setBalance(address owner,

uint newBalance) public {

require(msg.sender == owner);

ownerToBalance[owner] = newBalance;

}

function getBalance(address owner) public

view returns (uint) {

return ownerToBalance[owner];

}

}

```

Once deployed, the contract would receive its own unique address. The address works similarly to any other public key on the network. Next, one can direct a transaction calling `setBalance` function to the contract's address. In our case, if they were using their \*own\* address (`require` at line 8) and a random balance as parameters, the contract would create the mapping (line 9). As you can see, the language has a mix of both JavaScript (as the main influencer) and Java keywords, plus some of its own.

Smart contracts are different than your usual code files. They are \*\*immutable\*\*. Once up there, users can interact with them forever. Without proper tests and checks, nasty problems can occur, but more on this later.

**Second question:**

What is the difference between a smart contract and code filles written in other languages?

???

\* Smart contracts are immutable once published

\* Smart contracts are more secure

\* Other code files compile faster

\* There is no difference other than syntax

### Solidity

**Description:** The popular smart contracts language

**Insight 1:** What is Solidity?

**First Question:**

What does solidity run on?

???

\* EVM

\* Blockchain

\* Any computer

\* Ethereum blockchain

**Explanation:**

\*\*Solidity\*\* is a programming language that can be compiled into bytecode and run by the EVM. But the relationship between Solidity and EVM is not one-to-one. Solidity is a contract-oriented language that can work on multiple platforms. Moreover, it is part of the set of languages in which Ethereum smart contracts can be developed.

It is a relatively new language, which was officially released early-mid 2015. It was created by some of the developers on the Ethereum core team.

The main features Solidity had over other Ethereum supported languages were static typing, mappings, structs, (multiple) contract inheritance, and type-safe functions.

While it is not particularly difficult to start writing Solidity code, the real milestone is actually understanding and following security best practices. Remember that once the smart contract is up there, there is no way to change it, although there is a kill switch. Over the years, multiple exploits in smart contracts allowed bad things to happend, one heist amounting over $50 millions worth of ether. Moreover, developers, to prove their honesty and prevent some exploits, are required to pay a gas fee (for things like storage) when publishing their contracts.

**Second question:**

Why does one pay gas to publish a smart contract on the blockchain?

???

\* To prove their honesty

\* To support the Ethereum foundation

\* To tip the nodes that run the code

\* To speed up the verification process

**Insight 2:** Security Concerns

**First Question:**

What is the name of an entity that links the outside world to the blockchain?

???

\* Oracle

\* API gate

\* Informatix

\* Developer

**Explanation:**

Unfixable bugs are the worst bugs. Normally, you would be able to patch an application if something came up: on the blockchain you can not. As such, security is a big concern when talking about deploying a smart contract.

Luckily, people working in this field recognize the need to over-share best practices: \*Zeppelin\* released an open-source set of secured, audited and tested contracts from which your own can inherit[1]. While there is probably no need to dive into them if you are starting with Solidity on Remix, it would be useful remembering the repository.

Smart contracts have their own address, from which they can be accessed. Normally, everyone could access them. However, smart contracts have a way to track the message sender (`msg.sender`) and ignore anyone other than the owner for certain administrative functions, like contract disabling.

Moreover, in order for consensus to be reached about blocks, the data in them has to be deterministic (no `random`). Otherwise, every node would obtain a different identifier for the smart contract. Consequently, smart contract by themselves are locked to only accessing data from the blockchain itself and transactions in previous blocks.

But this does not mean smart contracts can not eventually access say the temperature in London on the 3rd of July 2018. Ethereum has this concept of \*oracles\*: smart contracts maintained by trusted third-party APIs who return, given the same parameters, the same data. For this to happen, a request is made to the Oracle, the Oracle calls a `storeData` function on its smart contract with a question key and the result and stores the mapping on the blockchain.

**Second question:**

Identify the entity responsible for the security of a smart contract:

???

\* The developer

\* The blockchain maintainer

\* The user

\* No one, smart contracts are inherently secure

**Insight 3:** Ether Heists Cause by Bugs

**First Question:**

What was the outcome of the theDAO hack with regard to Ethereum?

???

\* Funds were returned through a hard-forked

\* The address was black-listed through a soft-forked

\* No funds were returned, but the address was black-listed

\* ICOs are now illegal

**Explanation:**

Ethereum found itself in a rough spot after theDAO heist: the initial soft-fork that froze the thief's wallet was followed by a hard-fork that modified the smart contract and allowed investors to withdraw their ETH.

DAO means Decentralized Autonomous Organization. DAO's goal is to digitize the decision making process of an organization, effectively creating a completely decentralized environment. In other words, a DAO is a smart contract that could work as an investment fund: people store ether and each month they vote for a project to invest in. The returns are then split by the weight of each investment.

That is how theDAO was supposed to function: a fully decentralized crowdfunding platform. The project was such a success that it managed to gather around 15% of all ether available at that date. ($150 mil at an average market cap of $940 mil in May 2016). But everything went down when an exploit that would withdraw the someone's invested ether before updating the total balance was discovered.

In total, $70 millions worth of ether were stolen. Faced with a tough situation, Ethereum Foundation decided to hard-fork such that all funds could be returned. But a part of the minority strongly believed that the blockchain should be immutable. As such, they decided not to upgrade and continue building on the old chain under the name of \*Ethereum Classic\*.

**Second question:**

If coins are found to be stolen, identify what should be done to revert the theft transactions

???

\* hard fork

\* soft fork

\* ban the thief's address

\* transactions cannot be reverted

### Ethereum II

**Description:** All about Ethereum token standards

**Insight 1:** Token Standards

**First Question:**

What is an Ethereum token standard?

???

\* A smart contract interface

\* A subdenomination of ether

\* An alternative to ether

\* A cryptocurrency

**Explanation:**

An Ethereum \*\*token\*\* is a smart contract that has a mapping between user addresses and balances. In other words, it is a contract that keeps track of how much of something each address has. A \*\*token standard\*\* is a smart contract that contains some standard functions that all other contracts share, like `transfer` or `balanceOf`.

Token standards are therefore Ethereum's way of seizing and simplifying the creation of new cryptocurrencies. There are 2 main standards:

\* ERC20

\* ERC721

By following either one of them, programmers can ditch releasing a blockchain platform of their own and start using an already secured and successful platform. Moreover, the interface is common between all tokens such that interactions with external platforms are also eased. Exchanges, for example, are platforms that benefit a lot from the shared interface.

**Second question:**

Which of the following is not an Ethereum token standard?

???

\* ERC10

\* ERC20

\* ERC721

**Insight 2:** ERC20 Tokens

**First Question:**

Does ERC20 token standard support decimals?

???

\* True

\* False

**Explanation:**

\*\*ERC20\*\* is the standard used to represent currencies. It allows for basic account functionalities like transfer, get address balance, or get total supply.

The basic ERC20 interface (that describes no implementation) looks like this[1]:

```bash

contract ERC20BasicInterface {

function totalSupply() public

view returns (uint256);

function balanceOf(address who) public

view returns (uint256);

function transfer(address to,

uint256 value) public returns (bool);

event Transfer(address indexed from,

address indexed to, uint256 value);

}

```

And it was extended to include[2]:

```bash

contract ERC20Interface

is ERC20BasicInterface {

function allowance(address owner,

address spender)

public view returns (uint256);

function transferFrom(address from,

address to, uint256 value)

public returns (bool);

function approve(address spender,

uint256 value)

public returns (bool);

event Approval(address indexed owner,

address indexed spender, uint256 value);

}

```

The basic functions are straight-forward. The `Transfer` event is triggered when tokens are transferred and can be listened to. An event has to be explicitly called in the function.

The second contract interface extends the basic one with functionality to manage funds on behalf of users. One use case is having a second smart contract contain the logic that deals with token transfers. Like `Transfer`, `Approval` should be triggered at the end of `approve` function.

Some information needed to define a ERC20 token is:

\* address of smart contract

\* total amount of tokens available

\* name

\* decimals[3] (from 0 to 18)

**Second question:**

Does the ERC20 standard define the actual implementation of the `balanceOf` function?

???

\* False

\* True

**Insight 3:** ERC721 Tokens

**First Question:**

Does ERC721 token standard support decimals?

???

\* False

\* True

**Explanation:**

\*\*ERC721\*\* tokens, on the other hand, are better suited for things you can not have 0.5 of, say cryptokitties[1]. ERC721 tokens are assumed to be unique and indivisible. You can only move them round in whole units, each one having a unique ID. They are known as \*non-fungible tokens\*.

To better understand the difference, you can imagine that, while ERC20 contains an `address => balance` mapping, ERC721 contains an `uniqueObjectId => address` mapping. The ID must be generated deterministically and must also be a positive integer (most of the times - strings also work).

The list of methods looks similar to that of ERC20 standard:

```bash

contract ERC721 {

event Transfer(address indexed \_from,

address indexed \_to, uint256 \_tokenId);

event Approval(address indexed \_owner,

address indexed \_approved,

uint256 \_tokenId);

function balanceOf(address \_owner)

public view returns (uint256 \_balance);

function ownerOf(uint256 \_tokenId)

public view returns (address \_owner);

function transfer(address \_to,

uint256 \_tokenId) public;

function approve(address \_to,

uint256 \_tokenId) public;

function takeOwnership(uint256 \_tokenId)

public;

}

```

Note: The ERC721 standard is currently a draft, and there is no officially agreed-upon implementation yet. However, you can find an example implementation in OpenZeppelin repository.

**Second question:**

What are usually the types of the main mapping of a ERC721 token?

??? => ???

\* unique ID

\* address (uint256)

\* int256

\* balance (uint256)

\* string

**Insight 4:** Initial Coin Offerings

**First Question:**

What is the moment in which the token acquires value?

???

\* When it appears on an exchange

\* When it is bought

\* When the company launches its product

\* The token will always be worthless

**Explanation:**

Now that you know of token standards, it is easier to explain how ICOs work. \*\*ICO\*\*s, or Initial Coin Offerings, are crowdfunding campaigns that are recorded on the Ethereum blockchain.

Companies that want to raise capital create such a campaign in exchange for which they offer tokens, or coins. These coins represent shares in their company, much like it works in the stock market.

Bear in mind that much like the stock market, the shares are worthless unless the company went public, i.e. their token can be found on an exchange.

Given the hype and fear-of-missing-out (FOMO) born over the wild growth period that was November-December 2017, a lot of scam ICOs were launched. Due diligence and proper research are required if you're looking to invest in ICOs.

**Second question:**

ICO tokens are similar to

???

\* stocks

\* bonds

\* options

### Ethereum on the Web

**Description:** todo

**Insight 1:** MetaMask

**First Question:**

What currencies can Metamask keep track of?

???

\* Ethereum and ERC20

\* Ethereum and Bitcoin

\* Ethereum, ERC20 and Bitcoin

\* Ethereum

**Explanation:**

Like for bitcoins, you need a wallet to be able to hold any ether. The Ethereum client works well as an online wallet if you are looking to also run a node. But, as any client, it requires you to sync to the latest block.

\*\*MetaMask\*\* is a browser extension that allows you to interact with the Ethereum network without having to download the entire chain locally. It connects to an already running client that makes initiating and receiving transactions easy. Moreover, it supports any ERC20 token (standards for the win!) alongside ETH.

It is a simple interface that stores any generated keys offline, on your computer. With MetaMask, you can access websites that interact with the Ethereum blockchain in any browser that supports the extension.

**Second question:**

Where does Metamask store your wallet?

???

\* Offline, on your computer

\* Online, on their servers

\* Online, on the cloud

\* Nowhere, that's not what metamask does

**Insight 2:** MyEtherWallet

**First Question:**

Can Metamask be integrated with MyEtherWallet?

???

\* Yes

\* No

\* Yes, but in specific cases

**Explanation:**

\*\*MyEtherWallet\*\*[1] is a free, open-source, client-side interface. It allows for direct interactions with the blockchain without transferring your keys and funds anywhere. A new account on MyEtherWallet means generating a private key/public key pair, whose handling happen inside the browser.

But keep something in mind (as it is mentioned all over on the website): you are responsible for keeping track of whatever keys you're generating as MyEtherWallet is just an \*interface\* and cannot be used to recover wallets. You being in charge of your security and funds is part of blockchains' raison d'être.

## Note that if you are already using MetaMask, you can easily connect it to MyEtherWallet.

## Footnotes

[1: MyEtherWallet]

<https://www.myetherwallet.com/>

**Second question:**

Does MyEtherWallet communicate with the Ethereum network?

???

\* Yes

\* No

**Insight 3:** Web3.js

**First Question:**

Is the JavaScript `web3` the only available version of `web3`?

???

\* No

\* Yes

**Explanation:**

\*\*Web3.js\*\* is the de-facto Ethereum API for JavaScript. It allows interactions between a regular web page and the distributed apps' smart contracts (DApps) running on the Ethereum network. The only constraint is that the server must run an Ethereum node for this to be possible.

MyEtherWallet is an example of a website using it. If you open the link in your favorite browser, open up the JavaScript console and type-in `this.web3`, the `web3` object should be returned.

`Web3` is also available for languages like Python, Java, Scala or PHP.

**Second question:**

What does `web3` need in order for it to work?

???

\* The server to run an Ethereum client

\* A global web3 JavaScript object

\* For the website to be written in JavaScript