**Block Chain Development.**

**Cryptography:**

**SHA 256:**

**Web3:**

**Truffle:**

**Infura:**

**EDCSA Algorithm:**

**RSA Algorithm:**

**Private Key:**

**Public Key:**

**Address Key:**

**Block chain:**

**Ethereum:**

## What is Web 3.0?

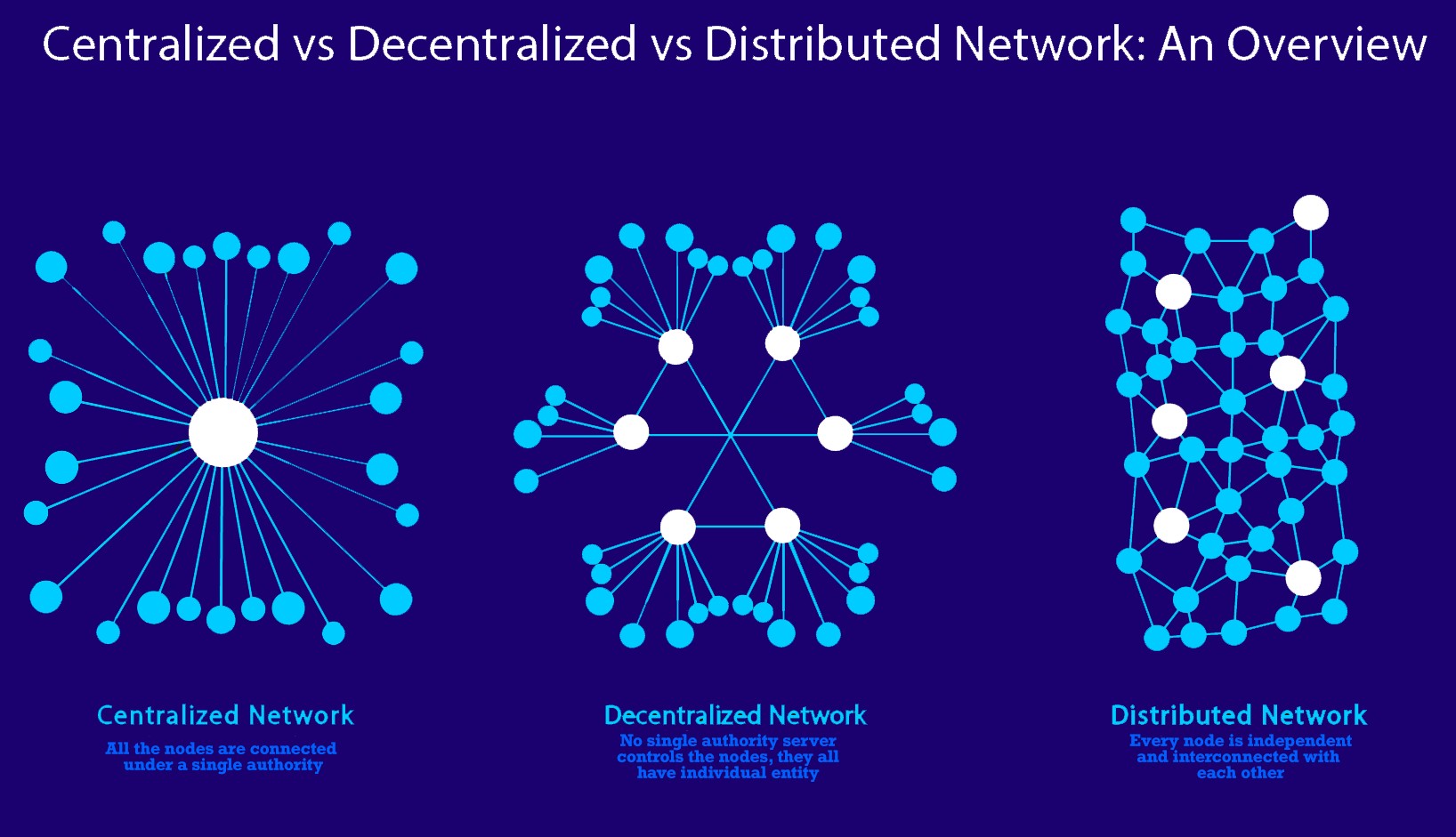
There are a few fundamental differences between web2 and web3, but decentralization is at its core.

Web3 enhances the internet as we know it today with a few other added characteristics. Web3 is:

* Verifiable
* Trustless
* Self-governing
* Permission less
* Distributed and robust
* Stateful
* Native built-in payments

In web3, developers don't usually build and deploy applications that run on a single server or that store their data in a single database (usually hosted on and managed by a single cloud provider).

Instead, web3 applications either run on block chains, decentralized networks of many peer to peer nodes (servers), or a combination of the two that forms a crypto economic protocol.



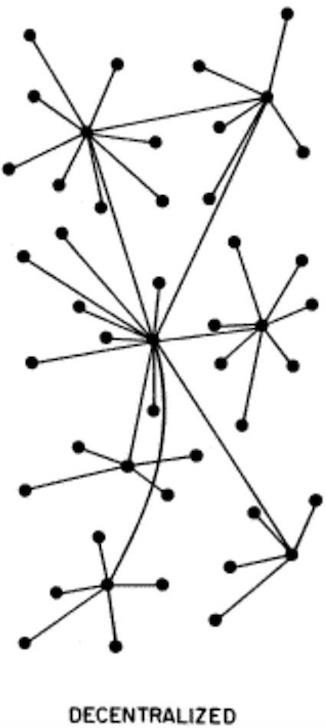
Centralized vs Decentralized vs Distribute

### https://blockchainengineer.com/wp-content/uploads/2019/01/c.pngCentralized Applications

Most of the Internet applications we use every day are centralized, they are owned by a particular company or person that provision and maintain the source code to execute on a computer, server or maybe even a cluster.

Centralized applications are the majority of applications that engineers are used to building and users are used to using.

Data resides on a centrally owned database controlled by a company. You ultimately have to trust that this company is doing things correctly and in your interest.



### Decentralized Applications

Distributed means computation is spread across multiple nodes instead of just one. Decentralized means no node is instructing any other node as to what to do.

The code runs on a peer-to-peer network of nodes and no single node has control over the dApp. Depending on the functionality of the dApp, different data structures can be used to store the application data. Bitcoin uses a blockchain decentralized ledger of transactions.

### https://blockchainengineer.com/wp-content/uploads/2019/01/di.pngDistributed Applications

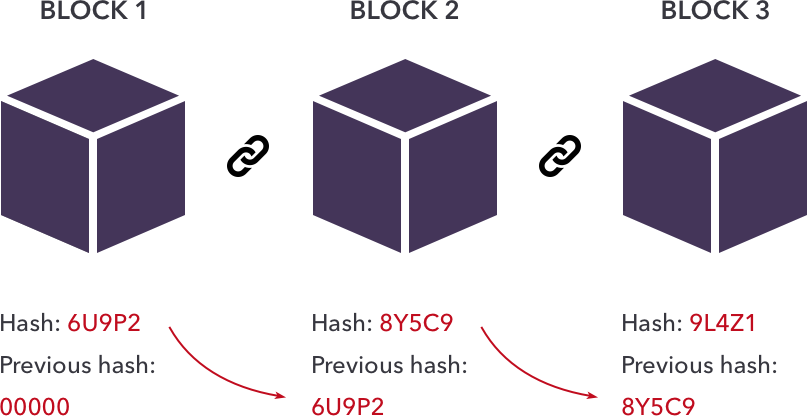
Applications in which computation is distributed across components, communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal.

Some distributed applications examples are:

* + CDN
  + AWS
  + Cloud Instances
  + Google, Facebook

**Block chain:**

Let’s try to understand [blockchain](https://www.investopedia.com/terms/b/blockchain.asp), in the simplest possible terms, by breaking down its etymology. Blockchain is a kind of database that stores information in blocks and then connect those blocks together with a chain. You can think of it as a link list where consecutive blocks are connected with their unique hashes.



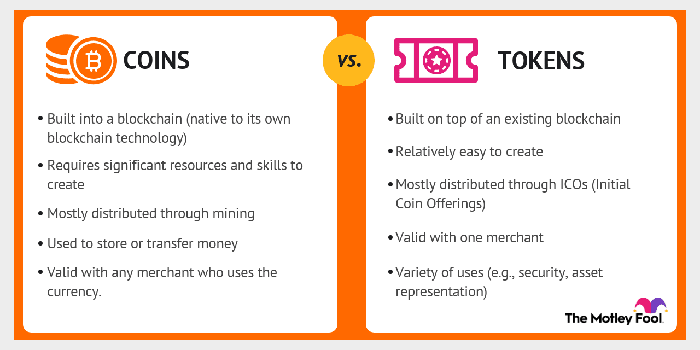
Cryptocurrency uses the decentralized blockchain, which means the database is spread among a number of devices so that no single party owns it but all of them hold it. All transactions are immutable, meaning they can’t be changed once recorded. We’ll learn about transactions, smart contracts, coins, tokens and lot more in this article.

**Solidity:**

Solidity is an object-oriented programming language for writing smart contracts. It is used for implementing smart contracts on various blockchain platforms, most notably, Ethereum.

**Smart Contract:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| The term “smart contract” was first used by Nick Szabo in 1997 , long before Bitcoin, He is a computer scientist, law scholar and cryptographer , he wanted to use a distributed ledger to store contracts | | | |
|  | | | Now smart contract are just like contract in the real world. The only difference is that they are completely digital. In fact a smart contract is actually a tiny computer program that is stored inside a block chain |
|  | | | Let’s take a look at an example to understand how smart contract work. You probably are familiar with Kickstarter the larger fundraising platform Product teams can go to Kickstarter, create a project, set a funding goal and start collecting. |
|  | | | Kickstarter is essentially a third party that sits between product teams and supporters. This means that both of them need to trust Kickstarter to handle their money correctly. If the project get successfully funded, the project team expects Kickstarter to give them the money, On the other hand, supporters want their money to go to the Project if it was funded or to get a refund when it hasn’t reached its goals. Both the product team and its supports have to trust Kickstarter. But with smart contracts, we can build a similar system that doesn’t require a third-party like Kickstarter |
| ***Supporters transfer money to Smart contract***    ***Smart contract transfer money to creator of the project***      ***If Failed money automatically goes back*** | | | So let’s create a smart contract for this! We can program the smart contract so that it holds all the received funds until a certain goal is reached. The supporters of a project can now transfer their money to the smart contract. If the project gets fully funded, the contract automatically passes the money to the creator, and if the project fails to meet the goal, the money automatically goes back to the supporters. And because smart contracts are stored on a block chain, everything is completely distributed with this technique, no one is in control of the money |
| Why should we trust a smart contract? Well because smart contracts are stored on a block chain, they inherit some interesting properties, they are Immutable and they are Distributed. | | | |
|  | | Being immutable means that once a smart contract is created, it can never be changed again. So no one can go behind your back and tamper with the code of your contract | |
|  | | Being distributed means that the output of your contract is validated by everyone on the network, so a single person cannot force the contract to release the funds because other people on the network will spot this attempt and mark it as invalid, Tampering with smart contracts becomes almost impossible. | |
| Smart contract can be applied to many different things, not just on crowdfunding. Banks could use it to issue loans or to offer automatic payments. Insurance companies could use it to process certain claims. Postal companies could use it for payment on delivery, and so on and so on…. | | | |
|  | So, now you might wonder where and how you can use smart contracts. Right now there are a handful of block chains who support smart contracts, but the biggest is one is Ethereum, it was specifically created and designed to support smart contracts. They can be programmed in a special programming language called solidity, This language was specifically created for Ethereum and uses a syntax that resembles JavaScript , It’s also worth noting that bitcoin also has support for smart contract although it’s a lot more limited compared to ethereum. So now you now what smart contract are and what problems they solve | | |

**Difference between Coins and Tokens?** 

## What is Cryptocurrency wallet?

A cryptocurrency wallet is a device, physical medium, program or a service which stores the public and/or private keys for cryptocurrency transactions. In addition to this basic function of storing the keys, a cryptocurrency wallet more often also offers the functionality of encrypting and/or signing information.

## What does a wallet actually store?

A wallet's main function really is about storing and handling crypto assets. Even though we usually speak of digital wallets in analogy to physical wallets that we carry in our pockets, crypto wallets don't actually store any crypto assets. After all, a bitcoin does not really exist. It is merely a ledger entry on the Bitcoin blockchain.

What is actually stored in a crypto wallet are the [seed phrase and its private keys](https://www.mtpelerin.com/blog/public-key-private-key-seed-phrases) that correspond to the very crypto assets at hand. If you lose access to your digital wallet with no way of retrieving it, your crypto still exists on the blockchain but cannot be used anymore as the corresponding private keys are lost.

## Third-party involvement or online exposure?

When it comes to choosing a crypto wallet, you must basically ask yourself the following two questions and choose accordingly:

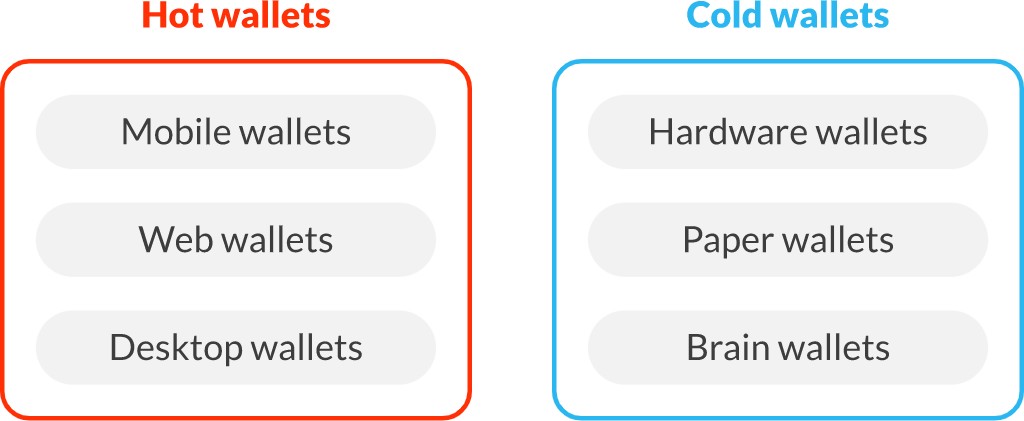
* + Will I be using an online "hot" wallet or an offline "cold" wallet?
  + Will I be using a third-party custodial wallet or a non-third-party non-custodial wallet?

For more explain what these two questions mean visit the below link:

* <https://www.mtpelerin.com/blog/the-different-types-of-crypto-wallets>

## The different types of crypto wallets

Today, there exist numerous different wallet types. There are web wallets, mobile or desktop wallets as well as hardware and paper wallets. Let's dive into each category!



## HOT Wallets

## Mobile wallets

Mobile wallets are apps that can be installed on a smartphone and are one of the easiest and most convenient ways to setup a non-custodial crypto wallet. In the Apple App Store or on Google Play, many different mobile wallets can be found and downloaded on your smartphone. Most of them will get you tarted by creating a new wallet and will walk you through the backup of your seed phrase.

A key convenience of mobile wallets is the possibility to scan QR codes, which allow to enter a crypto address without room for copying errors.

* Examples:
  + [Bridge Wallet](https://www.mtpelerin.com/bridge-wallet)
  + [Edge](https://edge.app/)
  + [Trust Wallet](https://trustwallet.com/)
* Pros: Can be used anywhere, convenient, fast, more secure than web and desktop wallets. Phone features such as QR code scan.
* Cons: Exposed to mobile viruses and malware. A phone can also be easily lost or stolen.

## Web wallets

Web wallets are online wallets that are accessible through a web browser. They let you control your crypto assets through an online gateway, the web browser, without the need to install any extra software. Some of them even exist as browser plugins, like MetaMask.

* Examples:
  + [MetaMask](https://metamask.io/)
  + [MyEtherWallet](https://www.myetherwallet.com/)
  + [MyCrypto](https://mycrypto.com/)
* Pros: Convenient, fast and broadly available. Let you access your crypto and make transactions in seconds.
* Cons: Because web wallets are online, they are at a greater risk of being exposed to malware and hackers.

## Desktop wallets

Desktop wallets are similar to mobile wallets, they are software that you install on your computer and are usually non-custodial. They are a convenient way to store your crypto assets using your computer but be aware: the risks of hacks are also present with desktop wallets. Securing your computer is a must and a backup of the private keys should be made in any case. It is not recommended to store large amounts of coins and tokens on a desktop wallet, except in combination with a hardware wallet.

* Examples:
  + [Ledger Live](https://www.ledger.com/ledger-live)
  + [Exodus](https://www.exodus.com/)
  + [Atomic](https://atomicwallet.io/)
* Pros: Convenient, easy to use
* Cons: Vulnerable to the many virus and malwares targetting desktop computers

## Cold Wallets

## Hardware wallets



Most of the time, they come in the form of UBS like devices that are specifically designed to store private keys. Hardware wallets are more secure than mobile wallets or desktop wallets, mainly because they are incapable of connecting to the Internet directly.

The way they usually work is through a secure enclave on their hardware (hence their name), where transaction signing takes place. When a transaction is signed, it is done in a secure, hermetic offline environment using the relevant private key. Only upon successful validation by the user can a transaction be sent to the network. Validation or confirmation always needs to happen through the user pressing a physical button or touching a touchscreen. There is no way a hacker that is not present can initiate a transaction from a hardware wallet. When sending a transaction as a user though, you always want to make sure the recipient's address on the hardware wallet is the correct one. This way you are sure that you are not being shown a wrong address and that the funds will be sent to the right place.

Once the signing is done, only the signed transaction leaves the hardware wallet and is broadcasted onto the blockchain. This setup significantly reduces the attack surface for hackers, since the device cannot be manipulated remotely. As other wallets, a hardware wallet also has to be backed up by a seed phrase. In case of loss or theft, assets stored on the hardware wallet can be recovered with the seed phrase on a new device. As always, [the](https://www.mtpelerin.com/blog/storing-crypto-assets) [safe storage of the seed phrase](https://www.mtpelerin.com/blog/storing-crypto-assets) is of uttermost importance.

* Examples:
  + [Ledger](https://shop.ledger.com/?r=27a88c3f663e)
  + [Trezor](https://shop.trezor.io/?offer_id=10&aff_id=5415)
  + [Bitbox](https://shiftcrypto.ch/)
* Pros: Hardware wallets are one of the safest crypto wallet options. They are perfect for long-term holders storing a great amount of crypto assets.
* Cons: Rather expensive. Although they have great interfaces, the whole setup is not very intuitive for newcomers.

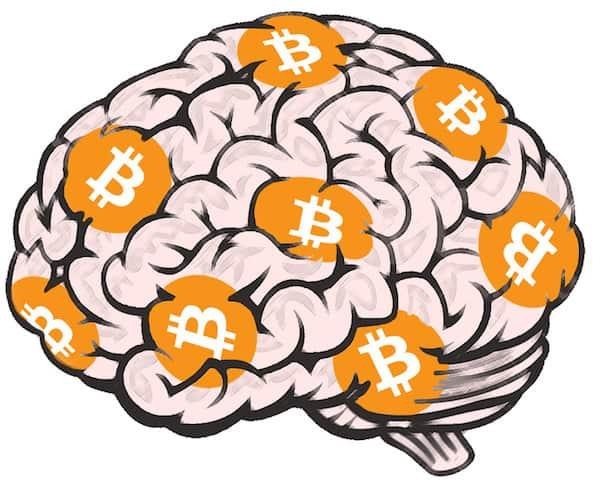
## Paper wallets

Another cold storage wallet solution are paper wallets, also known as a physical wallet. Private keys as well as public addresses are printed or written down on a sheet of paper. Paper wallets can be created with an open source paper wallet generator. Users are recommended to be offline when generating the keys and to delete their browser history once the keys are generated. For maximum security, the use of a brand-new computer is recommended to avoid malware interference. Also, when a paper wallet created by a third- party is used, there exists the risk that the private key on the paper wallets is also kept by the issuer.

Compared to a hardware wallet, paper wallets can be seen as quite a hassle. It's safe to say that hardware wallets have replaced paper wallets, at least when it comes to funds that need to be stored securely but need to be used regularly.

* Examples: [Paper Wallets from Bitcoin Suisse](https://www.bitcoinsuisse.com/crypto-certificates)
* Pros: Paper wallets can be created by oneself by following certain instructions. If done right, they are very secure. They can also be a nice gift to introduce someone to crypto.
* Cons: These types of wallets are rather inconvenient as you will need to make several manipulations in order to use and transfer funds stored on them. They are also considered unsafe as they can be easily lost, destroyed or stolen.

## Brain wallets

Because at their core crypto assets represent information, their respective private keys can also be "stored" in your own brain by simply memorizing the seed phrase. It's only 12 words to remember after all! If the seed phrase is not written down everywhere else, the relevant crypto assets can be thought of only existing in the owner's head.

Creating a brain wallet works by using an offline seed phrase generating software or the seed phrase can also be created using several unrigged dices.

* Pros: This is a very low-tech solution but if conducted and set up correctly, it is a great seed phrase backup option as it does not exist anywhere else than in the owner's brain.
* Cons: The brain wallet's biggest strength can also become its biggest weakness as human memory is fallible. If you forget the memorized seed phrase, which may happen because of an accidental brain damage for example, crypto assets stored that way would be irreversibly lost.

**MetaMask (ETH Wallet-Chrome Extension):**

****

|  |  |  |  |
| --- | --- | --- | --- |
| **BLOCKCHAIN** | **Written In** | **Language support** | **Consensus** |
| ARK | JavaScript | JavaScript, Go, Python, C#, TypeScript, Kotlin, Ruby, Swift, PHP | DPoS |
| CORDA | Kotlin | Java, Kotlin, |  |
| ETHEREUM | Go, C++, Rust | Solidity | PoW/PoS |
| EOS | C++ | WebAssembly, C, C++ | DPoS |
| HYPERLEDGER FABRIC | Go, Java, JavaScript, Python | Go, Java, Kotlin |  |
| LISK | JavaScript, Node.js | JavaScript | DPoS |
| NEO | C# | C#, Java, Kotlin, Python | PoS |
| QTUM | C++, Python, TypeScript | C++, Python, Rust, Go, Lua | PoS |
| STRATIS | C++, C# | C# | PoS |
| WAVES | Scala | Scala | DPoS |