



Blockchain

UniTn

Federico Brancasi
October 3, 2024

1	WHAT IS A BLOCKCHAIN?	4
1.1	Definition	4
1.2	Decentralized Ledgers	4
1.3	Distributed Ledgers	5
1.4	The Role of Trust and the Evolution of the Blockchain	5
2	BLOCKCHAIN CORE CONCEPTS	7
2.1	Decentralization/Distribution: What Sets Them Apart?	7
2.2	Methods of Decentralization	7
2.3	Unveiling the Core of Blockchain	8
2.4	The Anatomy of a Block	9
3	CONSENSUS MECHANISMS	10
3.1	High Level Considerations	10
3.2	DLT Categories	10
3.3	Byzantine Generals Problems	11
3.4	Security Mechanisms	12
3.5	Type of Consensus Mechanisms	12
3.6	Blockchain Trilemma	13
3.7	Distributed Consensus	14
3.8	Distributed Consensus Protocols	15
3.9	Proof of Work (PoW)	15
3.10	Proof of Stake (PoS)	16
4	SMART CONTRACTS	17
4.1	Smart Contract Idea and Evolution	17
4.2	Ricardian Contracts	18
4.3	Bitcoin or Ethereum?  	18
4.4	Smart Contract Compliance Issues	18
4.5	DApps	19
4.6	Oracles	21
4.7	Blockchain and Smart Contracts Summary	21

5	TRANSACTIONS AND WALLET	23
5.1	Transaction Types	23
5.2	Accounts in the Blockchain Context	23
5.3	Wallet	24

1. WHAT IS A BLOCKCHAIN?

1.1. DEFINITION

Blockchain is an innovative technology introduced by **Satoshi Nakamoto** in 2008 through his famous paper "**Bitcoin: A Peer-to-Peer Electronic Cash System**" [3]. It is a revolutionary cryptography-based electronic payment system designed to overcome the imitations of traditional systems and promote decentralization.

Defined as "*a Blockchain*," each block contains a cryptographic hash of the previous block, a timestamp and transaction data. This blockchain structure makes each block inextricably linked to the others, creating an immutable and transparent digital ledger.

Blockchain's basic purpose is to act as a decentralized ledger, which means that transactions are recorded permanently and immutably without the need for a central authority. It also enables consensus to be maintained in a trust-free environment, allowing unknown and potentially untrusted participants to collaborate securely.

Trade Ledger is not that far back in the history of accounting evolution. Interestingly, the double-entry accounting system described in Luca Pacioli's book has origins dating back to the possible influence of the Muslim accounting practice implemented by the Venetians [5]. So yes, ledgers are not new at all!

Now that we understand the essence of blockchain technology and its historical roots, let us delve into the concepts of **Distributed Ledgers** and **Decentralized Ledgers**. Understanding these concepts is critical as they form the backbone of various blockchain implementations.

1.2. DECENTRALIZED LEDGERS

A decentralized ledger is essentially a registry that is managed and updated by multiple parties without the need for a central authority. Imagine this ledger as a document shared among several participants, but instead of being controlled by a single entity, it is distributed over a peer-to-peer (P2P) network. In this network, each node has a copy of the ledger and can verify its transactions.

Strengths:

- **Enhanced Security:** Because there is no central point of control, decentralized ledgers are less vulnerable to cyber attacks.
- **Transparency and Accountability:** All parties involved can see and verify transactions, promoting transparency and accountability.
- **Trustless Environments:** No need to rely on intermediaries to verify transactions, creating a trustless environment.

Drawbacks:

- **Processing Speed:** Due to the consensus required by multiple parties, transactions can be slower than in centralized systems.
- **Complexity of Implementation:** Multi-party coordination and agreement are required to effectively implement a decentralized ledger.
- **High Costs:** Maintaining the network requires significant resources in terms of computing power and energy.

1.3. DISTRIBUTED LEDGERS

A distributed ledger is a ledger that is managed and updated by multiple parties and stored in multiple locations. Unlike a decentralized ledger, which is distributed over a peer-to-peer network, a distributed ledger operates over a distributed network, where each node in the network owns a copy of the ledger and can validate transactions.

Strengths:

- **Enhanced Processing Speed:** Distributed ledgers can be faster than decentralized ledgers because they can process transactions in parallel.
- **Improved Scalability:** Being more scalable than decentralized ledgers, distributed ledgers can be expanded to accommodate more users and transactions.
- **Increased Flexibility:** Distributed ledgers can be more flexible than decentralized ledgers because they can be customized to meet specific needs.

Drawbacks:

- **Potentially Lower Security:** Unlike decentralized ledgers, distributed ledgers may be less secure because there is a central control point that could be targeted by hackers.
- **Reduced Transparency and Accountability:** Some parts of the network may have more control over the ledger than others, reducing overall transparency and accountability.
- **Complexity and Management:** Distributed ledgers can be more complex and difficult to manage than decentralized ledgers because they require coordination and agreement among multiple parties and multiple copies of the ledger.

1.4. THE ROLE OF TRUST AND THE EVOLUTION OF THE BLOCKCHAIN

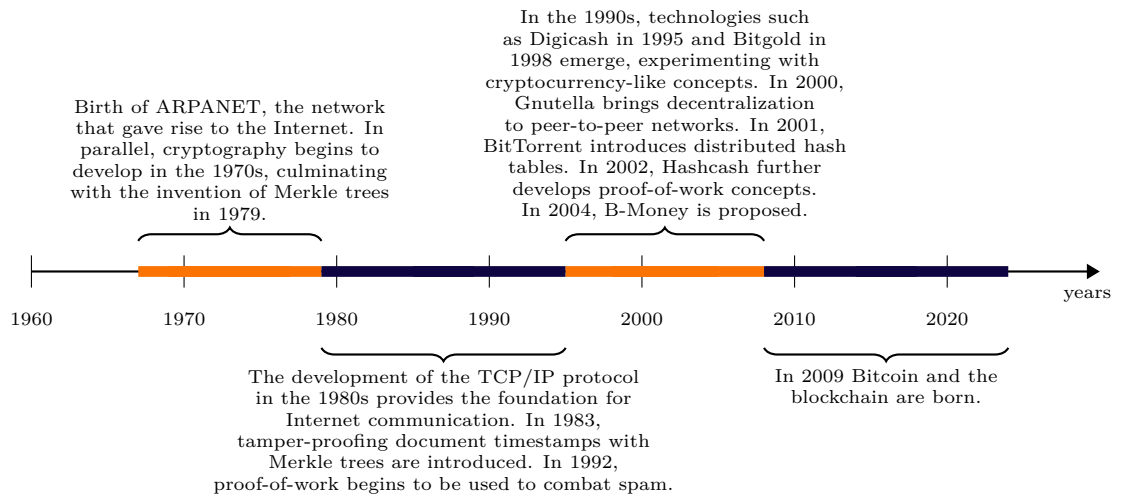
Trust plays a central role in understanding blockchain, going beyond mere technology and raising crucial philosophical and political questions concerning governance and the distribution of power.

Throughout history, philosophers and political theorists have examined different forms of social organization and the role of trust in institutions. From Plato to Hayek, perspectives vary widely, influencing our conceptions of power and trust.

The 2008 financial crisis acted as a catalyst for the adoption and development of blockchain and Bitcoin. The collapse of banks and financial institutions profoundly undermined trust in central institutions, pushing for greater decentralization and financial autonomy.

Moreover, blockchain does not simply represent a new technological invention, but rather an amalgamation of existing technologies that have been developed and refined over time.

It is through the evolution of these technologies that blockchain has taken shape and begun to play a significant role in the digital landscape. To better understand this process, let us examine a rough timeline of key inventions:



Blockchain Key Inventions Timeline

2. BLOCKCHAIN CORE CONCEPTS

2.1. DECENTRALIZATION/DISTRIBUTION: WHAT SETS THEM APART?

Let us now restate the difference of the concepts of decentralization and distribution in the context of blockchain.

Decentralization refers to the distribution of control and decision-making power from a central authority. In blockchain, this means that no single entity has control over the network. Instead, the network is managed by a group of nodes that work together to maintain the integrity of the network.

Distribution, on the other hand, refers to the physical dispersion of data or computing resources in a network. In blockchain, this means that the data and computing resources needed to maintain the network are distributed across multiple nodes. This distribution makes blockchain technology extremely resilient and secure: even if one or more nodes go offline, the network can continue to function because the other nodes can assume their responsibilities.

2.2. METHODS OF DECENTRALIZATION

In this section, we will explore the three distinct degrees of decentralization and how they vary based on the implementation of two key concepts: **disintermediation** and **competition-driven decentralization**.

1. **Fully Centralized:** In this case, all operations, controls and decisions are managed by a single central authority or entity. There is no form of decentralization. Disintermediation is absent and there is no competition for control of the network.
2. **Semi-Decentralized:** In a semi-decentralized structure, some aspects of the system are decentralized, while others are still centrally controlled. This may result in some degree of disintermediation, where some functions can be performed without third-party intervention, but is not completely eliminated. Competition-driven decentralization may be present in some parts of the system, but may not be applied in all aspects.
3. **Fully Decentralized:** In a fully decentralized system, there is no central authority controlling the system or making decisions. All operations are handled by network users in a distributed manner. This level of decentralization leads to strong disintermediation, where transactions and operations can take place directly between parties without intermediaries. In addition, competition for control of the network can be a key element, where network actors openly compete to influence consensus decisions.

It is clear from the above explanations that decentralization and distribution are two crucial concepts in the design of blockchain networks. However, it is worth noting the *subtle difference* between the two.

Decentralization deals with decision making and who has the power to make decisions (**governance**), while distribution deals with the physical arrangement of resources in the network (**technical concept**).

Interestingly, a network can be decentralized without being distributed, and vice versa.

However, in practice, most decentralized networks are also distributed (this is because distribution is crucial to ensure network integrity and resilience).

2.3. UNVEILING THE CORE OF BLOCKCHAIN

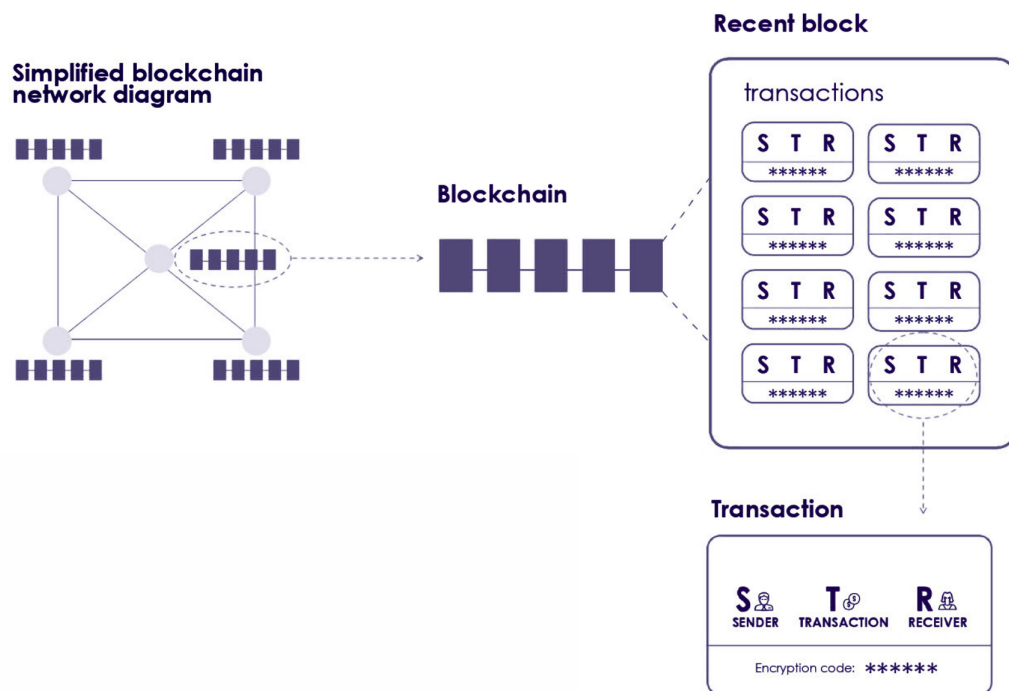
Finally, we reach the *juiciest part*: the fundamental concepts of blockchain. Let's explore the pillars on which this revolutionary technology is based!

Nodes: Nodes constitute the cornerstone elements of the Blockchain community. They represent the participants or entities that drive the network. These can be computers, laptops or servers belonging to individuals or organizations. Each node assumes a crucial role in the blockchain architecture, contributing its resources and capabilities to the functioning of the entire system.

Transactions: Each transaction in the blockchain represents a major event permanently recorded in digital history. When two or more nodes interact, a transaction is noted in the blockchain through a process known as hashing. This process ensures the security and immutability of each transaction, preserving the integrity of the data transmitted.

Blockchain: Let us now imagine all these transactions organized and collected into blocks. Each block constitutes a key point in the blockchain's history, containing a series of related transactions. These blocks are linked consistently through consensus mechanisms, forming an unbroken chain of information.

Ledger: Finally, there is the ledger, the core of the blockchain. This distributed public ledger is hosted on all community nodes, representing the cornerstone of transparency and authenticity. Every block of transactions finds its place within this ledger, recorded sequentially and accessible to all participants. It is the trusted guardian of every movement and transaction within the blockchain.



Visual Example of the Blockchain [1]

2.4. THE ANATOMY OF A BLOCK

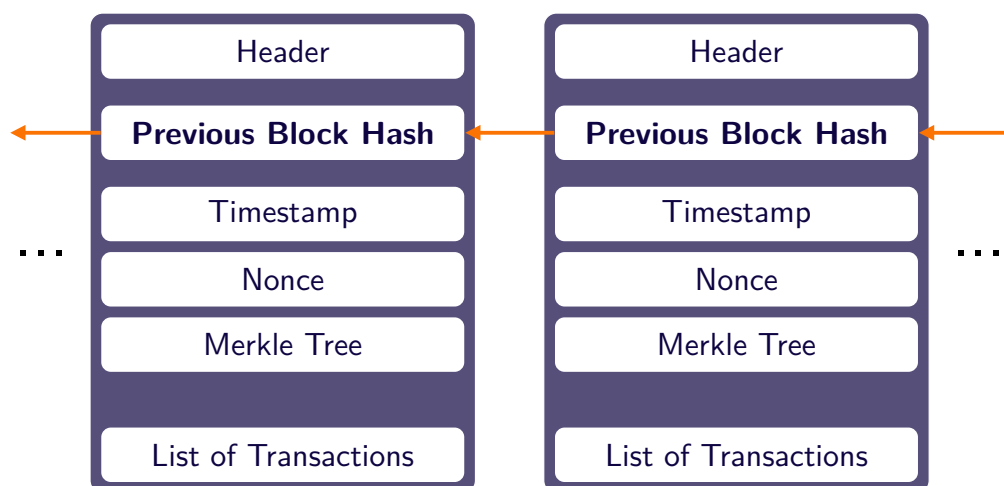
Let's go a little deeper! Let us now examine a block with a magnifying lens 🔍

Header:

- **Hash of the Previous Header Link:** This field contains the hash value of the previous block, thus creating an immutable chain of blocks. Hashes, such as the popular SHA-256 (256-bit Secure Hash Algorithm), are cryptographic functions that transform data into fixed-length strings, which are essential for verifying data integrity.
- **Nonce:** This is an arbitrary value that is changed during the mining process until a hash is found that meets specific criteria.
- **Timestamp:** This indicates the precise time when the block was created.
- **Merkle Root:** This is the root hash of a Merkle tree, a data structure that enables efficient verification of the integrity of transactions contained in the block.

Body:

- **List of Transactions:** This section lists all the transactions included in the block. Each transaction represents an exchange of value or data within the blockchain network, contributing to the traceability and security of transactions.



Visual Example of the Blocks

3. CONSENSUS MECHANISMS

3.1. HIGH LEVEL CONSIDERATIONS

Before diving into the details of consensus mechanisms, let's take a look at the advantages and limitations of using blockchain:

Pros	Cons
Among the main benefits, decentralization emerges as one of the key pillars, enabling a distributed network in which no central authority has absolute control. This not only increases the security and reliability of the network, but also promotes transparency and trust among participants, making every transaction traceable and verifiable by anyone. In addition, the immutability of data recorded on the blockchain, coupled with its high availability , ensures that information is protected from manipulation and always accessible when needed. These benefits, along with process simplification, efficiency in regulations and cost savings , turn blockchain into a reliable and programmable platform for a wide range of applications. Specifically, cost savings result from the elimination of intermediaries in transactions, enabling a direct transfer of value between the parties involved.	However, despite its advantages, blockchain technology also has some significant limitations that hinder its wider adoption. The appearance of the new technology , for example, poses challenges in terms of familiarization and implementation for many organizations. In addition, the scalability of blockchain, especially in public networks, may be limited with respect to the needs of high-frequency transactions. The issue of data privacy and confidentiality remains an area of concern, as information recorded on the blockchain is permanent and accessible to all participants. Limited adoption, interoperability between different blockchain platforms, and complex regulatory issues represent additional challenges that must be overcome to maximize the potential of blockchain technology.

Benefits and Limitations of Blockchain Technology

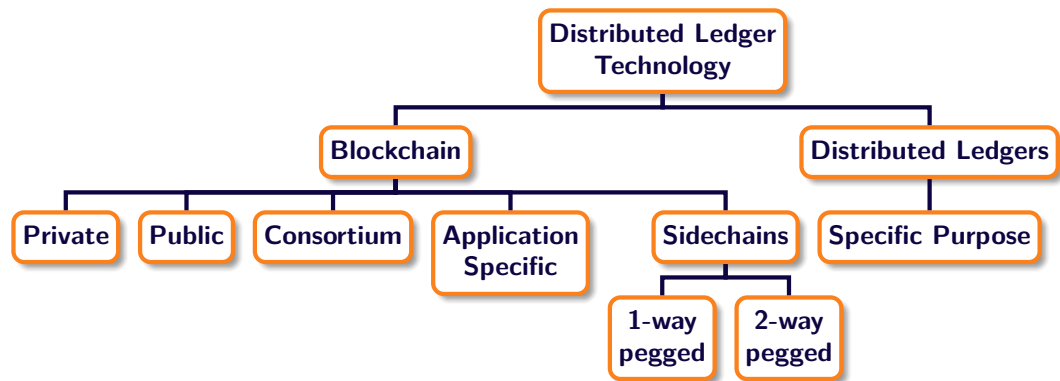
3.2. DLT CATEGORIES

Distributed Ledger Technologies (DLTs) can be categorized in different ways based on their use and structure. One of the main categories is Blockchain, which is a specific type of DLT used for shared and immutable ledgers.

Blockchain can be further divided into three main types: public, private and consortium. **Public blockchains** are open to anyone who wants to participate and are decentralized. **Private blockchains** are controlled by a centralized organization or entity and are accessible only to authorized users. **Consortial blockchains** are managed by a group of organizations working together to manage the network.

In addition to blockchains, there are other types of DLTs such as Sidechains and Distributed Ledgers. **Sidechains** are blockchains linked to a main blockchain and can be used for specific purposes or to enhance the functionality (scalability) of the main blockchain.

Distributed Ledgers, on the other hand, are shared ledgers that can be customized for specific purposes and are not necessarily blockchain-based.



Distributed Ledger Technologies Categories

3.3. BYZANTINE GENERALS PROBLEMS

The Byzantine Generals problem is a fundamental concept in the context of distributed consensus mechanisms. Imagine a situation in which several commanders of an army must coordinate to attack or retreat, but some of them may be traitors or communications between them may be compromised. In this scenario, it is essential that the generals reach a consensus on the strategy to be adopted despite the uncertainty and the presence of potential mistakes or sabotage.

PBFT (Practical Byzantine Fault Tolerance) solves this problem by establishing a set of requirements to ensure reliable consensus among participants (Of course, we are no longer talking about battles, but about blockchain!). These requirements include:

- **Agreement:** All nodes must agree on a single value proposition or decision.
- **Integrity:** Information and transactions must be authentic and unchanged during the consensus process.
- **Validity:** Only valid and legitimate transactions must be accepted and confirmed.
- **Fault Tolerance:** The system must be able to function properly even if some nodes fail or misbehave.
- **Termination:** The consensus process must eventually reach a conclusion and produce a final outcome.

Using PBFT, participants can collaborate safely and reliably even in the presence of faulty or malicious nodes. This consensus mechanism is critical to ensuring the consistency and reliability of transactions within distributed networks, such as those based on blockchain technology.

3.4. SECURITY MECHANISMS

In the blockchain, the application of security mechanisms is critical to ensure the security and integrity of transactions. These functions are used in different contexts, below we make a list and it is likely that you already know how some of them work.

Cryptographic Hash Functions: Hash functions are fundamental in the blockchain and are used to confirm and validate new blocks of transactions, to create unique addresses for accounts on the blockchain, to verify the authenticity of messages sent over the network and in Merkle trees.

Asymmetric cryptography: A cryptographic technique involving two keys: public and private. In the encryption process, the sender encrypts the message with the recipient's public key and, in the case of digital signatures, signs the message with his own private key, while the recipient verifies the signature using the sender's public key.

Merkle trees: A data structure that ensures the integrity of transactions in the blockchain. Transactions are organized in a binary tree, and hashes are used to ensure the integrity of the structure. They allow for quick verification of whether a transaction is included in a blockchain without having to examine all transactions.

3.5. TYPE OF CONSENSUS MECHANISMS

When it comes to exploring the vast world of consensus mechanisms within blockchains, we are faced with a myriad of options. However, we will focus mainly on three of them: the Proof of Work, the Proof of Stake and the Delegated Proof of Stake.

🪙 **Proof of Work (PoW):**

In PoW, participants, called “miners,” compete to solve complex cryptographic puzzles. The first miner to solve the puzzle gains the right to create and confirm a new block of transactions on the blockchain. This process requires an enormous amount of computing power, making the system secure but also energy intensive.

💎 **Proof of Stake (PoS):**

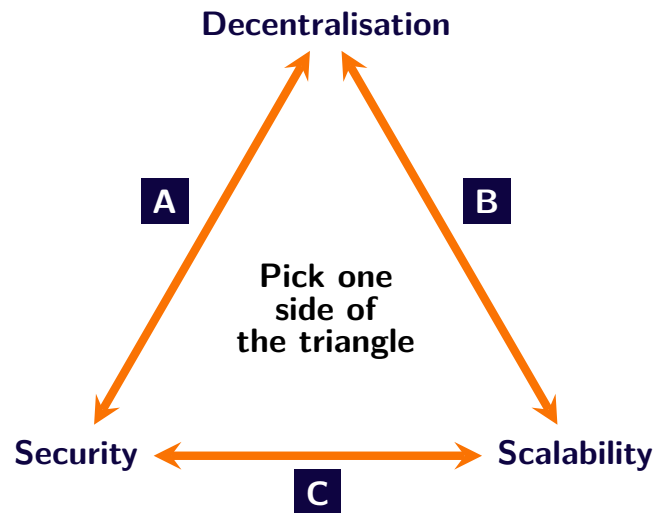
In PoS, the right to create and validate a new block depends on the amount of cryptocurrency owned and “played” by participants, called “validators.” The more cryptocurrency that is “played,” the greater the probability of being chosen to validate a block and receive rewards. Because it does not require as intensive computing power as PoW, PoS is more energy efficient.

👥 **Delegated Proof of Stake (DPoS):**

In DPoS, participants elect “delegates” responsible for creating and confirming blocks. These delegates are voted on by other participants based on trust and reputation. DPoS is designed to scale more efficiently than PoW and PoS, allowing for faster confirmation of transactions.

3.6. BLOCKCHAIN TRILEMMA

The trilemma of **decentralization**, **scalability** and **security** is one of the most significant challenges in designing and implementing an effective blockchain. The consensus mechanism plays a crucial role in balancing these three dimensions.



The Blockchain Trilemma

Decentralisation refers to the **distribution of decision-making power** among network participants rather than concentrating it in a single centralized entity. An effective consensus mechanism should ensure that no single actor or group of actors has complete control over the network, thus promoting decentralization. However, decentralization can be challenged by the need to reach agreement on transactions and state updates among geographically distributed nodes.

Scalability refers to the **ability of the system to handle an increasing number of transactions** and to grow efficiently as the workload increases. An efficient consensus mechanism should enable the network to process a large number of transactions quickly and economically without compromising its security or decentralization. However, achieving scalability can be hampered by the computational complexity and communication required to achieve distributed consensus.

Security is critical to **protect the network from malicious attacks, manipulation, and fraud**. A robust consensus mechanism should ensure the integrity and immutability of data stored on the blockchain, as well as resistance to attack attempts by malicious actors. However, maintaining a high level of security may require compromises on the speed or scalability of the network.

To address the trilemma, there are two main approaches in designing consensus mechanisms:

- **Message-based consensus mechanisms:** in this approach, network participants communicate with each other to reach consensus on the validity of transactions and the state of the blockchain.
- **Shared memory-based consensus mechanisms:** in this approach, participants share a common memory space and use synchronization techniques to ensure that all nodes in the network agree on the state of the system.

3.7. DISTRIBUTED CONSENSUS

In the context of distributed systems, reaching consensus among nodes that do not trust each other on the final state of data is of paramount importance. This process ensures that all nodes within the network agree on the validity and order of transactions. Several algorithms are employed to accomplish this task, falling primarily into two categories:

Approaches Based on Proof and Leader Election by Lottery: These consensus mechanisms rely on cryptographic proof or random selection of leaders to propose and validate transactions. An example of this category is Proof of Work (PoW), where nodes compete to solve cryptographic puzzles in order to validate transactions and add blocks to the blockchain.

Based on Byzantine Failure Tolerance (BFT): These approaches focus on the ability to resist failures and malicious actions within the system. A notable example is Practical Byzantine Fault Tolerance (PBFT), which involves a consensus process among a select group of nodes that must be at least a majority to ensure the integrity of the system.

Let us now look at some key aspects related to Distributed Consensus:

Failure Tolerance and Replication

Failure tolerance is critical in a distributed environment because it implies the **ability of the system to continue to function** even in the presence of malfunctions or failures. Inevitably, any system can incur problems, and the distributed system is no exception. Replication is an effective strategy for dealing with problems related to failing nodes. It replicates the behavior of nodes so that, should one or more nodes fail or behave non-compliantly, the system can still operate consistently.

Machine State Replication is a technique that aims to ensure that all servers have a consistent and identical view of data state. This is achieved by ensuring that all servers start from the same initial state, receive requests in a specific order, and produce the same deterministic output for the same input. This replication process ensures consistency and uniformity across the system, regardless of the condition of each node.

Lower Limits and Security

When dealing with Distributed Consent, it is important to understand the lower limits necessary to ensure that the system works properly. These limits are defined in terms of the **number of nodes required to achieve consensus**. In the case of Classical Federated Consensus (CFT), at least $2F + 1$ nodes are required to ensure consensus, while in the case of Byzantine Failure Tolerance (BFT), at least $3F + 1$ nodes are required. These minimum requirements are essential to ensure that the system is protected from malicious attacks or accidental failure.

N.B. “F” represents the maximum number of nodes that can fail within a distributed system without compromising its integrity or ability to achieve consensus.

In addition, for the consensus process to be considered valid and reliable, transactions **must meet certain security and functionality requirements**. These include agreement on an end state of the data, integrity of the transactions, their validity against the rules of the protocol, and the defined and stable conclusion of the consensus process.

3.8. DISTRIBUTED CONSENSUS PROTOCOLS

Here are some very different approaches to achieving consensus within a distributed network.

Paxos (Lamport)

- Uses $2F + 1$ processes to ensure failure tolerance.
- It is based on a two-stage protocol: the **preparation stage** and the **acceptance stage**.
- Participants include **proposers**, who propose values, and **acceptors**, who accept the proposed values.

Raft (Ongaro & Ousterhout)

- Assumes that the **leader is always honest**.
- Elects a single leader in each “term” (unit of time).
- Solves three main problems: leader election, log replication, and security.

Practical Byzantine Fault Tolerance (PBFT)

- It consists of three sub-protocols: **normal operation**, **view change** and **checkpoint**.
- It divides operations into **pre-preparation**, **preparation** and **confirmation phases**.
- The participants are **replicas**, with one of them designated as the primary node and the others as backups.
- It ensures Byzantine failure tolerance with a number of nodes $N \geq 3F + 1$.
- Uses digital signatures and certificates to maintain the integrity of operations.
- The **view-change protocol** is triggered when the primary node is suspected of being faulty.
- The **checkpoint sub-protocol** discards old messages in replica logs to ensure data consistency.

Paxos, Raft, and PBFT are distributed consensus protocols with specific features and mechanisms to ensure fault tolerance and secure operations. However, let us now look in detail at the alternatives that are most popular today, namely Proof of Work (PoW) and Proof of Stake (PoS).

3.9. PROOF OF WORK (POW)

Proof of Work (PoW) works through a competitive process in which miners compete to solve complex mathematical problems. *But how does it work practically?*

Basically, a node proposing a block must find a “nonce” (a random number) such that the **hash of the block**, together with the nonce, is **less than a threshold value**. This process requires considerable computational effort. Miners search for the nonce through trial and error, varying the nonce until they find a value that meets the difficulty requirement. Once found, the proposed block is inserted in the network and **nodes check the validity** of the block and add it to the chain.

Hence, PoW has some important properties, which we list below:

- **Common Prefix Property:** Ensures that honest nodes share a large common prefix in the blockchain. If this is not done, agreement between nodes on the same chain cannot be guaranteed, compromising the security of the network.
- **Chain Quality Property:** Ensures that the blockchain contains a minimum level of correct blocks created by honest nodes. If this chain quality is compromised, the validity

of the chain itself cannot be guaranteed, opening up the possibility of security breaches.

- **Chain Growth Property:** Implies that new correct blocks are continuously added to the blockchain. If chain growth is affected, the network's ability to maintain its uptime (liveness) can be compromised, potentially causing blockages or system paralysis.

Furthermore, **target difficulty** is an essential parameter in the context of the Bitcoin protocol. It is a fundamental value that is regularly updated every 2016 blocks, ensuring a stable block creation rate of approximately 10 minutes. It represents the **maximum value a block hash must have** to be considered valid.

Incentives also play a vital role in this case. In the case of Bitcoin, miners are rewarded with a per block reward and transaction fees. These economic incentives **motivate miners to continue participating in the mining process**, ensuring the security and stability of the network.

3.10. PROOF OF STAKE (POS)

Proof of Stake (PoS) is a consensus algorithm used in blockchain that differs significantly from Proof of Work (PoW). In PoS, there is no traditional concept of “mining” as in PoW, where miners compete to solve complex mathematical problems. Instead, PoS participants are called validators or stakeholders and **participate in the consensus process by putting a portion of their cryptocurrency**, known as a “stake,” into play.

Variants of PoS include **Chain-based PoS**, **Committee-based PoS** and **Delegated PoS**, each with its own characteristics and consensus mechanisms. For example, in Chain-based PoS, blocks are validated based on the number of coins owned by the participant, while in Committee-based PoS, a selected committee validates blocks based on specific rules. Delegated PoS allows participants to delegate their stake to a trusted node that acts as a validator on their behalf.

The process of earning rewards in PoS is sometimes called “virtual mining” because participants do not have to solve complex mathematical puzzles, but instead **must prove that they own a significant amount of cryptocurrency** and contribute to the security and validity of the network. Control over the majority of the network also remains crucial in PoS, as control over a large amount of stakes allows a single actor to influence consensus decisions and potentially attack or control the network.

4. SMART CONTRACTS

4.1. SMART CONTRACT IDEA AND EVOLUTION

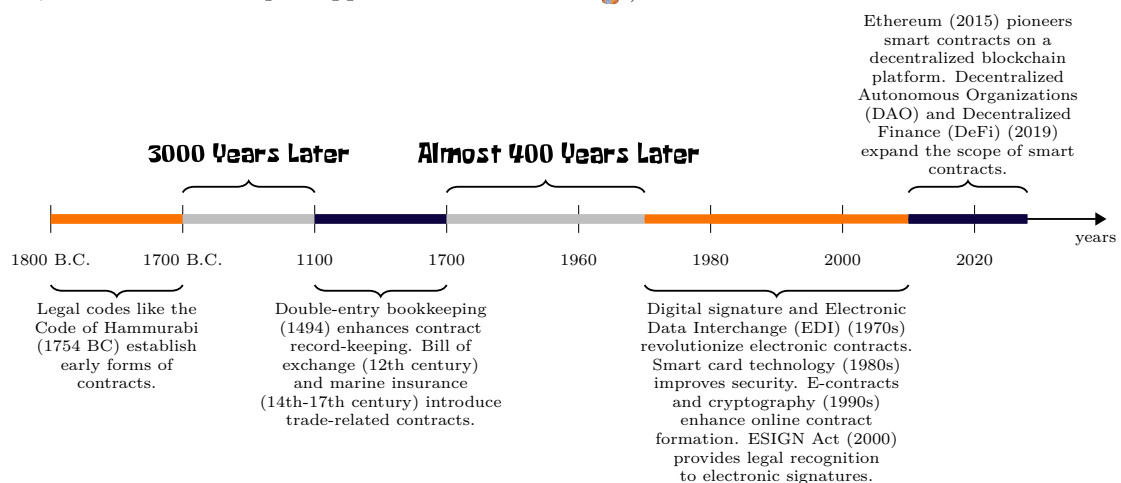
Smart contracts represent a milestone in the evolution of blockchain technology. They are electronic transaction protocols that execute the terms of a contract automatically and securely. Contrary to the perception of the term “*smart*”, smart contracts do not refer to contracts with artificial intelligence, but rather **executable programs** based on computer code.

As stated by **Nick Szabo** [4], the conceptual pioneer of this innovation:

“A smart contract is an electronic transaction protocol that executes the terms of a contract. The general objectives are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries. Related economic goals include lowering fraud loss, arbitrations and enforcement costs, and other transaction costs.”

So, in essence, smart contracts promise to make contract processes more efficient, secure, and less burdensome by offering an innovative way to **automate and ensure the enforcement of contract terms**.

As with blockchain, smart contracts represent a combination of technologies that humans have created in their history. Below is a nice timeline (We hope the font will not make you ask, “*Who lives in a pineapple under the sea?*” 🍍):



Smart Contracts Key Inventions Timeline

So what is different from a normal contract? In traditional contracts, the verbal text on paper is interpreted by human beings or lawyers (“wet” code). Their legal enforceability is subject to the **discretionary compliance and semantic flexibility** of human contracts. So, legal validity depends on the willingness of the parties involved to abide by the rules and the ability to interpret the text flexibly and adaptable to the situation. On the other hand, smart contracts are rules and contracts executed as in a distribution machine (“dry” code). They are legally binding in a technological way, being contracts executed inexorably

by code (as **Lessig** argues: “*code is law*” [2]), which **cannot be violated and proceed unstopably even if conditions have changed**.

4.2. RICARDIAN CONTRACTS

A Ricardian contract is a legal document that is **comprehensible and accepted by both humans and machines**, and that is linked to all anticipated future operations (transactions). This type of contract possesses several distinctive properties. First, it is offered by an issuer to holders and represents a right of value managed by the issuer. It is easily readable by people, like a contract on paper, but is also interpretable by programmes, like a database. The contract is digitally signed and contains keys and server information, as well as being associated with a unique and secure identifier. This is basically an **evolution of the original contract**, indeed each contract can be represented by a graph of objects in prose, interacting in an ecosystem.

But hey, to really paint the picture, what sets it apart from a smart contract? Ricardian Contracts focus primarily on being a primary document for the issuance of a digital asset, with an emphasis on **richer semantics**. In contrast, Smart Contracts are **more oriented towards programmability** and follow a deterministic approach. So, while Ricardians aim to provide a comprehensive document that is easily understood by all parties involved, Smart Contracts put more emphasis on automating contractual activities through code.

Let's put it in terms of desserts: Ricardian contracts are like grandma's homemade apple pie - wholesome, comforting, and always a crowd-pleaser. Smart contracts, on the other hand, are more like molecular gastronomy - innovative, impressive, but you're not entirely sure if it's still food...

4.3. BITCOIN OR ETHEREUM? ₿ ⬥

Ethereum and Bitcoin differ significantly in their ability to execute smart contracts. **Bitcoin**, using a full non-Turing script language, is **limited in its ability to handle complex contracts** or sophisticated decentralised applications. On the other hand, **Ethereum** offers a **more powerful environment for smart contracts** through the Solidity language, running on its **Ethereum virtual machines** (EVMs), which are full Turing. This allows developers to create much more sophisticated smart contracts on Ethereum, with the ability to perform a wide range of calculations or programmable logic. In short, Ethereum offers a more flexible and powerful platform for smart contract development than Bitcoin.

4.4. SMART CONTRACT COMPLIANCE ISSUES

There are several problems with smart contracts that can cause uncertainty and legal challenges:

1. **Understanding the Computer Code:** courts and parties involved may have difficulty understanding the computer code on which smart contracts are based. This can make it difficult to assess all contractual obligations and terms fully and accurately.
2. **Legal Bindingness:** in some jurisdictions, the legality and bindingness of smart contracts are not clearly defined. Some courts may not recognise smart contracts as

legally binding agreements.

3. **Determinism of Smart Contracts:** smart contracts operate deterministically, performing only what they have been programmed to do. This means that they cannot adapt or resolve situations not foreseen in the code.
4. **Legal Considerations and Exceptions:** there are many legal considerations and exceptions that must be taken into account when entering into a contract. Smart contracts may not be able to handle these complex situations effectively.
5. **Different Laws and Jurisdictions:** laws and jurisdictions vary from country to country. Some jurisdictions may be more favourable to the recognition of smart contracts than others.

Taken together, these factors contribute to smart contracts being subject to legal uncertainty and may require further regulation and clarification to be fully accepted and reliably used.

Smart Legal Contracts are an evolution of traditional smart contracts, offering a solution to address the legal challenges associated with their implementation. While smart contracts consist of computer code and operate according to the “code is law” principle, Smart Legal Contracts combine legal language, parameters and code to create a legally binding contract.

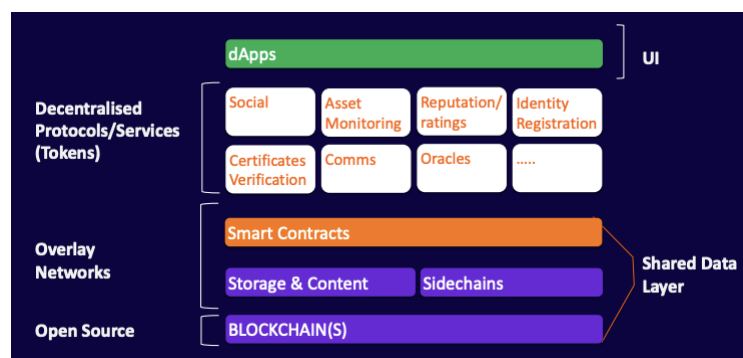
The main objective of Smart Legal Contracts is to provide neutral performance within the terms and conditions defined by the contractual parties. Unlike smart contracts, Smart Legal Contracts recognise that the **law has the final word and may be subject to legal interpretation and modification**.

In terms of modifiability, Smart Legal Contracts offer **more flexibility than traditional smart contracts**. While the code of smart contracts cannot be changed once activated, Smart Legal Contracts allow changes to the code and can be paused if necessary.

That’s why Smart Legal Contracts find application in a **wide range of industries and types of agreements**, as they offer a solution for creating legally binding contracts that also incorporate code-based automation and execution elements.

4.5. DAPPS

Now that we have studied smart contracts in detail, it is time to turn our gaze towards the stars of the show: the DApps. However, before we let ourselves be enchanted by their wonders, it is crucial to take a closer look at the ecosystem that supports them. This overview not only prepares us for the next act, but also shows us the interconnection and synergy between the layers that make the DApp venture possible.



Representation of the Blockchain Stack

From bottom to top:

- **Blockchain:** This is the foundation on which the entire ecosystem is based.
- **Storage, Content and Sidechains:** This layer manages the storage of data and the content associated with transactions on the blockchain. It also includes sidechains, which are separate blockchains connected to the main blockchain. Sidechains allow specific transactions to be performed without clogging up the main blockchain, improving scalability and enabling experimentation with new functionality without compromising security.
- **Smart Contracts:** *We have just seen them :)*
- **Decentralised Protocols and Services:** This layer comprises the decentralised protocols and services that enable the creation, management and exchange of digital tokens on the blockchain. **Tokens** can represent digital currency, digital assets, rights to use resources and much more. These protocols and services provide a standardised infrastructure to facilitate the interoperability and adoption of tokens within the blockchain ecosystem.
- **DApps:** This is the highest layer of the stack and represents decentralised applications that utilise the layers below to provide a wide range of services and functionality to users. DApps exploit the unique features of blockchain, such as security, immunity from censorship and transparency, to offer innovative and reliable solutions.

Basically, decentralised applications are software programmes that operate by different methods. In addition to their underlying architecture and functionality, DApps may also present different **user interfaces** (UIs) to allow users to interact with them in intuitive and user-friendly ways. They are generally classified into three types: Type 1, Type 2 or Type 3.

Type 1	Type 2	Type 3
Type 1 DApps operate on their own dedicated blockchain . A prominent example of this is Ethereum-based smart contract DApps. These DApps, such as decentralized finance (DeFi) applications, execute on the Ethereum blockchain and may utilize a native token like ETH.	Type 2 DApps utilize an existing established blockchain . They rely on Type 1 blockchains and introduce custom protocols and tokens. For instance, DAI is built on top of Ethereum but features its own stablecoins and mechanisms for distribution and governance.	Type 3 DApps leverage the protocols of Type 2 DApps . An example is the SAFE Network, which utilizes the OMNI network protocol. These DApps build upon existing decentralized infrastructure to offer specialized functionalities.

Benefits, Limitations, and Additional Considerations of Blockchain Technology

DApps must be completely open source and autonomous, with no one entity controlling most of their tokens, changes must be based on community consensus, data must be cryptographically secure on a decentralised public blockchain, and a cryptographic token must be used for access and to incentivise contributors, generated by the DApp itself as proof of value.

4.6. ORACLES

Unfortunately, smart contracts, and consequently DApps, do not have the ability to directly access external data. However, to overcome this limitation, special tools called **oracles** come into play.

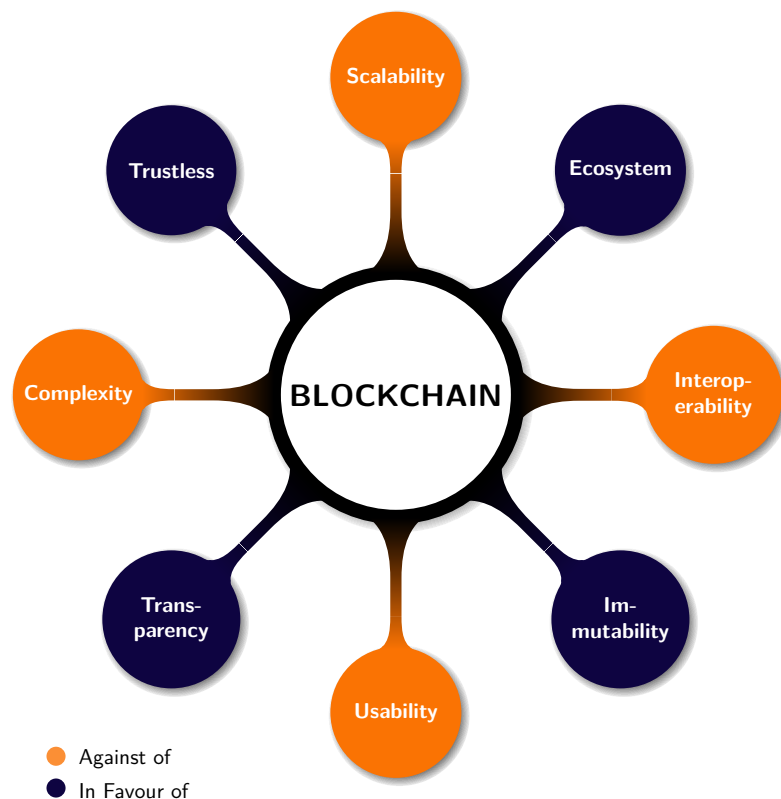
In DApps, oracles act as a bridge between the real world and the blockchain, allowing smart contracts to **access external data** and perform actions in response to certain conditions. Oracles are services provided by third parties and are designed to provide reliable external data to smart contracts on the blockchain. This data can cover a wide range of information, such as weather conditions, payment confirmations or price changes.

The operation of oracles varies depending on the implementation and the blockchain used. In Bitcoin, for example, an oracle can write **data about a specific transaction**, while in Ethereum, data can be **stored in the smart contract** itself for use by other smart contracts through message calls.

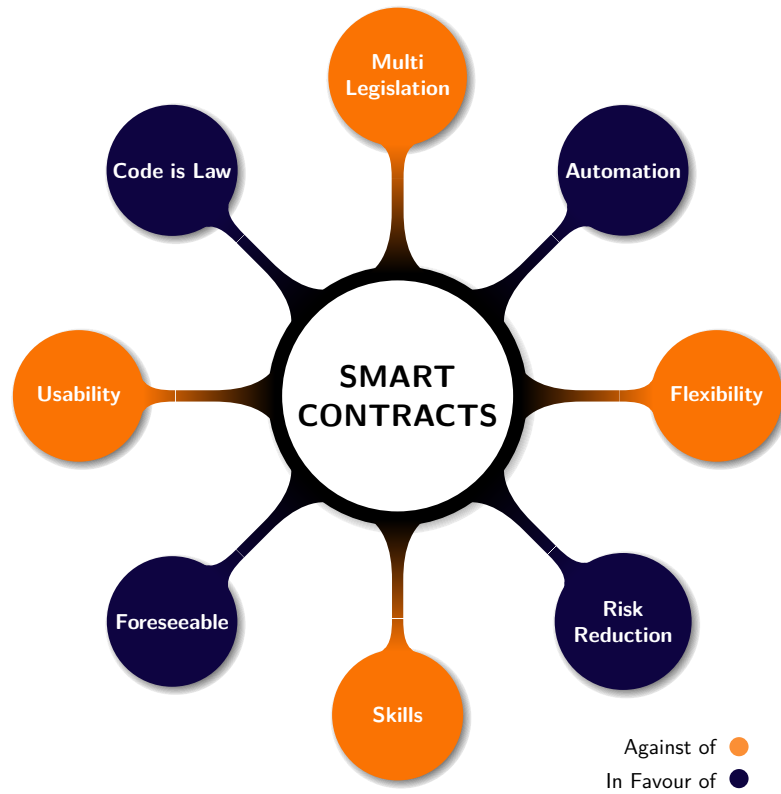
Oracles are key to enabling DApps to interact with the outside world and perform actions based on real events, thus expanding the possibilities of blockchain usage beyond the confines of the closed system of the blockchain itself.

4.7. BLOCKCHAIN AND SMART CONTRACTS SUMMARY

Here we include schemes that summarise the various concepts we have addressed in these sections:



Blockchain Concepts Summary



Smart Contracts Concepts Summary

5. TRANSACTIONS AND WALLET

5.1. TRANSACTION TYPES

Transactions in the blockchain can be classified into three main types:

Contract Creation: Contract creation occurs when a new smart contract is generated on the blockchain. Smart contracts are self-executing programmes that exist on the blockchain and contain code that can manage funds, perform calculations and even interact with other contracts. When a new contract is created, it is registered on the blockchain along with its code and initial parameters.

Calls: Calls occur when executing functions or interacting with existing smart contracts on the blockchain. These interactions can include performing operations defined in the contract code, such as transferring funds, modifying data or triggering events. Calls can be made by human users or other smart contracts and are essential for the execution of actions and updating the state of the blockchain.

Value Transfer: Value transfer represents the movement of cryptocurrencies or tokens from one address to another on the blockchain. This type of transaction simply involves changing balances in users' wallets. For example, when cryptocurrencies are sent from one wallet to another, a value transfer transaction is generated on the blockchain. These transactions allow users to exchange digital assets and are the foundation of financial activities within the blockchain.

5.2. ACCOUNTS IN THE BLOCKCHAIN CONTEXT

In Ethereum, there are two main categories of accounts:

- **EOAs (Externally Owned Accounts):** They represent user accounts on the blockchain. EOAs are managed via private keys and allow users to send transactions and interact directly with the Ethereum network.
- **CA (Contract Accounts):** These accounts are created during the creation of smart contracts on the blockchain. They too are identified by an address and are used to execute specific code via transactions.

Obviously, the state system evolves through transactions, meaning that **state is constantly being created or updated as a result of the interaction between accounts and the execution of transactions**. The state transition process in Ethereum consists of the following steps:

1. **Validation** of the transaction.
2. Calculation of the **transaction fee and signature** by the sender. Update of the account nonce.
3. Provision of sufficient ether (ETH) to cover the **cost of the GAS** required to execute the transaction. The GAS represents the **remuneration for the miner** and is **calculated according to the complexity of the transaction**.

4. If the transaction fails due to lack of funds to cover the GAS or for other reasons, it is cancelled, but the sender is **still required to pay the validators fees**.
5. Finally, the sender receives a **notification with the change** (if any) **and the commission**, while the miners get paid for their work. The blockchain then proceeds to a new state.

After discussing this process, it is important to understand two key concepts that play a fundamental role in managing the state of the Ethereum blockchain: **Account State** and **Storage Trie**. The former represents the **current status of all accounts on the blockchain**, including balances, nonce and other attributes, while the latter is a **data structure used to store information within smart contracts**, organising it in a tree structure for efficient access.

5.3. WALLET

A wallet represents a digital infrastructure dedicated to the management and storage of cryptocurrencies. It acts as a **database to store private keys** (along with all transactions).

Wallets are **hierarchical and deterministic**: they use a master password, the seed, from which the private keys are uniquely derived. The seed must be written down, saved and protected, since with it anyone can reconstruct the entire wallet and thus gain control over it.

There are two main types of wallets used in the context of cryptocurrencies:

Hot Wallet 🔥

This is a wallet implementation that keeps **private keys accessible via software on network-connected devices**. Although it offers immediate and convenient access to funds, it can be vulnerable to cyber attacks. An example of hot wallet is **MetaMask**.

Cold Wallet ❄️

In contrast to the hot approach, the cold wallet operates offline and stores **private keys in dedicated hardware devices**. While offering a higher level of security, it requires more effort to access the funds. An example of cold wallet is **Ledger**.

Access and control of funds within a wallet are governed by two key components:

- **Private Key**: This is a cryptographically secure sequence of characters that serves as a means of authorising transactions and asset transfers. The safekeeping of the private key is of paramount importance in ensuring the security of digital assets.
- **Address**: Represents a unique code used to identify the account within the blockchain network. Addresses enable the receipt of funds and are generated from the public key associated with the private key.

*But how are DApps able to communicate with identity and wallet? All thanks to Web3. **Web3 is the standard library** for facilitating the communication of DApps on the Ethereum blockchain. This library provides a number of functions that simplify the interaction between the environment in which DApps operate (such as browsers, DApps themselves, MetaMask) and the blockchain.*

LIST OF FIGURES

Blockchain Key Inventions Timeline	6
Visual Example of the Blockchain [1]	8
Visual Example of the Blocks	9
Distributed Ledger Technologies Categories	11
The Blockchain Trilemma	13
Smart Contracts Key Inventions Timeline	17
Representation of the Blockchain Stack	19
Blockchain Concepts Summary	21
Smart Contracts Concepts Summary	22

LIST OF TABLES

Benefits and Limitations of Blockchain Technology	10
Benefits, Limitations, and Additional Considerations of Blockchain Technology	20

BIBLIOGRAPHY

- [1] Finyear. *Blockchain: Powering the Internet of Value*. Accessed: 2024-03-15. 2020. URL: <https://www.finyear.com/attachment/637653/>.
- [2] Lawrence Lessig. *Code and Other Laws of Cyberspace*. New York, NY: Basic Books, 1999. ISBN: 0465039138.
- [3] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: 2024-03-15. Dec. 2008. URL: <https://bitcoin.org/bitcoin.pdf>.
- [4] Nick Szabo. "Formalizing and Securing Relationships on Public Networks". In: *First Monday* 2.9 (Sept. 1997). DOI: [10.5210/fm.v2i9.548](https://doi.org/10.5210/fm.v2i9.548). URL: <https://firstmonday.org/ojs/index.php/fm/article/view/548>.
- [5] Omar Abdullah Zaid. "WERE ISLAMIC RECORDS PRECURSORS TO ACCOUNTING BOOKS BASED ON THE ITALIAN METHOD?" In: *The Accounting Historians Journal* 27.1 (2000), pp. 73–90. ISSN: 01484184. URL: <http://www.jstor.org/stable/40698486> (visited on 03/15/2024).