cryptography

MDPI

Article

Beyond Bitcoin: Recent Trends and Perspectives in Distributed Ledger Technology

Diego Romano and Giovanni Schmid *

Istituto di Calcolo e Reti ad Alte Prestazioni, 80131 Naples, Italy; diego.romano@cnr.it
* Correspondence: giovanni.schmid@cnr.it; Tel.: +39-0816139529

Abstract: In the last four years, the evolution and adoption of blockchain and, more generally, distributed ledger systems have shown the affirmation of many concepts and models with significant differences in system governance and suitable applications. This work aims to analyze distributed ledger technology (DLT) critically. Starting from the topical idea of decentralization, we introduce concepts and building blocks currently adopted in the available systems centering on their functional aspects and impact on possible applications. We present some conceptual framing tools helpful in the application context: a DLT reference architecture, off-chain and on-chain governance models, and classification of consensus protocols. Finally, we introduce the concept of process authenticity, reviewing tools and strategies for integrating DLT with the physical world and proposing a constructive scheme for the authentication of a physical resource through alphanumeric data.

Keywords: blockchain technology; process authenticity; tokens; anchors; oracles

Citation: Romano, D.; Schmid, G. Beyond Bitcoin: Recent Trends and Perspectives in Distributed Ledger Technology. *Cryptography* **2021**, *5*, 36. https://doi.org/10.3390/cryptography5040036

Academic Editors: Rongxing Lu, Jun Shao, Duc-Phong Le and Cong Zuo

Received: 16 October 2021
Accepted: 9 December 2021
Published: 13 December 2021

1. Introduction

Bitcoin and related technologies are developing and spreading at unexpected rates and in unforeseen ways, impacting almost every industry. Since its conception, Bitcoin has been a discussion subject within restricted communities, such as IT and cryptographic experts, for nearly a decade. Nevertheless, over the years, the mass media have strongly influenced public opinion by emphasizing the possibility of profit by investing in cryptocurrencies. As a consequence, Bitcoin and a few alternative cryptocurrencies have given rise to a constantly expanding market in which people invest fiat money.

In parallel to the soaring exchange rates of Bitcoin, many researchers, companies, and entrepreneurs became interested in the underlying technology and its potentials. Speculations began to circulate about what could potentially be done with this new spectrum of technologies, starting from the vision of an "Internet of value" where assets, in the broadest sense of logical or physical resources with a value, are exchanged as quickly as information moves around the world just thanks to the Internet nowadays [1].

Currently, distributed ledger technology (DLT) [2,3]— especially those based on blockchain ledgers—is a subject of interest because many companies are rushing to take advantage of the perceived benefits of using a time-oriented, tamper-proof ledger of records as a public or intercompany backbone for transactions, data keeping, and process monitoring.

According to the leading provider of market and consumer data Statista (https://www.statista.com, accessed on 15 October 2021), worldwide spending on blockchain solutions will grow from 1.5 billion in 2018 to an estimated 15.9 billion by 2023. In the first quarter of 2021 alone, blockchain startup companies around the world amassed 2.6 billion U.S. dollars in venture-capital funding, more than the whole year of 2020 [4].

Expanding on the idea of a decentralized currency, as implemented in Bitcoin, many of these startups are explicitly building decentralized business models as an alternative to the existing centralized ones. They promote this idea as "cutting out the middleman" [5] thanks to a peer-to-peer network where customers and suppliers make transactions directly,

with the goal of decentralized data storage and saving money on transaction fees. In this regard, DLT offers new paradigms on how companies and people can interact and negotiate and new ways in which assets and data are represented and monetized. The benefits and prospects deriving from this received attention—sometimes perhaps a little too optimistically and superficially—for many application domains and related use cases. From smart grids to supply chains, e-government to e-health, and jurisprudence to finance, virtually no sector or application domain did not prospect cases and potential advantages in using some DLT. At the same time, however, non-trivial challenges arise because this technology is still under development, its related business models are largely unexplored, and a comprehensive legal framework is missing. Some surveys conducted by Statista, which involved hundreds of companies worldwide during the three years 2018–2020, reflect this state of affairs. For example, Figure 1 presents use cases for blockchain technologies as of 2021. This statistic shows that companies were working on use cases concerning the following four main broad topics: digital currencies/payments, data sharing/reconciliation, identity protection, and asset transfer/traceability, each of which can be in itself declined and adapted to many different application scenarios.

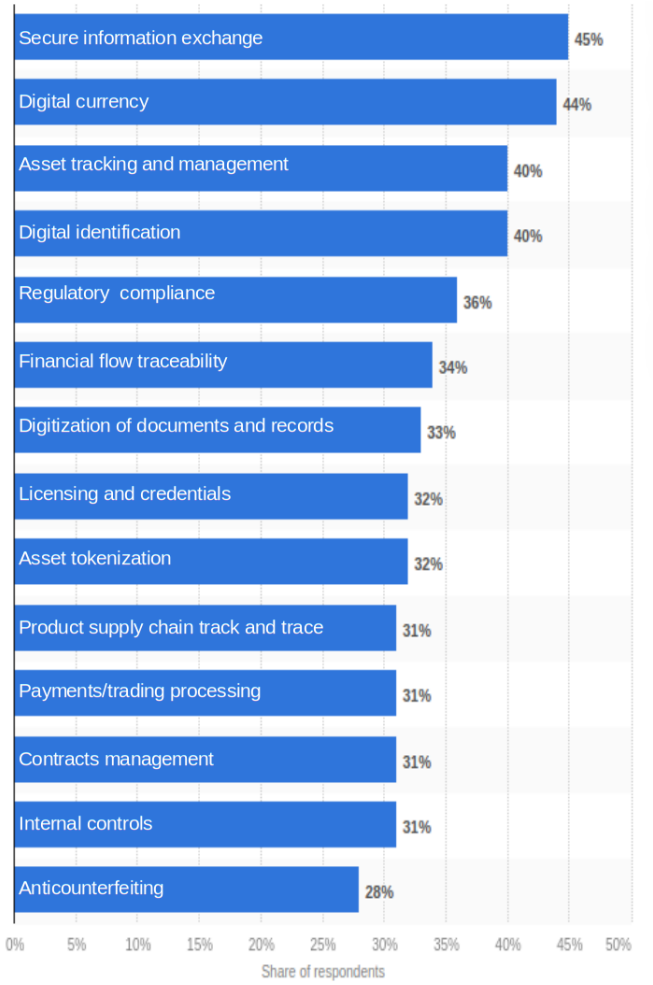

Figure 1. Global organizations' use cases for blockchain technology as of 2021 (©Statista) [6]. Percentages equal more than 100 percent because respondents were allowed to submit more than one answer.

Figure 2 instead shows the factors that over this span of years have been perceived as the main obstacles that discourage more significant investments in blockchain technology and platforms. As the outcomes show, only a strict minority of respondents believed there were no barriers. At the same time, significant uncertainties concerned the actual

usefulness and viability of this new technology, the risks associated with its adoption, and the lack of adequate regulations and standards.

1.1. Paper Contribution and Organization

Many of the contributions to the literature in recent years have tried to mitigate the above problems by orienting operators and people potentially interested in developing or using DLT thanks to an examination of existing solutions and their main characteristics in terms of both functionality and performance, referring to (a set of) specific application domains or use cases. In some cases (see Section 1.2), these contributions take the form of real *vademecum*,which is a ready but reasoned reference for a conscious choice of the most appropriate platform (often the best compromise among both DLT platforms and traditional systems and architectures) able to satisfy one's needs.

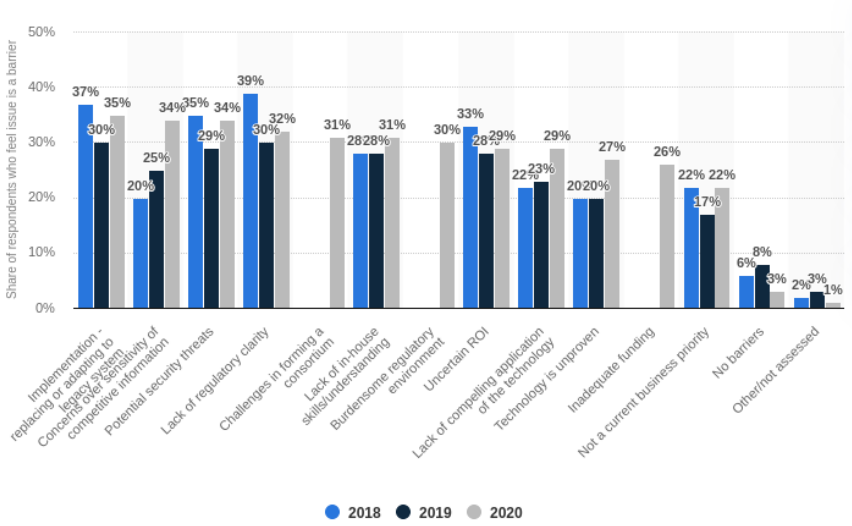

Figure 2. Perceived barriers for investing in blockchain technology during the three years 2018–2020 (©Statista) [7].

The purpose of the present work is different: it aims to discuss the evolution of some basic DLT concepts in terms of their scope and implementation, the emerging of new concepts and tools, and which prospects these concepts and tools can have in terms of the effectiveness of the decentralization solution obtained and its cyber-physical security. In particular, we will discuss the challenges for the successful adoption of blockchain when the use case considered provides that the system must interact somehow with the physical world because of the physical nature of the assets to be managed or the input sources for the smart contracts. Indeed, there is a gap between the cryptographic techniques to protect the digital information at the ledger layer and their interface to the physical realm, which can give rise to the so-called *garbage-in, garbage-out* issue [8,9] and nullify the purpose and reliability of the entire system. It should be clear that such a problem is very critical for any application scenarios where blockchain is supposed to be the winning solution for implementing an effective and trustful form of decentralization. It indeed affects the success of DLT solutions in at least two of the four broad use cases in which industry and academy are currently investing (see Figure 1), that is, identity protection and asset transfer/traceability. More generally, it represents a threatening barrier to the adoption of DLT outside the finance industry, where the actual use of this technology will depend on how and to what extent this problem will be solved.

Overall, this work intends to provide a broad overview of the main concepts, approaches, and technical solutions that have characterized the development of DLT so far. Its main contributions are:

- an in-depth critical discussion of the concept of *decentralization* and its implementation in current DL systems. Decentralization is the primary goal of any DL system, and we

discuss it for system architecture and governance, also considering how the system interacts with the outside world during its operation;
- a reference architecture for DLT, in order to provide an integrated and comprehensive view of the function and role of the various concepts and tools offered by this technology;
- a new classification for consensus protocols based on a five-component framework, aimed at underlining similarities and differences between the protocols recently introduced for DL systems and those introduced since the eighties for more traditional distributed systems. Rather than providing a broad array of blockchain consensus protocols with technical details, we focus on the milestones of research developments in this sector, citing only those protocols that we believe have introduced truly innovative features;
- a review of the most relevant tools and strategies introduced for integrating DL systems with the physical world. This integration is an emerging and exciting research field whose primary goal is to design and develop cyber-physical systems capable of implementing an extended notion of process authentication, which encompasses both digital and physical assets. In this context, we introduce the notion of *authentic data encoding for physical resources* (AD-ExPR), which results in a constructive scheme for the authentication of a physical resource through alphanumeric data. To the best of our knowledge, some authors in the literature implicitly used but did not explicitly define this scheme.

The remaining part of this survey is organized as follows. Section 2 discusses the two opposite models of achieving decentralization through DLT, with their significant differences in system governance, ledger management, and business models. Section 3.4.1 illustrates the core notions and their diverse implementations in major DLT platforms compared to a reference architecture. In particular, this section compares the different types of technical solutions proposed for the ledger structure and the consensus protocol to allow better performance and a more extensive set of possible applications. Section 4 discusses some emerging concepts and techniques whose scope is to reliably connect DLT networks to the physical context in which they must operate. The idea is to obtain cyber-physical systems capable of implementing an extended notion of process authentication, contemplating all the different agents that contribute to realizing the process, whether digital processes, human operators, or physical devices. Finally, Section 5 summarises our findings, indicating their implications for theory and practice.

1.2. Related Works

The scientific literature relating to DLT is vast, even if only considering contributions in reviews, surveys, and tutorials on the subject. Therefore, we will limit ourselves to pointing out those articles that most inspired the contents and organization of the present work.

The authors of [10] pursued two primary objectives: (i) providing an entry point for non-security experts to the security and privacy features enabled by blockchain technology and; (ii) helping specialists and researchers to explore the cutting edge cryptographic techniques in this context. Their analysis includes representative consensus protocols, anonymous signatures, secure multiparty computation, non-interactive zero-knowledge proofs, and secure verification of smart contracts. The aim of their contribution was to close the gap in understanding the security and privacy properties of blockchain systems in order to shed light on the root causes of the vulnerabilities in current deployment models and promote technological innovation on defense techniques. In the present work, we do not dwell on these aspects, which by now should have been sufficiently discussed in the literature. Instead, we discuss some approaches and tools that, also thanks to cryptography, serve to complement the security features of DL systems for their successful use in cyber-physical contexts.

The survey [11] explores blockchain architectures in terms of their network and data model, execution workflow, consensus protocol, and application domain; focusing on those crucial for deciding whether to adopt the technology or not and which of the available

solutions come closer to a given use case. This work's ultimate scope is to clarify the critical properties of blockchain systems, highlighting when and which blockchain technologies to choose and how they can be used and deployed. The authors of [12] pursued a similar objective since they present a comparative analysis of several major DLT platforms based on a number of quantitative and qualitative criteria in order to help developers and users to choose the best platform. In contrast, our approach focuses only on those aspects of the technology that we consider relevant for the use of DL systems in a wider context than the current one, with adequate levels of performance and reliability.

Xiao et al. [13] provide a comprehensive and detailed description of consensus protocols for distributed ledger (DL) systems mapping to a five-component framework. Their framework allows complete and in-depth analysis of how the blocks are proposed, transmitted, and validated within the network and the mechanisms and incentives to agree on which valid blocks add to the ledger. The framework we use in Section 3.3 to classify consensus protocols is inspired by their work. However, we have excluded the networking and incentive-related features, detailing aspects of the block proposal instead. This way, we tried to get a more homogeneous framework for evaluating the protocols, which reflects their method of execution.

A stimulating review of DLT technologies [14] presents a taxonomy structured in three tiers, focusing on application development. Moreover, the authors present a guide for developers to identify which DLT best fits the business model and the functionality requirements of their project. It is helpful to discriminate among options when the audience is already familiar with the concepts presented and aware of the related implementation difficulties.

To our knowledge, there are currently few references in the scientific literature of emerging technologies and approaches for the reliable interaction of DL systems with the physical world. One of the aims of this paper is to make a contribution in this sense, given that to illustrate these new technologies, we mainly had to draw on material published online or on our direct experiences in the field.

The recent book [15] investigates the challenges and opportunities for blockchain use in cyber-physical systems. However, it discusses blockchain architectural features useful in that context, rather than complementary tools introduced for integrating DL systems with the physical world.

The preprint [16] discusses the different kinds of blockchain *oracles*, elaborating on their potential role, technical architecture, and design patterns. Similar to what we do in Section 4.3, it illustrates, in some detail, the services offered by *Provable* and *Chainlink* as prime examples of centralized and decentralized oracle providers, respectively. However, the paper was published prior to some recent updates to these providers that are discussed in our work.

A classification for blockchain *crypto anchors* is given in [17], and, very recently, some of its authors mention the main features of a blockchain platform for product authentication that integrates different crypto anchors [18]. We have introduced the *AD-ExPR* notion as the first step towards a sounding theoretical and methodological approach in this direction.

2. The Quest for Decentralization and Its Realization through DLT

In addition to the financial speculations possible thanks to the cryptocurrency market, and regardless of the specific use cases, the success of DLT mainly lies in the demand for interoperable and reliable decentralization in internet services and architectures. *Decentralization* is a property that characterizes some distributed systems. A *distributed system* is a system whose components are on different networked computers, possibly in different geographic areas, which communicate and coordinate their actions in order to achieve a common goal. If some of these computers belong to different and independent administrative domains, so that they cannot be collectively managed through a unique organization but require the cooperation of multiple parties, then the distributed system is more appropriately called *decentralized*.

The adoption of a distributed system for implementing internet services has been a significant trend since the introduction of *microservice architectures* around 2011, a variant of service-oriented architectures where applications are collections of loosely-coupled, fine-grained processes that communicate over a network using technology-agnostic protocols, such as HTTP [19,20]. A growing number of big companies, smaller businesses, and startups have swapped monolithic applications for microservices architectures, which are deployed as distributed systems within a data center or at multiple remote sites, thanks also to virtualization and containerization techniques [21]. In this context, Cloud computing is a diffuse approach to offering and consuming IT services since it achieves economies of scale by sharing computer system resources and their on-demand availability without direct active management by users. However, companies usually deploy microservices and cloud architectures in a very centric way when administering their components and managing their trust. Amazon, Google, Facebook, Netflix, and many others were eventually prompted to shift from their giant and monolithic application stacks to cloud-based microservices [22,23], but only in order to improve their performance and scalability and to reduce difficulties in terms of software development, deployment, and management. From a governance point of view, their approaches remain purely centric, if not over-centric, where different service providers are administered by the same company (as in the case of YouTube and Google or Facebook and Whatsapp). In general, this envisioning has its roots in the economy of scale and the currently dominant business model on the Internet, with services substantially financed by online advertising, which in turn is all the more effective, the greater the ability to user profiling thanks to the analysis of their behavior on the net. Thus, the more online services a company owns (e.g., search engines and social media), the more behavioral data it can gather concerning its users, and the greater the chances it can provide advertisers and publishers with analytics for targeted advertising.

The demand for greater decentralization and balance of power among internet players originates from the needs of the base workforce and contributors. Firstly, developers, creators, and startup businesses must face a narrow space for innovation: fitting in the centralized offer from big players limits the operational freedom that characterizes innovative proposals. Even consortia of companies have limited functional space to propose innovative, collaborative products and services without relying on cloud services or centralized design frameworks. Some influential thinkers (e.g., techno-utopian Georges Gilder [24]) believe that DLT enables decentralized networks and business models. It can subvert organizational, social, and economic imprinting stemming from the business practices of companies, such as Google and Facebook, allowing for greater pluralism, the balance of power, and respect for human nature, rights, and freedom. Indeed, a slice of public opinion believes that the oligarchy currently characterizing the provision of the most common digital information and entertainment services may undermine the rights and freedoms of individuals. A few IT companies have a pivotal role in managing and distributing virtually any information to billions of individuals. Some fear this excessive concentration of power could degenerate into a dystopian world where people are continually monitored and conditioned to make choices against their natural, instinctive will. Shoshana Zuboff coined the term surveillance capitalism to denote "an emergent logic of accumulation in the networked sphere" based on "unexpected and often illegible mechanisms of [data] extraction, commodification, and control that effectively exile persons from their own behavior while producing new markets of behavioral prediction and modification" [25]. The sum of these ideas found in the blockchain keyword the strength to affirm a new decentralization proposal.

Whether or not one agrees with these analyses and conclusions, we believe that other practical circumstances point to decentralization as a critical factor for many application scenarios:

- risks deriving from assuming a single point of failure/trust, as in the case of the "trusted third party" assumption;

- costs reduction, both in the financial field and more generally in the commercial one, thanks to the elimination of intermediaries and interoperability "by design" ;
- need to coordinate safely and reliably complex distributed systems, which, by their nature, are better suited, in whole or in part, to peer-to-peer models.

In particular, decentralized-oriented technologies could support a killer feature in the context of a current primary trend in information processing, achieving a particular validated end after a pipeline of tasks performed by multiple and independent agents.

Cryptography, and more generally computer security, provides a series of concepts and tools to corroborate evidence for the authenticity of data stored or in transit and the identity of users and devices during their interactions. These are, for example, the concepts of message and entity authentication, as well as the related tools of message authentication codes, digital signatures, and challenge-response protocols. However, no existing single tool can encompass the notion of *process authenticity*, intuitively corresponding to the validity of all its steps and their proper binding. Now that many processes get implemented thanks to the cooperation of various and independent parties, relying on a single trusted party to guarantee the authenticity of these processes is, in our opinion, no more possible or suitable. Conversely, DL systems—with their tracking, linking, and tamper-proof features—represent a natural choice to enforce the above notion. However, our experiences in developing decentralized applications for different use cases show that reaching this goal is a task whose feasibility and difficulty strongly depend on the application scenario and the specific process considered.

2.1. Permissionless versus Permissioned Systems

A significant divide exists between the cryptocurrency realm and the world of regulated business. It stems from two alternative business models, gives rise to different and somewhat antithetical DLT requirements, and turns out in two opposite approaches to decentralization: permissionless versus permissioned systems. In a *permissionless* system, virtually anyone can participate in managing the ledger: an individual, usually anonymously, only needs to get the right software and possibly be enrolled in the community. On the contrary, only a set of pre-authorized and well-identified parties can participate in building a *permissioned* system; this set usually comprises representatives from companies and organizations that are in charge of managing the ledger and, conversely to in the permissionless case, it is somewhat limited in size, and stable over time.

In the vast and heterogeneous amount of blockchain-related literature, the above dichotomy is often confused with another view to classifying DL systems, which instead refers to the way users can interact with the ledger. It is indeed also helpful to distinguish between *public* and *private* systems, depending on whether reading access to the contents of the ledger (or submitting transactions) is allowed to anyone or only to groups of authorized users. This circumstance is unrelated and complementary to the previous one, and in fact, there are examples of blockchains for each of the four possible combinations of the two classifications. For example, regardless of the public, permissionless Ethereum main network (https://ethereum.org, accessed on 15 October 2021), everyone can deploy an Ethereum-based private chain (for a tutorial on this topic, see [26]), whereas Hyperledger Fabric (https://www.hyperledger.org/use/fabric, accessed on 15 October 2021) is an example of a permissioned framework for developing applications both in the private and public models. Hybrid cases for each of the two classifications are also possible. For example, some decentralized applications manage certain information flows confidentially thanks to the use of access control policies and encryption. However, they run on public and permissionless systems. Furthermore, there are hybrid systems that combine a permissioned DL with a permissionless one. For example, Medicalchain (https://medicalchain.com, accessed on 15 October 2021) uses a dual blockchain structure: the first blockchain controls access to health records using Hyperledger Fabric; the second uses an ERC20 token on Ethereum and underlies all the applications and services for the platform [27].

From a technical perspective, the private/public classification only affects the access control policies for end-users, while the permissionless versus permissioned distinction has profound implications on system management.

In permissionless DLT-based systems, the system's existence is voluntary and not predetermined: peers freely decide if and when to be involved in the ledger management. Because of this, participants receive adequate incentives; otherwise, the network would not survive. This critical point represents the strength of permissionless systems, as incentives encourage communities of participants to cooperate. Nevertheless, the prospect of easy incentives would lead users to misbehave and try multiplying their benefits. In particular, in a similar way to *Sybil attacks* [28] that can plague reputation systems, the same user could hide behind a multiplicity of anonymous identities to be able to grab a more significant number of benefits or to increase the probability of accessing benefits proportionally. Therefore, participation in the management of the ledger must occur at the price of not easily reproducible consumption of a resource and a specific cost, unlike standard digital data.

Bitcoin represents the first practical solution to the above "catch-22" dilemma: in this system, a cryptographic problem makes a user unlikely to be able to find data capable of allowing the addition of a new block, and thanks to the execution of a computationally burdensome process, called *proof-of-work*, that probability can proportionally increase. To compensate participants for their investments in resources and computation time, those who solve the current proof-of-work first receive a reward in the form of a certain number of bitcoins. With its related consensus protocol and the underlying incentive mechanism, the proof-of-work has proven to be very effective in preventing both Sybil attacks and another category of very relevant and specific cryptocurrency attacks, namely *double spending* [29].

However, this comes at the price of an underperforming and highly inefficient system, whose energy demand by design grows exponentially with the increase in the number of *miners*, i.e., those who participate in the proof-of-work. In Bitcoin, a miner generates a new block in about 10 min, and a user can safely consider a transaction permanently registered on the ledger only after an hour. Consequently, according to CBECI [30], the current estimate for the annual consumption of the global Bitcoin network is around 68 TWh; that is, more than Austria in the same period. On the other hand, digital transactions relying on a centralized authority for verification take an order of seconds to be confirmed, and they have a negligible cost when compared to Bitcoin. As of 22 July 2021, the estimated average Bitcoin electric power consumption per transaction exceeds one million Visa transactions [31].

Regardless of the opposition of many to pure cryptocurrencies, for their speculative risks and illicit uses (e.g., money laundering, ransomware, covert cryptomining), we believe that the growing emergency due to climate change will convince many countries and companies to give up the use of systems such as the current Bitcoin, generally opting for more eco-friendly DLT. Episodes in the spirit of this trend have already occurred (e.g., [32,33]). After all, researchers in business and academia recognized the limits of Bitcoin's proof-of-work since the early years of its launch and made efforts to obtain more efficient and performing permissionless systems. This attitude turned out also in revamping research in consensus protocols based on quorum mechanisms, a field in the context of studies on distributed systems whose first results have occurred since the eighties and which in recent years have undergone substantial developments precisely concerning permissioned or hybrid type blockchains. In addition to the staunch supporters of permissionless systems, according to which these would be the only ones able to implement a true decentralization in internet services, some people believe that decentralization is possible at various degrees and that a consortium-type approach for the provision of digital services can offer various advantages in different contexts of use and application domains.

As shown in Section 3.3, following the studies of the last few years, we have several consensus protocols that offer various trade-offs between scalability, energy efficiency, performance, and the threat assumptions under which they offer an adequate level of

reliability. In particular, thanks to the choice of a consensus protocol, it is possible to implement a certain level and type of decentralization according to the usage scenario, both in the permissioned and permissionless model. However, we must emphasize that a consensus algorithm could not be sufficient to implement the requirements of decentralization in a complete sense. First, there are blockchain networks whose functioning depends on data from off-chain sources: the way these systems receive and manage data can significantly affect decentralization, and it is helpful to discuss how in these systems we can preserve decentralization also thanks to cryptography, which is one of the subjects of Section 4. Moreover, at least according to techno-utopians who advocate for permissionless systems, decentralization should be inspired by the principles of freedom, pluralism, and balance of power. Therefore, to be fully decentralized, a network must adopt an adequate governance model: to what extent can a network be considered truly representative of all its participants if a few only decide its constitution and evolution? These aspects transcend the fields of cryptography and, more generally, those technological; however, they are crucial and often underestimated in the current literature. Therefore we will mention them in the following subsection.

2.2. Governance Models

Since the inception of Bitcoin by Satoshi Nakamoto, the concept of *governance* has interested actual and potential stakeholders. Who governs Bitcoin, who is accountable in the case of problems? These are a few of the many questions users have tried to answer before pouring money into a new digital currency system.

Actually, Nakamoto designed Bitcoin with the idea of leaving it working without any additional governance apart from the one embedded in the protocol. We do not even know who Nakamoto is because no one was intended to be the governor of the currency, depicting Bitcoin as a gift of a talented person to the human community.

The first attempts to introduce some form of governance were linked to adopting new features and overcoming protocol issues. Currently, a group of developers who volunteered to deal with the limits imposed by the original protocol maintains a reference implementation of Bitcoin. They declare to follow the consensus changes within the community rather than impose them. However, while leading the code distribution, they are in a prominent position to control opinions and deployments in the Bitcoin core system, as other influential players do. We can say that governance still resides in the protocol and the consensus mechanism, but with a sort of external lobbying. Meetings of relevant people (e.g., miners, developers, investors) can lead to agreements on protocol changes and eventually to fork.

Nevertheless, subsequent blockchain and DLT initiatives stayed at a distance from the original idea of a self-consistent currency framework, and faced governance issues, often with fragmentary, individual, and sometimes disordered approaches. Using blockchain technology in applications other than a cryptocurrency implied some redesign, introducing new concepts and tools. The transition from a gift of an anonymous donor to an open-source project automatically introduced some off-chain governance concepts, at least considering the authority to propose protocol changes.

From these few historical reasoning, we understand that talking about blockchain governance means two different concepts, which authors in the literature refer to as on-chain or governance *by* the blockchain and off-chain or governance *of* the blockchain.

2.2.1. On-Chain Governance

One aspect that blockchain enthusiasts embrace the most is the trustlessness of the systems. Before Bitcoin, any system was administered by an authority that users trusted for its correct functioning. Trustless systems overcome this limit by providing a protocol involving users through incentives to maintain integrity and functionalities. We can say that trustless systems do not eliminate trust. Instead, they require trust in the protocols that inherently incentives correct behaviors. This aspect strictly links to the consensus

mechanism. In Bitcoin, it relies on the economic advantage to keep the system working as expected. In other experiences, more democratic approaches emerged.

The developers of EOS [34] tried to introduce a voting system through delegated proof-of-stake (DPoS). Token holders vote to choose block producers, with the underlying idea of preventing actors from controlling multiple nodes. In a world of sincere voters, this would be a good approach. However, criticisms pointed out that corruption is possible, as is cartel forming. If a consistent number of tokens are controllable by a few actors, the voting role within the system integrity mechanisms is compromised, so does the trust in the system.

Tezos [35] presents similar criticisms. By using a peculiar proof-of-stake, this platform adapts to the needs of stakeholders by issuing a voting process over an election cycle lasting a certain number of blocks and subdivided into four quarters. In each quarter, proposed amendments of the protocol are subject to a voting stage and, if passed, to promotion in the next quarter, until final adoption:

- initially, an approval voting is issued to accept proposed tarballs containing new protocols. Each proposal collects some preferences;
- stakeholders vote for the most preferred protocol again counting votes for, against, and abstained;
- if a certain quorum of votes is reached, including those explicitly abstained, with a minimum approval rate of 80%, the new protocol goes into the test chain. In each cycle, the system automatically updates the quorum to avoid lost coins;
- stakeholders vote again to adopt the tested protocol in the main chain. Furthermore, in this case, quorum and 80% of preferences are the minimums to pass.

The most pivotal point of this mechanism is the ability to completely change the underlying protocol of stakeholders' preferences, exposing all the criticalities of direct democracy. As an extreme example, a misinformed majority could progressively bring the system to a catastrophic failure, even if democratically chosen.

An even more ambitious vision constitutes the building block of internet computer governance [36]. Based on DFINITY [37] mechanism, a Network Nervous System (NNS) governs the system through a liquid democracy [38] paradigm. The fundamental element is called *neuron* and consists of a stake deposited and time-locked that enables the user to vote on several proposal types, such as economics, policy, protocol, and client upgrades. As an incentive to vote responsibly, the stake gets locked for months following a request to dissolve the neuron. An innovative feature allows stakeholders to delegate votes on specific matters to proxies: by following influencers on social networks, a stakeholder can get acquainted with opinions about complicated matters and explicitly delegate votes to the ID of the trusted influencer. Unlike what happens in representative democracy, where a prefixed number of representatives confront and decide on the majority about discussed proposals, in this liquid democracy system, a charismatic figure can collect enough followers to freely propose and grant critical changes on essential thematics, such as protocol or economics.

The last tentative of on-chain governance that we present is Decred [39]. This system uses a time-locked voting ticket purchasing to enable stakeholders to vote on governance issues and within the consensus mechanism. Through the *Politeia* web platform, community members browse, discuss, and submit proposals about every aspect of project governance, including development funding. Every proposal is subject to an off-chain voting procedure that involves ticket purchasing by time-locking a number of coins. Even if the procedure does not get recorded on-chain, a timestamping mechanism ensures the immutability of the records.

Stakeholders can also vote on consensus mechanism changes on-chain, adopting a time-locking funds procedure to obtain voting tickets. A significant majority of actors involved in the consensus mechanism must first adopt a proposed change before submitting it to voting. If a proposal reaches a 75% threshold of approval during the voting period, it replaces the previous mechanism after a fixed number of blocks by issuing a hard fork.

A last similar voting mechanism is part of the consensus protocol, where five randomly chosen ticket holders vote to approve the proof-of-work block creation by the current mining winner. If a block gets three approvals, the block definitely goes in the chain. The involved ticket holders get 30% of the transaction fees as a reward.

In general, Decred, similar to other direct democracy systems, is subject to the issues mentioned above, which can lead to disastrous outcomes.

2.2.2. Off-Chain Governance

When endowing a blockchain project with time or money, an investor is concerned about governance. The underlying software protocol is just as important as the laws and regulations of a traditional business. Organizations investigating the adoption of blockchain in their asset management usually take governance factors into account and considering the high number of variables involved, the individuation of critical aspects is often challenging. In [40], the authors provide an interesting framework to help identify the peculiarities of specific blockchain governance features. The analyzer can subdivide their interest range into three layers: off-chain community, off-chain development, on-chain protocol. For each of these layers, the authors isolate five factors, called *dimensions*, to help the analyzer make the right questions: roles, incentives, membership, communication, and decision-making. Interpreting as many aspects as possible for each factor can help the investor understand how a specific blockchain initiative fits their intentions.

As a case study, the authors present the application of their framework to the Ethereum blockchain. For example, the analysis of the off-chain development layer include:

- *Roles*: Contributors, Maintainers, Ethereum Improvement Proposal (EIP) editors.
- *Incentives*: Contributors expect potential value increase of Ether from working voluntarily, as well as fun and social recognition. The Ethereum Foundation (EF) pays some maintainers.
- *Membership*: Everybody is free to contribute. There is no formal selection procedure for maintainers or EIP editors. Usually, the most recognized contributors are called to relevant positions by EF.
- *Communication*: Contributors communicate via GitHub comments, meet-ups, events, and scheduled calls. Core developers' calls notes are published.
- *Decision Making*: Decisions happen during calls or through the EIP process.

We can quickly notice the analyzed layer weaknesses and strengths, even if some aspects, such as accountability, are still missing. For example, how does the EIP process work? Who is involved in the calls? The EIP process consists of filling a proper request by the EIP author, followed by a formal verification by the editor who can send back the proposal for completion, and a reviewing by client developers who decide to accept after an apposite EIP presentation during an *AllCoreDevs* call. Usually, the developers discuss the technical merits of the EIP, evaluate the impact on other clients, and coordinate the eventual implementation of the network update. Even if this seems a pretty technical process, it is rather political: in practice, the developers refuse an EIP if it is divisive and could lead to a network split. Such an attitude could inhibit the adoption of effective but unpopular measures and have a profound political value. This kind of impact on governance is difficult to highlight and requires a deep investigation.

Other than the above orienting frameworks, we must point out that off-chain governance uses a looser set of rules, procedures, and social conventions than a code-based system. Because of this less formal structure, it is more challenging to manage and monitor. As a result, users can more easily avoid those policies. On the other hand, off-chain governance has a high degree of malleability, allowing the system to respond swiftly and seamlessly to unanticipated events and rapidly adjust to changes in the environment. The vague definition of off-chain governance rules allows for the required flexibility to limit or broaden the reach of these rules on a case-by-case basis, while the rigidity of on-chain governance rules means that malevolent actors may use them to undermine the system or shape it to their benefit if there is a design fault.

In general, on-chain and off-chain governance may show complementary limits. Let us take chain rewrites as an example: in the case of The DAO, where a set of programming weaknesses led to a catastrophic transfer of 3.6 Million Ethers to a single account, the EF, with an off-chain ethical decision, created a hard fork moving that amount to a recovery address, and implicitly creating the Ethereum Classic blockchain where the on-chain code is law, and code vulnerabilities might be unethical but lead to valid transactions. To deal with such cases, Tezos and DFINITY introduced on-chain voting on rewrites to let the stakeholders revert malevolent transfers. It seems a promising approach to deal with situations, such as The DAO, but suffers in any case of the above-mentioned direct democracy risks. This case represents the limits of on-chain and off-chain governances well, which both tend to overcome national and international legislation by allowing individuals or communities to set impacting rules in grey areas.

3. DLT Reference Architecture and Building Blocks

Although DL systems vary significantly in their implementations, they share a typical reference architecture and some key concepts and technologies that give rise to their building blocks. We will use this reference architecture in the present section to overview the main characteristics of techniques and the cryptographic concepts and mechanisms that enforce such features. Moreover, we will use the scheme as a map to place concepts discussed in the following sections in the right place for the system. Figure 3 illustrates the four-layer architecture that virtually characterizes any DL system. The *application layer* defines the types of users involved in the systems and how they can interact with them, offering a suite of concepts and tools that can vary depending on the specific platform considered. The *ledger layer* encompasses all the data structures and mechanisms which give rise to the ledger. Some of them, such as linked lists and Merkle trees, make the ledger data structure, while others enforce consensus or keep track of objects and workflows managed at the application layer. The *consensus layer* concerns the components of the multiparty agreement protocol used to select a unique ledger among possible diverse instances, thus assuring its consistent state. Finally, the *network layer* has to do with how the peer-to-peer (P2P) network of nodes is built and shares information so that the ledger can be queried and managed, and a node can always get its updated state.

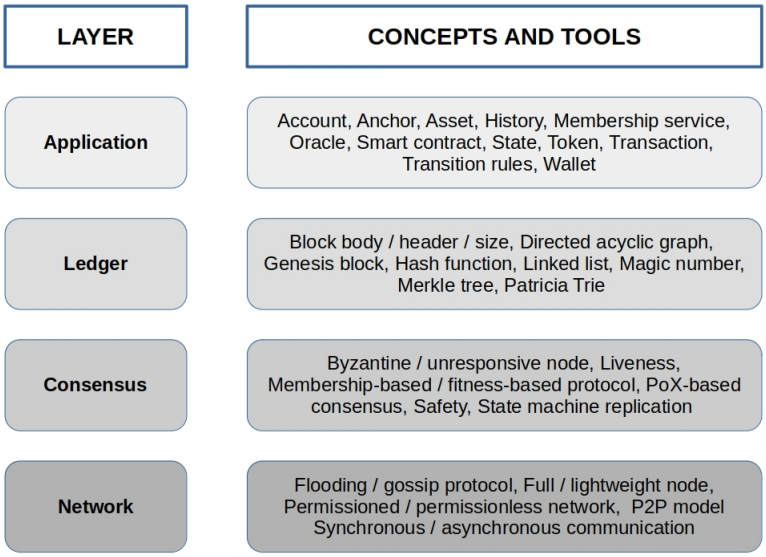

Figure 3. A DLT reference architecture.

3.1. Application

This layer is responsible for two primary tasks: user management and specification of the services that can be delivered by the system, possibly according to users' roles and attributes.

3.1.1. User Management

User management is very different depending on whether it is related to permissioned or permissionless systems: in permissionless systems, the difference between nodes composing the system network and client nodes lies in a grey area, whereas in permissioned systems, there is a clear distinction between clients and servers, and the latter can have different roles. In general, nodes that submit requests to register transactions on the ledger or access the ledger in reading mode are considered users or clients. In contrast, system nodes actively manage the network, representing the set of distributed computational resources that replicate the ledger and offer its related services to the clients. In permissionless systems, clients can opt to take an active role, and vice versa, so that a given node can act as a client, system node, or both, and this happens at the node owner's discretion and not in a predetermined manner, as encoded in some system account policy. Accordingly, there is no distinction between clients and blockchain nodes: they are considered blockchain nodes that can transact with the other nodes and participate in building the blockchain. In order to perform the above tasks, a blockchain node needs a unique identifying address, and this address must be such that another node can verify that the emitted and received data are genuinely coming from or destined to that address. A blockchain address, along with its balance in digital coins or some other kind of asset, is named an *account* or *wallet*: it represents how users are identified and managed in the system. Accounts can be implemented thanks to a digital signature scheme [41] by generating pairs of private-public keys: if a source uses a private key to sign a transaction, the corresponding public key is the one identifying the source address; analogously, if a transaction has a public key as the destination address, only the owner of the corresponding private key will be able to redeem it. In a permissionless system, private-public key pairs for a given node are generated independently by the node owner, without any support by a Certification Authority (CA) [42]. Therefore no connection is attested between the identity of the node owner and such keys. Some systems, such as Bitcoin, add further protection to users' anonymity thanks to a transaction model that resembles cash transfers, which will be touched upon shortly. Things are very different in the case of permissioned systems, for which multiple CAs can create a Public Key Infrastructure (PKI) [42] architecture capable of reflecting the different organizations that belong to the network and the roles of their units, both in terms of departments, blockchain nodes, and personnel. An example in this regard is Hyperledger Fabric, where a component called Membership Service Provider (MSP) [43] is designed to abstract away all cryptographic mechanisms and protocols behind issuing certificates, validating certificates, and user authentication.

We point out that the permissionless and permissioned approaches represent two radically different ways of implementing decentralization in digital services, each with pros and cons. The permissionless technology allows the creation of platforms to provide IT services as an alternative to cloud platforms. Similar to cloud platforms, permissionless DL systems are deployed and used globally. New generation systems, such as Ethereum, can provide computing infrastructures and software services in analogy with the *IaaS* and *SaaS* models [44]. However, unlike cloud platforms, they are collaborative platforms that aim to achieve massive decentralization thanks to the involvement of users, openly and anonymously. This approach avoids concerns and security issues related to unreliable or unqualified cloud service providers. However, it comes at the price of minor performance and greater complexity in system architecture and management, which could give rise to security issues. As pointed out in Section 2.2, permissionless systems are not trustless: they simply shift trust from a central authority to a suite of protocols and the community in charge of implementing and managing them. Instead, we can consider permissioned technology an evolution of cloud technology: it allows to decentralize trust in *multi-tenant* SaaS [45] environments hosted on clouds. A new category of cloud computing services, known as *blockchain-as-a service* (BaaS), has emerged in the last few years. Many cloud service providers offer BaaS turnkey solutions running on their cloud infrastructures (e.g., AWS, IBM, Microsoft, and Oracle).

3.1.2. Service Specification

At the core of service specification for DL systems, there is the notion of *transaction*. It indeed represents the atomic operation to be tracked in the ledger and the only way to trigger a change in the system; in some platforms (e.g., Hyperledger Fabric), transactions represent the only way to interact with the ledger since they are required not only to write data on it but also to read data from it. In a system backed by a cryptocurrency, transactions are first and foremost static data records tracking digital coin transfers between senders and receivers. In Bitcoin, for example, a transaction specifies a list of inputs and a list of outputs where each input encodes a value in bitcoins (BTC) representing an unspent transaction output (UTXO) for the sender, and each output encodes a value in BTC along with the receiver address, so that the sum of the input values is not less than that of the output values. In order to get a valid transaction, the sender has to sign each transaction input properly to redeem its value and be able to use it in an output, possibly summed to other input values. If output values result in a smaller sum than the inputs, the system adds a UTXO for the sender to the transaction to equalize the difference. For example, Alice could redeem 0.05 BTC and 1.181 BTC from two of her UTXOs to transfer 0.67 BTC and 0.63 BTC to Bob and Eve, respectively, getting back a new UTXO of 0.381 BTC by the system.

All decentralized applications rely on state, transition rules, and history, but the implementation of these notions can significantly vary depending on the particular system. The *state* is data currently stored in the ledger that, along with the *transition rules*, determines the following (finite) set of possible valid states. When a transaction is a request for a change in the ledger's state, it will be considered valid or invalid by the system according to its current state and its transition rules. Finally, the *history* is the set of all previous transactions and the order in which they occurred in the ledger.

In the Bitcoin system, the state is the set of all UTXOs with their addresses at a specific time. For each transaction, its encoded program instructions check if the transaction inputs correspond to previous UTXOs for the sender, which she cryptographically signed for redemption. If the answer is yes, the transaction is registered into the ledger, indicating that its output values are UTXOs for the many related output addresses.

The UTXO model allows for the simple and effective management of transfers of fungible assets, such as money; however, it does not explicitly associate ownership of assets with users, as required by many use cases other than cryptocurrencies. Therefore, more recent systems do not adopt the UTXO model; but instead, they have an explicit notion of *account balance* used to keep track of all the cryptocurrency or digital tokens owned by a user. Some systems (e.g., Hyperledger Fabric) further extend this functionality by allowing users to own generic digital assets not necessarily linked to a cryptocurrency.

Already in Bitcoin, the program instructions are written in the scripting language Script, which for each transaction redeem its inputs and make them available for its outputs, represent a first and straightforward form of transition rules (see [46] for a detailed description of their implementation). Subsequent systems have expanded the programmable logic related to transactions up to the point of allowing the creation of complex programs for the exchange of digital assets of a different nature or for triggering actions in the physical world. Digital artworks, information, goods or services, funds or rules involving events, or other transactions can be managed according to specific contractual requirements, thanks to programs that execute predefined actions based on the state of the system and the rights and obligations of those who participate in bargaining. For this reason, such programs are called *smart contracts*: they indeed represent rules resulting from agreements between the participants in a decentralized network, published and managed through the network. For efficiency reasons, smart contracts are not stored on the ledger since they may result in programs of considerable length. However, systems supporting them provide mechanisms also relying on the ledger to guarantee their authenticity, integrity, and public verifiability, as it happens, for example, for Ethereum smart contract Application Binary Interface (ABI) [47] or Hyperledger Fabric chaincode lifecycle [48]. In these systems, smart

contracts work like sets of transaction rules, and they represent the backend code of decentralized applications. Parallel to the extended notion of transaction, blockchain systems of this type support different kinds of assets (e.g., Ethereum tokens), plus tools designed to uniquely and authentically associate a digital asset with a physical good (anchors) or to ascertain the authenticity of external data or processes (oracles). Indeed, the shift from cryptocurrencies to more general assets urge for developing strategies and tools that avoid the recording of flawed or corrupted data on a blockchain, where they would just become immutable garbage. We will explore these emerging topics in Section 4.

3.2. Ledger

Introduced in Bitcoin [49], the blockchain is a tamper-proof, append-only data structure consisting of a linked list of transaction blocks. The key concept at the basis of the blockchain data structure is a *collision-resistant hash function* [50]. Haber and Stornetta, in 1990 [51], introduced lists of messages linked by (including in the current message) the hash digest of the previous one. The authors consider the linkage of digital certificates issued by a timestamping service as a countermeasure to back-date or forward-date a digital document by the owner, even with a colluding service. In Bitcoin, however, the hash function is used in the context of *Merkle trees*, a kind of binary tree for efficiently checking the integrity or authenticity of a set of data [52]. Transactions get grouped in ordered blocks, and each transaction in a block is associated with its hash digest, thus resulting in the tree's leaves. Left and right hash strings are then recursively joined and hashed again to get the digest value for their parent node in the tree until the tree root (Figure 4, sketch (a)). This way, the hash value associated with the Merkle root uniquely represents the ordered block of transactions with overwhelming probability. Such string is then inserted, along with the hash digest of the header of the previous block, in the header of the current block for reference to realize the blockchain data structure (Figure 4, sketch (b)).

Like a physical chain, a blockchain can be resistant to tampering but only provided that its links are sufficiently robust, i.e., difficult to replace with new ones. Assuming that an attacker cannot alter a given block (a digital signature on the block fulfills this assumption, giving rise to the linked list of digital certificates introduced in [51]), the collision-resistance property peculiar of the hash function makes it hard to alter any other previous block in the chain. However, since hash functions are efficiently computable, a given blockchain could be built from scratch starting from its initial (*genesis*) block, with any modification in its subsequent blocks as required by the attacker (*a long-range* attack). The critical idea in Bitcoin was to make the construction of a new block relatively tricky, thanks to a kind of cryptographic challenge introduced for anti-spam purposes in [53]. The resulting proof-of-work is a component of the Nakamoto consensus, and as such, we will briefly discuss it in Section 3.3.

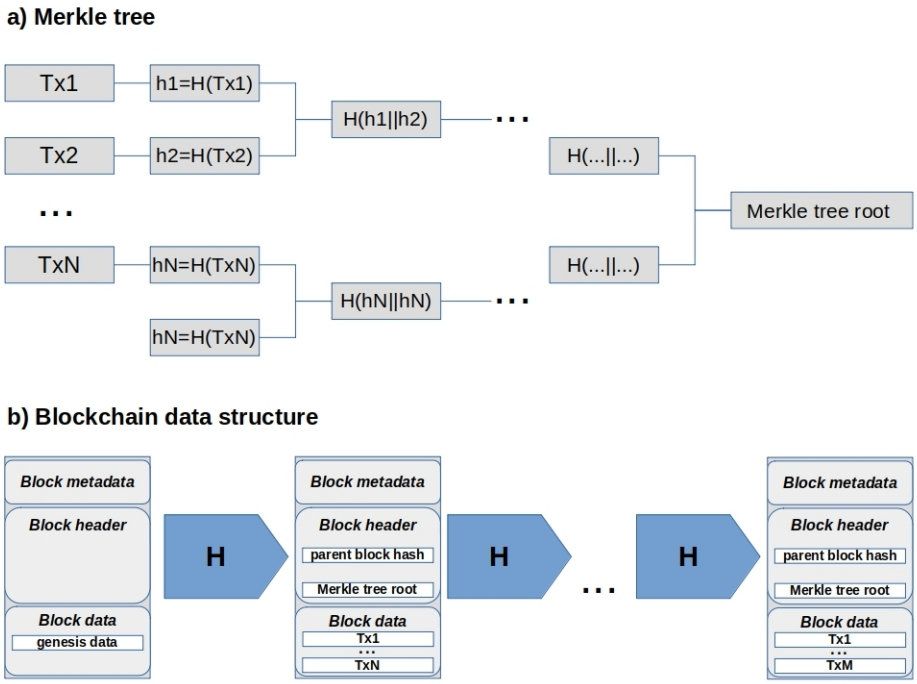

Figure 4. (**a**) Merkle tree construction for an odd number N of transactions, in which case the last hash digest gets duplicated to get a balanced tree, and; (**b**) The linked list of blocks starting from the genesis block, which results in the blockchain data structure. In both cases, H denotes a collision-resistant hash function.

3.2.1. Block Structure

Although sketch (b) in Figure 4 is a reference model for all blockchain-based systems, the exact structure of a block can vary considerably from system to system, according to the functionalities and workflows of the application layer and the (one or more) consensus protocol supported. For example, the block header in Hyperledger Fabric contains just three records storing the block number, the parent block hash, and the current block hash, which is the Merkle tree root of the transactions recorded in the block data. In turn, each transaction is composed of its header, signature, proposal, response, and endorsements fields. Things are very different in Ethereum, where a block's header is a data structure consisting of 14 fields, and a transaction consists of 7 fields. We illustrate the differences between these two systems in Table 1 for block headers and in Table 2 for transactions. These differences ultimately reflect the different nature of permissionless and permissioned systems concerning the management of decentralized computations.

Hyperledger Fabric is a consortium-oriented system whose primary goal is to support the reliable execution of business processes involving a small number of collaborating parties, among which there are no implicit relationships of trust. In this platform, there is a clear distinction among who is in charge of controlling the correct execution of the business process and who has to build the blockchain accordingly. An *endorsement policy*, a boolean expression defined to fulfill the use case-specific trust requirements, encodes the identities of a set of nodes called *endorsers* and their respective approvals or disapproval contribution to the final decision when checking a transaction proposal. Endorsement is also at the basis of the collaborative development and management of *chaincode*, the program in the form of transactions to be registered on the ledger, logically organized in smart contracts. Instead, a separate set of nodes called *orderers* is in charge of ordering in blocks the endorsed transactions and registering such blocks in the ledger. Unlike endorsers, orderers use an appropriate consensus protocol to agree with a majority on transactions ordering within a block and blocks ordering within the blockchain. In both cases, however, the decision of a node is expressed thanks to its digital signature, each final

decision involves a limited number of nodes, and final decisions are reached through the definitive collection of their respective preferences and cannot change at later times. This design turns out in a simple block header without all the fields concerning the consensus protocol, whereas a transaction provides the fields to enforce the endorsement policy. The advantages of the "endorser-orderer" approach compared to those traditionally adopted in blockchain systems are discussed in Section 4.

Table 1. Fields composing the block header in Hyperledger Fabric (HLF) and Ethereum (ETH).

HLF Field	**Description**
Block number	An integer starting at 0 (the genesis block), and increased by 1 for every new block appended to the blockchain.
Current block hash	The Merkle tree root of all the transactions contained in the current block.
Previous block hash	The hash digest from the previous block header.
ETH Field	**Description**
Number	Number of ancestor blocks (genesis block = 0).
Hash	Hash digest, which combined with Nonce proves that the PoW has been carried out for this block.
Parent hash	The hash digest from the previous block header.
Nonce	The solution to the PoW for this block.
Uncles hash	The hash digest of the list of *uncles* for this block, which are children of the parent block that are not parents of this block.
Logs bloom	Bloom filter of the indexable info contained in the log entry from the receipt of each transaction in the transaction list.
Transactions root	Hash digest of the Patricia trie built from the transaction list.
State root	Hash digest of the Patricia trie built from the *world state*, which maps addresses to account states.
Receipts root	Hash digest of the Patricia trie built from the receipts of the transactions in the transaction list.
Miner	Address for the fees collected because of successful mining.
Extra data	Arbitrary byte array (<32 byte) containing data relevant to this block.
Gas limit	Current limit of gas expenditure per block.
Gas used	Total gas used for the transactions in this block.
Timestamp	Unix time at the inception of this block.

Ethereum follows an entirely different approach since its permissionless nature does not distinguish nodes based on specific roles: all nodes are potentially in charge of validating transaction proposals and ordering valid transactions in blocks to be registered on the blockchain. Due to the vast number of nodes involved in the above task, the system needs implicit mechanisms to prevent denials of service by constraining the duration of each transaction and its repeated submission. The *gas* concept and its related fields in both the block header and transaction data structures account for the first constraint, while the Nonce field in each transaction serves to ward off reply attacks. Implicit agreement mechanisms are also required to get consensus: the present running version of Ethereum relies on the proof-of-work *Ethash* algorithm, so its block header comprises the fields required to implement it. In addition to the specific fields of the Ethash algorithm (e.g., Hash, Nonce, and Miner), the block header also provides the Uncles hash field to take into account the rule for consensus finalization, given that running Ethash can result in several simultaneously valid blocks (two more fields, Difficulty and Total difficulty, are provided in the Ethereum block data to account for Ethash and its finalization rule).

We must point out that the above macroscopic differences between Ethereum and Fabric blockchains also stem from two different approaches to modularity in software design. Fabric modularity is from scratch: consensus is implemented differently at the application and ledger layers, and we can opt among different possible protocols. Ethereum instead hard-codes consensus into incentives and computing resources, achieving modularity at the application layer through tokenization. As we will illustrate in Section 4, both platforms

represent the cornerstone of general-purpose computing frameworks for decentralized applications, although for very different trust management models.

Although consensus finalization concerns more appropriately one of the characteristics of the protocols presented in Section 3.3, we will briefly discuss the existing alternatives to select the "official" chain here, since this can affect the structure of a block, as shown by the previous example. Furthermore, it may be helpful to spend a few more words on this point as people often misunderstand it, and in some systems, such as Bitcoin and Ethereum, it concerns the significant notion of "blockchain state" introduced in the previous section.

In Nakamoto's paper [49], there is some ambiguity about the rule for finalizing a block, which is called the "longest chain rule" but it is explicitly referred to as the chain of blocks starting from the genesis block that "has the greatest proof-of-work effort invested in it". Following in a strict sense, the longest chain, as detailed, e.g., in [13], is not safe when the consensus procedure exploits a difficulty target, as in proof-of-work and proof-of-stake protocols (see next section), since longer chains may exist that contain far less mining work. For example, Ref. [54] describes two intriguing improvements to the *selfish mining* attack introduced in [55], which can be significantly advantageous for the attacker in catching up with the main official chain. In these systems, the correct method for selecting the current valid chain amongst the possible forks deriving from multiple block proposals is to compute the cumulative amount of work required to produce the chain. This fix was implemented early on in Bitcoin, which consists of the *heaviest chain rule*.

Table 2. Fields composing a transaction in Hyperledger Fabric (HLF) and Ethereum (ETH).

HLF Field	**Description**
Header	Metadata about the transaction (e.g., name of the relevant *chaincode*, and its version).
Signature	A cryptographic signature by the client application is used to check if the transaction details have not been tampered with.
Proposal	Encodes the input parameters supplied by an application to the smart contract, which creates the proposed ledger update.
Response	The output of a smart contract that captures the world state before and after this transaction .
Endorsements	A list of signed transaction responses from each required *endorser* sufficient to satisfy the *endorsement policy*.
ETH Field	**Description**
Nonce	A sequence number of transactions from a given address, which is increased by one for each new transaction in order to prevent reply attacks.
Gas price	Price of Gas in Gwei (1 Gwei = 10^{-9} ether).
Gas limit	Limit of the amount of ETH the sender is willing to pay for the transaction.
Recipient	An external owned address (EOA) or a contract address, which is the destination of this transaction.
Value	The amount of ether/wei from the sender to the recipient. Value is used for both transfer money and contract execution.
Data	Data for activities, such as deployment or execution of a contract.
(v,r,s)	Components of the transaction signature by the sender with the Elliptic Curve Digital Signature Algorithm (ECDSA).

The *Greedy Heaviest Observed Subtree* (GHOST) finalization protocol was proposed in 2013 by Vitalik Buterin for Ethereum [56] to achieve a faster block proposal rate without incurring in too many *orphan* blocks, i.e., valid blocks that eventually got cast off as a heavier chain achieved dominance. By allowing new blocks to reference multiple predecessors, a parent, and zero or more *uncles*, GHOST includes orphan blocks in the calculation of which chain has the greatest cumulative difficulty; moreover, it establishes that the miners of its uncles get a fraction of the base reward for a new block. This last rule helps to get fairer management of the efforts by honest nodes, protecting them against nodes (or pools

thereof) that exploit their hashing power to create small blocks with only a fraction of the transactions heard over the network. The protocol currently implemented in Ethereum is not GHOST, as erroneously reported by many sources, but a variant that only implements the second rule: uncles do indeed receive a reward, but they do not count towards the total difficulty of a chain [57,58].

However, the Ethereum blockchain has all the fields required for the implementation of GHOST, and the choice not to implement it seems to be due only to the fact that the current protocol, although simpler, still allows for a three times transaction throughput (10–15 tx/s versus 3–5 tx/s) and a much shorter transaction confirmation mean time (about 1 min versus 10 min) when compared to Bitcoin, although the number of confirmations after which one can consider a transaction safely recorded into the ledger is six for Bitcoin but much more for Ethereum (according to [56], one should wait for at least 7 confirmations, but major exchanges wait for around 50 confirmations to consider an Ethereum transaction complete). Nevertheless, after all, the designers of Ethereum are confident in the next *Serenity* release for boosting efficiency and performance, as we are going to explain in Section 3.3.

3.2.2. Blockchain Alternatives

The systems after Bitcoin usually do not deviate from the linked list approach, at least logically (i.e., concerning the way they record and retrieve transaction history). Instead, they often rely on extended or complementary data structures for the blockchain to support more advanced functions or improve performance. For example, the Ethereum blockchain makes use of (modified) Patricia tries [59] instead of Merkle trees, while Hyperledger Fabric associates a key-value database, storing the current state of the ledger, to a linked list of transaction blocks very similar to Bitcoin.

However, critical insight is that the blockchain serial structure restricts the object of consensus, thus hindering transaction throughput. This property is particularly relevant in the permissionless setting, where the consensus is a large-scale burdensome process. Indeed, starting from 2015 [60], there has been increasing attention toward non-linear ledger structures to improve transaction throughput in permissionless systems. Ledgers with a directed acyclic graph (DAG) structure [61] have received the most attention since, in this case, the flux of incoming transaction proposals drives the ledger shape. IOTA (https://www.iota.org, accessed on 15 October 2021) and Dagcoin (https://dagcoin.org, accessed on 15 October 2021) are two relevant permissionless systems built around a DAG ledger.

DAG ledgers can be block-based or transaction-based. In the first case, every vertex in the graph contains a block of transactions assembled similarly to blockchains, while in the second case, transactions do not get bundled into blocks, but each corresponds to a graph node. In both cases, however, a node references one or more parent nodes identified by their hashes. The links among nodes form a DAG, which confirms earlier nodes and establishes their partial order. As new nodes get added, earlier nodes can receive more and more confirmations, and the users who include them as parents in the node they create get their related fees (if any). If an attacker tries to edit or delete a confirmed node in the graph, the links to all its children and any further descendants will break since the hash and signature of any node depend on its parents' hashes [62].

The rules governing how a node gets added to the graph can vary from system to system and result in its consensus protocol, which usually does not diverge far from Nakamoto's consensus except for the "longest chain rule" (e.g., [63,64]). However, these approaches are far from being consolidated, and they often are based on arguments for which there are insufficient experimental simulations and proofs, lacking ascertained validation by experts in the field. Moreover, it is unclear if these approaches offer real advantages compared to permissionless systems based on blockchain. For example, the *Tangle* design [65] provided in the first IOTA implementation has been recently superseded by a new one named *Coordicide* [66]. By the very admission of the IOTA designers [66],

one reason was that the "honest transaction majority condition" assumed in the previous approach (i.e., the majority of transactions always come from honest network participants) has to be realized at a large scale, resulting in a very inefficient and energy-consuming system. Otherwise, it would be easy for malicious agents to buy enough hashing power and overtake the network.

In the context of permissioned systems, Corda (https://www.corda.net, accessed on 15 October 2021) represents another relevant system that does not adopt the blockchain as its ledger structure. In a Corda network, nodes have long-term stable identities, which they can prove ownership of to others: this way, the network entry process rules out Sybil attacks and a consensus protocol, such as the proof-of-work, is no more necessary. It is not even necessary to group transactions in blocks, a choice in Bitcoin and other permissionless systems motivated by having to keep the number of new block proposals to reduce conflicting blocks and related forks in the blockchain. In Corda, a transaction is, in effect, represented by a graph encoding all its transitive dependencies, and each transaction gets presented to a *notary service* that performs a breadth-first search over the transaction graph, downloading any missing transactions into local storage from the counter party and validating them by checking their signatures. Corda allows for multiple notary services, possibly decentralized, over coalitions of different parties using a consensus protocol of their own choice. Since each notary service provides for transaction ordering and timestamping services only for a part of the overall transactions managed by the network, the Corda ledger results in a set of multiple DAGs [67]. We stress that the above structure reflects that the Corda design initially had to address the interoperability needs of regulated financial institutions. Now that Corda has been generalized to encompass a diversity of enterprise use case requirements, in the same spirit of Hyperledger Fabric, this design results in a network governance complexity that tends to grow quicker than in Fabric.

3.3. Consensus

Consensus in distributed computing systems refers to the process of achieving agreement about data that are managed and stored through a set of network participants so that such nodes share the same state as the correct one for the system at a given point in time. In the context of blockchain systems, the consensus has to guarantee that all the ledger replicas deployed through the system are the same (blockchain *consistency*), except possibly for a small number of terminal blocks still awaiting a final decision (*eventual consistency*). These consistent ledger replicas provide a certain degree of tolerance to unresponsive nodes and those that arbitrarily deviate from the application logic, thus representing a more resilient and secure alternative to the trusted third-party approach. Trust indeed gets split over multiple nodes, which must collude or get hacked simultaneously so that the system as a whole deviates from its correct functioning. In order for things to work, however, procedures that can filter out those nodes trying to undermine a correct agreement process are necessary, and this is precisely the goal of every consensus protocol.

A node can become unresponsive because of a network failure resulting in the disconnection of the node from the network, or as a consequence of a *crash fault*, where the node abruptly stops working without quickly resuming because of a power outage, a DoS attack, or an error in its software or hardware. Regardless of the nature of the fault that halted a node, a consensus protocol must terminate correctly as long as the percentage of failures does not exceed a certain threshold. As suggested by the intuition, a protocol for which consensus follows from a majority quorum, as it is common practice in the P2P model, can withstand a certain number of unresponsive nodes less than half of the total.

Much more challenging to manage than nodes subject to halt faults are those that try to hinder a correct consensus process thanks to subtle and illegal actions of various types, ranging from the tampering of existing transactions or the creation of false ones (as in double spending) to the modification of the protocol itself, through message manipulation and reordering. These actions may include not responding to messages sent by the other peers, thus encompassing halt failures.

In general, some consensus protocols tolerate unresponsive nodes but not malicious ones (e.g., Zab [68], Raft [69]), whereas a protocol designed to withstand malicious nodes will also tolerate unresponsive nodes to a greater extent. Depending on the blockchain consensus protocol, we can measure fault tolerance as a percentage of the number of malicious nodes or their used resources from the total number of nodes or resources involved in the consensus process. This measure reflects the first fundamental discrimination for blockchain consensus protocols, which we can divide into the two broad classes of *membership-based* procedures and *fitness-based* procedures, in addition to a further class of hybrid methods obtained by combining the two previous approaches.

In membership-based protocols, a predetermined and reduced set of nodes with explicit identities and roles, often coordinated by a leader (a.k.a. primary) node that acts as an interface with clients, forms a committee, reaching agreement through multiple rounds of message exchange to achieve a consensus. In this respect, the committee represents a server consortium that has to order transaction proposals as they arrive by clients, arranging them in a valid chain of blocks despite some servers failing. Accordingly, the correctness of consensus for these protocols can be formulated by recasting two concepts introduced in [70]: (i) every server correctly executes the same sequence of (transaction) requests (*safety*), and; (ii) all requests are served (*liveness*). In this context, we can call malicious nodes and faults "Byzantine", whereas consensus protocols that exhibit some degree of tolerance to them are called *Byzantine-fault tolerant* (BFT) protocols. This terminology has come into use after [71], in which the authors introduce the consensus problem making the analogy where a group of generals of the Byzantine army must agree on a common battle plan, communicating only through messengers and despite some traitors hiding among them generate conflicting messages to hamper agreement.

It is shown in [72] that BFT protocols cannot tolerate a critical number of Byzantine nodes greater than or equal to one-third of the total number of nodes involved in the server consortium. Another prominent limitation of these protocols, and more generally of membership-based protocols, is that they require explicit message passing among the N nodes composing the consortium, typically exhibiting a $O(N^2)$ message complexity, which turns out to be a considerable networking overhead already for consortia of a few tens of nodes. These protocols are, therefore, suitable only for permissioned blockchains and can offer a limited degree of decentralization. As for the bright side, membership-based protocols offer *consensus finality*, in that each consensus decision, as returned by the protocol, is conclusive and cannot be subject to changes or cancellation at later times. Thus, a block of transactions chosen to get transcribed on the blockchain can no longer change or get canceled later, allowing block intervals and transaction throughput comparable to those of centralized systems. Some protocols (e.g., PBFT [73] and its derivatives) that follow the *state machine replication* (SMR) scheme formalized in [74] fall into the BFT category, with other more recent protocols having different message workflows (e.g., HoneyBadgerBFT [75] and the related protocol suite BEAT [76]).

Fitness-based consensus protocols have a design suitable for large and open sets of participating nodes whose exact size and members are not known a priori. The crucial element underlying this broad category of protocols is the use of a resource, internal or external to the system, which serves to quantify the fitness of the nodes (i.e., their influence or power) in the consensus process. Such a resource is necessary for the context of permissionless networks, where an arbitrary (and anonymous) node has the opportunity to participate in the management of the blockchain. It indeed serves to constrain the node power to the quantity of the resource they wish or can invest in the consensus process, thus counteracting Sybil attacks and their derivatives.

Another central element that virtually distinguishes all protocols suitable for permissionless systems is the ability of the node to add a new block to prove publicly (i.e., to the rest of the network) that it is indeed enabled to do so based on its fitness. Accordingly, authors in the literature denote these protocols as *proof-based* consensus protocols.

Since these protocols operate on P2P networks, allowing the decentralized management of trust and services in a much broader and flexible way than in client-server models, they have been singularly studied in recent years precisely in conjunction with and about the development of DLT, of which they represent a core aspect. Indeed, since the activation of the Bitcoin network, research contributions have multiplied, with dozens of new protocol proposals that, in many cases, have been implemented in functioning systems. Due to their importance for DLT and their primary role in the enforcement of decentralization and such technologies, we will allude below to the main types of these protocols; the interested reader should refer to specific literature contributions for a more comprehensive and in-depth treatment of this topic (e.g., [13,77–79]).

Note that these protocols are categorized quite differently by different authors, depending on the feature or approach that, in their opinion, primarily distinguish or unite them. Indeed, these protocols often consist of more than one key element, and the choice of the one considered most relevant influences their classification. In what follows, we have chosen the classification that, in our opinion, best highlights the articulation of proposals in order to overcome security and performance issues concerning previous solutions. This way, we hope to shed some more light on the evolutionary trend of this crucial DLT research field.

Beyond the names assigned by their respective authors, often derived from the type of fitness resource that the protocol allows to verify publicly or "proof" (work, stake, activity), it is significant to note that researchers oriented towards designing more efficient and performing permissionless consensus protocols by recurring to approaches and tools borrowed from the membership-based protocols previously introduced. In this respect, the main distinction concerns the part of the network involved in creating the new blocks. In the Bitcoin protocol and the first alternatives based on crypto coins owned by users (*stakes*), the whole network is involved in this process. Indeed, in these protocols, any node in the network may check if it is fit to add the new current block, giving public evidence of that. As this approach has resulted in poorly performing and computationally costly protocols, many subsequent proposals have adopted a scheme in which a tiny subset of nodes are selected and allowed to add a new block. Then within this *committee*, the nodes that will add new blocks, also known as *leaders*, get selected. Accordingly, we will divide fitness-based consensus procedures first and foremost into *network-oriented* and *committee-oriented* protocols, leaving the later levels the distinctions based on the type of fitness resource used and, in the context of committee-oriented protocols, on how committees and leaders get selected.

Among the multitude of proposals concerning consensus protocols for DL systems, below, we will limit ourselves to discussing only those that, in our opinion, have introduced significant innovations, currently representing the milestones of the development of research on this topic. These protocols, along with their main features, are listed in Table 3.

3.3.1. Network-Oriented Consensus Protocols

These protocols do not rely on explicit message exchanges among a committee to establish consensus, but rather on an algorithm that allows nodes by themselves to determine if they may add the new current block of transactions, giving corroborated evidence of this to the other nodes thanks to a *proof* that can be easily and publicly verified. Typically, the algorithm involves solving a computational problem by the node, which can publish the solution found as proof that it is fit to insert the new block. In this regard, the solution to the problem must depend on the current transaction block, although not predictable, and it must not get more manageable knowing additional data, such as the blocks previously recorded on the blockchain. For these reasons, the algorithm makes use of a cryptographic primitive, which returns pseudo-random outputs (precisely, a cryptographic hash function). The choice of the computational problem to solve is critical: a too complex problem affects the liveness of the system, whereas a problem that is too easy to solve weakens its consistency. In fact, in the first case, it could be challenging to find a solver in a reasonable

time, while in the second, there could be many solvers and many potentially valid blocks to add to the ledger. Designers usually identify an adequate trade-off thanks to using a system resource for which the node must invest a specific effort and whose availability increases the probability of solving the problem in a relatively short period. This way, the amount of the resource mentioned above represents the *fitness* a node has in solving the problem. At the same time, its solution is a sort of ticket granting access to the final tender, which establishes who will be able to add the current block to the blockchain. Indeed, there can be multiple problem solvers that correspond to as many current blocks, all potentially valid, but which must get selected according to a system rule to obtain a single shared list of consecutive blocks. It follows that network-oriented consensus protocols consist of the three following phases, according to a timeline composed of time frames in each of which potentially all the nodes in the network compete for the addition of a new block in the blockchain:

- *Proposal*: where a node assembles a block of transactions and attaches proof to it demonstrating its investment of the fitness resource in such a task;
- *Validation*: in which nodes check the validity of the blocks proposed in the previous phase, alongside with their alleged proofs;
- *Finalization*: in which the network, following a rule usually based on the majority of participants, selects a unique block among those that passed the validation phase.

Table 3. A list of some major network-oriented (NET) and committee-oriented (COM) blockchain consensus protocols.

NET Protocol	**Proposer Selection**	**Validation Check**	**Finalization Rule**
PBFT	Client request	Signature	Mutual agreement
Nakamoto	PoW solver	PoW solution	Heaviest chain
Ethash + GHOST	PoW solver	PoW solution	GHOST variant
PoET	TEE waiting time	TEE attestation	Longest chain
Ethash + Casper FFG	PoW solver	PoW solution	Casper FFG
COM Protocol	**Proposer Selection**	**Validation Check**	**Finalization Rule**
Tendermint	Round robin	Stake, signature	Mutual agreement
Chain of Activity	FTS output	FTS output	Longest chain
Proof of Authority	Certification	Signature	Mutual agreement
Honey Badger BFT	Client request	Signature	Mutual agreement
Ouroboros	FTS and PVSS	PVSS output	Longest chain
Algorand	VRF + agreement	VRF, signature	Mutual agreement
Ouroboros Praos	VRF output	VRF output	Longest chain
Snow White	PRF (application)	PoS solution	Longest chain
Delegated PoS	Stake delegation	Delegate eligibility	Mutual agreement

Because of their implicit agreement model, network-oriented consensus protocols scale much better than membership-based protocols to the number of nodes involved: their message complexity can be indeed limited to $O(1)$ for each node by exploiting the communication models described in Section 3.4. However, this comes at the price of both higher costs and worse performance. Consensus does not conclude with the validation phase, but it requires an extra finalization process, which by its nature is not deterministic and could take time since it depends upon a large number of loosely coupled nodes. Therefore, it impacts negatively on the transaction throughput. Cost, on the other hand, does not depend so much on sending and receiving messages and on the local processing required about this, as in the membership-based protocols, but rather on the fact that it is necessary to counteract Sybil attacks and those related to them (e.g., double spending) thanks to considerable consumption of the fitness resource by the nodes. After more than a decade of research in the field, the actual cost of a proof-based protocol can be far lower than for the proof-of-work implemented in Bitcoin since it strongly depends on the choice and management of the fitness function. However, these costs are decidedly higher for

network-oriented consensus protocols than those of the membership-based protocols, if only because many more nodes are involved in the consensus process.

Falling in this category: protocols at the core of many well-established cryptocurrencies (e.g., Bitcoin [49], Ethereum [80], Peercoin [81], Nxt [82]), altcoin proposals never came to light as Permacoin [83], as well as systems not intrinsically tied to a cryptocurrency (e.g., Hyperledger Sawtooth [84]). The fitness resource used by the nodes in these systems is typically hashing power or stake value; however, other choices are also possible, as the amount of stored data in Permacoin or the disposal of a trusted execution environment (specifically, Intel SGX [85]) in Sawtooth.

Nakamoto consensus was the pioneering network-oriented protocol; it makes use of a proof-of-work (PoW) algorithm to accomplish the proposal and validation phases, and the so-called *longest chain rule* for finalization, already discussed in Section 3.2. PoW algorithms employ a cryptographic hash function to enforce a search problem called *mining*, where nodes have to find a string called *nonce* to get a hash digest whose value in hexadecimal has not less than a given number of leading zeros, known as the *difficulty target*. The hash function inputs the nonce and a digest resulting from the Merckle tree of the transactions in the current block and the hash digest of the previous block header so that the nodes can instantiate the search problem for the current block. In this way, they cannot circumvent or make it easier by prior computations, but they can only solve it by an iterated error-retry approach.

A *difficulty adjustment algorithm* tunes the difficulty target by computing at each mining node the time required by the network to generate a given number of previous blocks so that a solution is moderately hard to find by the totality of miners in a statistical sense. The goal is to balance the block delivery rate and the frequency of the occurrence of multiple valid blocks, which undermine blockchain consistency and burden the finalization phase. In some PoW implementations, the hashing algorithm can have some additional properties in order to limit substantial increases in hashing rate thanks to the use of specific hardware, such as the *ASIC resistance* provided by the *Ethash* algorithm in Ethereum, which also has other features according to the design rationale described in [86].

However, the real problem that plagues all PoW systems is that miners have to compete to solve a computational challenge within a preset time, resulting in a *nonce rush* that is more expensive the larger the network is [46]. Well before bitcoins became the best known and most used cryptocurrency globally, the Bitcoin developer community had glimpsed in *Proof-of-Stake* (PoS) algorithms alternatives with much lower and sustainable energy consumption. The term *stake* refers to digital coins or tokens owned by participants and managed through the blockchain network, used as a counterweight to hash power. The difficulty of mining activity is inversely proportional to the number of crypto coins owned by the miner at the time; this way, a virtual resource replaces physical ones (specifically computational devices and electric power).

A pure PoS-based system follows the same principles and construction as Nakamoto consensus, with the sole exception of the implementation of the hashing challenge: the calculation of the hash function can be performed only once for each unit of time (e.g., one second), according to a virtual shared timer realized by keeping synchronized within a given margin of tolerance the local clocks at each node, while the difficulty to solve the search problem is inversely proportional to the stake value invested by the node in the challenge. In this way, PoS avoids the brute-force hashing race characteristic of a PoW, but the significant savings in energy and hardware come at the price of worsening some issues (e.g., selfish mining and centralization) and a series of possible attacks to which PoWs are immune. Indeed, the public availability of staking history in the blockchain weakens the PoS pseudo-randomness while simultaneously incentivizing corruption: two factors exploited by the *stake-grinding* [87] and *bribing* [88] attacks, respectively. On the other hand, the virtual nature of mining paves the way for low-cost simulations: a malicious node may create an alternative branch to the main chain starting at any point of its choice without any substantial cost. That, in turn, makes it easier for users to overtake the main chain by

collusion as in the *long-range* attack or to bet in multiple competing chains simultaneously by fractionating their stakes as in the *nothing-at-stake* attack, which in both cases hinder the resiliency of the system to double spending [89]. This resiliency is theoretically equal to 50% of the fitness resource for both PoW and PoS systems: (pool of) attackers must have more than 50% hashing power or stake value to mount successful double spending attacks. However, malicious mining strategies, such as the selfish mining for Bitcoin [55], and those at the core of the two previous attacks, can substantially reduce that threshold: in [55], the authors prove that a colluding group of any size could eventually compromise the Bitcoin network under some circumstances, and a similar result presumably also applies to many other PoS and PoW systems.

By allowing a limited set of users to take control of the network, selfish mining and other illegal mining strategies ultimately threaten the decentralization aimed by permissionless blockchains. However, a centralization risk for proof-based consensus systems stems from the "wealthy get wealthier" paradigm: nodes having more fitness (hashing power or stake value) than others can lawfully exploit this advantage to accumulate more and more wealth over time, potentially reducing the system to an oligopoly. This attitude is particularly true for PoS systems, where miners can reinvest their profits into staking over and over again.

Next, [82] tries to mitigate some of the above issues thanks to the following rules: (i) designers determined token supply at the origin and the reward for inserting a new block only comes from transaction fees; (ii) users cannot split stakes over multiple simultaneous mining activities, and; (iii) the stake value is appreciated due to the mining activity during a block cycle, but it is reset to the base value once the cycle ends.

These and other similar rules can encourage users to participate and honestly validate transactions; however, they do not overcome the two primary limits of current network-oriented systems. The first limit concerns performance: the fact that the consensus process potentially involves all the network nodes entails a considerable time for the finalization phase. Even adopting particular strategies for determining the main chain, such as GHOST or the variant currently implemented in Ethereum (see Section 3.2), the times for confirming a transaction are orders of magnitude greater than for membership-based systems. The second limit concerns security: the arguments supporting the security of these systems are indeed only of a heuristic nature due to the lack of formal reference models and precise definitions. Therefore, these systems could prove to be vulnerable by design at a later time, with potential irremediable damage to their investors, which the use of a cryptocurrency can exacerbate. The case of selfish mining for Bitcoin is exemplary in this respect: this network has not yet been the subject of such an attack, presumably solely because its users believe that its longevity and reputation can safeguard their earnings.

Therefore, two somewhat complementary principles guided research efforts in recent years: (i) drawing inspiration from skills and methodologies acquired in the thirty-year development of membership-based consensus systems, and; (ii) exploiting formal mathematical proofs based on adequate cryptographic primitives to obtain provably secure consensus protocols.

These led to the protocols briefly discussed in the following section.

3.3.2. Committee-Oriented Consensus Protocols

These protocols combine the proof-based approach with the notion of "committee", already used in membership-based protocols, by adhering to the following high-level procedure, an alternative to the one presented in the previous section.

- *Committee selection*: to designate a (relatively small) subset of nodes in the network as eligible to participate in the consensus process for adding one or more upcoming new blocks;
- *Leader selection*: to designate a node in the committee as the one that is actually in charge of adding the next block or of leading the consensus process for adding the next block;

- *Proposal*: where the leader node returns the new current block along with a proof to demonstrate the block validity;
- *Validation*: in which nodes check the validity of the blocks proposed in the previous phase, alongside with their alleged proofs;
- *Finalization*: in which the network, following a rule, usually based on the majority of participants, selects a unique block among those that passed the validation phase.

It is essential to notice here that committee-oriented consensus protocols cannot correctly work without the finalization phase. Unlike in membership-based consensus, a committee might not always get uniquely defined for the entire network, and this could also happen, within a given committee, for the leaders appointed to add the new block. Indeed, in these protocols, the committee is selected starting from a piece of information shared between the nodes thanks to the blockchain to take into account the distribution of the fitness function between them. However, the "blockchain view" required by a given protocol may not be the same for all the nodes in the network because of blockchain forks due to communication delays or illegal behavior. In other words, these protocols can guarantee that each election gives rise to a single committee only with a specific (albeit high) probability, not absolutely. Similarly, depending on the selection method used for choosing a leader within a committee, this could result in more than one leader and as many respective potentially valid blocks.

In this category fall many protocols for which the committee selection works by taking into account the stake values of the participating nodes. Since the committee has to be chosen so that an attacker can hardly predict the participating nodes, to avoid being subject to corruption or boycott, committee members usually get selected by a pseudo-random process whose output can be publicly verified. Thus, most of these protocols rightfully belong to the PoS class (e.g., [90–93]), whereas few exceptions, such as *Tendermint* [94], cannot be adequately addressed as PoS systems, although they use the stake values to select the committee. Other committee-oriented systems consider fitness measures alternative to the distribution of stake values among participating nodes. An example is the *proof-of-authority* [95] by the co-founder of Ethereum, Gavin Wood, where committee members have to go through a mandatory certification process to build up their authority.

We can trace back the idea of selecting a committee to add a new block to Tendermint and the *proof-of-activity* protocol [88] in 2014, although, in these systems, the committee is only implicitly defined and dynamically configured for each new block. In Tendermint, the consensus process gets divided into time intervals called rounds, for each of which a chosen leader (also known as proposer) proposes the new block, starting a voting process involving the other nodes in the network through a BFT algorithm. The leader selection function is a weighted round-robin, where nodes rotate proportionally to their voting power, and the voting process can only return reliable outputs if less than one-third of all voters are dishonest, as follows from the tolerance limits for BFT protocols. This last condition is challenging to verify, given that the voting nodes are not defined a priori but depend on the expected times for carrying out the various phases of the BFT protocol. Furthermore, the deterministic nature of selecting leaders makes it necessary to protect their true identities by using proxy nodes that are not supposed to divulge those identities. Therefore Tendermint's consensus protocol appears unconvincing, despite the alleged proof of its security given in the more recent draft [96].

Other proposals inject randomness in selecting leaders (or committees) to mitigate corruption or boycott attacks.

The *chain-of-activity* (CoA) protocol [97] uses the pseudo-randomness tied to a sequence of previous blocks in the blockchain (e.g., by computing the hash digest for each block and taking the leading bit of each digest) to generate a shared sequence of N pseudo-random seeds. To protect against possible blockchain forks, each node following the protocol must consider a sequence of blocks such that the last one precedes a predetermined number of final blocks in the received chain. The N seeds are then iteratively passed

in input to the *follow-the-satoshi* (FTS) algorithm [88] to get the sequence of leaders responsible for building the following N blocks. The FTS algorithm treats the string received in input as an index of a previously coined satoshi, inspecting the blockchain to determine which user this satoshi belongs to at the current moment. In this way, the system selects the sequence of leaders, taking into account their respective stake values, but in a way that it is difficult for an attacker to determine their identities for just a short time interval before they contribute to the construction of the blockchain. The use of the FTS algorithm makes CoA much more resilient than Tendermint to corruption attacks. However, a powerful attacker could exploit the short time interval during which the identities of the following leaders get disclosed for an *adaptive corruption attack*, where the attacker promises rewards to some of the selected leaders if they opt for specific blocks.

Ouroboros [90], the consensus protocol exploited by the Cardano cryptocurrency (https://cardano.org/, accessed on 15 October 2021), follows a similar approach to that of CoA. In this case, a *publicly verifiable secret sharing* (PVSS) [98], where the participants in the consensus process and everybody in the network can verify the distribution of the shares, determines the sequence of shared seeds passed in input to the FTS algorithm.

PVSS is a particular case and a building block of the *multiparty computation* (MPC) [99] framework, concerned with enabling some different and untrusted parties to carry out a joint computation of a given function. MPC assumes a threat model very similar to that of BFT protocols, where some malicious entity, even in the form of a subset of participating nodes, tries to hamper the correct computation of the joint function to get a prescribed output or acquire some private information. Accordingly, two essential requirements of secure MPC protocols are *privacy* and *correctness*: the privacy requirement states that parties should learn the function output and nothing else, while the correctness requirement states that each party should receive its correct output.

The public use of the FTS algorithm allows Ouroboros to implement the privacy requirement only at a time before the actual work of the new committee of leader nodes, leaving this system as CoA unprotected to adaptive corruption attacks. Moreover, by requiring strong network synchrony for leaders to use precisely their time slots, Ouroboros is vulnerable to *desincronization attacks*.

On the contrary, *Snow White* [93] works in *partially synchronous* networks (see Section 3.4), and, thanks to the *sleepy* [100] consensus model, it can also accommodate sporadic participation in which nodes can arbitrarily switch from an online to offline status. However, Snow White is not able to completely counter adaptive corruption attacks. Snow white first determines the composition of the committee of nodes allowed to add new blocks during a time-lapse known as an *epoch*, thanks to a fitness function defined at the application layer and to the fitness status of the nodes as deduced by inspecting a blockchain prefix. Then, within each epoch ϵ, it selects the nodes in charge of adding the new block at time $\tau \in \epsilon$ as the solvers of the hashing challenge $H_\epsilon(\tau, pk) < D$, where H_ϵ is a hash function that depends upon the epoch, pk is the node public key, and D determines the challenge difficulty. This way, just before a new epoch begins, an attacker can determine which nodes in the committee will have to insert the new blocks relating to the time slices for that epoch.

Ouroboros Praos [92] is a protocol that, thanks to an analysis of Ouroboros, overcomes its two above issues and requires far fewer message exchanges between nodes to accomplish the MPC procedure. First, each elector executes a *verifiable random function* (VRF) [101] locally to define its proposal for the following sequence of leaders, so that each other node can verify if it got chosen as a leader with respective slots, without being able to determine the other leaders in the sequence. This mechanism turns out in a secure MPC procedure with the requisites of privacy and correctness, assuming that the majority of the stake value belongs to honest participants. Second, some empty time slots up to a maximum value are allowed for each epoch to help electors resynchronize when necessary, thus resulting in a protocol designed for partially synchronous networks.

Like Ouroboros Praos, *Algorand* [91] is another provably secure PoS protocol designed for networks that are not firmly synchronous and where attackers can also perform adaptive

corruption. Algorand too uses a VRF scheme to select the committee randomly, although taking into account the stake values of participating nodes, called cryptographic *sortition*. However, in this case, it is the committee as a whole to decide which block to add thanks to an appropriate BFT algorithm, not a single leader, which turns out in a protocol reaching consensus finality in a slightly stronger assumption than partial synchrony, known as *weak synchrony*. This way, more resilience to corruption attacks is gained at the price of reliance on network synchrony.

Two other relevant PoS protocols falling in the committee-oriented class are the *delegated proof-of-stake* (DPoS) used in different systems (e.g., *bitshares* [102] and *EOSIO* [103]) and *Casper FFG* [104], which will appear in the upcoming version 2.0 of Ethereum, also known as *Serenity*. DPoS protocols were already discussed in Section 2.2, as they are generally used to define the system's governance. Casper FFG is an addition to the Ethash PoW, which incorporates PoS into block finalization. Ethereum developers intended to change the variant of GHOST currently implemented with a protocol where a set of validators use a BFT algorithm to select the new block, guaranteeing backward compatibility. Thanks to this new design, and the combined use of *sharding techniques* [105], Ethereum should significantly increase its performance, going from the current tens to thousands of transactions per second [13].

3.4. Network

The network layer is the foundational infrastructure of blockchain systems, which establishes the role and functions of its nodes, the way they communicate, and how application workloads get partitioned among them. This layer also serves to specify the modality of the participation of nodes (continuous, sporadic) and their ability to synchronize each other through the exchange of messages. In particular, this layer concerns the synchronization assumption underlying the design and usage of a specific consensus protocol by indicating the way and amount of delays affecting message transmissions. We can also consider other characteristics for qualifying or quantifying the reliability of communications, and therefore also concerning the threat model considered, implicitly defining the capabilities of attackers to hinder the consistency and finality of consensus processes.

3.4.1. Network Architecture

Permissionless blockchains adopt a pure peer-to-peer (P2P) architecture [106]. Every node can be both supplier and consumer of resources, and it is an equally privileged and equipotent participant in the application that can freely join and leave the network without any authorization. The only relevant difference is between *full nodes* and *lightweight nodes*: the first ones maintain a full copy of the ledger, while the second ones download just a subset of it so to verify if one or more transactions have been included in a block, possibly thanks to the support of a full node.

Permissioned blockchains instead adopt a hybrid model, where nodes must be authorized first and then participate with a revealed identity, and often with a specific role and functions. Conversely to permissionless systems, in this second type of blockchain network, a set of client nodes can generally be distinguished from a server component made up of a multiplicity of nodes divided according to their specific service function.

In both cases, every node (or peer) in the network operates autonomously for a given set of rules encompassing the communication protocol and other functionalities, such as transaction processing, consensus participation, and ledger management. These last possibly depend on its role.

3.4.2. Communication Models

From the communication viewpoint, peers implement some form of virtual overlay network on top of the Internet, which allows them to exchange information directly through indexing and neighbor node discovery, regardless of the physical network topology. In

order to optimize performance and scalability, peers connect only to a limited number of their neighbors through a *flooding* or *gossip* protocol [11]. In flooding protocols, messages get transmitted to all the nodes recognized as neighbors, while gossip protocols relay messages only to a subset of randomly selected neighbor nodes. Fast and pervasive spreading of transactions over the network is a crucial factor for effective management of the ledger status through the consensus protocol, and both approaches are appropriate, although with different transaction bandwidth and delay performance. As a rule of thumb, flooding protocols are more bandwidth-intensive but less prone to network splitting attacks than gossip protocols because of the more connections involved in transaction propagation. Typically, however, the relative importance of bandwidth and delay of messages depends on the type of traffic workloads; moreover, bandwidth gains are at the expense of delayed transmissions, as in the case of the announce-and-request signaling implemented in the Bitcoin flooding protocol, which requires two more communication steps. Therefore, different systems often implement different protocols, and the choice should be inclined to obtain the best performance compromise according to the type of network and the functions offered at the application level. In general, transaction flooding best fits in permissionless systems because of its major pervasiveness and resistance to network splitting attacks. In contrast, message dissemination through gossiping goes well for permissioned networks thanks to minor bandwidth requirements and a faster reaction in the case of faulty nodes or slow network links. Ethereum tries to take the best of both approaches by adopting a hybrid protocol, in which some messages are sent to all neighboring peers while others go to a limited number of them.

Transaction propagation in blockchain networks has to also deal with privacy and security issues. Transaction payloads can be encrypted depending on whether the blockchain is private or public and the nature of communications among subsets of peers in the network. Nevertheless, in any case, peers must exchange messages with confidence about the authenticity of senders and receivers and the integrity of data payloads. Moreover, depending on the implemented P2P protocol, permissionless networks can be more or less susceptible to techniques that bind IP addresses to blockchain accounts, thus being detrimental to the alleged anonymity of their users. In this respect, flooding usually offers more protection than gossiping, presumably the main reason Bitcoin and the first generation of cryptocurrencies adopted it. However, if a significant amount of transaction traffic is collected and analyzed, the association between blockchain accounts and their IP addresses can be determined regardless of the P2P protocol, as shown in [107,108].

3.4.3. Communication Uncertainty

The mere fact of considering a distributed system makes it necessary to consider failures that may concern the nodes of which it is composed or the devices through which they communicate. Some nodes participating in the consensus process or an external attacker may cause malfunctions in communications, manifesting in delays or non-delivery of messages. Therefore, a proper design or deployment must consider assumptions concerning the way and amount of delays affecting message transmissions that the consensus protocol can tolerate without undoing its consistency and finality.

The theory of distributed systems considers the three following main communication uncertainty models, plus some minor variations obtained by strengthening or weakening in some way the assumptions related to these models (e.g., the weak synchronous model considered in [91]):

- *Synchronous*, which assumes the existence of some a priori known finite-time bound Δ, such that each message gets transmitted with a delay of at most Δ;
- *Asynchronous*, where the delivery messages might require any finite amount of time so that each message must eventually be delivered, without prediction of time needed to get to the destination;
- *Partial synchronous*, whose assumption is that there exists some known finite-time bound Δ and a special event called *Global Stabilization Time* (GST) such that: (i) the GST

eventually happens after some amount of unknown finite time, and; (ii) the delivery of any message sent at time τ must happen by time $\Delta + \max(\tau, GST)$.

The synchronous assumption is, in practice, too optimistic unless the value of Δ is overestimated, with the result of significantly worse performance. On the other hand, underestimating the value of Δ can negatively impact the consistency of a consensus protocol since nodes receiving messages after the Δ bound will miss some protocol steps. Notably, all protocols derived from the Nakamoto consensus rely on synchrony for their security (a relatively simple proof of this is in [109]).

Conversely, the asynchronous assumption corresponds to the worst-case scenario where we cannot predict the number of network delays. Consensus protocols designed under this assumption (e.g., HoneyBadgerBFT [75]) are usually very robust in terms of consistency and finality since they depend on neither time bounds for message delivery nor fixed values for timeouts. However, this comes at the price of higher complexity and lower performance.

Two direct results mark the gap between consensus protocols in the synchronous and asynchronous settings. Fisher, Lynch, and Paterson in 1985 [110] showed that any protocol that solves consensus withstanding the hang failure of just one node could not terminate in the asynchronous model. By contrast, several membership-based consensus protocols (see Section 3.3) introduced over the years can withstand up to half minus one dishonest member in the synchronous model (e.g., [111–113]).

Partial synchrony was introduced for consensus protocols in 1988 by [114] as a middle ground between the synchronous and asynchronous models. It indeed assumes that the system behaves asynchronously up to the occurrence of the GST, which turns out to be equivalent to say that there is some finite but unknown upper bound on message delivery. Over time, that has proved to be an excellent assumption for designing practical distributed systems, which usually are built on networks behaving synchronously except in case of unexpected events, such as a DoS attack or the temporary unavailability of one or more communication links. Indeed, this model allows designing consensus protocols that always guarantee consistency, providing liveness and finality only after the GST occurrence.

4. Emerging Concepts and Their Relevance in Applications

The simple Bitcoin transaction workflow sketched at the end of Section 3.2 was questioned in late 2013 by Vitalik Buterin, a programmer and co-founder of Bitcoin Magazine. Buterin argued to the Bitcoin core developers that many other applications and business sectors, besides money transfer and fintech, could have benefited from blockchain technology, provided, however, that they had a more powerful and flexible language than the Script language [115] provided for Bitcoin. After some work spent trying to bring out this point of view within the Bitcoin community, with his short involvement in the Colored Coins project (see Section 4.1.2), Buterin eventually conceived Ethereum as an alternative general platform for decentralized applications [56].

At the time of writing—shortly before the entry into service of the 2.0 version, which should modify the consensus protocol transitioning from a PoW to a hybrid PoW-PoS system—Ethereum looks like Bitcoin in terms of data structures and protocols for managing the ledger. However, it differs substantially in the application stack, especially for account management and transaction life cycle. Rather than enforcing one specific set of rules targeted toward the creation and usage of a cryptocurrency, Ethereum makes use of a cryptocurrency to allow users to write programs specifying whichever rules they want, upload the programs to the blockchain, and let the blockchain interpret and enforce the rules for them in a decentralized fashion. In order to achieve that, Ethereum's design is far more complicated than Bitcoin's. Transactions are no more designed to transfer coins by managing the addressing of UXTOs and their corresponding values but rather are cryptographically signed instructions by real accounts in the system to update the state of the network. Furthermore, in addition to having accounts associated with human users called *externally owned accounts* (EOAs), Ethereum also provides a *contract-type* account:

while an EOA state keeps the account's balance in *ether*, Ethereum's internal crypto-token, the state of a contract-type account saves the contract's code and storage. The contract's storage is encoded as a key-value database, whereas the contract's code gets encoded in Solidity [116], an object-oriented, high-level language for implementing smart contracts. These arbitrary transition rules in Ethereum govern the behavior of accounts within the system state. All the accounts and smart contracts which compose the Ethereum state live in the Ethereum Virtual Machine (EVM) [117], a particular distributed state machine deployed through the Ethereum clients and whose interpreter is Solidity. Unlike Script, Solidity is a Turing-complete programming language, for which it is not possible to predict whether a program will terminate or not (the Halting problem undecidability proven by Alan Turing in 1936 [118]). Thus, an inattentive or malicious programmer could create a contract containing not-terminating code, send a transaction to the contract, and crash the system. More generally, users could execute very demanding programs in terms of computational resources, causing system overloads and slowdowns. The solution in Ethereum is the concept of *gas*: the transaction sender specifies a maximum amount of computational resources (computing time and memory space) that they authorize the code execution to take and pay a transaction fee in ethers that is proportional to the number of resources spent. Ethereum must, therefore, be considered a *computing-as-a-service* platform, where ethers serve as an incentive for peers who manage the system and as a currency for processing. This radical change from Bitcoin's approach has sparked new perspectives in using blockchain technologies while entailing the evolution of some previous notions and tools plus the emergence of new ones.

4.1. Assets and Tokens

In financial accounting, an *asset* is anything tangible or intangible owned or controlled since it has positive economic value, usually quantifiable as a given amount of money, and cash itself is also considered an asset. It follows that, since their beginning, blockchains have been systems conceived and aimed at managing certain types of assets. Bitcoin had a native design to manage just the bitcoin cryptocurrency. However, a few years after its launch, several researchers—including Vitalik Buterin mentioned above—were trying to expand the system to allow it to manage other types of digital assets, comprising those suitable for representing physical assets in a unique and unalterable way. In 2012, Meni Rosenfeld [119] introduced a mechanism called *Colored Coins* to take advantage of the decentralized, trustless trading of bitcoins by allowing users to issue transaction outputs tagged for particular purposes, corresponding metaphorically to the emission of (fractions of) bitcoins having a unique "color", and to state their obligations to owners of coins of these colors. Today, the term "Colored Coins" loosely describes a class of methods for representing and managing real-world assets on top of Bitcoin, thanks to the fact that Script allows storing small amounts of metadata on the ledger, which can represent asset manipulation instructions. There are different labeling schemes (e.g., OBC, EPOBC, Open Assets [120]), but in any case, their purpose is to make it impossible to have coins of a specific color in a different way than receiving coins already of that color, following a chain of transactions traceable back to the color's genesis. This way, for example, it is possible to encode in a transaction that a certain amount in units of a new asset was issued, attaching a real-world value to those units by the asset issuer's promise to redeem them for some goods or services.

Since the emergence of more general-purpose and feature-rich blockchain systems, the idea of Colored Coins and their related tools have lost much of their interest; however, they represent the first attempts to find and build applications based on a cryptocurrency paving the way for an instrumental DLT concept. Today, this concept has reached its maturity and development in implementing Ethereum tokens and Hyperleger Fabric assets, respectively, in the context of permissionless and permissioned systems.

4.1.1. Ethereum Tokens

Essentially, tokens are programmable digital assets built on top of a digital ledger, such as a blockchain, whose lifecycle depends on smart contracts called *token contracts*. Therefore, other than representing a medium of exchange, such as a cryptocurrency, tokens are used to trigger certain functions in the smart contract(s) for a decentralized application, guaranteeing their authenticity and non-clonability. In the Ethereum system, tokens can represent a diverse range of digital assets and real-world, tangible objects. Depending on their scope and function, they are created and managed through a standard application programming interface (API) within smart contracts. The standardization process helps ensure tokens remain *composable* and *interoperable*. Their related smart contracts can be composed of and interoperate with any other standard-compliant smart contract on the Ethereum network. These standards are provided by and for the Ethereum community in EIPs; specifically, contract standards are Ethereum improvement proposals of category ERC, which encompasses application-level standards and conventions. Hence, they are often referenced with ERC instead of EIP in the literature but with the same numbering. The most relevant Ethereum token-related standards that at the time of writing have reached the "final" status are:

- *EIP-20: Token Standard* [121] is an API for *fungible* tokens, i.e., tokens all identical in specifications that can get expressed into smaller units, and both tokens and units can be implicitly interchangeable provided that they sum up to the same value. The first and foremost example of a fungible token is cash, and all other examples (e.g., game chips, cryptocurrencies, commodities, common shares) are functionally equivalent to it. Accordingly, this API provides functionalities, such as the tokens' transfer from one account to another, the current token balance of an account, and the total supply of the token available on the network; other functionalities include creating and checking allowances. EIP-20 is composed of six mandatory functions, three optional functions, and two events.
- *EIP-721: Non-Fungible Token Standard* [122] is an API for non-fungible tokens (NFTs). NFTs are used to identify something or someone uniquely since this kind of token comes up in unique samples, even if generated from the same smart contract. NFTs can be used to encode and manage the ownership of digital files, such as collectible items (e.g., photos, audio, and other digital artworks), access keys, and nominal tickets. However, access to any copy of the original file does not restrict the buyer of the NFT. Thus NFTs provide the owner with proof of ownership that is separate from copyright. EIP-721 provides core methods that allow tracking the owner of a unique identifier and a permissioned way for the owner to transfer the asset to others. This standard requires compliant tokens to implement ten mandatory functions and three events; moreover, they must implement the *EIP-165: Standard Interface Detection* [123].
- *EIP-777: Token Standard* [124] defines advanced features to interact with tokens while remaining backward compatible with EIP-20. This API introduces operators for sending tokens on behalf of another account and sending/receiving hooks offering holders more control over their tokens. It is composed of thirteen mandatory functions and five events.
- *EIP-1155: Multi-Token Standard* [125] outlines instead an interface that can represent any combination of fungible and non-fungible tokens in a single contract and transfer multiple token types at once. This API serves as an alternative to EIP-20 and EIP-721, which require the deployment of a separate contract for each token type or collection, placing a lot of redundant bytecode on the blockchain and limiting certain functionalities in some usage scenarios (e.g., blockchain games). It is composed of six mandatory functions and four events.

Apart from the accepted token standards, the Ethereum community is actively working on more advanced token functionalities. Particularly relevant in this context are securities-related tokens, whose goal is to code assets, such as a debt or equity claim, to the

issuer and must be, therefore, more heavily regulated than other tokens. The following are two interesting token standards for securities currently at the "draft" stage:

- *EIP-1450* [126] extends EIP-20 in order to facilitate the issuance and trading of securities in compliance with the U.S. Securities Exchange Commission (SEC) regulations. According to SEC requirements, this proposed standard mandates that the issuer is the only role that may create a token and assign the Registered Trade Agent (RTA), while the RTA is the only role that is allowed to execute the mint (creation), burn (destruction), and transfer of tokens. Implementers must maintain off-chain services and databases that record and track the investor's private information (name, physical address, Ethereum address, and security ownership amount) so that: (i) implementers and the SEC can access it on an as-needed basis; (ii) issuers and the RTA can produce a current list of all investors with names, addresses, and security ownership levels for every security at any given moment, and; (iii) issuers and the RTA can re-issue securities to investors for a variety of regulated reasons.
- *EIP-1462* [127] is another extension to EIP-20 that provides compliance with securities regulations and legal enforceability. However, in contrast to EIP-1450, this API defines a minimal set of additions to comply with the U.S. and international legal requirements. Such requirements include Know Your Customer (KYC) and Anti Money Laundering (AML) regulations, besides the ability to lock tokens for an account and restrict them from transfer due to a legal dispute. Another important requirement provided by this standard is the ability to attach additional legal documentation to set up a dual-binding relationship between the token and off-chain legal entities.

Ethereum's token idea is to have different digital assets to effectively represent the state and life cycle of diverse kinds of corresponding assets in the physical world, with their specific characteristics and behaviors. Just as ethers and fungible tokens intend to offer a more efficient and secure alternative to money transfers, mainly if related to fiat-backed stablecoins, NFTs and tokens related to securities aim to optimize the transfer of tangible or intangible assets, improving the assessment of their status and ownership.

In particular, NFTs can unlock the value of many kinds of assets by facilitating their sale and creating new markets. For example, digital artworks: a criticism often raised concerns over the ineffectiveness of an NFT in guaranteeing the actual possession of such assets, given that digital information can be replicated and shared at will. However, a closer look shows that the real problem does not derive from the NFT approach but the source of the data, which is the artist who produced the work. Assuming the originality and uniqueness of the artwork, as guaranteed by its creator, an NFT could, in principle, associate a single owner with this work and allow only the latter to have (full) access to it. In this case, of course, the artwork should be stored in a data repository with confidentiality protection, where the owner defines access rules and, for example, could decide to make copies at a reduced resolution of the original for public download or allow just viewing the original artwork through special devices at specific locations. Moreover, the concept of ownership should not be confused with usability: the fact of being able to see a piece of art freely certainly does not give the right to sell it or show it for a fee.

These and similar arguments should convince skeptics that issues and measures concerning a digital artwork (authenticity, rating, fruition, protection) are entirely analogous to those for a physical piece of art and that NFTs can help in implementing their management. The market for digital artifacts supported by NFTs is, in fact, already a very profitable reality, as shown for example by the CryptoKitties case (https://www.cryptokitties.co/, accessed on 15 October 2021) and by Christie's (https://www.christies.com/, accessed on 15 October 2021) online sale for $69 million of the file "Everydays: The First 5000 Days" by the graphic designer Mike Winkelmann (known as Beeple).

As we will see in Section 4.3, reliable and secure management of digital artworks through NFTs and, more generally, the registration of authentic off-chain data on the blockchain can be guaranteed thanks to the use of appropriate trusted third parties called *oracles*. A different approach is instead necessary to manage physical assets as, in this case,

it will be necessary to guarantee a unique and unalterable link between the object and its representing token. Of course, such a link must connect to the object's specific and unique physical characteristics, easily encodable as digital data. If, for example, a distributed application concerns the buying and selling of cars, then the NFT representing a car could store the license plate number and the chassis number of the car, plus some descriptive elements aimed at identifying the type of car (e.g., make, model, color), in a similar way to what is recorded today on the documents certifying the ownership of a vehicle. However, there are cases where more friendly and secure mechanisms can be put in place to build the tamper-proof binding between the physical object and its token, and this gives rise to the notion of an *anchor* presented in Section 4.2.

Resorting to anchors and oracles may be necessary not only in the context of the NFT technology pursued by Ethereum and similar systems. They also concern different approaches for using a blockchain, such as the one pursued in Hyperledger Fabric, which we are going to discuss in the next section.

4.1.2. Assets and Tokens in Hyperledger Fabric

Hyperledger Fabric was started in 2015 as an open-source project under the aegis of the Linux Foundation to design a blockchain system suitable for supporting interoperability between companies belonging to a consortium organization.

Therefore, its design goals were from the beginning different from those of the blockchain platforms prevailing at the time, offering some key differentiation capabilities in the management of both network nodes and assets. This system adopts a permissioned network model in which nodes with different functions cooperate to commit transactions on the blockchain. The resulting *execute-order-validate* workflow, illustrated in Figure 5, allows overcoming the following major drawbacks of the order-execute paradigm commonly adopted in blockchain systems [128]:

1. the trust model for transaction validation is determined at the consensus layer rather than at the application layer, thus forcing transactions to follow a validation workflow that can diverge from the actual application requirements;
2. consensus is hard-coded within the platform, resulting in monolithic applications that are difficult to upgrade and adapt to different network environments and threat scenarios;
3. the fact that all peers should execute all transactions degrades performance, increases consumption, and makes it more challenging to enforce privacy when necessary;
4. smart contracts must be written in a fixed, non-standard, or domain-specific language to avoid the occurrence of non-deterministic or non-terminating executions.

The first of the above issues can find a resolution through the *execute phase*: depending on the trust model at the application layer, an *endorsement policy* in the smart contract specifies which are the nodes, equipped with a local copy of the blockchain and running the smart contract, that have to check for the correctness of a transaction proposal issued by a client. These nodes, also known as *endorsers*, verify the correct encoding of the transaction proposal and its proper execution, returning the client a proposal response.

The client then collects endorsements until they satisfy the endorsement policy, proceeds to create the transaction, and passes it to the *ordering service* to give rise to the *order phase*. In this phase, the ordering service establishes a total order on all submitted transactions and outputs transaction blocks suitable to be stored in the blockchain, using a consensus protocol to tolerate faulty or byzantine orderers. Since orderers assemble transactions without verifying them, the consensus protocol can be unplugged from the application layer, thus overcoming the second issue.

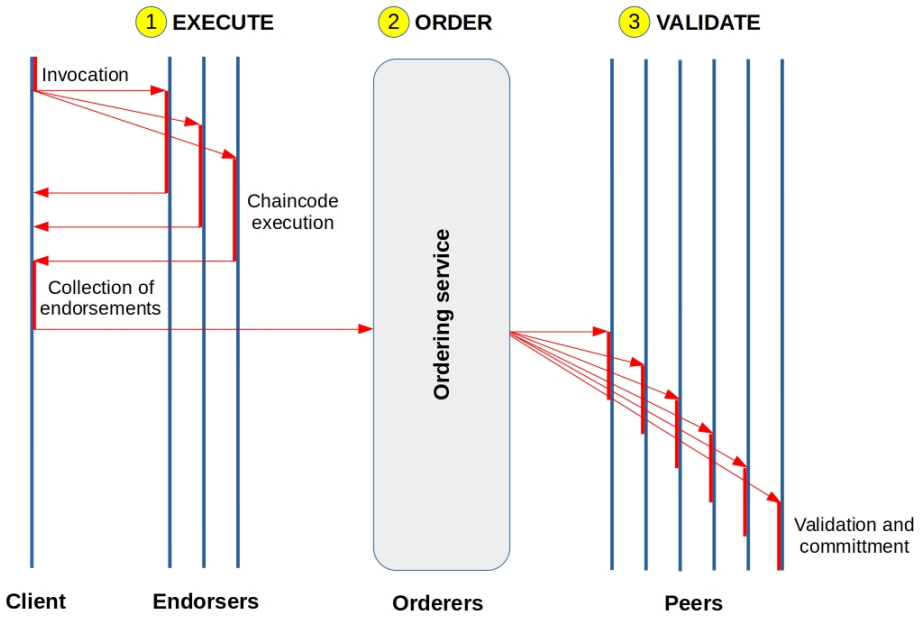

Figure 5. The transaction processing in Fabric entails three types of nodes, and as many steps as follows: (**1**) peers acting as endorsers check for the correctness of a transaction proposal by a client through the verification of its fields and the execution of the operation(s) provided in its payload; (**2**) orderers assemble all the endorsed transactions provided by clients in ordered blocks, where each block contains a reference to the previous block, and; (**3**) peers check the ordered blocks for valid transactions and store the resulting blocks in their local copy of the blockchain.

The ordering service then sends transaction blocks to the network for the *validate phase*, in which the nodes record the blocks in their local copy of the blockchain after tagging their transactions as valid or invalid. This second validity check compares the endorsement responses in the transactions with the current status of the ledger so that transactions are executed only by their endorsers, and the third issue is solved too. Moreover, the double-check performed through the *execute* and *validate* phases prevents the occurrence of non-deterministic or non-terminating executions. So, we can write smart contracts using a general-purpose language, such as Go, Java, or Javascript.

By exploiting the above design, Fabric can manage a complex semantic for assets and transactions directly at the ledger layer with its native API, without resorting to additional APIs, such as those defined for Ethereum's tokens. Accordingly, an asset in Fabric is represented by a generic but finite collection of key-value pairs, with state changes recorded as transactions on the ledger. Specifying suitable collections of key-value pairs makes it possible to consider virtually any kind of asset, from the tangible to the intangible. However, the management of assets does not inherently encompass their monetary value through the system since a cryptocurrency does not back Fabric; in other words, Fabric requires explicit programming to turn its assets into tokens. A research team at IBM has recently designed and implemented the two components Fabric Smart Client and Fabric Token SDK, which should facilitate token exchanges in enterprise contexts [129]. The goal was to design a modular token management system adaptable to various privacy and security regulations. This system adopts the UXTO model pioneered by Bitcoin with some tweaks to support issuance, redemption, transfer, and atomic swap of assets.

4.2. Anchors

Assuring the authenticity of products or their components is fundamental for many industries, from aerospace to electronics, pharmaceuticals, and the food industry. Unnoticed, faked products, such as food, drugs, diagnostic tests, electronic components, hardware

parts, and raw materials, pose a high risk of causing significant harm. Therefore, a related demand is to track and trace the logical and physical route, condition, and chain of custody or ownership of these goods throughout the supply chain and their lifecycle. Blockchain systems have built-in tamper-proof track and trace functionalities. However, a close link between physical objects and their digital representation is essential to avoid the *garbage-in, garbage-out* issue, where a blockchain-managed digital asset is tied to and used to represent an inauthentic asset in the physical world.

Typically, a product is linked to a digital record through an alphanumeric string (e.g., the Global Trade Item Number (GTIN) [130]) uniquely associated with the individual product or a class of products by model, batch, production site, manufacturer, or similar. This unique identifier (UID) is printed, embossed, or attached like a tag to the product or its packaging, and in many cases, it can be easily copied or transferred to a clone of the original product. UIDs by themselves cannot authenticate physical objects, also if they are cryptographically protected (e.g., authenticated), since they are not intrinsically tied to them in an unalterable way. What characterizes the authentication of a physical resource through alphanumeric data is indeed the following:

1. there must be a one-to-one unforgeable association between data items and resources, so that each data item corresponds to only one resource and vice versa, without any possible swapping or substitution in such correspondence;
2. the data item associated with the object must be authentic; that is, only the party who creates or holds the resource must be able to generate a valid data item representing it;
3. if a resource is subject to changes in its status (e.g., ownership, physical properties), then the one-to-one unforgeable association considered in (1) must guarantee a one-to-one correspondence with all the states that characterize the resource life cycle, thanks to an appropriate field in the data item.

We will refer to the above scheme as *authentic data encoding for physical resources* (ADExPR). Keep in mind that property (1) and its optional extension (3) in turn requires the following two properties: (i) *non-clonability* of the data items, and; (ii) *non-dissociability* between a resource and its related data item. *Clonability* refers to the fact that the data item for a given product can be reproduced (e.g., photocopied) and used for other products as well. *Dissociability* refers to a tag that can be detached from the original product and used for a different product.

Suppose a watch manufacturer decided to claim the originality of its products through QR tags encoding data encrypted with the manufacturer's private key that can be checked by customers knowing the related public key. Although the tag contains authentic data, it could be exposed to both cloning and detaching. For example, a standard QR tag encrypting solely the brand name can be copied multiple times and put on the packaging of faked watches. Moreover, a QR code produced with anti-cloning micro-printing technologies that encrypts the watch UID can fail in protecting its authenticity if it is printed or embossed on a physical medium that can be detached from the watch. These examples should clarify the gap between ADExPR and the classic notion of data authentication, along with the cryptographic tools to guarantee these last (i.e., message authentication codes and digital signatures).

An *anchor* ties a UID to the physical object with a property of the object that is hard to clone, forge, and transfer to another object. Such a property may be inherent to the object, or it may be entangled (anchored) so that, if removed, it destroys the object, an object functionality, or the property itself. This way, the property enforces the requisites of non-clonability and non-dissociability and, provided that the creators or owners of the objects authentically generated UIDs, anchors result in an ADExPR scheme. If some part of the above encoding has to rely on a cryptographic tool (e.g., a digital signature to guarantee the authenticity of the source of origin for the UIDs), then the more specific term *crypto-anchor* is often used.

In [17], the classification for anchors depicted in Figure 6 represents a starting point for the design of a blockchain-based *crypto-anchor platform* (https://www.ibm.com/blockchain/

solutions/transparent-supply, accessed on 15 October 2021) recently described in [18]. Deployed by IBM in collaboration with some leading vendors of crypto-anchor technologies, this platform supports application development ranging from sustainable sourcing to supply-chain management and consumer engagement.

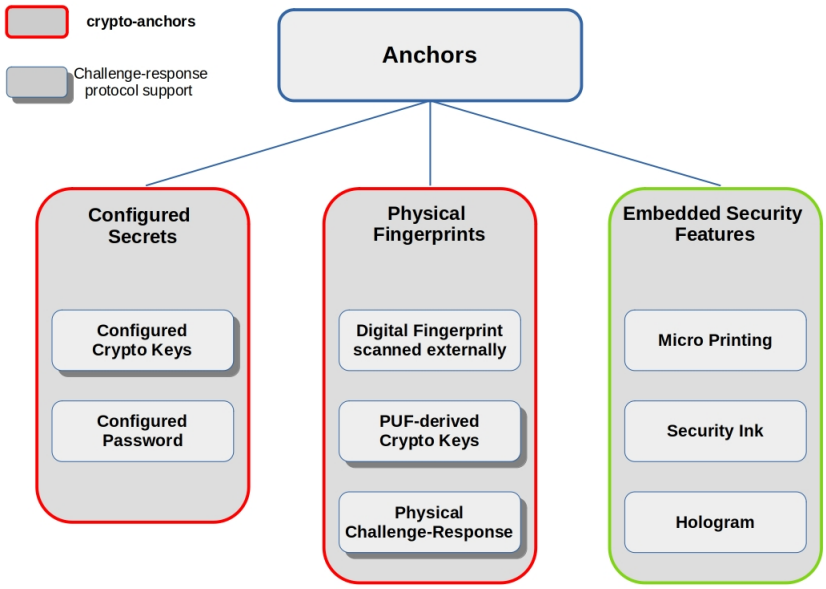

Figure 6. A classification of anchors based on the mechanism used to implement the entanglement of the UID to the physical object (adapted from [17]).

Configured secrets are useful for electronic devices. A manufacturer, distributor, or owner can configure these to store a secret, such as a password or a private key. Public-key cryptography allows for a challenge–response protocol through which the device can prove its identity and thwart reply attacks. However, in both symmetric and asymmetric schemes, the secret might be retrieved by an adversary able to physically interact with the device or retained by the person who configured the device for being reused multiple times. Tamper-proof hardware and cryptographic protocols resilient to side-channel attacks can mitigate the first issue at the price of more expensive devices.

In some cases, storing a secret in a device can be avoided by exploiting physical patterns or behaviors uniquely attributable to that device. Uncontrollable and unpredictable slight variations in the structure of specific materials or the outcome of some production processes can indeed give rise to *physical fingerprints*, such as in the production of semiconductors (e.g., transistors) or the texture of a surface. The above variability can be an intrinsic side effect of manufacturing or introduced on purpose and externally, as in doping material with extraneous particles. Since the variations are uncontrollable, they cannot be duplicated, not even by the original manufacturer; moreover, their unpredictable nature can work as a source of randomness for the setup of cryptographic material, such as secret keys and nonces. However, in order to be eligible as fingerprints of a given set S of objects, the physical variations affecting the objects in S must be characterized in some analytical or statistical sense to be attributable only to the objects in S, in a way that any two different objects in S give rise to two diverse variations. For example, the doping of semiconductors should be attributable to a unique manufacturer, semiconductor type, and semiconductor item. This circumstance means that the physical fingerprints for the objects in S represent the values of a function defined on S and injective therein, called *physical unclonable function* (PUF) [131].

Embedded security features do not rely on cryptographic mechanisms, requiring instead an expensive application process such as micro-printing, hologram generation, or printing with a security ink (e.g., photosensitive ink that is visible to the naked eye but changes color or disappears when placed under a UV light). Anyone can read but not copy a tag

produced in this way without special equipment. The difficulty and cost to reproduce an embedded security feature should deter most attackers; however, users often mistake false features for true ones unless they have special equipment for detecting fakes, as in the case of counterfeit banknotes.

An example of an anchor based on *embedded security features* was the *ValiGate* tag produced by *tesa scribos*, a company that until mid-2021 was part of *tesa group* (https://www.tesa.com/en-in/about-tesa, accessed on 7 July 2021). ValiGate looked like a normal QR-code but contained a data field protected with a dual encryption algorithm and printed thanks to a proprietary micro-printing technique, requiring a dedicated smartphone scanning app. In this case, encryption provided an authenticated source of origin, while micro-printing served to counteract cloning attacks by preventing a simple "scan and print" approach. A recent change of ownership of tesa scribos to *SCRIBOS* (https://www.scribos.de/en, accessed on 15 October 2021)—a subsidiary of the german *KURZ Group* specialized in protecting brands with product markings and digital tools—introduced a new version of the Valigate tag removing special printing features, making it compatible with the world's most widely available printing machines and not requiring dedicated scanning apps.

An alternative in the class of the *configured secrets* is a special kind of signature patented by one of the authors [132], which is only valid if the associated physical asset has a given status and gets univocally tied to a characteristic data set for the asset. In this case, the resilience to cloning is because a valid tag can be generated only by a source of authority (e.g., a product brand), and one or more data uniquely tied to the asset only get disclosed when the asset reaches a given status (e.g., when a product on sale gets sold).

4.3. Oracles

Distributed ledger *oracles* are third-party services that provide smart contracts with external information. They intend to prevent another possible cause of the *garbage-in, garbage-out* problem, i.e., when off-chain data from unverified sources or data providers become input to smart contracts, which could give rise to incorrect processing and related results, with potentially dangerous consequences for both the system and its users. An oracle for a given decentralized application intercepts data originating from one or more sources, possibly processing such data and then delivering them to the smart contract application along with an *authenticity proof*, a cryptographic guarantee proving the reliability of such data production and that no one tampered with it. Data collected by oracles can originate from databases or other online software sources of information and physical devices, such as electronic sensors, barcode readers, and robots; in some cases, humans can pass data as inputs to oracles directly.

An essential aspect of oracles is the trust model they implement concerning the multiplicity and heterogeneity of the data they feed a given decentralized system. The oracles needed by smart contracts and their relative trust models can indeed strongly influence the type and level of decentralization implemented at the core system level. Consider, for example, a blockchain system aimed at tracing and guaranteeing the quality of products for a specific production chain. Suppose each party that executes a part of the process independently certifies the quality of their sub-process or by-product through a dedicated and autonomously managed oracle. In that case, the resulting system maintains the degree of decentralization initially provided by the blockchain architecture. Things are pretty different if, on the contrary, a single company acting as a certifying body manages all the oracles used by the system. In the latter case, the system preserves decentralization only if the oracles themselves implement this model of trust.

The blockchain ecosystem currently already provides for both centralized and decentralized oracles. For the first ones, a single entity acting as a trusted third-party controls them, while the latter implement some form of DLT to increase the reliability of the information provided to smart contracts by not relying on a single source of truth. In order to preserve decentralization, it is clear that the former kind of oracles requires a smart contract

that compares multiple sources and selects the data items to be taken into consideration by the majority.

Provable (https://provable.xyz, accessed on 15 October 2021) is an example of an *attestation-as-a-service* company that delivers the data a customer needs along with proof of their authenticity. Provable provides platform-agnostic oracles possibly integrated with centralized or decentralized systems at the customer side. Integration is currently achievable with the smart contracts of some major blockchain platforms, such as Ethereum, EOS, Hyperledger Fabric, and R3 Corda.

Some of the authenticity proofs provided by Provable are [133]:

- *TLSNotary Proof*; which relies on a feature of TLS 1.x protocols to enable the splitting of the TLS master key among three parties: the server, an auditee, and an auditor. Provable is the auditee in this scheme, while a locked-down instance of a specially designed, open-source virtual machine on *Amazon Elastic Computing Cloud* (https://aws.amazon.com/ec2, accessed 15 October 2021) acts as the auditor.
- *Android Proof*; which makes use of a software remote attestation technology developed by Google, called SafetyNet [134], to validate that a given Android application is running on a safe, non-rooted physical device connected to Provable's infrastructure. It also remotely validates the application code hash, enabling authentication of the application running on the device.
- *Ledger Proof*; which leverages a trusted execution environment comprising an STMicroelectronics secure element, a controller, and BOLOS [135], an operating system developed for hardware wallets by *Ledger* (https://www.ledger.com/the-company, accessed 15 October 2021). BOLOS exposes a set of kernel-level cryptographic APIs, including attestation: any application can ask the kernel to measure its binary and produce a signed hash.

Provable's authenticity proofs can be either delivered directly to a smart contract or uploaded and stored on some alternate storage medium, such as the *interplanetary file system* (IPFS) [136], a P2P file system that combines a distributed hashtable, an incentivized block exchange, and a self-certifying namespace so that its nodes do not need to trust each other.

ChainLink (https://chain.link/, accessed on 15 October 2021), on the other hand, is an example of a company that is revamping in a joint effort with academia, with the goal of approaching decentralized oracles. In the recently updated whitepaper [137] concerning ChainLink goal and architecture, the authors envision an increasingly expansive role for decentralized oracles thanks to a broader oracle notion and a new definition for the oracle network concept previously introduced in [138]. In this up-to-date design, an oracle is a general-purpose, bidirectional, and "compute-enabled" interface between on-chain and off-chain systems, while a *decentralized oracle network* (DON) is a set of oracles provided by third parties and maintained by a committee of Chainlink nodes through a consensus protocol. A DON supports a range of oracle functions chosen for deployment by the committee, acting as a blockchain abstraction layer and providing interfaces to off-chain resources for both smart contracts and other systems. Its goal is to enable a flexible and secure combination of on-chain and off-chain computations connected to external resources, resulting in the *hybrid smart contract*-oriented architecture sketched in Figure 7.

Ultimately, the ambition of the above design is to push the degree of abstraction achieved by Chainlink to the point of implementing a layer of *meta contracts* that abstract away the on-chain/off-chain distinction for all classes of developers and users, allowing the seamless creation and use of decentralized services.

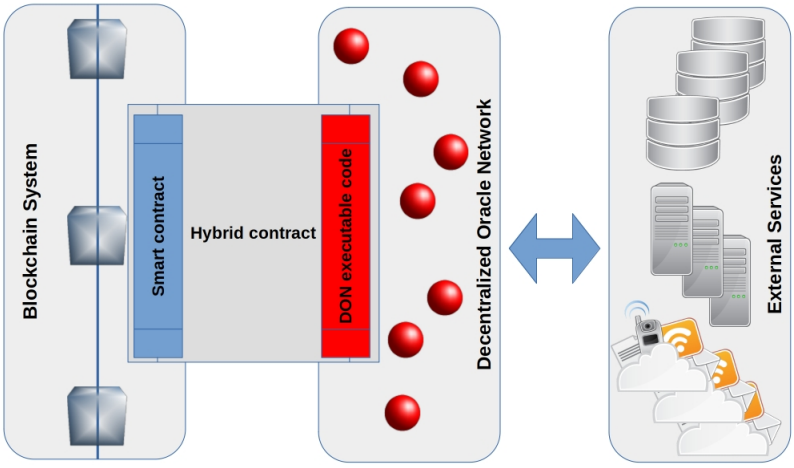

Figure 7. A hybrid smart contract consists of a smart contract component and an off-chain component that executes on a DON. The DON connects the two components and the hybrid contract with off-chain resources, such as web services, databases, other blockchains.

5. Conclusions

Following the critical approach used in [46] during the booming of DLT, in this work, we presented the most relevant and impacting concepts within the context of technologies sprouted from the affirmation of Bitcoin. Such concepts, sometimes already present in the past literature, now receive an injection of concrete applicability. Opportunities to make a profit or affirm new governance approaches have nurtured a plethora of initiatives that we tried to organize by concepts and impact in implementing the actual applications.

This paper represents our tentative to exhibit sometimes underexposed but still impactful new or consolidated aspects of DLT. It was born from our need to have a clearer view of tangled and often just-sketched proposals available in the literature or on the market during our involvement in applicative projects. We propose a digest of valuable concepts for beginners and experts looking for unexpected links among ideas in the DLT scene. In particular, we propose a reference architecture for DLT, offering an overview of concepts and tools introduced by this technology. In Section 3.3 about consensus, we propose a classification of protocols following a historical path marking the progression of ideas that emerged with the evolution of DL systems. We propose the concept of process authenticity, which we believe will be more and more prominent in the following years, presenting a review of the most relevant tools and strategies. Furthermore, in the context of anchors, we introduced the ADExPR scheme to characterize the authentication of physical resources through alphanumeric data, which can facilitate the reader to discriminate from the qualities of digital data authentication.

Author Contributions: Conceptualization, G.S.; methodology, G.S.; writing—original draft preparation, D.R. and G.S.; writing—review and editing, D.R.; supervision, G.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in article.

Acknowledgments: We wish to thank the anonymous reviewers, whose comments and suggestions allowed to improve the presentation of our work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Truong, N.B.; Um, T.W.; Zhou, B.; Lee, G.M. Strengthening the blockchain-based internet of value with trust. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas, MO, USA, 20–24 May 2018; pp. 1–7.
2. Rauchs, M.; Glidden, A.; Gordon, B.; Pieters, G.C.; Recanatini, M.; Rostand, F.; Vagneur, K.; Zhang, B.Z. *Distributed Ledger Technology Systems: A Conceptual Framework*; The Cambridge Centre for Alternative Finance: Cambridge, UK, 2018.
3. Sunyaev, A. Distributed ledger technology. In *Internet Computing*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 265–299.
4. Dossier Blockchain. 2021. Available online: https://www.statista.com/study/39859/blockchain-statista-dossier/ (accessed on 23 September 2021).
5. Witzig, P.; Salomon, V. *Cutting Out the Middleman: A Case Study of Blockchain-Induced Reconfigurations in the Swiss Financial Services Industry*; Technical Report; Université de Neuchâtel: Neuchâtel, Switzerland, 2018.
6. Statista2021usecases. 2021. Available online: https://www.statista.com/statistics/878732/worldwide-use-cases-blockchain-technology/ (accessed on 23 September 2021).
7. Statista2020barriers. 2020. Available online: https://www.statista.com/statistics/878686/worldwide-investment-barriers-blockchain-technology/ (accessed on 23 September 2021).
8. Babich, V.; Hilary, G. OM Forum—Distributed ledgers and operations: What operations management researchers should know about blockchain technology. *Manuf. Serv. Oper. Manag.* **2020**, *22*, 223–240. [CrossRef]
9. Powell, W.; Foth, M.; Cao, S.; Natanelov, V. Garbage in garbage out: The precarious link between IoT and blockchain in food supply chains. *J. Ind. Inf. Integr.* **2021**, *25*, 100261. [CrossRef]
10. Zhang, R.; Xue, R.; Liu, L. Security and privacy on blockchain. *ACM Comput. Surv.* **2019**, *52*, 51. [CrossRef]
11. Belotti, M.; Božić, N.; Pujolle, G.; Secci, S. A vademecum on blockchain technologies: When, which, and how. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3796–3838. [CrossRef]
12. Chowdhury, M.J.M.; Ferdous, M.S.; Biswas, K.; Chowdhury, N.; Kayes, A.; Alazab, M.; Watters, P. A comparative analysis of distributed ledger technology platforms. *IEEE Access* **2019**, *7*, 167930–167943. [CrossRef]
13. Xiao, Y.; Zhang, N.; Lou, W.; Hou, Y.T. A survey of distributed consensus protocols for blockchain networks. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1432–1465. [CrossRef]
14. Antal, C.; Cioara, T.; Anghel, I.; Antal, M.; Salomie, I. Distributed Ledger Technology Review and Decentralized Applications Development Guidelines. *Future Internet* **2021**, *13*, 62. [CrossRef]
15. Dorri, A.; Kanhere, S.; Jurdak, R. *Blockchain for Cyberphysical Systems*; Artech House: London, UK, 2020.
16. Beniiche, A. A study of blockchain oracles. *arXiv* **2020**, arXiv:2004.07140.
17. Balagurusamy, V.S.K.; Cabral, C.; Coomaraswamy, S.; Delamarche, E.; Dillenberger, D.N.; Dittmann, G.; Friedman, D.; Gökçe, O.; Hinds, N.; Jelitto, J.; et al. Crypto anchors. *IBM J. Res. Dev.* **2019**, *63*, 4:1–4:12. [CrossRef]
18. Prada-Delgado, M.A.; Dittmann, G.; Circiumaru, I.; Jelitto, J. A blockchain-based crypto-anchor platform for interoperable product authentication. In Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Korea, 22–28 May 2021; pp. 1–5.
19. Fowler, M.; Lewis, J. Microservices. 2014. Available online: http://hashcash.org/docs/hashcash.txt (accessed on 14 June 2021).
20. Newman, S. *Building Microservices: Designing Fine-Grained Systems*; O'Reilly Media, Inc.: Newton, MA, USA, 2015.
21. Guo, D.; Wang, W.; Zeng, G.; Wei, Z. Microservices architecture based cloudware deployment platform for service computing. In Proceedings of the 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE), Oxford, UK, 29 March–2 April 2016; pp. 358–363.
22. Spoonhower, D. Lessons from the Birth of Microservices at Google. 2018. Available online: https://dzone.com/articles/lessons-from-the-birth-of-microservices-at-google (accessed on 23 September 2021).
23. Hillpot, J. 4 Microservices Examples: Amazon, Netflix, Uber, and Etsy. 2021. Available online: https://blog.dreamfactory.com/microservices-examples/ (accessed on 23 September 2021).
24. Gilder, G. *Life after Google: The Fall of Big Data and the Rise of the Blockchain Economy*; Simon and Schuster: New York, NY, USA, 2018.
25. Zuboff, S. Big other: Surveillance capitalism and the prospects of an information civilization. *J. Inf. Technol.* **2015**, *30*, 75–89. [CrossRef]
26. Pathirana, N. How to Set Up a Private Ethereum Blockchain. 2019. Available online: https://medium.com/swlh/how-to-set-up-a-private-ethereum-blockchain-c0e74260492c (accessed on 23 July 2021).
27. Medicalchain Whitepaper 2.1. 2018. Available online: https://medicalchain.com/Medicalchain-Whitepaper-EN.pdf (accessed on 6 November 2021).
28. Douceur, J.R. The sybil attack. In *International Workshop on Peer-to-Peer Systems*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 251–260.
29. Osipkov, I.; Vasserman, E.Y.; Hopper, N.; Kim, Y. Combating double-spending using cooperative P2P systems. In Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS'07), Toronto, ON, Canada, 25–27 June 2007; p. 41.

30. Bevand, M. Cambridge Bitcoin Electricity Consumption Index. 2017. Available online: https://cbeci.org/ (accessed on 27 July 2021).
31. Bevand, M. Energy Consumption of a Bitcoin (BTC, BTH) and VISA Transaction. 2021. Available online: https://www.statista.com/statistics/881541/bitcoin-energy-consumption-transaction-comparison-visa/ (accessed on 27 July 2021).
32. Cox, J. Yellen Sounds Warning about "Extremely Inefficient" Bitcoin. 2021. Available online: https://www.cnbc.com/2021/02/22/yellen-sounds-warning-about-extremely-inefficient-bitcoin.html (accessed on 27 July 2021).
33. Kolodny, L. Elon Musk Says Tesla Will Stop Accepting Bitcoin for Car Purchases, Citing Environmental Concerns. 2021. Available online: https://www.cnbc.com/2021/05/12/elon-musk-says-tesla-will-stop-accepting-bitcoin-for-car-purchases.html (accessed on 27 July 2021).
34. Xu, B.; Luthra, D.; Cole, Z.; Blakely, N. EOS: An Architectural, Performance, and Economic Analysis. 2018; p. 25. Available online: https://whiteblock.io/wp-content/uploads/2019/07/eos-test-report.pdf (accessed on 1 August 2021).
35. Goodman, L.M. Tezos—A Self-Amending Crypto-Ledger White Paper. 2014; p. 17. Available online: https://tezos.com/whitepaper.pdf (accessed on 1 August 2021).
36. Internet Computer Governance. Available online: https://sdk.dfinity.org/docs/introduction/welcome.html (accessed on 1 August 2021).
37. DFINITY Technical Library. Available online: https://dfinity.org/technicals (accessed on 1 August 2021).
38. Kahng, A.; Mackenzie, S.; Procaccia, A. Liquid Democracy: An Algorithmic Perspective. *J. Artif. Intell. Res.* **2021**, *70*, 1223–1252. [CrossRef]
39. Decred Documentation. Available online: https://docs.decred.org/ (accessed on 1 August 2021).
40. Pelt, R.V.; Jansen, S.; Baars, D.; Overbeek, S. Defining Blockchain Governance: A Framework for Analysis and Comparison. *Inf. Syst. Manag.* **2021**, *38*, 21–41. [CrossRef]
41. Johnson, D.; Menezes, A.; Vanstone, S. The elliptic curve digital signature algorithm (ECDSA). *Int. J. Inf. Secur.* **2001**, *1*, 36–63. [CrossRef]
42. Cooper, D.; Santesson, S.; Farrell, S.; Boeyen, S.; Housley, R.; Polk, W. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. 2008. Available online: http://tools.ietf.org/html/rfc5280 (accessed on 27 June 2021).
43. Membership Service Providers. 2020. Available online: https://hyperledger-fabric.readthedocs.io/en/release-2.2/msp.html (accessed on 14 August 2021)
44. Rani, D.; Ranjan, R.K. A comparative study of SaaS, PaaS and IaaS in cloud computing. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **2014**, *4*, 158–161.
45. Rimal, B.P.; Maier, M. Workflow scheduling in multi-tenant cloud computing environments. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *28*, 290–304. [CrossRef]
46. Romano, D.; Schmid, G. Beyond bitcoin: A critical look at blockchain-based systems. *Cryptography* **2017**, *1*, 15. [CrossRef]
47. Contract ABI Specification. 2020. Available online: https://docs.soliditylang.org/en/develop/abi-spec.html (accessed on 14 August 2021).
48. Fabric Chaincode Lifecycle. 2020. Available online: https://hyperledger-fabric.readthedocs.io/en/release-2.2/chaincode_lifecycle.html (accessed on 14 August 2021).
49. Nakamoto, S. Bitcoin: A Peer to Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 27 June 2021).
50. Rogaway, P.; Shrimpton, T. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *Fast Software Encryption*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 371–388.
51. Haber, S.; Stornetta, W.S. How to time-stamp a digital document. In Proceedings of the Conference on the Theory and Application of Cryptography, Santa Barbara, CA, USA, 11–15 August 1990; Springer: Berlin/Heidelberg, Germany, 1990; pp. 437–455.
52. Merkle, R.C. A digital signature based on a conventional encryption function. In Proceedings of the Advances in Cryptology—CRYPTO'87, Santa Barbara, CA, USA, 16–20 August 1987; Springer: Berlin/Heidelberg, Germany, 1987; pp. 369–378.
53. Dwork, C.; Naor, M. Pricing via processing or combatting junk mail. In Proceedings of the Advances in Cryptology—CRYPTO'92, Santa Barbara, CA, USA, 16–20 August 1992; Springer: Berlin/Heidelberg, Germany, 1993; pp. 139–147.
54. Howell, A. The Longest Blockchain Is Not the Strongest Blockchain. 2019. Available online: https://cryptoservices.github.io/blockchain/consensus/2019/05/21/bitcoin-length-weight-confusion.html (accessed on 27 July 2021) .
55. Eyal, I.; Sirer, E.G. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 436–454.
56. Buterin, V. Ethereum White Paper. 2013. Available online: https://ethereum.org/en/whitepaper/ (accessed on 3 July 2021).
57. Gervais, A.; Karame, G.O.; Wüst, K.; Glykantzis, V.; Ritzdorf, H.; Capkun, S. On the security and performance of proof of work blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 3–16.
58. Szilagyi, P.; Wilcke, J.; Lange, F.; Zsolt, F. Go-Ethereum/Core/Blockchain.go: Difficulty Calculation. 2017. Available online: https://github.com/ethereum/go-ethereum/blob/525116dbff916825463931361f75e75e955c12e2/core/blockchain.go#L840 (accessed on 27 July 2021).
59. Morrison, D.R. PATRICIA—Practical algorithm to retrieve information coded in alphanumeric. *J. ACM* **1968**, *15*, 514–534. [CrossRef]

60. Lerner, S.D. DagCoin Draft. 2015. Available online: https://bitslog.files.wordpress.com/2015/09/dagcoin-v41.pdf (accessed on 27 July 2021).
61. Thulasiraman, K.; Swamy, M.N. *Graphs: Theory and Algorithms*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
62. Ribero, Y. DagCoin Whitepaper. 2018. Available online: https://prismic-io.s3.amazonaws.com/dagcoin/f4e531e1-a5db-43b6-930c-14bf705e65ee_Dagcoin_White_Paper.pdf (accessed on 27 July 2021).
63. Sompolinsky, Y.; Lewenberg, Y.; Zohar, A. SPECTRE: A fast and scalable cryptocurrency protocol. *IACR Cryptol. ePrint Arch.* **2016**, *2016*, 1–71.
64. Li, C.; Li, P.; Zhou, D.; Xu, W.; Long, F.; Yao, A. Scaling nakamoto consensus to thousands of transactions per second. *arXiv* **2018**, arXiv:1805.03870.
65. Popov, S. The Tangle, 2018. Available online: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf (accessed on 27 July 2021).
66. Popov, S.; Moog, H.; Camargo, D.; Capossele, A.; Dimitrov, V.; Gal, A.; Greve, A.; Kusmierz, B.; Mueller, S.; Penzkofer, A.; et al. The Coordicide. 2020. Available online: https://files.iota.org/papers/20200120_Coordicide_WP.pdf (accessed on 27 July 2021).
67. Hearn, M.; Brown, R.G. Corda: A Distributed Ledger. 2019. Available online: https://www.r3.com/wp-content/uploads/2019/08/corda-technical-whitepaper-August-29-2019.pdf (accessed on 27 July 2021).
68. Junqueira, F.P.; Reed, B.C.; Serafini, M. Zab: High-performance broadcast for primary-backup systems. In Proceedings of the IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN), Hong Kong, China, 27–30 June 2011; pp. 245–256.
69. Ongaro, D.; Ousterhout, J. In search of an understandable consensus algorithm. In Proceedings of the USENIX Annual Technical Conference ATC 14, Philadelphia, PA, USA, 19–20 June 2014; pp. 305–319.
70. Lamport, L. Proving the correctness of multiprocess programs. *IEEE Trans. Softw. Eng.* **1977**, *SE-3*, 125–143. [CrossRef]
71. Lamport, L.; Shostak, R.; Pease, M. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* **1982**, *4*, 382–401. [CrossRef]
72. Pease, M.; Shostak, R.; Lamport, L. Reaching agreement in the presence of faults. *J. ACM* **1980**, *27*, 228–234. [CrossRef]
73. Castro, M.; Liskov, B. Practical byzantine fault tolerance. *OSDI* **1999**, *99*, 173–186.
74. Schneider, F.B. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.* **1990**, *22*, 299–319. [CrossRef]
75. Miller, A.; Xia, Y.; Croman, K.; Shi, E.; Song, D. The honey badger of BFT protocols. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 31–42.
76. Duan, S.; Reiter, M.K.; Zhang, H. BEAT: Asynchronous BFT made practical. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 2028–2041.
77. Cachin, C.; Vukolić, M. Blockchain consensus protocols in the wild. *arXiv* **2017**, arXiv:1707.01873.
78. Bano, S.; Sonnino, A.; Al-Bassam, M.; Azouvi, S.; McCorry, P.; Meiklejohn, S.; Danezis, G. Consensus in the age of blockchains. *arXiv* **2017**, arXiv:1711.03936.
79. Wang, W.; Hoang, D.T.; Hu, P.; Xiong, Z.; Niyato, D.; Wang, P.; Wen, Y.; Kim, D.I. A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access* **2019**, *7*, 22328–22370. [CrossRef]
80. Wood, G. Ethereum: A Secure Decentralized Generalized Transaction Ledger. 2016. Available online: https://ethereum.github.io/yellowpaper/paper.pdf (accessed on 27 July 2021).
81. King, S.; Nadal, S. Ppcoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. 2012. Available online: https://peercoin.net/assets/paper/peercoin-paper.pdf (accessed on 27 July 2021).
82. Community, N. Nxt White Paper. 2014. Available online: https://www.jelurida.com/sites/default/files/NxtWhitepaper.pdf (accessed on 3 July 2021).
83. Miller, A.; Juels, A.; Shi, E.; Parno, B.; Katz, J. Permacoin: Repurposing bitcoin work for data preservation. In Proceedings of the 2014 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 18–21 May 2014; pp. 475–490.
84. Chen, L.; Xu, L.; Shah, N.; Gao, Z.; Lu, Y.; Shi, W. On security analysis of proof-of-elapsed-time (poet). In Proceedings of the International Symposium on Stabilization, Safety, and Security of Distributed Systems, Boston, MA, USA, 5–8 November 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 282–297.
85. Costan, V.; Devadas, S. Intel sgx explained. *IACR Cryptol. ePrint Arch.* **2016**, *2016*, 1–118.
86. Ethash Design Rationale. 2020. Available online: https://eth.wiki/concepts/ethash/design-rationale (accessed on 3 July 2021).
87. Proof of Stake FAQs. 2020. Available online: https://eth.wiki/concepts/proof-of-stake-faqs (accessed on 3 July 2021).
88. Bentov, I.; Lee, C.; Mizrahi, A.; Rosenfeld, M. Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake. In Proceedings of the SIGMETRICS 2014 Workshop on Economics of Networked Systems, Austin, TX, USA, 16–20 June, 2014.
89. Li, W.; Andreina, S.; Bohli, J.M.; Karame, G. Securing proof-of-stake blockchain protocols. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 297–315.
90. Kiayias, A.; Russell, A.; David, B.; Oliynykov, R. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 20–24 August 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 357–388.
91. Gilad, Y.; Hemo, R.; Micali, S.; Vlachos, G.; Zeldovich, N. Algorand: Scaling byzantine agreements for cryptocurrencies. In Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, 28 October 2017; pp. 51–68.

92. David, B.; Gaži, P.; Kiayias, A.; Russell, A. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, 30 April–4 May 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 66–98.
93. Daian, P.; Pass, R.; Shi, E. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In Proceedings of the International Conference on Financial Cryptography and Data Security, St. Kitts, Saint Kitts and Nevis, 18–22 February 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 23–41.
94. Buchman, E. Tendermint: Byzantine Fault Tolerance in the Age of Blockchains. Ph.D. Thesis, University of Guelph, Guelph, ON, Canada, 2016.
95. Wood, G. PoA Private Chains. 2015. Available online: https://github.com/ethereum/guide/blob/master/poa.md (accessed on 27 July 2021) .
96. Buchman, E.; Kwon, J.; Milosevic, Z. The latest gossip on BFT consensus. *arXiv* **2019**, arXiv:1807.04938.
97. Bentov, I.; Gabizon, A.; Mizrahi, A. Cryptocurrencies without proof of work. In Proceedings of the International Conference on Financial Cryptography and Data Security, Christ Church, Barbados, 22–26 February 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 142–157.
98. Stadler, M. Publicly verifiable secret sharing. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Saragossa, Spain, 12–16 May 1996; Springer: Berlin/Heidelberg, Germany, 1996; pp. 190–199.
99. Cramer, R.; Damgård, I. Multiparty computation, an introduction. In *Contemporary Cryptology*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 41–87.
100. Pass, R.; Shi, E. The sleepy model of consensus. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 380–409.
101. Micali, S.; Rabin, M.; Vadhan, S. Verifiable random functions. In Proceedings of the 40th Annual Symposium on Foundations of Computer Science (cat. No. 99CB37039), New York, NY, USA, 17–19 October 1999; pp. 120–130.
102. Bitshares Documentation: Delegated Proof of Stake (DPOS). Available online: https://how.bitshares.works/en/master/technology/dpos.html (accessed on 11 September 2021).
103. Lavin, J.; Larimer, D.; Hourt, N.; Ma, Q.; Prioriello, W. EOS.IO Technical White Paper v2. 2018. Available online: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md (accessed on 27 July 2021).
104. Buterin, V.; Griffith, V. Casper the friendly finality gadget. *arXiv* **2017**, arXiv:1710.09437.
105. Dang, H.; Dinh, T.T.A.; Loghin, D.; Chang, E.C.; Lin, Q.; Ooi, B.C. Towards scaling blockchain systems via sharding. In Proceedings of the 2019 International Conference on Management of Data, Amsterdam, The Netherlands, 30 June–5 July 2019; pp. 123–140.
106. Schollmeier, R. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In Proceedings of the First International Conference on Peer-to-Peer Computing, Linköping, Sweden, 27–29 August 2001; pp. 101–102.
107. Fanti, G.; Viswanath, P. Deanonymization in the bitcoin P2P network. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 1364–1373.
108. Biryukov, A.; Tikhomirov, S. Deanonymization and linkability of cryptocurrency transactions based on network analysis. In Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroS&P), Stockholm, Sweden, 17–19 June 2019; pp. 172–184.
109. Ren, L. Analysis of Nakamoto Consensus. *IACR Cryptol. ePrint Arch.* **2019**, *2019*, 943.
110. Fischer, M.J.; Lynch, N.A.; Paterson, M.S. Impossibility of distributed consensus with one faulty process. *J. ACM* **1985**, *32*, 374–382. [CrossRef]
111. Liu, S.; Viotti, P.; Cachin, C.; Quéma, V.; Vukolić, M. XFT: Practical fault tolerance beyond crashes. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016; pp. 485–500.
112. Dolev, D.; Strong, H.R. Authenticated algorithms for Byzantine agreement. *SIAM J. Comput.* **1983**, *12*, 656–666. [CrossRef]
113. Abraham, I.; Malkhi, D.; Nayak, K.; Ren, L.; Yin, M. Sync hotstuff: Simple and practical synchronous state machine replication. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–21 May 2020; pp. 106–118.
114. Dwork, C.; Lynch, N.; Stockmeyer, L. Consensus in the presence of partial synchrony. *J. ACM* **1988**, *35*, 288–323. [CrossRef]
115. Bitcoin Wiki. Script. Available online: https://en.bitcoin.it/wiki/Script (accessed on 3 July 2021).
116. Solidity Documentation. Available online: https://solidity.readthedocs.io/en/develop/ (accessed on 3 July 2021).
117. Ethereum Virtual Machine (EVM). 2020. Available online: https://ethereum.org/en/developers/docs/evm/ (accessed on 14 August 2021).
118. Turing, A.M. On computable numbers, with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.* **1937**, *2*, 230–265. [CrossRef]
119. Rosenfeld, M. Overview of Colored Coins. 2012. Available online: https://bitcoil.co.il/BitcoinX.pdf (accessed on 14 August 2021).
120. Bitcoin Wiki. Colored Coins. Available online: https://en.bitcoin.it/wiki/Colored_Coins (accessed on 3 July 2021).

121. Vogelsteller, F.; Buterin, V. EIP-20: Token Standard. 2015. Available online: https://eips.ethereum.org/EIPS/eip-20 (accessed on 14 August 2021).
122. Entriken, W.; Shirley, D.; Evans, J.; Sachs, N. EIP-721: Non-Fungible Token Standard. 2018. Available online: https://eips.ethereum.org/EIPS/eip-721 (accessed on 14 August 2021).
123. Reitwießner, C.; Johnson, N.; Vogelsteller, F.; Baylina, J.; Feldmeier, K.; Entriken, W. EIP-165: Standard Interface Detection. 2018. Available online: https://eips.ethereum.org/EIPS/eip-165 (accessed on 14 August 2021).
124. Dafflon, J.; Baylina, J.; Shababi, T. EIP-777: Token Standard. 2017. Available online: https://eips.ethereum.org/EIPS/eip-777 (accessed on 14 August 2021).
125. Radomski, W.; Cooke, A.; Castonguay, P.; Therien, J.; Binet, E.; Sandford, R. EIP-1155: Multi-Token Standard. 2018. Available online: https://eips.ethereum.org/EIPS/eip-1155 (accessed on 14 August 2021).
126. Shiple, J.; Marks, H.; Zhang, D. EIP-1450: A Compatible Security Token for Issuing and Trading SEC-Compliant Securities. 2018. Available online: https://eips.ethereum.org/EIPS/eip-1450 (accessed on 14 August 2021).
127. Kupriianov, M.; Svirsky, J. EIP-1462: Base Security Token. 2018. Available online: https://eips.ethereum.org/EIPS/eip-1462 (accessed on 14 August 2021).
128. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018; pp. 1–15.
129. Androulaki, E.; De Caro, A.; El Khiyaoui, K. Making Fungible Tokens and NFTs Safer to Use for Enterprises. 2021. Available online: https://www.ibm.com/blogs/blockchain/2021/06/making-fungible-tokens-and-nfts-safer-to-use-for-enterprises/ (accessed on 27 July 2021).
130. Global Trade Item Number (GTIN). 2020. Available online: https://www.gs1.org/docs/idkeys/GS1_GTIN_Executive_Summary.pdf (accessed on 14 August 2021).
131. Prada-Delgado, M.Á.; Baturone, I.; Dittmann, G.; Jelitto, J.; Kind, A. PUF-derived IoT identities in a zero-knowledge protocol for blockchain. *Internet Things* **2020**, *9*, 100057. [CrossRef]
132. Schmid, G. Un procedimento di Anti-Contraffazione su Base Collaborativa. 2016. Available online: http://brevettidb.uibm.gov.it/download/177979616119184_W3_RM2014A000256-G07-4-28-023.pdf (accessed on 23 September 2021).
133. Authenticity Proofs. 2019. Available online: https://docs.provable.xyz/#ethereum-quick-start-authenticity-proofs (accessed on 27 July 2021).
134. SafetyNet Attestation API. 2021. Available online: https://developer.android.com/training/safetynet/attestation (accessed on 27 July 2021).
135. Introducing BOLOS: Blockchain Open Ledger Operating System. 2016. Available online: https://www.ledger.com/introducing-bolos-blockchain-open-ledger-operating-system (accessed on 27 July 2021).
136. Benet, J. IPFS–content addressed, versioned, p2p file system. *arXiv* **2014**, arXiv:1407.3561.
137. Breidenbach, L.; Cachin, C.; Chan, B.; Coventry, A.; Ellis, S.; Juels, A.; Koushanfar, F.; Miller, A.; Magauran, B.; Moroz, D.; et al. Chainlink 2.0: Next Steps in the Evolution of Decentralized Oracle Networks. 2021; p. 136. Available online: https://research.chain.link/whitepaper-v2.pdf (accessed on 1 August 2021).
138. Ellis, S.; Juels, A.; Nazarov, S. ChainLink—A Decentralized Oracle Network. 2017; p. 38. Available online: https://research.chain.link/whitepaper-v1.pdf (accessed on 1 August 2021).