Drawing the boundaries between Blockchain and Blockchain-like systems: A Comprehensive Survey on Distributed Ledger Technologies

Badr Bellaj [*] [$], Aafaf Ouaddah[*], Emmanuel Bertin [§](IEEE Senior), Noel Crespi[$](IEEE Senior), Abdellatif Mezrioui[*]
[$]Telecom SudParis, Paris, France, [*]Institut National des Postes et Télécommunications, Rabat, Morocco, [§]Orange, France
Corresponding e-mail: bellaj.badr@mchain.uk

***Abstract*—Bitcoin's success as a global cryptocurrency has paved the way for the emergence of blockchain, a revolutionary category of distributed systems. However, the growing popularity of blockchain has led to a significant divergence from its core principles in many systems labeled as "blockchain" This divergence has introduced complexity into the blockchain ecosystem, exacerbated by a lack of comprehensive reviews on blockchain and its variants. Consequently, gaining a clear and updated understanding of the diverse spectrum of current blockchain and blockchain-like systems has become challenging. This situation underscores the necessity for an extensive literature review and the development of thematic taxonomies.**

This survey seeks to offer a comprehensive and current assessment of existing blockchains and their variations, while delineating the boundaries between blockchain and blockchain-like systems. To achieve this objective, we propose a holistic reference model for conceptualizing and analyzing these systems. Our layer-wise framework envisions all distributed ledger technologies (DLT) as composed of four principal layers: data, consensus, execution, and application. Additionally, we introduce a new taxonomy that enhances the classification of blockchain and blockchain-like systems, offering a more useful perspective than existing works.

Furthermore, we conduct a state-of-the-art review from a layered perspective, employing 23 evaluative criteria predefined by our framework. We perform a qualitative and quantitative comparative analysis of 44 DLT solutions and 26 consensus mechanism, while discussing differences and boundaries between blockchain and blockchain-like systems. We emphasize the significant challenges and trade-offs encountered by distributed ledger designers, decision-makers, and project managers during the design or adoption of a DLT solution. Finally, we outline crucial research challenges and directions in the field of DLTs.

***Index Terms*—Blockchain, DLT, Consensus, Blockchain-like.**

I. Introduction

The blockchain space has rapidly evolved, starting with the introduction of Bitcoin [1] a decade ago, and progressing to the development of modern enterprise versions of DLT. Bitcoin or Bitcoin-based project such as Litecoin [2] and Peercoin [3] are often acknowledged as blockchain 1.0. The introduction of Ethereum in 2015, along with projects like IOTA [4], Hyperledger [5], and Solana [6], ushered in the era of blockchain 2.0. This phase witnessed significant deviations from Nakamoto's original blockchain design, giving rise to a distinct category of technologies. The emergence of this new category, deeply influenced by blockchain principles but not confined to them, has led the industry to market a more inclusive term: DLT when describing this particular category. Despite this shift, a lack of rigorously defined terminologies and a universally accepted taxonomy has resulted in confusion. Terms such as "blockchain," "DLT", or even "distributed database" have been subject to misunderstanding, misuse, and misinterpretation. Many projects and enterprises extensively employ the term "blockchain" as mere marketing jargon, further complicating the landscape. Despite several proposals aiming to standardize blockchain (ITU [7], ISO [8]-[9], IEEE [10]), there is currently no standardized recognized definition of blockchain or DLT. Consequently, varying opinions may arise regarding the extent to which a system qualifies as a blockchain. Additionally, a noticeable pattern is the rising use of unclear and inconsistent language in diverse projects. This has led to instances where the same term signifies different concepts. This linguistic discordance poses a potential hindrance to the development and widespread adoption of the DLT sector. Another ramification of this lack of consensus and standardization manifests as a deficiency in interoperability between DLT networks. The DLT ecosystem lacks interoperability as the DLT community, driven by intense competition and commercial pressures, focuses on the introduction of new systems that emphasize improved performance. Yet, the absence of a standard technical reference model makes it challenging to evaluate and compare these systems. In light of these challenges, our goal here is to establish clearer distinctions between different categories within the DLT ecosystem. We propose a new taxonomy designed to highlight the unique differences between different DLT groups. Additionally, we adopt a systematic and holistic approach to conceptualize and scrutinize DLTs as functional systems, emphasizing key layers across four levels of analysis. This comprehensive effort aims to provide a structured framework for understanding the diverse technologies within the evolving DLT landscape.

A. Motivation, aims and impact of the survey

1) Problem statement and motivations: At the time of writing this survey, the evolution of blockchain technology initiates an era marked by inconsistency and intense technical variations. In fact, there is a big number of blockchain-based projects under development. Some of which are sim-

ple replications of well-known projects, such as Bitcoin or Ethereum, whereas others propose entirely new functionalities and architectures. The current variations in blockchain systems pose a number of concerns from different perspectives, particularly concerning heterogeneity, coupled with the lack of interoperability, which may hinder the adoption of blockchains in our techno- and socio-economic systems. Furthermore, the diverse designs of DLT and their adaptable configurations pose a challenge for software architectures and developers when it comes to making informed decisions for constructing genuinely decentralized systems.Furthermore, like other technologies, *the lack of standards* harms privacy, security, governance, and more importantly, interoperability.
Several potential issues may arise, including but not limited to:

1) Hindering consistency in the formulation of of DLT regulatory laws and policies.
2) Causing confusion in the application of consumer protection laws and regulations.
3) Diminishing the precision of academic research aimed at exploring the foundational concepts essential for the development of innovative applications and solutions, thereby potentially impeding advancements in various fields.
4) Hindering the widespread adoption of the technology and its interoperability and integration with existing standardized technologies, thus hindering the utilization of blockchains/DLT and their potential applications.

In fact, any DLT or Cryptocurrency regulation or law begins by providing a correct legal definition of these elements. A DLT taxonomy helps establish that basis and assists regulators in crafting a flexible and granular DLT regulatory framework that considers the differences between existing projects rather than treating them as a single entity. For certain regulations such as eIDAS (electronic Identification, Authentication and Trust Services), the GDPR (General Data Protection Regulation), or Mica(markets in crypto-assets), the scope of applicability can differ from one category to another, depending on the specifications of a given DLT project (e.g., Data shareability). Thus, in the absence of a standard that helps regulators differentiate between different types of DLTs, confusion may arise regarding when to apply consumer protection laws and regulations. Furthermore, the visibility provided by the proposed framework makes it easier to integrate blockchain-based registries into other legislation.

2) Our proposed approach and methodology: A comprehensive solution to the aforementioned challenges involves proposing a reference model that delineates standardized structures, elements, and relationships. Such a model would serve as a framework for distinguishing authentic blockchain systems from those that exhibit blockchain-like characteristics. Establishing clear boundaries between these categories is imperative to establish a fair and equitable environment, facilitating the design and adoption of blockchain-based products or services by industry participants and community members. Drawing parallels with the Internet, where various standardization bodies (such as IETF in collaboration with W3C, ISO/IEC, ITU) continually define standards to enhance interoperability among systems and technologies, a similar approach is essential for the blockchain domain. By instituting blockchain standards, we can encourage the development and widespread adoption of interoperable blockchain and blockchain-like applications. Analogous to how standards for the World Wide Web contribute to interoperability, accessibility, and usability of web pages, blockchain standards would play a pivotal role in fostering the flourishing and widespread use of blockchain technologies. To achieve this goal, we conduct the following steps:

(1) Definition and Vocabulary Framework: An initial step involves scrutinizing the vocabulary and terms to clarify any ambiguities and resolve discrepancies. Conducting a literature review of existing technologies serves as the foundation for streamlining complexity and structuring information systematically. This process culminates in the development of a comprehensive vocabulary encompassing key blockchain terms. This established vocabulary serves as the groundwork for readers, enabling a better understanding of the subsequent classification and taxonomy presented in the analysis.

(2) Framework Setting Component and Property Identification: A review of the DLT literature by adopting a common multi-layered approach that inspects each key design separately. In fact, layering is a basic structuring technique used in different models such as TCP/IP, Open Systems Interconnection (OSI) [11], and Linux Systems. This layerization divides the studied systems into distinct layers to make it easier to understand and analyze. The structure of the different defined layers helps us to determine to which taxon a system belongs.

(3) Blockchain Classification: Finally, by examining the components and properties identified in each layer, we introduce and compare two categories. However, due to the continuous evolution of technology, these categories are expanding over time. Consequently, for the purpose of simplicity, our study focuses on two primary configurations for each component and property.

3) Results and impact of the survey: The outcome of the analysis at the component level yields a comprehensive blockchain taxonomy and a layered framework. This framework organizes major components hierarchically, elucidating their functional relationships and potential design patterns. This conceptual framework (DCEA) serves as a tool that assists DLT designers and decision-makers in analyzing the state of DLTs and their interactions for a comprehensive understanding of the DLT landscape and different proposed solutions. Moreover, the proposed framework helps DLT designers and DLT adopters build a structured vision of the proposed solutions in different DLTs and thus opt for the suitable design choice. In general, assessing the quality of a taxonomy or ontology proves challenging, particularly in dynamic domains such as blockchains. Taxonomies and ontologies are typically crafted to manage complexity and structure information, each

serving distinct purposes and undergoing evolutionary changes over time (as seen, for instance, in the evolution of the renowned Linnaean taxonomy in biology). Our taxonomy seeks to lay the groundwork for classifying diverse blockchain components, acknowledging that it does not claim to be the definitive structure. Nevertheless, the proposed taxonomy holds practical significance in various scenarios. It can:

(1) Support software architectures in exploring different system designs, evaluating, and comparing diverse design options;
(2) Serve as preparatory work for the development of blockchain standards, aiming to enhance the widespread adoption of blockchain-based solutions and services;
(3) Facilitate research into architectural frameworks for blockchain-based systems, fostering the adoption of blockchain-based systems through increased interoperability and compatibility.

B. Contributions

This paper significantly extends our previous works [12], [13], contributing in the following key ways:

1) Reference Model Establishment: The paper introduces a reference model that offers a comprehensive perspective on Distributed Ledger Technology (DLT) systems over time. This model categorizes current systems across four distinct layers: data, consensus, execution, and application, providing a thorough exploration of the state of the art.
2) DLT Taxonomy: A novel taxonomy of DLTs is proposed, challenging the distinctions between blockchain and blockchain-like systems. This taxonomy is based on different architectural configurations across the four layers defined in the reference model.
3) Comprehensive Comparison: The paper provides the first comprehensive comparison of a broad spectrum of DLT technologies, encompassing 44 projects and 26 consensus mechanism, offering valuable insights into their unique features and characteristics.
4) Consensus Mechanism Evaluation: A qualitative and quantitative comparison of various consensus mechanisms, including recent contributions, is conducted through the established framework. This evaluation enhances understanding and aids in decision-making for system designers.
5) Academic Achievements Summary: The paper summarizes practical academic achievements that have positively impacted the design and performance of DLTs. This inclusion provides a succinct overview of advancements in the field.
6) Exploration of New Trends: Emerging blockchain trends such as Blockchain modularity, zKvms, accounts abstractions, and others are presented and discussed. This forward-looking analysis identifies and explores evolving aspects within the blockchain landscape.

The paper will serve as a valuable guide for blockchain system designers, aiding them in making informed design choices for new DLT implementations. By encompassing a broad spectrum of DLT technologies, offering a novel taxonomy, and addressing current and future trends, the paper provides a comprehensive resource for both researchers and practitioners in the blockchain domain.

C. Paper's Organization

The structure of the survey is outlined as follows. Section II provides the necessary background, defines the adopted terminology, and offers a synopsis of the contextualized history of the Distributed Ledger Technology (DLT) evolution. In Section III, the proposed DCEA framework is introduced, outlining its layers and components. The section also elaborates on how this framework is employed to classify Distributed Ledger Technologies (DLTs). Sections IV, V, VI, and VII individually delve into the DCEA layers, addressing the data, consensus, execution, and application layers, respectively. Within each section, we provide an overview of the key components and properties of the examined layer, accompanied by an exploration of the latest developments in the field. Section VIII provides an in-depth comparative assessment and critical analysis of 44 diverse DLTs in academic and industrial settings. This evaluation employs the proposed referential framework and encompasses the examination of 26 consensus protocols. Section IX discusses the lessons learned from the reviewed literature. Section X focuses on the open challenges and research directions. Finally, we conclude with a summary in Section XI.

D. Comparison with existing surveys and tutorials

In the past few years, many studies have been conducted on reviewing and taxonomizing DLT technologies, covering multiple topics. Generally, we observe three types of surveys as shown in Table I. First, there are components-oriented surveys focusing on specific parts of DLTs, e.g., consensus mechanisms. Second, application-oriented surveys covering DLT applications in different domains; and third, surveys providing conceptual and holistic views of the DLT landscape. Our work belongs to the last category, even though none of the prior work has the scope or the width we adopt in our survey. A comparative summary of earlier similar works is shown in Table III.

Here, we provide an overview of some recent holistic research. [14] attempts to reflect the state of the art in the area of fully distributed digital currencies. However, it predominantly focuses on the Bitcoin protocol, its building blocks, and its applications, with a very short introduction to other Bitcoin-inspired projects. [15] proposes a taxonomy that captures a limited number of architectural characteristics of blockchains, namely: the level of decentralization, computation and client storage, as well as blockchain configuration. Besides, the authors provide only a brief surface-level review of concepts and notions needed to understand or classify DLTs. In [16], a good academic literature review on Blockchain technology is presented, with a focus on providing a non-exhaustive list of architecturally-relevant characteristics and quality attributes.

However, the authors define the blockchain as a distributed ledger technology without setting a clear distinction between the two.

Interestingly, [17] proposes a conceptual architecture, a taxonomy that distinguishes cryptoeconomic design (CED) from DLT, and a classification of 29 distributed ledger systems (DLT). However, the paper does not present a detailed overview of the underlying concepts of DLTs. Besides, the proposed taxonomy focuses more on CED and considers the blockchain as a mere data structure without a clear and systematic definition. Similarly, [18] introduced a comprehensive taxonomy of DLTs with extensive coverage of multiple relevant concepts that revolve around CED. The proposed taxonomy lacks a conceptual architecture that defines what a blockchain or a DLT is. More particularly, this taxonomy does not explicitly differentiate between CED and DLT or between different types of distributed ledgers.

[44] is a tutorial that explains the fundamental elements of blockchains using Ethereum as a case study. The paper considers comparing, at a high level, blockchains to traditional distributed systems. However, the paper compares both categories in a manner that hardly differentiates between a traditional distributed system and a permissioned blockchain—at least for many settings in which the private blockchain can be configured.

[45] proposes an exhaustive literature review on blockchain interoperability. The paper presents the necessary background and highlights definitions tailored for both industry and academia. It categorizes blockchain solutions into three categories considering solely the interoperability aspect.

[46] presents a comparative study of the applications and trade-offs of blockchain. The paper explains the architecture of the blockchain adopted by Bitcoin and Ethereum without considering other variants, e.g., IOTA or Hyperledger. Besides, most of the reviewed DLTs are briefly introduced without discussing their inner working mechanisms.

[47] provides a conceptual framework with a multi-layer abstraction of a blockchain. The authors propose a vademecum containing the information necessary to understand blockchain from a technical perspective, then introduce a decision model on which, when, and how to use blockchain technology. However, this paper covers a limited number of DLTs (7) and consensus mechanisms, making it an incomplete guide for readers looking to understand new DLT technologies operating in different modes than conventional blockchains.

In summary, the current state of the art on DLT system taxonomies suffers from a few limitations. First, the number of evaluated DLTs or consensus mechanisms across the papers varies significantly, from 0 to 29 for the DLTs and from 0 to 15 for the consensus mechanisms, and the provided evaluations are often superficial. Second, most papers adopt the same vision that often distinguishes real-world DLT systems based on their operation modes—private or public—or based on their access modes—permissioned or permissionless—which are insufficient characteristics to define whether a system is a blockchain or not. Third, none of the papers, except one, proposes a conceptual framework for DLTs and blockchain systems; none offers a systematic definition for fundamental notions like DLTs, Blockchain, DAGs (Directed Acyclic Graph), and more.

Table II presents a comparison between our proposed taxonomy and other existing taxonomies. Guided by the adopted criterion, our taxonomy surpasses the simplistic classifications found in the literature, which typically distinguish DLTs based on permissibility (permissioned or permissionless) or their public or private nature. Our taxonomy introduces a new perspective that aims to define what constitutes a blockchain and differentiates it from similar systems.

The framework used for taxonomizing DLTs has a positive impact on the DLT design process by facilitating decision-making through a systematic comparison of the capabilities of different design options. Furthermore, it illustrates the impact of these design choices on various quality attributes. The trade-off analysis of these quality attributes forms the foundational basis for effective comparisons.

In contrast to existing literature, our contribution entails a comprehensive, standalone tutorial offering a thorough exploration of the architecture underlying blockchain technology. Moreover, we propose a new taxonomy and classification that is highly comprehensive with a robust view of the DLT landscape based on a rigorous and systematic definition of what blockchain is and is not. Furthermore, we evaluate the largest number of DLTs (44) and consensus mechanisms (26) compared to other holistic surveys with a focus on covering the whole DLT landscape. Both the taxonomy and the evaluation are intended to help decision-makers architecting new blockchain-based systems or choosing among the existing solutions through enabling a systematic comparison between the capabilities of different design choices.

E. Survey's approach and methodology

To conduct our survey, we followed the five-step process outlined by Biolchini et al. [53] for elaborating a systematic review, as explained below:

Problem Formulation:

- RQ1: How can blockchain technology be defined within a unified layered-wise reference model?
- RQ2: What are the properties and components distinguishing each layer?
- RQ3: How can we distinguish a blockchain system from a blockchain-like one?
- RQ4: What obstacles and challenges does blockchain technology currently face?
- RQ5: What are the existing research gaps in the realm of blockchain technology?

Data Collection: After defining the layered-wise reference model, we gathered information for each layer from scientific literature, official documentation of reviewed projects, and reliable online sources (e.g., DLT experts' blogs), known as Grey Literature. This collection and selection were carried out using the systematic review protocol suggested by Kitchenham [54],

TABLE I
THREE BROAD TYPES OF SURVEY STUDIES REPORTED IN THE BLOCKCHAIN LITERATURE: COMPONENT-ORIENTED, APPLICATION-ORIENTED, AND HOLISTIC SURVEYS.

Survey Type	Subcategory	Paper	Year	Targeted topic
Component-oriented	Consensus protocols	[19]	2021	A survey and taxonomy of consensus protocols for blockchains
		[20]	2022	A survey and taxonomy of consensus protocols for blockchains
		[21]	2023	A Survey Paper on Blockchain Technology and Consensus Algorithms
		[22]	2017	A Taxonomization and evaluation of consensus protocols
		[23]	2019	A comprehensive survey of permissionless blockchain consensus protocols focusing on the underpinning incentive mechanism.
		[24]	2019	A self-complete tutorial on different types of distributed consensus protocols.
		[25]	2020	A survey taxonomizing blockchain consensus algorithms based on incentivization.
	Smart contract	[26]	2020	A survey on vulnerabilities and Attacks on Ethereum smart contracts
		[27]	2019	A survey on security, application and performance of smart contracts.
		[27]	2020	A survey on challenges, advances and platforms of Smart Contracts
		[28]	2020	A Survey of Smart contracts for blockchain-based reputation systems.
	Security and privacy	[29]	2020	A Survey on the attacks targeting the public blockchain
		[30]	2020	A survey examining of the security aspects within blockchain systems
		[31]	2023	A Survey on Blockchain Security and Its Impact Analysis
	Scalability	[32]	2020	A survey on the Scalability of blockchain
		[33]	2020	A survey on sharding in blockchains
		[32]	2020	A survey on the scalability solution in blockchains
Application-oriented	IOT	[34]	2019	A survey on Blockchains in Internet of Things
		[35]	2018	A survey on the application of Blockchain in Internet of Things
		[36]	2019	A survey on Blockchain for Internet of Things
		[37]	2020	A Applications in Blockchain Systems: Architecture, Consensus, and Traffic Modeling
	Different domains	[38]	2018	A Survey on Blockchain Applications in Different Domains
	Smart cities	[39]	2019	A Survey on the application of Blockchain Technology in Smart Cities
	Telecoms	[40]	2020	A state of the art survey on the application of Blockchain in 5G networks
	Cloud and Edge computing	[41]	2019	A survey on the application of Blockchain on Edge Computing Systems
	Blockchain and machine learning	[42]	2020	A survey on the application of Blockchain in machine learning
	Blockchain and artificial intelligence	[43]	2019	A survey on the application of Blockchain in AI
Holistic and conceptual	This category is presented separately in Table II			

involving three stages: (1) elaborating the search string; (2) applying the string on chosen search engines; (3) filtering out and extracting primary papers based on pre-established exclusion criteria from search results. The implementation of these steps is illustrated in a streamlined process model representing the methodology employed in this research, as shown in Figure 1. The reviewed projects from both the scientific and Grey literature are selected based on their notoriety and scientific significance. We assessed the quality and relevance of the sources from the grey literature within the exclusion criteria suggested by Garousi et al. [55]. These criteria represent 7 quality categories, ranging from the credibility of the producer to the objectivity of the study, as outlined in Table IV under the assessment of quality grey literature.

The choice of the 44 studied solutions was driven by the need to have good representativity of different components defined by the DCEA framework. For instance, we aimed to select DLTs that represent all different consensus mechanisms (26 different consensus mechanisms) and other DLTs representing other settings defined by the DCEA framework at the four layers.

- Data Evaluation: We systematically evaluate the DLT literature, highlight the pros and cons of each solution, and discuss how each solution matches the components and properties defined in each layer.

- Analysis and Interpretation Process: Within the same multi-layered approach, we conduct a qualitative and quantitative analysis of the surveyed solutions based on the 23 criteria (components and properties) defined in the first step.

- Conclusion and Presentation: Finally, we discuss the main remaining issues and draw future research directions and insights for the four defined layers.

II. HISTORY AND BACKGROUND

A. Contextualized History

The earliest identified occurrences of the concept of a 'blockchain' can be traced back to a 1991 research paper [56] entitled "How to Time-Stamp a Digital Document" by Haber and Stornetta, introducing the notion of a cryptographically secured chain of blocks. They proposed a timestamping system where the server would link a document to a previous document to avoid tampering with data. In 1992, they upgraded their design by introducing Merkle trees and collecting document certificates in blocks. Although their project has a different structure than that of the current blockchain system, they laid the basis for the modern blockchain.

Around the same time, cryptocurrencies were already emerging. In fact, David Chaum proposed "untraceable payments" in 1983 [57], in which he conceptualized an anonymous cryptographic system for processing digital money that could be transferred in the form of blindly signed coins. The

TABLE II
COMPARISON OF PROPOSED FRAMEWORKS FOR UNDERSTANDING AND UTILIZING BLOCKCHAIN TECHNOLOGY

Taxonomy	Focus	Classification Dimensions	Strengths	Limitations	State-of-the-art Review	Technical review	Solutions covered
A Taxonomy of Blockchain Consensus Protocols [48]	Consensus protocols	Fault tolerance model, Block creation method, Leader selection mechanism, Scalability	Technical precision and depth of analysis	Limited to consensus protocols	Partially - Focuses on consensus protocols	Yes	28 consensus protocols
A Taxonomy of Centralization in Public Blockchain Systems [49]	Centralization levels in public blockchains	Application, Contract, Consensus, Incentive, Network and Data layer	Captures various aspects of centralization and evaluate multiple research papers	Emphasizes the assessment of research papers rather than concentrating on Blockchain projects. focuses on evaluating research papers rather than Blockchain projects	Partially - Focuses on centralization aspects	No	2 blockchain projects
Taxonomy of Blockchain Technologies. Principles of Identification and Classification [50]	Standardization of blockchain architectures	consensus, sharing and rewarding systems, Transaction capabilities, Native currency/Tokenization, Extensibility, Security and Privacy, Code Base and Identity Management	Bottom-up approach, Component-based taxonomy	Early stage analysis, Potential complexity, Focus on software architecture, Limited discussion of existing projects	No - Engages with research and standardization discussions, but doesn't explicitly compare specific frameworks or solutions	No	None
A Taxonomy For Governance Mechanisms Oftowards A Taxonomy For Governance Mechanisms Of Blockchain-Based Platforms [51]	Blockchain governance structures	Centralization level, Decision-making process, Stakeholder participation	Analyzes different governance models	Limited to blockchain governance	Partially - Focuses on formal governance structures	No	None
A Taxonomy for Characterizing Blockchain Systems [52]	Composition of blockchain systems	Platform, P2P Network, Distributed Ledger, Smart Contract, Consensus Protocol, Digital Wallet, Token and Network Node	Comprehensiveness, Hierarchy, Software Development guidance, Case Studies	Complexity, limited discussion and comparison, The layer-based taxonomy may result in overlapping classifications, complicating information organization	Partially - Offers a restricted overview of the state-of-the-art, focusing on whether a layer or component exists in a solution without providing detailed explanations.	Yes - Provides a high-level technical review and doesn't cover consensus mechanisms.	10 blockchain projects and 11 consensus mechanism
A Vademecum on Blockchain Technologies [47]	Practical guide to blockchain decision-making	Decision factors, Use cases, Platform selection, Development	Accessibility to a broad audience	Focuses primarily on the decision-making process for adopting blockchain technology, not a comprehensive analysis of the technology itself. The provided analysis focuses on only seven projects	No - Offers practical insights, not exhaustive review	Yes	7 blockchain projects
A Taxonomy of DLTs (Our Work)	Reference model for DLTs (blockchain and blockchain-like)	Data, Consensus, Execution, Application layers	Comprehensiveness, Component-based taxonomy, Inclusion of Blockchain and Blockchain-Like Systems, A 360° overview of the technical and practical dimensions of DLTs, Technical review of each layer of the DLT	Analysis limited to 44 selected DLT projects	Yes - Comprehensive examination of the state-of-the-art, considering both technical and design perspectives.	Delve into the technical details of different blockchain platforms or specific implementation challenges.	44 projects blockchain and 26 consensus mechanism.

proposed blind signatures provide a cryptographic means to prevent linking users to coins and to enforce fungibility, while still allowing a central server to ensure protection against double-spending. Later, Chaum proposed DigiCash [58] in 1990, as the first global electronic cash system. DigiCash gained some interest, but its centralized nature led to its failure and stoppage in 1998.

In 1997, the Hashcash [59] project proposed a cryptographic hash-based algorithm called proof-of-work to protect against denial of service, a concept that will be used later in Bitcoin [1] as a fraud countermeasure. In 1998, Wei Dai proposed "b-money" [60], characterized as a pseudo-anonymous, distributed electronic cash system with a replicated, transparent, and public ledger. Several years later, in 2005, Nick Szabo described Bit Gold; a decentralized cryptocurrency inspired by Hashcash. Bit Gold was notable for using an underpinning system similar to the current blockchain. It included many of the basic components of modern cryptocurrencies, such

TABLE III
COMPARISON OF RELATED COMPREHENSIVE AND HOLISTIC SURVEYS

Paper	Publication year	Conceptual framework	Taxonomy of DLTs	Number of DLTs attributes	Evaluated DLTs	Evaluated consensus	Limitations	Solutions	Surveyed layers*	DLTs categories	Applications
[45]	2020	No	No	18	21	0	Yes	Yes	C,E	No	Yes
[44]	2020	No	No	17	9	5	Yes	Yes	C,E	Yes (Distributed Database, Blockchain)	No
[47]	2019	No	Yes	19	7	15	Yes	Yes	D,C,E	No	Yes
[46]	2019	No	Yes	11	7	5	Yes	No	D,C	No	Yes
[17]	2018	Yes	Yes	10	29	0	No	No	D,C,A	Yes (Cryptoeconomic design, distributed ledger)	No
[16]	2018	No	No	13	0	7	Yes	Yes	D,C,E	No	Yes
[15]	2017	No	Yes	4	0	4	Yes	No	D,C	No	No
[18]	2017	No	Yes	25	0	6	No	No	D,C,E	Yes (Cryptoeconomic design, distributed ledger)	No
[14]	2016	No	No	12	0	5	Yes	Yes	D,C,E	No	Yes
Our work	2020	Yes	Yes	25	44	26	Yes	Yes	D,C,E,A	Yes (Blockchain, Blockchain-like)	Yes

* *D* : Data *C*: Consensus *E* : Execution *A*: Application.

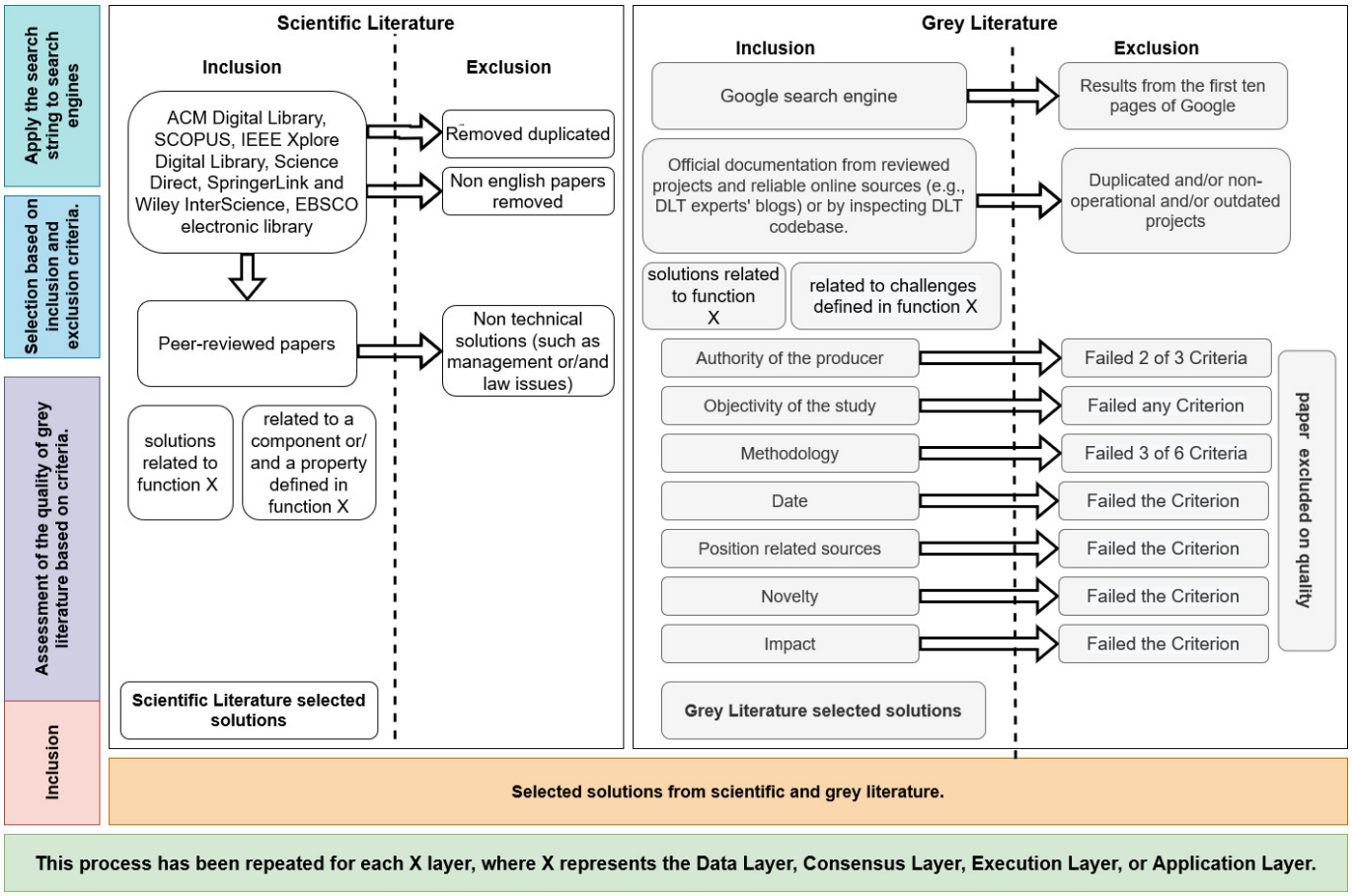

Fig. 1. Search and selection strategy, depicting a process model for a single layer.

as decentralization, anonymity protections, public registry, a proof-of-work mechanism, chains of hashes, and time-stamping. However, Bit Gold was never implemented, and its design suffers from the double-spending problem, wherein users can spend a coin twice in the network. Nick Szabo is also renowned for a major contribution in the blockchain field. He proposed (in 1997) the concept of smart contracts, which enables counterparties to formally codify a cryptographically enforceable agreement without the need for any third party — a concept that can be considered a predecessor of Bitcoin's scripting mechanism.

Despite the vast advancement in cryptographic-based currencies, the problem of proposing a decentralized network resistant to the double-spending problem remained unsolved until 2008, when Bitcoin was introduced by the anonymous person or persons, Satoshi Nakamoto, in "Bitcoin: A Peer-to-Peer Electronic Cash System" [1]. One year later, Bitcoin network was widely deployed, with the genesis block mined on or around January 3, 2009. Bitcoin presented a convincing solution for preventing double spending in a global decentralized system without depending on trust in any third party, using a mix of techniques developed in earlier projects. Since then, multiple alternative cryptocurrencies with different features have been created, such as Ethereum [61] and IOTA [4].

In 2014, motivated by the decentralized automation of Bitcoin, the blockchain technology was separated from the

TABLE IV
QUALITY CRITERIA GREY LITERATURE

Category	Exclusion Criteria	Criteria to Satisfy
Authority of the producer	The publishing organization is reputable, or the individual author is associated with a reputable organization The author has published other work in the field The author has expertise in the area (e.g. job title)	2/3
Objectivity of the study	The statement of the sources is objective There are no vested interests Conclusions are supported by data	3/3
Methodology	The source has a clearly stated aim The source has a clearly stated methodology The source is supported by authoritative, documented references Limits are clearly stated The work covers a specific question The work refers to a particular population	4/6
Date	The item has a clearly stated date	1/1
Position related sources	Key related GL or formal sources have been linked/discussed	1/1
Novelty	The item enriches or adds something unique to the research The item strengthens or refutes a current position	1/2
Impact	The GL source should have citations and backlinks to substantiate the arguments made in the study	1/1

currency, mostly by the media and the industry, in an attempt to harness the power of Bitcoin's underlying technology and its potential for other purposes. One year later, global financial companies formed R3, a consortium of 42 institutions [62] with an agenda of exploring and harnessing blockchain and DLT technology for financial, inter-organizational transactions with a blockchain-like product called Corda [63]. Around the same time, the Linux Foundation launched Hyperledger [64] as an umbrella project of open-source DLT solutions to encourage the use of blockchain to support global business transactions.

B. Terminology and Background

In the blockchain literature, there is significant overlap between the terms DLT and blockchain. The situation is made even worse by the absence of a universal standard defining a blockchain and its boundaries, which results in conflicting use of the terminology and different meanings being assigned to the same terms. The precise definition of multiple notions presented hereafter may be controversial, but for a better understanding and categorization of blockchain and similar systems, we provide here the terminology we have chosen to adopt throughout this paper.

- **Nodes:** Nodes are entities, maintained by individual participants of the network, that issue and validate transactions in a DLT network.
- **Ledger:** The term ledger refers to the single storage endpoint hosted by a node in a blockchain or DLT network. It represents a local append-only log of ordered transactions validated by a distributed network.
- **DLT:** An umbrella term for distributed ledger technologies, representing technologies operating as replicated, shared and synchronized ledgers throughout a distributed network, whereby parties who do not fully trust each other maintain consensus about shared information. A DLT, as a data structure, can adopt some of the design principles of blockchain technology whilst dropping others and might be governed in a partially or fully centralized manner. Although blockchain is initially a distributed ledger, the term DLT is now used frequently to separately identify technologies which do not embrace the full Nakamoto vision. To the best of our knowledge, the DLT term was initially adopted, in the industry, for commercial purposes to differentiate [63] some solutions (e.g. Corda or Hyperledger Fabric) from traditional blockchain systems like Bitcoin.
- **Blockchain:** A distributed system that shares fundamental architectural principles laid out by Nakamoto in Bitcoin's initial paper. That is, we can define a blockchain as a replicated database, managed by a consensus mechanism, solving the double spending problem, in a peer-to-peer fashion and shared by non-trusting parties. The inner database presents special key properties such as a chain of blocks, data structure, data immutability, global data shareability, the use of cryptography to secure data and ensure direct ownership, and interaction through transactions. The core fundamental characteristic of a blockchain is its fully or mostly decentralized and censorship-resistant nature. Initially, the term blockchain (as a single word) did not appear in Bitcoin's initial paper, but instead the term "chain of blocks" was used.
- **Blockchain-like:** A general term encompassing distributed ledger systems that are constructed in a similar manner to the conventional blockchain system (Bitcoin), while not necessarily conforming to Nakamoto's vision (Bitcoin's white paper [1]).
- **State:** While state can have different meanings depending on the context, the use of state within the blockchain and distributed ledger environment refers to the atomic unit of data stored in the ledger, where it is prone to change under the execution of validated transactions. As described by Gavin Wood in the Yellow Paper [61], a blockchain is a cryptographically secure, transactional singleton machine with shared state.
- **P2P gossip:** DLT networks are built upon P2P networks that transport all data needed to support the DLT protocol. In a blockchain network, nodes are equally privileged, equipotent participants in the application, whereas a Blockchain-like system is usually a hybrid combination of a peer-to-peer network and centralized entities where some nodes are more privileged than others. Blockchain's P2P networks are structured [65] or unstructured [1] networks where nodes randomly connect to other nodes. Data is exchanged directly over the underlying TCP/IP network implemented as multiple TCP unicasts between connected nodes, but at the application level, nodes are able to communicate with each other directly, via the virtual overlay connections built upon the underlying physical network. These virtual overlay networks facilitate node discovery and indexing, besides they make the P2P system independent from the physical topology. In order to converge on an eventually consistent sequential log of transactions. Many blockchain and blockchain-like systems adopt gossip protocol [66][67] whereby nodes exchange notifications about new data (a transaction or a block) [68]. At a high level, a node first discovers and learns about a set of peers (a list of nodes it knows), then it constantly advertises each new data it has received

and validated, to his peers announcing its availability. Afterward, the peers who do not have the advertised data, in their turn, request the missing data avoiding thus broadcasting data directly to the nodes [68] [69]. The gossip process continues until all reachable nodes receive the new data.

- **On-chain/off-chain:** We consider the terms 'off-chain' or 'off-ledger' to refer to actions that occur outside the formal boundaries of the transactional ledger distributed between the nodes. Conversely, 'on-chain' or 'on-ledger' refers to actions that occur within the boundaries of the distributed ledger.

Blockchain and DLT are often mistakenly conflated: a blockchain is a sub-class of DLTs, but a distributed ledger does not always comprise a blockchain. Thus, in the rest of this paper, we use the term DLT as a unifying term covering blockchains and blockchain-like systems.

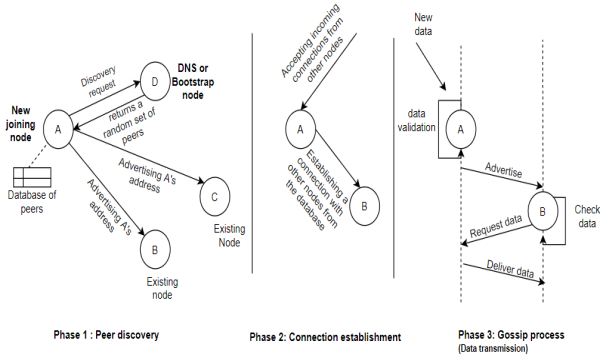

Fig. 2. Communication phases in a blockchain P2P network (Bitcoin)

III. DCEA Framework: A Taxonomy-Oriented Approach to Conceptualize and Explore Distributed Ledger Technologies (DLTs)

We present the DCEA framework, a structured model defining a layered and diverse stack tailored for the intricate realm of Distributed Ledger Technologies (DLTs). It's application organizes DLTs into 4 fundamental and distinct layers: data, distributed consensus protocols, execution, and application layers. Each layer serves a precise and integral function within the overarching DLT architecture.

Utilizing the DCEA framework results in a nuanced dual-level classification of Distributed Ledger Technologies (DLTs). The primary classification is rooted in the varied configurations of key designs, while the secondary classification involves a comprehensive two-dimensional taxonomy, distinguishing between blockchain and blockchain-like technologies. This taxonomy evaluates the impact of different DCEA settings at 3 levels (refer to Figure 3). The results of the exhaustive review and taxonomy, outlined in Sections IV, V, VI, and VII, are applied in Section VIII to scrutinize and categorize a diverse spectrum of DLTs into the two overarching categories.

A. Introduction to the DCEA Framework

The DCEA framework encompasses the components of DLTs along with their key properties, as illustrated in Table V. The components are logically organized, aligning with the modular architecture of DLTs. This layered structure enhances the comprehension of DLTs and establishes a foundational basis for comparative analyses across various DLT variants. Below, we present the four layers constituting the DLT stack.

- Data Layer: encompasses the representation, storage, and flow of data within the distributed network. This layer deals with the transactions and states recorded in the ledger, forming the core information infrastructure of the blockchain. The Data Layer ensures that entries in the ledger are recorded under consensus, meaning all participants in the network agree on the validity of the data. These records can be elements built-in the underlying DLT protocols (e.g., cryptocurrency or Tokens or smart contracts states) or data sourced from external environments (e.g., IoT data). This layer encompasses both on-chain storage (stored directly on the blockchain) and off-chain storage (utilizing a distributed database).
- Consensus Layer: Establishes a set of rules through software definition to reach consensus between network participants, ensuring alignment on a unique ledger (source of truth). This layer defines governance mechanisms that govern the validation, addition, and verification of transactions. The critical role of the Consensus Layer merits detailed exploration, which is extensively covered in Subsection VIII-B.
- Execution Layer: Serves as the computational environment where distributed programs, including smart contracts, are executed. These programs encode specific logic (e.g., business logic) into programmatic instructions which are executed to manipulate the states recorded in the ledger. In essence, the Execution Layer acts as the engine that interprets and processes these instructions, facilitating the decentralized execution of logic encoded in smart contracts. It plays a pivotal role in ensuring the integrity and automation of transactions within the distributed ledger, providing a secure and transparent framework for the execution of programmable business logic across the network.
- Application Layer: Refers to the topmost layer of the technology stack, serving as an abstraction layer that provides protocols and APIs (Application Programming Interfaces). This layer is designed to enable the development and execution of distributed applications, commonly known as DApps (Distributed Applications). It acts as a bridge between external entities, such as end-users or other applications, and the underlying code and data stored on the blockchain ledger.

Built upon the layered architecture discussed earlier, we introduce a comprehensive four-layered taxonomy (Figure 5) designed to categorize DLT systems. This taxonomy serves a dual purpose:

TABLE V
LAYERS AND COMPONENTS OF DCEA FRAMEWORK

Application Layer	Integrability			DLT orientation and purpose			Wallet and identity management		
Execution Layer	Execution environment		Turing-completeness		Determinism		Openness	Interoperability	
Consensus Layer	Safety	Liveness	Finality	Network model	Failure model	Adversary model	Governance model	Transaction ordering	Conflict resolution
Data Layer	Data structure		Data shareability		Data immutability		States storage		

- To classify the various DLT systems presented in academic and industrial domains.
- To assess the inherent strengths and weaknesses of existing systems, pinpointing deficiencies within the current landscape of DLTs.

At each layer within the DCEA framework, DLTs encompass diverse configurations of DCEA properties detailed in Table V. Through the examination of these property combinations across the four layers, distinct classes of DLTs can be characterized. For example, distinctions at the data layer arise between Directed Acyclic Graph (DAG) and chain based DLTs. Also, the consensus layer facilitates categorizations such as permissioned and permissionless DLTs, contingent on the identity model of the consensus mechanism. The execution layer enables the differentiation between Smart-contract and script based DLTs. Meanwhile, at the application layer, classifications emerge, including DApps-oriented DLTs and Cryptocurrency-oriented DLTs. This comprehensive analysis results in the classification of DLT systems into two major categories: blockchain and blockchain-like systems.

B. Distinguishing Between Blockchain and Blockchain-Like Systems

In scrutinizing DLT systems through the lens of the DCEA framework and discerning the disparities between the two overarching taxonomic categories, blockchain and blockchain-like, we identify shared commonalities alongside distinctive traits (refer to Table III-A). Taking a broader perspective, we assert that a system falls under the blockchain-like category, rather than being a blockchain, if it exhibits at least two of the following attributes, as depicted in Figure 3. First,

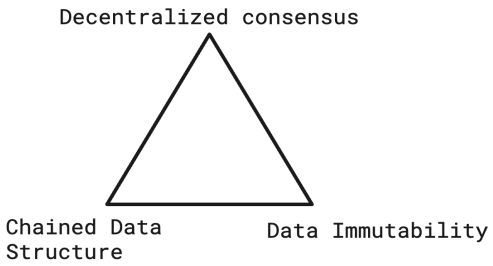

Fig. 3. According to our definition, a system resembling a blockchain can only possess two of these specified traits.

a noticeable lack of robust decentralization—several DLTs may opt to compromise decentralization for reasons such as enhanced scalability, streamlined governance, or suitability for deployment in contexts where full decentralization is unnecessary. Second, a blockchain-like system permits data tampering and provides weak immutability for either states or transactions. Third, its data structure does not rely on the chaining of blocks of transactions to store data.

Caption: Table III-A summarizes distinctive settings within each category, aiding the differentiation between blockchain and blockchain-like DLTs. Despite not being entirely separate, these categories often exhibit significant overlap. It is possible to find a system with blockchain-like features, encompassing all properties except one or two. Moreover, our distinction is not contingent on operational settings—whether public or private, permissioned or permissionless—as the demarcation between blockchain and blockchain-like is not bound by these factors. For instance, a project initially designed for public and permissionless settings (e.g., Ethereum) can seamlessly transition to private and permissioned settings, and vice versa. In Table XII, we present a classification of various DLT technologies as either blockchain-like or blockchain, based on our established criteria.

In Figure 4, we present a graphical dichotomous key aimed at facilitating the differentiation between the two categories of DLTs. In Figure 4, we present a graphical dichotomous key to facilitate the differentiation between the two categories of DLTs.

The methodology adopted first deconstructs any DLT (blockchain or blockchain-like system) project into four layers that encompass different components. Second, it evaluates the different components to define the design choices adopted at the four levels. Third, based on the distinctive settings defined for the different components at each layer, as presented in Table IV, we classify a system as either a blockchain or a blockchain-like system. For instance, Bitcoin is classified as a Blockchain since it exhibits all the settings (Table IV) used to define a system as such. Hyperledger Fabric, on the other hand, is classified as a blockchain-like system since it displays some distinctive settings adopted for defining a system as such. Specifically, at the data layer, Hyperledger Fabric does not adopt a chainless model, and it does not provide strong immutability. Moreover, at the consensus layer, it adopts a centralized governance with a leader-based consensus mechanism.

TABLE VI
CONFIGURATIONS OF BOTH BLOCKCHAIN AND BLOCKCHAIN-LIKE SYSTEMS IN THE DCEA FRAMEWORK

	Components and properties	Blockchain	Blockchain-like
Data Layer	Data structure	Chain of block	Chainless model
	Shareability	Global	Restricted by design
	States management	On-chain	Off-chain
	Immutability	Strong	Weak
Consensus Layer	Consensus identity model (membership)	Permissionless	Permissioned
	Governance	Democratic, Oligarchic	Dictatorship, Oligarchic
	Data ordering	Decentralized and open	Centralized or reserved
Execution Layer	Conflict resolution	Longest-chain/ No-Forks	Longest-chain/No-Forks
	Turing completeness	Turing/Non-Turing complete	Turing and Non-Turing complete
	Openness	Closed/Oracle-based	Open/Oracle-based
	Interoperability	Non-interoperable/Interoperable	Non-interoperable/Interoperable
	Determinism	Deterministic	Non-deterministic
	Execution environment and rules enforcement	VM, Script runtimes	VM, Script runtimes
Application Layer	Integrability	High, Medium, Low	High, Medium, Low
	DApp orientation	DApps, Cryptocurrency	DApps/ Cryptocurrency
	Wallet management	Built-in	Built-in or External

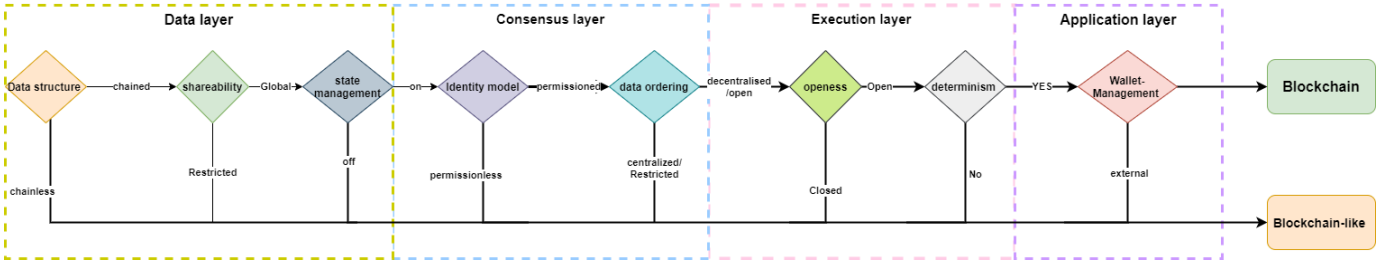

Fig. 4. A graphical dichotomous key for distinguishing between blockchain and blockchain-like systems

Following the comprehensive analysis of DLTs across the four primary layers in sections IV, V, VI, and VIII, we present and discuss the proposed taxonomy by highlighting distinguishing properties. Additionally, we offer an in-depth exploration of the current state of the art at the data, consensus, execution, and application layers, respectively.

IV. DATA LAYER

In this section, we delineate the fundamental components and their respective characteristics that form the data layer, as outlined in Table V. Additionally, we provide an overview of the current state of the art regarding various data structures employed by prominent DLT solutions. The section concludes with an examination of key challenges and trade-offs inherent in the design of the data layer.

A. Components and properties

The ledger in DLTs essentially functions as a distributed ”source of truth”, with data is replicated and synchronized across multiple nodes. The macroscopic organization of data structures differs among DLT technologies, primarily falling into two main models: the linear chain of blocks and chain-less models.

1) Data-Structure:

a) : Linear Chain Model

Chain of Blocks In the chain of blocks model, transactions are bundled into blocks that are chained together in chronological order, creating an unalterable history of data. Each block is cryptographically linked to its predecessors, ensuring the integrity of the entire chain and preventing unauthorized modifications. A block comprises a header and a transaction record, as illustrated in Figure 6.The header’s inclusion of a timestamp and a cryptographic reference to the previous block empowers all network participants to independently verify the block’s authenticity and its rightful place in the chain. This decentralized validation process eliminates the need for a central authority, fostering trust and transparency. At the transaction level, data is conveyed and stored in the ledger, with transactions representing the fundamental data type. Transactions are organized and hashed into a Merkle tree at the block level, and the hash root is embedded in the corresponding block’s header. This architecture guarantees a cryptographically sealed and tamper-resistant data repository, providing an immutable record of transactions. The cryptographic links between blocks, enforced by the hash pointers makes it computationally infeasible to alter any part of the data without affecting the entire chain, thus safeguarding against unauthorized modifications. This robust design contributes to

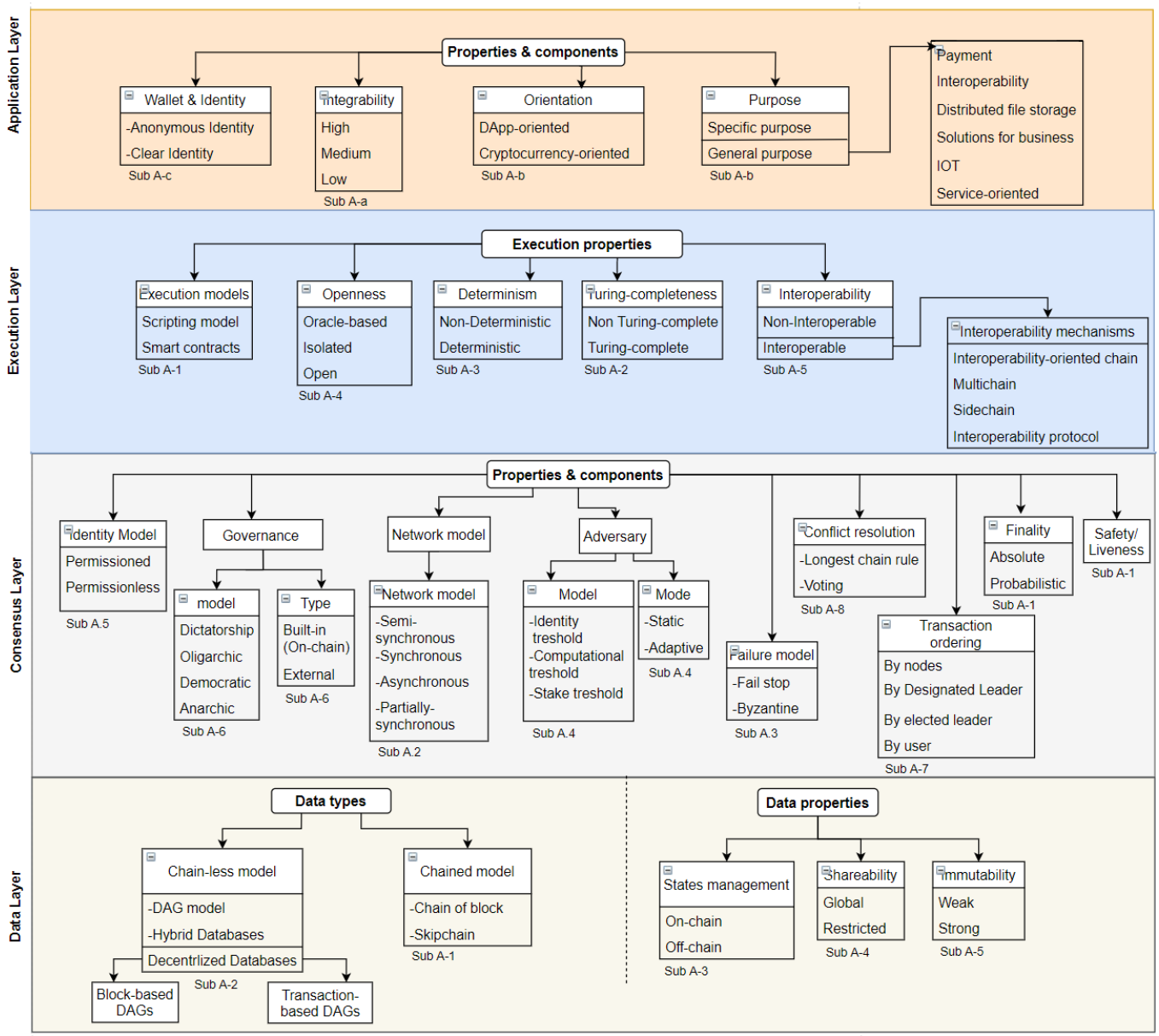

Fig. 5. Layered taxonomy of DLT systems (With references to the corresponding sections and subsection)

the trustworthiness and reliability of the blockchain, fostering transparency and accountability in data management.

Skipchain Skipchain derives their data structure inspiration from skip lists [70]. Skip lists are characterized by multiple linked lists arranged in layers with varying skip distances. This architectural influence enhances the efficiency of blockchain operations in a Skipchain by selectively engaging a subset of nodes in block validation. This hierarchical structure improves efficiency and scalability in consensus processes, streamlining the validation of new blocks. In addition to its unique consensus mechanism, a Skipchain employs an organized data management approach. The skip list-inspired structure allows for expedited searches and verification, enhancing the overall performance of the blockchain. This adaptation reduces the computational burden on individual nodes, contributing to faster transactions and improved scalability while maintaining the integrity and security of the distributed ledger.

b) Chainless model: Some DLTs break away from the conventional chain of blocks, opting instead for non-linear data structures like DAGs to achieve performance gains.

Direct Acyclic Graph The DAG is essentially a graph without cycles, progressing in one direction. It serves as a meticulously woven network of interconnected transactions, where each transaction (or block of transactions) functions as an individual node or vertex in the graph. This structure maintains a chronological growth pattern, resembling the organic expansion of branches from a continually developing tree rooted in the initial node. The distinctive characteristic of

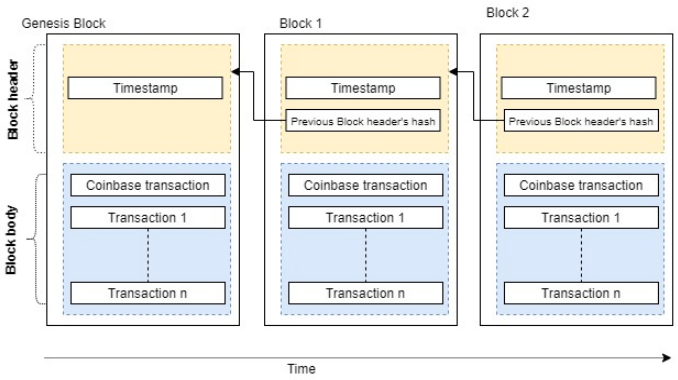

Fig. 6. Basic chain of blocks data structure

the DAG lies in its cyclical-free and unidirectional growth, allowing for automatic transaction confirmation based on preceding transactions within the network. DAGs are further categorized into two types based on the nature of their nodes:

- **Transaction-based DAGs:** Individual transactions are depicted as nodes within the DAG. Transactions are linked through directed edges, establishing a clear order of execution.
- **Block-based DAGs:** Nodes signify blocks containing multiple transactions, resembling traditional blockchains. Transactions are aggregated into blocks before integration into the DAG.

Decentralized database model decentralized databases distribute the responsibility of data maintenance across multiple nodes, allowing various parties, even those with mutual distrust, to collaboratively contribute to and synchronize shared records. This model leverages novel technologies to enable a distributed, secure, and transparent framework for managing data. Data is typically stored in a distributed fashion across the network of nodes, employing cryptographic techniques to ensure integrity and security.

c) Hybrid data model: An Hybrid Data Model, integrates both blockchain and block-less models to govern transactions and states within the network. This hybridization is implemented to harness the strengths of each model, facilitating enhanced scalability and swift transaction validation. In this framework, states are typically stored in external dedicated key-value databases, while the blocks exclusively contain transactions impacting the ledger's states. The use of key-value databases simplifies direct access to the updated value of a state, eliminating the need to traverse transaction trees for calculation. This duality of blockchain and key-value databases represents a pragmatic approach, combining the security of the blockchain with the efficiency of direct state access, thereby optimizing the overall performance of the distributed ledger system.

Distributed Hash Table A Distributed Hash Table (DHT) is a decentralized, distributed infrastructure that provides a scalable and fault-tolerant method for organizing and retrieving key-value pairs in a network. In a DHT, the keys and associated values are distributed across multiple nodes, allowing for efficient decentralized storage and retrieval of data. Each node in the network is responsible for a specific range of keys, and the DHT algorithms ensure that nodes can efficiently locate the node responsible for a given key. DHTs are commonly used in peer-to-peer (P2P) networks, distributed file systems, and other decentralized applications to enable efficient and resilient data storage and retrieval across a network of interconnected nodes.

2) State Management: DLTs offer various approaches to managing data within their systems. While they all share a distributed ledger for recording transactions, they diverge in how they store the actual data itself. Some DLTs, like traditional blockchains, keep data directly on the shared ledger (*on-chain/on-ledger*), while others opt for off-chain storage in separate databases (*off-chain/off-ledger*).

When examining the management of general states, such as user balances, in existing DLTs, three prevalent models come to the forefront:

1) UTXO Model: In the UTXO (Unspent Transaction Output) model, transactions are constructed as a web of interconnected links, where each new transaction (UTXO) (*new UTXO*) explicitly references the previous transactions that it draws upon as inputs (*inputs*). This structure creates a traceable path of ownership and spending history and ensures that each UTXO can only be spent once, preventing fraudulent activities.
2) Account Model: The account model resembles a traditional bank ledger, maintaining a central record of individual account balances. Transactions update these balances directly, offering a familiar and straightforward approach for managing user-specific data.
3) World-State Model: Represents a design where the current state of the ledger is stored and managed as a single, comprehensive snapshot. This model separates the current state from the transaction log, allowing for efficient querying and retrieval of the most recent data.

A complete review of the UTXO model can be found in [71].

3) Data Shareability in DLT Networks: Within a DLT network, the exchange of transactions among all participating nodes is central to achieving consensus. However, the concept of data shareability varies across different DLT systems, often influenced by privacy considerations. Two predominant visions of data shareability emerge within this landscape.

Global Shareability: In certain DLT systems, a philosophy of complete shareability is embraced. This approach, which we refer to as *global shareability*, advocates for an unrestricted exchange of all data among all nodes in the network. In this model, transparency and accessibility are paramount, with every participant having access to a comprehensive and unfiltered view of the shared data. This promotes a decentralized environment where information is universally accessible, fostering a high degree of transparency and collaboration.

Restricted Shareability: Contrastingly, some DLT systems adopt a vision of restricted shareability. In this model, the

perimeter of data shareability is delineated, encompassing specific nodes while excluding others. This restricted approach is motivated by privacy concerns, allowing for a more selective dissemination of information. Nodes within this restricted shareability model have limited access to certain data, ensuring a controlled flow of information based on predefined criteria. While this approach may limit universal transparency, it provides a mechanism to tailor data access to specific needs, enhancing privacy and security.

4) Data immutability / Atomicity: There is a common belief is that records in DLT (especially a blockchain) are inherently immutable and impervious to alteration. However, this assumption requires clarification, as the degree of immutability varies across different DLT systems based on their design. Consequently, in some DLTs nodes may retain inconsistent states or witness the rollback of transactions and states after confirmation. For data immutability, we differentiate between:

- Strong Immutability: Strong immutability is the property of a system where data, once created and defined, is permanently unalterable. No operation, actor, or event can modify the data after its initial creation. This ensures absolute, tamper-proof preservation of historical records and guarantees data integrity through all time.
- Weak Immutability: Weak immutability allows data to be updated under defined and controlled conditions. Modifications follow specific rules and governance mechanisms, ensuring transparency and accountability in the change process. This balance preserves historical records while enabling necessary adaptations.

It's noteworthy that in certain systems characterized by strong immutability, updates to their states can occur without compromising this immutability. This is accomplished through the use of tree structures, allowing for the persistent storage of old and new values associated with a specific entry (e.g, Ethereum state structure 7).

B. State-of-the-Art in Data Layer

This subsection aims to provide a comprehensive overview of DLT projects that align with the diverse data structures delineated in our framework. Additionally, it entails an evaluation of the properties associated with these projects.

1) Data-structure: 1.1) Chained DLTs

Many DLTs adopt the foundational linear data chain structure initially introduced by Bitcoin. Within this broad category, data is stored in inter-linked blocks; however, various projects within this domain introduce distinct inner block structures to tailor the architecture to their specific needs and objectives.

a) Bitcoin: In the Bitcoin ecosystem and its derivatives, transactions are assembled within the block's body and are subsequently linked in a Merkle tree. The Merkle root is a unique cryptographic hash created by combining the hashes of all the data blocks in a Merkle tree. Beyond the Merkle root, the block header encapsulates the hash of the previous block. Beyond the Merkle root, the block header encapsulates additional critical information, including the hash of the previous block. This hash of the previous block is crucial for maintaining the chronological order and integrity of the entire blockchain. It creates a cryptographic link between the current block and the one that preceded it, establishing a continuous and tamper-resistant chain of blocks. This interconnection of blocks through their headers ensures the immutability and trustworthiness of the Bitcoin blockchain. Notably, Bitcoin relies on the UTXO model's distributed tracking of spendable value within the blockchain, enabling the inference of wallet balances through UTXO analysis. Nodes often cache frequently accessed UTXOs in memory for faster retrieval, improving transaction processing efficiency.. This approach contributes to the efficiency and scalability of the Bitcoin blockchain.

b) Ethereum: Ethereum employs a state tree as a meticulously organized database, mirroring the account model to oversee account balances, smart contract data, and other critical network states. This tree-like structure relies on a specialized data structure known as a Merkle Patricia Tree (MPT), facilitating efficient retrieval and verification of information to maintain synchronization across all nodes. This highly efficient, self-balancing tree structure combines the strengths of Merkle trees and Patricia tree data structures. Each node in the MPT represents a key-value pair, storing the name (key) of a state variable and its corresponding value. Nodes have 16-byte keys and contain either a leaf value (for end nodes) or 16-byte hashes of child nodes (for intermediate nodes). As a state variable changes, a new node is added to the tree to reflect the updated value 7. Recalculation of the root hash captures the change, ensuring secure and tamper-proof state tracking.

c) Bitcoin-NG: proposed by Eyal et al. in [72], Bitcoin-NG uses a special data structure inspired from bitcoin's chain of blocks. This structure is consisting of two kind of blocks key-blocks and Microblocks. The Keyblocks are produced using proof-of-work and share the same structure as Bitcoin's blocks. Their role is to determine the block miner. Between two key-blocks, the selected block miner creates and signs multiple Microblocks, each contains a collection of signed transactions. The miner responsible for creating Microblocks is chosen based on a predefined algorithm, such as a rotating leader schedule or Proof of Stake. As in Bitcoin, Bitcoin-NG blocks form a chain where Microblocks reference previous Microblocks and Keyblocks, as illustrated in Figure 8. Notably, Microblocks significantly reduce the overall size of data stored on the blockchain, improving storage efficiency and network bandwidth usage.

d) ByzCoin: Taking inspiration from Bitcoin-NG, ByzCoin [73] embraces the concept of decoupling in two blocks. However, rather than adhering to a single chain structure, ByzCoin introduces a novel approach by forming two distinct and parallel chains: Keyblocks and Microblocks. The Keyblocks serve as anchors or reference points, securing the overall structure, while the Microblocks contain a more granular level of transactional data.

1.2) Skipchain

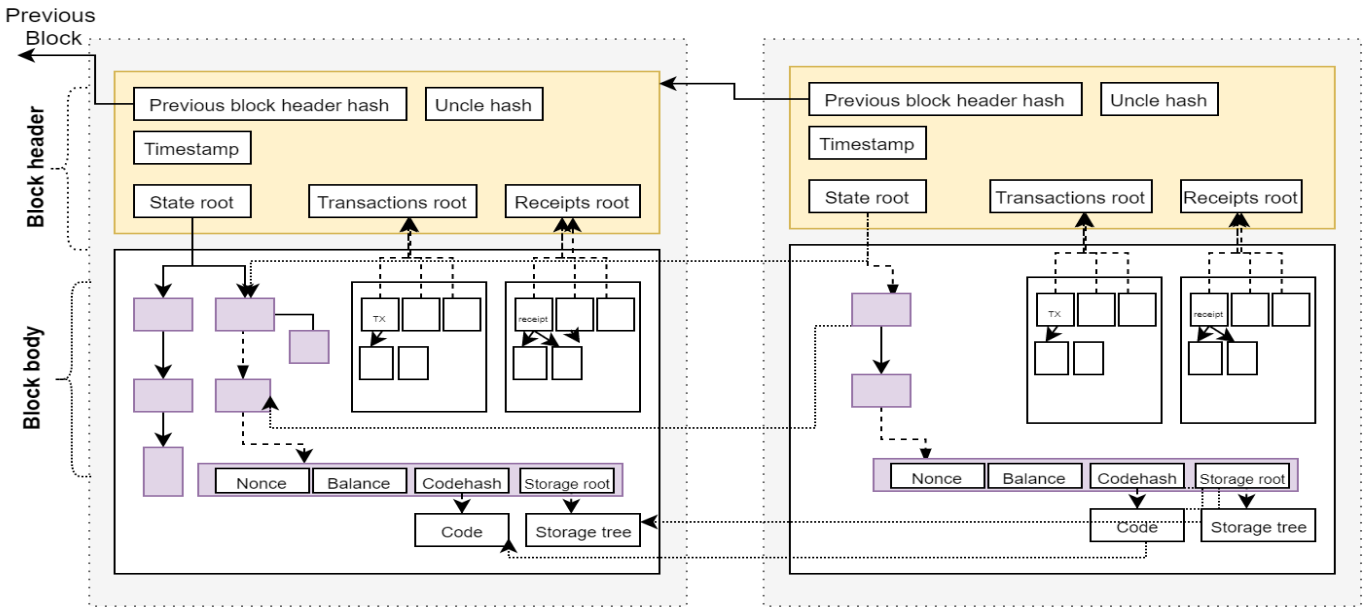

Fig. 7. Structure of Ethereum's chain of blocks

e) Chainiac: Introduced by Nikitin et al. [74], addresses challenges related to offline transaction verification. This involves enabling nodes to ascertain whether a transaction has been committed without requiring a complete copy of the ledger. The innovative solution proposed by Chainiac involves introducing traversability forward in time through the use of a Skipchain, as depicted in Fig. 9. Back-pointers are represented by cryptographic hashes, while forward-pointers are realized through collective signatures. By incorporating long-distance forward links and employing collective signatures, Chainiac facilitates efficient transaction verification for a client or node at any point in time.

1-3) Chainless DLT

f) DAG based chains: The concept of utilizing Directed Acyclic Graphs (DAGs) as the underlying data structure has garnered significant attention from DLT designers across various projects. Notable examples include Byteball [75], DagCoin [76], IOTA [4], Nano [75], Phantom [77], and Hedera [6]. Some studies have explored the integration of DAGs instead chain of blocks. For example, the GHOST protocol [78], A DAG-Based Modification for Bitcoin, aims to reduce confirmation time and to secure the network [79].

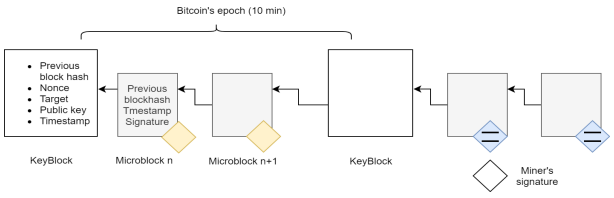

Fig. 8. Bitcoin-NG Blockchain Data Structure. Keyblocks and Microblocks contain the public key and the signature belonging to the miner.

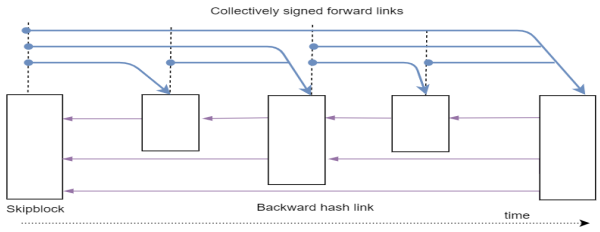

Fig. 9. Backward and forward links in skipchain

g) IOTA: IOTAcitePopov2017 is a DLT designed to provide a scalable and fee-less environment for machine-to-machine transactions. IOTA is among the early adopters of a DAG data structure known as Tangle. The tangle is a block-less DAG where transactions are directly interlinked without the need for traditional blocks. Unlike traditional blockchain systems, IOTA's Tangle eliminates the need for miners and introduces a decentralized validation model, where each participant contributes to the network's security by validating two previous transactions for every transaction they make. Transactions in the Tangle are stored as DAG edges, each referencing two immediate predecessors and forming a directed acyclic network with confirmed, unconfirmed, and potentially conflicting subsets (Fig. 10).

IOTA's graph begins with a root transaction known as the Genesis Transaction. This transaction serves as the starting point, initiating the creation of the Tangle. In the creation of the Tangle, by design, a single address holds all the created tokens (IOTA tokens); The Genesis Transaction then orchestrates the distribution of tokens to various accounts within the network. This distribution sets the stage for a decentralized ownership structure as tokens move from the centralized address to numerous accounts.

h) Hashgraph: [80] deviates from traditional blockchain technology by employing a unique, chainless architecture to achieve significantly higher throughput. Instead of relying on linear chains of blocks, each node within the Hashgraph network maintains its own DAG. The vertices of this DAG are referred to as ”events”, which are analogous to blocks in traditional blockchains. Similar to blocks, events contain a single or several transactions alongside other critical data. This data includes:

- Event parent hashes: These are the hashes of two previous events – one created by the gossip receiver and the other by the sender.
- A timestamp.
- A signature from the node that created the event.

Figure 11 depicts the structure of an event within the Hashgraph network.

i) Byteball: Described by Churyumov in [75], Byteball is a transaction-based Directed Acyclic Graph (DAG) system that differs from Bitcoin-NG. Instead of utilizing a hybrid structure of key-blocks and microblocks, Byteball relies solely on a DAG where each transaction directly references one or multiple previous transactions. Transactions establish a clear lineage through direct references to their parent transactions, ensuring data integrity. The DAG experiences dynamic growth as new transactions can attach to any existing transaction, enabling parallel processing and efficient network expansion. Initially employing proof-of-work for consensus, Byteball utilizes miners who compete to solve cryptographic puzzles, similar to Bitcoin. Additionally, Byteball employs a weighting system, where transactions are assigned weights based on age and lineage depth within the DAG, prioritizing older and more deeply connected transactions to promote network stability.

j) Nano: formerly known as RaiBlocks [81], employs a unique DAG data structure called Blocklattice. In the Nano network, each account possesses its dedicated chain of blocks, as illustrated in Figure 12. These individual chains are replicated across all network peers, enabling the simultaneous growth of multiple single chains. Significantly, only the wallet owner has the authority to make changes to their specific chain, allowing for asynchronous updates to each wallet. In contrast, Dexon [82] integrates a Blocklattice single chain into a globally-ordered chain without requiring additional communication. In the work by Sompolinsky et al. [83], [84], a DAG-based DLT is considered, where the nodes of the DAG represent blocks rather than individual transactions. Referred to as BlockDAG, this structure involves blocks referencing multiple other blocks, with newly added blocks pointing to recent blocks that are not yet referenced. This innovative paradigm serves as the foundation for emerging consensus protocols such as Inclusive [85], SPECTRE [83], and PHANTOM [84].

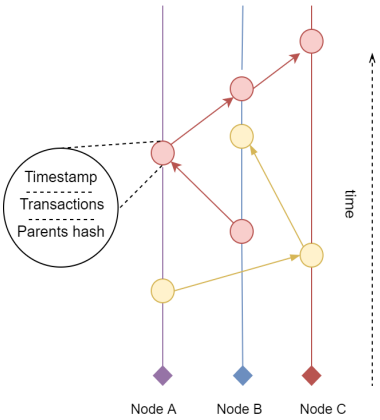

Fig. 11. Evolution of the Hashgraph

2) Decentralized Databases:

a) Corda R3 [63]: In Corda, each node has its own private, secure RDBMS (Relational Database Management System) called a ”vault”. The vault efficiently stores time-stamped records of transactions, contracts, and other relevant data for that node. Corda’s design differs from traditional blockchains where all nodes share a single, replicated ledger. The vault includes tables dedicated to various services, such as ledger management, transactions (NODE_TRANSACTIONS), states (NODE_STATES), participants (NODE_IDENTITY), and additional metadata.

While the metadata associated with a state is stored in the vault tables, the actual state data itself is stored in a separate binary format [86]. Attachments, which are files linked to

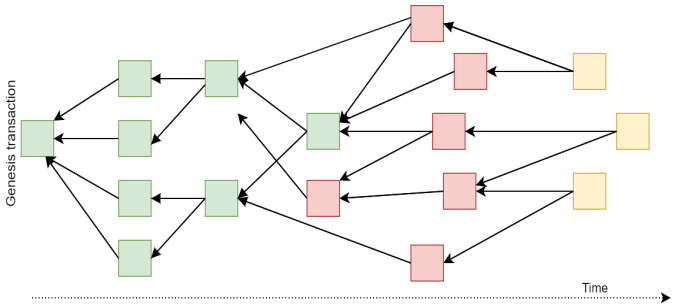

Fig. 10. IOTA Tangle’s DAG structure. Green blocks represent final validated transactions; Red blocks represent uncertain validated transactions; Yellow blocks represent transactions awaiting validation.

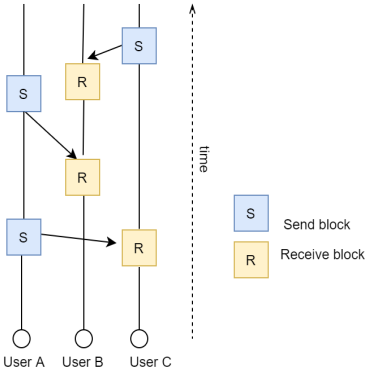

Fig. 12. Structure of DAG in Blocklattice

states, are stored in a distinct location on the node's file system. This separation enhances scalability and performance, particularly for managing large attachments.

In tandem with its storage approach, Corda adopts a UTXO model for handling state data. This model dictates that a transaction can consume existing states and may or may not produce new states, providing a clear representation of the state changes within the distributed ledger.

3) Hybrid DLTs:

a) Hathor: is a DLT that combines DAG and traditional blockchain architectures. Unlike traditional blockchains, Hathor utilizes a DAG structure for transaction processing, allowing for high scalability and throughput. The chain of blocks structure ensures security when the number of transactions per second is small, whereas the DAG prevails when the number increases significantly.

b) Hyperledger Fabric [87] : Hyperledger Fabric incorporates both a chain of blocks for storing validated transactions and a classical key-value database for managing the system's states, as depicted in Figure 13. While the ledger captures transaction metadata and state hashes, the actual state data—encompassing details like asset ownership and financial information—is deliberately stored off-chain. This strategic off-chain storage is designed to optimize efficiency and uphold privacy considerations. Hyperledger Fabric employs three distinct stores:

- Ledger: Stores transaction metadata and state hashes.
- Peer State Database: Manages state data associated with transactions validated by the peer.
- Private Data Storage: Houses private state data accessible only to authorized nodes.

Within the Fabric chain, the block structure resembles that of a traditional blockchain, yet it incorporates an additional segment: block metadata. This supplementary section includes a the certificate, public key, timestamp and signature of the block writer. In contrast to certain other blockchain structures, the Fabric block header is straightforward, and transactions within the block body are ordered without Merklization.

c) EOS [88]: EOSIO employs a multi-layered approach to data storage, seamlessly integrating the strengths of on-chain, off-chain, and RAM storage (Fig. 14). This layered architecture ensures efficient data management, catering to

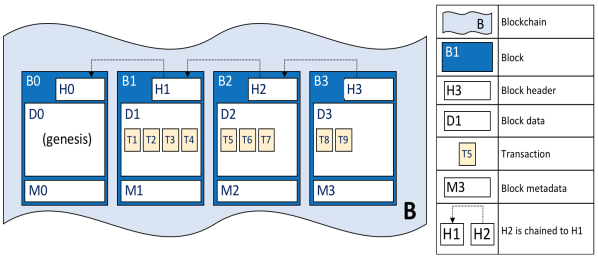

Fig. 13. Structure of Hyperledger fabric's chain of blocks (Source: Hyperledger official documentation)

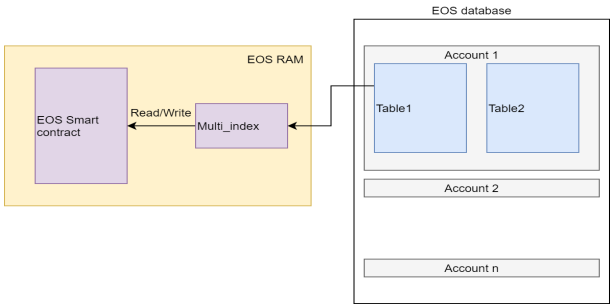

Fig. 14. EOS database structure and its interactions with smart contracts

diverse needs while maintaining security and performance. On-chain storage forms the bedrock of the system, acting as an immutable ledger for transactions and key state data. Replicated across the network, this data remains permanently accessible and tamper-proof. Within smart contracts, multi-index tables provide structured storage, allowing developers to organize and efficiently retrieve specific data. For data that requires scalability and privacy, off-chain storage comes into play. EOSIO offers various options to meet these needs, including the specialized Chainbase database for fast access and management of crucial data like user accounts, token balances, and smart contract states. Other options like MongoDB, SQLite, and RocksDB cater to specific data types and access requirements. To further optimize performance, RAM storage serves as a temporary caching layer. Frequently accessed data is stored in the memory of nodes, significantly reducing retrieval times and enhancing user experience. However, RAM resources are limited and require purchase or lease, encouraging efficient data management.

d) BigchainDB: BigchainDB's architectur [89] revolves around two key databases: the "Backlog," a staging area for pending transactions, and the "Chain of Blocks," a repository for validated transactions, ensuring efficient processing and secure record-keeping. It bridges the gap between NoSQL databases and blockchains, offering a hybrid approach that blends the flexibility of MongoDB with the immutability and security of blockchain technology. BigchainDB stores all its data in Json documents. These documents are then organized into three main categories:

- Transactions: Represent actions like creating assets, transferring ownership, or voting. Each transaction includes details like sender, receiver, asset information, and timestamp.
- Blocks: Combine multiple transactions and additional metadata to form a chain of data. This chain ensures data integrity and immutability.
- Votes: Nodes use votes to confirm the validity of transactions and blocks, maintaining consensus within the network.

e) Ripple: : [90] utilizes the XRP Ledger (XRPL), a hybrid of a database and a chain of blocks, offering unique advantages for its global payment network. The XRPL employs a multi-layered data storage architecture, combining centralized

and decentralized databases for optimal efficiency. Primary data, decentralized and managed by NuDB or Cassandra, provides redundancy. Secondary data, using PostgreSQL, can be centralized or distributed. Chainbase, for user accounts and smart contracts, operates centrally on dedicated servers. Off-chain databases like MongoDB, SQLite, and RocksDB are flexible, optimizing for specific data types. The XRPL features a Ledger Object Model (LOM) for structured data storage, efficient data retrieval with unique identifiers (LODIDs), state pruning for storage optimization, and a chain of blocks ensuring an immutable record of transactions. Distributed consensus via the Ripple Consensus Protocol (RLCP) and robust security features, including cryptographic hashing and digital signatures, enhance the authenticity and integrity of XRPL data.

4) Data shareability: Many DLTs designed for global cryptocurrency platforms inherently embrace a model of global shareability for transactions. For instance, Bitcoin and Ethereum operate in a relay mode, where nodes propagate transactions throughout the entire network without restrictions. Hashgraph, another DLT, employs a different approach where senders deliver transactions to a select set of nodes responsible for incorporating them into their DAG and sharing them through Gossiping. In addition to these, various other blockchain and decentralized ledger projects contribute to the landscape of data shareability. The Interledger Protocol (ILP)[91] facilitates interoperability, enabling different ledgers to share data and value seamlessly between network peers. Polkadot's heterogeneous multi-chain framework[92] allows connected blockchains to share data through its relay chain, promoting interoperability while controlling data access. Cosmos, aiming for an "Internet of Blockchains," enables different blockchains to selectively share data through the Cosmos Hub, emphasizing interoperability.

Furthermore, IOTA's Tangle structure fosters global shareability, with all network peers having access to efficiently interlinked transactions. Filecoin's decentralized storage network enables global data shareability, allowing peers to access shared data stored across the network. Nervos Network's two-layer architecture supports global shareability of state and assets, with all peers having access to the shared data on the Nervos network. Dfinity's decentralized cloud platform enables global data shareability by providing secure and scalable computing resources, accessible to network peers.

In contrast, DLTs tailored for business applications, such as Corda and Hyperledger Fabric, prioritize restricted shareability of transactions to address privacy concerns. Corda adopts a node-centric approach, where each node maintains a distinct database containing only relevant data, limiting the visibility of the ledger to individual peers. Similarly, Hyperledger Fabric introduces the concept of channels [93] to restrict data shareability. Each transaction occurs on a private subnetwork (channel), accessible only to authenticated and authorized parties, resulting in a distinct ledger for each channel. Other projects like Quorum, built on Ethereum, ensure privacy by facilitating private transactions among network participants through constellations[94] or Tessera [95]. Ripple, with its XRP Ledger (XRPL), employs a hybrid model that combines elements of both global and restricted data shareability. The XRP Ledger serves as a decentralized global payment network, where transactions are recorded in a chain of blocks. This chain ensures immutability and transparency of transactions, contributing to global shareability.

5) Data immutability: The pinnacle of strong immutability is exemplified in Bitcoin, where robust cryptographic hashing and chained blocks forge an unalterable record, rendering it ideal for applications requiring unwavering security and transparency. However, this rigidity may be less suitable for projects seeking adaptability and evolution. Other protocols, such as Ethereum or Avalanche, prioritize strong immutability, ensuring that once data is written to the ledger, it remains unalterable or erasable.

Business-oriented DLTs like Hyperledger Fabric illustrate the possibilities of weak immutability. By leveraging tree structures and state updates, they permit data modifications under specific conditions while preserving historical values, offering greater flexibility for projects to adapt to changing needs without compromising overall ledger integrity.

Projects like Multichain provide customizable data sharing with varying degrees of immutability, while Hyperledger Sawtooth enables the implementation of different consensus mechanisms and diverse levels of immutability.

Tezos introduces a unique feature known as the "rollback-enabled model," offering weak immutability. This model allows for the reversal of specific on-chain transactions in exceptional circumstances, such as critical bugs or vulnerabilities.

C. Data Layer: Discussion

The design choice of a data structure and its properties is complex, involving the assessment of numerous challenges and tradeoffs, as described below:

*1) **Tradeoff: Data Integrity (Block Size) versus Transaction Throughput (Performance)**:* The sequential nature of the chain of blocks ensures a high level of security and data integrity. However, it limits the number of transactions accepted by the network due to the block size (in megabytes or gas limit). This limitation penalizes system throughput. For instance, Bitcoin introduced a block size limit (1 MB) to prevent DOS attacks caused by huge blocks, but this decision hampers network performance (4 transactions per second). Increasing the block size cannot be an effective solution, as it risks breaking the network's decentralization, making full nodes more expensive to operate, increasing orphan block rates, and delaying propagation speed. To address this dilemma, Bitcoin adopted the Segregated Witness (SegWit) technique [96] [97], which separates signature data from Bitcoin transactions and allows its pruning from the ledger, safely increasing the block size limit to 2-4 MB. Additionally, to enhance throughput without increasing orphan blocks, the Bitcoin network reduced block propagation time by incorporating rapid relay networks [98]-[99]-[100]. In Ethereum, miners set the gas limit for each block. The gas limit is a parameter that defines the

maximum amount of gas units that can be consumed in a block. Gas is the computational unit used to measure the computational work performed by the Ethereum network, and it is separate from the cryptocurrency ether. That limit helps prevent malicious users from creating transactions that require excessive computational resources, potentially disrupting the network. In contrast to the chain of blocks, the parallel nature of DAGs enables the processing of transactions in parallel, contributing to higher throughput. However, a DAG structure comes with its own challenges. A reduction in the volume of transactions may expose it to attacks (e.g., parasite attack [101]). Therefore, DAG-based DLTs often resort to centralized mechanisms (e.g., coordinators in IOTA) to mitigate this risk.

*2) **The Challenge of Fast-Growing Ledger Size**:* Addressing the issue of fast-growing ledger sizes is a significant concern across all DLTs. Various solutions have been devised to handle this challenge at the data level. For example, projects like Bitcoin employ straightforward methods like ledger pruning. In this approach, nodes download blocks, validate them, and then discard irrelevant data to conserve disk space. On the other hand, more complex solutions, such as sharding techniques, are utilized by projects like Zilliqa [102] and Ethereum 2.0. Sharding involves running multiple parallel subsets of nodes, known as shards, where each shard maintains its sub-ledger. This enables parallel transaction processing and distributes data storage among multiple nodes.

Additionally, there's an active development of off-chain processing techniques to alleviate the network's load, whether in terms of storage or computation. Examples include Bitcoin's Lightning Network [103] and Raiden [104]. In the realm of business-oriented implementations, cloud providers offer Blockchain as a Service (BAAS) solutions, such as IBM BAAS [105] and Azure BAAS [106]. These solutions involve deploying a ready-to-use DLT solution on the cloud, providing the necessary storage space and network bandwidth. For more in-depth information on current BAAS solutions, readers are encouraged to refer to [107].

*3) **The transparent aspect of the blockchain versus privacy of shared data**:* The transparency of the DLTs often raises concerns regarding data privacy, particularly in DLTs with global shareability. While most public blockchains employ pseudo-anonymization to introduce an initial layer of privacy, the persistent risk of deanonymization, as demonstrated by [108] [109], remains. This risk is heightened when users neglect to employ distinct single-use addresses for each operation, a practice supported by hierarchical deterministic wallets [110]. To enhance anonymity and facilitate confidential transactions in the blockchain sphere, solutions such as Monero [111] and Zcash [112] have been proposed. Zcash introduced zkSNARKs [113], a zero-knowledge proof system, while Monero utilizes ring signatures [111] to conceal transaction senders and recipients. An additional strategy to bolster the anonymity of existing public blockchains (e.g., Bitcoin or Ethereum) involves mixing schemes like CoinJoin [114], Coinshuffle [115], MimbleWimble [116], and Grin [117]. These solutions employ transaction shuffling mechanisms to safeguard anonymity. Additionally, initiatives like Zether [118], WaterCarver [119], and Aztec [120] aim to leverage zkSNARKs, introducing an enclave of privacy to public Ethereum for enabling private transactions. It's worth mentionning that entities such as StarkNet, which currently lack native support for the Ethereum Virtual Machine (EVM), are actively investigating transpilers to overcome the programming language gap. The future is in favour of zkEVMs, as they compete for supremacy not only within the realm of zk-rollups but also against their optimistic counterparts.

V. Consensus Layer

DLTs have sparked a renewed interest in the development of innovative distributed consensus protocols. In fact, the literature has witnessed the proposal of numerous consensus algorithms tailored for DLTs, each presenting distinct properties and functionalities. This section introduces the properties and features incorporated into the DCEA framework, essential for the study and differentiation of these protocols. Additionally, the second subsection provides an overview of the current state of the art in this domain. Section VIII delves into the presentation and discussion of the results derived from a comparative analysis of the examined protocols.

A. Components and Properties

1) Basic Properties: The concepts of safety and liveness properties are fundamental to understanding and evaluating consensus algorithms in distributed systems. These properties were originally introduced by Leslie Lamport in 1977 in his seminal work "Proving the Correctness of Multiprocess Programs" [121]. Since then, they have become essential tools for analyzing and designing consensus protocols in various distributed computing scenarios.

a) Safety: In the context of DLT networks, safety denotes the assurance that correct nodes will not simultaneously validate conflicting outputs or make conflicting decisions, such as chain forks. The safety property plays a crucial role in ensuring availability, ensuring that transactions submitted by honest users are efficiently integrated into the ledger [122]. Additionally, safety encompasses preventing undesirable phenomena like reorganizations (reorgs), where previously confirmed transaction history undergoes revisions, often due to factors like network forks or changes in consensus rules.

b) Liveness: a consensus protocol ensures liveness in a DLT if it guarantees eventual progress, agreement even in the presence of faults or delays.

c) Finality: we define the finality as the affirmation and guarantee that once a transaction is recorded on the ledger, it cannot be altered or reversed, guaranteeing data integrity and providing a strong foundation for trust in the system. Finality can be divided into two categories:

- Probabilistic finality, where the certainty of a transaction being irreversible increases with time after its inclusion in the ledger. This means that while the transaction may theoretically be reversed, the probability of such an event

occurring diminishes significantly as more blocks are added to the chain
- Absolute finality guarantees that a transaction is irreversible once it has been validated by a majority of honest nodes in the network.

2) Network Models: In the literature of distributed systems and DLT consensus protocols, we adhere to the message passing model where nodes communicate by exchanging messages over the network, operating with different assumptions of network synchrony. In this survey, we adopt the taxonomy defined by [123], which categorizes network synchrony assumptions.

- **Synchronous:** Assumes a known upper bound on message delay, ensuring messages are consistently delivered within a specified time after being sent.
- **Partially-synchronous:** This assumption is based on a Global Stabilization Time (GST), indicating that messages sent will be received by their recipients within a predetermined time frame. Before reaching the GST, messages may encounter unpredictable delays.
- **Asynchronous:** Messages sent by parties are eventually delivered, with arbitrary delays and no assumed bound on the delivery time.

3) Failure Models: Different failure models have been considered in the literature; we list here two major types.

- **Fail-stop failure (Also known as benign or crash faults):** Nodes go offline because of a hardware or software crash. Fail-stop failures can be detected relatively easily, as the node simply becomes unreachable.
- **Byzantine faults:** This category of faults was introduced and characterized by Leslie Lamport in the Byzantine Generals Problem [124] to represent nodes behaving arbitrarily due to software bugs or a malicious compromise. In a Byzantine fault, a node behaves arbitrarily, potentially due to software bugs, security breaches, or malicious intent, allowing the node to send conflicting messages to different nodes, forge false information, and collude with other Byzantine nodes to mislead the system.

Therefore, we consider a protocol as fault-tolerant if it can gracefully continue operating without interruption in the presence of failing nodes.

4) Adversary Models: Distributed systems, including blockchain networks, operate in environments where malicious actors may attempt to disrupt their operation or exploit vulnerabilities for their own benefit. To design secure and resilient systems, it is crucial to understand different adversary models and how they can impact the network.

- Threshold Adversary Model (Hirt and Maurer) [125]: This model, commonly used in traditional distributed computing literature, assumes that the Byzantine adversary can corrupt up to any f nodes among a fixed set of n nodes. This model typically assumes a closed membership where permission is required to join the network. The consensus protocol should ensure consensus in the presence of Byzantine nodes as long as their numbers is below a specific threshold.
- Computational Threshold Adversary: A model introduced by Bitcoin (and applicable to other decentralized systems), where the adversary's control over the network is bounded by computational power not by the number of nodes they can control. This model typically assumes, the membership is open to multiple parties, and the bounding computation involves a brute force calculation.
- Stake Threshold Adversary [126]: In this model, the adversary's control is bound by the control over a specific resource: the system's native token or currency. The Stake Threshold Adversary is the one who hoards enough of these shares (tokens) to gain a significant say in the system's operation.

5) Adversary Modes: Consensus protocols consider various types of adversaries based on their capabilities and the time required to compromise a node.

- **Static adversary:** Represents a Byzantine entity with the ability to corrupt a predetermined number of nodes in advance, exerting complete control over them. However, the static adversary exhibits limited adaptability, as it cannot modify the selection of nodes it has corrupted after the initial attack. Furthermore, the adversary achieves instantaneous control over the compromised nodes, without any delay or gradual process.
- **Adaptive adversary:** Characterizes a Byzantine entity that possesses the ability to dynamically control nodes within the network, making adjustments based on changing circumstances. This adversary can flexibly change the nodes under its control, enhancing its overall influence over time.
- **Mildly adaptive adversary:** Describes a Byzantine entity capable of an corrupting nodes based on past messages.It can observe and learn from past messages exchanged within the system. This allows it to tailor its attacks based on the acquired information. Importantly, this adversary lacks the ability to alter messages that have already been sent. While the adversary can corrupt entire groups of nodes, this action comes with a time penalty. Corrupting a group takes longer than the group's normal operating phase.
- **Strongly adaptive adversary:** Represents a Byzantine entity that possesses the capability to gain knowledge of all messages sent by honest parties. This adversary leverages this information to make decisions, determining whether to corrupt a party by altering its message or delaying message delivery. With a high level of adaptability, the strongly adaptive adversary strategically utilizes its comprehensive knowledge to compromise the system.

6) Identity Model: Distributed ledger consensus protocols adopt different approaches to managing nodes' membership, generally falling into two contrasting categories:

- **Permissionless:** This model embraces an open membership system, enabling any node to freely join the network and participate in the validation of new entries.
- **Permissioned:** In contrast, the permissioned model re-

stricts membership, allowing only a predefined set of approved members to validate new entries.

In the context of Distributed Ledger Technology (DLT), the identity model is intricately tied to the network's openness, which may manifest as private, public, or a consortium. A comprehensive exploration of these identity types can be found in [127] and [128].

7) Governance Model: The governance model pertains to the decision-making framework embraced by a DLT network for determining protocol rules and their updates and upgrades. Given that system governance is inherently a social concept, various governance models can be identified:

- **Anarchic:** Protocol upgrade proposals undergo approval by every participant in the network. Each participant has the autonomy to accept or reject a given proposal, potentially leading to network splits.
- **Democratic:** Participants engage in voting on new rules and protocol upgrade proposals. Ultimately, all participants are obligated to follow the decision of the majority, even those who opposed it.
- **Oligarchic:** New rules and protocol upgrades are suggested and endorsed by a select group of participants.

Considering the shift of most DLTs in handling governance and related matters "on-chain" or "off-chain", we also recognize the distinction between:

- Built-in (On-chain governance): The decision-making process is defined as part of the underlying consensus protocol within the network.
- External governance (Off-chain governance): The decision-making process relies on procedures independently performed outside the DLT network.

8) Transactions Ordering: Maintaining the chronological order of transactions is crucial for preventing fraud and inconsistencies in both linear and non-linear DLTs, including DAGs. To address this requirement, diverse approaches have been developed within consensus protocols to ensure correct transaction ordering. While transaction ordering is typically integrated into the consensus mechanism for efficiency, specific scenarios may necessitate decoupling it from transaction execution and validation. This allows for greater flexibility and optimization based on specific application needs. Two key approaches for decoupling ordering include:

- Optimistic ordering: Transactions are speculatively executed and finalized only after their order is confirmed, allowing for faster processing but requiring additional validation steps.
- Off-ledger ordering: Transactions are ordered outside the DLT before being submitted for validation and execution, providing increased scalability but adding complexity to the system.

9) Conflict Resolution Model: In certain DLT networks, the coexistence of temporary conflicting ledger versions, known as forks, may arise due to factors such as network latency or parallel block validation. To converge towards a single source of truth, networks and consensus mechanisms implement various rules, with the "longest chain rule" from the Bitcoin protocol being one of the most prominent. In case of conflicting orders, the PoW-based networks converge to a single order following the longest chain — the chain with the largest accumulated Proof of Work discarding the others. This rule is adopted by various protocols, each potentially utilizing a different cumulative parameter (witness votes, endorsements, etc.). Table XI illustrates different rules employed by various DLTs to resolve conflicting orders and mitigate discrepancies. In Proof of Stake (PoS) blockchains, the authoritative chain may be the one with the highest support from validators, chosen based on their stake in the network. The chain with the highest level of validator support is considered the valid chain.

B. Consensus Layer: State of the Art

In this subsection, we introduce various consensus mechanisms and their properties. While it is beyond the scope of this paper to provide an exhaustive taxonomy of existing protocols (see fig. 15), we categorize the reviewed protocols into six groups. This classification forms the basis for categorizing DLTs.

1) BFT Consensus Family (PBFT-like): This family encompasses classical consensus mechanisms derived from traditional distributed computing literature and their recent variations. The BFT family is distinguished by its characteristic of conducting all-to-all voting rounds [129]. Nodes within the network possess known identities, and the participant count is limited. Given the multitude of protocols within this family, our paper's review primarily concentrates on the most widely employed algorithms in the context of DLT, namely PBFT, Raft, IBFT, DBFT, POA (AURA, Clique), HoneyBadgerBFT, and HotStuff.

a) Practical Byzantine Fault Tolerance (PBFT): The PBFT algorithm, introduced by Castro and Liskov [130], stands out as a well-established consensus mechanism often synonymous with BFT consensus. Designed for practical, asynchronous environments with a Byzantine minority of f out of a total of $3f+1$ nodes, PBFT relies on a leader-based approach. In each view (leader election term), a leader is elected to append log entries (blocks in the case of a blockchain).

PBFT ensures asynchronous safety by employing a three-phase protocol: pre-prepare, prepare, and commit. The first two phases (pre-prepare and prepare) facilitate ordering requests

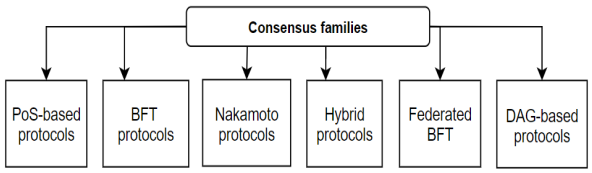

Fig. 15. Our taxonomy for consensus protocols

sent within views, even when the leader is faulty, while the last phase ensures that committed requests are totally ordered across views. However, PBFT exhibits a weak level of liveness, aiming to prevent faulty nodes from forcing frequent view changes and consistently promoting a faulty node to the primary role.

While PBFT enables high throughput and low latency, it does have certain drawbacks, including the costliness of recoveries in case of faulty leaders and performance degradation with an increasing number of nodes. As a result, PBFT is considered suitable primarily for a small-to-medium group of known participating nodes.

b) RAFT [131]: is a log-replication consensus algorithm that uses a leader-based approach. A single leader is elected, chosen through randomized timer timeouts, to append new entries to the log (append-only) and replicate it across the other nodes in the network. When the leader becomes faulty or slow, at most one of the nodes with the most up-to-date log is chosen as the new leader. Similar to PBFT, all nodes in RAFT must be known and interconnected to exchange messages in epochs.

RAFT guarantees network safety by ensuring that every log is correctly replicated and commands are executed in the same order across all nodes. Additionally, only one leader is ever elected at a time, preventing conflicting updates. Notably, attempts have been made to extend RAFT's tolerance to Byzantine faults, as seen in works by [132]and [133].

c) Istanbul BFT (IBFT) [134]: is a Byzantine Fault Tolerant (BFT) state machine replication-based consensus algorithm. IBFT inherits from the original PBFT by using a three-phase consensus protocol, where before each round, nodes will elect a leader (called Proposer) who will be responsible for proposing new blocks in the network along with the PRE-PREPARE message. In each stage, validators broadcast the state message and wait for 2f+1 state messages (PREPARE, COMMIT) to commit the current state (insert the block to the chain). IBFT can tolerate at most f faulty nodes in a network with 3f+1 nodes.

Three-Phase Consensus in IBFT:

- Pre-prepare: The Proposer broadcasts a PRE-PREPARE message containing the proposed block to other nodes.
- Prepare: Upon receiving the PRE-PREPARE message, validators verify the block and then broadcast a PRE-PARE message if they agree.
- Commit: Once a validator receives 2f+1 PREPARE messages for the same block, it broadcasts a COMMIT message and adds the block to its chain.

While IBFT demonstrates scalability to a large number of peers, it comes with a trade-off as it demands higher communication overhead compared to other consensus algorithms.

Proof of Authority (PoA)

d) Proof of authority PoA [135]: is a Byzantine Fault Tolerance (BFT) leader-based consensus mechanism designed for permissioned blockchain deployments with at least *n/2+1* honest nodes, where *n* is the total number of nodes.

PoA relies on a set of trusted and identifiable nodes called authorities, responsible for creating new blocks and securing the blockchain. PoA protocols operate through a series of steps, each with an authority elected as a validation leader using a rotating process. PoA is a reputation-based consensus protocol where the authority's reputation is at stake instead of financial or computational power.

PoA has two major implementations: Aura [136] and Clique [137]. These implementations differ in their process:

- Aura : In the Block Proposal phase, the current leader takes the initiative to propose a new block. Following this, in the Acceptance phase, a subsequent round unfolds where other authorities within the network participate in voting to determine the acceptance of the proposed block. This two-phase approach ensures a systematic and distributed mechanism for reaching consensus on the inclusion of new blocks in the blockchain.
- Clique: In Clique, the current leader proposes a new block, and block acceptance is immediate as other authorities directly append the proposed block to their chains. This streamlined approach eliminates a separate acceptance phase, enhancing efficiency in the consensus process.

Unlike PBFT, PoA requires fewer message exchanges, resulting in better performance [21]: Aura offers lower transaction acceptance latency and more predictable block issuance with steady time intervals. In contrast, Clique achieves faster block creation by eliminating a separate block acceptance round from its consensus process. However, Deangelis et al. [135] argue that PoA algorithms might not be the most suitable choice for permissioned blockchains deployed over the internet. They advocate for PBFT as a potentially better alternative for permissioned settings.

e) Democratic BFT (DBFT): [138] is a deterministic and partially synchronous consensus algorithm that can tolerate up to $f < n/3$ Byzantine nodes, where n is the total number of nodes. Unlike practical Byzantine Fault Tolerance (PBFT) protocols that rely on a single, correct leader to finalize the consensus process, DBFT is a leaderless protocol.

DBFT offers several key features:

- Multiple Proposers: Instead of relying on a single leader, DBFT allows multiple nodes to propose sets of transactions for inclusion in a block, enriching the pool of potential options and potentially accelerating block creation.
- Disjoint Sets: Proposers in DBFT can propose disjoint sets of transactions, meaning they don't necessarily need to agree on the exact contents of a block. This allows for flexibility and can potentially lead to higher throughput.
- Asynchronous Rounds: Nodes in DBFT can complete asynchronous rounds, meaning they don't need to wait for all other nodes before proceeding. This can further improve performance, especially in large networks.
- Democratic Decision-Making: Each node plays a similar role in the consensus process, contributing to a more

"democratic" decision-making mechanism compared to leader-based protocols.

This combination of features allows DBFT to achieve high throughput and scalability, making it suitable for large-scale blockchain deployments. However, DBFT is comparatively complex among BFT protocols, potentially posing implementation challenges. While it exhibits promising performance, ongoing research focuses on formally verifying its security properties.

f) HoneyBadgerBFT or HBBFT [139]: is a Byzantine fault-tolerant consensus algorithm designed for fully asynchronous networks with an honest majority ($n > 3f$). Unlike traditional BFT, it eliminates the need for a special leader node to propose transactions; instead, every node assumes the role of a proposer. In each epoch, participating nodes exchange a batch of encrypted transactions and collectively agree on them using a randomized agreement protocol. HBBFT relies on threshold encryption, where transactions are encrypted with a shared public key and can only be decrypted when the elected consensus committee collaborates. This ensures that adversaries cannot determine which transactions are proposed by specific nodes until an agreement is reached. Additionally, HBBFT incorporates optimizations to enhance system performance, including communication-optimal reliable broadcast [140], binary consensus as proposed by [141], and batching [140]. It is particularly suitable for networks with a small number of known (permissioned) validator nodes. However, it's essential to note that HBBFT, like any consensus algorithm, has its drawbacks. One potential drawback is the computational overhead associated with the use of threshold encryption, which may impact the algorithm's overall performance, especially in resource-constrained environments. Additionally, the complexity of the algorithm may pose challenges in terms of implementation and maintenance. Despite these considerations, HBBFT remains a viable choice, particularly for networks with a small number of known (permissioned) validator nodes.

g) HotStuff [142]: is a leader-based Byzantine fault-tolerant replication protocol designed for partially synchronous networks, offering both linearity (linear change view) and responsiveness. It re-examines the original BFT designs and aims to significantly reduce the authenticator complexity of PBFT. In cases of a correct leader or view-change, the complexity is brought down from O(n^2) and O(n^3) to O(n), where *n* is the number of nodes. Unlike PBFT, HotStuff adopts a three-round process, rotating the leader every three rounds after a single attempt to commit a command. The leader replacement (view-change) operation is integrated with the normal process, eliminating a separate view change process and reducing overall complexity. HotStuff streamlines PBFT authentication complexity by introducing a new proposer in the first phase carrying only a single commit-certificate, and in the second phase, replicas can reject a proposition conflicting with their highest-level certificate without requiring a leader proof. Additionally, it shifts PBFT's communication model from a mesh to a star topology, relying on the leader for communication between nodes, and utilizes threshold digital signature schemes to further reduce authenticator complexity. Notably, HotStuff ensures optimistic responsiveness, with a leader requiring only *n-f* (where *f* is the number of faulty nodes) votes from other nodes to guarantee progress. It maintains simplicity and modularity by decoupling safety (voting and commit rules) from liveness, guaranteed by the pacemaker mechanism after the global stabilization time (GST). Furthermore, HotStuff enhances BFT scalability by linearizing the algorithm's authenticator complexity, making it suitable for wide networks. However, the introduction of three phases in HotStuff, compared to the two phases in PBFT, induces additional latency, limiting the throughput to a single commit per three phases. For a comprehensive exploration of other Byzantine consensus algorithms with extensive analysis, refer to [143].

2) Nakamoto consensus family: The Nakamoto consensus family comprises protocols that utilize a chain of block data structures and employ the longest chain fork choice rule (or variants like GHOST [79]) to ensure network safety, all while incorporating economic incentives as a motivating factor. Originally designed to facilitate secure global currency transfers, these protocols offer a simpler alternative to PBFT while tolerating significant corruption (up to *n/2* nodes). Notably, they operate on a permissionless basis, allowing nodes to freely join or leave the network without requiring prior authentication. Here, we explore some of the most prominent protocols within this category: PoW, memory-bound PoW, and BitcoinNG.

a) proof-of-work (or PoW): PoW is a consensus mechanism where a leader, commonly known as a miner, is chosen probabilistically based on the computational power contributed. Upon solving a cryptographic puzzle, the miner constructs and submits a block, accompanied by a valid proof-of-work nonce, to the network. Verification by other peers involves computing the hash of the block header and ensuring it meets the condition of being smaller than the current target value. Valid blocks are added to the chain with the largest cumulative difficulty. In PoW, each peer effectively casts a vote on transaction validity using its hashing power. The protocol relies on partial hash collisions to thwart Sybil attacks[144], preventing the system from being tainted by multiple misbehaving nodes.

A noteworthy aspect of PoW is its resilience to potential forks or reorganizations in the blockchain. In the event of competing blocks being added simultaneously by different miners, the network may experience a temporary fork. However, PoW relies on the principle that the longest chain is considered the valid one. Nodes in the network continuously work on extending the chain, and the longest chain, representing the majority of computational power, ultimately becomes the accepted version. This mechanism ensures a coherent and agreed-upon transaction history. Additionally, PoW introduces economic measures and incentives, such as transaction fees and mining rewards, to actively discourage denial-of-service attacks and protect the network against spam, contributing to

the overall robustness of the protocol.

b) Memory bound PoW: Originally conceived as an egalitarian process accessible to all, mining in traditional PoW faced concerns over centralization due to the widespread adoption of application-specific integrated circuits (ASICs). These specialized devices provided a significant advantage over conventional hardware, leading to the concentration of mining power among a few entities and the dominance of ASIC manufacturers.

To address this issue, various ASIC-resistant solutions were introduced. Dwork et al. [145] [146] [147] proposed a memory-bound proof-of-work that relies on random access to slow memory rather than computational hashing power. This design makes ASIC mining less efficient, as ASICs cannot accommodate the substantial memory requirements. Several PoW implementations, such as Scrypt [148], Primecoin [149], Equihash [150], and CryptoNight [151], have adopted memory-bound mining processes.

For example, Ethereum 1.0 employed a PoW protocol called Ethash [152], intentionally designed to be ASIC-resistant through memory-hardness. Miners on the Ethereum network are required to compute a sizable in-memory Directed Acyclic Graph (DAG), which stood at 4 GB as of December 23, 2020 [153], to mine new blocks. Ethereum is further planning to enhance its Ethash algorithm by transitioning to ProgPoW (programmatic proof-of-work), aiming to improve resilience against ASIC mining before transitioning to the proof-of-stake [154] consensus protocol. This strategic approach seeks to maintain a fair and decentralized mining ecosystem within the Ethereum network.

c) Bitcoin-NG [72]: Bitcoin-NG was proposed to scale PoW-based Nakamoto protocols by decoupling transaction verification from the leader (miner) election process. Similar to Bitcoin, Bitcoin-NG operates in epochs, where each epoch features a single leader chosen via proof-of-work. Once a leader is identified, they are responsible for unilaterally serializing transactions via Microblocks in the succeeding epoch, respecting predefined limits on rate and block size, until a new leader is chosen. The decoupling in Bitcoin-NG reduces delays caused by periodic leader elections and scales Bitcoin in terms of transaction throughput and propagation latency. However, Bitcoin-NG does not ensure strong consistency, as short forks may occur during the leader-switching process.

3) Proof of stake and its variants: Proof-of-Stake (PoS) emerged as an alternative to the resource-intensive Proof of Work (PoW) and was initially proposed in PPCoin [155]. Instead of engaging in a competitive hash-calculation race, participants aspiring to become validators and forge new blocks in a PoS system must lock a specific amount of coins into the network as a financial stake. The likelihood of a node being selected as the next validator is then determined by the size of their stake. Various implementations of PoS have been introduced, each bringing unique features to address specific challenges. Notable representatives include Ethereum PoS, which combines PoW and PoS, Delegated Proof of Stake (DPoS) as seen in EOS, Liquid Proof of Stake (LPoS) allowing users to "lease" their coins to validators, and Ouroboros and its variants. Snow White is another example within the PoS landscape. The evolution of PoS and its diverse variants reflects ongoing efforts to refine and optimize blockchain consensus mechanisms.

a) Ouroboros: was introduced by Aggelos Kiayias et al [156] in 2017 as a provably secure proof of stake protocol. The protocol operates in epochs, with each epoch divided into slots during which blocks are produced by a randomly chosen slot leader. Within each epoch, a committee of stakeholders employs a secure multiparty implementation of a coin-flipping protocol to generate the required randomness for electing a random list or committee of block producers for the slots in the current epoch. From this list, slot leaders are then elected using a random lottery algorithm called Follow the Satoshi (FTS). Given that Ouroboros is a PoS system, the probability of selecting a block producer is proportional to their stake. While Ouroboros was presented as the first provably secure proof-of-stake in the synchronous setting, it is susceptible to desynchronization attacks. This vulnerability arises as slot leaders in Ouroboros require precise synchrony to utilize their allocated slots accurately, potentially leading to network stalling or hindering liveness.

b) Ouroboros Praos [157]: extends the Ouroboros protocol to accommodate semi-synchronous networks, addressing desynchronization attacks and providing security against fully-adaptive corruption. In contrast to Ouroboros, Praos incorporates a special verifiable random function (VRF) that enables the randomly selected slot leader to anticipate the slots they will lead in advance. Unlike Ouroboros, where stakeholders learn about the slot leader once they publish a block, Praos stakeholders privately verify the corresponding VRF to determine the slot leader. The protocol is underpinned by formal analysis and delivers a robust Proof-of-Stake (PoS) implementation with a security level equivalent to Proof of Work (PoW).

c) Ouroboros Genesis [158]: represents a variant of the Ouroboros Praos protocol designed to address the synchronization challenge for new nodes joining Proof-of-Stake (PoS) systems. In contrast to Praos, which maintains moving checkpoints, Genesis introduces a distinct chain selection rule, employing the so-called $maxvalid$ procedure [158]. This innovation enables new nodes to safely participate in the protocol without requiring external information beyond the genesis block. As a result, new validators can verify the true longest chain with only knowledge of the genesis block. The security of Ouroboros Genesis has been formally proven in [158] against a fully adaptive adversary controlling less than half of the total stake in a partially synchronous network.

d) Ouroboros Chronos [159]: extends the Ouroboros Genesis protocol, introducing a significant innovation by eliminating the necessity for a global clock to maintain network synchronization. This design removes the dependency on an external service providing timestamps, such as NTP [160]. Ouroboros Chronos achieves this by introducing a novel synchronization mechanism that allows nodes to synchronize

their local clocks based solely on knowledge of the genesis block and the assumption that their local clocks, initially desynchronized, advance at approximately the same speed.

e) Snow White: was introduced by Daian et al. [161] as a PoS protocol providing end-to-end and formal proofs of security in asynchronous and permission-less settings. This blockchain-style protocol emphasizes the ability of users to freely enter and exit the network, incorporating a modified version of the sleepy consensus protocol introduced by [162]. Similar to Ouroboros, Snow White operates in epochs. In each epoch, a leader is randomly and publicly elected from a committee of stakeholders to append a block to the blockchain. To address security concerns related to committee reconfiguration and random block-proposer selection, particularly adaptive chosen key and randomness-biasing, Snow White introduces a novel "two-lookback" mechanism. In each epoch, a new consensus committee is determined in advance (multiple blocks before) than the randomness seed, making seed prediction challenging for malicious nodes. The previously generated randomness seeds of the previous blocks serve as a source of randomness entropy to seed the new random oracle for electing a slot's leader. Additionally, akin to PoW, Snow White's nodes always choose the longest chain in the presence of multiple chains.

f) Delegated Proof of Stake (DPoS) [88]: was originally conceptualized by Daniel Larimer and implemented in the Bitshares blockchain [163]. Serving as a variant of the PoS consensus, DPoS relies on a group of delegates, commonly referred to as witnesses, who are elected by token holders (stakeholders) to generate and validate blocks on behalf of other network participants. DPoS networks typically feature an odd number of elected witnesses, ranging from 21 to 101, who take turns forming and signing new blocks for approval through a voting system. However, the specific range of witnesses can vary depending on the chosen DPoS implementation and network configuration. Some networks might employ a lower number of witnesses, such as 11 or 17, for improved performance or efficiency. Conversely, others might utilize a higher number, exceeding 101, to accommodate larger network sizes or enhance decentralization.

The specifics of the voting system in DPoS can vary between implementations, but generally, each witness presents a single proposal when soliciting votes from other witnesses. Similar to PoS, a voter's weight is determined by their stake in the network. Moreover, witnesses in DPoS cannot sign arbitrary blocks or produce a block outside their scheduled time slot. Refusal by witnesses to produce blocks results in swift expulsion, with replacement by other elected witnesses.

While DPoS offers enhanced scalability compared to PoS or PoW, it faces criticism for the significant risk of centralization arising from the concentration of power among a limited number of actors (delegates).

4) Hybrid protocols : Hybrid consensus protocols, drawing strength from established mechanisms like PoW and PoS, combine elements of different approaches to overcome individual limitations and achieve better performance. This hybrid approach offers the potential for enhanced scalability, improved security, and flexibility to adapt to changing network conditions. However, increased complexity, potential conflicts, and finding the right balance between mechanisms remain challenges to be addressed.

a) Byzcoin: was introduced by Kokoris-Kogias et al. [164] as a solution to enhance Bitcoin's consistency and performance by leveraging the advantages of Practical Byzantine Fault Tolerance (PBFT) and employing collective signing. In contrast to Bitcoin, Byzcoin forms a BFT consensus group through the collaboration of Proof of Work (PoW) miners. The protocol incorporates a proof-of-membership mechanism based on PoW to establish the consensus group using a fixed-size sliding share window. Membership shares are distributed to successful miners for mining a valid block, and the cumulative shares represent their voting power in the consensus process. As miners mine new blocks, the share window advances, and miners without valid shares exit the consensus group.

Byzcoin employs a collective signing protocol named scalable collective signing (CoSi) [165] to aggregate thousands of signatures, reducing the PBFT communication complexity from O(n^2) to O(n). This scalability enables BFT protocols to accommodate large consensus groups. Moreover, ByzCoin ensures safety and liveness under Byzantine faults, with near-optimal tolerance allowing for up to f faulty group members among *3f + 1* participants. To minimize transaction processing latency, ByzCoin adopts the decoupling of transaction verification from block mining, a concept introduced by Bitcoin-NG.

b) Solana [166] [167]: Solana actually utilizes a hybrid consensus mechanism that combines both Proof of History (PoH) and Proof of Stake (PoS). This is one of the unique aspects of Solana's architecture, allowing it to achieve high scalability and transaction throughput. Proof of History (PoH) operates on the principles of Verifiable Delay Function (VDF) and Cryptographic Hash Functions to establish a reliable and verifiable timeline of events on the Solana blockchain. The VDF serves as a time-sequencing mechanism, designed to be computationally expensive to solve but easy to verify. Validators independently execute VDFs, generating unique outputs that act as proofs of the time spent solving the function. These outputs are then hashed and linked together to form a continuous chain of timestamps.

The PoH process involves the execution of VDFs by validators, generating unique outputs that serve as proofs of time. These outputs are hashed and linked in a chain, creating an immutable and tamper-resistant history of timestamps. Validators share their hash chains with the network, and other validators can easily verify the chain by checking the validity of each VDF output and ensuring the cryptographic integrity of the chain.

Consensus is built based on these verified timestamps, allowing validators to agree on the order of transactions and establish the current state of the blockchain. PoH introduces additional features to enhance its functionality, such as Sealevel, a parallel execution environment that enables

multiple validators to work on different parts of the VDF chain simultaneously, thereby improving processing speed and scalability. Tick Verification is another feature that ensures the accuracy of timestamps by allowing validators to periodically compare their internal clocks and make adjustments if necessary. Solana's Proof of Stake (PoS) and Proof of History (PoH) work in tandem, each playing a distinct role in the blockchain's consensus mechanism. Their contributions can be outlined as follows:

- PoH: Primarily responsible for ordering transactions and creating a verifiable timeline of events. This allows validators to quickly and efficiently reach consensus on the state of the network without needing to rely on computationally expensive calculations like in Proof of Work (PoW).
- PoS: Primarily responsible for securing the network by incentivizing validators to act honestly. Validators stake their SOL tokens to participate in the consensus process and earn rewards. This economic incentive helps to prevent malicious actors from attempting to disrupt the network.

This unique approach allows Solana to achieve high scalability and transaction throughput.

c) Algorand [168]-[169]: is a PoS algorithm that employs secret self-selection to randomly choose a leader and validators committee through cryptographic Sortition. The committee and leader selection are secured by privately computing the verifiable random function (VRF) using users' private keys and a seed generated in the previous block, without any communication among users. The selection process favors validators with the highest token balance (Algos) in their account, serving as a protection mechanism against Sybil attacks. As a result, the selected parties only discover their selection through the lottery when they propagate their winning tickets and their validation decision for the block. This renders it impractical for a malicious actor to corrupt the committee, influence their decision, or launch a DDoS attack against the members.

However, the committee's random selection process does not prevent the election of two-thirds malicious delegates. Once selected, the committee achieves consensus on the new block using a Byzantine agreement protocol called BA*, a variant of PBFT. BA* allows the participating members to reach consensus on a new block with low latency and without the possibility of forks (forks may occur with negligible probability). BA* guarantees consensus as long as an honest majority of $n > 2f/3$ exists (assuming synchronous communication) and utilizes threshold signatures for efficiency and fault tolerance. This allows the protocol to handle large numbers of users with low latency and minimal risk of forks.

d) Thunderella [170]: is a novel protocol built upon Pass and Shi's Sleepy and Snow White protocols, introducing a permissionless chain for failure recovery. Thunderella integrates two blockchains: a BFT chain representing the "fast path" and an underlying chain, considered a slow "fall-back" path, which can be any standard blockchain like Bitcoin or Ethereum. The fast path facilitates optimistic instant confirmation of transactions, while the synchronous slow chain ensures consistency and liveness.

The fast path is centralized, with a designated central authority, the Accelerator, serving as a leader responsible for transaction linearization. Concurrently, a validating committee comprising stakeholders is randomly elected using various approaches, such as utilizing all stakeholders as the committee (similar to Snow White or Algorand) or employing recent miners. The fast path confirms new transactions as long as the accelerator and $3/4$ of the committee are both online and behaving honestly; otherwise, the chain halts, awaiting recovery.

Periodically, Thunderella posts messages (referred to as alive messages) containing the hash and notarization of a checkpoint block to the underlying chain. In the event of a halt in the fast path due to a faulty accelerator or a dishonest committee, Thunderella nodes transition to the slow chain for recovery.

e) Casper Friendly Finality Gadget or CFFG [171]: is a protocol that Ethereum plans to utilize as a transitional method for transitioning from PoW to PoS (CASPER). Casper FFG represents a chain-based hybrid incorporating elements of both PoS and PoW. Blocks are mined using PoW, while PoS validators regularly verify the 100th block (known as the checkpoint) created by miners. In this setup, PoS validators do not confirm blocks or add transactions but ensure transaction finality.

To become active validators, participants lock Ethers (Ethereum's cryptocurrency) in Casper's smart contract running on the PoW chain. The verification and validation of new blocks are ensured by block validators selected based on their stake, with the voting power of each validator equivalent to the amount of their stake. Additionally, Byzantine behavior is prohibited through stake slashing.

f) Tendermint [172]: is a fault-tolerant protocol inspired by the PBFT SMR algorithm [173] and the DLS algorithm [123], designed for partially synchronous networks in both permissioned and permissionless settings. In a permissionless setting, it utilizes PoS as the underlying security mechanism.

Tendermint operates as a leader-based BFT protocol proceeding in rounds similar to PBFT's rounds. In each round, a new leader responsible for proposing a new block is elected through deterministic round-robin selection from a set of validators who have locked financial stakes. The frequency of a validator being chosen as a leader is proportional to their share of the total stake. If a validator misbehaves during their turn, they can be punished by having a portion of their deposited stake slashed. After a set duration, the stakes are unlocked and returned to the validator.

For block finality, Tendermint requires a supermajority (a quorum of over $2/3$) of all validators to validate the block. Tendermint can tolerate up to $1/3$ Byzantine validators, but if more validators disagree or become unresponsive, the network chooses to halt instead of proceeding with potentially incorrect transactions. Therefore, Tendermint prioritizes consistency

over availability.

g) LibraBFT [174]: is a variant of the HotStuff consensus protocol, specifically utilizing chained HotStuff, and introducing several enhancements and improvements. In LibraBFT, the elected leader proposes a block (a set of transactions) to extend the longest chain of requests it knows. Nodes vote for the proposed block, unless it conflicts with a longer chain they already know. In such cases, they send their votes to the next leader to help it learn the longest chain.

In contrast to HotStuff, LibraBFT employs a different approach for leader election by randomizing the process using a VRF scheme. It utilizes aggregate signatures, eliminating the need for a complex threshold key setup, to preserve the identity of validators signing quorum certificates. Additionally, LibraBFT specifies a clear implementation of the Pacemaker mechanism introduced by HotStuff to ensure the advancement of rounds. The pacemaker of each validator keeps track of votes and time, triggering a leader election when a timeout occurs due to a faulty leader, lack of votes, or after a leader successfully proposes a block.

LibraBFT guarantees safety when at least *2f+1* nodes are honest and provides liveness as long as there exists a global stabilization time (GST). These properties, coupled with rapid consensus, make it suitable for permissioned blockchains, such as the Libra blockchain.

5) DAG-based Protocols:

a) IOTA [4]: operates on a unique hybrid consensus mechanism, mixing Proof-of-Work (PoW) with a custom Tangle structure. Transactions first undergo a lightweight PoW puzzle demanding computational effort to prevent spam and ensure network integrity, as transactions within the network are fee-less. Additionally, the new transaction should randomly approve two previous valid transactions, referred to as tips, by extending them and thereby increasing their initial weights.

In IOTA, the tip selection process involves choosing two previous valid transactions (tips) to reference when adding a new transaction to the Tangle. This is performed through a proof-of-work mechanism by the end-user or device. While the protocol does not strictly dictate the tip selection process, users often employ the recommended Markov Chain Monte Carlo (MCMC) weighted random walk. This process is biased toward transactions with higher weights, contributing to the prioritization of transactions with more confirmations in the Tangle.

During the network's initial stages, a central node called the Coordinator aids in achieving consensus by emitting milestones—special transactions that solidify the Tangle and serve as validation reference points. This ensures consensus in small networks with a sparse Tangle. However, the IOTA project plans to replace the centralized Coordinators with a new voting system called "Shimmer" [175] as part of the Coordicide project [176].

b) Avalanche Protocol: Avalanche stands out as a leaderless Byzantine fault tolerance protocol that operates on a metastable mechanism through network subsampling. This process allows nodes to achieve consensus without relying on a designated leader, eliminating potential vulnerabilities associated with centralized control.

- Random Sampling: Each node repeatedly samples a small random subset of the network, significantly reducing communication overhead and preventing any single node from dominating the consensus process.
- Query Rounds: Sampled nodes participate in multiple rounds of communication, exchanging information and collecting responses. This iterative approach allows for the emergence of a consensus decision even when faced with conflicting information or Byzantine behavior.
- Threshold Adoption: Upon reaching a predetermined threshold based on the majority of responses, nodes adopt the agreed-upon decision. This ensures that the network converges towards a consistent state even in the presence of faulty or malicious nodes.

Avalanche organizes transactions in a DAG. Each new transaction extends one or more transactions. When faced with conflicting transactions, Avalanche utilizes the DAG structure, relying on a transaction's progeny (all children transactions). Corrected nodes then vote positively on valid transactions based on the entire ancestry's validity.

Avalanche ensures strong probabilistic safety, even when faced with $f \leq n/3$ Byzantine nodes. This is achieved through random sampling, minimizing the impact of individual Byzantine nodes. Nodes utilize probabilistic verification with weighted random samples for increased security, and transactions necessitate multiple confirmations from independent nodes to enhance fraud detection.

Additionally, Avalanche employs a financial mechanism, the AVA token, to safeguard the network against sybil attacks, ensuring open membership and enabling economic governance.

c) Hashgraph [80]: operates as an asynchronous Byzantine fault-tolerant protocol, employing a DAG as its underlying data structure to store events, which encompass transactions and associated details. The protocol utilizes a combination of a voting algorithm and a gossip protocol to achieve consensus.

In the Hashgraph protocol, nodes engage in random gossiping, sharing their knowledge of known events with other nodes. This can involve either events originating from the gossiping node itself or those received from other nodes. This continuous gossiping process facilitates the widespread dissemination of transactions.

Each event within Hashgraph comprises two critical elements: a timestamp and two hashes.

- The timestamp records the precise time the event was created, providing a chronological order to the events within the DAG.
- The two hashes reference two prior events: one from the gossip receiver and another generated by the sender. These references establish a clear lineage within the DAG and allow nodes to verify the validity of transactions.

Additionally, the gossiping node signs the shared event information, creating a verifiable audit trail and preventing malicious modifications. Consequently, each node in the network

possesses a comprehensive record of the transaction history, along with details about nodes that previously received this information.

To ensure consensus on the order and validity of transactions, nodes participate in a virtual voting process. IN this phase, no votes are cast or exchanged. Instead, each node leverages its understanding of the DAG (constructed through gossip history) and the timestamps of events to calculate what other nodes should vote for. A transaction is considered valid and finalized only if it receives virtual validation from at least $2/3$ of the nodes in the network. This threshold ensures a high degree of confidence in the consensus decision, even in the presence of malicious actors or network failures. Hashgraph operates under a closed membership model, implying that the total number of nodes in the network is known and fixed. This feature simplifies the virtual voting process by eliminating uncertainties about the voting quorum. Furthermore, it enables efficient operation without the need for synchronized clocks or global knowledge, making the protocol suitable for geographically dispersed networks. By leveraging virtual voting and gossip-based information dissemination, Hashgraph achieves rapid consensus on the order and validity of transactions. However, Hashgraph faces limitations with a closed membership model, reliance on patented technology, and potential transparency issues. Scalability challenges and vulnerability risks from its gossip protocol reliance further contribute to considerations for widespread adoption.

6) Federated BFT: Ripple [90] was the first the first implementation of a Federated Byzantine Agreement System (FBAS). The Federated Byzantine Agreement (FBA) approach redefines Byzantine Fault Tolerance (BFT) settings, introducing an open membership service based on a trust model. Unlike traditional BFT protocols, FBA protocols, exemplified by the Unique Node List (UNL) in Ripple and the quorum slice in Stellar, allow nodes to interact with a limited group of trusted peers, eliminating the need for a global unanimous agreement among network participants.

Ripple's consensus algorithm, the Ripple Protocol Consensus Algorithm (RPCA), functions in rounds where validators from a server's Unique Node List (UNL) strive for a supermajority consensus, typically set at least 80%. In each round, a designated server proposes a candidate set of transactions, and validators on its UNL individually vote on the proposal. The iterative process continues until a supermajority is achieved, indicating widespread agreement among validators. If consensus is not reached, the server identifies and blocks less-supported transactions, ensuring the reliability and security of the network. However, if only less than 20% of nodes in the network agree, a temporary network halt may occur. The network's safety and liveness depend on the proper server configuration and the intersection of correct nodes' UNLs. While Ripple suggests a minimum overlap requirement of 20% of the UNL, RPCA guarantees safety and liveness under specific conditions, including a minimum overlap requirement of 40% [177].

Stellar Consensus Protocol (SCP)[178] is based on Ripple protocol. It provides a first provably safe consensus, while assuming network transitivity and strong concreteness [179]. Unlike Ripple's fixed Unique Node List (UNL) and supermajority voting, SCP utilizes flexible quorum slices, allowing nodes to define their own sets of trusted validators, enabling efficient and adaptable consensus. This, coupled with federated voting and a provably safe design, ensures network stability and Byzantine fault tolerance (up to 33% Byzantine nodes). Furthermore, SCP promotes decentralization through open membership, empowering anyone to participate in the network.

VI. Execution Layer

In this section, We will unpack the execution layer, examining its essential building blocks and their properties, before introducing the now-ubiquitous execution component driving state-of-the-art technologies.

A. Components and properties

DLT systems offer two primary avenues for translating agreements into code: smart contracts, which provide extensive flexibility for crafting custom logic, and built-in scripts, which offer a more structured and protocol-defined approach to rule execution.

1) Execution environment:

a) Smart Contract Model: In this paradigm, agreements between participants are encoded as self-executing programs that operate on a predefined set of states. Typically implemented in a dedicated language or using existing programming languages like Java or C++, these programs, known as smart contracts, are executed within a specialized environment such as a virtual machine or compiler. The execution involves processing the clauses specified in the triggering transaction, producing an output (Fig. 16), and often updating states. While smart contracts can facilitate native asset manipulation (e.g., tokens or cryptocurrency), their versatility extends beyond this function. They can be utilized to implement a wide range of logic and automate complex workflows, enabling applications beyond simple asset transfers. Despite the term "self-executing," smart contracts require external triggering transactions to initiate their execution. While smart contracts enforce agreed-upon collaboration logic, they do not possess legal contract status.

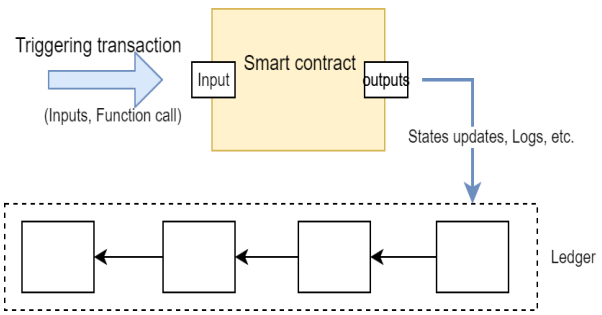

Fig. 16. An overview of the smart contract machine

b) Scripting Model: While smart contracts offer extensive programming flexibility, scripting model guides users within a predefined framework of rules and functions, ensuring adherence to specific usage patterns and protocol-level constraints. This model leverages a predefined and limited set of rules established by the DLT protocol itself, restricting the possible scenarios for implementation. By restricting the scope of permissible logic, the scripting model reduces the potential attack surface and minimizes security vulnerabilities that complex smart contracts might introduce. The predefined rules eliminate the need for developing and deploying custom smart contracts, streamlining the process and reducing the overall complexity of the DLT system. Typically found in DLTs emphasizing the secure manipulation of built-in assets (e,g. Bitcoin), the scripting model focuses on providing a framework for executing predefined rules rather than supporting universal program execution.

2) Turing Completeness: In a general sense, an environment or programming language is deemed Turing-complete if it is computationally equivalent to a Turing machine [180]. This means that a Turing-complete smart contract language or environment can execute any possible calculation within finite resources. Some DLTs support a Turing-complete execution environment, enabling users the flexibility to define intricate smart contracts. Conversely, certain DLTs employ Non-Turing complete execution environments, characterized by inherent limitations, such as the inability to have iteration structures with arbitrarily high upper bounds.

3) Determinism: Determinism is a crucial characteristic of the execution environment in DLT systems. Given that distributed programs, such as smart contracts, are executed across multiple nodes, deterministic behavior is imperative to produce consistent and identical outputs, avoiding discrepancies within the network.

To guarantee determinism within DLT systems, various approaches are employed:

- Disabling non-deterministic features: Some DLTs opt for a conservative approach by simply disabling non-deterministic operations altogether. This ensures complete predictability but restricts the range of functionalities that can be implemented.
- Sandboxing and controlled environments: Certain DLTs employ sandboxes or other controlled environments for executing programs involving non-deterministic features. This allows for some flexibility while maintaining isolation and preventing unintended consequences.
- Deterministic alternatives: Developers strive to design deterministic alternatives for non-deterministic operations whenever possible. For instance, cryptographic hash functions can be used to generate deterministic pseudo-random numbers.

4) Runtime Openness: In the majority of DLTs, the execution environment or runtime is intentionally designed as an isolated component with no connections to external networks, such as the Internet. This isolation ensures security and immutability of the ledger, but it also limits the capabilities of DLT applications. However, there are scenarios where the need to access information from outside the DLT arises, such as weather forecasts, stock prices, or exchange rates. To accommodate this requirement, various design choices have been introduced, leading to three distinct approaches:

- Isolated: Prohibiting interactions between the smart contract execution environment and external environments.
- Oracle-based: Allowing interactions with external environments through members of the network known as oracles. Oracles can be third parties or decentralized data feed services providing external data to the network.
- Open: Enabling the execution layer to connect directly to external environments.

5) Interoperability: Currently, DLT networks are intentionally siloed and isolated from each other. Interoperability, the ability to exchange data, assets, and transactions across different DLTs, emerges as a critical need to unlock the true power of this transformative technology. Given its significance, various solutions have been proposed to facilitate interoperability among different existing DLTs, falling into the following approaches:

- Sidechain [181]: A blockchain operating in parallel with another chain (main chain) allowing the transfer of data (cryptocurrency) from the main chain to itself. Sidechains typically operate in either a one-way pegged or two-way pegged mode, with the former facilitating data movement to and from the main blockchain using locking mechanisms, and the latter allowing data movement only toward the sidechain.
- Multichain [182]: A network of interconnected blockchains designed to facilitate seamless cross-chain communication and interaction. It features a central "major ledger" that governs and synchronizes transactions across various sub-ledgers, each representing a specific blockchain. This architecture enables users to securely swap assets, tokenize real-world objects, and build decentralized applications that function across diverse blockchain ecosystems.
- Interoperability protocols: function as bridges between distinct DLTs, enabling seamless communication and exchange of data or assets. They often leverage smart contracts and other technical mechanisms to establish compatibility and facilitate cross-chain interactions.
- Interoperable DLT: New DLTs are being designed with interoperability as a core principle. These DLTs incorporate features and protocols specifically intended to facilitate seamless interaction with other DLT platforms.

B. Execution layer: state of the art

In this section, we present a comprehensive overview of the most prevalent execution environments implemented in both industry and academic literature, along with a discussion of their distinctive properties. Notably, our focus extends to the Ethereum Virtual Machine (EVM), given its widespread adoption across numerous existing DLTs. In a broader classification, current DLT-based smart contract platforms can be

categorized into two primary groups: those compatible with the EVM-compatible and those not compatible with EVM.

1) Execution environments:

a) Ethereum Virtual Machine (EVM): Smart contracts in Ethereum are written in high-level languages [183] such as Solidity, LLL, Viper, or Bamboo. These programs are compiled into low-level bytecode using an Ethereum compiler, and the resulting bytecode is stored in a dedicated account on the blockchain, effectively providing it with an address.

The bytecode resides in the ledger (Fig. 17) and is assigned an address for interaction. Interactions with a smart contract are facilitated through transactions, which carry inputs and specify the function to be called. The associated bytecode for the invoked function is simultaneously executed on the Ethereum Virtual Machines (EVMs) of all network nodes, processing the transaction's payload.

Upon successful termination of the bytecode execution, the smart contract's states are updated on the blockchain's state tree, capturing the outcomes of the executed code. This process ensures that all network nodes maintain a consistent view of the smart contract's state and its interaction history.

Operating as a stack-based virtual machine, the EVM efficiently processes bytecode and manages state updates. The stack, with a maximum size of 1024 entries, employs a 256-bit register architecture, enabling simultaneous access and manipulation of the most recent 16 items. The stack's dynamic nature facilitates the execution of complex operations within smart contracts. Complementing this stack, the EVM incorporates volatile memory, organized as a word-addressed byte array. Each byte is uniquely identified by its memory address, providing a flexible data structure for contract execution. In contrast, the EVM features persistent storage represented as a word-addressable word array. This storage, comprising 2^{256} slots, each holding 32 bytes, operates as a non-volatile key-value mapping. Unlike volatile memory, the contents of storage persist across transactions, forming an essential component of the Ethereum blockchain's state.

Furthermore, the EVM operates as a sandboxed runtime, creating an isolated environment for smart contracts execution. Each smart contract running within the EVM lacks access to the network, file system, or other processes running on the host computer. As a security-oriented virtual machine designed to execute potentially unsafe code, the EVM implements strict isolation measures. To counter Denial-of-Service (DoS) attacks, the EVM incorporates the gas system, where every computation within a program must be prepaid in a dedicated unit called gas, as per the protocol's definition. If the provided gas amount fails to cover the execution cost, the transaction is unsuccessful. However, it's important to note that the gas mechanism, while mitigating DoS attacks, can still be vulnerable if settings are not appropriately configured, as demonstrated by [184] [185]. Assuming adequate memory and gas, the EVM can be considered a Turing-complete machine, allowing the execution of a wide range of calculations.

b) Bitcoin Scripting: Bitcoin employs a stack-based scripting engine. This engine operates on a stack-based architecture, utilizing a Forth-like language to define scripts that control how funds are transferred within the network. A Bitcoin script is a sequence of instructions, known as opcodes, that are loaded into a stack and executed sequentially (Fig. 18). The script follows a push-pop stack approach, executing from left to right. A script is deemed valid if the top stack item is true (non-zero) upon completion of its execution.

Bitcoin utilizes two key scripts for handling transactions:

- ScriptPubKey: This script functions as a locking script attached to the output of a transaction. It specifies the conditions that must be fulfilled for a recipient to redeem the corresponding funds. For instance, it might require a specific signature or a combination of signatures from multiple parties.
- ScriptSig: This script acts as the unlocking script. It serves as a proof that the recipient fulfills the conditions set by the ScriptPubKey, essentially unlocking the funds for transfer.

Bitcoin scripting deliberately lacks Turing completeness. Additionally, the execution time is constrained by the script's length, capped at 10 kilobytes after the instruction pointer [186]. This limitation serves to mitigate denial-of-service attacks on nodes responsible for block validation. The language used in Bitcoin scripting is acknowledged as complex

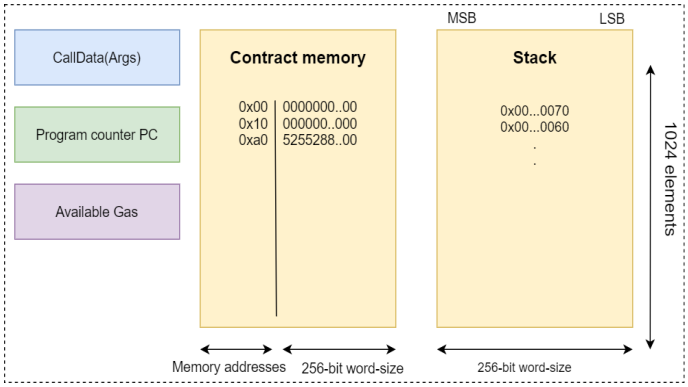

Fig. 17. The stack-based architecture of the EVM

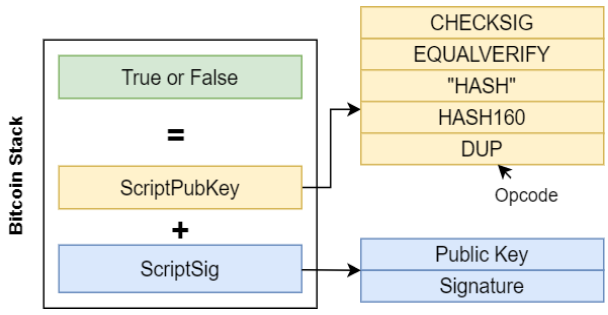

Fig. 18. Bitcoin loads and executes the locking and unlocking scripts onto the stack. If the supplied public key matches the public-key hash and the supplied signature matches the provided public key, the execution is correct (True).

and limited for smart contract development. To address this, various projects have emerged, including Ivy [187], Simplicity [188], and BitML [189]. These projects offer high-level languages with enhanced features that compile into Bitcoin scripts. Furthermore, BALZaC [190] and Miniscript provide alternatives—a high-level language based on formal models and a structured approach to writing (a subset of) Bitcoin Scripts, respectively. Notably, Rootstock (RSK) [191] was proposed as an EVM-compatible two-way pegged sidechain with Ethereum, using a merge-mining process involving both Rootstock and Bitcoin. RSK's virtual machine, the Rootstock Virtual Machine (RVM), is based on the EVM. This means that smart contracts written for Ethereum are generally compatible with Rootstock with minimal modifications. This compatibility allows developers to build smart contracts with the security of the Bitcoin blockchain.

c) Stellar: Unlike Ethereum and other platforms that rely on virtual machines and dedicated smart contract languages, Stellar [178] takes a distinct approach to smart contracts. Instead of executing general-purpose code, Stellar Smart Contracts (SSCs) are constructed from a series of interconnected transactions subject to specific constraints. Participants engaging with SSCs do not directly interact with on-chain code but instead agree to the conditions specified within transactions. These transactions are constructed using a predefined set of 13 operations, each representing an individual command that can modify the Stellar ledger. Furthermore, various constraints can be applied to transactions, enhancing their functionality. Stellar supports built-in constraints such as Multisignature, Batching, Atomicity, Sequence, Time bounds, among others [192]. SSCs are not Turing complete and developers can write SSCs in multiple programming languages (such as Python, C, Ruby, Scala, C++) using the Stellar SDK [193]. [193].

d) NXT and Ardor: NXT prioritizes security with predefined smart contract templates, known as smart transactions [194] . These templates minimize code vulnerabilities, making NXT ideal for secure transactions like asset transfers and multi-signature accounts.

Ardor, as NXT's successor, adopts the Java Virtual Machine (JVM) for smart contracts, enabling Turing-complete code execution. This unlocks a broader range of computational capabilities, supporting more complex smart contracts. Ardor further introduces Lightweight Contracts, allowing developers to automate tasks without the full computational cost of Turing-complete contracts. These Java classes are executed executed by a subset of nodes selected to run the ContractRunner addon which facilitates the execution of smart contracts on the Ardor blockchain. Together, NXT and Ardor exemplify varied smart contract approaches, catering to different blockchain use cases.

e) NEO Virtual Machine (NeoVM): NEO introduces the lightweight NeoVM (NEO Virtual Machine) [195], a virtual stack-based machine designed for processing smart contracts. NeoVM is designed to be language-agnostic, meaning that it supports multiple programming languages. Languages such as C, Java, Python can be used to write smart contracts and NEO's compiler (NeoCompiler) translates the resulting source code (with limitations [196]) into a unified bytecode. This enables cross-platform programming. Notably, NeoVM provides an InteropService that facilitates communication between the virtual machine and the underlying blockchain infrastructure. This service allows smart contracts to interact with the blockchain, access data, and perform various operations. Designed to be Turing-complete, NeoVM adopts a gas concept similar to Ethereum's, contributing to predictable resource management during contract execution.

f) EOS virtual machine: The EOS Virtual Machine (EVM) primarily uses the WebAssembly language for smart contracts. WebAssembly is a portable binary format designed to provide a high-performance, secure, and platform-agnostic environment for executing code. Thus, smart contracts are typically written in languages like C++ or Rust and then compiled into WebAssembly bytecode for deployment on the EOSIO blockchain. The compiled WebAssembly bytecode is deployed onto the EOSIO blockchain. EOS utilizes a resource model where users need to stake tokens to obtain resources (CPU and NET) to execute their smart contracts. EOS employs a unique resource model where users need to stake tokens to access resources like CPU and NET for executing their smart contracts. The staked tokens act as a form of rent, ensuring fair access to resources.

g) Cardano CCL: The Cardano blockchain consists of two essential layers: the Cardano Settlement Layer (CSL) and the Cardano Computational Layer (CCL). The focus on the Cardano Computational Layer (CCL) lies in its role as a platform for decentralized applications (DApps) and smart contracts. Operating above the Settlement Layer, the CCL enables developers to create diverse applications, such as decentralized finance (DeFi) and identity verification solutions. The CCL uses Plutus [197], which is inspired from Hashkall, as a new smart contract language. Similarly, to the EVM, CCL utilizes a cost accounting model to prevent DoS attacks. Any changes resulting from the execution of the smart contract are reflected in the global Cardano ledger. The Cardano Settlement Layer (CSL) is responsible for maintaining this ledger, tracking the state changes brought about by the execution of smart contracts.

h) Zilliqa virtual machine: Zilliqa [102] introduces a smart contract engine called Zilliqa Virtual Machine (ZVM). The ZVM adopts a sharding architecture that allows for parallel execution of smart contracts across different shards. This parallelization significantly increases transaction throughput and network scalability compared to sequential execution models The ZVM primarily uses Scilla[198] as its smart contract language. Scilla stands out as an intermediate-level programming language designed specifically for crafting safe smart contracts with formal verification. Its intermediate nature positions it as a dual-purpose tool: first, as a compilation target for high-level languages like Solidity, and second, as an autonomous programming framework in its own right.

Scilla is designed to facilitate the formal verification of smart contract programs, ensuring correctness and eliminating

known vulnerabilities at the language level. Once compiled to Scilla, the program is interpreted using an interpreter (Scilla-runner), which takes the Scilla code, the current contract's state, and the message triggering the execution as inputs, mutating the smart contract states accordingly. Although Scilla helps write more secure and easily verifiable smart contracts, it lacks expressiveness as it is non-Turing-complete. Zilliqa adopts a gas protocol based on the computation complexity, storage usage, and network congestion involved in processing a smart contract.

i) Java Virtual Machine: Hyperledger Fabric takes a distinctive approach by leveraging the Java Virtual Machine (JVM) and nodeJs [199] runtime as smart contract environments. Consequently, smart contracts, referred to as Chaincodes, can be written in Java, JavaScript, TypeScript, and, Go or any other language compatible with the supported runtimes. Fabric Chaincodes are executed within Docker containers to ensure execution isolation from the peer, providing an additional layer of security. Similarly, Corda R3 opts for the JVM as its smart contract execution environment, without containerization, and utilizes Kotlin and Java as the primary languages for smart contracts.

j) Stratis CLR: Stratis leverages the Microsoft .NET framework, and specifically, the Common Language Runtime (CLR) as an execution environment for its smart contracts. Developers write smart contracts in languages supported by the .NET framework, such as C or F and The smart contract code is compiled into Intermediate Language (IL) code [200]. IL is a low-level, platform-independent representation of the smart contract logic. This design choice allows Stratis to support CIL, enabling the use of theoretically any language that can be translated into CIL for writing smart contracts. Stratis also employs the "gas" mechanism for paid execution, similar to Ethereum's gas model, to manage resource consumption and prevent denial-of-service attacks.

k) NEM: NEM is a Java-based blockchain platform and and features a native P2P cryptocurrency (XEM). It encompasses both private and public blockchains, offering key value-added features such as ease of deployment, extensive customization, high performance, and robust security. NEM provides additional features such as the creation of custom digital assets (mosaics), identity verification through namespaces, and document timestamping using apostille.

l) MOVE for Libra: Move is a bytecode language designed for direct execution in Move's Virtual Machine (VM). Its distinctive feature is the ability to define custom resource types with semantics inspired by linear logic, reminiscent of Rust. In Move, a resource can only be moved between program storage locations and cannot be copied or implicitly discarded, similar to Rust's ownership system. This uniqueness aligns with Rust, where values can only be assigned to one name at a time, making them inaccessible under the previous name after reassignment.

Move's transaction script introduces flexibility by supporting both one-off and reusable behaviors. Smart contract functions can be executed multiple times, providing a broader range of capabilities compared to Ethereum, which is limited to invoking a single smart contract method for reusable behaviors. Moreover, Move's executable format is a typed bytecode that is higher-level than assembly yet lower-level than a source language. The bytecode undergoes on-chain checks for resource, type, and memory safety by a bytecode verifier before being executed directly by a bytecode interpreter. This approach allows Move to offer safety guarantees typically associated with a source language without adding the source compiler to the trusted computing base or incurring the cost of compilation on the critical path for transaction execution. While Move was originally designed for the abandoned Libra's blockchain, Move is still actively used in the Diem project [1]

m) Solana Runtime: Departing from conventional virtual machine reliance, Solana adopts an operating system-inspired model, enabling direct program execution on validators' machines and eliminating the interpretational overhead associated with VMs, resulting in unparalleled speed and efficiency. This architecture relies on Rust-powered programs compiled into bytecode, state-holding accounts, signed transactions, and decentralized validators. The execution process involves transaction submission, validation, account lookup, direct program execution, and consensus. Solana's optimized mechanism, featuring parallel processing through Sealevel, predictable fees, and upgradeable programs, underscores its commitment to performance, scalability, and user-friendly interactions, positioning it as a highly efficient and innovative blockchain platform.

2) Interoperability: The lack of communication between isolated DLTs has posed a notable obstacle to the advancement of the blockchain ecosystem. As a result, several suggestions have surfaced to overcome this challenge. In the following section, we underscore key strategies implemented at the execution layer to address this issue.

a) Sidechains: Various sidechains have been proposed in the DLT ecosystem. Rootstock [191] serves as a sidechain of Bitcoin, featuring an integrated Ethereum virtual machine known as RVM. The Rootstock chain is connected to the Bitcoin (BTC) blockchain through a two-way peg mechanism. This innovative approach facilitates seamless transfers between Bitcoin (BTC) and SBTC (Rootstock's native currency). This connection is established by utilizing Bitcoin scripts, allowing for interoperability between the two blockchains. In this process, users can send their BTC to the Rootstock chain, locking it up in a special smart contract. In return, they receive an equivalent amount of SBTC (Rootstock Bitcoin) on the Rootstock chain. Similarly, Counterparty [201] utilizes a sidechain architecture built on top of the Bitcoin blockchain. To facilitate asset movement onto the Counterparty sidechain, users "lock" their Bitcoin by sending them to a designated address, effectively making them unavailable on the main chain. Drivechain [202] proposes a mechanism for transferring BTC between the Bitcoin blockchain and sidechains. In contrast to most DLTs where the sidechain is a separate

[1] https://www.diem.com/en-us/

project, Cardano introduces Cardano KMZ as an integral part of its ecosystem. Cardano KMZ is a protocol facilitating the movement of assets from its two-layer CSL to the CCL (Cardano Computation Layer) or other blockchains supporting the Cardano KMZ protocol. Another noteworthy sidechain project is Plasma [203], which aims to create hierarchical trees of sidechains (or child blockchains) using smart contracts on the root chain (Ethereum). Plasma enhances Ethereum's scalability by shifting transactions to sidechains operated by individuals or a group of validators rather than the entire underlying network. Currently, Plasma is actively developed and utilized by projects such as OmiseGo [204], focusing on building a peer-to-peer decentralized exchange, and Loom [205], providing tools for constructing high-performance DApps while operating on the Ethereum network.

b) Interoperability Protocols: Interledger (ITL) [91]-[206] is a standardized protocol developed by the World Wide Web Consortium for facilitating payments across different ledgers. It consists of a network of untrusted connectors that link various ledgers and employ escrow transactions (conditional locks of funds) to facilitate transfers between accounts on different ledgers. Additionally, Atomic swap [207] allows the trading of digital assets across unrelated blockchains. Atomic swaps use Hashed Time-Lock Contracts (HTLC) [103] to coordinate operations, such as trading digital assets, on different chains. These operations are triggered by the revelation of a specific hash preimage. Alternatively, the Hyperledger project proposes the Hyperledger Labs Blockchain Integration Framework [208], a communication model enabling permissioned blockchain ecosystems to exchange on-chain data independently of the platform (e.g., Hyperledger Fabric, Quorum) without the need for intermediaries. The BTCRelay [98] project serves as a bridge between Bitcoin and Ethereum, implementing a BTCRelay smart contract on Ethereum that acts as a Bitcoin SPV (Simplified Payment Verification) node. It stores Bitcoin block headers provided by external parties known as Relayers, allowing other Ethereum contracts to verify transactions on the Bitcoin network.

c) Multi-chains: Polkadot [92] constitutes a network of interconnected chains, featuring a central connector called the Relay chain and multiple linked ledgers known as $Parachains$. The Relay chain finalizes transactions, facilitates cross-chain transactions [209], and shares states. To link the Relay chain with other networks like Ethereum or Bitcoin, Polkadot introduces bridge Parachains [210], enabling two-way compatibility. Similarly, COSMOS [211] is composed of "Zones," which are blockchain networks interconnected through a central hub known as the Cosmos Hub Network. Each Zone operates by maintaining its own state through validators that secure the blockchain and contribute to the consensus algorithm. Validators in a Zone operate independently with their own validator set. Meanwhile, intercommunication between Zones is facilitated by the Inter-Blockchain Communication (IBC) protocol [212], enabling the secure transfer of assets and information between Zones. Each Zone manages its state using a state machine, and the network as a whole benefits from the interoperability provided by the IBC protocol, allowing for a seamless exchange of value across different Zones in the Cosmos ecosystem. A key distinction between Polkadot and Cosmos lies in child chain sovereignty. Cosmos zones are independent chains built using the Cosmos SDK without sharing the same underlying environment. In contrast, in Polkadot, Parachains are dependent on and bound by the root chain's governance model, technical design choices, and limitations.

d) Interoperable chains: Gravity Hub is a blockchain designed with the capability to communicate with other blockchains like Waves[213] or Ethereum. Gravity Hub nodes can, for instance, fetch block headers from the Ethereum network and transmit them to the Waves Platform, providing proof of a specific transaction on Ethereum. Another DLT showcasing built-in interoperability is Wanchain [214]. Wanchain is a blockchain platform focused on enabling interoperability between different blockchains. Initially centered on Ethereum, it now extends its cross-chain capabilities to include Bitcoin, EOS, and projects like AION. Wanchain facilitates the seamless transfer of assets across these blockchains, contributing to the development of a decentralized financial infrastructure. Wanchain proposes to interconnect different blockchains through a decentralized bridge infrastructure utilizing secure multi-party computation (sMPC) and threshold key sharing. This approach ensures the security and privacy of cross-chain transactions by distributing key functions and enhancing overall blockchain interoperability. [215] presents further details on interoperability solutions, including other projects like AIO, Blocknet, ARK or others.

3) Determinism: The Achilles' heel of many DLTs lies in their vulnerability to non-determinism, where seemingly identical transactions can produce inconsistent results due to factors like execution order or environmental variables. This undermines data integrity and consensus, jeopardizing the very foundation of trust and reliability in DLTs. To tackle this challenge, three main approaches have emerged (table VI-B3):

- Determinism by design: This strategy eliminates non-determinism at the core, exemplified by Ethereum's Ethereum Virtual Machine (EVM). By excluding non-deterministic operations like floating-point arithmetic and external randomness sources, the EVM ensures consistent transaction execution across nodes. Solidity, the primary programming language for Ethereum smart contracts, also enforces determinism by design. However, recognizing the importance of randomness, the RANDAO [216] project proposes a decentralized autonomous organization (DAO) for registering random data on the Ethereum blockchain.
- Deterministic environments: Recognizing the need for occasional randomness, some projects embrace existing runtime environments like Java Virtual Machine (JVM) or Google's V8 engine, but modify them to enforce determinism. Examples include: Multichain [182], [217], and Stratis [200].
- Determinism by endorsement: Introduced by Hyperledger

Fabric, this novel approach leverages the network's endorsement process to achieve consensus on determinism itself. Each transaction undergoes simulation and execution by designated "endorsement" nodes. If any node produces a divergent result, the transaction is deemed invalid and rejected, preventing inconsistent outcomes from entering the ledger. Chaincode, the native smart contract language for Hyperledger Fabric, is specifically designed to facilitate deterministic execution under this model.

C. Environment Openness

In many DLTs, oracles play a pivotal role in acquiring data from external sources. Essentially, an oracle serves as a smart contract maintained by an operator, facilitating interaction with the external world. Various data feeds are deployed for smart contract systems like Ethereum, including Town Crier [218], Oraclize.it [219], Band Protocol (BAND), Tellor (TRB), API3 (API3), DIA (DIA), Witnet (WIT), and Uma (UMA). Oraclize.it [219] relies on the reputation of the service provider, while Town Crier incorporates the concept of enclave hardware root of trust [220]. Alternatively, oracles like Gnosis [221] and Augur [222] utilize prediction markets [223]. For MakerDAO [224], a decentralized lending platform on the Ethereum blockchain, ensuring both reliable price data and decentralization for its assets is a priority. To achieve this, MakerDAO adopts a multi-tiered oracle system. At its core is the Medianizer [225], which aggregates data from 14 independent price feeds, thereby ensuring accurate pricing for Ethereum. This approach aligns with the principle employed by ChainLink [14], another decentralized oracle solution that gathers data from diverse sources, contributing to a robust and decentralized data acquisition mechanism within the blockchain ecosystem. In contrast to most DLTs, Fabric's Chaincode can interact with external sources like online APIs [226]. However, if different endorsers receive divergent answers from the API, the endorsement policy fails, preventing the transaction from occurring. Other DLTs, such as Aeternity [227], integrate an oracle into the blockchain consensus mechanism [228], eliminating the need for a third party.

D. Execution Layer: Discussion

Despite the promising benefits of smart contracts, past implementations have unveiled critical security and performance pitfalls. Several recent studies have reported security issues in smart contracts [229], [230], [231], [232], [233], [234], [235], including:

a) Immutability vs. Smart Contract Security: Due to the immutable nature of most DLTs, patching and correcting detected bugs and security vulnerabilities is challenging, leading to potential fund losses, as demonstrated by the Ethereum DAO project [236]. To address this issue, various automated tools like SolidityCheck [237], Securify [238], and ChainSecurity [239] have been proposed to assist in writing secure code and analyzing bytecodes on different platforms. Additionally, ÆGIS [240] is a dynamic analysis tool designed to protect smart contracts from exploitation during runtime.

b) Insecure Languages for Writing Safe Smart Contracts: The inherent risks in certain languages used for writing smart contracts have led to the proposal of new security-oriented languages, such as Flint [241] and SOLIDITYX [242]. Moreover, in [243], dependent types from the IDRIS language [244] are employed to write provable smart contracts for Ethereum. Several projects are actively adopting formal verification, involving mathematical proofs to demonstrate that a given contract satisfies specific safety properties. For example, Scilla is designed to be amenable to formal verification, Tezos uses the Coq Proof Assistant for facilitating formal verification of smart contracts, and KEVM [245] introduces a complete semantics of the Ethereum virtual machine.

c) Untrusted Execution Environment: To enhance the security of the execution environment, particularly in private blockchain platforms where execution outcomes are susceptible to tampering, some DLT platforms, such as Sawtooth Lake or Fabric [246], execute smart contracts in Trusted Execution Environments (TEEs), such as Intel Software Guard Extensions (SGX). Although the setup of TEEs is complex, they play a crucial role in improving the privacy and security of data. TEEs securely store sensitive data, such as encryption keys, without leakage and provide evidence of the correct execution of the contract.

d) Smart Contracts Upgradability: An ongoing concern in the blockchain space is the upgradability of smart contracts. Due to the immutability of smart contracts on blockchains, upgrading them poses a challenge. Current recommendations advise adopting an upgradable design pattern, as outlined in [247] and [248], which involves deploying contracts alongside another dispatcher contract. Some platforms, such as Kadena [249], propose solutions for implementing upgradable smart contracts.

e) Lack of Interoperability: Despite considerable efforts to foster interoperability across chains, challenges persist, and accomplishments remain incomplete. This situation is attributed to two main factors. Firstly, existing interoperability projects are predominantly Ethereum-centric, necessitating more robust endeavors to facilitate interoperability with other DLTs. Secondly, the absence of a universal and unified standard hinders interoperability across different DLTs. It is noteworthy that ongoing standardization initiatives, like those by the Enterprise Ethereum Alliance and the GS1 initiative [250], aim to address this standardization gap.

f) Privacy: Many smart contract environments lack privacy, exposing contract states. Addressing this concern, Hawk [251] was proposed as a framework for constructing privacy-preserving smart contracts using cryptographic primitives like zero-knowledge proofs. Additionally, Ekiden [252] introduced a solution based on executing smart contracts in a trusted execution environment. Enigma [253] facilitates private smart contracts through the use of distributed hash-tables (DHT) and multi-party computations (MPC). Notably, the implementation of zero-knowledge proof techniques to enable private

TABLE VII
APPROACHES TO ADDRESSING NON-DETERMINISM IN DLTS WITH EXAMPLES

Approach	Blockchain Projects (Examples)
Determinism by Design	Ethereum (EVM, Solidity), Hyperledger Besu, Qtum, Tezos, Constellation Network, Elrond, Chainlink, Zcash, Narcissus Protocol, Diem, Cardano, Solana, Avalanche, NEAR Protocol, Cosmos
Deterministic Environments	Multichain, Corda, Stratis, Hyperledger Fabric (Chaincode), DFINITY, Hedera Hashgraph, EOSIO, Rchain, Hyperledger Burrow, Tezos Interledger Protocol (TIP), Polkadot, ICON, Neo3, Fabric 2.0, Hashgraph Consensus Service
Determinism by Endorsement	Hyperledger Fabric (endorsement nodes), Ripple, Stellar, Libra, EOS, Algorand, VMware Blockchain, R3 Corda (Consensus service), Quorum, Hyperledger Indy, POA Network, Byzantine Fault Tolerance (BFT) projects (Tendermint, Hyperledger Sawtooth), Ripple Consensus Protocol (RCP)

transactions or smart contracts has gained attention. However, their adoption comes at a significant cost [254] and introduces latency. Moreover, several privacy-preserving solutions may require a trusted party, potentially compromising the decentralized nature of smart contracts.

g) Layer 2 (L2) Scaling Solutions: Rollups are scaling solutions that use the underlying blockchain layer (L1) to store transaction data, while the actual transaction processing and computation occur on the rollup itself. Rollups periodically post specific data on L1 (e.g., Ethereum), such as state roots or compressed transaction data, enabling anyone to verify the validity of the rollup state and transition to a new state. This data availability on the main chain allows anyone to transition the rollup into a new state and prove the validity of the transition through validity or fraud-proof issuance. Here's a breakdown of the key components of the rollup process:

1) Off-chain Execution:
   - Prover(s): Verify transaction validity and generate cryptographic proofs.
   - Verifier(s): Check the proofs on the mainnet, ensuring the validity of the batch without needing to process individual transactions.
2) Data Availability:
   - Validity Rollups: Only proofs are submitted, requiring trust in the prover(s).
   - Fraud Proofs: Transaction data is also stored on-chain, enabling anyone to challenge fraudulent transactions.
   - Optimistic Rollups: Assume all transactions are valid unless challenged, offering faster finality but relying on fraud detection mechanisms.
3) State Commitment:
   - Rollups maintain a compressed state snapshot on the mainnet, representing the current state of the off-chain ledger.
   - State updates: Merkle trees or other efficient methods track changes in the off-chain state.
   - Withdrawal: Users can withdraw their assets from the rollup back to the mainnet.
4) Security:
   - Fraud proofs: Anyone can challenge invalid transactions in fraud-proof rollups.
   - Validity proofs: Verifier contracts on the mainnet ensure the validity of proofs in validity rollups.

Rollups generally fall into two categories: Optimistic Rollups (e.g., Optimism, Arbitrum) and ZK-Rollups (e.g., dYdX, Loopring, ZK Sync) [255]. ZK-Rollups offer significant scalability advantages over Optimistic Rollups due to their data compression capabilities. For example, while Optimism posts data after every transaction, dYdX only posts data reflecting account balances, resulting in a $1/5th$ L1 footprint and an estimated $10x$ higher throughput. This translates to lower fees for ZK-Rollups. However, ZK-Rollups may have higher computational costs during proof generation and may have limitations in smart contract functionality compared to Optimistic Rollups.

h) Modular Blockchains, A New Design Paradigm for Enhanced Scalability and Security: Many blockchain networks encounter challenges related to transaction volume, high fees, and optimization. A recent approach to achieve high scalability with robust security involves breaking down the blockchain into multiple components that can be scaled independently. In a traditional monolithic blockchain, the base consensus layer handles data availability, settlement, and execution. However, settlement and execution are typically coupled, limiting the system's overall capacity by design. In contrast, the modular blockchain paradigm separates the responsibilities, with the base consensus layer focusing solely on data availability, making transaction data available without executing it (see Figure 19). This design alleviates the burden on the base consensus layer, allowing it to concentrate on ensuring data availability. While technologies like Ethereum scaling rollups and Avalanche subnets incorporate modular components, many public blockchains remain designed as monolithic entities. Modularity has become a prominent trend in the blockchain ecosystem, with the concept introduced by the Celestia project[2]. In a modular architecture, one of the network components (execution, consensus, or data availability) is decoupled (see Figure 19), enabling projects to deploy their blockchains without the complexities of establishing a new consensus network. Several projects have embraced modularity:

- Celestia: The first modular blockchain focusing on Consensus and Data Availability. It allows easy creation of individual blockchains, enabling developers to concentrate on DApp development.
- Celestiums: Integrates Celestia and Ethereum, serving

[2]https://www.projectcelestia.com/

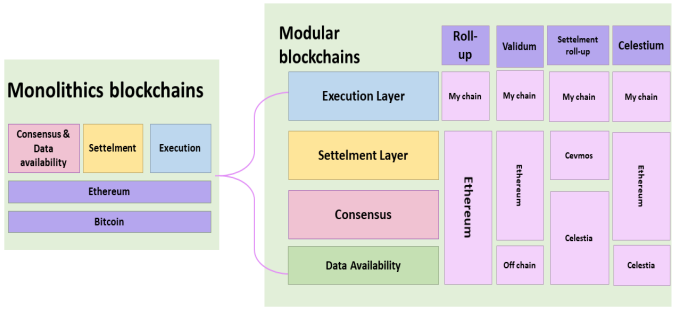

Fig. 19. Modular versus monolithic blockchains

as an Ethereum Layer 2 chain utilizing Celestia for data availability and Ethereum for settlement and dispute resolution.

- MEL (Fuel Labs): Prioritizes building the fastest modular execution layer (Modular Execution Level or MEL) on Celestia. MEL is designed as a fraud-provable computing system for modular blockchains, and Fuel v1 was launched as the first optimistic roll-up for scaling Ethereum.

The modular design offers several advantages over monolithic design:

- Scalability and Speed: Multi-level distribution allows modular blockchains to implement scalability mechanisms, significantly increasing throughput without compromising decentralization and security.
- **F**lexibility: Modularity reduces the cost of deploying smart contracts and facilitates experimentation with different technologies and environments. Notably, SwaySwap, a decentralized exchange similar to Uniswap, operates efficiently at the modular level.

VII. Application Layer

In this section, we present a concise introduction to the components and attributes outlined by our DCEA framework at the application layer. Furthermore, we provide an overview of the current state-of-the-art within this context.

A. components and properties

a) Integrability: DLTs go beyond mere data storage and transaction processing to offer real value by seamlessly integrating with existing technologies and systems. This focus on integrability is crucial for user experience and adoption. We propose integrability as a qualitative property, allowing the establishment of a "Level of Integrability" to assess if a DLT can easily integrate with Web, mobile, and other existing systems without requiring major overhauls?. This scale ranges from "High" to "Low" providing insights into the integrability of a DLT ecosystem:

- High: Indicates strong integrability with other technologies, particularly web and programming technologies.
- Low: Suggests a lack of official integrability tools or limited availability with restricted capabilities in the DLT ecosystem.
- Medium: Represents an intermediate level between the two extremes.

b) DApp Orientation and DLT's Purpose: Decentralized Applications, or DApps, represent a groundbreaking type of software that breaks away from the centralized model of traditional applications. Operating autonomously on decentralized networks, primarily blockchains, they bring resilience, transparency, and user empowerment to the forefront. In contrast to regular apps relying on centralized servers, DApps spread their architecture across multiple nodes, preventing any single entity from having control or the ability to censor. However, not all applications utilizing blockchain technology qualify as true DApps. Simply storing data or relying on timestamps while keeping core logic outside the blockchain doesn't suffice. A genuine DApp leverages the full potential of a DLT by running core functions such as business logic and state transitions directly on the chain. Recognizing the important role DApps play in shaping the future of technology, some DLTs prioritize their development by becoming "DApp-oriented." These specialized networks go beyond simply providing blockchain infrastructure and actively cater to the needs of DApp creators.

c) Wallets and Identity Management: Wallets play a crucial role in the application layer, serving as a central component for managing users' cryptographic identities. In the majority of DLTs, identities and ownership are established through public/private key pairs, making wallets a primary entry point to the network. Wallets are in charge of handling all the complex cryptographic tasks linked to creating or managing a user's cryptographic credentials, as well as signing transactions. They essentially act as a secure gateway, ensuring the safety and confidentiality of digital assets by managing cryptographic keys and authentication.

B. Application Layer: State of the Art

Considering the varied methodologies embraced by DLTs at the application layer, we present a summary of the application layer in several widely recognized DLTs.

a) Integrability: DLTs typically bring about a layer of integration that acts as a bridge between external entities and their data and execution layer. Notable DLTs such as Ethereum [65], NEO [195], and EOS [256] offer a rich toolset for integration. Ethereum provides a robust JSON-RPC API with strong support for JavaScript. In fact, Web3.js [257], a JavaScript library, facilitates interaction with Ethereum-compatible nodes over JSON-RPC. For smooth integration into legacy systems, the Camel-web3j connector [258] The Camel-web3j connector is used to integrate Apache Camel with the web3j library for interacting with Ethereum. Infura [259] offers remote Ethereum nodes that developers and users can access through APIs to interact with the Ethereum blockchain. Metamask [260], a popular cryptocurrency wallet,

utilizes Infura's nodes by default to connect to the Ethereum network. This eliminates the need for users to run their own node, simplifying wallet functionality. Similarly, EOS offers an extensive set of tools and features, simplifying integration with external systems. EOS provides various APIs, such as EOSIO RPC API, with implementations in different languages like EosJs [261], Py Eos [262], Scala Eos wrapper [263], and Eos Java [261]. These tools empower developers to interact with EOS across various programming platforms. Business-to-business (B2B) focused DLTs like Hyperledger Fabric [19] or the Corda platform [217] address integrability challenges by providing robust integration SDKs. For instance, Fabric offers a Fabric SDK that simplifies the development and integration of NodeJs and Java [199] applications within the Hyperledger Fabric blockchain framework. In contrast, other systems like Bitcoin, Litecoin, Dogecoin, or similar, offer limited integrability, as they were not designed to communicate with other systems. Bitcoin offers basic RPC features, and multiple unofficial implementations exist in various languages (e.g., BitcoinJ[264], pybtc[265]), but lacks the comprehensive integrability features found in more business-oriented DLTs.

It is noteworthy that RPC can be a potential vulnerability vector susceptible to exploitation for launching Denial-of-Service attacks, particularly when there are RPC security issues [266] or inadequate server configurations. Unprotected JSON-RPC endpoints pose a security risk, as attackers may exploit them to transfer cryptocurrencies to accounts controlled by the attackers or to acquire admin privileges over a node.

b) DApp orientation and DLT's purpose: Bitcoin and its counterparts, like Zcash, Litecoin, and others, lead the charge in a digital cash revolution. These projects share a common bold goal: to challenge traditional finance and free value from the tight control of centralized authorities making them Cryptocurrency-oriented. Other DLTs aim to provide additional functionalities beyond cryptocurrency transactions. For instance, storage-oriented DLTs like Sia Network [3], Storj [4], FileCoin [5], and Ipfs manage data storage in addition to a cryptocurrency. Similarly, service-oriented DLTs offer specific services that consume the inherent token, such as "Steemit" [6]for a social network or Namecoin for decentralized DNS. On the contrary, several DLTs are DApp-oriented, enabling developers to create diverse applications. Examples include Ethereum, EOS, Stellar, TRON, among others, which provide a more flexible development environment for building decentralized applications (DApps) with built-in tokens. For a comprehensive overview of the current blockchain DApps landscape, refer to the study by Wu et al. [267].

b-1) Decentralized Finance (DeFi): DeFi applications leverage smart contracts to facilitate a range of functionalities such as margin trading, derivatives, stablecoins, and lending/borrowing. By incorporating smart contract capabilities and utilizing data oracles like Band Protocol, DeFi platforms achieve fully permissionless, enduring, and scalable operations. A notable example is Aave [7], an open-source, non-custodial liquidity protocol that enables users to earn interest on deposits and borrow assets. Additionally, Uniswap, a decentralized exchange, allows users to trade cryptocurrencies directly with each other without the need for a central intermediary. MakerDAO, a decentralized stablecoin platform, uses smart contracts to maintain the value of its stablecoin, DAI, pegged to the US dollar. Compound, another lending/borrowing platform, allows users to earn interest on their crypto holdings and borrow assets against their collateral.

b-2) NFTs and Asset Tokenization: Tokenization [268] [269] involves the digital representation of real-world assets as tokens traded on a blockchain platform and managed by smart contracts. The tokenization enables to capitalize on traditional blockchain advantages such as indisputable ownership, transparency, trustless transactions, and an openly accessible ledger of records. Non-Fungible Tokens (NFTs), a notable application of tokenization, have garnered significant attention in the blockchain realm. They represent unique and indivisible digital assets on a blockchain, with each NFT being distinct and often associated with digital art, collectibles, virtual real estate, or other unique digital items.

b-3) Prediction Markets: Leveraging smart contracts and data oracles, prediction markets on blockchain platforms enable the incorporation of real-world, open-internet data. This includes information on market movements, weather conditions, sports results, and more. The use of smart contracts and oracles enhances the transparency and reliability of these prediction markets. Developers and end-users can create niche betting or prediction platforms, where the outcome of events is automatically determined and payouts are executed based on predefined rules encoded in smart contracts. This eliminates the need for centralized authorities in overseeing and settling predictions, offering a decentralized and trustless environment for participants.

c) Wallets and Identity Management: Bitcoin, as the original cryptocurrency, utilizes a simple private key system for user access and management, with wallets, whether software, hardware, or paper, ensuring secure storage and transfer capabilities. Ethereum's wallets act as gateways to its ecosystem, managing Ether (ETH) and ERC-20 tokens (or others), connecting to DApps, and enabling transaction signing. Notable options like MetaMask, MyEtherWallet, and Ledger hardware wallets enhance user interaction within the Ethereum network. Ethereum has introduced Account abstraction (AA). By decoupling transaction signing from the traditional private key model, AA introduces a layer of programmability and flexibility that opens up exciting possibilities for wallets. AA enhances wallet security by reducing direct user management of private keys, implementing advanced recovery mechanisms like multi-signature schemes, and allowing programmable permissions for added security against unauthorized transactions. Solana, emphasizing speed, offers wallets such as Phantom and

[3]https://sia.tech/
[4]https://www.storj.io/
[5]https://filecoin.io/
[6]https://steem.com/
[7]https://aave.com/

Solflare, prioritizing rapid transaction processing, integration with dApps, and support for staking and managing Solana Programmatic NFTs (pNFTs). Hyperledger Fabric, designed for enterprise use, focuses on permissioned networks with access control, integrating wallets with identity management systems based on PKI to ensure authorized access and secure transactions.

C. Application Layer: Discussion

The concept of decentralized applications (DApps) takes full advantage of the unique characteristics of DLTs but also inherits some of their limitations. According to a recent report [270], DApp projects in 2019 faced ongoing challenges, including poor user attraction and retention due to complex user experiences or the perceived uselessness of their services. As a result, a significant number (estimated at 1300) of DApps were abandoned in 2019 [270]. Successful DApps, such as CryptoKitties (an online game built on Ethereum) or EIDOS (an EOS token), highlighted a critical limitation of public DLTs, namely, scalability. The popularity of these DApps led to unprecedented congestion on their underlying chains, causing thousands of unvalidated transactions. To address this, developers are increasingly exploring the use of L2, rapid sidechains like Lightning, Raiden, or Loom Network, which offer faster transaction processing compared to the main chains (e.g., Ethereum).

VIII. Evaluation and discussion

In this section, we undertake a comparative analysis and evaluation of DLTs. The analysis is conducted at two levels; First, we compare and evaluate, at a high level, the chosen DLTs based on the properties outlined by our framework; Second, we compare and evaluate multiple consensus protocols against the criteria introduced in the section V.

A. A Comparative evaluation of Blockchain and blockchain-like system based on DCEA framework

Tables XI and XII provide a comprehensive overview of a diverse and substantial selection of Distributed Ledger Technologies (DLTs) from both industry implementations and recent research contributions. The comparative analysis encompasses four key dimensions: the composition of the four-component DCEA framework, operational scope, level of decentralization, and the higher taxon classification. In this subsection, our attention is directed towards a detailed examination of governance and conflict resolution methodologies, with a concurrent evaluation of decentralization. These properties assume an important role in determining the categorization of a system as a blockchain or not.

1) Decentralization: Decentralization is a fundamental element in the design of DLT. Our assessment delves beyond the surface, scrutinizing the topology of nodes, the cost and distribution of running full nodes, and the mechanisms governing decision-making. A truly decentralized DLT resists the control of any singular entity, whether physical or logical. This multi-layered analysis dissects the network's backbone, ensuring power is distributed rather than concentrated across nodes and participants. A DLT is deemed decentralized if it avoids physical or logical control by a singular entity, with consideration given to the aforementioned factors. The 44 DLTs, as listed in Table XI, underwent assessment on a three-step scale:

- centralized : in this model, a single entity or a small group of entities control all aspects of the system. They possess full decision-making power, maintain the ledger, and dictate the rules and processes for transaction validation and consensus.
- semi-decentralized : This model introduces some elements of decentralization but still retains significant control in the hands of specific entities.
- fully decentralized : This model aims to distribute power and control among all participants in the network, eliminating the need for a central authority. Decisions are made collectively through consensus mechanisms, and no single entity has exclusive control over the system.

The pursuit of enhanced performance and scalability in DLT projects often compels developers to grapple with the fundamental tension between decentralization and centralization. Our comparative analysis reveals striking examples of this compromise, where projects like Ripple, Stellar, and Libra integrate centralized elements to optimize network efficiency, albeit at the expense of absolute decentralization. In Ripple's case, a pre-defined "starter list" of trusted nodes, chosen by the founding team, serves as the initial validator set. While users can theoretically update this list, concerns about divergent pathways and compromised security (as noted by Armknecht [177]) deter most users from doing so. This static list effectively cedes operational and decisional power to the starter nodes, creating a de facto centralized system heavily influenced by the Ripple company, which also holds a significant stake in the network's token (XRP). This centralized control raises concerns about potential censorship, manipulation, and vulnerability to single points of failure. Similarly, Stellar's network exhibits dependence on two core validators managed by the Stellar Foundation. [271] highlights that deleting these nodes could trigger a network-wide collapse, further accentuating the centralized nature of its governance and operational structure. These examples illustrate the complexities and trade-offs inherent in the decentralization vs. performance dilemma for DLT projects. While centralized elements can offer undeniable benefits in terms of speed and stability, they simultaneously introduce weaknesses in trust, security, and resilience. The long-term viability and societal impact of these projects hinge on finding innovative solutions that bridge this gap and pave the way for scalable, yet truly decentralized, DLT frameworks. Conversely, most Proof-of-Stake (PoS)-based DLTs exhibit decentralization. Nonetheless, PoS faces criticism for potentially favoring entities with a larger token stake, leading to centralized validation in instances of unfair token distribution. [272] demonstrates the significant impact of the ratio between block reward and total network

stake on the decentralization of PoS networks. IOTA serves as an example of a semi-decentralized DLT, utilizing Coordinator (COO) nodes run by the IOTA foundation to safeguard the network from 34% attacks. Transactions cannot be confirmed unless approved by the Coordinator through milestones. Delegated Proof-of-Stake (DPoS)-based DLTs face criticism for susceptibility to validation centralization due to the potential collusion of elected validators.

The case of EOS exemplifies this tension. Despite its claim to democratic governance through staking-based voting, concerns persist regarding centralization tendencies within its ecosystem. Studies have uncovered correlations in votes for different candidates, suggesting potential collusion among a limited group of influential actors [273]. Additionally, the extreme concentration of EOS tokens, with the top 100 holders possessing over 75.13% of the total supply [274], raises worries about the undue influence they may wield in validator selection and network governance. These findings align with broader concerns expressed by researchers such as Micali, who caution against poorly designed incentive mechanisms exacerbating centralization within blockchains. Kwon et al. echo these sentiments, emphasizing the inherent challenges in achieving robust decentralization in permissionless blockchain systems [275].

2) Governance: In our assessment, we prioritized the examination of the decentralized governance structures implemented in the chosen Distributed Ledger Technologies (DLTs). Diverse decision-making approaches are employed across different projects to modify DLT protocol parameters and upgrade network rules, reflecting varied political forms of governance. Networks such as Bitcoin and Ethereum adopt an anarchic governance model. In these systems, a anybody can initiate an improvement proposal to address issues or to change protocol settings (e.g., increasing block size). The proposal undergoes public discussion, and upon garnering sufficient favorable peer review, it is implemented into the project's codebase. Depending on the technical enhancements, deploying it might need either a soft fork or a hard fork. A soft fork is compatible with the existing system and doesn't require all network nodes to accept it, whereas a hard fork requires nodes to update their software. For instance, Bitcoin's SegWit upgrade was a soft fork, and Ethereum's shift from Proof-of-Work to Proof-of-Stake was a hard fork. The risk with a hard fork is that it may lead to a split in the network, as upgraded nodes disconnect and reject nodes with a different protocol version that didn't undergo the hard fork. Bitcoin employs on-chain governance through Version Bits voting [276] for Soft-fork implementations, measuring miner support. Projects like Qtum aim to avoid significant disruptions, such as hard forks, for minor changes like block size and gas parameters. They achieve this by including built-in features that allow participants to collectively decide on system adjustments through dedicated smart contracts, as seen in the Decentralized Governance Protocol. Similarly, Tezos utilizes the Tezos governance protocol, enabling the network to decide on protocol upgrades. In enterprise-grade DLTs such as Hyperledger Fabric designed primarily for private or consortium contexts networks governance is managed by a predefined governing body. For example, a technical or administrative committee makes decisions, halts the network [277], and implements upgrades.

3) Conflict Resolution: An intriguing aspect within the selected DLTs lies in their methodologies for transaction ordering and conflict resolution. In systems employing Nakamoto consensus protocols, miners autonomously arrange transactions in blocks, validate them, and include them in the chain of blocks. In the event of conflicting chains, nodes shift toward the longest chain. DPoS protocols also follow the longest chain rule when a block fails to receive the majority (*2/3 +1*) of votes from block producers. Nevertheless, a recent study [304] questions the safety of applying the "longest-chain PoS" rule to PoS protocols. Conversely, Nano's DAG relys on balance-weighted voting such that the network reaches consensus through individual node decisions and voting power.

Certain networks, such as Hyperledger Fabric, utilize a specialized ordering service and dedicated mechanisms. The Ordering Service sequences transactions into blocks, ensuring a specific processing order and contributing to conflict prevention in the ledger. Hyperledger Fabric, employs Multi-version Concurrency Control (MVCC) to allow parallel transactions without conflicts. Each ledger key maintains a version history, preventing data inconsistencies. Additionally, an Endorsement Policy mandates validations from specific peers before submitting transactions, preventing invalid or conflicting entries. Similarly, Corda employs the notary service [305] to order transactions and detect conflicts utilizing multiple consensus mechanisms like RAFT, PBFT, or custom implementations. [305]. Once the notary service validates a transaction and adds it to the ledger, it becomes final and binding for all participating nodes. This provides certainty and immutability to the transaction data. Transaction ordering in IOTA is partially ensured by senders and weights. Senders choose two previous confirmed transactions (called "tips") to attach their new transaction to, creating a DAG structure. While some suggest the tips selection can be random from confirmed transactions, the IOTA Foundation recommends using a Markov Chain Monte Carlo (MCMC) weighted random walk. This method prioritizes attaching to heavier branches, which helps them grow faster and become the dominant valid tangle. Hashgraph aims to ensure "ordering fairness" [6] through its gossip-about-gossip protocol. This concept ensures that transactions on a blockchain or distributed ledger are processed and committed in the same order that they were received by the network nodes. Recognizing the significance of correct and fair ordering, Asayag et al. proposed Helix [306] which leverages an "in-protocol randomness" mechanism to elect validators from a larger pool and determine which messages should be included in a block. This randomness helps prevent manipulation and ensures fair ordering of transactions. Moreover, Kelkar et al. introduced Aequitas [307] which uses a lottery-based approach to select validators and assign transaction slots within a block. This lottery is designed to be provably fair and

TABLE VIII
APPLICATIONS OF BLOCKCHAIN AND BLOCKCHAIN-LIKE SYSTEMS

	Cryptocurrency and payment	Digital identity	IoT	Healthcare	Logistics	Smart city	Telecom	Smart grid	AI	Prediction markets	Security	E-government	Banking	Finance	Banking	Games and E-sports
Blockchain	[1] [278]	[279]	[280] [281]	[282]	[283]	[284]	[285]	[286]	[287] [288]	[289]	[290] [291]	-	-	-	-	[292]
Blockchain-like	[293]	[294]	[295]	[296]	[297]	[284]	[298]	[299]	[300]	-	-	[301]	[297]	[302]	[303]	-

tamper-proof.

4) Application Scenarios for Blockchain and Blockchain-like Systems: In recent years, the utilization of DLTs proliferated across diverse domains. The examples presented in Tables VIII and I highlight numerous sector-specific applications. While there may be some overlap in their application domains, it is clear that these technologies address distinct business scenarios with unique requirements. Broadly speaking, blockchain systems are well-suited for global decentralized C2C models, emphasizing high user autonomy. In contrast, blockchain-like platforms find greater utility in B2B use-cases within a single organization or consortium, especially in corporate settings where controlled governance and restricted data access are crucial. For instance, a consortium of financial actors may opt for a blockchain-like system to limit the sharing of financial transaction details to relevant parties. Conversely, blockchain systems excel in egalitarian networks, providing considerable freedom to end-users, such as enhanced transparency, decentralized control, and heightened security. A prime example is the utilization of cryptocurrencies like Bitcoin and Ethereum. In these blockchain networks, participants benefit from a more equitable distribution of authority and access, fostering a trustless environment where transactions are verifiable and immutable.

B. A comparative analysis of consensus protocols

For a better comparison of DLTs, it is pertinent to compare and contrast the consensus mechanisms separately as they have a direct influence on other aspects and properties of the DLTs (e.g. centralization, membership, scalability). To achieve a thorough comparison, we adopt the metrics defined in section V, and we consider the evaluative scale presented in Table IX. Table X and Figure 20, summarize the result of our analysis and evaluation.

An analysis of the results reveals important observations about the strengths and limitations of the analyzed protocols. Nakamoto consensus protocols face criticism for their performance limitations and susceptibility to centralization [308]. Bitcoin's proof-of-work, a well-known representative, is notorious for its energy consumption and significant latencies [309]. Pass and Shi [310] show that, in order for Nakamoto consensus protocols to maintain security, the block interval must be set as a constant factor larger than the network's maximum delay. Consequently, the inherent design of these protocols imposes a limit on the potential network's throughput, resulting in a scalability bottleneck.

TABLE IX
THE EVALUATIVE SCALE USED TO COMPARE THE CONSENSUS PROTOCOLS

Liveness	Strong	Weak	
Safety	Strong	Weak	
Transaction throughput	Very high (>1000 tps)	High([1000,100 tps])	Low(<100 tps)
Finality	Absolute	Probabilistic	
Network Model	Synchronous	Partially synchronous	Asynchronous
Adversarial mode	Strongly adaptive	Middly adaptive	Adaptive | Non adaptive
Identity model	Permissioned	Permissionless	

Despite these limitations, Nakamoto consensus protocols exhibit interesting properties. They demonstrate resilience against significant Byzantine minorities ($n > f/2$) with anonymous open membership. Moreover, they do not necessitate extensive message exchange between nodes to reach an agreement, allowing them to scale efficiently to a large number of participants. These characteristics make them widely employed in global cryptocurrency networks and well-suited for public permissioned networks, where implementing economic incentives is feasible as a safeguard mechanism to ensure liveness and network security.

PoS is praised as a better alternative to Nakamoto protocols, as it is energy-efficient and allows unlimited open membership with equivalent fault tolerance ($n > f/2+1$). However, it is vulnerable to numerous security threats such as Short and Long-range attacks [311] and the nothing-at-stake attack. These issues are mitigated by some PoS protocols, such as Ouroboros, which employs only one designated leader in each round. Moreover, PoS protocols face the weak subjectivity issue, where a node joining the network for the first time or after a long absence has to rely on other nodes to synchronize the correct ledger. This dependence undermines the trustless nature of blockchains entirely.

Algorand is a PoS protocol designed to scale independently of the network's size [312]. It achieves this scalability by utilizing the verifiable random function (VRF) to randomly select private delegates in a representative way without the need for coordination between nodes. However, similar to other PoS protocols, Algorand is secure only against a $1/3$ adversary bound. Additionally, it results in an inherently slower block production rate compared to protocols with probabilistic finality, such as Snow White or Ouroboros, due to the requirement of multiple rounds.

On the other hand, BFT protocols are efficient and well-suited for small or midsize permissioned networks. However, they involve extensive communication exchanges between nodes and demand accurate knowledge of membership. Take PBFT, for example, which incurs quadratic communication

TABLE X
A COMPARISON OF CONSENSUS PROTOCOLS

	Network Model	Adversarial model	Adversary mode	Fault tolerance	Identity Model	Safety	Liveness	Finality	transaction throughput	Type*
PBFT	Asynchronous	Threshold Adversary	NA	f<n/3	Permissioned	Strong	Weak	Absolute	Very high	Blockchain-like
RAFT	Asynchronous	Crash-failure	NA	f<n/2	Permissioned	Strong	Weak	Absolute	Very high	Blockchain-like
RIPPLE	Asynchronous	Threshold Adversary	Adaptive	f<n/5	Permissionless	Strong	Weak	Absolute	High	Blockchain-like
STELLAR	Asynchronous	Threshold Adversary	Adaptive	f<n/3	Permissionless	Strong	Weak	Absolute	Very High	Blockchain-like
HONEYBADGER	Asynchronous	Threshold Adversary	Non adaptive	f<n/3	Permissioned	Strong	Strong	Absolute	Very high	Blockchain-like
POW	Partially-synchronous	Threshold Adversary	Strongly adaptive	f<n/2	Permissionless	Weak	Strong	Probabilistic	Low	Blockchain
BITCOIN-NG	Partially-synchronous	Threshold Adversary	Strongly adaptive	f<n/2	Permissionless	Weak	Strong	Probabilistic	Very high	Blockchain
BYZCOIN	Partially-synchronous	Threshold Adversary	Non adaptive	f<n/3	Permissionless	Strong	Weak	Absolute	Very high	Blockchain
GHOST	Partially-synchronous	Threshold Adversary	Non adaptive	f<n/2	Permissionless	Weak	Strong	Probabilistic	High	Blockchain
CASPER FFG	Asynchronous	Stake Threshold Adversary	Non adaptive	f<n/3	Permissionless	Weak	Strong	Probabilistic	High	Blockchain
CASPER TFG	Asynchronous	Stake Threshold Adversary	Non adaptive	f<n/3	Permissionless	Strong	Weak	Absolute	High	Blockchain
DPoS (EOS)	Partially-synchronous	Stake Adversary	Non adaptive	f<n/3	Permissionless	Weak	Strong	Absolute	High	Blockchain-like
OUROBOROS	synchronous	Stake Threshold Adversary	Middly adaptive	f<n/3	Permissionless	Weak	Strong	Probabilistic	High	Blockchain
OUROBOROS PRAOS	Partially-synchronous	Stake Threshold Adversary	Strongly adaptive	f<n/3	Permissionless	weak	Strong	Probabilistic	High	Blockchain
OUROBOROS GENESIS	Partially-synchronous	Stake Threshold Adversary	Strongly adaptive	f<n/3	Permissionless	Weak	Strong	Probabilistic	High	Blockchain
OUROBOROS CHRONOS	Partially synchronous	Stake Threshold Adversary	Strongly adaptive	f<n/3	Permissionless	weak	strong	Probabilistic	High	Blockchain
TENDERMINT	Partially synchronous	Stake Threshold Adversary	Non adaptive	f<n/3	Permissioned	Strong	Weak	Absolute	High	Blockchain-like
ALGORAND	Partially synchronous	Stake Threshold Adversary	Strongly adaptive	f<n/3	Permissionless	Strong	Weak	Absolute	High	Blockchain
THUNDERELLA	Synchronous	Stake Threshold Adversary	Middly adaptive	f<n/3	Permissionless	Strong	Weak	Absolute (Fast Path)	Very high	Blockchain-like
HOTSTUFF	Partially-synchronous	Threshold Adversary	Adaptive	f<n/3	Permissionless	Strong	Weak	Absolute	Very high	Blockchain-like
LIBRABFT	Partially synchronous	Threshold Adversary	Adaptive	f<n/3	Permissionless	Strong	Weak	Absolute	Very high	Blockchain-like
SPECTRE	Partially synchronous	Threshold Adversary	Non adaptive	f<n/3	Permissionless	Strong	Weak	Probabilistic	High	Blockchain-like
IOTA	Partially synchronous	Threshold Adversary	Non adaptive	f<n/3	Permissionless	Strong	Weak	Probabilistic	High	Blockchain-like
HASHGRAPH	Asynchronous	Threshold Adversary	Non adaptive	f<n/3	Permissioned	Strong	Weak	Probabilistic	Very high	Blockchain-like
SNOW WHITE	Asynchronous	Stake Threshold Adversary	Strongly adaptive	f<n/3	Permissionless	Weak	Strong	Probabilistic	High	Blockchain
AVALANCHE	Partially synchronous	Stake Threshold Adversary	Non adaptive	f<n/3	Permissionless	Strong	Weak	Probabilistic	Very high	Blockchain

f : is the faulty nodes or actors and n: is the total number of nodes that are coming to consensus.
* We present the DLT type according to the analysis performed and summarized in table VII.

overhead, making it challenging to scale in terms of the number of participants. Additionally, these protocols are vulnerable to Sybil attacks, rendering them unsuitable for public DLTs but well-suited for private DLTs.

Despite these limitations, some protocols aim to leverage the advantages of PBFT to construct highly performant open consensus protocols for public blockchains while mitigating its inherent drawbacks. Tendermint [313], for example, combines BFT and DPoS. It prevents Sybil attacks and offers open membership based on proof of stake.
Safety and liveness are crucial properties for DLTs. Figure 20 provides an expressive overview of the safety and liveness of the analyzed consensus mechanisms. We observe that PoW-based protocols sacrifice safety (forking can happen) for strong liveness. These protocols provide probabilistic safety as the network converges toward a canonical chain using the probabilistic Nakamoto's longest chain fork choice rule (or a similar rule) coupled with economic incentives. Similarly, many PoS-based protocols favor safety over liveness. For instance, Algorand and Casper FFG ensure safety and liveness if dishonest participants control less than $1/3$ of the deposited stake. Ouroboros is proven to achieve safety and liveness in synchronous settings, under the assumption of an honest majority of all stake in the system, even if a significant portion of participants is offline [158]. Algorand achieves safety with a "weak synchrony" assumption, whereas to achieve liveness, Algorand assumes strong synchrony. Other protocols such as BFT protocol and its variants [124], [314], PBFT implementations and FBA favor fault tolerance and termination over safety. This means that in case of accidental fork, the network halts waiting for recovery (restoration of the consensus). In the Stellar Federated Byzantine Agreement, nodes choose their quorum slice (set of trusted nodes) according to their trust

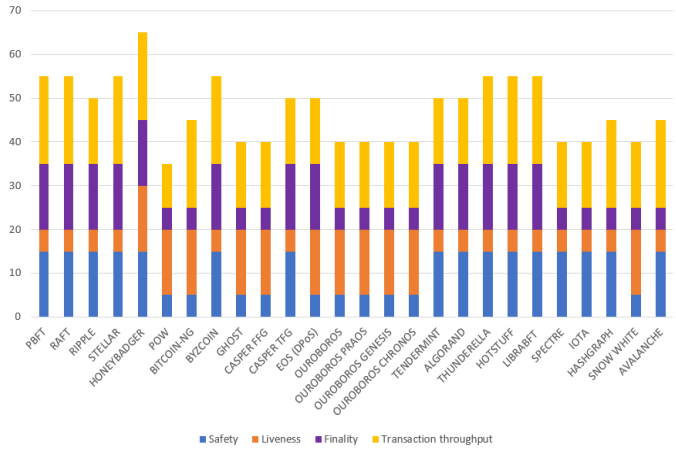

Fig. 20. Visual comparison of consensus protocols in terms of safety, liveness and finality

relationship. Thus, the safety and security of the protocol is highly dependent on the structure of the quorum slices. However, [271] shows that the Stellar system is significantly centralized and proved that FBA is not better than PBFT in terms of safety and liveness.

Another important aspect to consider when analyzing consensus protocols is finality. Protocols such as Snow white [161], Ouroboros [315], Casper FFG [171], and PoW achieve probabilistic finality. DLTs adopting these probabilistic protocols define rules to avoid the risk of double spending by urging users to wait for a given delay –usually expressed number of blocks— before considering the transaction as final. For instance, in Bitcoin, it is widely advised to wait at least for 6 confirmation blocks (around 60 minutes) before accepting the validated payment. The reason behind, is that

TABLE XI
COMPARATIVE ANALYSIS TABLE FOR SELECTED DLTS

	DATA LAYER				CONSENSUS LAYER					EXECUTION LAYER						APPLICATION LAYER		
DLT solution	Data Structure	Data sharability	States management	Immutability	Consensus	Governance	Governance nature	Transaction ordercr	Fork management (Non-malicious forks)	Turing complete-ness	Environment openness	Execution environment	Determinism	Languages	Interoperable	Orientation	Integrability	Purpose
Æternity	CoB	G	On	S	GHOST, Bitcoin-NG (for security) and PoS (for governance)	De	Bl	Randomly selected miner	Lc	Tc	Op (Built-in oracle)	Æternity VM	Dc	Sophia, Varna, solidity	No	Cy and DA	H	Ge
Algorand	CoB	G	On	S	Algorand	Ol	Ex	Randomly selected leader (stake weighted election)	Nf	NTc	Is (OB)	Algorand VM	Dc	TEAL	No	Cy and DA	M	Ge
Ardor	CoB (One parent chain with multiple child chains)	G	On	S	PoS	Ol	Bl	Forging account (NXT forging algorithm)	Lc	NTc	Op	Deterministic Java VM	Dc	Java	Yes (between child chains)	Cy and DA	M	Ge
Bigchaindb 2.0	HDS (a database and a CoB)	G	On/Off	W	Tendermint	Ol or De	Ex	Elected leader orders transactions by arrival time	Nf	Nsp					No	DA	M	Bo
Bitcoin	CoB	G	On	S	PoW	An	Bl and Ex	Randomly selected miner	Lc	NTc	Is	Script runtime	Dc	Bitcoin scripting, Miniscript	No	Cy	M	Po
BitShares	CoB	G	On	S	DPoS	De	Bl	Elected Witness	Lc	NTc	Is	Bitshares runtime	Dc	C++	No	Cy	L	So
Byzcoin[a]	Skipchain	G	On	S	Byzcoin	An	Ex	Randomly selected miner	Lc	NTc	Is	Byzcoin runtime	Dc	Go	No	Cy	L	Ge
Cardano	CoB	G&R	On	S	Ouroboros	De	Bl	Randomly selected leader	Lc	Tc	Is	IELE VM	Dc	IELE and Plutus	Yes (with cardano sidechain)	Cy and DA	M	Ge
Corda (R3)	DDB	G	Off	W	RAFT, BFT-SMaRt or KAFKA	Di, Ol or De	Ex	BFT-SMaRt distributed notary	Nf	Tc	Is (OB)	Java VM	NDc	Kotlin, Java	No	DA	H	Bo
Cosmos	CoB (Hub and multiple zones)	G&R	On	S	Tendermint	De (with Veto)	Bl	Elected block producer	Nf	Tc	Is	WebAssembly VM	Dc	Wasm languages (cosmos SDK)	Yes (Cross-chain Interoperability)	Cy and DA	M	Ge
Decred	CoB	G	On	S	PoW and PoS	De	Bl	Randomly selected miner	Lc	NTc	Is	Script runtime	Dc	Modified bitcoin scripting	No	Cy	L	Po
Elrond	SCoB (Metachain and shards)	G	On	S	Secure PoS (variant of Algorand)	De	Bl	Randomly selected proposer	Lc	Tc	Is (OB)	Elrond VM	Dc	IELE, Wasm languages	Yes (between Metachain and shards)	Cy and DA	L	Ge
EOS	CoB	G	Off	W	Transactions-as-PoS	Ol	Bl	Elected block producer	Lc	Tc	Is (OB)	WebAssembly VM	Dc	Wasm languages	Yes (with EOSYS sidechain)	Cy and DA	H	Ge
Ethereum 2.0	CoB	G	On	S	PoS	An	Ex	Randomly elected miner	Lc	Tc	Is (OB)	Ethereum VM	Dc	Solidity, Vyper, LLL, Julia	No	Cy and DA	H	Ge
Ethereum 1.0	CoB	G	On	S	PoW (ETHASH)	An	Ex	Randomly elected miner	Lc	Tc	Is (OB)	Ethereum VM	Dc	Solidity, Vyper, LLL, Julia	No	Cy and DA	H	Ge
Ethereum Enterprise	CoB	G&R	On	W	PoA, RAFT or IBFT	Di, Ol or De	Ex	Elected leader	Lc	Tc	Is	Ethereum VM	Dc	EVM languages	No	DA	H	Bo
Exonum Enterprise	CoB	G & R	On	S	Exonum protocol	Di, Ol or De	Bl	Predefined leader	Nf	Tc	Op (Built-in oracle)	Java VM and Rust runtime	NDc	Java, Rust	Yes (one way with Bitcoin)	DA	M	Bo
Elastos	CoB (main chain and sidechains)	G	On	S	DPoS and POW (merged mining with Bitcoin)[b]	De	Bl	Randomly elected miner	Lc	Tc	Is	CAR runtime, EVM (Ethereum Sidechain) , NEOVM(NEO sidechain)	Dc	C++, Java, Swift, JavaScript, Golang, solidity	Yes (Sidechains can transact with each other)	Cy and DA	M	Ge
Filecoin	CoB	G	On	S	Proofs-of-Spacetime	Ol	Bl	Elected leader (using Expected Consensus (EC))	Lc/ Hc[c]	Tc	Is	FilecoinVM	Dc	Golang	No[d]	DA	H	DSo
Hashgraph	DAG (Transaction-based DAG)	G&R	On	W	Hashgraph	Ol	Bl	Fair ordering via Consensus Time Stamping	Nf	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	Yes (with Hyperledger Fabric)	Cy and DA	M	Ge
Hyperledger fabric	HDS	G&R	Off	W	PBFT	Ol or De	Ex	Ordering service node	Nf	Tc	Op	Java VM and Nodejs runtime	Dc	Go, JavaScript	No	DA	H	Bo
IOTA	DAG (Transaction-based DAG)	G	On	W	IOTA	Ol	Ex	End-user and coordinator node	Hb	Nsp (Qubic a smart contract protocol is under development)[e]					Yes (with Hyperledger Fabric)	Cy	M	IOT
Lisk	CoB (Main and sidechains)	G	On	S	DPoS	De	Bl	Elected leader	Lc	Tc	Is	NodeJs runtime	Dc	JavaScript	No	Cy and DA	H	Ge
Multichain	CoB	G&R	On/Off	W	Mutlichain protocol (variant of PBF)	Ol	Ex	Predefined leader	Lc	Multichain do not support smart contracts					No	DA	L	Ge
NEO	CoB	G&R	On	S	DBFT (Delegated Byzantine Fault Tolerance)	Ol, De	Bl and Ex	Elected leader	Nf	Tc	Is (OB)	NeoVM	Dc	.NET and JVM languages (Java, Kotlin)	Yes (Between private blockchains connected to NEO)	Cy and DA	M	Ge
Omniledger[f]	SCoB	G	On/Off	S	ByzCoinX (Variant of Byzcoin)	Ns	Ns	Randomly elected leader	Nf	Not supported					Yes (with its shards)	-	-	Ge
Parity substrate	CoB	G&R	On	W	Pluggable consensus (Hybrid PBFT, Aurand, Rhododendron, Shaft, ouroboros, PoW)	Ol	Bl	Depends on the chosen consensus mechanism	Nf	Tc	Is	WebAssembly VM	Dc	WASM languages	Yes (with Polkadot)	DA	H	Bo
Polkadot (Relay chain)[g]	CoB (relay chain)[g]	G	On	S	GRANDPA and BABE (PoS)	De	Bl	Randomly elected leader	Lc	Tc	Is	WebAssembly VM[h]	Dc	WASM languages	Yes (Cross-chains interoperability)	Cy and DA	H	Io
Quorum	CoB	G&R	On	W	IBFT or RAFT or Clique POA	Di, Ol or De	Ex	Elected leader	Nf	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	No	DA	H	Ge
Qtum	CoB	G	On	S	PoS	De	Bl	Elected leader	Lc	Tc	Is (OB)	Ethereum VM and X86 VM	Dc	EVM languages, C, C++, Rust, Python	Yes (Atomic swap with bitcoin)	Cy and DA	Hign	Ge
Ripple	DDB (chain of ledgers stored as key-value)	G	On	S	Ripple (FBA)	Ol	Ex	Validating nodes converge toward a canonical order	Nf	NTc	Is	Built-in specialized payment types	Dc	JavaScript	No	Cy	M	Po
Rootstock	CoB	G	On	S	POW (merged mining with Bitcoin)	Ol	Bl	Randomly selected miner	Lc	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	Yes (with bitcoin)	Cy and DA	L	Ge
Steem	CoB	G	On	S	DPoS	De	Bl	Elected leader	Lc	Not supported						Cy	L	So
Stellar	DDB (chain of ledgers stored as key-value)	G	On	S	Stellar (FBA)	Ol	Ex	Validating nodes (using transactions sequence number)	Nf	NTc	Is	Stellar runtime	Dc	Java, JavaScript, Go	Yes (Atomic swap with other blockchains)	Cy	Hign	Po
Sia	CoB	G	On	S	PoW	An	Ex	Randomly elected miner	Lc	Not supported					No	DA	M	DSo
Stratis	CoB (Main chain and sidechains)	G	On	S	PoA or PoS	Ol	Ex	Randomly elected miner	Lc	Tc	Is	.NER runtime	Dc	.Net languages (e.g C#)	Yes (with its sidechains and with bitcoin)	Cy and DA	M	Ge
Nano	DAG (block-lattice)	G	On	S	Open Representative Voting (ORV) (based on DPoS)	Ol	Bl	Users (Sender and recipient)	Nf[i]	Not supported					No	Cy	L	Ge
Tezos	CoB	G	On	S	DPoS (Liquid PoS) and Emmy	De	Bl	Elected miner	Lc	Tc	Is (OB)	Tezos interpreter	Dc	Michelson	No	Cy and DA	M	Ge
Wanchain	CoB	G&R	On	S	PoS	De	Bl	Elected miner	Lc	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	Yes (Cross-Chain Communication Protocol)	Cy and DA	L	Io
Waves	CoB	G	On	S	Waves-NG (PoS based on Bitcoin-NG)	An	Bl	Elected miner (PoS)	Lc	NTc	Is (OB)	Waves runtime	Dc	Rideon	Yes (Atomic swap with other blockchains)	Cy and DA	L	Ge
Zilliqa	SCoB	G	On	S	PBFT and POW (Ethash)	De	Bl	Elected leader	Nf	Tc	Is (OB)	Zilliqa VM	Dc	Scilla	No	Cy and DA	M	Ge
Libra (Facebook)	DDB	G	On	W	LibraBFT	Ol	Ex	Elected leader	Nf	Tc	Is	Libra VM	Dc	Move	No	Cy	M	Po
Artis	CoB	G	On	S	HoneyBadgerBFT	Ol	Bl	Correct nodes	Nf	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	Yes (with Ethereum)	DA	M	Ge
VeChain	CoB	G	On	S	PoA	Ol	Bl	Elected leader (deterministic pseudo-random process)	Lc	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	No	Cy and DA	M	Ge
Polyon	CoB	G	On	S	PoS	An	Bl	Elected proposers	Nf	Tc	Is	EVM	Dc	Ethereum languages	No	Cy and DA	H	Ge

LEGEND :
CoB: Chain of blocks, SCoB: Sharded Chain of blocks, DDB: Distributed database, HDS: Hybrid data structure
G: Global, S: Strong, R: Restricted, W: Weak, Cy: Cryptocurrency, DA: DApps, Ol: Oligarchic, De: Democratic, An: Anarchic, Di: Dictatorship
Bl: Built-in, Ex: External, Ns: Not specified, Nsp: Not supported, Nf: No forks, Lc: Longest chain, Hb: Heaviest branch, Hc: Heaviest chain,
Op: Open, Is: Isolated, OB: Oracle-based, Dc: Deterministic, NDc: Non-Deterministic, H: High, M: Medium, L: Low, Ge: General.
Bo: Business-oriented, Po: Payment-oriented, So: Service-oriented, DSo: Decentralized storage-oriented, IoT: IoT-oriented, Io: Interoperability-oriented

Based on the project repository on https://github.com/dedis/cothority/tree/master/byzcoin
Elastos sidechains can theoretically use any consensus mechanism
Filecoin incentivizes miners with greater storage capacity
Different software implementations of the Filecoin protocol should be able to work together seamlessly.
https://qubic.iota.org
Based on the implementation available on https://github.com/dedis/student_18_byzcoin
Unlike the main chain, known as the Relay Chain, which has a standardized structure, parachains can be customized with a wide range of data models and representations.
Parachains Runtime logic provides isolation between parachains. If one chain experiences an issue or vulnerability, it doesn't automatically affect the others.
In Nano, a single network-wide DAG is shared by all participants and a balance-weighted voting system is used to handle conflicting transactions.

after 6 confirmation the probability of reverting the transaction decreases to 0.02428% [[1] page 8] assuming that an adversary controls less than 10% of the overall hash-power which is very unlikely. In contrast, multiple consensus mechanisms such as PBFT-based protocol (e.g Tendermint), Ripple, Stellar, DPoS-based protocols, some PoS protocols (e.g. Algorand), Thunderella's fast path and others achieve absolute finality if the validating majority is honest. Moreover, DLTs networks adopt measures to secure finality. For instance, EOS, a DPoS DLT, utilizes the concept of Last irreversible block (LIB) to improve finality and honest nodes wait for 330 confirmation blocks (less than 2 seconds) before considering a transaction irreversible.

When considering scalability, a crucial property for DLTs, the situation is continually improving. Recently proposed protocols like Avalanche [316] (3400 Transactions Per Second or tps) and Algorand [312] (around 1000 tps) outperform classical blockchain consensus protocols like PoW [317] (about 4 tps), while providing a comparable level of security and better finality. Permissioned protocols such as PBFT can achieve much higher throughput, for example, $15,000$ tps if the number of validating peers is under 16 [139]. However, the throughput falls to under 5000 tps when the number of validating peers is 64 [139]. HoneyBadgerBFT attains roughly equal performance of between $10,000$ and $15,000$ tps [139], a performance that comes with higher latency (around 6 minutes in a network of 104 nodes [139]). On the other hand, Tendermint can process thousands of transactions per second with very low latency (one-second block latency). However, similar to other PBFT-like protocols, the scalability of Tendermint in large-scale networks is questionable, as demonstrated in [318] with only a maximum of 64 nodes.

C. Zero-Knowledge Rollups and zkEVM: A Comparative Overview

The Zero-knowledge Virtual Machine (zkVM) is a emerging technology currently in the early stages of development, designed to enhance rollup capabilities through the utilization of zero-knowledge proofs. At its core, zkVM introduces innovative features, including the execution of smart contracts within the rollup in a manner that prioritizes both security and privacy. Unlike traditional transaction verification methods, zkVM goes beyond individual transactions and validates the entire computation within smart contracts. This approach significantly reduces the volume of on-chain data, contributing to improved efficiency.

One of the primary technical aspects of zkVM lies in its ability to execute smart contracts securely and privately. By leveraging advanced cryptographic techniques associated with zero-knowledge proofs, zkVM can conceal sensitive details of smart contract execution, including transaction amounts and asset types. This heightened level of privacy protection aligns with the growing demand for secure and confidential transaction processing on blockchain networks.

The potential technical advantages of zkVM are noteworthy:

- Increased Scalability: zkVM's offloading of smart contract execution to the rollup has the potential to significantly amplify transaction throughput, addressing scalability concerns associated with traditional on-chain execution.
- Enhanced Privacy Mechanisms: Leveraging zero-knowledge proofs, zkVM ensures robust privacy guarantees for smart contract execution, mitigating concerns related to data exposure.
- Facilitation of Complex Use Cases: zkVM's ability to validate entire computations within smart contracts opens doors to new use cases that require not only heightened privacy but also on-chain verification of intricate and computationally intensive processes.

However, it is important to underscore the challenges associated with zkVM's early developmental phase:

- Technical Complexity: Implementing and verifying zkVM proofs necessitates the application of advanced cryptographic techniques, introducing a level of technical complexity that requires careful consideration.
- Limited Ecosystem: The current support for zkVM is constrained, with ongoing efforts in the development of necessary tools and applications. The ecosystem is evolving, and broader adoption is contingent on continued maturation.

As zkVM progresses through its developmental stages, these technical considerations will likely shape its trajectory, determining its viability and potential impact within the broader blockchain landscape. In table XIII we provide a comprehensive comparison of some well-known zk protocols within the Ethereum ecosystem, highlighting their distinctive features, applications, and trade-offs. zkSync and zkPorter both utilize zk-SNARKs, emphasizing scalability and privacy, with efficient proof generation and strong privacy guarantees. However, they exhibit limitations in smart contract functionality when compared to Optimism. Hermez and zkTube, employing PLONK, share similar focuses on scalability and privacy, boasting fast verification and versatile proof systems. Nevertheless, akin to zkSync and zkPorter, they present constraints in smart contract functionality compared to Optimism. StarkWare, relying on STARK, prioritizes scalability and security, delivering highly secure proofs and fast verification processes. However, it places less emphasis on privacy compared to zk-SNARKs. zk-rollups on Arbitrum leverage various zk-SNARKs and PLONK implementations to address scalability and privacy concerns, capitalizing on existing Arbitrum infrastructure. Despite the flexibility, users face the challenge of navigating through multiple implementations, introducing complexity in selecting the most suitable one. Aztec Protocol, utilizing zk-SNARKs, stands out for its focus on privacy-preserving DeFi, ensuring strong privacy for financial transactions. Nevertheless, its smart contract functionality is somewhat limited for non-DeFi applications. Each protocol exhibits a unique set of advantages and disadvantages, catering to specific use cases and user priorities within the

TABLE XII
A COMPARISON OF SELECTED DLTS

DLT	Scope	Decentralization Level	Taxon
Æternity	Public	Decentralized	Blockchain
Algorand	Public	Decentralized	Blockchain
Ardor	Public	Decentralized	Blockchain
Bigchaindb 2.0	Private	Semi-centralized	Blockchain-like
Bitcoin	Public	Decentralized	Blockchain
BitShares	Public	Decentralized	Blockchain
Byzcoin	Public	Decentralized	Blockchain
Cardano	Public	Decentralized	Blockchain
Corda (R3)	Private or consortium	Semi-Decentralized	Blockchain-like
Cosmos	Cosmos hub is public, zones can be public or private	Decentralized	Blockchain
Decred	Public	Decentralized	Blockchain
Elrond	Public	Decentralized	Blockchain
EOS	Public	Semi-Decentralized	Blockchain-like
Ethereum 1.0 and 2.0	Public	Decentralized	Blockchain
Ethereum Enterprise	Private or consortium	Decentralized	Blockchain
Exonum Enterprise	Public or private	Semi-Decentralized	Blockchain-like
Elastos	Public	Decentralized	Blockchain
Filecoin	Public	Decentralized	Blockchain
Hashgraph	Public	Centralized	Blockchain-like
Hyperledger fabric	Private or consortium	Semi-Decentralized	Blockchain-like
IOTA	Public	Semi-Decentralized	Blockchain-like
Lisk	Public	Decentralized	Blockchain
Multichain	Private or consortium	Semi-Decentralized	Blockchain
NEO	Public or private	Semi-Decentralized	Blockchain
Omniledger	Public	Decentralized	Blockchain
Parity substrate	Private or consortium	Semi-Decentralized	Blockchain
Polkadot (Relay chain)	Public	Decentralized	Blockchain
Quorum	Private	Semi-Decentralized	Blockchain
Qtum	Public	Decentralized	Blockchain
Ripple	Public	Centralized	Blockchain-like
Rootstock	Public	Decentralized	Blockchain
Steem	Public	Decentralized	Blockchain
Stellar	Public	Centralized	Blockchain-like
Sia	Public	Decentralized	Blockchain
Stratis	Main chain is public, seidechains are private	Main chain is decentralized, the BAAS is centralized	Blockchain-like
Nano	Public	Semi-decentralized	Blockchain-like
Tezos	Public	Decentralized	Blockchain
Wanchain	Public	Decentralized	Blockchain
Waves	Public, private or permissioned	Decentralized	Blockchain
Zilliqa	Public	Decentralized	Blockchain
Libra (Facebook)	Public	Centralized	Blockchain-like
Artis	Public	Decentralized	Blockchain
VeChain	Public	Semi-Decentralized	Blockchain
Red Belly	Public	Semi-Decentralized	Blockchain

Ethereum ecosystem.

D. DApps Attractiveness: A Comparative Overview

In evaluating the appeal of specific blockchain network solutions for DApp builders, the Total Value Locked (TVL) metric serves as a crucial benchmark within the cryptocurrency sector. TVL quantifies the total U.S. dollar value of digital assets locked or staked through decentralized finance (DeFi) platforms or decentralized applications (DApps). Figure 21 illustrates that Ethereum stands as the predominant network for DApp development, controlling more than half of the total value locked across blockchains. Unsurprisingly, alternative Layer-1 (L1) chains, particularly those compatible with Ethereum Virtual Machine (EVM), have proven attractive to developers. The top five contenders—Arbitrum, Optimism, Base, Polygon, and Era— collectively controlling about 10% of the TVL. This phenomenon can be attributed to the appeal of EVM-compatible L1 chains like Polygon and Avalanche, which leverage their compatibility to entice Ethereum users seeking improved transaction speeds, reduced fees, and potentially higher returns. This establishes a mutually beneficial relationship between the L1 chain and its user base. Conversely, non-EVM projects such as Solana, Cardano, and Bitcoin exhibit lower TVL percentages, with Solana at 2%, Cardano at 1%, and Bitcoin at less than 1%. The divergence in TVL may be explained by the ease of DApp development on EVM-compatible platforms compared to their non-compatible counterparts.

IX. LESSONS LEARNED AND TUTORIALS FOR DESIGNING NEW DLT SOLUTIONS

While preparing this survey, we found multiple challenges that directly impact the performance of DLT. Here, we summarize the most crucial challenges across the four layers.

TABLE XIII
COMPARISON OF ZK PROTOCOLS IN THE ETHEREUM ECOSYSTEM

Protocol	ZK Protocol Used	Focus	Advantages	Disadvantages
zkSync	zk-SNARKs	Scalability, privacy	Highly efficient proofs, strong privacy guarantees	Limited smart contract functionality compared to Optimism
Hermez	PLONK	Scalability, privacy	Fast verification, versatile proof system	Limited smart contract functionality compared to Optimism
StarkWare	STARK	Scalability, security	Highly secure proofs, fast verification	Less focus on privacy compared to zk-SNARKs
zkPorter	zk-SNARKs	Scalability, privacy	Efficient proof generation, strong privacy guarantees	Less mature than other protocols
zkTube	PLONK	Scalability, privacy	Fast verification, versatile proof system	Limited smart contract functionality compared to Optimism
zk-rollups on Arbitrum	Various zk-SNARKs and PLONK implementations	Scalability, privacy	Leverages existing Arbitrum infrastructure	Multiple implementations, complexity in choosing the right one
Aztec Protocol	zk-SNARKs	Privacy-preserving DeFi	Strong privacy for financial transactions	Limited smart contract functionality for non-DeFi applications

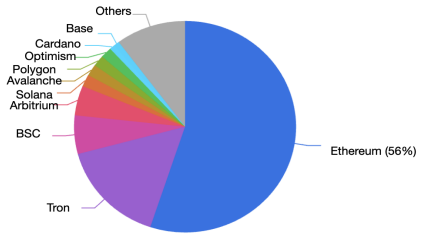

Fig. 21. TVL in the most prominent DApp-oriented blockchains (data source: DeFiLama)

a) Managing Blockchain Size: When exploring different blockchain projects, a common problem that surfaces is what we call "blockchain bloat". Blockchain bloat is a phenomenon where the size of a blockchain becomes excessively large, posing various challenges. It results from factors like large block sizes, increased transaction volume, data redundancy, and the immutability of blockchain data. This condition has significant consequences, including increased storage requirements, scalability limitations, and concerns about decentralization. Several solutions are being explored to address blockchain bloat, such as state pruning, sidechains, sharding, and Layer-2 solutions. Tackling this challenge is crucial for ensuring the sustainability and scalability of blockchain technology in the long run.

Ensuring the security of smart contracts presents a significant challenge in the DLT domain, especially in public blockchains with associated financial risks. Several strategies have been recommended to enhance smart contract security, encompassing the implementation of code analyzers, the adoption of secure smart contract libraries like OpenZeppelin [319], formal verification techniques, and the establishment of coding best practices. Due to the lack of a clear upgrading process for vulnerable smart contracts in the majority of DLTs, designers often focus on providing secure architectural and design approaches, upgradability patterns, and detailed best practice guidelines. These measures aim to assist developers in writing secure smart contracts, avoiding well-known vulnerabilities, and steering clear of security pitfalls. Furthermore, for risk mitigation, formal verification proves to be an efficient means of addressing security concerns and ensuring better smart contracting. Interestingly, bug bounty programs have proven to be an effective and cost-efficient way to enhance smart contract security.

b) Upgrading Consensus Mechanisms: There is a plethora of consensus protocols in the literature, with continuous research efforts aimed at producing new ones. However, when designing a new DLT, one should be aware of the upgradability pitfall. For various reasons, such as the emergence of a high-performing protocol or a new improvement, a project may need to shift or upgrade its underlying consensus protocol. In public blockchains, the protocol upgrade can come with its risks. For instance, Ethereum's plan to shift from PoW to PoS will have financial implications for its current miners, which may lead to opposition and a potential network split. Thus, it is crucial to choose the most suitable mechanism at the beginning of the project. Moreover, the decision to switch from one protocol to another must be fault-tolerant and should be secured through built-in mechanisms (e.g., Ethereum difficulty bomb [320]).

c) Accessible applications: The complexity of the actual end user experience is a common observation among most DLTs. This is explained by the fact that the inherent design does not consider to provide a good user experience but rather focuses on the internal machinery. To remedy that, the current solutions (e.g. wallets plugins) are not only non-intuitive for a new user but they are also challenging to use and maintain. A DLT designer should consider providing gateways, APIs and SDKs, and other enabling solutions that will allow developers to design more user friendly DApps and allow a seamless interaction between both product and end-user. An accessible blockchain design should enable average users to interact with the hosted DApps without prior knowledge or the need to synchronize the whole ledger, in order to lower the entry level of blockchain use.

X. BLOCKCHAIN CHALLENGES AND FUTURE RESEARCH DIRECTIONS

DLTs (blockchain and blockchain-like), despite their immense potential, face significant challenges that need to be addressed for their wider adoption and mainstream success. Here, we present a non-exhaustive list of the key challenges:

a) Limited Scalability: This problem is primarily due to the challenge of finding a perfect balance among decentralization, security, and scalability. A well-known paradigm, called the scalability trilemma [321], posits that it is impossible to build a system with all the aforementioned characteristics. However, multiple approaches to scaling distributed protocols are presented, such as off-chain processing, sharding (dividing a whole blockchain into multiple shards), and overhead reduction (Segwit, MAST, etc.). Nevertheless, these solutions themselves raise new challenges and security concerns. For example, the work in [322] highlights the remaining challenges

of sharding mechanisms, such as intra-consensus safety, cross-shared communication, and more. It is worth noting that although the theoretical propositions are not fully sharded, the blockchain is still operational.

b) Formal Verification: Recent costly bugs in smart contracts have highlighted the critical role of formal verification in ensuring the correctness and security of these programs. While significant research has explored formal verification techniques for smart contracts, achieving promising results [323]-[324], existing approaches exhibit limitations in handling complex contracts. Current methods often struggle with contracts featuring intricate control flow, extensive state transitions, or interactions with external oracles, hindering their practical applicability in real-world scenarios. Additionally, the computational resources required for verifying complex contracts can be substantial, further limiting their scalability.

c) Data immutability and integrity in DLTs: In private or consortium DLTs with permissioned access, ensuring a high level of data immutability is challenging compared to the assurance provided by public blockchains. This difficulty raises concerns about the suitability of DLTs in such environments where immutability is a crucial aspect. While there are some partial solutions like Exonum (which anchors data on the Bitcoin blockchain) or Kadena, the issue is far from being fully resolved.

d) Data availability problem: The efforts of scaling blockchain face significant challenge: guaranteeing data availability. In simpler terms, all nodes on the network, not just the block producers, need to be confident that the data in each new block is complete and hasn't been tampered with. Traditionally, verifying this meant downloading the entire block, which is inefficient and impractical for large blockchains. To address this hurdle, most scaling projects utilize data availability proofs. These mechanisms allow nodes to confirm, with near certainty, that all block data is present, even if they only download a tiny fraction of the block itself.

e) Decentralized governance mechanisms: Implementing effective decentralized governance poses a multifaceted challenge due to the absence of a central authority. Decentralized Autonomous Organizations (DAOs) initially appeared promising for transparent and decentralized management, but practical implementations revealed crucial limitations. One significant drawback is the lack of robust mechanisms for establishing user reputations within the system, affecting trust assessment and governance quality. Additionally, the anonymous nature of DAOs renders them vulnerable to Sibyl attacks, allowing malicious actors to manipulate voting processes and undermine governance integrity. The regulatory landscape surrounding DAOs remains unclear in many jurisdictions, creating uncertainty and hindering widespread adoption. Looking ahead, exploring alternative governance models beyond token-based voting, implementing decentralized dispute resolution mechanisms, and assessing the broader societal and economic implications of decentralized governance practices are essential considerations for overcoming these challenges.

f) Quantum resistance: Many cryptographic algorithms used by different DLTs are not quantum-resistant. For instance, the use of Schnorr or ECDSA (used by Bitcoin and Ethereum and others) for signing transactions is under threat. Aware of this problem, a few researchers have attempted to advance efficient solutions. Notably, [325] reported the experimental realization of a quantum-safe blockchain. However, as research remains limited, more effort should be placed on the adoption of alternative cryptographic signature schemes (e.g. XMSS, hash ladder signatures, and SPHINCS) to replace the classical schemes and build a secure, resilient post-quantum DLT protocol.

g) Smart Contracts for IoT: Despite the proposition of a few projects dedicated to IoT (e.g. IOTA, Vechain [326]), there is no platform providing a smart contracts environment tailored to the special IoT requirements (e.g. lightweight execution runtime.). Most solutions propose hybrid DLTs where the IoT objects rely on external resources to run smart contracts.

h) Useful Proof-of-work: Proof-of-work (PoW) is one of the secure consensus mechanisms, but it is severely criticized for being wasteful. [327] and [328] proposed to build new useful Proof-of-Work protocols solving useful calculation.

i) Sustainable Liquidity pools: One of the biggest problems that DeFi protocols face is the difficulty of sustainably attracting long-lasting liquidity. Most of these protocols distribute an important proportion of their native tokens into the liquidity mining incentives. This usually attracts investors and accelerate the growth of DeFi projects quickly. However, the vast majority of liquidity is unloyal and moves to new projects that offer better financial incentives which creates a huge selling pressure for the native token and thus drops its value. To mitigate this financial risk new DeFI projects, considered as being part of the next generation DeFi (DeFi 2.0), try to attract long-lasting liquidity without depending on the never-ending cycle of subsidising the users with liquidity mining rewards. To reach that goal more effort should be placed to design new protocols creating sustainable liquidity through a decentralized market-making and enabling a quick bootstrapping phase and attracting initial capital to a new chain or L2.

j) Upgrading Runtimes Without Forks: While hard forking is a prevalent approach for upgrading public blockchains, it proves to be inefficient and error-prone in large-scale networks. The challenges stem from the considerable offline and online coordination needed to prevent network splits. Emerging solutions, such as those seen in projects like Polkadot, are investigating alternative methods. One promising avenue involves leveraging portable technologies like Wasm on-chain, allowing nodes to autonomously adopt upgraded logic at a predefined block height, eliminating the need for external intervention.

k) Low-Latency Byzantine Agreement Protocols Using RDMA: Remote Direct Memory Access (RDMA) is a technology that enables networked computers to exchange data in shared memory to improve throughput and performance. The shared memory model has been widely researched for key/value stores, databases and distributed file systems. How-

ever, the leverage of RDMA for consensus mechanisms especially for BFT has been neglected. Further research is needed to integrate blockchain with the shared memory and RDMA technologies and to propose BFT protocols for low-latency and RDMA-enabled consensus algorithms.

l) Emerging Trends in Zero-Knowledge Proof (ZKP) Protocols: Zero-Knowledge Proof (ZKP) protocols have evolved significantly in recent years, playing a vital role in enhancing privacy and security across various domains.

Non-Interactive Zero-Knowledge Proofs (NIZKPs) [329]: Traditionally, ZKPs involve interaction between the prover and verifier. However, there is a growing interest in Non-Interactive Zero-Knowledge Proofs (NIZKPs), where the prover can generate a proof that can be verified without interaction. Recent research has focused on constructing NIZKPs based on various cryptographic assumptions, including the use of bilinear pairings, hash functions, and algebraic structures. NIZKPs are not only theoretically intriguing but also practical for decentralized systems where participants may not be online simultaneously.

Lattice-Based Zero-Knowledge Proofs: Lattice-based cryptography, considered a post-quantum alternative, is being explored in the context of ZKPs for its resistance to quantum attacks. The challenge lies in designing efficient lattice-based ZKPs without sacrificing performance. Ongoing research aims to strike a balance between security and practicality in these constructions. Integration with Homomorphic Encryption: Combining ZKPs with homomorphic encryption allows computations to be performed on encrypted data without decryption, providing an additional layer of privacy and security. Research in this area explores ways to integrate ZKPs with homomorphic encryption schemes, enabling secure and private computations on encrypted data while proving the validity of the computations. This integration has potential applications in secure multi-party computation and privacy-preserving data analytics. Succinct Arguments of Knowledge: Efficiency remains a key concern in ZKP protocols, and succinct arguments of knowledge aim to reduce the size of proofs and improve verification speed without compromising security. Techniques like zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) have demonstrated significant success in achieving succinctness. Ongoing research focuses on refining zk-SNARK constructions, exploring new mathematical frameworks, and addressing limitations to further enhance their efficiency and applicability.

XI. CONCLUSION

This document introduces an extensive examination of contemporary DLTs through a thorough and multi-layered state-of-the-art analysis. We introduce a conceptual and referential framework aimed at enhancing comprehension and categorization of DLTs. Our specific focus is on defining clear boundaries and reducing ambiguity between blockchain and other DLT systems, referred to as "blockchain-like". Moreover, we employ the provided framework as a guide to examine the different design-choices adopted by various DLTs across four layers: data structure, execution, consensus, and application layers. The application of our reference framework results in the creation of a novel taxonomy for classifying DLTs, broadly categorizing them into two groups: blockchain and blockchain-like systems. Additionally, we conduct a qualitative and comparative analysis of numerous existing DLTs, with a dedicated examination of the most significant and recent consensus mechanisms. This survey aims to provide valuable insights for designers of new DLT systems and consensus mechanisms, along with assisting decision-makers. It offers a clear and detailed overview of recent contributions at each layer, highlighting trade-offs and limitations resulting from different design choices. Furthermore, the analysis can aid in advocating the selection of a DLT solution for building decentralized systems. Finally, we identify several important open issues that merit attention in future research.

In conclusion, our work contributes to advancing the understanding of DLTs, their classifications, and the nuances among various systems. We hope this survey serves as a valuable resource for both academia and industry, guiding the development of robust and efficient decentralized technologies.

REFERENCES

[1] S. Nakamoto, "Bitcoin : A Peer-to-Peer Electronic Cash System," pp. 1–9, 2008.

[2] L. Team, "Litecoin." [Online]. Available: https://litecoin.org/

[3] S. King and S. Nadal, "Peercoin–secure & sustainable cryptocoin," *Aug-2012 [Online]. Available: https://peercoin. net/whitepaper ()*, 2012.

[4] S. Popov, "The Tangle," Tech. Rep., 2017.

[5] H. Foundation, "Hyperlegder Project." [Online]. Available: https://www.hyperledger.org/

[6] Hedra, "Hedera Hashgraph." [Online]. Available: https://www.hedera.com/

[7] ITU, "Focus Group on Application of Distributed Ledger Technology." [Online]. Available: https://www.itu.int/en/ITU-T/focusgroups/dlt/Pages/default.aspx

[8] E. N. Dawson, A. Taylor, and Y. Chen, "ISO/TC 307 Blockchain and distributed ledger technologies." [Online]. Available: https://www.iso.org/committee/6266604.html

[9] ISO/TR, "ISO/TR 23455:2019 Blockchain and distributed ledger technologies — Overview of and interactions between smart contracts in blockchain and distributed ledger technology systems."

[10] I. S. Association, "IEEE blockchain standards." [Online]. Available: https://blockchain.ieee.org/standards

[11] C. Wittbrodt and S. Hares, "Essential Tools for the OSI Internet."

[12] B. Bellaj, A. Ouaddah, E. Bertin, N. Crespi, and A. Mezrioui, "Sok: a comprehensive survey on distributed ledger technologies," in *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2022, pp. 1–16.

[13] ——, "Untangling the overlap between blockchain and dlts," in *Science and Information Conference*. Springer, 2022, pp. 483–505.

[14] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys*, 2015. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7423672/

[15] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A Taxonomy of Blockchain-Based Systems for Architecture Design," in *Proceedings - 2017 IEEE International Conference on Software Architecture, ICSA 2017*. Institute of Electrical and Electronics Engineers Inc., 5 2017, pp. 243–252.

[16] B. J. Butijn, D. A. Tamburri, and W. J. V. D. Heuvel, "Blockchains: A Systematic Multivocal Literature Review," 6 2020. [Online]. Available: https://doi.org/xxx

[17] M. C. Ballandies, M. M. Dapp, and E. Pournaras, "Decrypting Distributed Ledger Design – Taxonomy, Classification and Blockchain Community Evaluation," 10 2018. [Online]. Available: http://arxiv.org/abs/1811.03419

[18] P. Tasca and C. J. Tessone, "A Taxonomy of Blockchain Technologies: Principles of Identification and Classification," *Ledger*, vol. 4, 2 2019.

[19] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9591. Springer Verlag, 2016, pp. 112–125.

[20] C. Cachin and M. Vukolić, "Blockchain consensus protocols in the wild," in *Leibniz International Proceedings in Informatics, LIPIcs*, vol. 91. Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 10 2017.

[21] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, pp. 1085–1100.

[22] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Consensus in the Age of Blockchains," 11 2017. [Online]. Available: http://arxiv.org/abs/1711.03936

[23] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks," *IEEE Access*, vol. 7, pp. 22 328–22 370, 2019.

[24] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A Survey of Distributed Consensus Protocols for Blockchain Networks," Tech. Rep.

[25] M. S. Ferdous, M. J. M. C. arXiv, and u. 2020, "Blockchain Consensus Algorithms: A Survey," *ui.adsabs.harvard.edu*. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2020arXiv200107091S/abstract

[26] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses," 6 2020.

[27] Z. Zheng, S. Xie, H. N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 4 2020.

[28] A. S. Almasoud, F. K. Hussain, and O. K. Hussain, "Smart contracts for blockchain-based reputation systems: A systematic literature review," *Journal of Network and Computer Applications*, vol. 170, p. 102814, 11 2020.

[29] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. H. Nyang, and D. Mohaisen, "Exploring the Attack Surface of Blockchain: A Comprehensive Survey," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 3, pp. 1977–2008, 7 2020.

[30] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Generation Computer Systems*, vol. 107, pp. 841–853, 6 2020.

[31] M. R. Islam and M. M. Rashid, "A survey on blockchain security and its impact analysis," in *2023 9th International Conference on Computer and Communication Engineering (ICCCE)*. IEEE, 2023, pp. 317–321.

[32] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to Scalability of Blockchain: a Survey," *IEEE Access*, vol. 8, pp. 16 440–16 455, 2020.

[33] A. Hafid, A. S. Hafid, and M. Samih, "Scaling Blockchains: A Comprehensive Survey," *IEEE Access*, vol. 8, pp. 125 244–125 262, 2020.

[34] H. N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for Internet of Things: A Survey," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8076–8094, 10 2019.

[35] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of Blockchains in the Internet of Things: A Comprehensive Survey," pp. 1676–1717, 4 2019.

[36] X. Wang, X. Zha, W. Ni, R. P. Liu, Y. J. Guo, X. Niu, and K. Zheng, "Survey on blockchain for Internet of Things," pp. 10–29, 2 2019.

[37] L. Lao, Z. Li, S. Hou, B. Xiao, S. Guo, and Y. Yang, "A survey of IoT applications in blockchain systems: Architecture, consensus, and traffic modeling," 2 2020.

[38] W. Chen, Z. Xu, S. Shi, Y. Zhao, and J. Zhao, "A Survey of Blockchain Applications in Different Domains," in *ICBTA 2018: Proceedings of the 2018 International Conference on Blockchain Technology and Application*. Association for Computing Machinery, 12 2018, pp. 17–21. [Online]. Available: https://doi.org/10.1145/3301403.3301407

[39] J. Xie, H. Tang, T. Huang, F. R. Yu, R. Xie, J. Liu, and Y. Liu, "A Survey of Blockchain Technology Applied to Smart Cities: Research Issues and Challenges," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 3, pp. 2794–2830, 7 2019.

[40] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for 5G and beyond networks: A state of the art survey," p. 102693, 9 2020.

[41] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated Blockchain and Edge Computing Systems: A Survey, Some Research Issues and Challenges," pp. 1508–1532, 4 2019.

[42] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. Leung, "Blockchain and Machine Learning for Communications and Networking Systems," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 2, pp. 1392–1431, 4 2020.

[43] K. Salah, M. H. U. Rehman, N. Nizamuddin, and A. Al-Fuqaha, "Blockchain for AI: Review and open research challenges," *IEEE Access*, vol. 7, pp. 10 127–10 149, 2019.

[44] J. Kolb, M. Abdelbaky, R. H. Katz, and D. E. Culler, "Core Concepts, Challenges, and Future Directions in Blockchain: A Centralized Tutorial," *ACM Computing Surveys*, vol. 53, no. 1, 2 2020. [Online]. Available: https://doi.org/10.1145/3366370

[45] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A Survey on Blockchain Interoperability: Past, Present, and Future Trends," *2020. A Survey on Blockchain Interoperability: Past, Present, and Future Trends*, vol. 1, no. 1, p. 60, 2020. [Online]. Available: https://arxiv.org/abs/2005.14282

[46] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," pp. 117 134–117 151, 2019.

[47] M. Belotti, N. Bozic, G. Pujolle, and S. Secci, "A Vademecum on Blockchain Technologies: When, Which and How," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 7 2019.

[48] S. Bouraga, "A taxonomy of blockchain consensus protocols: A survey and classification framework," *Expert Systems with Applications*, vol. 168, p. 114384, 2021.

[49] A. R. Sai, J. Buckley, B. Fitzgerald, and A. Le Gear, "Taxonomy of centralization in public blockchain systems: A systematic literature review," *Information Processing & Management*, vol. 58, no. 4, p. 102584, 2021.

[50] P. Tasca and C. J. Tessone, "Taxonomy of blockchain technologies. principles of identification and classification," *arXiv preprint arXiv:1708.04872*, 2017.

[51] J. Werner, S. Frost, and R. Zarnekow, "Towards a taxonomy for governance mechanisms of blockchain-based platforms," 2020.

[52] F. E. Alzhrani, K. A. Saeedi, and L. Zhao, "A taxonomy for characterizing blockchain systems," *IEEE Access*, vol. 10, pp. 110 568–110 589, 2022.

[53] J. Biolchini, P. Gomes Mian, A. Candida Cruz Natali, and G. Horta Travassos, "Systematic Review in Software Engineering," 2005.

[54] B. Kitchenham, "Procedures for Performing Systematic Reviews," 2004.

[55] V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering," *Information and Software Technology*, vol. 106, pp. 101–121, 7 2017. [Online]. Available: https://arxiv.org/abs/1707.02553v4

[56] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," in *Conference on the Theory and Application of Cryptography*. Springer, 1990, pp. 437–455.

[57] D. Chaum, "Blind signatures for untraceable payments," in *Advances in cryptology*. Springer, 1983, pp. 199–203.

[58] D. L. Chaum, "World's First Electronic Cash Payment Over Computer Networks," *Digicash, press release*, vol. 26, 1994.

[59] A. Back, "Hashcash-a denial of service counter-measure," 2002.

[60] W. Dai, "B-Money-an anonymous, distributed electronic cash system," 1998.

[61] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," 2014.

[62] R3, "R3 strory." [Online]. Available: https://www.r3.com/history/

[63] M. Hearn, "Corda: A distributed ledger," *Corda Technical White Paper*, vol. 2016, 2016.

[64] V. Dhillon, D. Metcalf, M. Hooper, V. Dhillon, D. Metcalf, and M. Hooper, "The Hyperledger Project," in *Blockchain Enabled Applications*. Apress, 2017, pp. 139–149.

[65] V. Buterin, "A next-generation smart contract and decentralized application platform," *Etherum*, no. January, pp. 1–36, 2014. [Online]. Available: http://buyxpr.com/build/pdfs/EthereumWhitePaper.pdf

[66] X. He, Y. Cui, and Y. Jiang, "An improved gossip algorithm based on semi-distributed blockchain network," in *Proceedings - 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2019*. Institute of Electrical and Electronics Engineers Inc., 10 2019, pp. 24–27.

[67] N. Berendea, H. Mercier, E. Onica, and E. Rivière, "Fair and Efficient Gossip in Hyperledger Fabric," 4 2020. [Online]. Available: http://arxiv.org/abs/2004.07060

[68] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *IEEE P2P 2013 Proceedings*. IEEE, 2013, pp. 1–10.

[69] S. Kyun Kim, Z. Ma, S. Murali, J. Mason, A. Miller, and M. Bailey, "Measuring Ethereum Network Peers," p. 14, 2018. [Online]. Available: https://doi.org/10.1145/3278532.3278542

[70] W. Pugh, "Skip lists: a probabilistic alternative to balanced trees," *Communications of the ACM*, vol. 33, no. 6, pp. 668–676, 1990.

[71] S. Delgado-Segura, C. Pérez-Solà, G. Navarro-Arribas, and J. Herrera-Joancomartí, "Analysis of the Bitcoin UTXO set," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10958 LNCS. Springer Verlag, 2019, pp. 78–91.

[72] I. Eyal, A. Gencer, and E. Sirer, "Bitcoin-ng: A scalable blockchain protocol," *13th USENIX Symposium*, 2016. [Online]. Available: https://www.usenix.org/system/files/conference/nsdi16/nsdi16-paper-eyal.pdf

[73] P. Jovanovic, "ByzCoin: Securely Scaling Blockchains," *Hacking, Distributed, August*, 2016.

[74] K. Nikitin, E. Kokoris-Kogias, P. Jovanovic, N. Gailly, L. Gasser, I. Khoffi, J. Cappos, and B. Ford, "{CHAINIAC}: Proactive software-update transparency via collectively signed skipchains and verified builds," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1271–1287.

[75] A. Churyumov, "Byteball: A decentralized system for storage and transfer of value," *URL https://byteball. org/Byteball. pdf*, 2016.

[76] S. L. W. paper and u. 2015, "DagCoin: a cryptocurrency without blocks."

[77] A. D. Dwivedi, G. Srivastava, R. Singh, and A. Dhar Dwivedi, "PHANTOM Protocol as the New Crypto-Democracy Big Data and Social Media Influence View project Fuzzy based Decision making system View project PHANTOM protocol as the new Crypto-democracy," *Springer*, vol. 11127 LNCS, pp. 499–509, 2018. [Online]. Available: https://www.researchgate.net/publication/327155886

[78] A. Kiayias and G. Panagiotakos, "On trees, chains and fast transactions in the blockchain," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11368 LNCS. Springer Verlag, 2019, pp. 327–351.

[79] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 507–527.

[80] L. Baird, "THE SWIRLDS HASHGRAPH CONSENSUS ALGORITHM: FAIR, FAST, BYZANTINE FAULT TOLERANCE," Tech. Rep., 2016.

[81] C. LeMahieu, "RaiBlocks: A feeless distributed cryptocurrency network," *URL https://raiblocks. net/media/RaiBlocks_Whitepaper__English. pdf*, 2017.

[82] T.-Y. Chen, W.-N. Huang, P.-C. Kuo, H. Chung, and T.-W. Chao, "DEXON: A Highly Scalable, Decentralized DAG-Based Consensus Algorithm," *arXiv preprint arXiv:1811.07525*, 2018.

[83] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: A Fast and Scalable Cryptocurrency Protocol." *IACR Cryptology ePrint Archive*, vol. 2016, p. 1159, 2016.

[84] Y. Sompolinsky and A. Zohar, "Phantom, Ghostdag," 2020.

[85] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8975. Springer Verlag, 2015, pp. 528–547.

[86] C. R3, "Corda Documentaion." [Online]. Available: https://docs.corda.net/docs/corda-os/4.4/node-database-tables.html

[87] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, and Y. Manevich, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, 2018, pp. 1–15.

[88] I. Grigg, "Eos-an introduction," *White paper. https://whitepaperdatabase. com/eos-whitepaper*, 2017.

[89] T. McConaghy, R. Marques, A. Müller, D. De Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare, and A. Granzotto, "Bigchaindb: a scalable blockchain database," *white paper, BigChainDB*, 2016.

[90] D. Schwartz, N. Youngs, and A. Britto, "The ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, vol. 5, no. 8, 2014.

[91] S. Thomas and E. Schwartz, "A protocol for interledger payments," *URL https://interledger. org/interledger. pdf*, 2015.

[92] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," *White Paper*, 2016.

[93] S. Brakeville and P. Bhargav, "Blockchain basics: Glossary and use cases," 2016. [Online]. Available: https://developer.ibm.com/technologies/blockchain/tutorials/cl-blockchain-basics-glossary-bluemix-trs/

[94] Jpmorganchase, "GitHub - jpmorganchase/constellation: Peer-to-peer encrypted message exchange." [Online]. Available: https://github.com/jpmorganchase/constellation

[95] Tessera, "GitHub - jpmorganchase/tessera: Tessera - Enterprise Implementation of Quorum's transaction manager." [Online]. Available: https://github.com/jpmorganchase/tessera

[96] Segwit, "bips/bip-0009.mediawiki at master · bitcoin/bips · GitHub." [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0009.mediawiki

[97] E. Lombrozo, J. Lau, and P. Wuille, "Segregated Witness (Consensus layer)," 2015. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki

[98] B. r. Network, "The Bitcoin Relay Network." [Online]. Available: https://bitcoinrelaynetwork.org/

[99] S. BASU, I. EYAL, and E. G. SIRER, "FLACON," p. 2016. [Online]. Available: https://www.falcon-net.org/

[100] U. Klarman, S. Basu, A. Kuzmanovic, and E. Gün Sirer, "bloXroute: A Scalable Trustless Blockchain Distribution Network WHITEPAPER," Tech. Rep. [Online]. Available: https://medium.com/@bloxroutelabs/bloxroute-business-model-nov-2019-bb1b2f6d0bde,

[101] A. Cullen, P. Ferraro, C. King, and R. Shorten, "Distributed Ledger Technology for IoT: Parasite Chain Attacks," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7112–7122, 3 2019. [Online]. Available: http://arxiv.org/abs/1904.00996http://dx.doi.org/10.1109/JIOT.2020.2983401

[102] Z. Team, "The ZILLIQA technical whitepaper," *Retrieved September*, vol. 16, p. 2019, 2017.

[103] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[104] b. l. Est., "THE RAIDEN NETWORK." [Online]. Available: https://github.com/raiden-network/raiden

[105] IBM, "Blockcchain- Entreprise blockchain solutions & Services." [Online]. Available: https://www.ibm.com/blockchain

[106] Microsoft, "Azure Blockchain Service." [Online]. Available: https://azure.microsoft.com/en-us/services/blockchain-service/

[107] Y. Chen, J. Gu, S. Chen, S. Huang, and X. S. Wang, "A Full-Spectrum Blockchain-as-a-Service for Business Collaboration," in *2019 IEEE International Conference on Web Services (ICWS)*. IEEE, 2019, pp. 219–223.

[108] P. Koshy, D. Koshy, and P. McDaniel, "An analysis of anonymity in bitcoin using p2p network traffic," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 469–485.

[109] I. D. Mastan and S. Paul, "A new approach to deanonymization of unreachable bitcoin nodes," in *International Conference on Cryptology and Network Security*. Springer, 2017, pp. 277–298.

[110] Deterministic wallet, "Deterministic wallet." [Online]. Available: https://en.bitcoin.it/wiki/Deterministic_wallet

[111] S. Noether, "Ring SIgnature Confidential Transactions for Monero." *IACR Cryptology ePrint Archive*, vol. 2015, p. 1098, 2015.

[112] Zcash, "Zcash is a privacy-protecting, digital currency built on strong science." [Online]. Available: https://z.cash/

[113] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic Span Programs and Succinct NIZKs without PCPs." Springer Berlin Heidelberg, 2013, pp. 626–645. [Online]. Available: http://link.springer.com/10.1007/978-3-642-38348-9_37

[114] J. Barcelo, "User Privacy in the Public Bitcoin Blockchain," vol. 6, no. 1, 2007.

[115] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for bitcoin," in *European Symposium on Research in Computer Security*. Springer, 2014, pp. 345–364.

[116] A. Poelstra, "Mimblewimble," 2016.

[117] Grin, "The Best Automated Trading Robots ? ? ? ?" [Online]. Available: https://grin-tech.org/

[118] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh, "Zether: Towards Privacy in a Smart Contract World." *IACR Cryptology ePrint Archive*, vol. 2019, p. 191, 2019.

[119] J. Xin, P. Huang, L. Chen, X. Lai, X. Zhang, W. Li, and Y. Wang, "WaterCarver: Anonymous Confidential Blockchain System based on Account Model."

[120] Z. J. Williamson, "The aztec protocol," *URL: https://github.com/AztecProtocol/AZTEC*, 2018.

[121] L. Lamport, "Proving the Correctness of Multiprocess Programs," *IEEE Transactions on Software Engineering*, vol. SE-3, no. 2, pp. 125–143, 1977.

[122] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10211 LNCS. Springer Verlag, 2017, pp. 643–673.

[123] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the ACM (JACM)*, vol. 35, no. 2, pp. 288–323, 4 1988.

[124] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming*, 1982. [Online]. Available: http://dl.acm.org/citation.cfm?id=357176

[125] M. Hirt and U. Maurer, "Player simulation and general adversary structures in perfect multiparty computation," *Journal of Cryptology*, vol. 13, no. 1, pp. 31–60, 4 2000.

[126] I. Abraham and D. Malkhi, "The blockchain consensus layer and BFT," *Bulletin of EATCS*, vol. 3, no. 123, 2017.

[127] BitFury Group and J. Garzik, "Public versus Private Blockchains. Part 1: Permissionless Blockchains," pp. 1–23, 2015. [Online]. Available: http://bitfury.com/content/5-white-papers-research/public-vs-private-pt1-1.pdf

[128] R. M. Nadir, "Comparative study of permissioned blockchain solutions for enterprises," in *2019 International Conference on Innovative Computing (ICIC)*. IEEE, 2019, pp. 1–6.

[129] D. Antoine, E. Ben-Hamida, D. Leporini, and G. Memmi, "Asymptotic Performance Analysis of Blockchain Protocols," *arXiv preprint arXiv:1902.04363*, 2019.

[130] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *OSDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[131] D. Ongaro and J. Ousterhout, *In Search of an Understandable Consensus Algorithm*. [Online]. Available: https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro

[132] J. Clow and Z. Jiang, "A Byzantine Fault Tolerant Raft," 2017.

[133] C. Copeland and H. Zhong, "Tangaroa: a byzantine fault tolerant raft," 2016.

[134] H. Moniz, "The Istanbul BFT Consensus Algorithm," *arXiv preprint arXiv:2002.03613*, 2020.

[135] S. De Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain," 2018.

[136] T. PARITY, "Aura - authority round consensus." [Online]. Available: https://openethereum.github.io/wiki/Aura

[137] C. Consensus, "Clique Consensus." [Online]. Available: https://github.com/ethereum/EIPs/issues/225

[138] V. Gramoli, "The Red Belly Blockchain," *personal Communication, Facebook, USA.*, 2017.

[139] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of BFT protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 31–42.

[140] C. Cachin and J. A. Poritz, "Secure intrusion-tolerant replication on the Internet," in *Proceedings International Conference on Dependable Systems and Networks*. IEEE, 2002, pp. 167–176.

[141] A. Mostefaoui, M. Raynal, and F. Tronel, "The best of both worlds: A hybrid approach to solve consensus," in *Proceeding International Conference on Dependable Systems and Networks. DSN 2000*. IEEE, 2000, pp. 513–522.

[142] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 347–356.

[143] C. Berger and H. P. Reiser, "Scaling Byzantine Consensus: A Broad Analysis," 2018. [Online]. Available: https://doi.org/10.1145/3284764.3284767.

[144] J. R. Douceur, "The sybil attack," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2429. Springer Verlag, 2002, pp. 251–260.

[145] C. Dwork, M. Naor, and H. Wee, "Pebbling and proofs of work," in *Annual International Cryptology Conference*. Springer, 2005, pp. 37–54.

[146] J. Tromp, "Cuckoo cycle: a memory bound graph-theoretic proof-of-work," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 49–62.

[147] L. Ren and S. Devadas, "Bandwidth hard functions for ASIC resistance," in *Theory of Cryptography Conference*. Springer, 2017, pp. 466–492.

[148] C. Percival, "Stronger key derivation via sequential memory-hard functions," 2009.

[149] S. King, "Primecoin: Cryptocurrency with prime number proof-of-work," *July 7th*, vol. 1, no. 6, 2013.

[150] A. Biryukov and D. Khovratovich, "Equihash: Asymmetric proof-of-work based on the generalized birthday problem," *Ledger*, vol. 2, pp. 1–30, 2017.

[151] N. Van Saberhagen, "CryptoNote v 2.0," 2013.

[152] E. Wiki, "Introduction to Ethereum Mining." [Online]. Available: https://github.com/ethereum/wiki/wiki/Mining

[153] Nicehash, "DAG size limit problem for 4GB GPUs." [Online]. Available: https://www.nicehash.com/blog/post/dag-size-limit-problem-for-4gb-gpus

[154] P. Ranjan, "Ethereum Core Devs Meeting 81 Notes." [Online]. Available: https://github.com/ethereum/pm/blob/25bd9c2223635c3c3c5f4643fd924f6b44db62a6/AllCoreDevsMeetings/Meeting81.md

[155] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, 2012.

[156] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.

[157] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10821 LNCS. Springer Verlag, 2018, pp. 66–98.

[158] C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas, "Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 913–930.

[159] C. Badertscher, P. Gazi, A. Kiayias, A. Russell, and V. Zikas, "Ouroboros chronos: Permissionless clock synchronization via proof-of-stake," Tech. Rep., 2019.

[160] NTP, "Network Time Protocol." [Online]. Available: http://www.ntp.org/

[161] P. Daian, R. Pass, and E. Shi, "Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake," in *International Conference on Financial Cryptography and Data Security*. Springer, 2019, pp. 23–41.

[162] R. Pass and E. Shi, "The sleepy model of consensus," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10625 LNCS. Springer Verlag, 2017, pp. 380–409.

[163] F. Schuh and D. Larimer, "Bitshares 2.0: general overview," *accessed June-2017.[Online]. Available: http://docs. bitshares. org/downloads/bitshares-general. pdf*, 2017.

[164] E. Kogias, P. Jovanovic, N. Gailly, and I. Khoffi, "Enhancing bitcoin security and performance with strong consistency via collective signing," *25th USENIX Security*, 2016. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_kokoris-kogias.pdf

[165] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford, "Keeping authorities" honest or bust"

with decentralized witness cosigning," in *2016 IEEE Symposium on Security and Privacy (SP)*. Ieee, 2016, pp. 526–545.

[166] G. A. Pierro and R. Tonelli, "Can solana be the solution to the blockchain scalability problem?" in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2022, pp. 1219–1226.

[167] X. Li, X. Wang, T. Kong, J. Zheng, and M. Luo, "From bitcoin to solana–innovating blockchain towards enterprise applications," in *International Conference on Blockchain*. Springer, 2021, pp. 74–100.

[168] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine Agreements for Cryptocurrencies," in *SOSP 2017 - Proceedings of the 26th ACM Symposium on Operating Systems Principles*. Association for Computing Machinery, Inc, 10 2017, pp. 51–68.

[169] J. Chen and S. Micali, "Algorand: A secure and efficient distributed ledger," *Theoretical Computer Science*, vol. 777, pp. 155–183, 7 2019.

[170] R. Pass and E. Shi, "Thunderella: Blockchains with optimistic instant confirmation," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 3–33.

[171] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *arXiv preprint arXiv:1710.09437*, 2017.

[172] E. Buchman, "Buchman, E. (2016). Tendermint: Byzantine fault tolerance in the age of blockchains.Tendermint: Byzantine fault tolerance in the age of blockchains," 2016.

[173] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[174] M. Baudet, A. Ching, A. Chursin, G. Danezis, F. Garillot, Z. Li, D. Malkhi, O. Naor, D. Perelman, and A. Sonnino, "State machine replication in the Libra Blockchain," 2019.

[175] IOTA, "Shimmer IOTA: la solution du coordicide." [Online]. Available: https://www.iota-guide.com/module-shimmer/

[176] S. Popov, H. Moog, D. Camargo, A. Capossele, V. Dimitrov, A. Gal, A. Greve, B. Kusmierz, S. Mueller, A. Penzkofer, O. Saa, W. Sanders, L. Vigneri, W. Welz, and V. Attias, "The Coordicide," Tech. Rep., 2020.

[177] F. Armknecht, G. O. Karame, A. Mandal, F. Youssef, and E. Zenner, "Ripple: Overview and outlook," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9229. Springer Verlag, 2015, pp. 163–180.

[178] Stellar, "Intuitive Stellar Consensus Protocol - Developers Blog." [Online]. Available: https://www.stellar.org/developers-blog/intuitive-stellar-consensus-protocol

[179] D. Mazieres, "The stellar consensus protocol: A federated model for internet-level consensus," *Stellar Development Foundation*, vol. 32, 2015.

[180] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *J. of Math*, vol. 58, no. 345-363, p. 5, 1936.

[181] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling Blockchain Innovations with Pegged Sidechains," pp. 1–25, 2014. [Online]. Available: http://www.blockstream.com/sidechains.pdf://www.bitcoin.fr/public/divers/docs/sidechains.pdf

[182] G. Greenspan, "Multichain private blockchain-white paper," *URl: http://www. multichain. com/download/MultiChain-White-Paper. pdf*, 2015.

[183] S-tikhomirov, "GitHub - s-tikhomirov/smart-contract-languages: A curated collection of resources on smart contract programming languages." [Online]. Available: https://github.com/s-tikhomirov/smart-contract-languages

[184] T. Chen, X. Li, Y. Wang, J. Chen, Z. Li, X. Luo, M. H. Au, and X. Zhang, "An adaptive gas cost mechanism for ethereum to defend against under-priced DoS attacks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10701 LNCS, pp. 3–24, 2017.

[185] D. Perez and B. Livshits, "Broken Metre: Attacking Resource Metering in EVM," 2 2020.

[186] B. Script, "bitcoin/script.h at 0.19 · bitcoin/bitcoin · GitHub." [Online]. Available: https://github.com/bitcoin/bitcoin/blob/0.19/src/script/script.h

[187] IVY, "GitHub - ivy-lang/ivy-bitcoin: A high-level language and IDE for writing Bitcoin smart contracts." [Online]. Available: https://github.com/ivy-lang/ivy-bitcoin

[188] blockstream, "GitHub - ElementsProject/simplicity: Simplicity is a blockchain programming language designed as an alternative to Bitcoin script." [Online]. Available: https://github.com/ElementsProject/simplicity

[189] stefanolande, "GitHub - bitml-lang/bitml-compiler: Compiler for BitML." [Online]. Available: https://github.com/bitml-lang/bitml-compiler

[190] X. Zhang, F. Parisi-Presicce, R. Sandhu, and J. Park, "Formal model and policy specification of usage control," *ACM Transactions on Information and System Security*, vol. 8, no. 4, pp. 351–387, 11 2005. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1108906.1108908

[191] N. Mining, "Rootstock (RSK): Smart contracts on Bitcoin. Medium," 2018.

[192] Stellar, "Stellar Smart Contracts — Stellar Developers." [Online]. Available: https://www.stellar.org/developers/guides/walkthroughs/stellar-smart-contracts.html

[193] S. Developpers, "Horizon Reference Overview — Stellar Developers." [Online]. Available: https://www.stellar.org/developers/reference/

[194] Nxter, "IGNIS — NXTER.ORG." [Online]. Available: https://www.nxter.org/understanding-ignis/#smarttransactions

[195] NeoVM, "NeoVM." [Online]. Available: https://docs.neo.org/docs/en-us/basic/technology/neovm.html

[196] ——, "Smart Contract Writing Limitations." [Online]. Available: https://docs.neo.org/docs/en-us/sc/write/limitation.html

[197] Plutus, "GitHub - input-output-hk/plutus: The Plutus language implementation and tools." [Online]. Available: https://github.com/input-output-hk/plutus/

[198] Z. Team, "GitHub - Zilliqa/scilla: Scilla - A Smart Contract Intermediate Level Language." [Online]. Available: https://github.com/Zilliqa/scilla

[199] H. Fabric, "Smart Contract Processing — hyperledger-fabricdocs master documentation." [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.0/developapps/smartcontract.html

[200] startis academy, "Welcome to Stratis Academy — Stratis Academy documentation." [Online]. Available: https://academy.stratisplatform.com/

[201] Counterparty, "Counterparty." [Online]. Available: https://counterparty.io/

[202] Drivechain, "Projects — Drivechain: Peer-to-Peer Bitcoin Sidechains." [Online]. Available: http://www.drivechain.info/projects/index.html

[203] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," *White paper*, pp. 1–47, 2017.

[204] J. Poon and O. Team, "OmiseGo Decentralized Exchange and Payments Platform," *White paper. https://whitepaperdatabase.com/omisego-omg-whitepaper*, 2017.

[205] Loom, "Loom Network – Production-Ready, Multichain Interop Platform for Serious Dapp Developers." [Online]. Available: https://loomx.io/

[206] V. A. Siris, P. Nikander, S. Voulgaris, N. Fotiou, D. Lagutin, and G. C. Polyzos, "Interledger Approaches," *IEEE Access*, vol. 7, pp. 89 948–89 966, 2019.

[207] V. Buterin, "Chain interoperability," *R3 Research Paper*, 2016.

[208] Catus, "GitHub - hyperledger/cactus: Hyperledger Cactus is a new approach to the blockchain interoperability problem." [Online]. Available: https://github.com/hyperledger/cactus

[209] Polkadot, "Validator · Polkadot Wiki." [Online]. Available: https://wiki.polkadot.network/docs/en/maintain-validator

[210] Parachain, "Bridges · Polkadot Wiki." [Online]. Available: https://wiki.polkadot.network/docs/en/learn-bridges

[211] J. Kwon and E. Buchman, "Cosmos: A network of distributed ledgers," *URL https://cosmos. network/whitepaper*, 2016.

[212] IBC, "ics/ibc at master · cosmos/ics · GitHub." [Online]. Available: https://github.com/cosmos/ics/tree/master/ibc

[213] Waves, "Open platform for Web 3.0 applications." [Online]. Available: https://wavesprotocol.org/

[214] Wanchain, "Wanchain 4.0 T-Bridge Framework Tech Explainer: Part 1 — General Overview - Wanchain." [Online]. Available: shorturl.at/sZ145

[215] V. A. Siris, P. Nikander, S. Voulgaris, N. Fotiou, D. Lagutin, and G. C. Polyzos, "Interledger Approaches," *IEEE Access*, vol. 7, pp. 89 948–89 966, 2019.

[216] Randaow, "GitHub - randao/randao: RANDAO: A DAO working as RNG of Ethereum." [Online]. Available: https://github.com/randao/randao

[217] Corda, "Deterministic JVM — Corda OS 4.4 — Corda Documentation." [Online]. Available: https://docs.corda.net/docs/corda-os/4.4/key-concepts-djvm.html

[218] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town Crier: An Authenticated Data Feed for Smart Contracts," *dl.acm.org*, vol. 24-28-Octo, pp. 270–282, 10 2016. [Online]. Available: http://dx.doi.org/10.1145/2976749.2978326

[219] Provable, "Provable - blockchain oracle service, enabling data-rich smart contracts." [Online]. Available: https://provable.xyz/

[220] GlobalPlatform and Inc, "GlobalPlatform Security Task Force Root of Trust Definitions and Requirements," Tech. Rep., 2017.

[221] G. Team, "Gnosis-whitepaper," *URL: https://gnosis.pm/resources/default/pdf/gnosis_whitepaper. pdf*, 2017.

[222] J. Peterson, J. Krug, M. Zoltu, A. K. Williams, and S. Alexander, "Augur: a Decentralized Oracle and Prediction Market Platform," Tech. Rep., 2018. [Online]. Available: https://www.wunderground.com/history/airport/KSFO/2018/4/10/

[223] M. Abramowicz and M. T. Henderson, "Prediction markets for corporate governance," *Notre Dame L. Rev.*, vol. 82, p. 1343, 2006.

[224] Dai, "The Dai Stablecoin System Whitepaper," Tech. Rep. [Online]. Available: https://makerdao.com/

[225] Maker, "Maker - Feeds price feed oracles." [Online]. Available: https://developer.makerdao.com/feeds/

[226] Hyperledger Composer, "Calling external HTTP or REST services — Hyperledger Composer." [Online]. Available: https://hyperledger.github.io/composer/v0.19/integrating/call-out

[227] Aeternity, "æternity - a blockchain for scalable, secure and decentralized æpps." [Online]. Available: https://aeternity.com/

[228] Blog, "Blockchain Oracles - æternity blog." [Online]. Available: https://blog.aeternity.com/blockchain-oracles-657f134ffbc0

[229] X. Jiang, N. He, R. Zhang, H. Wang, L. Wu, X. Luo, Y. Guo, and T. Yu, "{EOSAFE}: Security Analysis of {EOSIO} Smart Contracts," *usenix.org*. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/he-ningyu

[230] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, "ZEUS: Analyzing Safety of Smart Contracts." in *NDSS*, 2018, pp. 1–12.

[231] R. M. Parizi, A. Dehghantanha, K.-K. R. Choo, and A. Singh, "Empirical Vulnerability Analysis of Automated Smart Contracts Security Testing on Blockchains," Tech. Rep., 2018. [Online]. Available: https://doi.org/xxxx

[232] T. Hewa, M. Ylianttila, and M. Liyanage, "Survey on blockchain based smart contracts: Applications, opportunities and challenges," *Journal of Network and Computer Applications*, vol. 177, p. 102857, 3 2021.

[233] S. Aggarwal and N. Kumar, "Blockchain 2.0: Smart contracts," *Advances in Computers*, vol. 121, pp. 301–322, 1 2021.

[234] T. M. Hewa, Y. Hu, M. Liyanage, S. S. Kanhare, and M. Ylianttila, "Survey on Blockchain-Based Smart Contracts: Technical Aspects and Future Research," *IEEE Access*, vol. 9, pp. 87 643–87 662, 2021.

[235] L. Ante, "Smart contracts on the blockchain – A bibliometric analysis and review," *Telematics and Informatics*, vol. 57, p. 101519, 3 2021.

[236] E. Blog, "CRITICAL UPDATE Re: DAO Vulnerability — Ethereum Foundation Blog." [Online]. Available: https://blog.ethereum.org/2016/06/17/critical-update-re-dao-vulnerability/

[237] P. Zhang, F. Xiao, and X. Luo, "SolidityCheck : Quickly Detecting Smart Contract Problems Through Regular Expressions," Tech. Rep.

[238] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Buenzli, and M. Vechev, "Securify: Practical security analysis of smart contracts," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 67–82.

[239] ChainSecurity, "Security Audit of WBTC DAO's Smart Contracts," Tech. Rep., 2018. [Online]. Available: https://chainsecurity.com

[240] C. Ferreira Torres, M. Steichen, R. Norvill, B. Fiz Pontiveros, and H. Jonker, "ÆGIS: Shielding Vulnerable Smart Contracts Against Attacks," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (ASIA CCS'20), October 5–9, 2020, Taipei, Taiwan*, 2020.

[241] F. Schrans, D. Hails, A. Harkness, S. Drossopoulou, and S. Eisenbach, "Flint for Safer Smart Contracts," *arXiv preprint arXiv:1904.06534*, 2019.

[242] SOLIDITYX, "SolidityX - Secure-by-default superset of Solidity." [Online]. Available: https://solidityx.org/

[243] J. Pettersson and R. Edström, "Safer smart contracts through type-driven development Using dependent and polymorphic types for safer develop-ment of smart contracts," Tech. Rep.

[244] E. Brady, "Idris, a general-purpose dependently typed programming language: Design and implementation," *Journal of functional programming*, vol. 23, no. 5, pp. 552–593, 2013.

[245] KEVM, "KEVM: A Complete Semantics of the Ethereum Virtual Machine - FSL." [Online]. Available: http://fsl.cs.illinois.edu/index.php/KEVM:_A_Complete_Semantics_of_the_Ethereum_Virtual_Machine

[246] M. Brandenburger, C. Cachin, R. Kapitza, and A. Sorniotti, "Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric," *arXiv preprint arXiv:1805.08541*, 2018.

[247] Nick Johnson, "upgradeable.sol · GitHub." [Online]. Available: https://gist.github.com/Arachnid/4ca9da48d51e23e5cfe0f0e14dd6318f

[248] Manuel Araoz, "Proxy Libraries in Solidity – OpenZeppelin blog." [Online]. Available: https://blog.openzeppelin.com/proxy-libraries-in-solidity-79fbe4b970fd/

[249] Kadena, "Kadena-PactWhitepaper — Kadena." [Online]. Available: https://www.kadena.io/kadena-pactwhitepaper

[250] GS1, "Blockchain — GS1." [Online]. Available: https://www.gs1.org/standards/blockchain

[251] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 5 2016, pp. 839–858. [Online]. Available: http://ieeexplore.ieee.org/document/7546538/

[252] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song, "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution," *arXiv preprint arXiv:1804.05141*, 2018.

[253] G. Zyskind, O. Nathan, and A. Pentland, "Enigma: Decentralized Computation Platform with Guaranteed Privacy," pp. 1–14, 2015. [Online]. Available: http://enigma.media.mit.edu/enigma_full.pdf://arxiv.org/abs/1506.03471

[254] A. Unterweger, F. Knirsch, C. Leixnering, and D. Engel, "Lessons Learned from Implementing a Privacy-Preserving Smart Contract in Ethereum," in *2018 9th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2018 - Proceedings*, vol. 2018-Janua. Institute of Electrical and Electronics Engineers Inc., 3 2018, pp. 1–5.

[255] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, 2022.

[256] "EOS Token Contract and Distribution Chart." [Online]. Available: https://etherscan.io/token/tokenholderchart/0x86fa049857e0209aa7d9e616f7eb3b3b78ecfdb0

[257] "GitHub - ethereum/web3.js: Ethereum JavaScript API." [Online]. Available: https://github.com/ethereum/web3.js/

[258] Camel-web3j, "camel/web3j-component.adoc at master · apache/camel · GitHub." [Online]. Available: https://github.com/apache/camel/blob/master/components/camel-web3j/src/main/docs/web3j-component.adoc

[259] Infura, "Ethereum API — IPFS API Gateway — ETH Nodes as a Service — Infura." [Online]. Available: https://infura.io/

[260] Metamask, "MetaMask." [Online]. Available: https://metamask.io/

[261] Eos Java, "EOSIO SDK for Java - EOSIO." [Online]. Available: https://eos.io/build-on-eosio/eosio-sdk-for-java/

[262] P. Eos, "GitHub - Netherdrake/py-eos-api: Unofficial Wrapper for EOS API (eosd) for Python 3.6+." [Online]. Available: https://github.com/Netherdrake/py-eos-api

[263] S. E. Wrapper, "GitHub - EOSEssentials/Scala-API-Wrapper: A Scala wrapper for EOS RPC API." [Online]. Available: https://github.com/EOSEssentials/Scala-API-Wrapper

[264] P. Xiao, "Java programming for blockchain applications," 2019.

[265] A.-K. Wickert, L. Baumgärtner, F. Breitfelder, and M. Mezini, "Python crypto misuses in the wild," in *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2021, pp. 1–6.

[266] R. Li and E. Unger, "Security issues with tcp/ip," *ACM SIGAPP Applied Computing Review*, vol. 3, no. 1, pp. 6–13, 1995.

[267] K. Wu, "An Empirical Study of Blockchain-based Decentralized Applications," *arXiv preprint arXiv:1902.04969*, 2019.

[268] D. Das, P. Bose, N. Ruaro, C. Kruegel, G. Vigna, and G.-V. Vigna, "Understanding security Issues in the NFT Ecosystem," *arxiv.org*, 2022. [Online]. Available: https://arxiv.org/abs/2111.08893

[269] G. Wang and M. Nixon, "Sok: Tokenization on blockchain," in *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, 2021, pp. 1–9.

[270] DAPP, "Dapp.com 2019 Annual Dapp Market Report By Dapp.com." [Online]. Available: https://www.dapp.com/article/dapp-com-2019-annual-dapp-market-report

[271] M. Kim, Y. Kwon, and Y. Kim, "Is Stellar as secure as you think?" in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2019, pp. 377–385.

[272] C. Nguyen, D. Hoang, D. Nguyen, D. N. I. . . . , and u. 2019, "Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities," *ieeexplore.ieee.org*. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8746079/

[273] Coindesk, "Debate 2017. No Incentive? Algorand Blockchain Sparks Debate at Cryptography Event." [Online]. Available: https://www.google.com/amp/s/www.coindesk.com/

[274] E. Voting, "EOS Voting Pattern Analysis." [Online]. Available: https://eosauthority.com/producers_relation?TB_iframe=true&width=1367.1&height=678.6

[275] Y. Kwon, J. Liu, M. Kim, D. Song, and Y. Kim, "Impossibility of full decentralization in permissionless blockchains," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 110–123.

[276] Bips, "bips/bip-0009.mediawiki at master · bitcoin/bips · GitHub." [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0009.mediawiki

[277] H. Fabric, "Procedure for Upgrading from v1.0.x — hyperledger-fabricdocs master documentation." [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/v1.1.0-alpha/upgrade_to_one_point_one.html

[278] V. Buterin, "Ethereum DEV plan," p. 35, 2014. [Online]. Available: https://www.ethereum.org/pdfs/Ethereum-Dev-Plan-preview.pdf

[279] S. Wang, R. Pei, and Y. Zhang, "EIDM: A Ethereum-Based Cloud User Identity Management Protocol," *IEEE Access*, vol. 7, pp. 115 281–115 291, 8 2019.

[280] O. Novo, "Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1184–1195, 4 2018.

[281] P. GIULIO, "Slock.it to Introduce Smart Locks Linked to Smart Ethereum Contracts, Decentralize the Sharing Economy," 11 2015. [Online]. Available: https://bitcoinmagazine.com/articles/slock-it-to-introduce-smart-locks-linked-to-smart-ethereum-contracts-decentralize-the-sharing-economy-1446746719

[282] X. Liang, J. Zhao, S. Shetty, J. Liu, and D. Li, "Integrating blockchain for data sharing and collaboration in mobile healthcare applications," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, vol. 2017-Octob. Institute of Electrical and Electronics Engineers Inc., 2 2018, pp. 1–5.

[283] Shipchain, "The ShipChain Ecosystem · Shipchain." [Online]. Available: https://docs.shipchain.io/docs/intro.html

[284] S. Loss, N. Cacho, J. M. D. Valle, and F. Lopes, "Orthus: A blockchain platform for smart cities," in *5th IEEE International Smart Cities Conference, ISC2 2019*. Institute of Electrical and Electronics Engineers Inc., 10 2019, pp. 212–217.

[285] M. B. Weiss, K. Werbach, D. C. Sicker, and C. E. Bastidas, "On the application of blockchains to spectrum management," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 2, pp. 193–205, 6 2019. [Online]. Available: https://experts.syr.edu/en/publications/on-the-application-of-blockchains-to-spectrum-management

[286] R. Khalid, N. Javaid, A. Almogren, M. U. Javed, S. Javaid, and M. Zuair, "A Blockchain-Based Load Balancing in Decentralized Hybrid P2P Energy Trading Market in Smart Grid," *IEEE Access*, vol. 8, pp. 47 047–47 062, 2020.

[287] J. D. Harris and B. Waggoner, "Decentralized and collaborative AI on blockchain," in *Proceedings - 2019 2nd IEEE International Conference on Blockchain, Blockchain 2019*. Institute of Electrical and Electronics Engineers Inc., 7 2019, pp. 368–375.

[288] B. Bellaj, A. Ouaddah, N. Crespi, A. Mezrioui, and E. Bertin, "Gb-trust: Leveraging edge attention in graph neural networks for trust management in p2p networks," in *22th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2023.

[289] J. Peterson, J. Krug, M. Zoltu, A. K. Williams, and S. Alexander, "Augur: a Decentralized Oracle and Prediction Market Platform (v2.0)," Forecast Fondation, Tech. Rep., 11 2019. [Online]. Available: https://www.wunderground.com/history/airport/KSFO/2018/4/10/

[290] C. Noyes, "BitAV: Fast Anti-Malware by Distributed Blockchain Consensus and Feedforward Scanning," *arXiv preprint arXiv:1601.01405*, 2016. [Online]. Available: https://arxiv.org/abs/1601.01405

[291] A. Ouaddah, A. Abou Elkalam, and A. Ait Ouahman, "FairAccess: a new Blockchain-based access control framework for the Internet of Things," *Security and Communication Networks*, 2017. [Online]. Available: http://doi.wiley.com/10.1002/sec.1748

[292] T. Min, H. Wang, Y. Guo, and W. Cai, "Blockchain games: A survey," in *IEEE Conference on Computatonal Intelligence and Games, CIG*, vol. 2019-Augus. IEEE Computer Society, 8 2019.

[293] EosJs, "GitHub - EOSIO/eosjs: General purpose library for the EOSIO blockchain." [Online]. Available: https://github.com/EOSIO/eosjs

[294] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "LSB: A Lightweight Scalable BlockChain for IoT Security and Privacy," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 180–197, 12 2017. [Online]. Available: http://arxiv.org/abs/1712.02969http://dx.doi.org/10.1016/j.jpdc.2019.08.005

[295] K. J. Peterson, R. Deeduvanu, P. Kanjamala, and K. Mayo, "A Blockchain-Based Approach to Health Information Exchange Networks," 2016.

[296] T. Antipova, "Using blockchain technology for government auditing," in *Iberian Conference on Information Systems and Technologies, CISTI*, vol. 2018-June. IEEE Computer Society, 6 2018, pp. 1–6.

[297] F. Casino, V. Kanakaris, T. K. Dasaklis, S. Moschuris, and N. P. Rachaniotis, "Modeling food supply chain traceability based on blockchain technology," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 2728–2733, 9 2019.

[298] R. M. Haris and S. Al-Maadeed, "Integrating Blockchain Technology in 5G enabled IoT: A Review," in *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies, ICIoT 2020*. Institute of Electrical and Electronics Engineers Inc., 2 2020, pp. 367–371.

[299] S. Dekhane, K. Mhalgi, K. Vishwanath, S. Singh, and N. Giri, "GreenCoin: Empowering smart cities using Blockchain 2.0," in *2019 International Conference on Nascent Technologies in Engineering, ICNTE 2019 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., 1 2019.

[300] P. Mamoshina, L. Ojomoko, Y. Yanovich, A. Ostrovski, A. Botezatu, P. Prikhodko, E. Izumchenko, A. Aliper, K. Romantsov, A. Zhebrak, I. O. Ogu, and A. Zhavoronkov, "Converging blockchain and next-generation artificial intelligence technologies to decentralize and accelerate biomedical research and healthcare," *Oncotarget*, vol. 9, no. 5, pp. 5665–5690, 11 2018. [Online]. Available: www.impactjournals.com/oncotarget/

[301] T. Antipova, "Using blockchain technology for government auditing," in *Iberian Conference on Information Systems and Technologies, CISTI*, vol. 2018-June. IEEE Computer Society, 6 2018, pp. 1–6.

[302] S. Liu and S. He, "Application of Block Chaining Technology in Finance and Accounting Field," in *Proceedings - 2019 International Conference on Intelligent Transportation, Big Data and Smart City, ICITBS 2019*. Institute of Electrical and Electronics Engineers Inc., 3 2019, pp. 342–344.

[303] P. D. Dozier and T. A. Montgomery, "Banking on Blockchain: An Evaluation of Innovation Decision Making," *IEEE Transactions on Engineering Management*, 2019.

[304] J. Brown-Cohen, A. Narayanan, and S. M. Weinberg, "Formal Barriers to Longest-Chain Proof-of-Stake Protocols *," *dl.acm.org*, pp. 459–473, 6 2019. [Online]. Available: https://doi.org/10.1145/3328526.3329567

[305] Corda, "Notaries — Corda OS 4.4 — Corda Documentation." [Online]. Available: https://docs.corda.net/docs/corda-os/4.4/key-concepts-notaries.html

[306] A. Asayag, G. Cohen, I. Grayevsky, M. Leshkowitz, O. Rottenstreich, R. Tamari, and D. Yakira, "A fair consensus protocol for transaction ordering," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 2018, pp. 55–65.

[307] M. Kelkar, F. Zhang, S. Goldfeder, and A. Juels, "Order-Fairness for Byzantine Consensus," Tech. Rep., 2020.

[308] A. Gervais, G. O. Karame, V. Capkun, and S. Capkun, "Is bitcoin a decentralized currency?" *IEEE security & privacy*, vol. 12, no. 3, pp. 54–60, 2014.

[309] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast

payments in bitcoin," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 906–917.
[310] R. Pass and E. Shi, "Rethinking large-scale consensus," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 2017, pp. 115–129.
[311] E. Deirmentzoglou, G. Papakyriakopoulos, and C. Patsakis, "A survey on long-range attacks for proof of stake protocols," *IEEE Access*, vol. 7, pp. 28 712–28 725, 2019.
[312] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine Agreements for Cryptocurrencies," in *SOSP 2017 - Proceedings of the 26th ACM Symposium on Operating Systems Principles*. Association for Computing Machinery, Inc, 10 2017, pp. 51–68.
[313] J. Kwon, "TenderMint : Consensus without Mining," pp. 1–10.
[314] M. Pease, R. Shostak, and L. Lamport, "Reaching Agreement in the Presence of Faults," *Journal of the ACM (JACM)*, vol. 27, no. 2, pp. 228–234, 4 1980.
[315] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10401 LNCS. Springer Verlag, 2017, pp. 357–388.
[316] T. Rocket, M. Yin, K. Sekniqi, R. van Renesse, and E. G. Sirer, "Scalable and Probabilistic Leaderless BFT Consensus through Metastability," 6 2019. [Online]. Available: http://arxiv.org/abs/1906.08936
[317] B. Charts, "Graphiques sur le Bitcoin." [Online]. Available: https://www.blockchain.com/charts/transactions-per-second
[318] E. Buchman, "Tendermint: Byzantine Fault Tolerance in the Age of Blockchains," Tech. Rep., 2016. [Online]. Available: http://atrium.lib.uoguelph.ca/xmlui/handle/10214/9769
[319] OpenZeppelin, "GitHub - OpenZeppelin/openzeppelin-contracts: OpenZeppelin Contracts is a library for secure smart contract development." [Online]. Available: https://github.com/OpenZeppelin/openzeppelin-contracts
[320] I. Age, "Ethereum Difficulty Bomb (Ice Age) - EthHub." [Online]. Available: https://docs.ethhub.io/questions-about-ethereum/what-is-the-difficulty-bomb/
[321] A. Altarawneh, T. Herschberg, S. Medury, F. Kandah, and A. Skjellum, "Buterin's scalability trilemma viewed through a state-change-based classification for common consensus algorithms," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2020, pp. 0727–0736.
[322] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, and R. P. Liu, "Survey: Sharding in Blockchains," *IEEE Access*, vol. 8, pp. 14 155–14 181, 2020.
[323] Y. Murray and D. A. Anisi, "Survey of formal verification methods for smart contracts on blockchain," in *2019 10th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2019 - Proceedings and Workshop*. Institute of Electrical and Electronics Engineers Inc., 6 2019.
[324] J. Liu and Z. Liu, "A Survey on Security Verification of Blockchain Smart Contracts," *IEEE Access*, vol. 7, pp. 77 894–77 904, 2019.
[325] X. Sun, Q. Wang, P. Kulicki, and X. Zhao, "Quantum-enhanced Logic-based Blockchain I: Quantum Honest-success Byzantine Agreement and Qulogicoin," 5 2018. [Online]. Available: http://arxiv.org/abs/1805.06768
[326] D. B. Schenker, "DB Schenker and VeChain pioneer in the use of blockchain for the logistics industry," 2019.
[327] M. Ball, A. Rosen, M. Sabin, and P. Nalini Vasudevan, "Proofs of Useful Work," 2017. [Online]. Available: https://allquantor.at/blockchainbib/pdf/ball2017proofs.pdf
[328] A. Lihu, J. Du, I. Barjaktarevic, P. Gerzanics, and M. Harvilla, "A Proof of Useful Work for Artificial Intelligence on the Blockchain," *arXiv preprint arXiv:2001.09244*, 2020.
[329] P.-W. Chi, Y.-H. Lu, and A. Guan, "A privacy-preserving zero-knowledge proof for blockchain," *IEEE Access*, 2023.