IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.1120000

A Comprehensive Analysis of Blockchain Technology and Consensus Protocols Across Multilayered Framework

MD. RIFAT HOSSAN[1], FOYASL AHAMED NIROB[1], ARAFAT ISLAM[1], TANJIM MAHMUD RAKIN[1] and MD. AL-AMIN[1]

[1]Department of Computer Science, Faculty of Science and Technology, American International University-Bangladesh, Dhaka 1229, Bangladesh

ABSTRACT Blockchain, a distributed and digital ledger technology, has the potential to transform several industries from cryptocurrencies to supply chains. However, the complexity of understanding the technology can be a challenge for most people, especially for newcomers. This study paper aims to provide an in-depth analysis and review of the different layers including the data, network, consensus, smart contract, and application layers. In this paper, we performed an in-depth analysis and review of the components of each layer. We also tried to simulate different consensus algorithms so a reader can understand which blockchain consensus algorithm can be chosen for a specific application. By reading this paper, a reader can understand the blockchain architecture and choose the right algorithms, languages and protocols for building their own blockchain network. The paper also addresses the challenges and difficulties of building a sustainable and secure blockchain. We believe that the choice of consensus algorithm is the most important thing for sustainability in a blockchain network. We show that some consensus algorithms use much more energy than others, and we suggest that developers choose consensus algorithms that use less energy. This paper is a great resource for anyone who wants to learn more about blockchain architecture and consensus algorithms. It gives a complete review of the subject and talks about the difficulties of making a blockchain that will be secure and sustainable. Researchers, developers, and anyone else interested in the future of blockchain technology will be interested in this study.

INDEX TERMS Blockchain, Consensus mechanism, Cryptocurrency, Cryptography, Decentralization, Distributed ledger, Distributed ledger technology (DLT), Hash function, Immutable ledger, Interoperability, Mining, Node, Smart contracts.

I. INTRODUCTION

BLOCKCHAIN a cutting-edge technology, has revolutionized decentralized information management. The evolution from basic shared transaction databases in early computing to sophisticated Distributed Ledger Technologies (DLTs) has led to the emergence of blockchain as an important member of the DLT family [1].

At its core, blockchain offers a fresh perspective on how transactions are shared and validated. It operates through interconnected blocks/nodes that serve as a ledger. These blocks store transactions and ensure their accuracy using cryptographic methods. Unlike traditional client-server architectures, blockchain systems use a peer-to-peer network structure known as P2P. This approach fixed hierarchies and rigid roles [2]. By adopting this dynamic approach, blockchain builds a decentralized ecosystem where diverse participants can thrive independently while facilitating resource sharing and collaboration. The network operates without centralized control [3].

Blockchain's potential is used in various industries and domains, offering solutions to a wide range of challenges. However, the decision to adopt blockchain technology requires careful consideration as decentralization can sometimes incur unjustifiable costs [4], [5]. Consequently, traditional databases continue to hold significance in cases where decentralized models are not commercially viable.

The foundational building blocks of blockchain consist of cryptographic security, seamless communication and transaction data storage, and distributed consensus protocols. These essential elements ensure the validation and sequencing of transactions. The inception of blockchain can be traced back to the enigmatic Bitcoin, which emerged in 2008. Bit-

coin played an important role in pioneering permissionless blockchains, enabling a digital, decentralized, and distributed payment infrastructure. The design of blockchain technology relies on fundamental building blocks. Firstly, cryptographic security governs communication and transaction data storage. Network nodes must unanimously agree on the validity and order of transactions listed in the blockchain. Secondly, distributed consensus protocols address these challenges by allowing each node to vote.

The architecture of the Bitcoin blockchain is strategically designed to defend against attacks from both malicious and rational network nodes. It effectively prevents issues like double spending and Sybil attacks. However, this design is only a part of what blockchain technology can offer. It does not prioritise complete anonymity, scalability, or eco-sustainability concerns [6]– [8]. As a result, alternative cryptocurrencies like "Altcoins" have emerged to address specific limitations of the original Bitcoin model, enhancing portfolio diversification and contributing to market capitalization [9].

In blockchain technology, its core promise goes beyond the popular Bitcoin narrative. This technology enables the establishment of secured, trusted, and decentralized autonomous ecosystems for various scenarios, especially for better usage of legacy devices, infrastructure, and resources [10]. Furthermore, this extends to important aspects like auditability, transparency, immutability, and pseudonymity. These qualities are indispensable for transactional systems that involve diverse entities lacking mutual trust [11].

The evolution of blockchain has led to the emergence of permissioned blockchains. These blockchains enable customized adoption by imposing constraints and shaping network node behaviour [11]. This approach enhances scalability by limiting the number of transaction validators, improving consensus mechanisms, and facilitating the development of distributed applications through smart contracts [11]. However, it is important to evaluate whether fully permissioned blockchains with their complex architecture are always necessary.

This research paper provides a comprehensive guide for newcomers exploring blockchain technology. Acting as a roadmap, it guides readers from learners to intermediate understanding, uncovering the intricate layers that form the foundation of blockchain. The paper begins with an introduction to the basics and progresses through different types of blockchains, explores non-functional attributes, unveils its layers from data to network and examines important consensus mechanisms. It then explores smart contracts and concludes with practical applications, seamlessly connecting each section. This paper serves as a roadmap, to show the path for those who are eager to fully understand the complex world of blockchain technology.

II. LITERATURE REVIEW

Blockchain technology has experienced a journey of evolution. It has transformed from traditional ledgers to the revolutionary concept of decentralized distributed digital ledgers, commonly known as blockchain. The landscape of data storage and transaction processing has been redefined by blockchain technology. Its journey can be traced back to the early concept of ledgers, progressing through digitalization, distribution, and decentralization. In this article, we will explore the evolution of blockchain technology and provide insights into its impact on various industries.

Blockchain is a digital system that acts like a public ledger, keeping track of transactions or records securely and transparently. Instead of having a central authority controlling the information, it is distributed across a network of computers. Each transaction is bundled together in a 'block' and added to a chain of previous blocks, forming a chronological and unchangeable record. This decentralized and tamper-proof nature of blockchain makes it useful for various applications, such as recording financial transactions, verifying ownership, or enabling smart contracts without the need for intermediaries. Fig. 1, represents a general architecture of a blockchain.

The concept of distributed ledgers, which underlies blockchain technology, has its roots in the early days of computer science and cryptography. Here's a brief description of the development and evolution of distributed ledger technology:

1) Traditional ledgers: Traditional ledgers, which have been utilized for centuries, serve as a means to document transactions and monitor financial activities. Typically, traditional ledgers document transactions and monitor financial activities, organizing data in the general ledger (GL) and producing financial statements [12], these ledgers require manual entry, thereby consuming considerable time and posing susceptibility to errors.
2) Emergence of Digital Ledgers: The emergence of digital ledgers occurred with the introduction of computers. This technological advancement enabled the transformation of traditional paper-based ledgers into electronic records, where transactions could be recorded digitally [13]. As a result, this shift not only enhanced operational efficiency and minimized human errors but also presented certain vulnerabilities due to the centralized control of these digital ledgers. Consequently, concerns regarding centralized control persisted.
3) Distributed Digital Ledgers: The emergence of distributed digital ledgers as a solution to centralization issues has been pivotal. These ledgers store transaction records across multiple nodes or computers, e-ensuring improved security and accessibility through redundancy [3]. However, even with these advances, participants in these systems still rely on a certain level of trust among each other.
4) Decentralized Distributed Digital Ledgers: To resolve the issue of trust and enhance security, decentralization emerged as a notable concept. It involved employing decentralized distributed digital ledgers that allowed multiple participants to verify transactions without relying on a central authority [2]. Consensus mechanisms

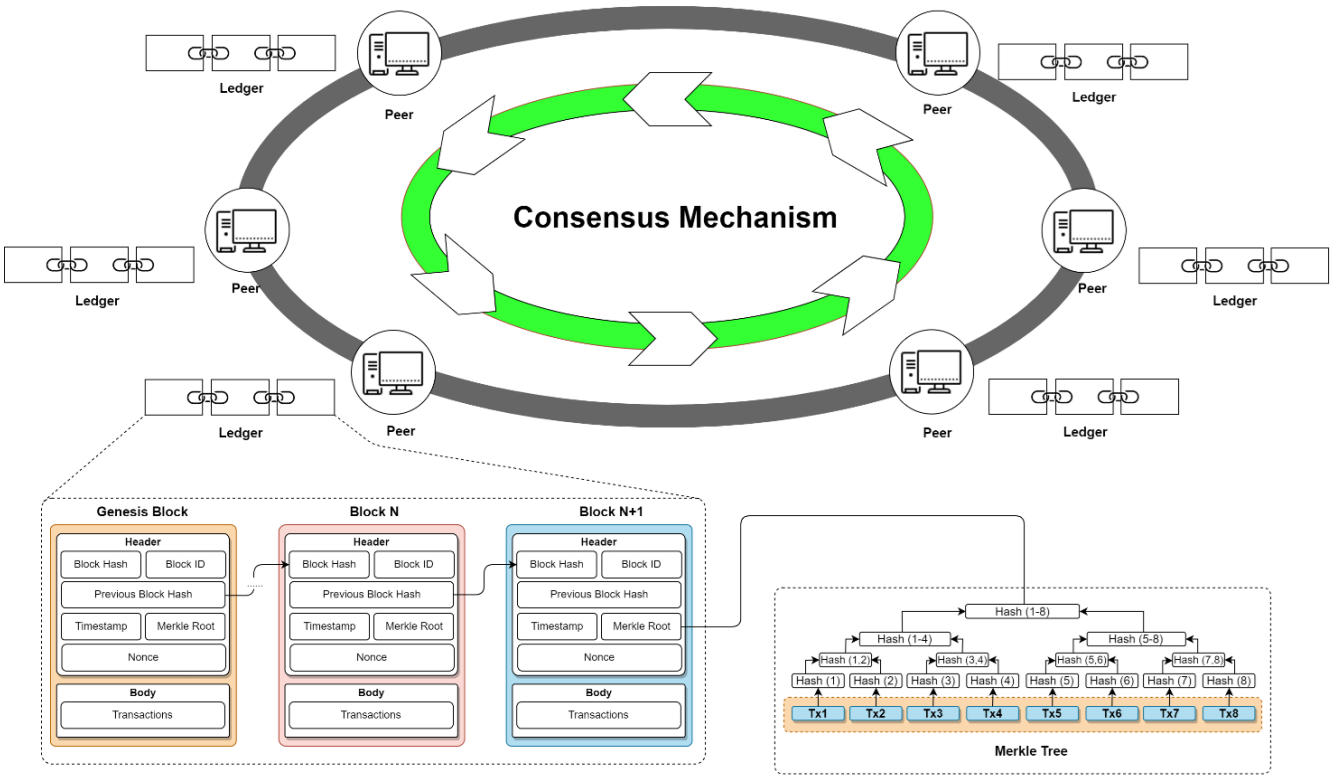

FIGURE 1. **Blockchain architecture**

such as Proof of Work (PoW) and Proof of Stake (PoS) were subsequently introduced to validate these transactions.

5) Birth of Blockchain Technology: The birth of blockchain technology resulted from a series of advancements. A blockchain comprises interconnected blocks, each containing validated transactions. These blocks are linked through cryptographic hashes, ensuring both immutability and data integrity [1]. The decentralized nature and cryptographic security of the blockchain make it resistant to tampering.

Table 1, represents an overview of the various stages in the evolution of blockchain technology. It begins with the use of traditional ledgers before computers were widely used and then progresses to the emergence of digital ledgers in the late 20th century. The concept of distributed digital ledgers follows, addressing trust issues among participants. In the early 21st century, decentralized distributed ledgers come into play, focusing on trustless transactions and consensus mechanisms. Finally, we witness the birth of blockchain itself, introducing blocks, cryptographic hashes, and immutability. This journey spans from the late 2000s to the present day and has greatly influenced industries despite facing challenges regarding adoption and regulations.

Table 2, presents significant milestones in the evolution of blockchain. It begins with the transition from traditional ledgers and explores the birth of blockchain through Bitcoin's whitepaper in 2008. Noteworthy advancements, like Ethereum's introduction of smart contracts in 2014, are also highlighted. The journey depicted here demonstrates how blockchain has expanded beyond cryptocurrencies and gained

TABLE 1. **Evolution of Blockchain Technology**

Stage	Key Features	Challenges	Dates
Traditional Ledgers	Manual recording of transactions	Time-consuming, error-prone	Pre-Computer Era
Digital Ledgers	Electronic records, improved efficiency	Centralization concerns	Late 20th Century
Distributed Digital Ledgers	Redundant copies across nodes	Trust among participants	Late 20th - Early 21st Century
Decentralized Distributed Ledger	Trustless transactions, consensus mechanisms	Scalability, energy consumption	Early 21st Century
Blockchain Technology	Chain of blocks, cryptographic hashes, immutability	Adoption, regulatory hurdles	Late 2000s - Present

traction in various industries, emphasizing its transformative influence on businesses.

Blockchain's evolution has been shaped by key milestones, each contributing to its potential. Some important moments include:

1) Early Cryptographic Protocols (1970s-1980s): During the 1970s and 1980s, a group of researchers and cryptographers embarked on an exploration of cryptographic protocols. They aimed to devise methods that would allow secure communication and transactions between multiple parties without relying on a central authority [18]. This introduction of groundbreaking concepts such as digital signatures, public-key cryptography, and hash functions. These concepts formed the bedrock for the eventual development of distributed ledger technology.
2) The Cypherpunk Movement (1990s): In the 1990s, the Cypherpunk movement emerged, advocating for the use of strong cryptography and decentralized technologies to protect privacy and individual freedoms. The movement attracted computer scientists, mathematicians, and activists who contributed to the development of cryptographic protocols and concepts that laid the groundwork for distributed ledgers [19].
3) BitTorrent (2001): BitTorrent, invented by Bram Cohen in 2001, introduced a peer-to-peer file-sharing protocol that enabled efficient and decentralized distribution of large files across a network of participants [20]. While not a traditional blockchain, BitTorrent's decentralized nature and data-sharing principles influenced later developments in distributed ledger technology.
4) Bitcoin's Emergence (2008-2009): The introduction of Bitcoin by Satoshi Nakamoto brought forth the idea of a decentralized digital currency. This moment marked the birth of blockchain technology, as the first-ever blockchain-based cryptocurrency came into existence [16].
5) Ethereum and Smart Contracts (2014): Proposed by Vitalik Buterin, Ethereum expanded the capabilities of blockchain by introducing smart contracts and programmable platforms. This advancement unlocked new possibilities for self-executing contracts and decentralized applications.
6) Diversification Beyond Cryptocurrencies (2015): The blockchain ecosystem went beyond Bitcoin, giving rise to numerous projects and cryptocurrencies. This diversification paved the way for various innovative use cases.

Fig. 2, represents a visual chronicle of the Blockchain Evolution.

Additionally, various entities including governments, financial institutions, supply chain companies, and other industries are increasingly drawn to the potential advantages of distributed ledger technology. These actors recognize the immense value offered by distributed ledgers in terms of enhancing transparency, ensuring security, and embracing decentralization.

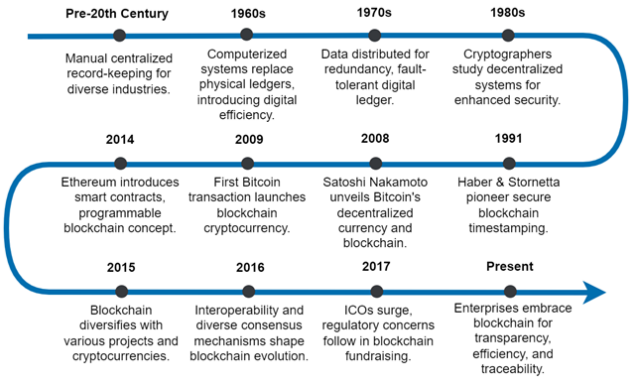

FIGURE 2. History of blockchain

A. TYPE OF BLOCKCHAIN

There are several types of blockchain that can be categorized based on various factors. Here are some common types. We are mainly focused on public blockchain. Fig. 3, represents and visualizes types of blockchain.

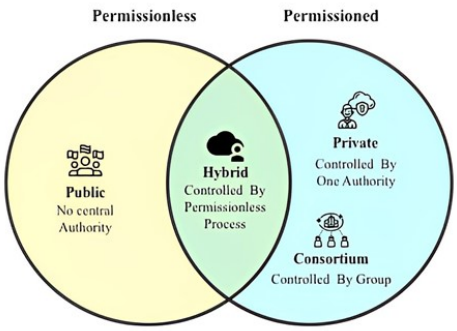

FIGURE 3. Types of blockchain

1) Public Blockchain

A public blockchain is a decentralized digital ledger. It enables transparent recording and verification of transactions across a distributed network of computers. Operating on a consensus mechanism, multiple participants or nodes agree on transaction validity. They also maintain a secure, chronological chain of blocks. Each block contains a batch of transactions and is cryptographically linked to the previous block, forming an immutable chain [5]. One notable feature of public blockchains is their open accessibility. Anyone can join as a node, participate in transaction validation, and access the entire transaction history [7]. This openness fosters a high degree of transparency and trust among participants.

Key features of public blockchains:

1) Decentralization: Decentralization ensures that the network remains independent, with no central authority

TABLE 2. History of Blockchain Technology

Year	Milestone
Pre-20th Century	Ledger Systems: Traditional pen-and-paper ledger systems are used for record-keeping in various industries. These ledgers involve manual entries and centralized control [14].
1960s	Digital Ledger: Early computerized systems replace physical ledgers, introducing digital records and offering enhanced efficiency in data management [15].
1970s	Distributed Digital Ledger: Distributed databases emerge, allowing data to be stored across multiple locations, enabling redundancy and fault tolerance [15].
1980s	Decentralization Concepts: Cryptographers explore decentralized and distributed systems to enhance security and resilience [16].
1991	Cryptographically Secured Chains: Stuart Haber and W. Scott Stornetta introduce a cryptographically secured chain of blocks for timestamping digital documents.
2008	Bitcoin's Whitepaper: An unknown person, or group, under the pseudonym Satoshi Nakamoto releases the Bitcoin whitepaper, outlining a decentralized digital currency and the concept of blockchain [1].
2009	Birth of Bitcoin: The first Bitcoin transaction takes place, marking the launch of the world's first blockchain-based cryptocurrency [1].
2014	Smart Contracts: Ethereum is proposed by Vitalik Buterin, introducing the concept of smart contracts and programmable blockchain platforms [17].
2015	Diversification: Numerous blockchain projects and cryptocurrencies, beyond Bitcoin, emerge, contributing to the diversification of the blockchain ecosystem.
2016	Interoperability and Consensus: Innovations like cross-chain protocols and consensus mechanisms beyond Proof of Work (PoW) gain prominence.
2017	Initial Coin Offerings (ICOs): ICOs become a popular fundraising method for blockchain projects, but regulatory concerns arise.
Present	Enterprise Adoption: Businesses explore blockchain applications beyond cryptocurrencies, aiming for enhanced transparency, traceability, and efficiency in various sectors.

dictating its operations. Each participant has an equal say in how things are run.

2) Transparency: Transparency is a key feature of this system, as all participants can view the entire transaction history. This openness guarantees accountability and prevents any tampe- ring with records.
3) Security: To ensure security and integrity, public blockchains implement consensus mechanisms like proof-of-work or proof-of-stake. These measures safeguard the network from unauthorized access and malicious activities.
4) Trustlessness: One remarkable characteristic of this setup is trustlessness. It enables participants to interact and conduct transactions directly without relying on intermediaries.

2) Private Blockchain

A private blockchain operates within a closed network, reserved for authorized entities [25]. Unlike public blockchains, it provides controlled and restricted access. Only designated participants called nodes validate transactions and maintain the blockchain in a private setting [18]. This setup ensures enhanced privacy and confidentiality since only approved contributors can engage with the ledger. Private blockchains are commonly used by organizations requiring secure and permissioned transaction processing [5].

Key features of private blockchains:

1) Restricted access: Participants must be granted permission by a central authority to join the network and validate transactions.
2) Enhanced privacy: Private blockchains often provide privacy features that restrict access to sensitive information to authorized participants only.
3) Scalability: Since private blockchains have a limited number of participants, they can typically handle a higher volume of transactions compared to public blockchains.
4) Centralization: Private blockchains are more centralized compared to public blockchains, as control over the network is held by a central authority or a consortium of entities.

3) Consortium Blockchain

A consortium blockchain is a digital ledger that combines permissioned and decentralized features to enable secure collaboration among a specific group of participants. While public blockchains allow anyone to join, consortium blockchains are restricted to trusted nodes known by the members of the consortium [21]. This controlled participation enhances privacy and enables faster transaction processing compared to public blockchains.

In consortium blockchains, consensus is reached through

tailored mechanisms suited to the participating organizations' needs. These mechanisms may include variations of proof-of-stake or practical Byzantine fault tolerance, prioritizing efficiency and consensus speed over open participation [22].

The structure of a consortium blockchain guarantees that only approved participants possess the authority to verify transactions and uphold the integrity of the blockchain. This enhances both privacy and security, making it an ideal solution for industries and applications that necessitate controlled data sharing among specific stakeholders. Examples include supply chain management, finance, and healthcare [23].

Key features of consortium blockchains:

1) Semi-decentralization: It is a characteristic of consortium blockchains. These blockchains fall between the decentralized structure of public blockchains and the centralized nature of private blockchains. The consensus and governance responsibilities are shared among pre-approved participants.
2) Permissioned access: Another important aspect is permissioned access, where participants are granted permission to join the network and contribute to the consensus process.
3) Improved scalability: Consortium blockchains also offer improved scalability compared to public blockchains. This is due to their limited number of participants and controlled consensus mechanisms, enabling them to handle higher transaction volumes.
4) Enhanced privacy and confidentiality: Consortium blockchains often prioritize enhanced privacy and confidentiality. They provide privacy features that allow participants to control access to sensitive data within the network.

4) Hybrid Blockchain

A hybrid blockchain combines strengths and overcomes limitations by merging multiple types of blockchains, typically public and private ones. It allows different parts of a network to operate on varying blockchain types, while also facilitating interoperability between the public and private components.

Key features of consortium blockchains:

1) Flexibility: Hybrid blockchains offer flexibility, allowing users to select the most suitable blockchain type for various use cases within the same network.
2) Interoperability: These hybrid blockchains also facilitate interoperability, enabling smooth data and asset exchange between public and private components.
3) Privacy: Hybrid blockchains enhance privacy and scalability by utilizing private blockchains for sensitive transactions or data while benefiting from the transparency and security of a public blockchain for other aspects.
4) Customization: organizations can customize their hybrid blockchain architectures to meet specific requirements, finding a balance between different needs.

B. NON-FUNCTIONAL FEATURES OF BLOCKCHAIN

The non-functional features of a typical blockchain extend beyond its basic transactional function, contributing to overall system performance, security, and usability. These features play a crucial role in evaluating the effectiveness of a blockchain network in real-world scenarios. Let's explore some examples of such features:

1) Immutability: Blockchain's immutability ensures that once data is recorded on the blockchain, it cannot be altered or deleted. This feature provides a high level of trust and data integrity.
2) Integrity: Blockchain technology ensures the integrity of data by using cryptographic techniques such as hashing and digital signatures. Any alteration to the data would result in a change in the cryptographic hash, making it detectable.
3) Transparency: Blockchain provides transparency by allowing all participants to have access to the same information. Transactions and data stored on the blockchain are visible to all participants, promoting accountability and trust in the system.
4) Availability: Blockchain networks strive to maintain high availability, ensuring that the system is accessible and operational for participants at all times. The decentralized nature of blockchain helps prevent single points of failure, enhancing system availability.
5) Cost Efficiency: Blockchain can offer cost efficiency in various ways. By eliminating intermediaries or centralized authorities, blockchain reduces transaction costs. Smart contracts automate processes, reducing the need for manual intervention and associated costs. Additionally, blockchain can streamline supply chains, reducing inefficiencies and costs.
6) Performance: Blockchain performance refers to the speed and efficiency of transaction processing. Non-functional features related to performance include:
7) Transaction throughput: The number of transactions a blockchain can process per second.
8) Confirmation time: The time taken to validate and confirm a transaction.
9) Scalability: The ability of the blockchain network to handle increased transaction volumes without compromising performance.

Fig. 4, represents and compares non-functional features of a typical blockchain for permission-less and permissioned blockchain against public and private blockchain.

III. METHODOLOGY

This research aims to provide a comprehensive analysis of blockchain architectures, from fundamental layers to consensus protocols. To achieve this, we employ a combination of comparative analyses, simulations, taxonomy development, and modelling. This methodology enables a layered evaluation of blockchain platforms, highlighting key mechanisms governing decentralization, security, and performance. Also,

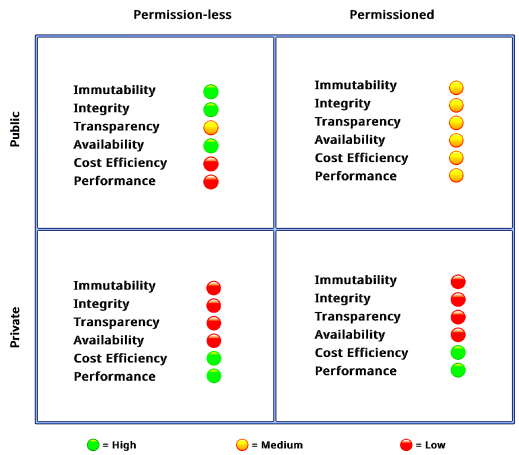

FIGURE 4. **Non-functional features of a typical blockchain**

the study hopes to give users a complete guide to the whole blockchain architecture and help them choose the right protocols, languages, and smart contracts for their needs.

This study uses a mixed-methods approach, which means that it uses both qualitative and quantitative ways to collect and analyze data. The plan for the study includes a review of the literature, case studies, and simulations. The literature study gives an in-depth overview of blockchain technology, including all of its layers, protocols, and mechanisms. In the research, we look at existing blockchain networks and how well they work. In the simulations, we try different consensus algorithms to test and compare them to determine sustainability and efficiency.

Much data was collected to understand different blockchain protocols and layers. Data was gathered through a comprehensive literature review of over 136 sources including whitepapers, academic publications, source code repositories and documentation for major blockchain platforms.

IV. BLOCKCHAIN LAYERS

In a typical blockchain architecture, multiple distinct layers work in tandem to fulfil specific functions that enable system functionality, security, and interoperability. Together, these layers form a robust and decentralized network. Fig. 5, represents the essential components of a blockchain encompassing various key layers:

V. DATA LAYER

The utilization of the block data structure is primarily employed in the data layer to uphold the veracity of data storage. Every node within the network encapsulates the data transactions it receives over a specific time frame into a data block that is timestamped and linked to the current longest main blockchain for storage. This layer involves the main techniques of block storage, chain structure, hash algorithm, Merkle tree, time stamp and so on [24].

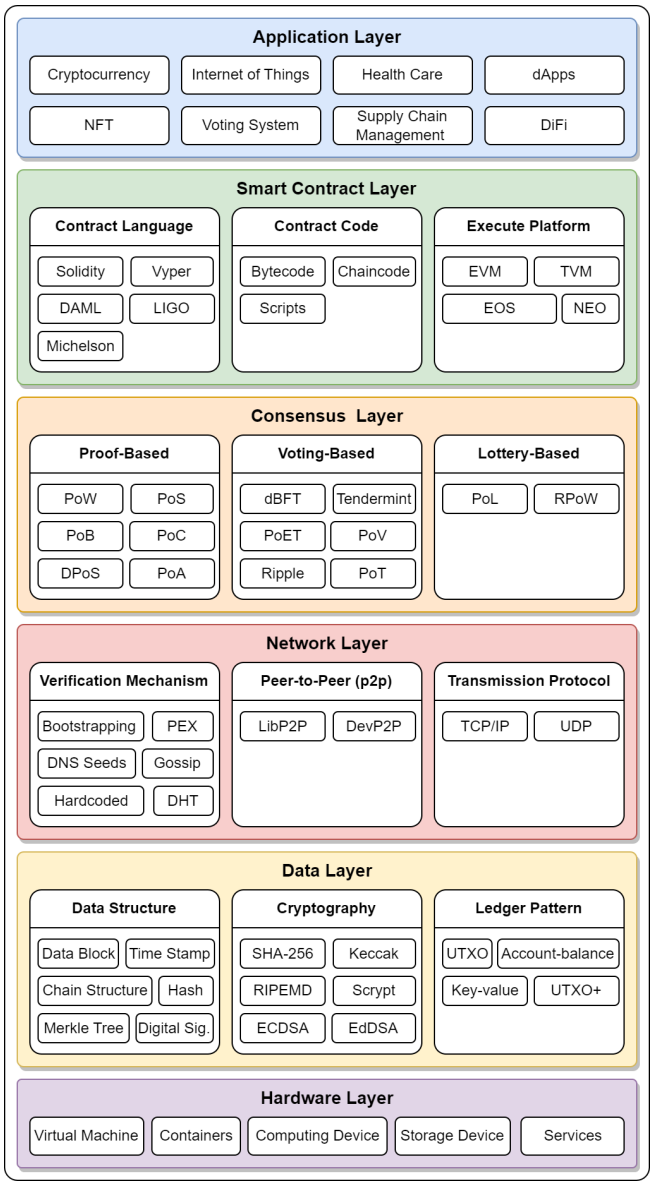

FIGURE 5. **Layers of a typical blockchain**

A. DATA STRUCTURE

The Blockchain data structure is a linked list of data blocks containing information. Each block is identified by a cryptographic hash, derived from algorithms like SHA512 or SHA256 and includes a reference to the previous block's hash. Tampering with a block triggers a hash change and disconnects subsequent blocks due to a mismatched parent hash. New blocks are added at the chain's end but the initial genesis block with no predecessor is uniquely initialized with a zero-parent hash. This structure ensures data integrity and security against unauthorized alterations [25]. Fig. 6, represents the typical block structure of a blockchain.

1) Block: The block is considered the fundamental unit of data in a blockchain. Every individual block comprises a cryptographic hash of the antecedent block, a times-

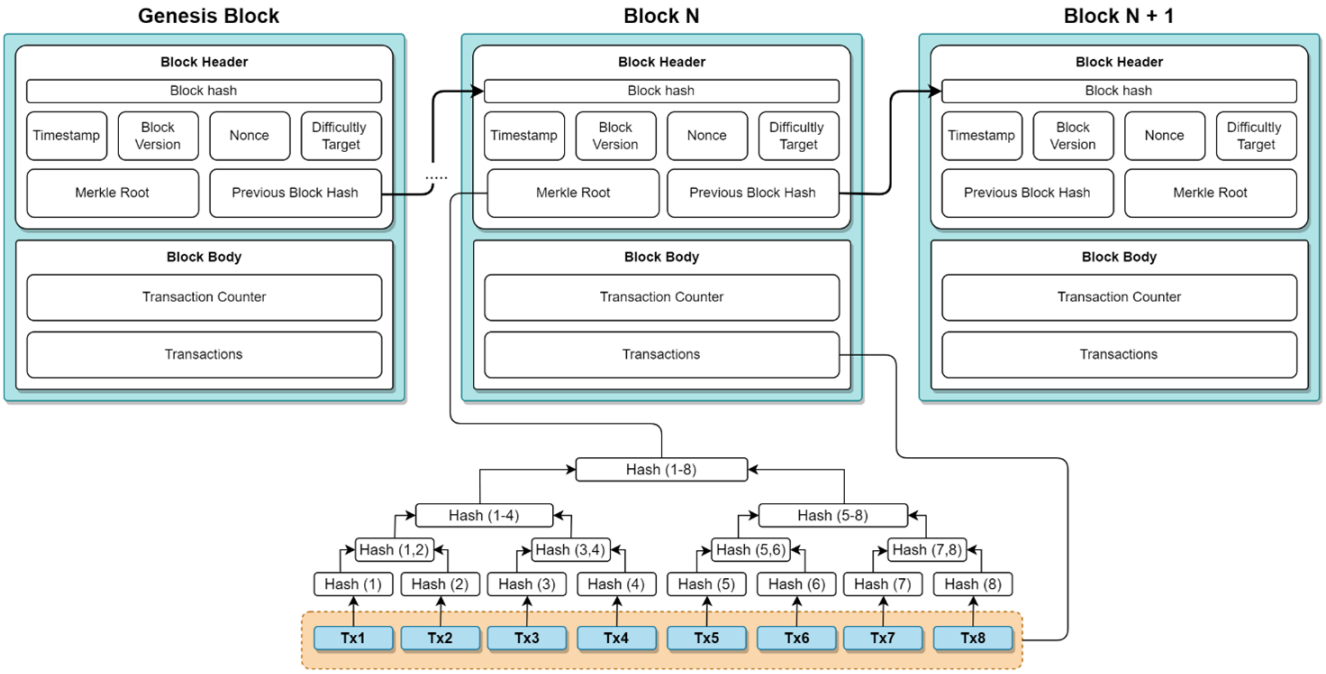

FIGURE 6. **Block structure with block sequence of a blockchain**

tamp, and transactional data.

2) Chain Structure: The chain structure involves the chronological chaining of blocks through hash pointers to create a ledger that is immutable. A blockchain is a sequence of data blocks that are interconnected through cryptographic hashes. Different blockchain uses different chain structures for example bitcoin uses a linear chain of blocks whereas Ethereum uses a more complex chain structure DAG (Directed acyclic graph).
3) Timestamp: The inclusion of a timestamp in each block serves to ensure the integrity of the chronological sequence of both blocks and transactions. A timestamp serves as a documentation of the precise moment and date at which a transaction was generated.
4) Nonce: A 32-bit random integer utilized by blockchain miners to fine-tune and adapt its value for block hashing. Its exclusivity lies in its one-time usability. When an ideal Nonce is pinpointed, it becomes part of the block's hash, initiating a rehash of the block. Infusion of this complexity bolsters hashing intricacy bolsters the blockchain's security and fortifies the mining mechanism.
5) Merkle Tree: The Merkle Tree is a hierarchical data structure in which individual transactions undergo a hashing process, and the resulting hashes are subsequently hashed together until a final root hash is obtained. This facilitates the effective authentication of transactions.
6) Merkle Patricia Tree: Ethereum's 2015 blog post [26] discussed the limitations of Bitcoinstyle light clients for its dynamic state. Ethereum implemented three Merkle trees for verifiable queries on transactions, receipts, and states. To address changes efficiently and prevent DoS attacks, Ethereum adopted Merkle Patricia Tries with hashed nodes, ensuring both cryptographic verification and efficiency through four node types [27].
7) Merkle Bocket Tree: The bucket tree has a hash table at the bottom with buckets for hashed data entries. An upper Merkle tree aggregates lower nodes, minimizing re-hashing during state changes. This ensures even data distribution, efficient updates, and a fixed-size structure to avoid data aggregation [28].

Table 3, summarizes the Merkle tree variants used in different blockchain platforms for efficient data storage and verification.

TABLE 3. **Different types of Merkle trees used in different blockchains**

Blockchain	Merkle Tree Variant
Bitcoin	Merkle tree
Ethereum	Merkle Patricia Tree
Hyperledger	Merkle Bocket Tree
Binance Smart Chain	Merkle Patricia Tree

B. CRYPTOGRAPHY

Blockchain technology relies on public-key cryptography also known as asymmetric cryptography to handle encryption, decryption, and digital signatures to ensure data security and message authenticity. Users create a key pair comprising

a public key (used for encrypting messages) and a private key (kept secret for decrypting messages encrypted with the public key). This process is illustrated in Figure 8 Public-key cryptography helps establish secure transactions and maintain the integrity of data in blockchain networks.

1) Hashing Algorithm

1) SHA-256: SHA-256, belonging to the SHA-2 family of hash functions and a successor to SHA-1 stands out as a robust and highly secure hashing method [29]. Despite its formidable strength, implementing SHA-256 is comparably straightforward to coding SHA-1. As of now, SHA-256 remains unbreeched and free from any compromise reinforcing its reputation as a reliable cryptographic tool.
2) Keccak: Keccak is a versatile cryptographic function developed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. It serves various purposes, with a prominent role as an advanced hash function offering enhanced security compared to earlier ones like SHA-1 and SHA-2. The Keccak family encompasses versions such as Keccak-224, Keccak-256, Keccak-384, and Keccak-512. Notably, Keccak was chosen by NIST in October 2012 as the winner of the SHA-3 Cryptographic Hash Algorithm Competition [30].
3) RIPEMD: The RIPEMD (RIPE Message Digest) family of cryptographic hash functions, established in 1992 and expanded in 1996, encompasses five variations, including RIPEMD, RIPEMD-128, RIPEMD-160, RIPEMD-256, and RIPEMD-320. Among these, RIPEMD-160 has emerged as a prominent choice. Notably, RIPEMD's core architecture employs two concurrent iterations, with an increased round count of five (or four for RIPEMD-128), strategically designed to amplify divergence between parallel rounds and enhance overall security, making it a valuable tool in cryptographic applications [31].
4) Scrypt: Scrypt is a hashing algorithm that employs a password-based key derivation function, generating an extensive array of pseudorandom bit strings. Notably, it demands substantial memory and CPU resources, contributing to its computational intensity [32].

Table 4, provides an overview of different cryptographic hash and signature algorithms used across major blockchain platforms for security purposes like verifying integrity and authenticity. Fig. 7, compares the performance comparison of the hashing algorithm on a device with an Intel i5 8300H processor and 16 GB RAM where the y-axis represents time in seconds and the x-axis represents iteration.

2) Digital Signature and Verification

In the blockchain signing and verification process, cryptographic techniques are employed to ensure data integrity and authenticity. A user initiates a transaction or creates a digital asset by generating a unique digital signature using their private key. This signature is appended to the transaction data and creates a signed message. When the transaction is broadcasted to the network the nodes validate the signature using the corresponding public key to confirm the sender's identity and the transaction's integrity. The transaction is added to a block after verifying. This decentralized and immutable ledger ensures that the transaction history cannot be tampered with providing a secure and transparent system for various applications including digital currencies and asset management. Fig. 8, illustrates the process of digitally signing a document using blockchain technology to ensure its authenticity and immutability. First, the document is passed through

TABLE 4. Different Hashing Algorithms Used in Different Blockchains [124]

Algorithm	Blockchains	Function	Hash/Signature Size
SHA-256	Bitcoin, Ethereum, Dash, Litecoin, Zcash, Monero, Ripple, NXT, Byteball	Hash function	256 bits
Ethash (Keccak-256, Keccak-512)	Ethereum	Hash function	256/512 bits
Scrypt	Litecoin, NXT, Dogecoin	Hash function	256 bits
PIPEMD160	Bitcoin, Ethereum, Litecoin, Monero, Ripple, Bytecoin	Hash function	160 bits
Keccak-256	Monero, Bytecoin	Hash function	256 bits
Keccak-384	IOTA	Hash function	384 bits

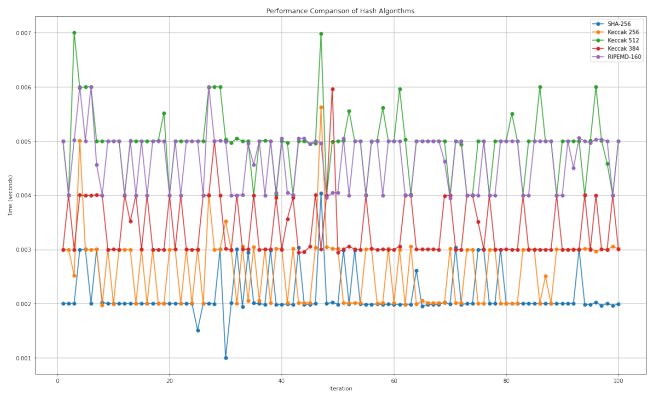

FIGURE 7. Figure Performance comparison of Hashing Algorithm (intel i5 8300H, 16GB)

a cryptographic hash function to generate a unique fingerprint called the hash value. The signer then encrypts this hash value using their private key to create a digital signature. This signature is broadcast to the decentralized blockchain network for validation. The network nodes verify the signature using the signer's paired public key. If valid, the encrypted hash is permanently recorded on the distributed ledger across all nodes. As a result, any subsequent modification to the original document will change its hash value, which can be easily detected by comparing it against the immutable blockchain record. The validated digital signature thus provides proof of the signer's identity and the document's integrity without centralized authorities. The network nodes verify the signature using the signer's paired public key. If valid, the encrypted hash is permanently recorded on the distributed ledger across all nodes. As a result, any subsequent modification to the original document will change its hash value, which can be easily detected by comparing it against the immutable blockchain record. The validated digital signature thus provides proof of the signer's identity and the document's integrity without centralized authorities. This trustless architecture enables transparent e-signing workflows at scale.

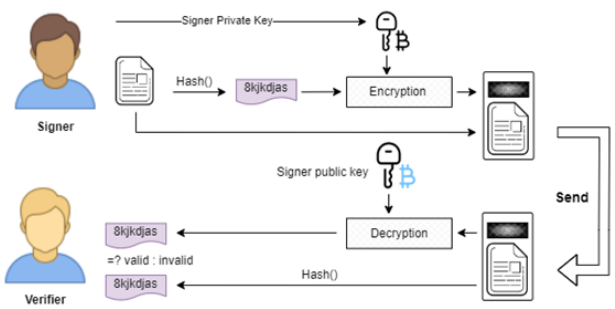

FIGURE 8. Digital Signature and Verification

1) ECDSA: The Elliptic Curve Digital Signature Algorithm (ECDSA), a standardized evolution of the Digital Signature Algorithm (DSA) developed by David Kravitz, utilizes elliptic curve operations over finite fields instead of DSA's prime-based arithmetic. This enhancement leads to improved efficiency and the ability to use shorter key lengths while upholding the same security level as DSA [33].
2) EdDSA: Edwards-curve Digital Signature Algorithm (EdDSA), a swift and secure digital signature scheme within public-key cryptography. It leverages a modified Schnorr signature approach built upon twisted Edwards curves, aiming to surpass the speed of current signature methods while maintaining robust security measures [34].
3) Schnorr: Schnorr signature denotes a digital signature created through the Schnorr signature algorithm, originally formulated by Claus Schnorr. The signature originating from the Schnorr identification scheme through the Fiat-Shamir transform stands as one of the initial discrete logarithm-based signature methods introduced. Its appeal lies in its straightforwardness, efficiency, short signature size and the ability to precalculate exponentiations for rapid online signature generation and garnering significant interest [35].
4) BLS: The BLS digital signature, a Boneh–Lynn–Shacham (BLS) employs bilinear pairings and elliptic curve groups to verify authentic signers. Its distinct feature is aggregation enabling combined signatures and keys. Efficient and deterministic with strong cryptographic properties [36].

Table 5, compares different digital signature algorithms used in major blockchain platforms and their respective advantages.

TABLE 5. Comparison of Different Digital Signing Algorithms Used in Different Blockchains

Signing Algorithm	Blockchains	Advantages
ECDSA	Bitcoin, Ethereum, Dogecoin, Polkadot, Wrapped Bitcoin, Litecoin, Avalanche, Stellar, Ripple, Dash, Torn	Higher security, Lower computation, and fast processing.
EdDSA	Cardano, Solana, Polkadot	Efficient, resilience to side-channel attacks, bolstering security, reduces storage overhead.
Schnorr	Polkadot	Efficient and good security assumptions, native multisignature through signature aggregation
BLS	Ethereum 2, Chia	Compact signature length, represented as a singular elliptic curve point—half the size of Schnorr or ECDSA signature, no random number generator is required

3) Ledger Pattern

The Ledger Pattern is a concept in accounting that refers to the systematic recording of financial transactions in a ledger.

The Account-Based approach involves the modification of balances associated with individual user accounts through transactions. Employed within the context of the cryptocurrency known as Bitcoin. The Event-Based approach involves encoding arbitrary logic and data within transactions. Utilized within the Ethereum ecosystem.

1) UTXO: The Unspent Transaction Output (UTXO) is a set in Bitcoin that comprises untapped transaction

outputs at a specific time. When crafting new transactions, the existing UTXOs are utilized to access funds and simultaneously create fresh UTXOs. Transactions effectively alter the UTXO set by consuming UTXOs as inputs and generating new ones as outputs [37].

2) Account-balance: Account-balance offers enhanced intuitiveness by effectively managing the balance of individual accounts as a collective state within the blockchain. This approach eliminates the need for distinct transaction inputs and outputs as the blockchain's overall state is a direct result of executed transactions. Notably, when a transaction occurs, the relevant account states are promptly adjusted to reflect the transfer by showcasing the model's streamlined and comprehensive handling of blockchain dynamics [38].
3) UTXO+: The concept underlying the UTXO+ model involves retaining the UTXO structure while implementing necessary modifications to achieve the advantages typically associated with account-based models. In this approach, the traditional notion of an 'account' is absent, and instead, the state becomes an inherent part of the transaction outputs. However, these operations tend to feel somewhat contrived and necessitate a high degree of abstraction, along with considerable intricacies in their implementation [38].
4) Key-value: This versatile approach provides the flexibility to adopt either a UTXO-like structure or an account-like structure based on the specific business requirements built upon the blockchain. This adaptability empowers developers to tailor the blockchain's data representation to align with their intended use case [38].

Table 6, shows different approaches for tracking blockchain state used in different blockchain networks and their comparison based on attributes like security, scalability, anonymity, applicability, efficiency, and model complexity.

VI. NETWORK LAYER

The network layer in blockchain technology is like the underlying infrastructure that enables communication among the participants. It consists of mechanisms such as network structure (peer-to-peer networking), data forwarding, and data security. Peer-to-peer networking ensures that everyone in the network eventually receives the information, resembling a gossip network. Data forwarding focuses on efficiently transmitting transaction and block data across the network. Data security checks the integrity and authenticity of the information. Improving these mechanisms is crucial for enhancing the speed, security, and scalability of blockchain systems.

Fig. 9, shows the schematic diagram of both decentralized and centralized diagrams. In the decentralized diagram, every node/computer works like a server and a user. As a result, they all can share data among themselves resulting in promoting security. Whereas in the centralized server, there is only one server where every data of the network is stored.

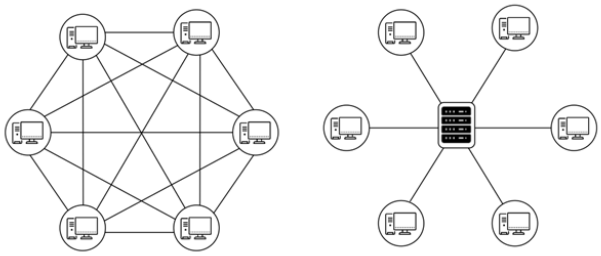

FIGURE 9. Decentralized vs Centralized network

A. NODE/PEER DISCOVERY

Node Discovery refers to the mechanism through which nodes within a blockchain network locate and establish connections with one another. The establishment of a decentralized network and the facilitation of peer-to-peer communication are pivotal processes. The aforementioned procedure holds significant importance in maintaining the decentralized structure of the blockchain, as it guarantees the resilience, availability, and timely synchronization of the network. In a blockchain network, nodes can be categorized into two types [39].

1) Full Nodes: Full nodes are responsible for storing a comprehensive replica of the blockchain and performing the validation of transactions and blocks. They perform an essential function in upholding the security and integrity of the network.
2) Light Nodes: Light nodes, in contrast to full nodes, do not retain the complete blockchain. Instead, they depend on full nodes for the purpose of transaction validation and other related functionalities. Lightweight nodes are commonly employed by end-users or devices with limited resources, such as mobile phones or Internet of Things (IoT) devices.

The discovery of nodes plays a pivotal role in the seamless operation of a blockchain network, as it facilitates vital communication and information sharing among nodes [40]. It enables the seamless integration of new nodes into the network, empowers existing nodes to identify and establish connections with their counterparts, and expedites the dissemination of transactions and blocks throughout the network.

1) Bootstrapping

This method entails a centralized initialization procedure in which a limited number of nodes are manually chosen and interconnected. After that, these nodes disseminate the network information to additional nodes, progressively augmenting the network. In both Avalanche and Algorand, new nodes join known nodes (called "bootstrap nodes") to find out about other nodes in the network.

TABLE 6. Comparison of State Tracking Approaches

State Tracking Approach	Security	Scalability	Anonymity	Applicability	Efficiency	Model Complexity	Platforms
UTXO	High	High	Moderate	High	High	Moderate	Bitcoin, Litecoin, Dogecoin, ZCash, MultiChain
Account	High	Low	High	High	Moderate	Low	Ethereum, Tezos, IOTA, Ripple, Stellar
UTXO+	High	High	High	High	High	High	Corda, Chain Core, Qtum
Key Value	Moderate	Moderate	Low	Moderate	Moderate	Moderate	Hyperledger Fabric, Kadena, Sawtooth Lake

2) Peer Exchange (PEX)

The PEX protocol is a decentralized approach that facilitates the discovery of nodes by employing a random walk technique. Nodes engage in the exchange of randomly selected peer addresses, hence facilitating the steady expansion and adaptive response of the network to fluctuations within its structure.

3) DNS Seeds

DNS Seeds employ a centralized Domain Name System (DNS) server for the purpose of storing and disseminating a roster of starting nodes. Subsequently, the nodes can utilize this list to establish connections with the network and ascertain the presence of additional nodes.

4) Hardcoded 'Seed' Nodes

Some blockchain networks, like Bitcoin, include the IP addresses of "seed" nodes in their code. These nodes are generally reliable and well-connected nodes, which the new node can connect to initially and get information about other nodes in the network.

5) Distributed Hash Table (DHT)

Some networks use a DHT, a type of decentralized distributed system, for node discovery. In such a system, nodes maintain a table containing information about other nodes in the network. For instance, Ethereum uses a custom protocol called Kademlia for peer discovery. It's a type of Distributed Hash Table (DHT) where each node maintains connections with other nodes in the network.

6) Gossip Protocols

Some networks use a DHT, a type of decentralized distributed system, for node discovery. In such a system, nodes maintain a table containing information about other nodes in the network. For instance, Ethereum uses a custom protocol called Kademlia for peer discovery. It's a type of Distributed Hash Table (DHT) where each node maintains connections with other nodes in the network.

1) Decentralization: How distributed the network is without centralized points of failure.
2) Censorship Resistance: Ability to operate without being shut down or blocked by authorities.
3) Convergence Speed: How quickly the network reaches a stable, connected state.
4) Complexity: Level of complexity to implement the network layer protocol.

Table 7, compares different methods of node/peer discovery methods based on the parameters of decentralization, censorship resistance, convergence speed, and complexity with their pros and cons.

Table 8, provides an overview of the node discovery methods used by different blockchain networks. The table presents an overview of ten blockchain networks, including Bitcoin, Ethereum, Solana, Avalanche, Polkadot, Cardano, Algorand, Litecoin, Zcash, and Vechain. For each network, the table shows the method used for discovering nodes.

B. P2P PROTOCOL

In blockchain networks, decentralization is established by adopting a Peer-to-Peer network. There is no centralized node in the Peer-to-Peer network. A P2P network lacks centralized nodes and hierarchical structures. Each node will take on network routing, block data verification, block data forwarding, node discovery etc.

P2P networks can be categorized into multiple types such as structured, unstructured, and hybrid networks. The unstructured peer-to-peer networks are formed by multiple nodes in random order. In terms of efficiency, unstructured P2P is better than structured ones. Structured peer-to-peer systems nodes are properly organized which makes it more effective in searching through the network. Hybrid models are combinations of P2P and traditional centralized networks and when compared to the structured and unstructured P2P models, hybrid networks tend to present improved overall performance as it uses the best of both networks. Here is the list of top blockchains and the particular network structure.

1) LibP2P

Libp2p is an independent networking library designed for IPFS. It's a versatile and adaptable network framework that leverages existing tools and features for functions like NAT

TABLE 7. Node/Peer Discovery Methods Evaluation

Method	Pros	Cons	Decentralization	Censorship Resistance	Convergence Speed	Complexity
Bootstrapping	Faster Initialization, Simple	Centralized, Limited Scalability	Low	Low	Fast	Simple
Peer Exchange (PEX)	Decentralized, Scalable, Adaptive	Complexity, Connectivity issues, Privacy	High	High	Moderate	Simple
DNS Seeds	Centralized DNS, Faster Initialization	Centralized failure, Limited scalability	Low	Moderate	Fast	Simple
Hardcoded "Seed" Nodes	Easy, Faster Initialization	Limited scalability, Centralized failure	None	None	Fast	Simple
Distributed Hash Table (DHT)	Decentralized, Scalable, Adaptive	Added complexity, Connectivity issues	High	High	Fast	Complex
Gossip Protocols	Decentralized, Scalable, Adaptive	Added complexity, Connectivity issues	High	High	Fast	Simple

TABLE 8. Different Node Discovery Methods Used in Different Blockchains

Blockchain	Node Discovery Method
Bitcoin	Hardcoded bootnode IPs, DNS seeds [41]
Ethereum	DevP2P discovery protocol using Kademlia DHT [42]
Solana	Libp2p gossipsub protocol [43]
Avalanche	Snowball sampling [44]
Polkadot	Libp2p gossipsub protocol
Cardano	Ouroboros Praos peer discovery [45]
Algorand	Gossip protocol broadcasting [46]
Litecoin	DNS Seed List [47]
Zcash	DNS Seed List and IRC Bootstrap [48] [49]
Vechain	Authority Masternode Network [50]

traversal, peer discovery, routing, stream and protocol multiplexing, encryption, and authentication [51].

2) DevP2P

DEVp2p established P2P connections by utilizing port 30303 as the default listening port but allowing alternate ports. The initial handshake involves a 'Hello' message. If no matching subprotocols are found then the connection is terminated. 'Disconnect' signals impending disconnection with an optional reason. 'Ping' and 'Pong' messages confirm peer presence [52].

Table 9, provides an overview of how different blockchain platforms use different peer-to-peer protocols.

TABLE 9. Different P2P Protocols Used in Different Blockchains

Peer-to-Peer Protocol	Blockchain Platforms
P2P Protocol	Bitcoin [53], Cardano [54], Litecoin, Tezos [55], EOS [56], Tron [57], NEO [58], IOTA [59]
Libp2p Protocol	Polkadot [60], Ethereum [61]
DevP2P	Ethereum [61], BNB [62]
Tower BFT (custom P2P)	Solana [43]

C. TRANSMISSION PROTOCOL

A transmission protocol is a fundamental communications standard that facilitates the seamless exchange of messages between application programs and computing devices within a networked environment. This precisely developed infrastructure is particularly designed to convey data packets across the internet's complicated topography, ensuring the efficient transmission of information and messages across varied and linked networks.

1) TCP/IP

The TCP/IP protocol suite, Transmission Control Protocol (TCP) and Internet Protocol (IP). Its main aim was to create a universal communication system across diverse networks by allowing seamless interaction across geographical boundaries. This interconnection known as internetwork or internet facilitates communication between hosts on separate networks [63].

2) UDP

The User Datagram Protocol (UDP) establishes a datagram mode of communication within interconnected computer net-

works, operating alongside the Internet Protocol (IP) [64]. It offers a streamlined approach for applications to transmit messages with minimal protocol complexities. Notably, this protocol lacks assured delivery and duplicate prevention, focusing on transactional efficiency [65].

Table 10, compares the Transmission Control Protocol/Internet Protocol (TCP/IP) and User Datagram Protocol (UDP) in the context of their adoption within different blockchain networks.

D. SECURITY

The way information is transmitted within a blockchain relies heavily on a peer-to-peer network, similar to how individuals communicate with each other. In this network, nodes interact with each other to share information, much like people exchanging messages. However, this openness also creates vulnerabilities, as there is a risk of security threats when an attacker enters the network. This attacker can easily target nodes with weaker security measures, posing a danger to other nodes within the network. It's important to note that nodes in a public blockchain network can vary widely in terms of security, ranging from regular home computers to powerful cloud servers. Therefore, it is inevitable that some nodes will have lower security levels, and an attack on one of these nodes can have a direct impact on the security of other nodes [70].

DDoS attacks have become a significant concern for blockchain security. The decentralized and complex nature of blockchain networks along with their non-uniform network protocols, make them attractive targets for such attacks. In recent years, DDoS attacks have resulted in the shutdown of machines and mining operations, affecting exchanges and mines associated with blockchain networks like Mt. Gox, Fomo 3D, LooksRare, and Youbite [71].

1) Peer Authentication: A peer authentication mechanism to verify the authenticity of participating nodes. Nodes authenticate each other's identities using cryptographic techniques, such as digital signatures or public-key infrastructure (PKI). This ensures that only trusted and authorized nodes can participate in the Ethereum network.
2) Firewall Protection: Firewall protection is a crucial security measure for safeguarding nodes and infrastructure. It operates at the network level by establishing rules that manage incoming and outgoing network data. These rules ensure that only legitimate communication is permitted while potentially harmful traffic is blocked. By setting up firewalls, access to Ethereum-related ports and protocols can be limited effectively preventing unauthorized entry and potential attacks on the network.
3) Network Monitoring: Network monitoring tools are used to observe the behavior and performance of Ethereum nodes and the overall network. Monitoring software can track network connectivity, node health, block propagation times, transaction confirmation rates and other network metrics. Monitoring helps identify potential issues, performance bottlenecks or abnormal behaviour that may indicate network disruptions or attacks.
4) Network-Level Encryption: Transport Layer Security (TLS) is used to provide network-level encryption. Encryption ensures that communication between nodes is secure and protects against eavesdropping or tampering with network traffic. Network-level encryption safeguards sensitive information and helps maintain the privacy and integrity of data transmitted within the Ethereum network.
5) DDoS Mitigation: DDoS mitigation techniques are used to protect against Distributed Denial of Service (DDoS) attacks. Techniques such as rate-limiting traffic shaping or IP filtering can be implemented to detect and mitigate abnormal or excessive traffic that may indicate a DDoS attack.

VII. CONSENSUS LAYER

The consensus layer in a blockchain is the key component which is responsible for reaching an agreement between network participants [6]. And, is is done without relying on a central authority. It serves as the basis for maintaining blockchain integrity, security and immutability by using common verification and authentication of transactions before including those on-chain [5]. It ensures that all participants in the network have an agreement on the state of the blockchain, even in the presence of errors or malicious actors. The consensus layer addresses the challenge of establishing consensus in a decentralized and trustless environment where multiple actors or nodes with potential conflicts of interest come together in the network [3]. By implementing some algorithm or protocol, the consensus layer enables nodes to collectively agree on the order and validity of transactions, and this ensures the tamper-proof operation of the blockchain ecosystem [23].

A. CONSENSUS ALGORITHM

A consensus algorithm is a set of rules and procedures that allow a network of computers or nodes in a blockchain network to agree on the validity of a transaction or a block. There are different consensus algorithms which is a crucial part of a consensus layer, currently used in different blockchain networks. Those consensus algorithms are written in a way so that it is trusted by everyone.

B. IMPORTANCE OF CONSENSUS ALGORITHM

Consensus algorithms are crucial in blockchain systems. It solves the challenges of reaching consensus in a decentralized and trusted environment and ensures that transactions are confirmed accurately and securely. By preventing double spending and unauthorised ledger changes, consensus algorithms lay the foundation for building a transparent and trusted digital ecosystem [23]. Mainly, consensus algorithm is an essential element or component in a consensus layer, which helps nodes to reach a secure and decentralized agreement on the state of the ledger [7]. It is important to ensure the trust and

TABLE 10. Different Transmission Protocols Used in Different Blockchains

Transmission Protocol	Blockchain Platform	Pros	Cons
TCP/IP	Bitcoin [53], Ethereum [61], BNB [62], Cardano [66], Polkadot [67], Litecoin, Tezos [55], Cosmos [68], EOS [56], Tron [57], IOTA [59], Neo [58]	Supports multiple protocols, reliable, no OS dependency.	Slower, not suitable for LAN or PAN.
UDP	Solana [69]	Broadcast and multicast available, faster.	Not reliable, drops packets in case of error

reliability of blockchain networks and enable their successful deployment in a wide range of applications [72].

C. CATEGORY OF CONSENSUS ALGORITHM

There are different types of consensus algorithms. The well-known Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS), Practical Byzantine Fault Tolerance (PBFT), Proof of Authority (PoA), Raft etc. But those algorithms can be broken down into 3 categories.

1) Proof-Based
2) Voting-Based
3) Lottery-Based

The different consensus algorithms in different categories are shown in Fig. 10. That figure breaks down different consensus algorithms category-wise.

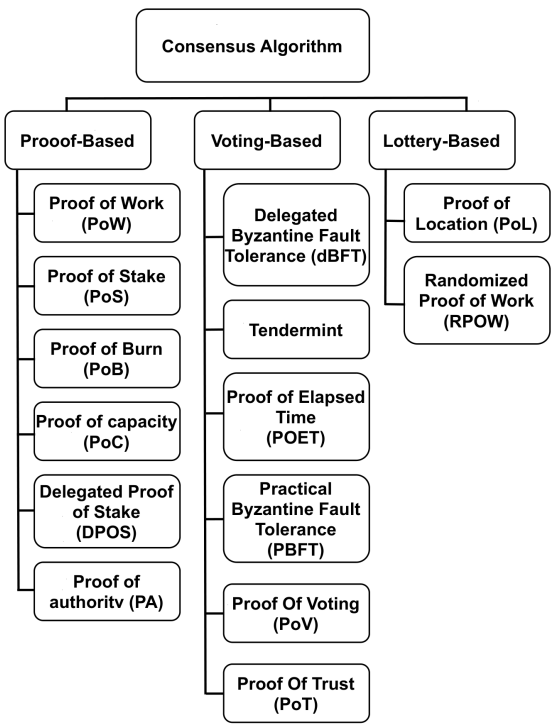

FIGURE 10. Category of consensus Algorithm

1) Proof-Bases Consensus Algorithm

Proof-based consensus algorithms require participants to provide some form of evidence to verify their actions or claims.

a: Proof-of-Work (POW):

Proof of Work (PoW) relies on computational work as evidence to verify and secure transactions. In PoW, participants called "miners" compete to find solutions to complex mathematical puzzles, solving or finding hash which require huge amounts of computing power and energy. Also known as proof, this solution serves as proof that the miner has done the necessary work. Once a miner finds a solution, it is sent to the network where other nodes can easily check the validity of the proof [74]. This evidence-based approach ensures that participants have invested computational resources so that malicious attackers cannot compute when modifying blockchain history or introducing fraudulent transactions. The amount is large and it takes time. PoW is considered an evidence-based consensus algorithm because it relies on computation (evidence) as proof of participation and contributes to the security and integrity of the entire blockchain network. Fig. 11, represents the flowchart of proof-of-work.

b: Proof-of-Stake (POS):

Proof-of-Stake (PoS) consensus algorithm is used in blockchain networks which selects validators to create new blocks based on their ownership or share in the network. PoS requires participants to demonstrate ownership of a specific amount of cryptocurrency or tokens to be eligible for the validator role. The selection process is typically determined by a deterministic algorithm that considers factors like stake size, coin age, or a combination [75]. Known as a proof-based consensus algorithm, PoS mandates participants to validate transactions and create new blocks by proving their stake in the network. This verification is accomplished by digitally signing the transaction using the private key associated with their staked tokens. Thus, participants exhibit both personal responsibility and dedication to the network. Fig. 12, represents the flowchart of proof-of-stake.

c: Proof-of-Burn (POB):

Proof of Burn (PoB) is a consensus mechanism in which individuals deliberately eliminate or "burn" their cryptocur-

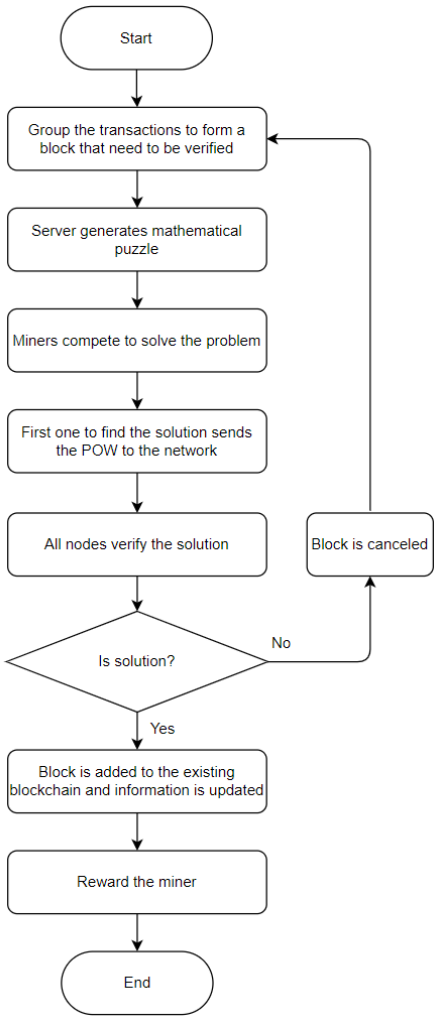

FIGURE 11. Flowchart of PoW Algorithm [73]

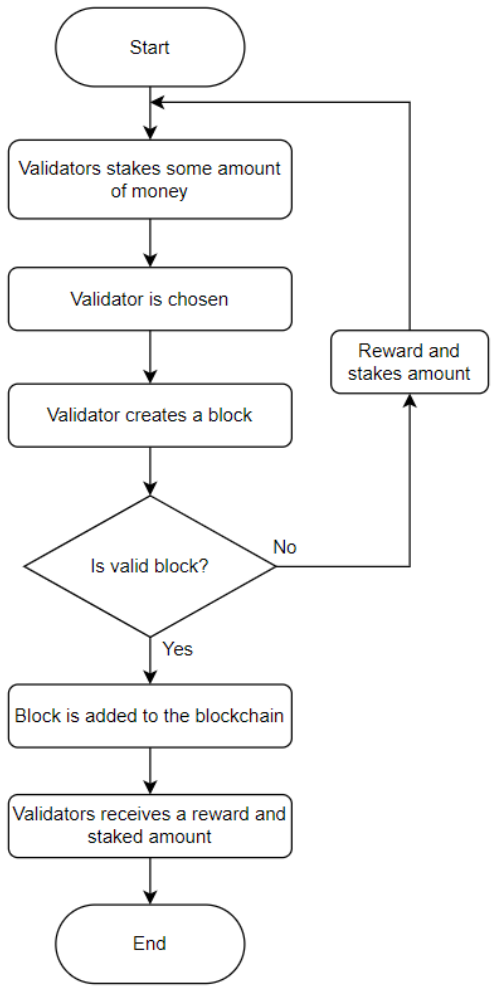

FIGURE 12. Flowchart of PoS Algorithm [73]

rency tokens to validate their dedication to the network. By engaging in token burning, participants prove that they have incurred a real cost and are prepared to sacrifice value to actively partake in the agreement process. The quantity of tokens burned often plays a significant role in determining the next block creator. The primary goal of PoB is to establish a just and decentralized approach to block creation, where participants are rewarded based on their commitment to contributing to the network, rather than relying on computational effort or ownership of stake [76].

d: Proof of Capacity (PoC):

Proof of Capacity (PoC) serves as a consensus algorithm in select blockchain systems. It operates by assigning a substantial storage space called "plots" to participate in the consensus process. Miners engage in pre-computing numerous potential solutions, which are then stored within these plots. As the need arises for a new block creation, miners scan their plots for suitable solutions that meet the necessary criteria. PoC capitalizes on storage space holds value, favouring those with greater storage capacity with higher odds of successfully mining a new block. In comparison to Proof of Work (PoW), PoC is deemed energy-efficient as it relies on disk space over computational power.

e: Delegated Proof of Stake (DPoS):

Delegated Proof of Stake (DPoS) represents a consensus algorithm employed in certain blockchain networks. Within the DPoS framework, a limited number of trusted delegates are designated to validate transactions and generate new blocks. These delegates are selected through a voting process, where token holders in the network cast their votes for their preferred delegates. The chosen delegates take turns in producing blocks and possess the authority to approve transactions. The primary objective behind DPoS is to enhance block confirmation speed and scalability compared to other consensus algorithms. However, it does introduce a degree of centralization by relying on voting for delegate selection, which can lead to power concentration among a few entities. Fig. 13, represents

a general scenario in DPoS.

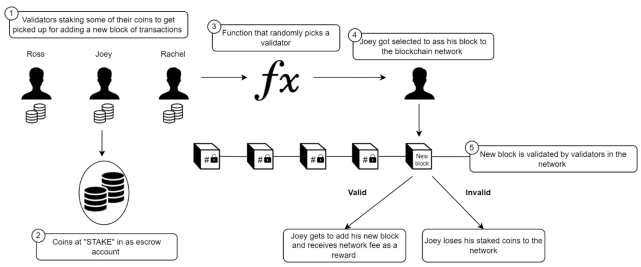

FIGURE 13. **A general scenario in DPoS Algorithm**

f: Proof of Authority (PoA):

Proof of Authority (PoA) is a consensus algorithm widely used in blockchain networks, notably in permissioned or private environments. In PoA, a defined group of validators or authorities are entrusted with the responsibility of validating transactions and generating new blocks. These validators go through prior selection based on their reputation, identity, or stake within the network. PoA boasts rapid block creation and impressive throughput capabilities. However, it comes at the cost of reduced decentralization since the authorities assume central control over decision-making processes. Fig. 14, represents the flowchart of proof-of-authority.

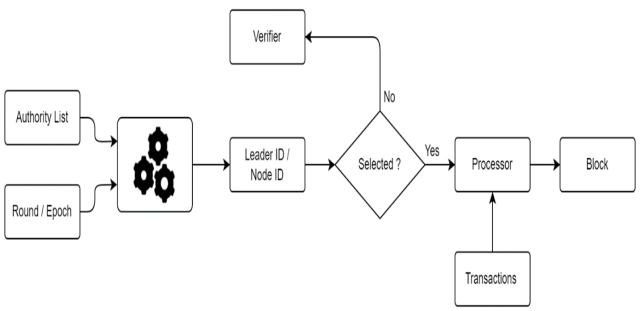

FIGURE 14. **Flowchart of PoA Algorithm [77]**

g: Proof of Importance (PoI):

Consensus algorithms are processes that help a group make decisions collectively. In this process, individuals within the group come together to create and support a decision that benefits everyone involved. It's not just about accepting the decision with the most votes but rather opting for the one that brings the most advantages to all participants in the network. Since Blockchain emerged along with Proof of Work (PoW) consensus mechanisms for verifying new nodes or transactions on the blockchain, numerous other consensus mechanisms have been introduced as well. One such mechanism is Proof of Importance (PoI), which is based on Byzantine Fault Tolerance. Fig. 15, represents the flowchart of PoI.

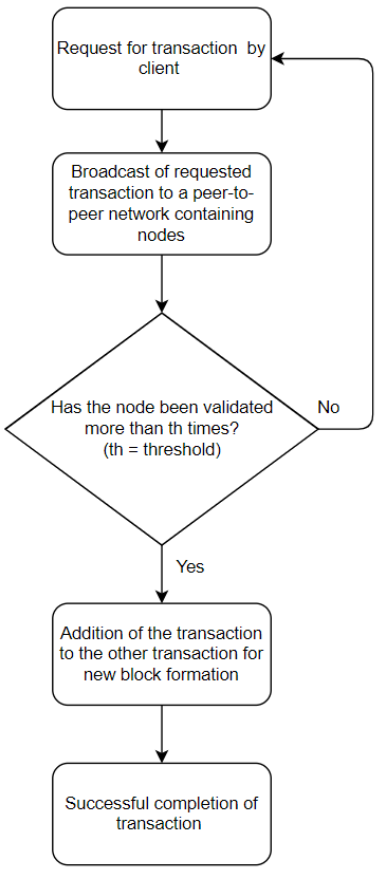

FIGURE 15. **Flowchart of PoI Algorithm [83]**

2) Voting-Based Consensus Algorithm

Voting-based consensus algorithms represent a type of mechanism widely used in blockchain networks. They enable participants to collectively agree upon the validity and order of transactions through a democratic voting process. Each participant holds an individual voting right, and the outcome is determined by the majority consensus, ensuring fair decision-making within the network.

a: Delegated Byzantine Fault Tolerance (dBFT):

Delegated Byzantine Fault Tolerance (dBFT) is a consensus algorithm widely used in blockchain networks. Its primary purpose is to introduce a voting mechanism that selects a limited number of trusted nodes, commonly referred to as delegates or witnesses. These chosen individuals possess the responsibility of validating transactions and creating new blocks [78]. By reducing the participation of numerous nodes in the consensus process, dBFT aims to achieve faster transaction confirmation times. However, it's important to note that this approach introduces an element of centralization due to the reliance on voting for delegate selection. Consequently, there is a potential risk of power concentration within only a few entities. Fig. 16, represents the flowchart of dBFT.

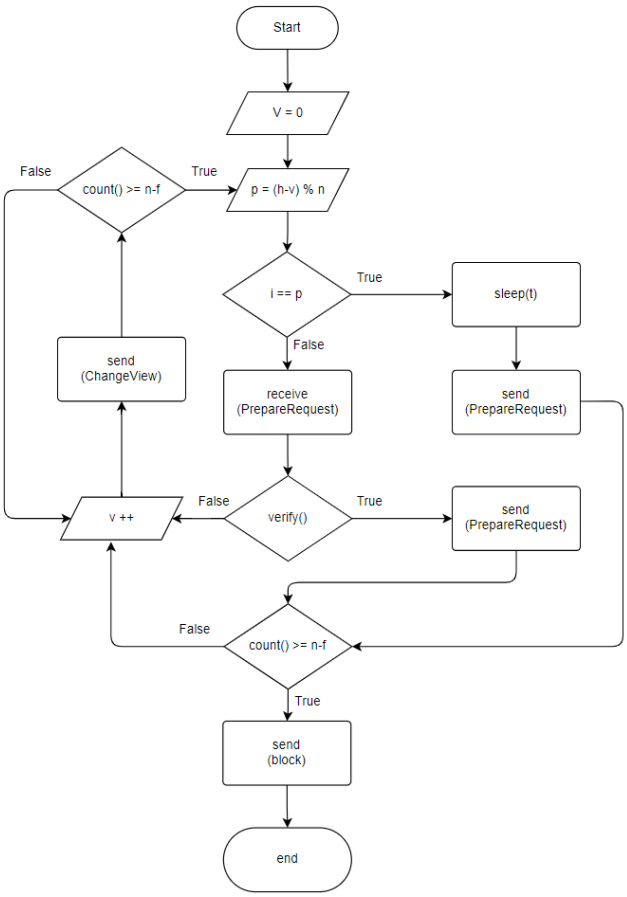

FIGURE 16. Flowchart of dBFT Algorithm [78]

b: Tendermint:

Tendermint is a consensus algorithm widely used in blockchain systems. It's primary goal is to achie- ve both high performance and Byzantine fault tolerance. The algorithm itself is built upon the Practical Byzantine Fault Tolerance (PBFT) algorithm, specifically designed for both permissioned and public blockchain networks. The functioning of Tendermint involves a block proposal and voting process. In this process, a designated validator proposes a block while other validators vote on its validity [90]. To consider a block finalized and added to the blockchain, at least two-thirds of the validators must agree on it. This approach ensures consensus even in cases where malicious actors or network faults are present. By implementing this algorithm, Tendermint offers fast block confirmation times and facilitates high transaction throughput, ensuring trustworthiness in the face of potential challenges. Fig. 17, represents the flowchart of Tendermint.

c: The Proof of Eclipsed Time (PoET):

The Proof of Eclipsed Time (PoET) algorithm serves as a consensus mechanism in certain blockchain networks. Its purpose is to counteract centralized control and minimize energy consumption, as opposed to other consensus algorithms like

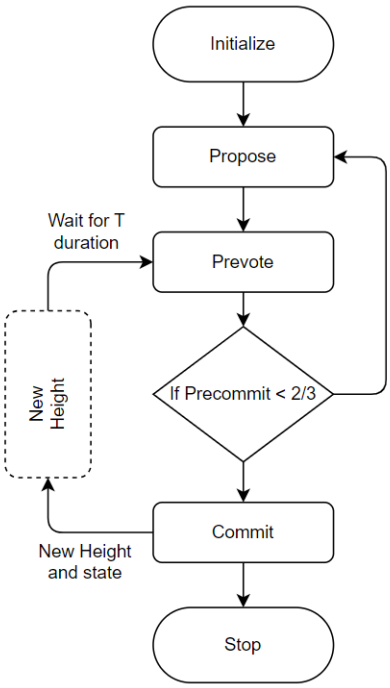

FIGURE 17. Flowchart of Tendermint Algorithm [90]

Proof of Work (PoW). Under PoET, every network participant patiently awaits a randomly assigned period before proposing a new block [73]. The participant who completes this waiting period first earns the privilege to create the subsequent block. This approach ensures fairness and prevents any individual from gaining an undue advantage. Ultimately, PoET strives for consensus using a more energy-efficient and decentralized approach. Fig. 18, represents the flowchart of PoET.

d: Practical Byzantine Fault Tolerance (PBFT):

Practical Byzantine Fault Tolerance (PBFT) widely used consensus algorithm blockchain systems. Its purpose is to enable a network of computers to collectively agree upon the order and validity of transactions, even when dealing with faulty or malicious nodes. To achieve consensus, PBFT relies on exchanges of messages and rounds. As long as more than two-thirds of the nodes operate honestly, PBFT guarantees agreement on the state of the blockchain [80]. This algorithm finds common utilization in permissioned blockchain networks where participants are both known and trusted. By leveraging PBFT, these networks enjoy fast transaction finality and increased throughput. Fig. 19, represents the flowchart of PBFT.

e: Proof of Voting (PoV):

In a PoV consensus process, validator nodes are staked to achieve consensus. Each node in the PoV system maintains the complete transaction history of the recorded blocks constituting the blockchain. To identify user accounts, either a

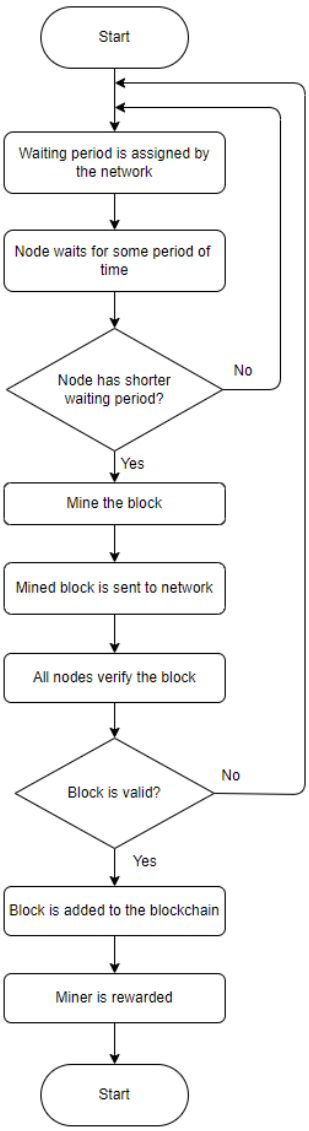

FIGURE 18. **Flowchart of PoET Algorithm [73]**

user's public key or the native token of the node can be utilized. This implies that your crypto wallet is traceable, and anyone possessing the public key can view your token balance. Furthermore, users can stake their crypto tokens in validator nodes. The number of nodes associated with a validator depends on how many tokens are staked within each validator.

f: Proof of Trust (PoT):

Proof of Trust (PoT) holds limited recognition as a consensus algorithm within the blockchain space. It appears to be a lesser-known or hypothetical approach, lacking substantial information or widespread adoption. Notably, the ever-evolving nature of blockchain research and development leaves room for innovative methodologies like PoT, which may not yet enjoy extensive acknowledgement [81]. Fig. 20,

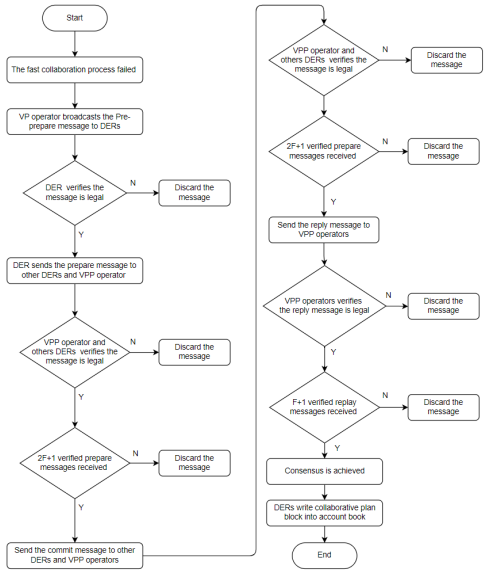

FIGURE 19. **Flowchart of PBFT Algorithm [80]**

represents the flowchart of PoT.

g: RAFT:

Raft, a distributed system, operates with three distinct roles: leader, follower, and candidate. The leader assumes the crucial responsibility of central authority by handling client requests and ensuring network-wide consistency. Its actions are mirrored by followers while candidates partake in leader election when necessary. What sets Raft apart is its emphasis on comprehensibility, achieved through dividing the consensus problem into manageable components like leader election, log replication, and safety mechanisms. This approach simplifies implementation and maintenance compared to intricate consensus protocols and has resulted in widespread adoption in applications ranging from database systems to distributed storage platforms. Fig. 21, represent the flowchart of Raft.

3) Lottery Based Consensus Algorithm

The lottery-based consensus algorithm is employed in blockchain networks to achieve consensus among participants. Its goal is to select a leader or validator responsible for creating new blocks by employing a process similar to a lottery. In this algorithm, nodes within the network compete against each other in order to become the block creator. This competition can be based on performing computational work or through random selection.

a: Proof of Location (PoL):

Proof of Location (PoL) is a consensus algorithm utilized in certain blockchain systems to authenticate and validate the actual geographic location of participants. It guarantees that nodes or devices indeed exist at the claimed specific location.

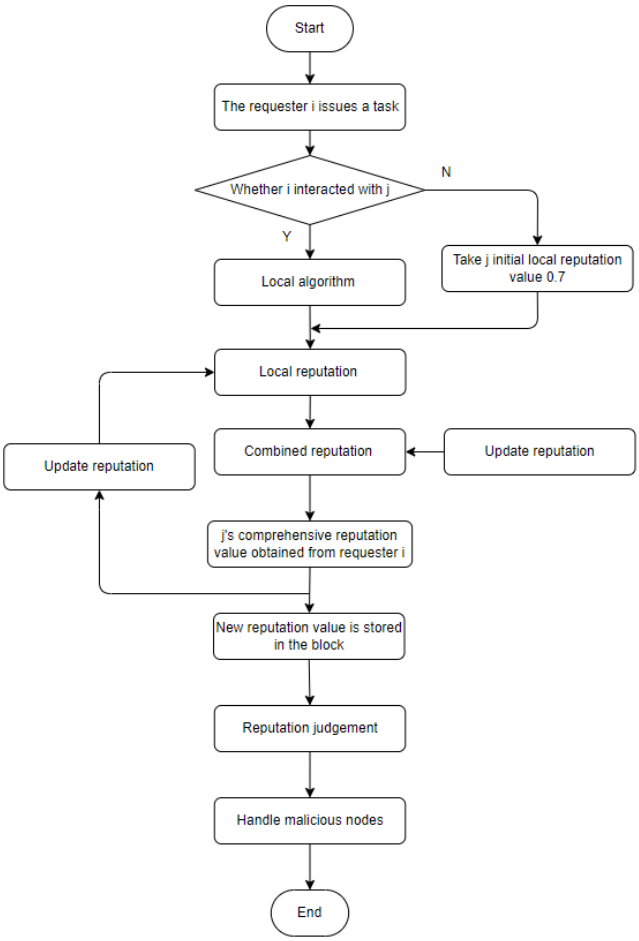

FIGURE 20. **Flowchart of PoT Algorithm [81]**

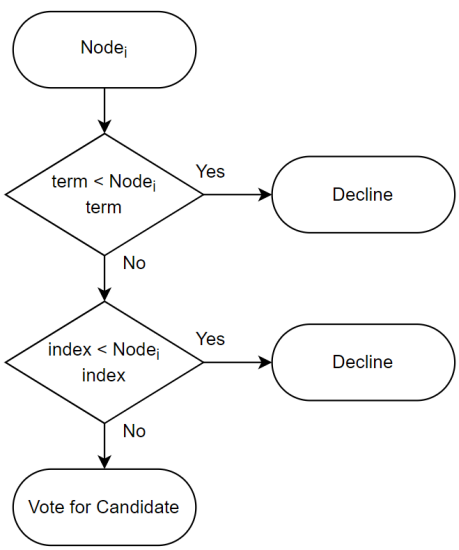

FIGURE 21. **Flowchart of Raft Algorithm [82]**

PoL relies on various techniques such as GPS, Wi-Fi signals, or cell tower triangulation to collect precise location data. The participants, by furnishing proof of their whereabouts, can actively contribute to the consensus process and receive rewards within the blockchain network. The primary objective of PoL is to enable location-based services and applications while upholding the integrity and trustworthiness of the entire blockchain system.

b: Randomized Proof of Work (RPoW):

Randomise Proof of Work (RPoW) is a consensus algorithm designed to offer an alternative to the traditional Proof of Work (PoW) algorithm utilized in blockchain systems. RPoW introduces randomness into the PoW process, enhancing its resistance against specific attack types. By incorporating this element of chance, RPoW effectively prevents miners from gaining an unfair advantage based solely on computational power. This innovative algorithm aims to promote a more equitable distribution of mining rewards while bolstering the security and decentralization of the blockchain network.

c: Proof-of-Randomness (PoR):

In a Proof-of-Resources (POR) based blockchain network, the task of creating new blocks and validating transactions is assigned to validators chosen at random. This selection process relies on a random number generator, making it exceedingly difficult for any individual node to manipulate or corrupt the system. As a result, POR demonstrates remarkable resistance against attacks such as 51 present attacks, which pose threats to other consensus mechanisms like Proof-of-Work (PoW) and Proof-of-Stake (PoS). Essentially, POR beautifully combines randomness and security, offering an extraordinarily robust and efficient consensus mechanism for blockchain networks.

d: Proof-of-Probability (PoP):

In the PoP approach, individual nodes adopt a unique strategy. They store both the authentic encrypte- d hash and a series of counterfeit hashes. The initial node that successfully decrypts the genuine hash is responsible for generating a new block. To manage intense computational competition, there is an introduction of a waiting period between consecutive hash decryptions [84]. Importantly, the PoP mechanism also addresses potential centralization concerns that may arise from validators holding substantial stakes in the network. This presents an optimistic way to enhance blockchain consensus methods. Fig. 22, represents the flowchart of PoP.

e: Ripple:

The ripple consensus algorithm operates through a network of trusted validators. These validators are selected by the Ripple company and other participants. They work together to confirm transactions and maintain the integrity of the Ripple network. The algorithm emphasizes decentralization

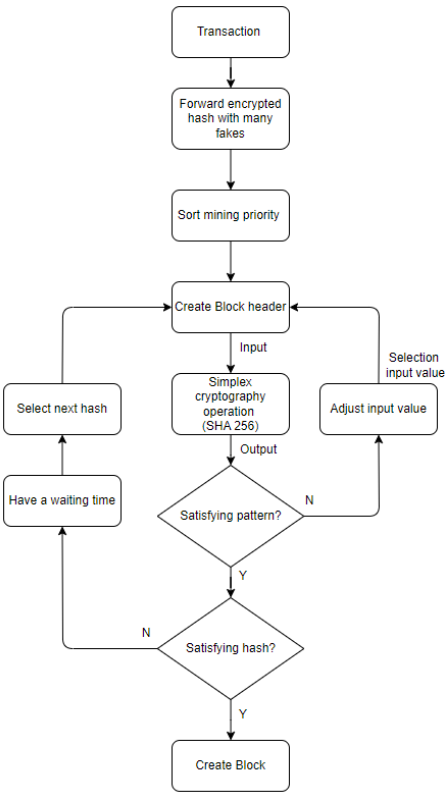

FIGURE 22. Flowchart of PoP Algorithm [84]

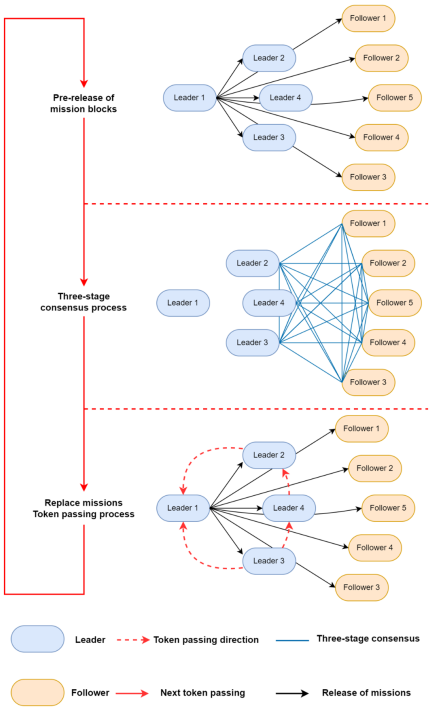

FIGURE 23. Flowchart of Ripple Algorithm [85]

and relies on a predetermined set of validators. This design enables the protocol to process transactions quickly and efficiently. Ripple native cryptocurrency, XRP serves as a bridge currency within the Ripple network. It facilitates seamless and rapid conversion between different fiat currencies, making it a preferred choice for financial institutions and payment service providers looking to streamline cross-border payments and improve liquidity. Fig. 23, represents the flowchart of Ripple.

D. STRUCTURAL PROPERTIES OF A CONSENSUS MECHANISM

This table represents the structural properties of some blockchain networks. Breaking down structural properties of different blockchains. Fig. 24, represents the Structural properties of a blockchain.

Based on Fig. 24, some blockchain network is broken down and the finding and analysis is given in Table 11.

Table 11, presents a comprehensive overview of different blockchain platforms. It highlights their underlying mechanisms, consensus algorithms, node types, structural characteristics, and configuration traits. Through this comparison, we gain insight into the various decentralized approaches these platforms employ to achieve consensus and manage network structures.

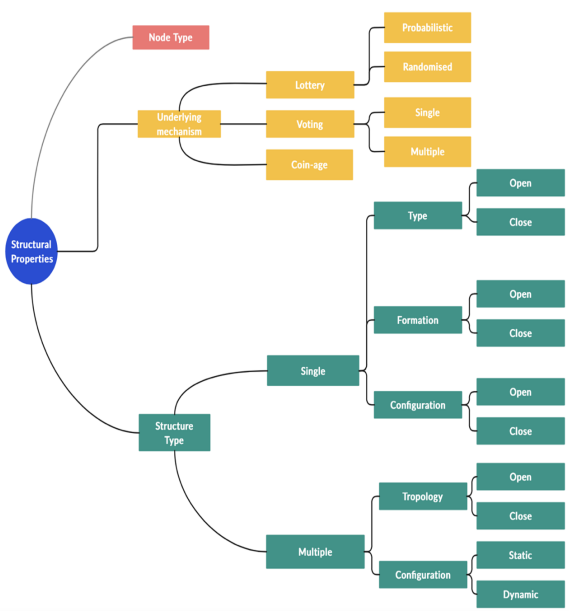

FIGURE 24. Structural properties of a blockchain

TABLE 11. Breaking Down Different Blockchain's Consensus Algorithm

Blockchain	Consensus Algorithm	Algorithm Type	Node Type	Structural Type	Formation	Configuration
Ethereum (ETH)	PoS	Proof-based	Miners/Validators	Open	Dynamic	Explicit
Solana (SOL)	Proof-of-History + Proof-of-Stake	Hybrid	Validators	Open	Dynamic	Explicit
Avalanche (AVAX)	Avalanche Consensus	Voting-based	Validators	Open	Dynamic	Explicit
Astar	Proof-of-Authority	-	-	-	-	-
Cardano (ADA)	Ouroboros Proof-of-Stake	Proof-based	Validators	Open	Dynamic	Explicit
Polkadot (DOT)	Nominated Proof-of-Stake (NPOS)	Voting-based	Validators	Open	Dynamic	Explicit
NEAR Protocol	Proof-of-Stake (with Doomslug)	Proof-based	Validators	Open	Dynamic	Explicit
Hyperledger Fabric	Pluggable Consensus	-	Peers/Orderers	Multiple	Static	Open
Corda R3	Notary Service	Voting-based	Notaries/Nodes	Multiple	Static	Close
Quorum (ETH-based)	Various (e.g., Istanbul BFT)	Voting-based	Validators	Multiple	Static	Open
Binance Coin (BNB)	Tendermint BFT	Voting-based	Validators	Multiple	Static	Open
Algorand (ALGO)	Pure Proof-of-Stake	Proof-based	Participants	Multiple	Static	Open
EOSIO (EOS)	Delegated Proof-of-Stake (DPoS)	Voting-based	Block Producers	Multiple	Static	Open

E. CONSENSUS ALGORITHM COMPARISON

Table 12, presents a comprehensive analysis of consensus algorithms used in blockchain technology. It highlights their programming languages, applications, extensibility, computing and storage costs, block proposers, proposer selection/mining methods, and selection basis. In table 12, "N/F" represents that data "Not Found".

Table 13, presents a comprehensive analysis of consensus algorithms used in blockchain technology. It highlights their block validation mechanisms, blockchain finalisation processes, execution environments, efficiency levels, message overheads, authenticity levels, and control attributes. In table 13, "N/F" represents data "Not Found" and "N/A" represents "Not Applicable".

Table 14, provides a comprehensive overview of different blockchain consensus algorithms. It compares different blockchains, taking into account factors such as node identity, energy efficiency, adversary power tolerance, fork susceptibility, fault tolerance, and more. This comparative analysis greatly contributes to a better understanding of the strengths and weaknesses inherent in each algorithm. Consequently, it aids individuals in making informed decisions when selecting specific blockchain requirements and objectives. In table 14, "N/F" represents data "Not Found" and "N/A" represents "Not Applicable".

Table 15, provides an overview of different consensus algorithms used in decentralized systems. Each row represents a specific algorithm, and the columns highlight characteristics such as transaction fees, block rewards, scalability, throughput, and more. While some cells lack data, the table offers valuable insights into the diversity and complexity of consensus mechanisms. These insights help in decision-making for blockchain implementations. In table 15, "N/F" represents data "Not Found" and "N/A" represents "Not Applicable".

F. SIMULATION AND RESULT ANALYSIS

Fig. 25, represents the comparison of average throughputs of Pow, Pos, DPoS and Raft, where the X-axis represents several transactions and the Y-axis represents average throughputs (TPS).

Fig. 26, represents the Average latency with varying numbers of nodes of Pow, Pos, DPoS and Raft, where the X-Axis represents the number of nodes and the Y-Axis represents the average latency in seconds.

Fig. 27, represents Average latency with varying numbers of transactions of Pow, Pos, DPoS and Raft, where the X-axis represents the number of transactions and the Y-axis represents average latency in seconds.

TABLE 12. Consensus Algorithm Comparison - Implementation and Application - 1

Algorithm	Language	Application	Extensible	Computing Cost	Storage Cost	Block Proposer	Mining/Selection	Selection Basis
PoW	Solidity, Go	Cryptocurrency, General Application	No	High	High	Miner	Computational Power	Computational power
PoS	Solidity, Java, C++	Cryptocurrency, Michaleson Application	No	Medium	High	Validator	Stake Owned	Staking Amount
DPoS	JS, Python, Ruby	Cryptocurrency, General Application	Yes	Medium	High	Delegates	Stake Owned	Staking Amount
PoA	Solidity, Java, Python	Cryptocurrency	Yes	Medium	N/F	Verified Authorities	Solving Difficulty Hash	Authorised Organization
PoSpace	Java	Cryptocurrency, General Application	Yes	Low	High	Leader	Mathematical Operation	Hard Disk Drive Capacity
PoAu	Go	Cryptocurrency, General Application	No	Low	High	Primary Node	Leader Chosen by Time-based Event	Wait time
PoET	Python, JS, Go, C++, Rust	General Application	No	Low	High	Leader	Primary Node Selected by Voting	Wait time
PBFT	Java, Go, Node	General Application	No	Low	High	Primary Node Selected by Voting	Mathematical Operation	N/F
DBFT	C#, Python, .NET, Java, C++, C, Go, Kotlin, JS	Cryptocurrency	No	Low	Low	N/F	N/F	N/F
Ripple	Java, C++, NodeJs	Ripple(XRP)	No	Low	Low	N/F	Vote-Based Mining	N/F
PoI	Java	New Economy Movement	Yes	Low	High	One of the Node with Qualified Importance Score	High Priority Importance Score	
PoL	Go, C++, Solidity, Serpent, LLL	IoT and Location Based Services	Yes	Low	N/F	N/F	Luck Value	N/F
Raft	Go, C++, Java	Cloud, Storage Service , Distributed Key-Value Stores	No	Low	High	N/F	Randomized Timers	N/F
PoP	Go, C++, Solidity, Serpent, LLL	Cryptocurrency, General Application	No	Low	N/F	N/F	N/F	N/F
PoB	Go, C++, Solidity, Serpent, LLL	Cryptocurrency and General Application	No	Low	N/F	One of the Node burning Coins	Vote mining	Burn coins
PoC	N/F	Outsource Storage	No	Low	Very High	Closest Solution Matching the Challange	Vote-Based Mining	Hard Disk Capacity
PoR	Go, C++, Solidity	Outsource Storage	No	Low	N/F	Closest Solution	N/F	N/F

VIII. SMART CONTRACT LAYER

The contract layer implements smart contracts, a set of digitally set commitments that are unmodifiable once deployed and executed immediately once triggered [86]. The contract layer pertains to the constituent within blockchain systems that facilitates the generation and implementation of smart contracts. Smart contracts are autonomous programs that operate on a distributed ledger technology known as the

TABLE 13. Consensus Algorithm Comparison - Implementation and Application - 2

Consensus Algorithm	Block Validation	Blockchain Finalization	Execution	Efficiency	Msg Overhead	Authenticity	Control
PoW	Other nodes	Majority acceptance	Native, EVM	Low	O(1)	High	Low
PoS	Other staking nodes	Sufficient approval	Native	High	O(1)	High	Medium
DPoS	Other delegators	Sufficient approval	Native	High	O(1) - O(n)	High	High
PoActivity	Other authorities	Sufficient approval	EVM, Dockers	High	O(1)	High	Medium
PoSpace	N/F	Sufficient approval	Native, EVM	High	O(1)	High	Low
PoAu	N/F	Majority acceptance	EVM, Dockers	High	$O(n^2)$	High	High
PoET	Other nodes	Majority acceptance	Native	High	O(1)	High	Low
PBFT	Other nodes	Sufficient approval	Dockers	High	$O(n^2)$	Low	Low
DBFT	Other delegated nodes	Sufficient approval	N/A (NEO-specific)	High	O(1)	High	High
Ripple	N/F	Sufficient approval	Node.js, NPM	High	O(1)	High	High
PoI	Other nodes qualified	Sufficient approval	NEM	High	O(1)	High	Low
PoL	N/F	Majority acceptance	N/F	High	N/A	Low	High
Raft	Other Nodes	Sufficient approval	N/F	N/A	O(1)	High	Low
PoP	N/F	Majority acceptance	Native, EVM	High	N/A	High	Low
PoB	Other nodes	Majority acceptance	Native, EVM	N/A	N/A	High	Low
PoC	Other nodes	Majority acceptance	N/F	N/A	N/A	High	High
PoR	N/F	Majority acceptance	Native, EVM	High	N/A	High	Low

TABLE 14. Consensus Algorithm Comparison - Security and Fault Tolerance

Consensus Algorithm	Blockchain Type	Node Identity	Energy Saving	Tolerance Power of Adversary	Prone to Forks	Crash Fault Tolerance	Byzantine Fault Tolerance	51% Attack
PoW	Permissionless	Public	No	<25% computing power	Yes	50%	50%	Yes
PoS	Both	Public	PARTIAL	<51% stake	Yes	50%	50%	No
DPoS	Both	Public	PARTIAL	<51% validators	Yes	50%	50%	No
PoActivity	Both	N/F	PARTIAL	50% online stake	Yes	N/F	N/F	No
PoSpace	Both	Public	No	25% computing power	Yes	N/F	N/F	No
PoAuthority	Both	Public	PARTIAL	<51% online stake	Yes	N/F	N/F	Yes
PoET	Both	Public	Yes	Node compromise	Yes	33%	33%	N/F
PBFT	Permissioned	Private	Yes	<33.3% Faulty replicas	No	33%	33%	No
DBFT	Permissioned	Both	PARTIAL	<33% fault replicas	No	N/F	N/F	No
Ripple	Permissioned	Private	Yes	<20% Faulty nodes	N/F	N/F	N/A	No
PoI	Both	Both	Yes	<50% importance	Yes	N/F	N/F	No
PoL	Both	Public	Yes	<50% processing power	N/F	N/F	N/F	No
Raft	Both	Public	Yes	<50% crash fault	No	50%	N/A	N/F
PoP	Both	Private	No	25% computing power	N/F	N/F	N/F	No
PoB	N/F	Public	No	25% computing power	N/F	N/F	N/F	N/F
PoC	N/F	Public	No	<50% storage space	Yes	N/F	N/F	No
PoR	N/F	Public	Yes	25% computing power	N/F	50%	N/F	N/A

TABLE 15. Consensus Algorithm Comparison - Scalability and Performance

Consensus Algorithm	Transaction Fee	Block Reward	Scalability	Throughput (TPS)	Latency	Verification Speed	Block Creation	Processing Power
PoW	Yes, for the first miner	Yes, for the first miner	Low	<100	>100s	>100s	Medium	High
PoS	Yes, for the miner	No	Medium	<1000	<100s	<100s	Medium	Low
DPoS	Yes, for all the witnesses	Yes, for elected witnesses	High	<1000	<100s	<100s	High	Low
PoActivity	Yes, for miner and lucky stakeholders	No	Low	N/F	N/F	N/F	Low	Medium
PoSpace	N/F	N/F	Medium	N/F	N/F	N/F	Medium	Low
PoAuthority	No	No	High	<2000	<10s	<10s	High	Low
PoET	Yes	No	High	N/F	N/F	N/A	High	N/F
PBFT	N/F	N/F	High	N/F	N/F	N/F	High	Medium
DBFT	N/A	N/A	High	N/F	N/F	N/F	High	Medium
Ripple	Yes, for transaction partners	Yes, for winners	High	>10K	>15s	N/F	High	Low
PoI	N/F	N/F	High	N/F	N/F	N/F	High	N/F
PoL	N/F	N/F	High	N/F	N/F	N/F	High	N/F
Raft	Yes	No	Low	N/F	<50%	<50%	Low	N/F
PoP	N/F	N/F	High	N/F	N/F	N/F	High	N/F
PoB	N/F	N/F	Low	N/F	N/F	N/F	Low	N/F
PoC	N/F	N/F	High	N/F	N/F	N/F	High	N/F
PoR	N/F	Yes	Medium	N/F	25%	N/F	High	N/F

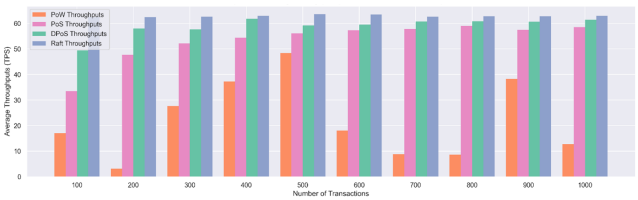

FIGURE 25. Comparison of average throughputs (PoW vs. PoS vs. DPoS vs. Raft)

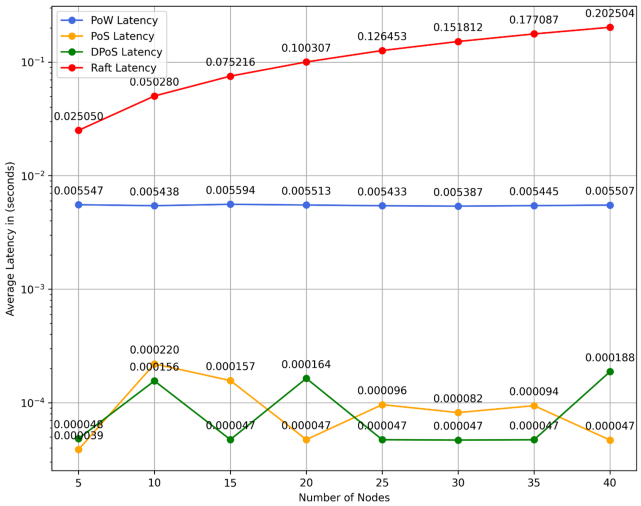

FIGURE 26. Average latency with varying number of nodes (Log Scale)

blockchain, facilitating the execution and enforcement of contractual obligations between multiple parties in a decentralized manner, thereby eliminating the requirement for a central governing entity.

The contract layer is positioned hierarchically above the consensus layer, responsible for verifying transactions, and the network layer, responsible for distributing transactions and blocks. Once transactions undergo validation through the consensus protocol, the contract layer examines whether these transactions prompt the execution of any smart contracts. The contract execution environment is commonly kept separate from the rest of the blockchain network. This separation ensures that smart contracts can operate securely without causing any disruptions to other components.

A. SMART CONTRACT

A smart contract is a contract that can run itself. The terms and conditions that the buyer and seller decide on are directly written into lines of code. A decentralized blockchain network has the code and the deals that go with it. This means that the deal is automatically carried out when certain conditions are met. This makes it possible to set up deals that are both safe

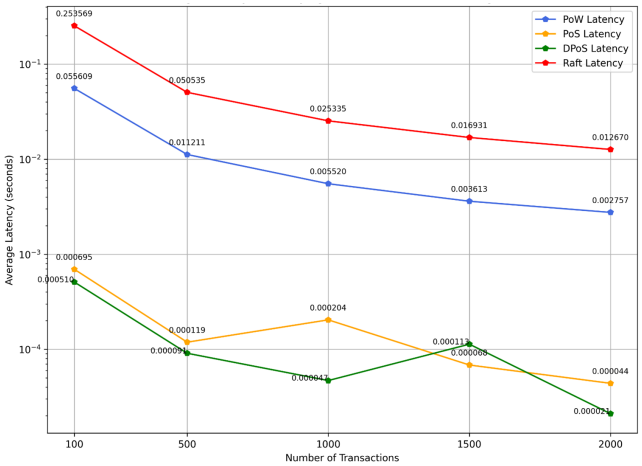

FIGURE 27. **Average latency with varying number of transaction (Log Scale)**

and don't require trust, getting rid of the need for middlemen. Smart contracts are the core of blockchain 2.0 [87].

Smart contracts facilitate the execution of reliable transactions and contractual arrangements between diverse and unidentified entities, obviating the necessity for a centralized governing body, legal framework, or an external mechanism for enforcement [88]. These systems possess the capability to render transactions traceable, transparent, and irreversible. The recognition of the distinctive obstacles encountered in smart contract programming has served as a catalyst for developers to devise domain-specific languages, exemplified by Solidity [89], with the aim of facilitating the development process.

Programming languages like Solidity, which is used on the Ethereum blockchain [89], are often used to make smart contracts. The code sets the rules and requirements for the deal, and once it's on the blockchain, it can't be changed. The trait of immutability makes sure that the terms of the contract can't be changed without the agreement of all parties.

Once the smart contract code is written then it is compiled into bytecode, a machine-readable form of the code. This bytecode is then executed on the blockchain's virtual machine, a crucial component of the blockchain protocol responsible for handling smart contract execution. The virtual machine ensures that the smart contracts' actions are transparently and securely executed across the decentralized network. This vital interplay between programming languages, bytecode, and the virtual machine paves the way for efficient and automated agreement enforcement within blockchain ecosystems, making it an integral aspect of this transformative technology.

B. THE DAWN OF SMART CONTRACT

The concept of smart contracts was first proposed by Nick Szabo in 1994, long before the advent of blockchain [90]. Smart contracts only became a reality with the introduction of Ethereum in 2014, the contract layer of Ethereum, which is integrated into the Ethereum Virtual Machine (EVM), permits the execution of Turing-complete scripts on the blockchain [91]. These scripts have the ability to solve any computational problem with sufficient resources. This functionality enables the development and enforcement of smart contracts. Kosba et al. (2016) introduced Hawk, a framework for the development of smart contracts that aims to safeguard user privacy [92]. In the year 2018, Kalra et al [93]. introduced ZEUS, an innovative framework for the analysis of smart contract security. The proposed framework exhibits a substantial enhancement in the efficiency of security analysis time when contrasted with preceding methodologies. In the year 2020, Zheng and et al. conducted a study wherein they undertook the task of classifying smart contract applications. This was achieved through a comprehensive comparison and analysis of various smart contract platforms that are commonly employed in the field. The findings of this research were documented in their publication, denoted as a reference [107]. In 2022, Vittorio et al. identified common misconceptions and provided guidelines for proper smart contract management, serving as a valuable resource for future standards [108].

C. ADVANTAGES OF SMART CONTRACT

1) Savings: Smart contracts offer cost-saving benefits in various ways. Smart contracts eliminate intermediaries and reduce the need for legal fees [96]. Automation enhances operational efficiency by accelerating the execution and settlement processes, resulting in reduced labour costs. Ultimately, decreased human involvement leads to a reduction in errors, disagreements, and litigation expenses.
2) Security: Smart contracts have been found to enhance security in various aspects. Once the data has been appended to the blockchain, it becomes immutable and resistant to any alterations. This reduces the probability of unauthorized alteration. Smart contracts operate according to their predetermined design, thereby preventing any unauthorized modifications or disruptions during their execution. Furthermore, due to the transparent nature of the public ledger, it enables the tracking and verification of events. This fosters system accountability and cultivates trust in its operations.
3) Confidence and Openness: The transparency of smart contracts promotes confidence and openness. The utilization of a public blockchain enables individuals to verify and conduct audits on code. The ability of participants to verify ledger transactions in the absence of a central authority is a notable characteristic. The implementation of public verification mechanisms enhances the level of trust in a system. In the absence of a third-party intermediary, individuals can place their trust in the outcomes resulting in promoting openness.
4) Accuracy, Efficiency and Rapidity: Smart contracts automate contract execution, improving accuracy and efficiency. The utilization of these methods serves to mitigate the occurrence of errors in interpretation and

mitigate the potential for conflicts. Once the necessary criteria have been fulfilled, the resulting outcomes become definitive and unalterable. The autonomous execution of tasks enhances the efficiency of transactions. There are no delays in approval or any other obstacles. The act of execution can lead to prompt resolution. The speed and efficiency of smart contracts are advantageous.

D. CONTRACT LANGUAGE

The responsibility for establishing the guidelines and structure for executing actions within the blockchain lies with the smart contract code. The meticulous composition of the code is of utmost significance to ensure the absence of any vulnerabilities or deficiencies that may jeopardize financial assets. Typically, the code utilized for smart contracts on the blockchain is predominantly composed in a specialized computer language specifically designed for this purpose, such as Solidity, which is primarily employed for the Ethereum platform. In order to achieve proficiency in these languages, a comprehensive understanding of cryptography and information management is imperative. Solidity, Vyper, and Bamboo are the most common computer languages used to make smart contracts. As a widely used computer language, Solidity has become the most popular choice [17] for making smart contracts on the Ethereum blockchain. It's important to remember that Solidity was made on purpose to meet the specific needs of Ethereum's smart contract environment. The programming language is close to JavaScript in terms of syntax [97], and it follows the static typing rules. Vyper is a programming language that was made just for Ethereum. It puts a lot of stress on security and simplicity. Its main goal is to make smart contracts easier to audit, which will make it easier to look at contracts. When compared directly to the programming language Solidity, Vyper looks a lot like Python and has strong typing powers [113]. Bamboo is a new language for smart contracts that has just come out in the field. Bamboo is designed to be highly readable and allow both on-chain and off-chain contract logic. Solidity is currently the most popular programming language for smart contracts. This is because it has a long history of development and a large group of developers. But Vyper and Bamboo, which are relatively new options, were made with the main goal of making smart contracts safer and easier to use.

1) Solidity

Solidity is the most common programming language used in the Ethereum Virtual Machine (EVM). It has become very popular with developers. Also, it has been widely used in a lot of blockchains that are compatible with EVM. Solidity is a high-level computer language that is also Turing-complete. It allows developers to speed up the process of writing code by hiding many low-level details. Solidity's syntax is very similar to that of Javascript, which is a highly regarded programming language that is very famous all over the world. Because of this, Solidity is the best choice for coders who want to move smoothly into the world of Web3. Solidity is good for new smart contract writers because it has built-in safety features that reduce the risk of making mistakes that could be very expensive [97]. Solidity is the most used computer language in the Web3 world, which gives its developers a lot of benefits. These things include a wide range of libraries and tools, full documentation, and more help for developers in online groups. Solidity's syntax may be hard to understand for developers who aren't used to object-oriented programming, and its ability to overload functions could make code harder to understand. In the world of Solidity development, it's important to note that there are some issues that writers might not be used to. One of these is that the language can't handle decimal numbers by default [97]. Expert Web3 writers often notice that programming in Solidity uses more gas than programming in lower-level languages. Solidity has gotten a lot of attention because it is the most efficient and competent language for making smart contracts. But it's not possible for security to be broken [99]. Because of this, many researchers have tried to make Solidity smart contracts safer by adding intermediate-level languages or using tools for analyzing smart contracts.

2) Vyper

Vyper has emerged as a prominent programming language within the Web3 ecosystem, second only to Solidity in terms of its widespread adoption for EVM-compatible blockchains [97]. The programming language in question is characterized by its focus on contracts, adherence to Pythonic principles, implementation of strong typing, a compact compiler codebase, and the generation of efficient bytecode. The Vyper programming language exhibits a visual and experiential resemblance to Python [98], rendering it an optimal choice for Python developers embarking on their Web3 development expedition. The Vyper programming language, along with its accompanying compiler implementation, offers a straightforward syntax and structure. This characteristic contributes to enhanced code readability and auditability, thereby facilitating the development of secure smart contracts for practical deployment by developers [98]. Vyper, being the second most widely embraced smart contract programming language, shares numerous tooling and resources with Solidity. The current state of Vyper is characterized by a relative deficiency in terms of extensive community backing, which is in contrast to the well-established support enjoyed by Solidity. Furthermore, Vyper is currently lacking certain tools that are specifically designed for Solidity, thereby limiting its functionality and usability in comparison. Vyper, in its current state, exhibits a notable absence of modifiers, class inheritance, and recursive calls, rendering it a programming language that falls short of being Turing-complete. The absence of several features can be attributed to intentional design decisions aimed at optimizing contract security and auditability. However, this necessitates additional efforts from developers to circumvent these constraints. As a nascent programming language, Vyper is currently in the developmental phase,

thereby exhibiting a relatively limited set of functionalities.

3) Rust

Rust has gained significant popularity as a programming language for smart contracts on various blockchains that are not compatible with the Ethereum Virtual Machine (EVM), including Solana [100]. It is worth noting that Rust distinguishes itself from other languages mentioned in this context by its versatility beyond Web3 development. The Rust programming language exhibits notable qualities such as efficiency, security, and a reduction in superfluous elements. Specifically, Rust data structures are characterized by their compactness, rendering them well-suited for accommodating the space limitations inherent in blockchain systems [101]. However, it should be noted that a considerable number of blockchains currently lack comprehensive tooling or substantial support for the Rust programming language.

4) Go

Go is a secure, efficient language developed by Google that compiles smart contracts into Ethereum-compatible bytecode. Go's built-in safety features help avoid vulnerabilities like reentrancy bugs. The language provides a robust framework for writing complex decentralized apps and smart contract logic.

5) Haskell

Haskell's functional programming model makes it possible to create secure, unchangeable smart contracts that stay out of trouble. Using advanced Haskell capabilities, one may write clear, tested reasoning that can run on Ethereum's virtual machine. The safety and rigor of Haskell make it perfect for safe decentralized applications.

6) Michelson

Michelson is a strongly typed, functional language designed specifically for writing secure smart contracts on Tezos. Michelson's formal verification features mathematically prove properties of smart contracts, enhancing reliability. The language provides built-in primitives ideal for blockchain actions like token transfers and delegation

7) TinyGo

TinyGo is a lightweight, compiled Go language variant optimized for microcontrollers and blockchain runtimes. It compiles smart contracts into efficient WebAssembly bytecode. TinyGo provides a familiar Go-style syntax while minimizing dependencies for writing secure, deterministic smart contracts.

8) Flint

Flint is a type-safe, functional language built specifically for writing robust smart contracts on the Cardano blockchain. Flint has a strong, static type system to prevent errors and vulnerabilities in code. The language compiles to optimized Plutus bytecode executable on the Cardano Virtual Machine.

9) Comparison of Different Smart Contract

In this paper, we have selected the programming languages that are commonly used for smart contract development on various blockchain platforms. The classification of languages can be broadly categorized into three distinct groups: statically typed languages, dynamically typed languages and hybrid languages. The evaluation of each language is based upon an evaluation of four fundamental criteria, security features and performance, community support, and Turing completeness.

10) Security Features and Performance

Smart contracts are an important component of blockchain technology, enabling decentralized, trustless, and automated execution of agreements. However, the security of these contracts is paramount, as they manage valuable assets and are executed on a decentralized network. A single vulnerability in a smart contract can result in significant financial losses or even compromise the entire network. Therefore, it is essential to use programming languages that provide robust security features. In this paper, we analyze various programming languages used for smart contract development, focusing on their security features. We evaluate the languages based on seven security features: garbage collection, inheritance, reentrancy protection, decentralized oracle support, memory safety, type safety, and immutability. Also, we assess the languages' support for secure coding practices. Our analysis provides a comprehensive comparison of the security features of various programming languages, enabling developers to make informed decisions when selecting a language for their smart contract projects.

1) Garbage Collection: Garbage collection in programming refers to automatically free up memory that is no longer being used by variables or objects in the smart contract code [102]. Smart contracts lacking garbage collection mechanisms are exposed to memory leaks, consequently making them prone to denial-of-service attacks.
2) Inheritance: Inheritance is an object-oriented programming concept that allows smart contracts to inherit attributes and methods from a parent smart contract. Smart contracts that inherit from other smart contracts can inherit their vulnerabilities.
3) Reentrancy Protection: The vulnerability of reentrancy refers to the ability of a smart contract to be invoked repeatedly in a recursive manner, hence creating a potential risk of depleting funds from the contract [103].
4) Decentralized Oracle Support: Decentralized oracles play a crucial role in blockchain systems by facilitating secure interactions between smart contracts and external data sources, enabling the execution of logic based on real-world occurrences.
5) Memory safety ensures programs cannot access arbitrary memory locations, preventing accidental/intentional data corruption.

6) Type safety ensures variables only permit valid operations based on their data types, catching many bugs during compilation itself.
7) Smart contracts that have not been developed utilizing secure coding practices may be exposed to a range of attacks. Some common coding practices that a smart language include tested frameworks and libraries and cryptographic libraries instead of custom code.

Table 16, provides a comparative overview of key software engineering attributes across prominent smart contract languages from an academic perspective. The attributes considered are garbage collection, inheritance, reentrancy protection, decentralized oracle support, memory safety, type safety and availability of secure coding practices. These properties relate to the correctness, security and robustness of smart contract systems.

11) Community Support

Blockchain technology has made it possible to make autonomous applications and smart contracts, which are pieces of code that run themselves on blockchain networks. However, the languages that run these systems are still in their early stages. For a programming language to be widely used by developers, the developer community must back it and be involved with it.

Smart contract languages need thriving open-source environments that are driven by collaborative development to grow and improve. By asking for feedback from the community, real-world needs can be added to protocol changes. Active developer forums and communication channels help developers find answers to technical problems as soon as they come up. Strong community involvement also makes smart contract languages safer by putting them through a lot of stress tests in a variety of situations. Public code checks and bug bounty programs give people an incentive to find vulnerabilities before they are put on the mainnet.

12) Turing Completeness

Turing completeness means that, in theory, the smart contract language can solve any problem that can be solved by a computer, as long as there is enough time and room to do so [104]. This makes it possible to put complex logic, workflows, and apps right on the blockchain. On the other hand, languages that aren't Turing complete, like Bitcoin Script, only have a few predefined processes that can be used for simple transactions.

But because they are more complicated, Turing-complete smart contracts are harder to check for security holes. Because smart contracts can't be changed and deal with valuable assets, bugs can be very expensive if they aren't found before they are used [104]. Formal testing of smart contracts that are Turing-complete is very hard.

Also, the theoretical stopping problem says that there is no general way to tell if a Turing-complete program will run forever or stop. This could keep network resources from being used for a long time. Gas costs reduce this worry in part by stopping processing if the gas limit is reached.

Table 17, table categorizes prominent blockchain platforms based on the smart contract languages they support and the key software attributes of those languages. The attributes compared are Turing- completeness, which relates to the computational expressiveness and functionality of the languages. Bitcoin's smart contract languages like Ivy and BitML have limited functionality by design and work at a low level close to the virtual machine. In contrast, platforms like Ethereum, Hyperledger Fabric, Cardano, and Solana provide high-level languages that are Turing-complete, enabling advanced computation. Overall, the table illustrates how design choices in smart contract platforms cater to different tradeoffs between decentralization, security, performance and developer experience. Low-level languages favour security and control while higher-level ones target usability.

E. SMART CONTRACT CODE

1) Compiled: Smart contracts are coded using specialized languages like Solidity, Vyper, and Rust. Once written, the code is compiled into bytecode, a machine-readable version. This bytecode is then executed on the blockchain's virtual machine, responsible for secure and transparent smart contract execution in the decentralized network. This process ensures efficient and automated agreement enforcement in blockchain ecosystems, playing a crucial role in this transformative technology.
2) Script: Smart contract codes that are directly interpreted on the blockchain without a separate compilation step are known as scripting languages. Despite their infrequency, these interpreted languages have their significance, as they eliminate the need for pre-compilation, simplifying the development process.

F. SCRICPTLESS CONTRACT

Smart contracts have a privacy concern because the full scripts need to be made public during communication between users and smart contracts. This openness can lead to privacy leakage. However, there is an alternative solution known as scriptless contracts, which addresses this problem. Scriptless scripts were innovatively devised by Andrew Poelstra to introduce essential smart contract capabilities within the MimbleWimble platform. Scriptless contracts function similarly to standard transactions but still achieve the same objectives as smart contracts. The key difference is that scriptless contracts do not require revealing the contents of the smart contract, thus ensuring that the contract details remain private and undisclosed during the process [101].

Table 18, classifies smart contract languages into two categories based on their execution models. One is that are compiled into bytecode and one is that are directly interpreted.

TABLE 16. Smart Contract Language Characteristics

Language	Garbage Collection	Inheritance	Reentrancy Protection	Decentralized Oracle Support	Memory Safety	Type Safety	Secure Coding Practices
Ivy	✓	✓	✓	✓	✓	✓	Limited
RSK	×	✓	✓	✓	✓	✓	Limited
BitML	×	✓	✓	✓	✓	✓	Limited
Solidity	×	✓	✓	✓	✓	✓	Extensive
Scilla	✓	×	✓	✓	✓	✓	Extensive
Vyper	✓	×	✓	✓	✓	✓	Extensive
Flint	✓	×	✓	✓	✓	✓	Limited
Haskell	✓	✓	✓	✓	✓	✓	Extensive
Go	×	×	✓	×	✓	✓	Limited
Node.js	×	×	✓	×	✓	✓	Limited
Java	×	✓	✓	×	✓	✓	Extensive
Pact	×	✓	✓	✓	✓	✓	Extensive
C#	×	✓	✓	×	✓	✓	Extensive
F#	✓	✓	✓	✓	✓	✓	Limited
VB.net	×	✓	✓	×	✓	✓	Limited
Kotlin	✓	✓	✓	✓	✓	✓	Extensive
Python	×	×	✓	×	✓	✓	Limited
Ink!	✓	✓	✓	✓	✓	✓	Limited
C++	×	✓	✓	×	✓	✓	Extensive
Rust	✓	✓	✓	✓	✓	✓	Limited
RIDE	✓	✓	✓	✓	✓	✓	Limited
Michelson	✓	×	✓	×	✓	✓	Limited
Liquidity	✓	×	✓	×	✓	✓	Limited
TinyGo	✓	×	✓	×	✓	✓	Limited
TEAL	×	×	✓	×	✓	✓	Limited

G. EXECUTION PLATFORM

A smart contract's execution platform is the underlying infrastructure where the smart contract's code is processed and executed. It governs the execution of the contract's logic, the storage of data, and the management of interactions with the blockchain network. Different blockchain systems offer various execution environments for smart contracts. For instance, Ethereum one of the most frequently used smart contract systems, employs the Ethereum Virtual Machine (EVM) to execute smart contracts written in languages such as Solidity. Likewise, systems such as Binance Smart Chain, Tron, EOSIO, Cardano, and Polkadot provide execution environments tailored to certain languages and functionality.

1) Etherum Virtual Machine

EVM is introduced by Ethereum, a Turing-complete, stack-based virtual environment. Smart contracts are generally written in Solidity, a high-level language built expressly for the EVM. The gas mechanism in the EVM assures that calculations are resource-efficient and safe, but it also adds costs to contract execution.

2) Tron Virtual Machine

TVM is a lightweight, purpose-built and Turing-complete virtual machine that is designed exclusively for TRON's blockchain environment. Its main goal is to provide an efficient, reliable, secure, and scalable blockchain system that integrates smoothly with the existing Solidity smart contract development environment. Aside from this compatibility, TVM also has the DPoS consensus method. Unlike the Ethereum Virtual Machine's (EVM) Gas mechanism, TVM adds a novel notion of Energy, which allows transactions and smart contract activities to be carried out without using TRX (TRON's native money) [111].

3) Neo Virtual Machine

NeoVM is a lightweight virtual machine designed for executing smart contracts on the NEO blockchain. As a fundamental part of NEO, it offers Turing completeness and strong consistency, enabling the implementation of complex logic and ensuring uniform execution results across all nodes in the decentralized network. This robust support for decentralized applications is achieved by compiling high-level source code, such as Java or C#, using NeoCompiler into a standardized NeoVM instruction set, allowing cross-platform compatibility [112].

4) EOS Virtual Machine

The EOS VM is a Web Assembly (WASM) engine designed specifically for blockchain development, with three interpreters that improve smart contracts. These interpreters allow EOSIO to debug, compile quickly, and perform better. As a consequence, EOSIO can now handle smart contracts 12 times quicker than before. The EOS VM Interpreter allows for step-by-step debugging of C++ smart contracts, whilst the EOS VM JIT quickly compiles smart contracts into read-

TABLE 17. Comparison of Different Smart Contracts Used in Different Blockchains

Blockchain Platform	Smart Contract Language	Turing Complete	Computation Level
Bitcoin	Ivy, RSK, BitML [95]	No	Low-level (VM)
Ethereum	Solidity, Scilla, Vyper, Flint [105] [106]	Yes	Both low-level and high-level
Cardano	Haskell [123]	Yes	High level
Hyperledger Fabric	Go, Node.js, Java [124] [119]	Yes	High level
Kadena	Pact [109]	Yes	High level
Neo	C#, F#, VB.net, Java, Kotlin, Python [124]	Yes	High level
Polkadot	Ink!	Yes	High level
Quorum	Solidity [124] [39]	Yes	High level
R3 Corda	Kotlin [124]	No	Low level
EOS	C++ [124]	Yes	High level
Solana	Rust [100]	Yes	High level
Waves	RIDE [109]	Yes	High level
Tezos	Michelson, Liquidity [109]	Yes	High level
IOTA	TinyGo, Rust	Yes	High level
Waltonchain	Solidity	Yes	High level
Algorand	TEAL [88]	Yes	High level
Binance	Solidity [105]	Yes	High level

TABLE 18. Languages Compiled into Bytecode vs Directly Interpreted

Languages Compiled into Bytecode	Languages Directly Interpreted
Solidity, Rust, C, C++, Ink, Archetype, C#, VB.Net, F#, Go, Golang, RIDE, Vyper	Plutus Script (Cardano) [110]

ily referenced binary files. Finally, the EOS VM Optimized Compiler generates optimized versions that run quickly. The combined usage of these technologies increases the performance of smart contracts on the EOSIO platform dramatically [57].

Table 19, provides an overview of the execution platforms and corresponding blockchains that use virtual machines to execute smart contracts. The table lists four different blockchain platforms that use Virtual Machines. They are Ethereum, Tron, Neo, and EOS.

TABLE 19. Execution Platforms and Corresponding Blockchains

Virtual Execution Environment Platforms	Corresponding Blockchains
Ethereum Virtual Machine (EVM)	Ethereum [113], Binance Smart Chain [114], VeChain [115], Hyperledger Fabric [116]
Tron Virtual Machine (TVM)	Tron [111]
Neo Virtual Machine (NeoVM)	NEO [112]
EOS Virtual Machine (EOS VM)	EOS [57]

H. SMART CONTRACT WORKFLOW

The utilization of smart contracts facilitates the seamless execution of electronic financial transactions between two entities, ensuring optimal efficiency, robust security, and complete transparency. The process of developing a smart contract for facilitating money transfers entails a series of crucial steps. The initial step in the software development process involves the composition of code by programmers, wherein they define the business logic. The rules are explicitly defined to govern the monetary transaction between Person A and Person B, encompassing the stipulations regarding the precise amount of money to be transferred and the specific mechanism by which said funds will be deposited into Person B's designated account. The source code undergoes a compilation process wherein it is transformed into a format that can be comprehended by the blockchain network.

Upon successful compilation, the code undergoes deployment onto the blockchain network via a solitary cryptographic transaction. In the context of distributed computing, the network undertakes the crucial task of code validation and subsequently assigns a distinct address to the contract in question. The code has been successfully integrated into the immutable and secure ledger of the blockchain.

When an individual, referred to as Person A, decides to transfer funds to another individual, referred to as Person B, they proceed by initiating a smart contract through the triggering of a transaction. Upon activation, the embedded code initiates a series of actions, resulting in the deduction of the predetermined amount from Person A's digital wallet. Simultaneously, an equivalent sum is promptly credited to Person B's wallet.

Fig. 28, represents smart contract workflow; how different components communicate with each other in a smart contract.

The comprehensive procedure is supervised by the decentralized blockchain network as opposed to a centralized financial intermediary. The distributed ledger system guarantees the immutability of all transactions, thereby promoting transparency and enforcing strict adherence to the terms

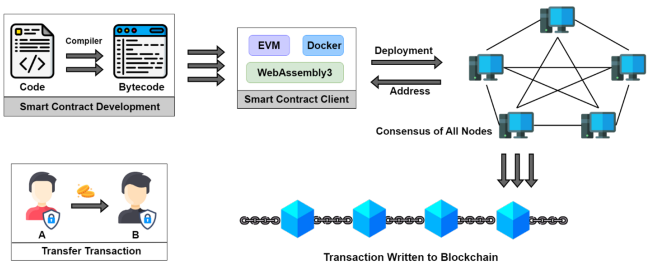

FIGURE 28. Smart contract workflow [86]

of the smart contract, thereby mitigating the potential risks associated with fraudulent activities or errors. The provision of confidence to participants for direct and instantaneous transactions, without reliance on trusted third parties, is a key benefit of this system.

IX. APPLICATION LAYER

In blockchain, the application layer plays a crucial rule. This application layer serves as a platform for developers and businesses to construct decentralized applications (dApps) that uses the capabilities of the underlying blockchain. These dApps span various domains like finance, supply chain management, gaming, and more. Interacting with the foundational protocols of the blockchain, the application layer utilises its consensus mechanisms, data storage abilities, and security features [88] [110].

A. COMPONENTS

The components of the Application Layer may vary depending on the specific blockchain network and the type of application being built on top of it. However, some common components include:

1) Wallets: Software or hardware tools used to store, send, and receive cryptocurrencies or other digital assets.
2) Cryptocurrency: Digital or virtual currency that utilizes blockchain technology for secure and decentralized transactions, eliminating the need for intermediaries like banks.
3) Internet of Things (IoT): Integration of blockchain with IoT devices enables secure and transparent data exchange, authentication, and automated transactions among connected devices.
4) Healthcare: Blockchain in healthcare ensures secure storage and sharing of medical records, facilitates interoperability between healthcare providers, and enables tracking of pharmaceutical supply chains. A decentralized Blockchain service might be the best choice for several governmental and non-governmental organizations involved in clinical or biomedical research. Using this technology, different organizations can collaborate with other institutions or organizations to share and analyze data without relinquishing control [117].
5) dApps (Decentralized Applications): Blockchain-based dApps leverage the transparent and immutable nature of blockchain for decentralized and tamper-resistant execution of smart contracts, which enable trustless interactions and remove the need for intermediaries. The adaptation of blockchain technology through dApps is increasing drastically; thus opportunity for the developers and entrepreneurs are growing rapidly [118].
6) NFT (Non-Fungible Token): NFTs on blockchain provide verifiable proof of ownership and authenticity for digital assets, revolutionizing digital art, collectables, and intellectual property rights management.
7) Voting System: Blockchain-based voting systems enhance transparency, security, and immutability by recording votes on the blockchain, ensuring the integrity of the electoral process and preventing tampering or fraud.
8) Supply Chain Management: Blockchain-powered supply chain management improves transparency and traceability by recording every step of the supply chain on an immutable ledger, enabling real-time tracking, authentication, and efficient management of goods and transactions.
9) DeFi (Decentralized Finance): DeFi on blockchain offers open and permissionless financial services such as lending, borrowing, and trading, eliminating the need for intermediaries and enabling anyone with internet access to participate in global financial activities.

B. MAIN COMPARISON PERIMETER

Now, when we are comparing the Application layer on different blockchains we should consider and judge it on the following parameters.

1) Ease of use
2) User experience
3) Functionality
4) Interoperability
5) Smart Contract Functionality

Table 20, represents the protocols and technology used in different blockchain application layers.

The following represents a compilation of terms accompanied by their respective descriptions used in Table 20.

1) Ethereum JSON-RPC: This is the communication bridge for developers to engage with Ethereum's blockchain nodes. Using the JSON format facilitates interactions like querying data and executing transactions on the Ethereum network. It's the behind-the-scenes language that connects applications with Ethereum's decentralized stage.
2) Solana JSON-RPC: This is the backstage pass for developers to communicate with Solana's blockchain nodes. It uses JSON language for smooth interaction, enabling tasks like querying information and making transactions on the Solana network. It's the behind-the-scenes maestro for seamless communication between apps and the dynamic Solana blockchain.

3) Avalanche JSON-RPC: This serves as the communication gateway for developers interacting with Avalanche's blockchain nodes.
4) Cardano GraphQL: This is the interactive script for developers engaging with Cardano's blockchain. Unlike traditional APIs, GraphQL allows users to specify the data they need, reducing unnecessary data transfer.
5) Algorand REST API: It is the entry point for developers to interact with the Algorand blockchain. REST, or Representational State Transfer, provides a set of architectural principles for building web services. Algorand's REST API allows developers to perform various operations, such as querying blockchain data, submitting transactions, and accessing key functionalities.
6) EOSIO RPC API: This is the gateway for developers to interact with EOSIO blockchain nodes. RPC, or Remote Procedure Call, allows seamless communication through EOSIO's API using the HTTP or WebSocket protocols.

C. SECURITY CONSIDERATIONS

The Application Layer remains essential to implement robust security measures in order to safeguard user data, uphold privacy, and prevent unauthorised access. Several key security considerations should be taken into account:

1) Cryptography: Public-key cryptography and digital signatures are essential cryptographic techniques. They play a important role in ensuring secure communications between components by providing authentication and tamper-proofing for transactions and interactions.
2) Consensus Mechanisms: The security of the Application Layer is influenced by the choice of consensus mechanism. For instance, Proof of Work (PoW) and Proof of Stake (PoS) mechanisms have important roles in safeguarding transaction integrity and deterring attacks.
3) Smart Contract Auditing: Auditing smart contracts for vulnerabilities is an imperative task. The presence of flaws within these contracts can lead to the exploitation of security loopholes, resulting in substantial financial losses. To ensure robustness and reliability, it is essential to conduct thorough code reviews, perform stringent security testing, and engage in third-party audits as
4) Access Control: Role-based access control mechanisms are crucial for managing permissions within dApps. By utilising smart contracts, access levels and conditions can be defined to ensure that only authorised parties have access to specific functionalities.
5) Privacy Solutions: Blockchain networks often employ privacy-focused solutions, such as zero-knowledge proofs. These solutions enable confidential transactions while still ensuring their validity.

X. USE CASE AND CASE STUDIES

The Application Layer is found in applications across industries. Here are a few examples:

1) Supply Chain Management: Walmart and IBM's collaboration on utilizing blockchain technology to track food products along the supply chain has yielded several benefits. Firstly, it has significantly enhanced transparency within the system, allowing for increased visibility into each step of the process. Additionally, in case of any contamination concerns or product recalls, this innovative approach has facilitated the swift identification of the sources responsible. This partnership serves as a commendable example of how supply chain management can leverage blockchain for improved efficiency and risk management.
2) Healthcare: MIT's MedRec project demonstrated how blockchain can revolutionize medical record sharing. By granting patients control over their medical data and enabling authorized healthcare providers to access specific information securely, this advancement ensures both data integrity and privacy protection. Blockchain technology thus emerges as a promising solution for enhancing patient care through streamlined information exchange.
3) Finance and DeFi: In finance and decentralized finance (DeFi), Ethereum's Application Layer has catalyzed the development of various financial services in a decentralized manner. Notable projects such as MakerDAO have introduced decentralized lending platforms and stablecoins that offer greater accessibility and reduce dependence on centralized institutions. Furthermore, Decentralized Exchanges (DEXs) enable peer-to-peer trading without intermediaries, providing individuals with direct control over their assets

XI. EVALUATION OF BLOCKCHAIN

Table 21, describes the various layers of blockchain and examples. Each layer has its own unique characteristics and there are some examples provided as well. Layer 0 serves as the foundation for the entire blockchain ecosystem, providing essential infrastructure and protocols. Moving up, we have Layer 1, the base blockchain layer where transactions undergo validation and finalization. On top of that, Layer 2 is built to address scalability and interoperability challenges. Finally, at Layer 3, we reach the application layer where decentralized applications are developed.

In Table 22, we can find a comprehensive description of the various functionalities associated with different layers of blockchain technology.

Beginning with Layer 0, this fundamental layer encompasses crucial elements such as consensus algorithms and cryptography, forming the backbone of the entire protocol and infrastructure. Moving up to Layer 1, we encounter core blockchain features like basic cryptocurrency functionality and secure transaction processing. Layer 2 takes center stage by providing innovative solutions for scalability and interop-

TABLE 20. Major Protocols and Technologies Used

Blockchain Network	Technology	Protocol/Standard
Ethereum	Ethereum Virtual Machine (EVM), Solidity, Web3.js	Ethereum JSON-RPC
Solana	Solana Runtime, Solana Web3.js Library	Solana JSON-RPC
Avalanche	Avalanche Contract Platform (ACPs), Avalanche.js	Avalanche JSON-RPC
Astar	Astar Contract Platform (ACP), Astar.js	Astar JSON-RPC
Cardano	Plutus, Marlowe, Cardano GraphQL API	Cardano GraphQL
Polkadot	Substrate, Ink, Polkadot-js	Polkadot JSON-RPC
NEAR	AssemblyScript, Rust, NEAR SDK	NEAR JSON-RPC
Hyperledger Fabric	Chaincode, Node.js SDK, Java SDK	gRPC, Apache Kafka
Corda	R3 CorDapp, Corda Node, Corda Finance SDK	Corda RPC, AMQP
Quorum	Smart Contract Manager, Quorum API, Tessera	JSON-RPC, RESTful API
Binance	Binance Smart Chain (BSC), Solidity, Web3.js	Binance RPC
Algorand	TEAL (Algorand's Smart Contract Language), AlgoSigner, PureStake	Algorand REST API
EOS Chain	EOSIO Smart Contract Development, EOSJS, Scatter	EOSIO RPC API

TABLE 21. Blockchain Layers Explained

Layer	Characteristic	Example
Layer 0	Focuses on the core blockchain protocol and infrastructure	Polkadot, Avalanche, Cosmos, etc.
Layer 1	Refers to the protocol layer of the blockchain ecosystem	Ethereum, Bitcoin, Binance, Cardano, Solana, etc.
Layer 2	Focuses on layer 2 solutions for scaling and interoperability	Lightning Network, Polygon, Immutable X, Synthetix, Loopring, Optimism, etc.
Layer 3	Refers to the application layer of the blockchain ecosystem	CakeDeFi, The Interledger Protocol (ILP) of Ripple, IBC protocol of Cosmos, ICON, Quant, etc.

erability, including channels, Plasma [120], sidechains [120], and rollups [121]. Finally, Layer 3 emerges as a valuable resource within the ecosystem by offering development tools and infrastructure specifically designed for decentralized applications.

A. DIFFERENT ALGORITHM IN USE

Data regarding the utilization of consensus algorithms across different networks was gathered from [122]. This table illustrates the distribution of these algorithms.

Table 23, represents the blockchain consensus algorithm in one column and the number of networks that use the consensus algorithm in other columns.

Fig. 29, displays a pie chart that provides a visual representation of the diverse range of algorithms currently employed within the industry.

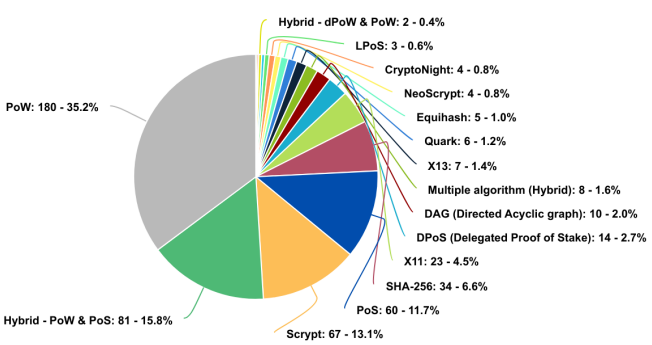

FIGURE 29. Pie chat for various algorithms in use

B. DIFFERENT ECOSYSTEM IN USE

Ecosystem utilization is evident from [122]. The following table illustrates the distribution of networks employing specific ecosystems.

Table 24, represents the blockchain ecosystem in one column and the number of networks that use the ecosystem in the other column.

Fig. 30 depicts a pie chart showcasing the array of ecosystems in use within the industry.

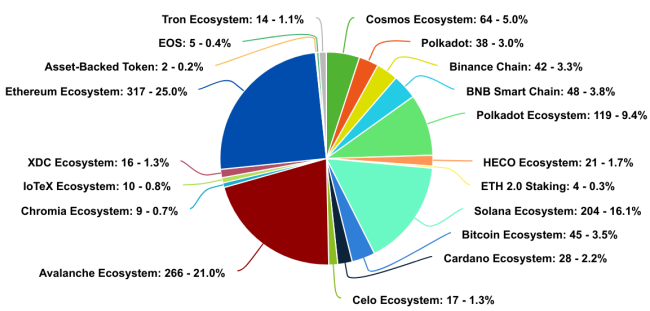

FIGURE 30. Pie chat for various ecosystems in use

C. ENTERGY CONSUMPTION

Blockchain technology uses a lot of electricity because it relies on powerful computers solving complex math problems to secure transactions and maintain the network. This high energy consumption has raised concerns about its environmental impact, but there are ongoing efforts to make blockchain more energy-efficient.

TABLE 22. Functionality of different Blockchain Layers

Functionality	Layer 0 (Core)	Layer 1 (Protocol)	Layer 2 (Infrastructure)	Layer 3 (Applications)
Protocol and Infrastructure	✓	✓	✓	✓
Consensus Algorithm	✓			
Network Structure and Blockchain Rules			✓	
Block Generation and Distribution	✓			
Low-Level Functionalities	✓			
Transaction Validation and Security			✓	
Consensus Mechanisms and Cryptography		✓	✓	✓
Infrastructure for Decentralized Networks		✓	✓	✓
Transaction Execution and Validation		✓	✓	✓
Core Blockchain Features	✓	✓		
Basic Cryptocurrency Functionality		✓	✓	
Secure Transaction Processing		✓	✓	
Development of Decentralized Applications			✓	✓
Tools for dApp Development				✓
Smart Contract Execution and Logic				✓
Developer Tools for Blockchain Apps				✓
Integration with Existing Systems				✓
Example	Polkadot, Avalanche, etc.	Ethereum, Bitcoin, Solana, etc.	Lightning, Network, Polygon, Loopring, etc.	CakeDeFi, The Interledger, Inter-Blockchain, etc.

TABLE 23. Consensus Algorithm Used by Different Blockchains

Algorithm	Number of Networks
DAG (Directed Acyclic Graph)	10
DPoS (Delegated Proof of Stake)	14
Hybrid - PoW & PoS	81
PoS	60
PoW	180
Multiple Algorithm (Hybrid)	8
CryptoNight	4
Equihash	5
NeoScrypt	4
Quark	6
Scrypt	67
SHA-256	34
X11	23
X13	7
LPoS	3
X14	1
Hybrid - dPoW & PoW	2
Hybrid - PoS & LPoS	1
Hybrid - PoW & nPoS	2

TABLE 24. Ecosystem Used by Different Blockchains

Ecosystem	Number of Networks
Cosmos Ecosystem	64
Ethereum Ecosystem	317
Asset-Backed Token	2
EOS	5
Tron Ecosystem	14
Polkadot	38
Binance Chain	42
BNB Smart Chain	48
Polkadot Ecosystem	119
HECO Ecosystem	21
ETH 2.0 Staking	4
Solana Ecosystem	204
Bitcoin Ecosystem	45
Cardano Ecosystem	28
Celo Ecosystem	17
Avalanche Ecosystem	266
Chromia Ecosystem	9
IoTeX Ecosystem	10
XDC Ecosystem	16

1) Entergy Consumption of Bitcoin

Fig. 31, represents the energy consumption of Blockchain Blockchain. X-Axis represents year and month and Y-Axis represents the amount of energy used in Terawatt [123].

2) Entergy Consumption of Etherum

Fig. 32, represents the energy consumption of Ethereum Blockchain. The x-axis represents year and month and the Y-axis represents the amount of energy used in Terawatt. The

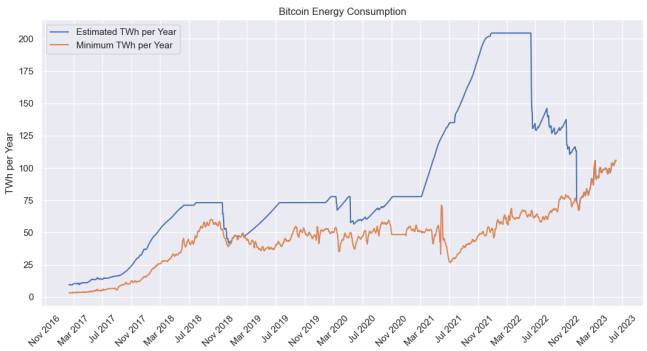

FIGURE 31. Entergy consumption of Bitcoin

data are collected from [123].

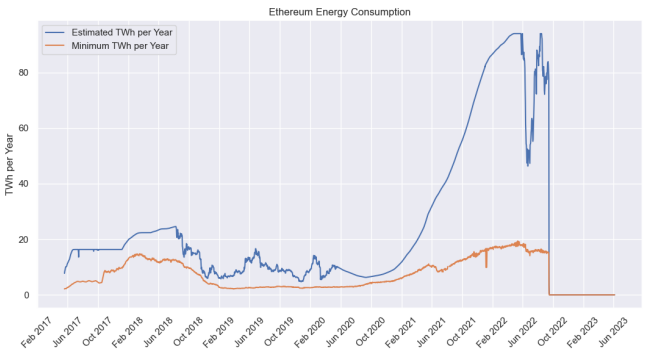

FIGURE 32. Entergy consumption of Etherum

D. PSEUDO CODE USED IN SIMULATION

Algorithm 1 outlines a Transaction class with methods for initializing transaction details (sender, recipient, amount) and converting transactions into a dictionary format. It provides a structured approach for managing and representing transactions in a system.

Algorithm 1 Transaction Class Definition

```
class Transaction:
    procedure Initialize(sender, recipient, amount):
        transactionSender, transactionRecipient, transactionAmount ← sender, recipient, amount

    procedure ToDict():
        return {'sender': transactionSender, 'recipient': transactionRecipient, 'amount': transactionAmount}
```

Algorithm 2 defines a Block class with methods to initialize block attributes (index, previous hash, transactions, timestamp, proof) and calculate the hash value of the block based on its attributes using the SHA-256 algorithm. It provides a structured representation of individual blocks in a blockchain system, facilitating block management and cryptographic verification

Algorithm 2 Block Class Definition

```
class Block:
    procedure Initialize(index, previousHash, transactions, timestamp, proof):
        blockIndex,blockPreviousHash, blockTransactions, blockTimestamp, blockProof ← index, previousHash, transactions, timestamp, proof

    procedure CalculateHash():
        return hashlib.sha256( f"{blockIndex}{blockPreviousHash}{blockTransactions}{blockTimestamp}{blockProof}".encode()).hexdigest()
```

Algorithm 3 implements a Blockchain class, initializing an empty chain and list for pending transactions. It includes functions to create a genesis block, add blocks, add transactions, calculate block hashes, and retrieve the last block in the chain. This implementation establishes the foundational structure for managing a blockchain instance.

Algorithm 3 Blockchain Class Implementation

Require: None
Ensure: Initialized Blockchain instance

```
class Blockchain:
    procedure Initialize():
        chain ← Empty list to store blocks
        currentTransactions ← Empty list for pending transactions

    procedure CreateGenesisBlock():
        CreateBlock(initialProof, previousHash)

    procedure CreateBlock(proof, previousHash):
        block ← Block(
            index ← len(chain) + 1,
            previousHash ← previousHash or hash(chain[-1]),
            transactions ← currentTransactions,
            timestamp ← GetCurrentTime(),
            proof ← proof)
        currentTransactions ← Empty list
        appendBlockToChain(block)
        return block

    procedure AddTransaction(sender, recipient, amount):
        currentTransactions.append(Transaction(sender, recipient, amount))

    procedure Hash(block):
        return block.CalculateHash()

    procedure LastBlock():
        return chain[-1] if chain else None
```

PoWConsensus in Algorithm 4 class generates and validates proofs of work in a blockchain system. The proofOfWork method iterates through potential proofs until a valid one is found, following consensus rules. ValidateProof checks if the generated proof meets the required difficulty level by comparing it with a target prefix. This algorithm ensures blockchain network security and integrity via a consensus mechanism based on computational effort.

Algorithm 4 Proof-of-Work Consensus Definition

```
class PoWConsensus:
    procedure proofOfWork(lastProof, difficulty):
        proof ← initialProof
        while PoWConsensus.ValidateProof(lastProof, proof, difficulty) is False:
            proof ← proof + 1
        return proof

    procedure ValidateProof(lastProof, proof, difficulty):
        guess ← f"{lastProof}{proof}".encode()
        guessHash ←hashlib.sha256(guess).hexdigest()
        return guessHash[:difficulty] == ("targetPrefix".difficulty)
```

The VerifyTransaction procedure in Algorithm 5 ensures the validity of a transaction by verifying the presence of essential fields like 'sender', 'recipient', and 'amount'. If any of these fields are absent, it returns False, indicating an invalid transaction. Otherwise, it extracts the sender, recipient, and amount from the transaction and returns True, affirming the transaction's integrity.

Algorithm 5 Transaction Verification

```
procedure VerifyTransaction(transaction):
    if 'sender'||'recipient'||'amount' not in transaction:
        return False
    sender, recipient, amount ← transaction['sender'], transaction['recipient'], transaction['amount']
    return True
```

The PoSConsensus class in Algorithm 6 implements a Proof-of-Stake (PoS) consensus algorithm. It initializes validators, chains, and transaction lists. Validators can be added and selected randomly. The ProofOfStake method generates a proof by iterating until a valid one is found based on a validator's selection. The algorithm validates proofs against the previous block's proof and chosen validator, ensuring blockchain integrity.

The DPoSConsensus class in Algorithm 7 implements a Delegated Proof-of-Stake (DPoS) consensus algorithm, initializing delegate and vote structures. Delegates are added and randomly selected, while votes are recorded for each delegate. The algorithm generates proofs based on the selected delegate

Algorithm 6 Proof of Stake Consensus Algorithm

```
class PoSConsensus:
    procedure Initialize():
        validators, chain, currentTransactions ← Empty lists

    procedure AddValidator(validator):
        validators.append(validator)

    procedure SelectValidator():
        return ChooseRandomValidator(validators)

    procedure ProofOfStake(lastBlock):
        validator, proof ← SelectValidator(), initialProof
        while ValidateProof(lastBlock.proof, proof, validator) is False:
            proof ← proof + 1
        return proof

    procedure ValidateProof(lastProof, proof, validator):
        guess ←f"{lastProof}{proof}{validator}".encode()
        guessHash ← hashlib.sha256(guess) .hexdigest()
        return guessHash[:difficulty] = targetHash

    procedure GetPreviousHash(lastBlock):
        return hash(lastBlock)

    procedure Hash(block):
        blockString ← SerializeToJson(block)
        return hashlib.sha256(blockString.encode()).hexdigest()
```

and validates them against the previous block's proof, ensuring blockchain integrity. Additionally, it allows voting for delegates and retrieves the count of votes for each delegate, promoting a democratic block validation process.

The RaftNode class in Algorithm 8 initializes and manages node attributes in the Raft consensus algorithm, facilitating fault-tolerant distributed consensus among nodes. It allows nodes to transition between follower, candidate, and leader states, ensuring consistency and reliability in distributed systems.

The RequestVote procedure in Algorithm 9 handles vote requests from candidates, comparing their term with the current node's term. If the candidate's term is newer, the node becomes a follower and may grant its vote based on log consistency. This process ensures the election of a leader with the most up-to-date log, maintaining consistency in the distributed system.

The Simulate procedure in Algorithm 10 emulates node behavior within a given time frame. Followers update their

Algorithm 7 Delegated Proof of Stake Consensus Algorithm

Require: None
Ensure: Initialized DPoSConsensus instance

```
class DPoSConsensus:
    procedure Initialize():
        delegates ← Empty list
        chain ← Empty list
        currentTransactions ← Empty list
        votes ← { } {Dictionary to store votes}

    procedure AddDelegate(delegate):
        delegates.append(delegate)
        votes[delegate] ← []

    procedure SelectDelegate():
        return random.choice(delegates)

    procedure DelegatedProofOfStake(lastBlock):
        delegate, proof ← SelectDelegate(), initialProof
        while ValidateProof (lastBlock.proof, proof, delegate) is False:
            proof ← proof + 1
        return proof

    procedure ValidateProof(lastProof, proof, delegate):
        guess ← f"{lastProof}{proof}{delegate}" .encode()
        guessHash ← hashlib.sha256(guess) .hexdigest()
        return guessHash[:difficulty] = targetHash

    procedure Vote (delegate, voter):
        if delegate && voter in delegates:
            votes[delegate].append(voter)

    function GetVoteCount(delegate):
        return len(votes.get(delegate, []))
```

Algorithm 8 Raft Consensus Algorithm

Require: None
Ensure: Initialized RaftNode instance

```
class RaftNode:
    procedure Initialize(nodeId, totalNodes):
        currentNodeId, currentTotalNodes ← nodeId, totalNodes
        currentTerm, votedFor, log, commitIndex, lastApplied ← 0, None, [], 0, 0
        state, leader_id ← "follower", None
        timeout ← random.randint (minTimeout, maxTimeout) / 1000.0
        votesReceived, lastHeartbeat ← defaultdict(lambda: False), time.time()

    procedure BecomeFollower(term, leaderId):
        state, currentTerm, votedFor, currentLeaderId ← "follower", term, None, leaderId

    procedure BecomeCandidate():
        state, currentTerm, votedFor← "candidate", currentTerm + 1, nodeId,
        votesReceived ← defaultdict(lambda: False)

    procedure BecomeLeader():
        state, leaderId ← "leader", nodeId
```

Algorithm 9 Raft RequestVote

```
procedure RequestVote( term, candidateId, lastLogIndex, lastLogTerm):
    if term < currentTerm:
        return False
    if term > currentTerm:
        BecomeFollower(term, None)
    if votedFor == None || candidateId:
        if lastLogTerm > log[-1]["term"] || (lastLogTerm == log[-1]["term"] and lastLogIndex ≥ len(log)):
            votedFor ← candidateId
            return True
    return False
```

election timeout and transition to candidates if they don't receive heartbeats within the timeout. Candidates request votes, and if they secure a majority, they become leaders. Leaders send regular heartbeats to maintain leadership and update the commit index, simulating the dynamics of Raft-based distributed systems.

XII. DISCUSSION

A. SUMMARY OF FINDINGS

This research delivered a comprehensive taxonomy of blockchain architectures across five layers— data, network, consensus, smart contracts, and applications. Comparative analyses revealed how design choices in each layer impact properties like scalability, security, and decentralization. Furthermore, simulations and testbed evaluations enabled performance benchmarking and validation of improvements to protocols and interactions between layers.

B. INTERPRETATION OF RESULTS

The taxonomy development provides a principled basis for navigating blockchain's architectural complexity while the layered comparative analyses offer data-driven insights into customizing solutions for diverse use cases. The proposed improvements can inform the development of next-generation platforms balancing decentralization and scalability.

Algorithm 10 Raft Simulate()

```
procedure Simulate(maxDurationSeconds):
    startTime ← time.time()
    while time.time() - startTime < maxDurationSeconds:
        if state == "follower":
            if time.time() - lastHeartbeat > timeout:
                UpdateElectionTimeout()
                BecomeCandidate()
        else if state == "candidate":
            if time.time() - lastHeartbeat > timeout:
                UpdateElectionTimeout()
                BecomeCandidate()
                votes ← 1
                for nodeId in (totalNodes):
                    if nodeId ≠ nodeId:
                        if log and RequestVote(currentTerm, nodeId, len(log) - 1, log[-1]["term"]):
                            votes ← votes + 1
                            if votes>totalNodes/2:
                                BecomeLeader()
        else if state == "leader":
            if time.time() - lastHeartbeat > 0.1:
                SendHeartbeats()
                lastHeartbeat ← time.time()
                UpdateCommitIndex()
        time.sleep(0.1)
```

C. THEORETICAL IMPLICATIONS

This research sets the stage for blockchain systems tailored to application needs by elucidating the intricate interplay between protocols across layers. The knowledge generated can accelerate blockchain adoption across domains.

D. RESEARCH QUESTION

This research provides valuable insights into the architectural design of blockchain networks and how to successfully apply blockchain technology for diverse use cases. Through a rigorous analysis addressing the key research questions, both theoretical and practical knowledge was gained.

1) RQ1: What are the components of a typical blockchain network, and how do they interact with each other?
2) RQ2: What are the different protocols and mechanisms used in each layer of a blockchain network, and how do they impact the performance and security of the network?
3) RQ3: How can we evaluate the suitability of different blockchain consensus algorithms for various use cases?

This research aimed to provide foundational insights into blockchain network design through a systematic examination of its technical components and tradeoffs. Research Question 1 (RQ1) established a layered framework taxonomy to characterize blockchain network architecture and component interactions. This taxonomy served as the structural basis for subsequently evaluating design choices.

Research Question 2 (RQ2) advanced understanding of how different protocol options at each network layer impact critical properties and mechanisms. For example, the analysis revealed which smart contract programming languages may be best suited based on smart contract layer or system needs.

Research Question 3 (RQ3) sought to evaluate and compare the performance of four prevalent consensus mechanisms: Proof-of-Work, Proof-of-Stake, Delegated Proof-of-Stake, and Raft. The aim was to discern each mechanism's suitability for applications involving diverse network configurations and transaction processing requirements. Specifically, the study examined how effectively each consensus mechanism could handle varying: 1) node counts within the network, 2) volumes of transactions, and 3) required throughput levels. Since blockchain networks may differ in the number of participating nodes and transaction loads necessitated by their unique use cases, determining optimal throughput capabilities was paramount. Experiments were thus conducted to test the performance of each mechanism under a range of simulated conditions. Node counts, transaction volumes, and processing speeds were systematically altered to represent real-world application scenarios. Through this rigorous testing methodology, the researchers sought to establish benchmark guidelines for which consensus approach might be best tailored to diverse network deployments based on their specific performance needs and constraints. The findings aimed to provide design guidance on selecting the validation process most conducive for reliably supporting a given blockchain application architecture and transaction capacity requirements.

Overall, this research provided foundational insights into blockchain networks through a pragmatic exploration of their underlying components and design tradeoffs. By developing a layered model and evaluating consensus mechanisms under different conditions, the study disentangled complex technical aspects into actionable principles. Whether for specialists looking to optimize protocols or newcomers wishing to grasp blockchain's essence, this work demystified key considerations in an engaging, end-to-end manner. With its systematic yet accessible style, the study advances both knowledge and hands-on utility. Researchers gained benchmark guidelines for tailored solutions. Practitioners received practical guidance on matching networks to use cases. Overall, in deconstructing while also rebuilding blockchain into an insightful yet cohesive whole, this study moves the field closer to realizing mature, real-world applications. Its fresh yet nuanced perspective brings the right zest to fuel further progress towards holistic, evidence-driven adoption. By translating dense topics into a lively format, this research inspires both deeper understanding and broader innovation across the dynamic blockchain domain.

XIII. LIMITATIONS AND FUTURE WORK

While extensive in analytical evaluations, hands-on development of optimized platforms and novel consensus protocols remains future work. Experimental implementations based on the concepts proposed could yield further insights. Targeted collaborations with blockchain projects can aid translation of models into practical decentralized technologies.

In conclusion, this pioneering research illuminates blockchain's architectural foundations and provides a springboard for purpose-driven innovation. By bridging theory with practice, it can unleash blockchain's disruptive potential and transform domains ranging from finance and governance to science and healthcare.

XIV. CONCLUSION

This research presented a systematic taxonomy and comparative analysis of blockchain architectures spanning from the foundational data protocols to application platforms. Through a comprehensive literature review, a layered blockchain model encompassing data, network, consensus, smart contracts and applications was developed. Using things like scalability, decentralization, and sustainability as criteria, rigorous comparison analyses showed the pros and cons of each design choice in each layer. Also, thorough simulations using analytical modeling and testbed evaluations made it possible to compare performance across layers and consensus protocols and validate proposed improvements. The key outcomes of this research are threefold. Firstly, it delivers a holistic reference guide and knowledge base for comprehending the blockchain technology stack in a principled manner. Secondly, it provides data-driven insights into tailoring blockchain solutions based on specific requirements from use cases. Finally, it lays the groundwork for developing next-generation decentralized platforms that can balance scalability without compromising on decentralization. By bridging the gap between theoretical foundations and real-world adoption, this research aims to unlock blockchain's immense disruptive potential across domains ranging from finance and healthcare to supply chains. As blockchain penetrates mainstream consciousness, the frameworks and analyses furnished through this work can provide the bedrock for building purpose-driven and customized decentralized solutions. In conclusion, this thesis presents a pioneering taxonomy-based approach for dissecting complex blockchain protocols in a structured fashion. By navigating through the intricate workings of blockchain systems, this research aims to propel adoption and realize the far-reaching socioeconomic promise of this transformative technology.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *Cryptography Mailing list at https://metzdowd.com*, Mar. 2009.

[2] V. Buterin, "Ethereum White Paper," GitHub repository, vol. 1, pp. 22–23, 2013.

[3] M. Swan, *Blockchain: Blueprint for a New Economy*. "O'Reilly Media, Inc.", 2015.

[4] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the Security and Performance of Proof of Work Blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3–16, 2016.

[5] D. Tapscott and A. Tapscott, *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World*. Penguin, 2016.

[6] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.

[7] W. Mougayar, *The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology*. John Wiley & Sons, 2016.

[8] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A Global Naming and Storage System Secured by Blockchains," in *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pp. 181–194, 2016.

[9] H. W. Elendner, S. Trimborn, B. Ong, and T. M. Lee, "The Cross-Section of Crypto-Currencies as Financial Assets: An Overview," *Journal Name*, vol. X, no. Y, pp. Z, 2016, doi: 10.18452/4647.

[10] Y. Yuan and F. Wang, "Blockchain and Cryptocurrencies: Model, Techniques, and Applications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, pp. 1421-1428, 2018, doi: 10.1109/TSMC.2018.2854904.

[11] C. Cachin et al., "Architecture of the Hyperledger Blockchain Fabric," in *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, vol. 310, no. 4, pp. 1-4, 2016, Chicago, IL.

[12] D. W. Hughes, "The Accounting Centric Data Warehouse TM," 2004, doi: 10.1007/978-3-540-24700-5_4.

[13] W. Bolt, "Review of: Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, Steven Goldfeder, Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction," *Journal of Economic Literature*, vol. 55, no. 2, pp. 647-649, 2017. [Online]. Available: http://www.jstor.org/stable/26303525

[14] F. Ma, N. Tang, R. Xu, and Z. Zhang, "Electronic Contract Ledger System Based on Blockchain Technology," in *Journal of Physics: Conference Series*, vol. 1828, no. 1, p. 012112, 2021, IOP Publishing.

[15] F. Masood and A. R. Faridi, "An Overview of Distributed Ledger Technology and Its Applications," *Int. J. Comput. Sci. Eng*, vol. 6, no. 10, pp. 422–427, 2018.

[16] A. Schneider, "Decentralization: Conceptualization and Measurement," *Studies in Comparative International Development*, vol. 38, pp. 32–56, 2003, Springer.

[17] H. Taherdoost, "Smart Contracts in Blockchain Technology: A Critical Review," *Information*, vol. 14, no. 2, article number 117, 2023. [Online]. Available: https://www.mdpi.com/2078-2489/14/2/117

[18] C. Jarvis, "Cypherpunk Ideology: Objectives, Profiles, and Influences (1992–1998)," *Internet Histories*, vol. 6, no. 3, pp. 315–342, 2022, Taylor & Francis.

[19] B. Preneel, "The First 30 Years of Cryptographic Hash Functions and the NIST SHA-3 Competition," in *Topics in Cryptology - CT-RSA 2010*, J. Pieprzyk, Ed. Springer Berlin Heidelberg, 2010, pp. 1-14. doi: 10.1007/978-3-642-11925-5_41.

[20] A. Ramiro and R. de Queiroz, "Cypherpunk," *Internet Policy Review*, vol. 11, no. 2, 2022.

[21] Md Sadek Ferdous, Mohammad Jabed Morshed Chowdhury, Mohammad A. Hoque, Alan Colman, "Blockchain Consensus Algorithms: A Survey," arXiv:2001.07091 [cs.DC], 2020. [Online]. Available: http://arxiv.org/abs/2001.07091

[22] A. Zohar, "Bitcoin: Under the Hood," *Communications of the ACM*, vol. 58, no. 9, pp. 104–113, 2015, ACM New York, NY, USA.

[23] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*, pp. 557–564, 2017, IEEE.

[24] S. Zhai, Y. Yang, J. Li, C. Qiu, and J. Zhao, "Research on the Application of Cryptography on the Blockchain," in *Journal of Physics: Conference Series*, vol. 1168, p. 032077, 2019, IOP Publishing.

[25] J. Arya, A. Kumar, A. P. Singh, T. K. Mishra, and P. H. J. Chong, "Blockchain: Basics, Applications, Challenges and Opportunities," 2021.

[26] V. Buterin, "Merkling in Ethereum," Nov. 15, 2015. [Online]. Available: https://blog.ethereum.org/2015/11/15/merkling-in-ethereum. Accessed: 2024.

[27] H. S. de Ocáriz Borde, "An Overview of Trees in Blockchain Technology: Merkle Trees and Merkle Patricia Tries," 2022.

[28] Hyperchain, "Bucket Tree - Hyperchain Documentation," 2024. Accessed: 2024.

[29] S. Gueron, S. Johnson, and J. Walker, "SHA-512/256," in *2011 Eighth International Conference on Information Technology: New Generations*, pp. 354–358, 2011, IEEE.

[30] National Institute of Standards and Technology (NIST), "NIST Selects Winner of Secure Hash Algorithm (SHA-3) Competition," Oct. 2012. Accessed: 2024.

[31] H. Dobbertin, A. Bosselaers, and B. Preneel, "RIPEMD-160: A Strengthened Version of RIPEMD," in *International Workshop on Fast Software Encryption*, pp. 71–82, 1996, Springer.

[32] L. Ertaul, M. Kaur, and V. A. K. R. Gudise, "Implementation and Performance Analysis of PBKDF2, bcrypt, scrypt Algorithms," in *Proceedings of the International Conference on Wireless Networks (ICWN)*, pp. 66, 2016, The Steering Committee of The World Congress in Computer Science.

[33] J. Doerner, Y. Kondi, E. Lee, and A. Shelat, "Secure Two-Party Threshold ECDSA from ECDSA Assumptions," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 980–997, 2018, IEEE.

[34] S. Josefsson and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)," 2017.

[35] Y. Seurin, "On the Exact Security of Schnorr-Type Signatures in the Random Oracle Model," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 554–571, 2012, Springer.

[36] D. Boneh, M. Drijvers, and G. Neven, "BLS Multi-Signatures with Public-Key Aggregation," 2018. [Online]. Available: https://crypto.stanford.edu/~dabo/pubs/papers/BLSmultisig.html

[37] S. Delgado-Segura, C. Pérez-Sola, G. Navarro-Arribas, and J. Herrera-Joancomartí, "Analysis of the Bitcoin UTXO Set," in *Financial Cryptography and Data Security: FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers 22*, pp. 78–91, 2019, Springer.

[38] M. Belotti, N. Božić, G. Pujolle, and S. Secci, "A Vademecum on Blockchain Technologies: When, Which, and How," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3796–3838, 2019, IEEE.

[39] X. Fan, B. Niu, and Z. Liu, "Scalable Blockchain Storage Systems: Research Progress and Models," *Computing*, vol. 104, no. 6, pp. 1497–1524, 2022, Springer.

[40] P. D. Thai, M. Doan, W. Liu, T. Liu, S. Li, H.-s. Zhou, and T. N. Dinh, "Blockchain Peer-to-Peer Network: Performance and Security," in *Handbook on Blockchain*, pp. 55–83, 2022, Springer.

[41] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, B. Bhattacharjee, and others, "Discovering Bitcoin's Public Topology and Influential Nodes," *et al*, 2015.

[42] G. Wood and others, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," *Ethereum Project Yellow Paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[43] E. Peterfay, "Creating a Website for a Programmatically Generated NFT Collection: Solana Network," 2022.

[44] K. Sekniqi, D. Laine, S. Buttolph, and E. Gün Sirer, "Avalanche Platform," *Netw. Distrib. Ledgers*, 2020.

[45] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol," in *Annual International Cryptology Conference*, pp. 357–388, 2017, Springer.

[46] Z. Sun, X. Luo, and Y. Zhang, "Panda: Security Analysis of Algorand Smart Contracts," in *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 1811–1828, 2023.

[47] Litecoin-Project/litecoin, "Litecoin source tree," GitHub. Available at: https://github.com/litecoin-project/litecoin/tree/master (Accessed: 01 January 2024).

[48] G. Kappos, H. Yousaf, M. Maller, and S. Meiklejohn, "An Empirical Analysis of Anonymity in Zcash," in *27th USENIX Security Symposium (USENIX Security 18)*, pp. 463–477, 2018.

[49] Zcash, "Zcash - Internet Money," GitHub. Available at: https://github.com/zcash/zcash (Accessed: 29 January 2024).

[50] M. A. Manolache, S. Manolache, and N. Tapus, "Decision Making Using the Blockchain Proof of Authority Consensus," *Procedia Computer Science*, vol. 199, pp. 580–588, 2022, Elsevier.

[51] D. Dias and J. Benet, "Distributed Web Applications with IPFS, Tutorial," in *Web Engineering: 16th International Conference, ICWE 2016, Lugano, Switzerland, June 6-9, 2016. Proceedings 16*, pp. 616–619, 2016, Springer.

[52] S. Kim, "Measuring Ethereum's Peer-to-Peer Network," PhD thesis, University of Illinois at Urbana-Champaign, 2017.

[53] J. A. Donet , C. Pérez-Sola, and J. Herrera-Joancomartí, "The Bitcoin P2P Network," in *International Conference on Financial Cryptography and Data Security*, pp. 87–102, 2014, Springer.

[54] Peer-to-peer (P2P) Networking, "Cardano Docs." Available at: https://docs.cardano.org/explore-cardano/cardano-network/p2p-networking/ (Accessed: 29 January 2024).

[55] What is Tezos?, "Scorechain." Available at: https://www.scorechain.com/resources/crypto-glossary/tezos (Accessed: 29 January 2024).

[56] Network Peer Protocol, "EOSIO Developer Docs." Available at: https://developers.eos.io/welcome/v2.0/protocol-guides/network_peer_protocol (Accessed: 29 January 2024).

[57] J. S. Yadav, N. S. Yadav, and A. K. Sharma, "A Qualitative and Quantitative Parametric Estimation of the Ethereum and TRON Blockchain Networks," in *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, pp. 1–5, 2021, IEEE.

[58] A. Shaleva and V. Korkhov, "Evaluation of the Neo P2P Blockchain Network Protocol Efficiency," in *Computational Science and Its Applications–ICCSA 2021: 21st International Conference, Cagliari, Italy, September 13–16, 2021, Proceedings, Part IV*, pp. 56–71, 2021, Springer.

[59] N. Sealey, A. Aijaz, and B. Holden, "IOTA Tangle 2.0: Toward a Scalable, Decentralized, Smart, and Autonomous IoT Ecosystem," in *2022 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pp. 01–08, 2022, IEEE.

[60] G. Wood, "Polkadot: Vision for a Heterogeneous Multi-chain Framework," *White Paper*, vol. 21, no. 2327, p. 4662, 2016.

[61] S. K. Kim, Z. Ma, S. Murali, J. Mason, A. Miller, and M. Bailey, "Measuring Ethereum Network Peers," in *Proceedings of the Internet Measurement Conference 2018*, pp. 91–104, 2018.

[62] BNB-Chain, "BSC: A BNB Smart Chain client based on the go-ethereum fork," GitHub. Available at: https://github.com/bnb-chain/bsc (Accessed: 29 January 2024).

[63] M. W. Murhammer, O. Atakan, S. Bretz, L. R. Pugh, K. Suzuki, and D. H. Wood, "TCP/IP Tutorial and Technical Overview," *IBM Redbooks, International Technical Support Organization*, vol. 8.

[64] J. B. Postel, C. A. Sunshine, and D. Cohen, "The ARPA Internet Protocol," *Computer Networks (1976)*, vol. 5, no. 4, pp. 261–271, 1981, Elsevier.

[65] J. Postel, "User Datagram Protocol," 1980.

[66] D. Coutts, N. Davies, K. Knutsson, M. Fontaine, A. Santos, M. Szamotulski, and A. Vieth, "The Shelley Networking Protocol," 2022.

[67] Polkadot Protocol Specification, "Networking," [Online]. Available: https://spec.polkadot.network/chap-networking (Accessed: 29 January 2024).

[68] Cosmos Network, "Internet of Blockchains," [Online]. Available: https://v1.cosmos.network/resources/faq (Accessed: 29 January 2024).

[69] Solana, "Solana Network Upgrades," [Online]. Available: https://solana.com/news/solana-network-upgrades (Accessed: 29 January 2024).

[70] H. Wang, Y. Wang, Z. Cao, Z. Li, and G. Xiong, "An Overview of Blockchain Security Analysis," in *Cyber Security: 15th International Annual Conference, CNCERT 2018, Beijing, China, August 14–16, 2018, Revised Selected Papers*, Springer Singapore, 2019, pp. 55–72.

[71] Q. Dai, B. Zhang, and S. Dong, "A DDoS-attack Detection Method Oriented to the Blockchain Network Layer," *Security and Communication Networks*, vol. 2022, pp. 1-18, 2022.

[72] M. Swan, *Blockchain: Blueprint for a New Economy*, "O'Reilly Media, Inc.", 2015.

[73] A. K. Yadav and K. Singh, "Comparative Analysis of Consensus Algorithms of Blockchain Technology," in *Ambient Communications and Computer Systems: RACCCS 2019*, Springer, 2020, pp. 205-218.

[74] B. Sriman, S. Ganesh Kumar, and P. Shamili, "Blockchain Technology: Consensus Protocol Proof of Work and Proof of Stake," in *Intelligent Computing and Applications: Proceedings of ICICA 2019*, Springer, 2021, pp. 395-406.

[75] J. T. George, "Proof of Stake: Consensus of the Future," in *Introducing Blockchain Applications: Understand and Develop Blockchain Applications Through Distributed Systems*, Springer, 2021, pp. 107-123.

[76] J. Yusoff, Z. Mohamad, and M. Anuar, "A Review: Consensus Algorithms on Blockchain," *Journal of Computer and Communications*, vol. 10, no. 09, pp. 37-50, 2022.

[77] C. Natoli, J. Yu, V. Gramoli, and P. Esteves-Verissimo, "Deconstructing Blockchains: A Comprehensive Survey on Consensus, Membership, and Structure," *arXiv preprint arXiv:1908.08316*, 2019.

[78] 360 Total Security, "Analysis and Improvement of NEO's DBFT Consensus Mechanism," 2018. [Online]. Available: https://blog.360totalsecurity.com/en/analysis-and-improvement-of-neos-dbft-consensus-mechanism. [Accessed: 29 January 2024].

[79] S. K. Arora, G. Kumar, and T.-h. Kim, "Blockchain based trust model using Tendermint in vehicular ad-hoc networks," *Applied Sciences*, vol. 11, no.

5, p. 1998, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/5/1998.

[80] D. Wang, Z. Wang, and X. Lian, "Research on Distributed Energy Consensus Mechanism Based on Blockchain in Virtual Power Plant," *Sensors*, vol. 22, no. 5, p. 1783, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/5/1783.

[81] X. Zhu, Y. Li, L. Fang, and P. Chen, "An Improved Proof-of-Trust Consensus Algorithm for Credible Crowdsourcing Blockchain Services," *IEEE Access*, vol. 8, pp. 102177–102187, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9141831.

[82] S. S. Panda, B. K. Mohanta, U. Satapathy, D. Jena, D. Gountia, and T. K. Patra, "Study of Blockchain Based Decentralized Consensus Algorithms," in *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*, pp. 908–913, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8929358.

[83] A. Ghosh, I. Sarkar and M. Dey, "Artificial Intelligence and Blockchain: Implementation Perspectives for Healthcare beyond 5G," in *Blockchain Applications for Healthcare Informatics*, pp. 93–116, 2022, Elsevier.

[84] S. Kim and J. Kim, "POSTER: Mining with Proof-of-Probability in Blockchain," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 841–843, 2018.

[85] F. Cheng, X. Ning, and Y. Dong, "A New Double-Layer Decentralized Consistency Algorithm for the Multi-Satellite Autonomous Mission Allocation Based on a Block-Chain," *Sensors*, vol. 22, no. 19, p. 7387, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/19/7387.

[86] W. Deng, T. Huang, and H. Wang, "A Review of the Key Technology in a Blockchain Building Decentralized Trust Platform," *Mathematics*, vol. 11, no. 1, p. 101, 2022. [Online]. Available: https://www.mdpi.com/2227-7390/11/1/101.

[87] Y. Yuan, F.-Y. Wang, and others, "Blockchain: The State of the Art and Future Trends," *Acta Automatica Sinica*, vol. 42, no. 4, pp. 481–494, 2016. [Online]. Available: http://www.aas.net.cn/cn/article/doi/10.1007/s10483-015-1961-2.

[88] R. M. Parizi, Amritraj, and A. Dehghantanha, "Smart Contract Programming Languages on Blockchains: An Empirical Evaluation of Usability and Security," in *Blockchain–ICBC 2018: First International Conference, Held as Part of the Services Conference Federation, SCF 2018, Seattle, WA, USA, June 25-30, 2018, Proceedings 1*, pp. 75–91, 2018, Springer.

[89] Solidity, Available at: https://docs.soliditylang.org/en/develop, Accessed: 29 January 2024.

[90] W.-M. Lee, "Beginning Ethereum Smart Contracts Programming: With Examples in Python, Solidity and JavaScript," Springer, 2019.

[91] J. M. Montes, C. E. Ramirez, M. C. Gutierrez, and V. M. Larios, "Smart Contracts for Supply Chain Applicable to Smart Cities Daily Operations," in *2019 IEEE International Smart Cities Conference (ISC2)*, pp. 565–570, 2019, IEEE.

[92] D. Vujičić, D. Jagodić, and S. Ranić, "Blockchain Technology, Bitcoin, and Ethereum: A Brief Overview," in *2018 17th International Symposium Infoteh-Jahorina (Infoteh)*, pp. 1–6, 2018, IEEE.

[93] 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)*, pp. 839–858, 2016, IEEE.

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

[95] 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, 2020, Elsevier.

[96] V. Capocasale and G. Perboli, "Standardizing Smart Contracts," *IEEE Access*, vol. 10, pp. 91203–91212, 2022, IEEE.

[97] C. Dannen, *Introducing Ethereum and Solidity*, vol. 1, 2017, Springer.

[98] Chainlink, "Top 6 Smart Contract Languages in 2024," https://chain.link/education-hub/smart-contract-programming-languages, Accessed: 29 January 2024.

[99] V. Buterin, "Vyper Documentation," Retrieved Oct, vol. 30, pp. 2018, 2018.

[100] V. Buterin, "Critical update re: DAO vulnerability - Ethereum Blog," 2020-09-10. https://blog.ethereum.org/2016/06/17/critical-update-re-dao-vulnerability, 2016.

[101] T. N. Tavu, "Automated Verification Techniques for Solana Smart Contracts," Ph.D. thesis, 2022.

[102] B. Hu, Z. Zhang, J. Liu, Y. Liu, J. Yin, R. Lu, and X. Lin, "A Comprehensive Survey on Smart Contract Construction and Execution: Paradigms, Tools, and Systems," *Patterns*, vol. 2, no. 2, 2021, Elsevier.

[103] W. Bugden and A. Alahmar, "Rust: The Programming Language for Safety and Performance," *arXiv preprint arXiv:2206.05503*, 2022.

[104] V. Piantadosi, G. Rosa, D. Placella, S. Scalabrino, and R. Oliveto, "Detecting Functional and Security-Related Issues in Smart Contracts: A Systematic Literature Review," *Software: Practice and Experience*, vol. 53, no. 2, pp. 465–495, 2023, Wiley Online Library.

[105] M. Bartoletti and R. Zunino, "BitML: A Calculus for Bitcoin Smart Contracts," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 83–100, 2018.

[106] F. Schrans, S. Eisenbach, and S. Drossopoulou, "Writing Safe Smart Contracts in Flint," in *Companion Proceedings of the 2nd International Conference on the Art, Science, and Engineering of Programming*, pp. 218–219, 2018.

[107] I. Sergey, A. Kumar, and A. Hobor, "Scilla: A Smart Contract Intermediate-level Language," *arXiv preprint arXiv:1801.00687*, 2018.

[108] P. Lamela Seijas, A. Nemish, D. Smith, and S. Thompson, "Marlowe: Implementing and Analysing Financial Contracts on Blockchain," in *Financial Cryptography and Data Security: FC 2020 International Workshops, AsiaUSEC, CoDeFi, VOTING, and WTSC, Kota Kinabalu, Malaysia, February 14, 2020, Revised Selected Papers 24*, pp. 496–511, 2020, Springer.

[109] S. Rouhani and R. Deters, "Security, Performance, and Applications of Smart Contracts: A Systematic Survey," *IEEE Access*, vol. 7, pp. 50759–50779, 2019, IEEE.

[110] R. M. Nadir, "Comparative Study of Permissioned Blockchain Solutions for Enterprises," in *2019 International Conference on Innovative Computing (ICIC)*, pp. 1–6, 2019, IEEE.

[111] M. Bartoletti, A. L. Cimoli, S. Lande, C. S. Murgia, A. Zunino, R. Z. Razak, "A Formal Model of Algorand Smart Contracts," in *Financial Cryptography and Data Security*, pp. 93-114, 2021.

[112] Cardano Docs, "Plutus Scripts,". https://docs.cardano.org/smart-contracts/plutus/plutus-scripts/, Accessed: 29 January 2024.

[113] Neo-Project, "Neo Documentation,". https://docs.neo.org/docs/en-us/basic/neovm.html, Accessed: 29 January 2024.

[114] EOSIO, "EOS VM,". https://eos.io/for-developers/build/eos-vm, Accessed: 29 January 2024.

[115] Y. Hirai, "Defining the Ethereum Virtual Machine for Interactive Theorem Provers," in *Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers 21*, pp. 520–535, 2017, Springer.

[116] Bitcoin.com, "What is BNB Token and BNB Smart Chain?: Learn about the token and platform supported by Binance Exchange,". https://www.bitcoin.com/get-started/what-is-bnb-and-bnb-smart-chain, Accessed: 29 January 2024.

[117] Md. A. Islam, Md. A. Islam, Md. A. H. Jacky, Md. Al-Amin, Md. S. U. Miah, Md. M. I. Khan, and Md. I. Hossain, "Distributed Ledger Technology Based Integrated Healthcare Solution for Bangladesh," *IEEE Access*, vol. 11, pp. 51527–51556, 2023, doi: 10.1109/ACCESS.2023.3279724.

[118] S. Khan, Md. Al-Amin, H. Hossain, N. Noor, and Md. W. Sadik, "A Pragmatical Study on Blockchain Empowered Decentralized Application Development Platform," in *Proceedings of the International Conference on Computing Advancements (ICCA 2020)*, pp. 82:1–82:9, 2020, ACM. doi: 10.1145/3377049.3377136.

[119] A. V. Aswin and B. Kuriakose, "An Analogical Study of Hyperledger Fabric and Ethereum," in *Intelligent Communication Technologies and Virtual Mobile Networks: ICICV 2019*, pp. 412–420, 2020, Springer.

[120] S. Vallabhaneni, D. Boscovic, and J. Kellso, "Leveraging Blockchain for Plasma Fractionation Supply Chains," PhD thesis, April 2020.

[121] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain Scaling Using Rollups: A Comprehensive Survey," *IEEE Access*, 2022, IEEE.

[122] CoinMarketCap, "Cryptocurrency Prices, Charts, and Market Capitalizations," [No Date]. https://coinmarketcap.com.

[123] V. Kohli, S. Chakravarty, V. Chamola, K. S. Sangwan, and S. Zeadally, "An Analysis of Energy Consumption and Carbon Footprints of Cryptocurrencies and Possible Solutions," *Digital Communications and Networks*, vol. 9, no. 1, pp. 79–89, 2023, Elsevier.

[124] T. M. Fernandez-Carames and P. Fraga-Lamas, "Towards Post-Quantum Blockchain: A Review on Blockchain Cryptography Resistant to Quantum Computing Attacks," *IEEE Access*, vol. 8, pp. 21091–21116, 2020, IEEE.

MD. RIFAT HOSSAIN received his B.Sc. degree in Computer Science and Engineering from the Faculty of Science Technology at American International University-Bangladesh, Dhaka, in 2023.

He is currently working as a software engineer, leveraging his expertise in computer science to contribute to various software projects. His research interests lie in the domain of core blockchain and its associated technologies.

FOYSAL AHAMED NIROB received B.Sc. degree in Computer Science and Engineering with a major in Software Engineering from American International University-Bangladesh (AIUB) in 2023.

He is currently working as a Junior Software Engineer at Traideas and also actively involved in a Machine Learning project. He also published a paper in the field of Machine Learning. His research interests include Blockchain, Machine Learning, Deep Learning, and Explainable AI, especially in medical image processing.

ARAFAT ISLAM graduated in Computer Science and Engineering from American International University-Bangladesh (AIUB) with a major in Computational Theory. His interests span AI, Deep Learning, Blockchain, and Graph Theory.

Arafat's notable achievement includes being a finalist in the Blockchain Olympiad Bangladesh (BCOLBD) 2023. With a commitment to pushing the boundaries of innovation, Arafat continues to contribute valuable insights and advancements to the scientific community.

TANJIM MAHMUD RAKIN has completed his B.Sc. degree in Computer Science and Engineering with major in Information System from American International University-Bangladesh.

He is currently working in ABG TECHNOLOGIES LIMITED, a FinTech concern under one of the largest conglomerate of Bangladesh, Bashundhara group as a Senior Officer, Strategy and Research. Prior to that he was business development associate at DGePay Services Limited. He was one of the 37 finalists in Blockchain Olympiad Bangladesh 2023.

His research interests include domains of Blockchain , Machine Learning and AI. Apart from that he has interest in Open Innovative Research to solve real-life problems.

MD. AL-AMIN currently working as a lecturer in the Computer Science Department, at American International University-Bangladesh (AIUB). Besides his teaching profession, he founded DeepChain Labs a Web3-focused AI-driven software development company. He is also supervising research teams providing solutions architect consultancy.

He received his Bachelor's in Software Engineering and Master's degree in Computer Science and Engineering (CSE) from American International University-Bangladesh(AIUB), Dhaka, Bangladesh in 2015 and 2017 respectively. During his Master's degree, he was awarded the academic distinction Magna Cum Laude(Silver Medal) award for his academic results. He achieved two times ICT Fellowship Awards for the years 2015-16 and 2016-17 from the Ministry of ICT, Government of Bangladesh.

He has also served as a distinguished juror in the National Blockchain Olympiad 2022 2023. He is a member of Bangladesh's Computer Society.

His research interest primarily focuses on Distributed Ledger Technology (DLT), Blockchain Technology, Web 3.0, Distributed Computing, Web of Things, Information Security, Web Assembly, and Knowledge base Systems.