IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 17 January 2023, accepted 5 February 2023, date of publication 13 February 2023, date of current version 16 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3244403

METHODS

Permissionless Blockchain Systems as Pseudo-Random Number Generators for Decentralized Consensus

RIAAN BEZUIDENHOUT[1], WYNAND NEL[1], AND JACQUES M. MARITZ[2]
[1]Department of Computer Science and Informatics, University of the Free State, Bloemfontein 9300, South Africa
[2]Department of Engineering Sciences, University of the Free State, Bloemfontein 9300, South Africa

Corresponding author: Riaan Bezuidenhout (bezuidenhoutr@ufs.ac.za)

ABSTRACT Consensus algorithms that function in permissionless blockchain systems must randomly select new block proposers in a decentralised environment. Our contribution is a new blockchain consensus algorithm called Proof-of-Publicly Verifiable Randomness (PoPVR). It may be used in blockchain design to make permissionless blockchain systems function as pseudo-random number generators and to use the results for decentralised consensus. The method employs verifiable random functions to embed pseudo-random number seeds in the blockchain that are *confidential*, *tamper-resistant*, *unpredictable*, *collision-resistant*, and *publicly verifiable*. PoPVR does not require large-scale computation, as is the case with Proof-of-Work and is not vulnerable to the exclusion of less wealthy stakeholders from the consensus process inherent in stake-based alternatives. It aims to promote fairness of participation in the consensus process by all participants and functions transparently using only open-source algorithms. PoPVR may also be useful in blockchain systems where asset values cannot be directly compared, for example, logistical systems, intellectual property records and the direct trading of commodities and services. PoPVR scales well with complexity linear in the number of transactions per block.

INDEX TERMS Consensus algorithm, decentralised consensus, permissionless blockchain systems, proof-based consensus algorithms, proof-of-publicly verifiable randomness, pseudo-random number generation, random number seeds, verifiable random functions, vote-based consensus algorithms.

I. INTRODUCTION

One of the fundamental concepts that were established when Nakamoto [1] introduced the idea of Bitcoin was the ability for individuals to transact without mediation or permission from a central authority. Holotescu [2] coined the term ''*disintermediation*'' to describe the absence of a central authority when applied to a blockchain system. Disintermediation is a requirement that springs from the distributed ledger (blockchain) that is shared and maintained over a peer-to-peer network [3]. Nodes in the peer-to-peer network share responsibility for storing copies of the blockchain, transmitting and verifying transactions and verifying the validity of the transaction record contained in the blockchain [4], [5].

The associate editor coordinating the review of this manuscript and approving it for publication was P. K. Gupta.

The distributed and disintermediated nature of blockchain systems requires a decentralised agreement protocol shared between participants of a blockchain system that will allow them to agree on the validity of the blockchain [6]. The authors in [7] describe the process as a community of participants that accept a set of digital governance rules or govern the blockchain system. These rules, also called consensus algorithms, provide a source of algorithmic trust [8] between participants and must be transparent if parties are to agree on the correctness of the blockchain [9]. This requirement for transparency further necessitates that blockchain systems operate on open-source principles [10]. Figure 1 shows how all these aspects, namely disintermediation, a peer-to-peer network, the distributed blockchain, algorithmic trust and open-source principles interact so that blockchain systems can function in a decentralised manner.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

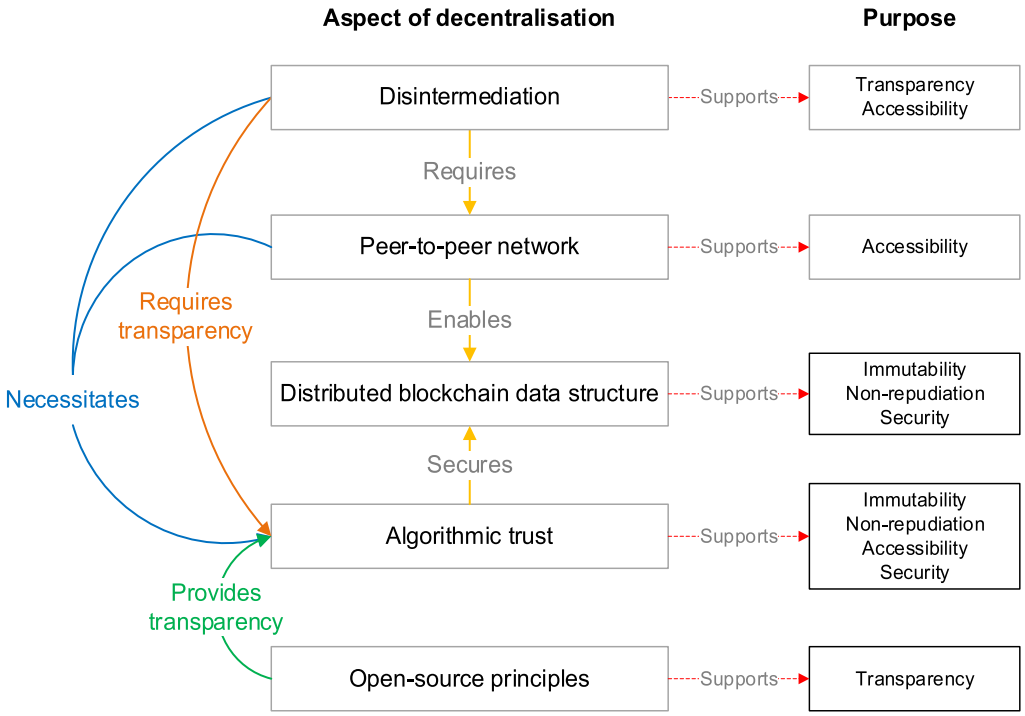

FIGURE 1. The aspects of decentralisation in permissionless blockchain systems [8].

In addition to the interaction between the aspects of decentralisation of a blockchain system, Figure 1 also illustrates how each aspect supports the various purposes of a permissionless blockchain system. These purposes of immutability, non-repudiation, security, transparency and accessibility are discussed in detail in II-B.

Decentralisation in permissionless blockchain systems revolves around algorithmic trust that allow disintermediated parties to maintain a distributed blockchain over a peer-to-peer network in an open-source environment [8].

At the heart of each consensus algorithm that functions in a decentralised blockchain system lies the goal of selecting the party with the right to add a new block to the blockchain at random [11]. It stands to reason, then, that ways must be found whereby permissionless blockchain systems can act as efficient pseudo-random number generators that produce pseudo-random numbers for use in consensus algorithms that select new block proposers. These pseudo-random numbers must be embedded in the blockchain data structure for public verification.

This study investigates decentralised blockchain consensus from the point of view of its most fundamental principles; as a problem in publicly verifiable randomness. In the background section, the authors strip the idea of permissionless blockchain systems down to their primary constituents and investigate what they intend to achieve (II-A to II-D). II-E and II-F give a thorough overview of the consensus algorithms that have so far been proposed and show that they are all attempts to achieve random block addition that is publicly verifiable. It also highlights the disadvantages inherent in these approaches that warrant continued investigations in the search for improvements. The subject of II-G is the myriad of ways to approach problems in publicly verifiable randomness.

The main contribution of this paper (III) is to introduce a generic approach to blockchain consensus, namely Proof-of-Publicly Verifiable Randomness (PoPVR). PoPVR consists of two main elements. It first builds a pseudo-random number generator to be incorporated into any blockchain system. Second, it shows how to use the pseudo-random number generator to select new block proposers in a confidential, tamper-resistant, unpredictable, collision-resistant, and publicly verifiable way. Blockchain designers can use PoPVR as an open-source approach to design new blockchain systems with a built-in consensus mechanism.

IV addresses the specific operational and security aspects of PoPVR. The paper concludes with a summary and future opportunities for further research.

II. BACKGROUND AND RELATED WORK

Blockchain technology is a way to keep the content of replicated ledgers shared by multiple participants and synchronised through community validation. Its inception was with Nakamoto's first cryptocurrency, Bitcoin, proposed in 2008 [1]. While Bitcoin was the first workable solution to design a decentralised digital currency, the idea of

decentralised digital money has been around since b-money was proposed in 1988 [12], [13]. Blockchain is a "*foundational technology, that leads the shift from trusting humans, to trusting machines and from centralized to decentralized control*" [14]. A detailed discussion of the terms blockchain and blockchain system, the purpose of blockchain systems, the environment in which permissionless blockchain systems function and their components follow in A to D.

A. BLOCKCHAIN AND BLOCKCHAIN SYSTEMS

The literature about blockchain-related topics often contains imprecise language related to the meaning of the term blockchain. It is sometimes referred to as a data structure and sometimes as a system. Some authors refer to blockchain as a data structure containing a sequence of blocks. Every block records a set of transactions, and blocks are cryptographically linked to create a tamper-proof historical transaction record [14], [15], [16]. The data structure interpretation is widely shared, described as an ordered list of blocks or a distributed ledger of transactions; the idea stems from the cryptographically linked, tamper-proof historical transaction record constructed through a blockchain [12], [15], [17]. Other data structure descriptions include a distributed ledger of transactions by [9], [18], and others. The authors in [19] call blockchain "*..an electronic ledger that records transactions in discrete chunks, referred to as blocks. The blocks are concatenated into a single immutable chain.*".

Alternatively, blockchain may also mean a combination of technologies, including distributed ledgers, cryptography and consensus algorithms used in combination as components in a decentralised transactional system. This interpretation then refers not only to a data structure but to a system, the purpose of which is to allow untrusted parties to agree on a single valid transaction history [10], [20], [21], [22]. Other systems type descriptions for blockchain that mean a combination of existing technologies are, for example, Merkle tree hashing, consensus algorithms and public-private-key encryption [10] or decentralised consensus, maintaining a synchronised distributed ledger over a peer-to-peer network [23].

A blockchain system can also be understood as a replicated state machine, a service replicated on many network nodes, protecting the system against the failure of single nodes. In this model, three components constitute the state machine. The blockchain represents the system's state, a peer-to-peer network containing blockchain replicas, and a consensus algorithm that runs as a distributed service to enable the nodes to converge on a single correct state [24].

In this paper, we explicitly differentiate the meaning of the terms blockchain and blockchain system and will use them as such throughout:

- A *blockchain* is a data structure with characteristics specifically designed to enable untrusted participants to agree on a single transaction record. It is a tamper-proof distributed ledger with cryptographically linked sequential blocks where each block contains a set of transaction data. A blockchain is a component of a blockchain system.
- A *blockchain system* means a combination of technologies, including cryptography, consensus algorithms, network topographies and the stakeholders that produce, consume or interact with blockchain-enabled services.

Blockchain systems fall into one of two groups. The first relies on a central authority to establish consensus. Participation in the system is limited to permitted entities with known identities only. These are called permissioned blockchain systems [6], [25] or sometimes private or enterprise blockchain systems [26]. This study does not consider these types of systems but focuses on permissionless or decentralised blockchain systems (sometimes called public blockchain systems). They require no restrictions on participants and use a probabilistic consensus mechanism to manage the addition of new blocks to the blockchain [26]. Bitcoin is an example of a permissionless blockchain system [6], [25].

B. THE PURPOSE OF A BLOCKCHAIN SYSTEM

A blockchain system's purpose is to record transactions. In this sense, transactions are not necessarily financial and may represent any type of electronic record. Examples of transactions recorded include traceability data in logistics [27], electronic voting data, decentralised domain name records and the code needed to execute smart contracts [28]. Blockchain transactions share the properties of immutability, non-repudiation, security, transparency and accessibilit *y* [10], [17]. These properties have the following definitions:

- Immutability: No party can alter a transaction recorded on the blockchain. A permissionless blockchain system has a probabilistic consensus model. It is, therefore, more accurate to say that a transaction can eventually not be altered. It is because the transaction becomes persistent as it is buried under a suitable number of new blocks [10], [14], [22], [29].
- Non-repudiation stems from immutability. If no one can alter a transaction, it cannot be undone, repudiated or deleted [17]. Immutability and non-repudiation are achieved by embedding cryptographic hash pointers in the blockchain. It forms a tamper-proof sequential record of transactions [26], [30].
- Security: In the context of blockchain systems, security means three things. First, public-key cryptography protects data ownership, which permits only the lawful owner of a private key to transact with its data on the blockchain [29], [31]. Second, cryptographic hash pointers act as a chain of links between blocks to create a tamper-proof transaction record. It protects the integrity of the blockchain [14], [30]. Third, a distributed consensus model replaces centralised authorities. Copies of the blockchain stored across many independent peers on a peer-to-peer network provide redundancy and mitigate single-point failure risk, improving fault tolerance [22].

- Transparency: Guarantees that all transactions on the blockchain are auditable by any stakeholder; this means any party with access to the Internet [10], [22], [26], [29].
- Accessibility: All participants have the same right to transact on and participate in blockchain processes [17]. Transacting is limited to each owner's data only, while participation allows the inspection of data of all participants to, for example, verify transaction validity.

C. PERMISSIONLESS BLOCKCHAIN SYSTEMS AND THEIR ENVIRONMENT

This study pertains to permissionless or public blockchain systems. Understanding the permissionless environment requires a distinction between distributed and decentralised systems, which can be defined in the following way [32]:

- "***Distributed system***: A system with multiple components that have their behaviour coordinated via message passing. These components are usually spatially separated and communicate using a network, and may be managed by a single root of trust or authority."
- "***Decentralized system***: A distributed system in which multiple authorities control different components, and no single authority is fully trusted by all others."

Therefore, decentralised systems are distributed, but not all distributed systems are decentralised. Permissionless blockchain systems relate to these definitions in that they function on a peer-to-peer basis, are not controlled by a central authority and use a decentralised consensus algorithm for the network to agree on the legitimate state of the blockchain [9], [10], [21], [22].

Layers in a blockchain system

A permissionless blockchain system's environment comprises three layers: external, primary, and secondary (Figure 2) [8].

The primary layer is central to a blockchain system. It provides the foundation for the mechanical operation of the system and contains three sub-components, the blockchain, a peer-to-peer network, and a consensus mechanism [9], [22], [30]. A blockchain system can contain a secondary layer of more sophisticated applications. These are constructed on top of the primary layer and are accessed by embedding autonomous instructions in blockchain transactions. These instructions can execute automatic functions provided by the primary layer, for example, create automated transactions when certain conditions are met or perform additional services on the secondary layer [21]. These autonomous instructions are called smart contracts, which have an expansive meaning [22]. Note that the secondary layer components cannot be accessed directly. They require that a transaction is initiated on the primary layer to trigger an event or action in a secondary layer component. Ethereum is an example of a blockchain system that supports secondary layer functionality [33].

The primary and secondary layers of the blockchain environment are technical. There exists an external social layer in which blockchain systems are immersed. Blockchain systems exist to fulfil a valuable function to consumers; they are maintained by developer communities and attract more and more attention from regulators and lawmakers. These external layer components include:

- Developers who create and maintain the software related to primary and secondary layer components [34], [35], [36]. These developers may include volunteer communities or businesses that operate for profit [21].
- Consumers or participants that transact with the blockchain system directly with the primary layer or indirectly with the secondary layer. Participants are not limited to individuals; they may be organisations or other systems, for example, IoT devices.
- Lawmakers are increasingly interested in blockchain systems and constantly re-evaluate the requirements for oversight in areas ranging from consumer protection and tax evasion to money laundering [10].

Figure 2 shows how the secondary layer resides inside the primary layer. The secondary layer cannot function without the primary layer. The external layer does not have direct access to the secondary layer. The primary layer controls its access, and participants interact with the secondary layer by initiating transactions on the primary layer. This paper focuses on a foundational infrastructure component of permissionless blockchain systems, the consensus algorithm. Figure 2 shows that blockchain consensus algorithms reside in the primary layer of the blockchain system. Accordingly, this primary layer needs more detailed scrutiny.

D. PRIMARY LAYER COMPONENTS OF A PERMISSIONLESS BLOCKCHAIN SYSTEM

The primary layer of a permissionless blockchain system comprises three components: the blockchain data structure containing the transaction record, a peer-to-peer network and a consensus algorithm. Each is now discussed in detail.

1) BLOCKCHAIN

Before the advent of blockchain systems, the idea of tamper-proofing a digital artefact when releasing it into the public domain was dependent on a trusted third party. This method was vulnerable to malicious acting by the third party until it was solved by [37] using cryptographic hash functions and public key digital signatures. These two cryptographic primitives, namely cryptographic hash functions and public key digital signatures (part of public key cryptography), form the basis of the tamper-proofing characteristics of blockchain systems. The cryptographic primitives that play an essential role in blockchain systems and their implementation in the construction of transactions and blockchain blocks are discussed in II-D 3.

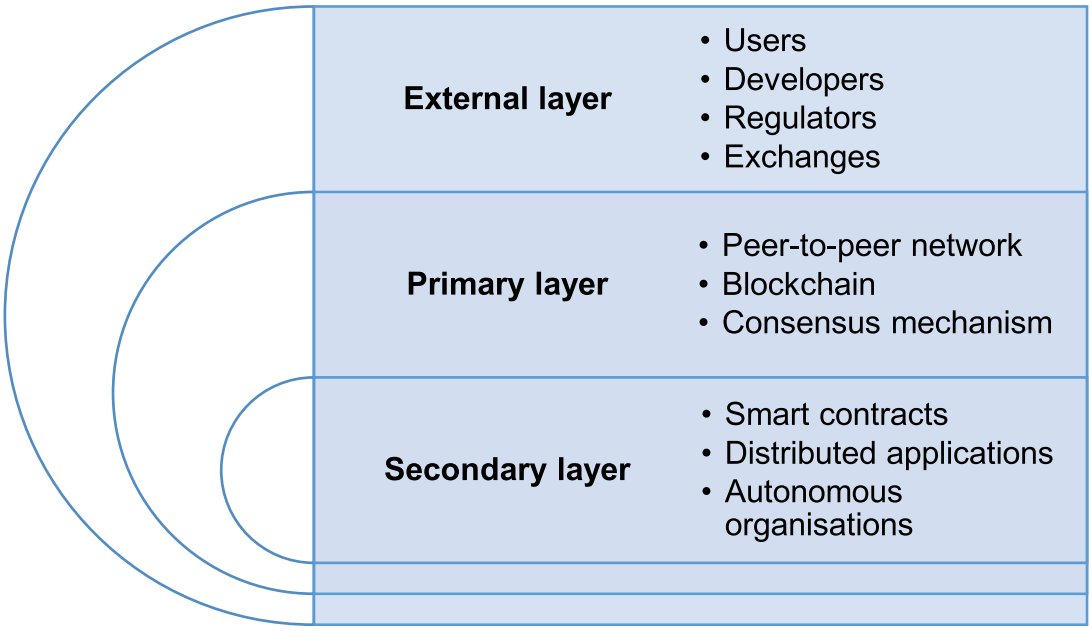

FIGURE 2. The layers in a blockchain system environment (Adapted from [8]).

a: CRYPTOGRAPHICALLY SECURE, COLLISION-RESISTANT HASH FUNCTIONS

Cryptographic hash functions (hash functions for short) are cryptographic primitives that take an arbitrary length message and produce an output, called a message digest, of a fixed length [38]:

$$\text{Hash (message)} \rightarrow \text{message\_digest} \tag{1}$$

A well-known hash function is the SHA256 hash function that returns an output of 256-bit length. Hash functions have three important properties. *First*, they are one-way functions, making it impossible to compute the input when the output is known. *Second*, good hash functions have a computational complexity of O(n) where n is the input length, and third, good hash functions are collision-resistant [30], [39].

Blockchain systems use hash functions to create a ''fingerprint'', known as the block hash, of a block of data. Figure 3 illustrates the process of creating a block hash for a set of transactions in a block.

Since a blockchain is a sequential series of data blocks, subsequent blocks point to previous blocks using the last block's hash as a unique pointer (Figure 4), producing a tamper-evident log of transactions.

In any attempt to alter the data in a transaction, the hash of the block that contains the transaction will be a valid pointer to the previous block. While the data structure of a blockchain is, in reality, more complex, this tamper-evident log is the fundamental idea of a blockchain (Figure 5) [9], [30].

b: PUBLIC KEY CRYPTOGRAPHY

Well-known public key primitives like RSA, Elgamal [38] and Elliptic Curve Cryptography [40] facilitate five cryptographic services used for data and communication security [39]:

The key generation algorithm generates two outputs, namely the public key and the secret key, for use in the four other services [38]:

$$\text{GenerateKeys}() \rightarrow \text{Key}_{\text{public}}; \text{Key}_{\text{secret}} \tag{2}$$

The public-key encryption algorithm inputs a public key and plain text message. It returns an encrypted message that only the owner of the corresponding private key can decrypt [38]:

$$\text{Encrypt}(\text{Key}_{\text{public}}, \text{Msg}_{\text{plain-text}}) \rightarrow \text{Msg}_{\text{encrypted}} \tag{3}$$

The decryption algorithm recovers the plain text message from the encrypted message using the secret key [38]:

$$\text{Decrypt}(\text{Key}_{\text{secret}}, \text{Msg}_{\text{encrypted}}) \rightarrow \text{Msg}_{\text{plain-text}} \tag{4}$$

Digital signature algorithms sign a plain text message using a secret key to allow authentication of the message by any party who knows the corresponding public key. Authentication means the message has not been tampered with and that it originated from the owner of the public key [38]:

$$\text{Sign}(\text{Key}_{\text{secret}}, \text{Msg}_{\text{plain-text}}) \rightarrow \text{Signature} \tag{5}$$

The signature verification uses the public key of the signer and the plain text message, and the signature to produce a boolean output to confirm authenticity [38]:

$$\text{Verify}(\text{Key}_{\text{public}}, \text{Msg}_{\text{plain-text}}, \text{Signature}) \rightarrow \text{Ver} \tag{6}$$

Public key cryptography plays a critical role in securing transactions on the blockchain [22]. While transactions vary between blockchain systems, they all share the same general construction process.

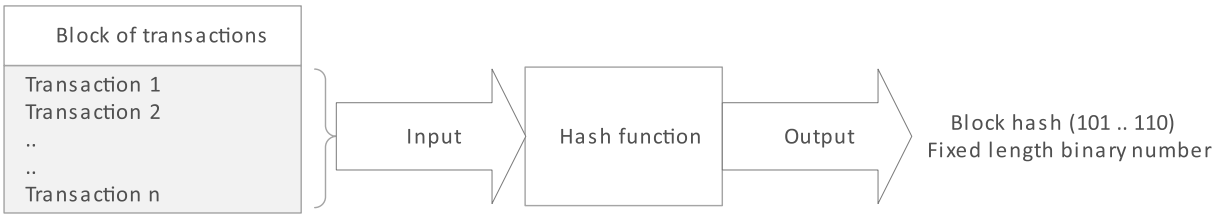

FIGURE 3. Constructing a block hash.

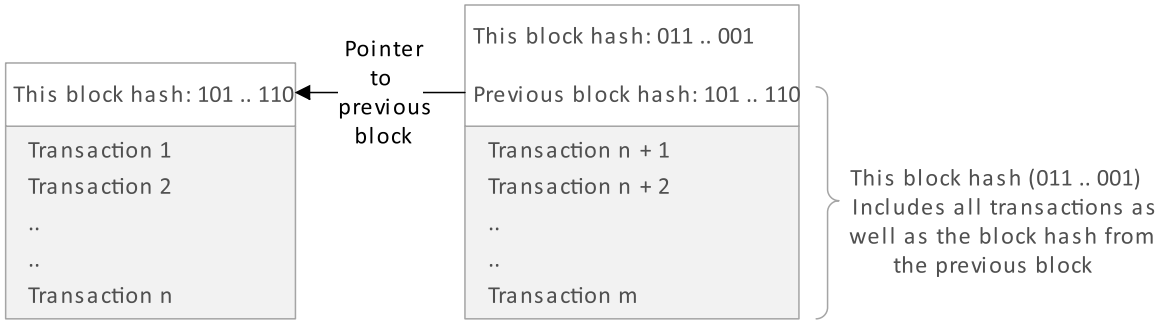

FIGURE 4. Blocks linked by a hash pointer.

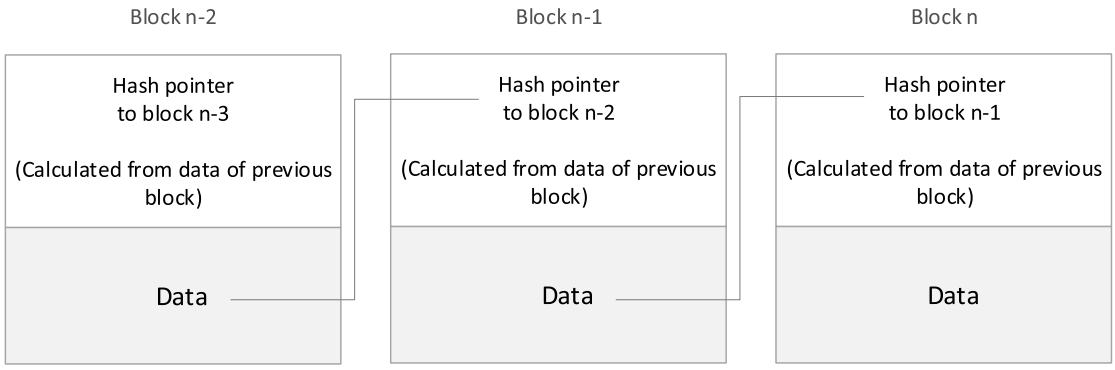

FIGURE 5. Simplified blockchain (Adapted from [9], [30]).

c: TRANSACTIONS

A blockchain needs not only to record financial assets. It may record other objects, such as intellectual property information, logistical and transportation data, executable smart contract code, etcetera. Generically, a transaction can be seen as an instruction or data inserted into a blockchain address. The address may belong to a participant in the external layer (Figure 2), for example, a user, or an entity in the secondary layer (Figure 2), for example, a smart contract [22], [33]. In reality, blockchain transactions may involve multiple addresses. For example, a cryptocurrency transfer may group amounts from multiple addresses of an owner and pay portions to multiple recipients. However, for simplicity, this paper will treat transactions as if only single addresses apply. Transactions must be atomical [16], meaning that they take place entirely or not at all.

Transactions originate from or between components of the external layer or external users, or components on the secondary layer, such as smart contracts or autonomous organisations, may trigger them. As an example of the transaction construction process, the case of an asset (i.e. cryptocurrency) transfer from a sender address to a receiver address is used. An address is a public key from a public key cryptography scheme. In practice, the receiver's address is often

represented by the hash of the public key, thereby enhancing its security and anonymity. It is only revealed once the receiver transacts with the address [31]. The sender constructs a transaction in a process consisting of three steps (Figure 6).

The first step is for the sender to specify the input address that contains its assets and the output address of the receiver. If the sender does not use all the assets in its input address, it also specifies a new output address for itself to receive the change. The reason for sending the change to a new address is two-fold. First, it closes out the existing addresses, the data of which can then be left out of the blockchain, keeping only the hash of the transactions in the Merkle tree. The following section discusses Merkle trees as part of the background on transactions. It makes it possible to store a lighter version of the blockchain in a process called ''Reclaiming Disk Space'' by [1]. Second, the new address enhances the security of the sender's remaining assets by using a new public-private key pair hidden by its hash.

In the second step, the sender adds additional data such as the amounts to be transferred, date and time information, etcetera. The data depends on the specific design of each blockchain system. Step two terminates by computing the hash of all the transaction data. By the collision-resistant property of hash functions, this hash serves as a unique identifier of the transaction data.

In the third step, the sender signs the transaction's hash with its private key. Blockchain systems often allow multiple parties to enter into transactions, in which case, the transaction signature uses distributed key generation and threshold signature schemes [41], [42]. Since a digital signature is the same length as the signed message, it saves space to sign the hash of the transaction identifier instead of all the transaction data. The security that the signature provides is two-fold. First, it proves that the public key owner (the sender's address) constructed the transaction and therefore is the rightful owner of the assets to be transferred. Second, it allows a verifier to check that the data in the transaction has not been tampered with while in transmission by computing the hash of the transaction data and using the public key verification algorithm (6).

Since objects in the secondary layer of a blockchain system may own addresses, they can automate the same three-step transaction construction process [22], [33]. Merkle trees store the transactions in each transaction block.

d: TRANSACTION BLOCK

A Merkle tree is a binary tree that builds in cryptographic tamper-proofing with hash functions [43] (Figure 7). Transaction data is stored in the leaf nodes, identified by each transaction identifier. Through a process of recursive hashing, pairs of nodes are hashed together to terminate in a single root hash. The root hash provides a tamper-evident signature of all the transactions contained in the leaf nodes [44]. Any attempt to change a single transaction will appear in the root hash value [43]. Traversing the transactions in a Merkle tree to search for a transaction has a computational complexity of O(n) if the leaves are unordered and O(log(n)) when ordered. Calculating the root hash has a computational complexity of O(log(n)).

e: HEADER BLOCK

A blockchain block must contain, at a minimum, three elements (depending on the design of the blockchain system, it may contain additional fields) to function as a tamper-proof transactions log. The first is a transaction Merkle tree, the second is a timestamp, and the third is a pointer to the previous block, also known as the previous block hash [18]. The Merkle tree root hash, timestamp and previous block hash constitute a header that can be hashed to produce an identifier or pointer to the block from the subsequent block [1], [22]. Figure 8 shows the structure of an elementary block.

This section discussed the basic design elements of the blockchain as the first primary layer component in a permissionless blockchain system. These elements include the cryptographic primitives of hash functions, public key cryptography and how they are used to construct transactions, and the transaction block and header block. The transaction block and header block, together with the block hash, make up a block in the blockchain. Figure 9 shows a more complete structure of a basic blockchain.

2) PEER-TO-PEER NETWORK

Message passing on a blockchain network takes place over a peer-to-peer (P2P) network, which forms the second component of the primary layer of a permissionless blockchain system. P2P is a well-known network topology. Its details are well described in the literature, for example, in [46] and [47] (Figure 10). Other than to point out that P2P refers to a decentralised topology [10] where nodes collaborate to share resources and services without a central authority to coordinate the processes, the details of P2P networks are deferred to the reader.

Since messages take time to propagate across a P2P network, it harms transaction processing times and the consensus process of a blockchain system. This lag is known as network latency and increases with network size [47].

3) CONSENSUS ALGORITHM

Consensus algorithms comprise the third component of the permissionless blockchain system's environment and exist because of two implications that follow from the P2P nature of permissionless blockchain networks. The first is the absence of a single authority to reconcile different versions of the blockchain that may exist between various participants in the system. The second implication is the assumption that not all nodes function correctly and that some may actively undermine the system. Blockchain consensus algorithms fall into two main categories: proof-based consensus algorithms and vote-based consensus algorithms [48]. The following two sections (E and F) give a thorough overview of the main ideas

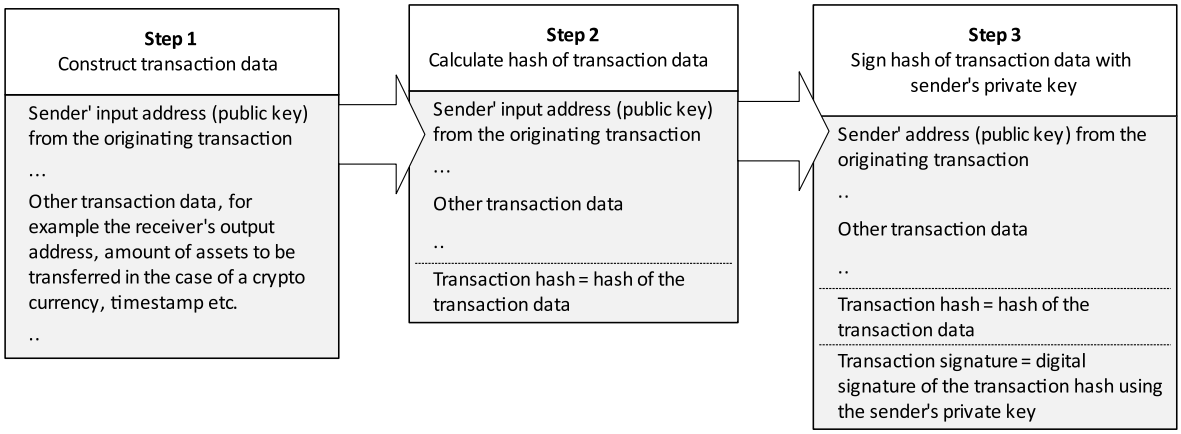

FIGURE 6. Generic steps in creating a blockchain transaction.

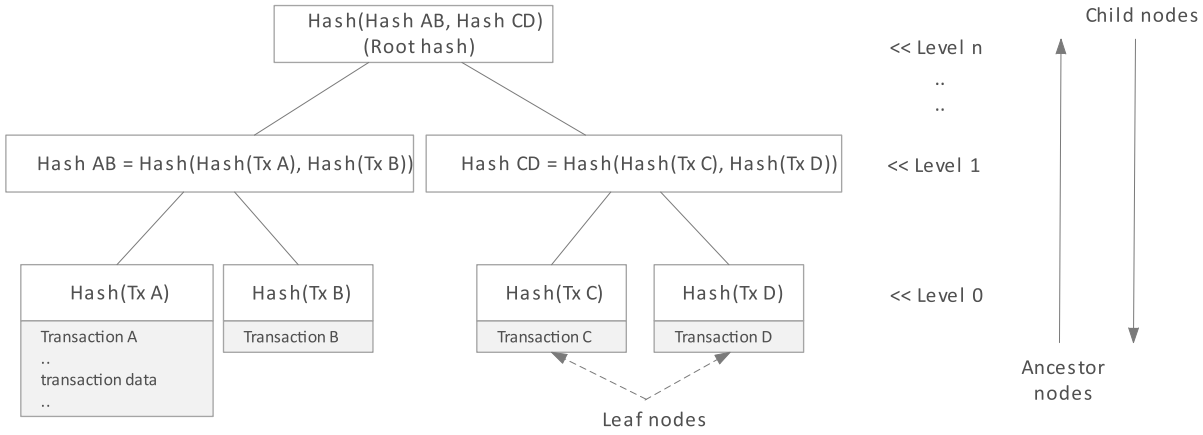

FIGURE 7. Transaction Merkle tree (Adapted from [43]).

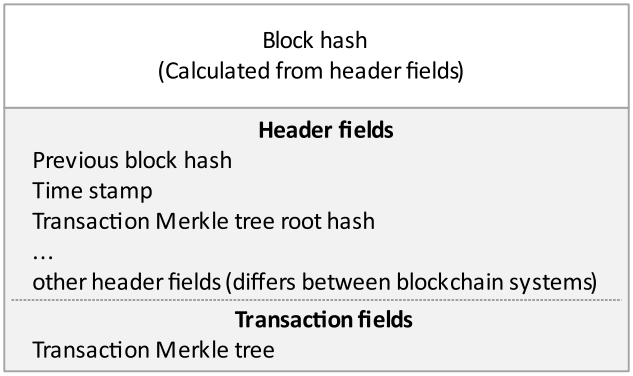

FIGURE 8. A simple blockchain block.

in each category, but they are not exhaustive because of the constant addition of new ideas.

E. PROOF-BASED CONSENSUS ALGORITHMS

Initially proposed for Bitcoin, proof-based blockchain consensus algorithms have seen many variants [48]. In general, proof-based consensus is probabilistic, meaning there is at least some probability that more than one honest participant may propose different but valid blocks. Refer to Appendix D for a summary of the proof-based consensus algorithms discussed in this section.

In **Proof-of-Work (PoW)**, miners vote on the correctness of a block by attempting to add a new block to the blockchain. The block that most miners add onto eventually becomes permanently embedded in the blockchain. A participant's voting power is proportional to its computational power as a proportion of that in the network because it relies on a process where participants solve a computationally hard puzzle to gain the right to add a new block [22]. Adding new blocks requires physical resources, so it becomes increasingly unlikely for an adversary to change a transaction in a block buried under an increasing number of new blocks [49].

PoW solves the double-spending problem and enables complete decentralisation [1]. However, it is computationally expensive. Repetitive hash calculations are required to find a valid block solution resulting in non-deterministic computational complexity. The high cost of block production protects the network from distributed denial of service attacks

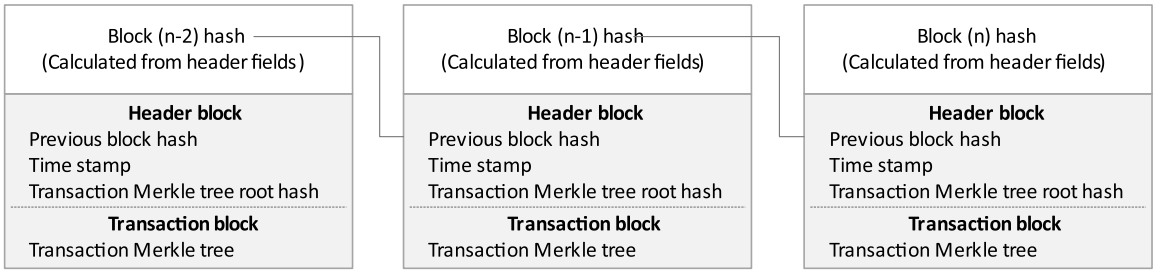

FIGURE 9. Structure of a basic blockchain.

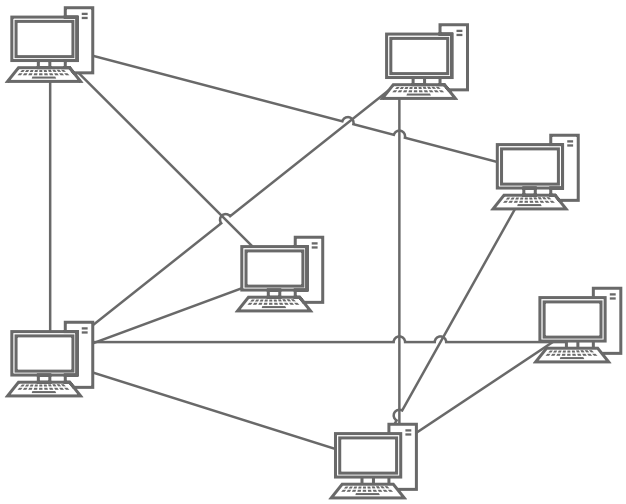

FIGURE 10. Peer-to-peer consensus network topology [102].

(DDoS) [23]. The disadvantages include that it is only secure under the assumption that most (51%+) of the computational power controlled by miners is honest [22], [49], [50], results in high energy consumption [9] and are vulnerable to miners that collude. Large-scale collusion (pooling of resources) can undermine the decentralised nature of PoW [51], [52]. PoW-type blockchain systems have to trade off network delay and block size. It is one reason for the limited adoption of Bitcoin as a payment system [47].

Proof-of-Stake (PoS) consensus algorithms were introduced to resolve the high energy consumption of PoW [19], [22]. It defers decision-making power for adding blocks to the blockchain in the proportion of assets owned (stake) by the participants [19], [49], [52]. During a block round, one of the existing coins in existence is chosen randomly, and the owner of the coin may add a new block to the blockchain. Stakeholders that own more coins are more likely to be selected [49], [52]. To increase the unpredictability, some additional form of randomisation may apply. For example, by considering the number and age of the assets [9], [53]. Computational complexity is not reported, but a popular algorithm for choosing a random coin, known as Follow the Satoshi, requires traversing the blockchain or portion thereof to establish the current owner of a minted coin; this process is linear in the number of blocks in the blockchain.

The advantage of PoS is that it solves the energy use problem of PoW [9], and it contains an inherent security measure since a participant that undermines the blockchain undermines its own assets [49], [52]. The staking process is automated and does not require the active involvement of participants to vote for block producers [26].

Disadvantages are its vulnerability to low-cost attacks because it is easy to create invalid blocks [9], which may lead to the Nothing-at-Stake problem, where participants perpetuate multiple forks on the blockchain, delaying ultimate consensus. Furthermore, when block proposers are selected based on both size of their stake and the age of coins, it may open the door to coin age accumulation attacks [53]. The fair initial distribution of coins may be problematic because participants that act rationally will seek unfair distribution of digital assets [49]. Similarly to mining pools in PoW, participants in PoS can collude by creating staking pools that may undermine system security [49], [52]. Individual participants may buy assets to increase their probability of proposing new blocks, causing shocks in the market, placing upward pressure on coin values [52] and ultimately reducing the participation of small asset holders in the block creation process. This type of wealth accumulation may lead to a degree of centralisation around large asset holders [54].

To address the challenges in transaction processing times of PoW and PoS, **Delegated Proof-of-Stake (DPoS)** responds by selecting a small subset of participants as witnesses to validate new blocks. This smaller validation pool can act more efficiently than the whole group of stakeholders [55]. DPoS blockchain systems do not assign the right to mine a block directly in proportion to a participant's share of ownership of assets but instead assign the right to elect delegates who must construct and validate new blocks on behalf of all participants [50], [52], [56]. DPoS blocks are created at set maintenance intervals by a committee of witnesses who serve for that interval. New witnesses are selected for each new maintenance interval, during which each witness has a turn to create a new block [57].

The advantages of DPoS are that it addresses the incomplete decentralisation found in PoS because of the cumulative influence of small stakeholders. No one is, therefore, wholly excluded from participation, making DPoS more democratic

than PoS [26], [56]. Large transaction volumes are possible with DPoS because the creation of new blocks is scheduled and needs little computational power [50], [54]. The small sub-section of participants required for consensus allows a higher transaction throughput, which makes DPoS more scalable than PoW and PoS [26]. DPoS provides a mechanism for changing the algorithm in the same way block creation happens. A committee selected, based on their proportion of stake, can also propose changes on behalf of the network [54].

The disadvantages of DPoS are, first, that the influence of participants ultimately stems from their stake and does, therefore, not completely improve on PoS [26] and second, DPoS relies on the owners of the digital assets to take a more active interest in governing the system [26], [54].

Proof-of-Elapsed-Time (PoET) depends on a trusted execution environment (TEE) to ensure sequential block creation. The TEE assigns participants a waiting time, and they announce blocks as soon as their waiting time expires [48]. When participants submit a new block, they also publish the random waiting time received from the TEE for verification by other parties [22], [58].

PoET has the advantage of energy efficiency because very little computation is required to add new blocks. While not reported, the bulk of computation will presumably result from constructing and verifying blocks with complexity linear in the number of transactions per block. The random waiting times are assigned fairly so that each participant has an equal chance of proposing new blocks.

The main disadvantage of PoET is its reliance on a trusted third party which makes it unsuitable for permissionless blockchain systems [26], [54]. The authors in [58] have also shown that PoET is vulnerable to a relatively small proportion of colluding participants (O[$\log \log n / \log n$] where n is the number of participants).

Nonlinear Proof-of-Work (nlPoW) [59] improves PoW by recognising that PoW assigns the same computationally hard puzzle to each participant in the blockchain network. nlPoW randomly gives a unique puzzle to each miner during each block round, making the probability of successfully mining a block nonlinear to its proportion of the computational power in the network. By disrupting the ability of participants to estimate their expected income from investment in computational resources, it aims to prevent the computational arms race that is currently found in, for example, Bitcoin.

nlPoW still requires repetitive hashing to find valid block solutions resulting in non-deterministic computational complexity.

PoW with a Cuckoo Hash Function [60] aims to find cycles of a specific length in a graph to produce a hash value. This process relies on Random Access Memory (RAM) rather than computational power as a resource for the mining process. The RAM usage scale with graph size and adjusts according to the historical block solution rate, similar to the difficulty adjustment of PoW. The advantage is that RAM does not benefit from the same economies of scale as processing power and should therefore dampen the competition between participants to invest in more infrastructure. The disadvantages are that one resource (electricity) is exchanged for another (RAM), and resource use is non-deterministic.

King [61] proposed the concept of **PoW by Searching for Prime Chains**. The argument is that PoW computations could operate for scientific gain, giving them real-world value [57]. He proposed that participants must search for prime chains instead of hash values to be stored on the blockchain and used for further research by mathematicians [62]. Like all PoW algorithms, it still needs a guess and test strategy, which is non-deterministic and inefficient.

Proof-of-Luck (PoL) [63] is an extension of PoET by using a TEE to require that a fixed amount of time must pass, during which participants may produce new blocks. PoL includes a parameter called luck, a random value generated by the TEE. Participants broadcast their new blocks when the waiting time expires, and the network selects the blockchain with the most significant sum of luck over all the blocks as the valid chain. The waiting time allows the propagation of the latest valid chain through the network. PoL's reliance on a TEE requires a trusted third party to provide the TEE and is, therefore, unsuitable for decentralised consensus systems. Computational complexity is not reported but presumably results from constructing and verifying blocks with complexity linear in the number of transactions per block.

The authors in [64] proposed **Proof-of-Human-Work (PoH)**, a cryptographic puzzle that humans can solve with moderate effort but is very hard for computers. It is similar to the CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) concept. However, advances in artificial intelligence (AI) may compromise the algorithm.

Proof-of-Burn (PoB) [65] requires that participants buy participation rights by sending some of their digital assets to a burn account from which they are unretrievable. The probability of the right to mine a block is related to the number of assets burned. Every burn transaction creates a hash, and the "best hash" (smallest) created during a block creation interval earns the right to add a new block. Participation, therefore, comes at a cost to the participant but does not require large amounts of electricity to be wasted. However, resources in the form of digital assets still go to waste. It also skews participation towards participants who can afford to employ more assets [66].

Proof-of-Space (PoSpace) [67] uses the same idea as PoW but requires that participants commit hard disk space instead of computational power. Attacking the blockchain, therefore, incurs a cost. The probability of adding a new block is proportional to the amount of disk space the participant can allocate. PoSpace trades one resource use (electricity in PoW) for another (disk space), which makes it difficult to assert that one consensus algorithm is better than the other.

Proof-of-Importance (PoI) [68] is similar to PoS. It uses the ownership of digital assets to select the participant to add a new block. It adds additional constraints on the participant's stake, like the period of ownership of the digital asset

and the transaction activity of the participant. PoI does not require a large amount of computational power and limits the advantages of coin hoarding, which is a drawback of PoS. However, participants can still choose strategies which skew the probability of influencing block creation to their advantage.

Proof-of-Responsibility (PoR) [69] is a PoW-type consensus algorithm that relies on trusted third parties to detect and intervene when identifying suspicious behaviour from participants. It depends on third parties to define suspicious behaviour and does not promote decentralisation.

Proof-of-Authority (PoA) [26] is a blockchain consensus algorithm that defers transaction validation and block addition to trusted, publicly verified parties. The argument is that a large enough network of trusted parties can overcome an attack by a compromised minority. PoA is efficient and can scale easily, but it is not decentralised. The computational complexity varies according to the implementation of the algorithm. If a single machine is to be trusted, it only needs to verify the transactions, resulting in linear complexity in the number of transactions per block. If multiple authorities need to vote on blocks, it may result in quadratic complexity in the number of authorities.

Dfinity [70] uses a built-in pseudo-random number generator to select participants that may add new blocks. It divides block addition into a fixed number of block additions, called epochs. Each epoch opens with a special block where participants interested in joining the block addition process register their intent to participate. The registration process includes a distributed key generation protocol with threshold signatures, allowing registered participants to operate a pseudo-random beacon for the duration of the epoch [71]. Dfinity can be described as a semi-permissionless blockchain system. It combines the security gains of permissioned blockchain systems with some degree of decentralisation, characteristic of permissionless blockchain systems.

F. VOTE-BASED CONSENSUS ALGORITHMS

Voting-based consensus algorithms require that participants in the verifying network are all known. These algorithms ensure that the system functions even if some participants malfunction, for example, when they crash or act maliciously [48], [72]. In [48], the authors distinguished two types of fault tolerance. First, crash-resistant fault tolerance lets the network function if a proportion of the participants are unresponsive, presumably crashed. Second, Byzantine fault tolerance functions even if some participants act maliciously.

Crash-Resistant Fault Tolerance was proposed by [73] as the Paxos protocol that remains operational as long as $(N/2)+1$ participants functions correctly [74]. [75] simplified the consensus proses (Paxos is notoriously complex) in an algorithm called Raft, which is easier to understand. Initially, Paxos and Raft were used in distributed database systems.

Raft works across a network of independent participants by establishing an election process for a leader that serves for a designated time interval. While the leader remains functional, it dictates the consistent state of the system, and all other participants (called followers) replicate the leader's state. Once its term expires or if it malfunctions, another leader is elected [75]. In blockchain consensus, the leader constructs the new blocks and sends them to the followers. Blocks are indexed so followers can request an updated sequence of blocks if it misses parts of the communication (go offline). If $(N/2)+1$ followers confirm that the block received from the leader is valid, the leader will commit the block to the blockchain and instruct the followers to do the same [48].

Practical-Byzantine-Fault-Tolerance (PBFT) [76] is a practical method to achieve consensus in asynchronous distributed systems when at least $[(2/3)N+1]$ participants are honest. It therefore allows that some participants are malicious. It relies on the election of a leader that controls the consensus process until its term expires or it malfunctions [22], [76]. The consensus process requires the transaction gathering and validation and the commit phases. During transaction gathering and validation, followers gather and validate transactions and then send them to the other participants, including the leader. Once the leader has enough transactions, or after a set interval, it constructs a block. The leader then broadcasts the new block to the followers, who store and forward it to others. Each follower that receives identical blocks from two-thirds or more other followers will announce that it is ready to commit the block. If it again gets the same commitment from two-thirds or more of the other followers, it will commit the block [48]. Transactions are confirmed immediately when recorded on the blockchain. Consensus in PBFT is, therefore, explicit, with no need to wait for more new blocks to increase confidence in a transaction's validity [26].

Federated Byzantine Agreement (FBA) [77] is applicable for networks where joining is not restricted. Subsets of participants that trust each other may cooperate exclusively in groups called slices to validate transactions among each other. Global consensus will emerge as long as the slices of participants overlap so that all participants are eventually connected.

Delegated Byzantine Fault Tolerance (dBFT) [57] functions on the principle of two participant types, ordinary participants and bookkeeping participants. Block addition defers to bookkeepers that act on behalf of ordinary participants. The blockchain is appended if a large enough proportion of the bookkeepers agree on a new block.

G. PUBLICLY VERIFIABLE RANDOMNESS

This paper focuses on permissionless blockchain systems that require a decentralised consensus mechanism. It is a problem for which the proposed solution strategies lie within the proof-based group of consensus algorithms. While consensus algorithms may differ in execution, they all share the same

underlying principle: they try to make the block addition process random [11]. Permissionless blockchain consensus can, therefore, fundamentally be viewed as a problem in publicly verifiable randomness. The consensus algorithm in a permissionless blockchain system can be seen as a pseudo-random number generator for selecting new block proposers.

Constructing systems that produce publicly verifiable random numbers is fundamental to blockchain systems [78]. The authors in [78] note that there are five desirable properties that publicly verifiable random numbers must satisfy. First, they must be unpredictable in the sense that no party must be able to know anything about a generated number before it is revealed. Second, the correctness of a random number, when published with verifying information, must be publicly verifiable by any party. Third, they must be tamper-resistant, meaning that the generation process must not be subject to interference from any party. Fourth, the generation process must be scalable, ideally not exceeding $O(n)$ where n is the number of participants in the system. Last, the generated number must be uniformly distributed if the generation process has at least one honest participant (the honest minority property).

Many methods exist for constructing publicly verifiable random number generators. As studied by [79], the output of **cryptographic hash functions in PoW** blockchain systems produces random public beacons. PoW algorithms, however, complete in non-deterministic polynomial time, which from a scalability perspective, makes them unsuitable.

Scrape is a protocol proposed by [80], which uses a distributed ledger and publicly verifiable secret sharing [81] to produce a random beacon between a set of participants. Scrape scales badly with cubic computational complexity.

RandHound [82] provides unique, publicly verifiable random numbers to each member of a group of participating stakeholders. It is based on a commit/reveal scheme that uses publicly verifiable secret sharing [81] and threshold signatures [42]. **RandHerd,** also by [82], is a further development of RandHound. It provides a stream of publicly verifiable but non-individualised random numbers to function as a random beacon. RandHound and RandHerd are both quadratic in their computational complexity.

HydRand is a protocol by [83] designed to function in a permissioned environment and produces a publicly verifiable random beacon. It uses publicly verifiable secret sharing [81] and random leader selection to operate and has quadratic computational complexity.

Dfinity is a blockchain consensus protocol by [70] with a built-in pseudo-random number generator that forms the basis for selecting new block proposers. It employs distributed key generation and Boneh-Lynn-Shacham (BLS) threshold signatures [84] to create a random beacon. Participants are allowed to join or leave the network but must register their intention to participate for a specific epoch. A special opening block at the beginning of each epoch allows would-be participants to register their intention to join or depart from the block addition process. According to [41], the distributed key generation protocol at the start of each epoch exhibits quadratic computational complexity. However, the repeated signature process for random number generation throughout the rest of the epoch is linear [70].

Randao is a protocol that leverages the Ethereum blockchain system to deliver a publicly verifiable random beacon [85]. The random number generation process functions in rounds where participants share the hash values of locally produced seeds on the blockchain. The seeds themselves are known only to each participant and are secret. A round ends when all the seed hashes are registered. Participants then reveal their seeds, which are combined to produce a single random value. Randao is linear in computational complexity but vulnerable to look-ahead attacks.

Ginar allows individual participants to generate a random number in cooperation with a group of independent participants on a blockchain system [78]. Every participant uses a verifiable random function [86] to determine if they meet an eligibility threshold to participate in the generation process of the requesting individual. Eligible participants encrypt a secret with the requester's public key and store it on a blockchain. The requester decrypts the sum of the encrypted secrets through the homomorphic property of Elgamal encryption. The decrypted sum constitutes the requester's random number. Ginar exhibits linear computational complexity [78].

Single Secret Leader Election (SSLE) by [87] enables the selection of a random block proposer (leader) in a blockchain system. It uses threshold fully homomorphic encryption [88] to obscure the leader's identity until it reveals itself. Obfuscation protects the leader from a denial-of-service attack by a malicious party. SSLE requires a public randomness beacon, which the authors do not specify. The leader selection process consists of two phases. The first is a setup phase, where participants register their intention to participate. The second phase consists of repeated election rounds that continue until there is a change in participants [87]. The authors do not discuss the computational complexity, but most of the computational load lies in registering the participants during the setup phase. Presumably, this phase has quadratic complexity, but the initial computational cost is amortised over many election rounds.

Verifiable Delay Functions (VDFs) [89] solve the problem of forced failure by an adversary in multi-party commit/reveal schemes. It calculates an output using a sequential set of steps that is time-consuming and cannot be executed in parallel. In random beacon construction, it may be used as a technique to force all parties to reveal their commitments before being able to calculate the outcome of the final random value. The correctness of the output can, however, be verified efficiently. VDFs use three algorithms, namely, *setup*, *evaluate* and *verify*. *Setup* determines the environment for evaluation of the VDF. *Evaluate* sequentially computes the output and its proof of correctness. *Verify* confirms that a prover completed *evaluate* correctly. As shown by [90], VDF *evaluate* is linear in the output's bit-length and *verify*

is constant in time complexity. However, it is unclear how to run setup in a distributed manner efficiently and may require a trusted source that does not participate in the consensus process [90].

Verifiable Random Functions (VRFs), first proposed by [91], are an application of public-key cryptography that allows a prover to compute a pseudo-random number together with proof of its correctness from a publicly known input, using a secret key. The correctness of the pseudo-random number can be verified by a verifier using the prover's public key, the pseudo-random number and the proof. The pseudo-random number has good random number properties, and the probability of distinguishing it from a random number is negligible [78].

B-Rand [92] and **Transient Random Number Seeds** (TRNS) [93] are methods for embedding random number seeds into blockchain transactions. Participants can then use these random number seeds in pseudo-random number (PRN) generators for blockchain processes that require publicly verifiable random selection, for example, selecting a new block proposer. B-Rand assigns the random number seed to the sender (creator) of the transaction using hash functions, homomorphic encryption and public-key signatures. TRNS assigns the random seed to the transaction recipient using hash functions and public-key signatures. Random number seeds embedded in blockchain transactions using B-Rand and TRNS are confidential, tamper-resistant, unpredictable, collision resistant and publicly verifiable. Both methods scale well and are linear in the number of transactions per block.

III. PROPOSED APPROACH

We now propose a method for using permissionless blockchain systems as pseudo-random number generators and apply the results to obtain a consensus algorithm named proof-of-publicly verifiable randomness (PoPVR). PoPVR randomly selects new block proposers from the participants in the blockchain system. The method uses verifiable random functions (VRFs), discussed in II G, to embed pseudo-random number seeds in blockchain transactions deployed by transaction recipients during each block round in a block selection lottery to "win" the right to propose a new block.

A. BLOCKCHAIN SYSTEMS AS PSEUDO-RANDOM NUMBER GENERATORS

Pseudo-random number (PRN) generators are deterministic random number generators. They produce a sequence of pseudo-random numbers that are deterministically dependent on an initial seed value [94]. Figure 11 shows the generic structure of a PRN generator.

According to [94], PRN generators must have three essential properties. First, the output must have good statistical properties that resist replay and correlation-based attacks. Second, an attacker must not be able to compute predecessors or successors from a known series of outputs of a PRN generator. Third, an attacker must be able to determine a previous random number or previous seed of a PRN generator.

The properties are achieved through two transformation functions, as shown in Figure 11. The first function, Ψ, uses the current seed value, also known as the internal state of the PRN generator, to produce a pseudo-random number. The second function, Φ, transforms the current internal state (current seed value) into a new internal state (new seed value). As required by the third property, if Φ is a one-way function, its input cannot be determined from its output, as is the case with, for example, hash functions. This one-way property prevents an attacker from calculating the last seed, preventing replay attacks. If Ψ and Φ are one-way functions, attackers cannot compute the internal state from any random number or previous internal states. [94] notes that hash functions, discussed in II-D 1, are good candidates for Ψ and Φ as long as they are not the same function because that would imply that the internal state of the PRN generator is the same as the last output.

$$\Psi \neq \Phi \xrightarrow{\text{implies}} \mathrm{r_n} \neq S_{n+1} \tag{7}$$

To build a blockchain pseudo-random number generator (BPRN generator) that produces publicly verifiable random numbers, we adapt the generic PRN generator from Figure 11 using ideas from TRNS and VRFs discussed in the previous section (Figure 12).

For illustration, we consider the verifiable random function RSA-FDH-VRF-SHA256 as defined by [95]. A VRF is a family of algorithms, namely, *generate* for key generation, *prove* for producing a pseudo-random output and its proof (representing Ψ in Figure 12) and *verify* for establishing the correctness of the output. The combination of the VRF and the SHA256 hash function (representing Φ in Figure 12) make up the components of the BPRN generator. The output from the BPRN is used to construct a consensus algorithm (PoPVR) for adding new blocks to the blockchain. POPVR consists of four functions: seed, propose, verify, and chain selection. These functions are discussed individually in III-A 1 to III-A 3.

1) SEEDING THE BPRN GENERATOR

Each recipient of a blockchain transaction generates the seed information it will require for the future production of publicly verifiable random numbers and new block proposals. The seed information consists of two components. The first component is an RSA key pair from the RSA cryptosystem. The RSA key pair is generated by the VRF's generate algorithm that returns the public and private keys $(\mathrm{VRF-Key}_{public}, \mathrm{VRF-Key}_{secret})$.

$$VRFgenerate\,() \rightarrow Key_{public}, Key_{secret} \tag{8}$$

The RSA cryptosystem here is the RSA-FDH-VRF-SHA256 above, but there are also other constructions for VRFs, for example, from elliptic curve cryptosystems [95].

The second component of the seed information is a local random value (S_0) obtained from the output of a seeded SHA256 hash function. S_0 serves as public "salt" for the

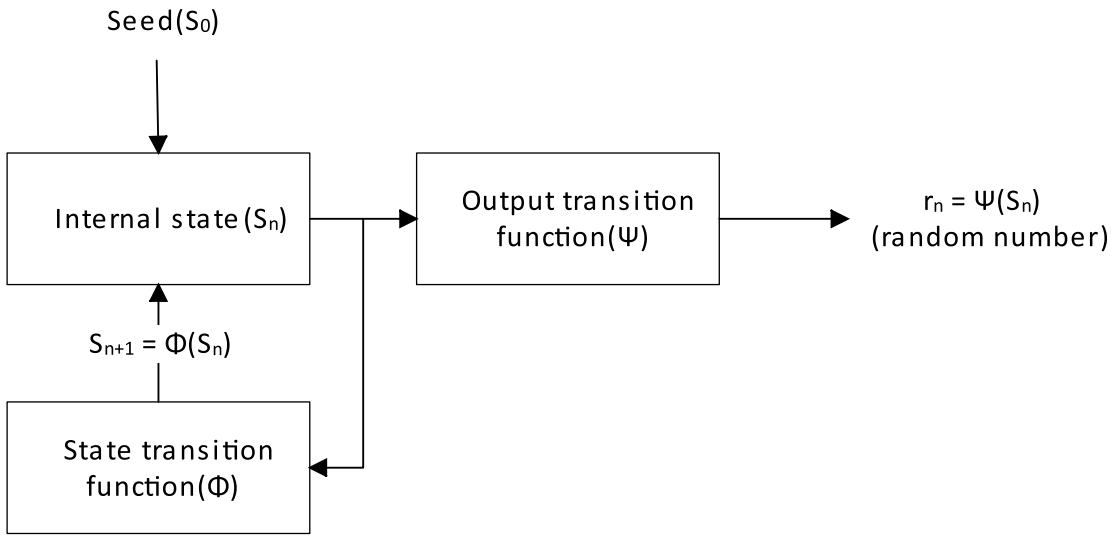

FIGURE 11. Generic design of a pseudo-random number generator [94].

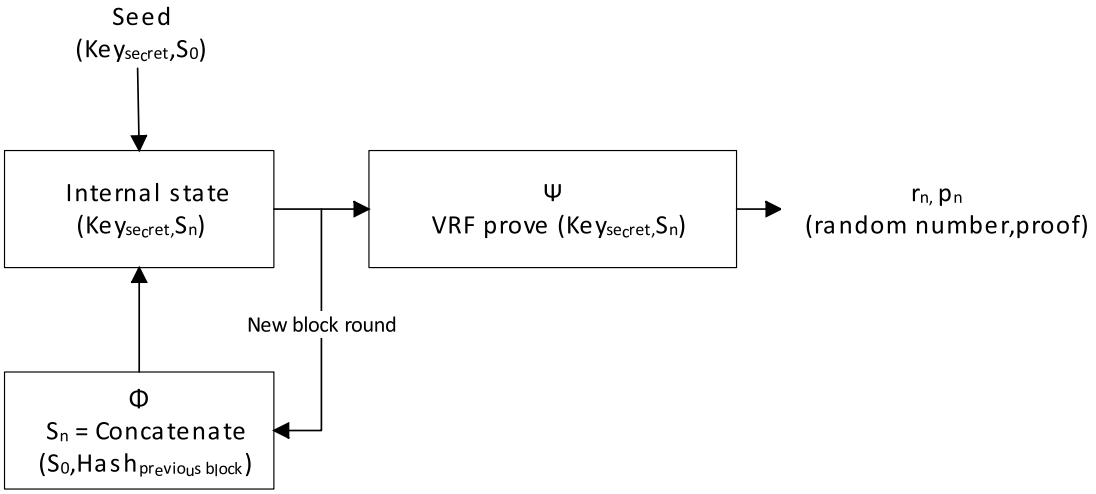

FIGURE 12. Adapted blockchain pseudo-random number generator.

BPRN generator to make dictionary attacks against the secret key of the VRF more difficult because, while S_0 is public, it is collision-resistant by the properties of hash functions and, therefore, different for each recipient. Note that S_0 is produced locally by the participant. Its production method is unenforceable by any authority, but it is in the interest of each honest participant to produce an S_0 indistinguishable from random. The construction of S_0 can be adapted from the method proposed by Rivest in [96] for seeding a hash function. We propose and tested the following method whereby recipients can obtain S_0: A string of 100 characters was generated by rolling a 26-sided die (each side representing a character from a to z) 100 times using a simulation in C#. The string is concatenated onto the system date and time at the moment of the 100th roll of the die. The SHA256 hash of the string gives S_0.

$$S_0 = \text{SHA256}\,(\text{concatenate}(\text{string}, \text{dateTime})) \qquad (9)$$

The method for obtaining S_0 was subjected to tests for both randomness and conformity to the uniform distribution. Randomness was tested using the US National Institute of Standards and Technology's (NIST) Statistical Test Suite for Random Number and Pseudorandom Number Generators for Cryptographic Applications (NIST Test Suite) [97] (Appendix A). For testing the randomness of the proposed S_0 generator, each S_0 was converted to a 32-bit unsigned integer to give S_0'. 100 bitstreams containing 3125 S_0' seeds was produced so that each bitstream contained 100 000 bits. The number and length of the bitstreams were based on the recommendations of Rukhin et al. for use in the NIST Test Suite. From the results in Appendix A, we conclude that the adapted Rivest method provided a good source of randomness for S_0. Likewise, by transforming each of the 100 samples of 3125 S_0' seeds to decimal values on the interval (0,1), the Kolmogorov-Smirnov Goodness of Fit Test (KS-Test) [98],

was applied to test for uniformity. The results in Appendix B show that the output from the Rivest method can be regarded as uniformly distributed.

In practice, the seeding information must be generated by the recipient simultaneously with its receiving address. In addition to its receiving address, the recipient uses its wallet application to generate the RSA key pair for its VRF and S_0. The recipient then gives the sender, which will be the creator of the transaction, its address, $\mathrm{VRF}-\mathrm{Key}_{public}$, S_0 and the signature (5) of S_0 with its private key:

$$\mathrm{Sign}\left(\mathrm{Key}_{\mathrm{secret}}, S_0\right) \rightarrow \mathrm{Signature}_0 \tag{10}$$

When the sender constructs the transaction, it adds the $\mathrm{VRF}-\mathrm{Key}_{public}$, S_0 and $Signature_0$ to the transaction fields before signing and broadcasting it on the blockchain network. Figure 13 shows the BPRN seeding process.

While key generation runs in linear time to key size, hash functions run in linear time to input size, and signature algorithms run in linear time to the message length, these three processes can be regarded as constant time processes because all recipients use the same key size and input size for computing S_0. The sender can create the seeding information in the transaction in constant time.

The seeding process described above imparts the following properties on each participant's BPRN seed: First, the seed is *confidential* because the private key of the VRF is confidential; only the recipient knows it. Second, the seed is *tamper-resistant*, as it is encoded in the recipient's transaction on the blockchain; therefore, no party, including the recipient, can alter it. Note also that the requirement to include $Signature_0$ in the transaction, allows any party to verify that the sender has correctly recorded $VRF-Key_{public}$ and S_0 as received from the recipient, using (6). Furthermore, since pseudo-random number generators depend deterministically on their seeds, these properties of confidentiality and tamper-resistance are also eventually valid for the pseudo-random numbers generated.

2) PROPOSING NEW BLOCKS

With each new block round, a recipient can act as a block proposer if it estimates that it possesses a lottery ticket with a high probability of winning the round. It does so by calculating a lottery ticket (r_n) using the VRF's prove algorithm (Ψ). The VRF-prove algorithm requires two inputs, namely a secret key ($VRF-Key_{secret}$) and an input value (S_n) and returns a random value (r_n) and a proof of correctness (p_n). In the case of using the prove algorithm from the RSA-FDH-VRF-SHA256 function, (r_n) is a 256-bit binary number.

$$\mathrm{VRFprove}(\mathrm{Key}_{\mathrm{secret}}, S_n) \rightarrow r_n, p_n \tag{11}$$

Furthermore, an adversary cannot compute the output without knowing the secret key. Knowing the recipient's public key $VRF-Key_{public}$, input value (S_n), output (r_n) and proof (p_n) allows any verifier to be satisfied that the recipient executed the VRF correctly and that the lottery ticket is indistinguishable from random [78], [91]. The input value to the VRF (S_n) is the concatenated values of S_0 and the hash of the block the proposer intends to extend ($Hash_{previousblock}$).

The proposer adds four fields to the block header when publishing a candidate block. First, a reference to the transaction in which it received its VRF public key and S_0 from the sender (the reference transaction). Second, its lottery ticket (r_n). Third, the proof (p_n) and last, the running mean $(\overline{r_n})$ of the lottery tickets of the previously successful blocks. This last requirement is essential in the chain selection process, and while it can be calculated directly from the blockchain, storing it in the header of each block saves computation. Figure 14 shows the generic block structure.

Note that S_n is not published. It is calculated from S_0 and the previous block hash. S_n changes for every proposer with each new block round. Therefore, if proposers publish candidate blocks that do not survive on the blockchain, an attacker can only launch a dictionary attack on the secret key of each proposer individually and not on all the proposers in a single attack.

From the perspective of the block proposer, computing and adding the lottery information to the block header is a constant time operation. Block construction, however, still requires that all the transactions are added to the transaction Merkle tree and therefore is linear in the number of transactions per block.

Lottery tickets have two important properties. First, they are *unpredictable* for the proposer of the new block and any other party. The reason is that the lottery ticket can only be computed by the proposer, as it owns the VRF secret key, and only after the previous block hash becomes known. It is helpful to think of every new block round as an update of the internal state of the BPRN generator. Second, lottery tickets are *collision-resistant* since the output of VRFs is collision-resistant.

3) VERIFYING LOTTERY TICKETS

By inspecting each proposed new block, a verifier can retrieve the proposer's VRF public key and S_0 (from the reference transaction), the previous block hash and the proof from the block header. Verification is a two-step process of reconstructing S_n and computing VRF-verify with S_n, the proof and the proposer's VRF public key:

$$\begin{aligned} &Step1 \\ &\mathrm{Concatenate}\left(S_0, \mathrm{Hash}_{\mathrm{previous\,block}}\right) \rightarrow S_n \qquad (12) \\ &Step2 \\ &\mathrm{VRFverify}\left(\mathrm{Key}_{\mathrm{public}}, S_n, r_n, p_n\right) \rightarrow true/false \qquad (13) \end{aligned}$$

Verification requires that the verifier checks the proposer's seed information in the reference transaction. This process is linear in the number of transactions per block. However, the VRF-verify algorithm can be considered $O(1)$ because Key_{public}, S_n, r_n and p_n are of the same size for all participants. In addition to the previous properties of confidentiality, tamper-resistance, unpredictability and collision resistance,

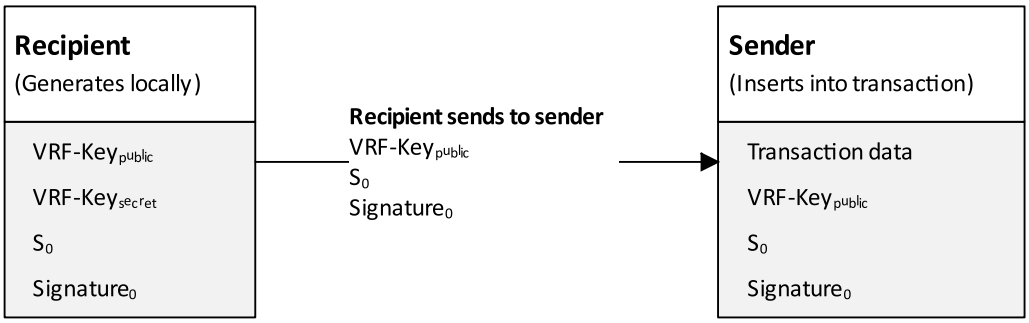

FIGURE 13. Seeding process for BPRN in PoPVR.

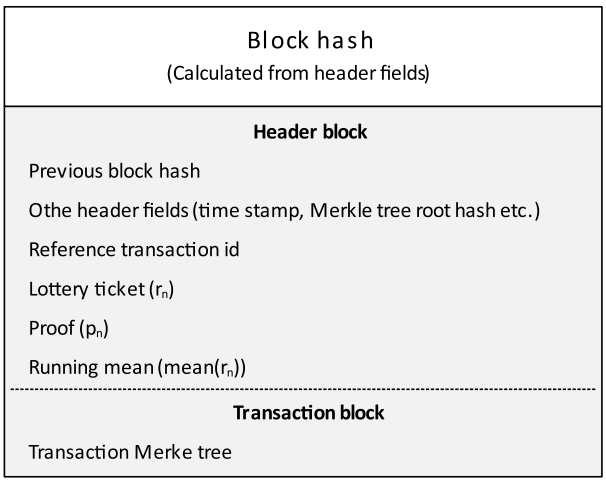

FIGURE 14. PoPVR block construction.

the verification process adds public verifiability to the lottery ticket.

4) CHAIN SELECTION

Nodes in the blockchain network select each new block to extend the blockchain from the candidate blocks proposed by the participating proposers for the applicable block round. From the set of candidate blocks, the block producing the lowest running mean of all the tickets in the blockchain (the lowest running mean rule):

$$\text{Selection rule} \xrightarrow{\text{select for}} \min(\bar{r_n}) \tag{14}$$

Comparing the running means of multiple blocks are linear in the number of candidate blocks a node receives, but this includes the verification of each lottery ticket and its complexity as per III-A 3.

Consider the samples of random numbers generated in Appendix B as simulations of the lottery tickets of new block proposers. It is possible because they share the same properties of the output from the VRF prove algorithm. In practice, the lottery tickets will be 256-bit pseudo-random numbers. However, the simulation substitutes 256-bit values for their decimal approximations as they provide a more intuitive understanding of the block selection process, albeit with some loss of resolution. Each sample summary contains the minimum value produced for the sample. Assuming that all 3125 sampled random values represent lottery tickets from newly proposed blocks, the blockchain network selects the block with the smallest lottery ticket. Appendix C summarises the minimum values (representing lottery tickets) and the running mean, represented graphically in Figure 15.

In a sample run of 100 samples with 3125 seeds per sample, the running mean equals 0,0003125337. Extending the sampling process to 100 samples with 10000 seeds each, the running mean decreases to 0,0000558115. This downward trend can be seen in Figure 15 when the running mean sequence stabilises from around sample number 65 onward. The running mean asymptotically tends towards zero.

IV. ANALYSIS

Analysing the functionality of PoPVR requires a few assumptions about the blockchain environment. These include some behaviours expected from honest nodes and the proportion of honest nodes in the network. These minimal assumptions are required to infer the possibility for an adversary to undermine the operation of the blockchain system. Up to now, the terms recipient, block proposer and node have been used somewhat interchangeably. Formally, a node maintains a blockchain replica and possesses the functionality to validate transactions and candidate blocks. Since the ability to propose a candidate block is limited to recipients of blockchain transactions, all nodes in a PoPVR blockchain system must be recipients. Furthermore, once a transaction is spent, the VRF seed expires so that only the active stakeholders in the blockchain system may act as nodes. While there is no way to force participation in a decentralised setting, and it is conceivable that, in practice, not all eligible parties may act as nodes, it is conceptually still useful to think of nodes as recipients with unspent transaction outputs.

From a participation standpoint, PoPVR function like a PoS blockchain network, with the notable difference that the size of asset ownership of the node does not skew the probability of successfully proposing a new block. It makes PoPVR useable in blockchain environments where blockchain assets, such as logistical or intellectual property recording systems, may not be objectively comparable.

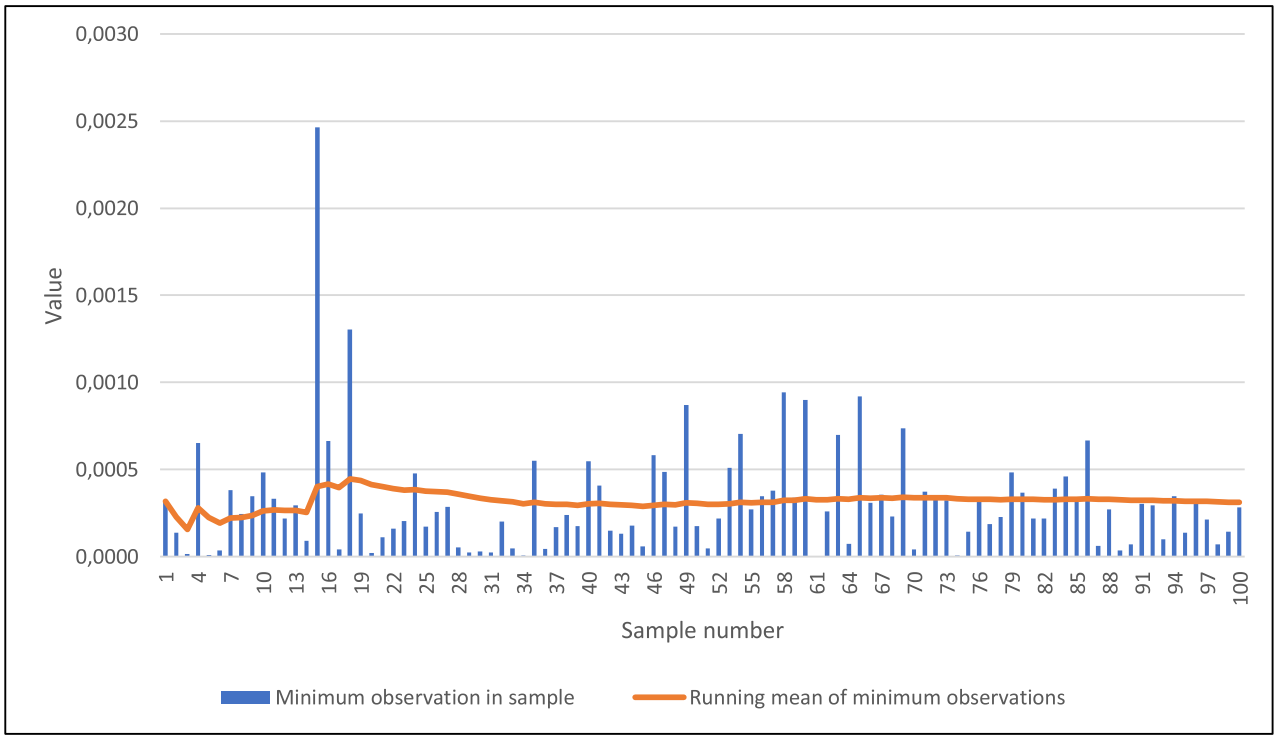

FIGURE 15. Sample minimum values and running mean of sample minimum values.

A. OPERATIONAL ASSUMPTIONS

The operation of a blockchain consensus algorithm is always subject to certain assumptions about the proportion of nodes functioning correctly as opposed to those which may malfunction or act maliciously. These assumptions are called trust assumptions or security assumptions [6], [48]. PoPVR assumes that at least 50% of nodes function correctly (are honest). In addition to the proportion of honest nodes, PoPVR also assumes the following specific properties about honest nodes:

i. An honest node forwards all valid transactions and blocks (messages) received from senders or other nodes immediately. Specifically, they do not withhold valid messages from other nodes or transactions from block inclusion.
ii. Honest nodes resolve forks at any point in the blockchain by following the route of the lowest running mean of lottery tickets.
iii. Honest nodes forward valid messages on average to at least two other honest nodes. It does not mean that nodes trust each other or that they have knowledge of the honesty of other nodes; it simply means that given that at least 50% of all nodes are honest, the probability of a message reaching other honest nodes grow (and reaches on average two honest nodes) as the number of nodes a message is passed to, grows.
iv. New messages (transactions or blocks) that enter the blockchain network are initially distributed evenly over all nodes. For example, if the network contains N nodes and M messages enter the network in a given interval, each node receives $\frac{M}{N}$ messages.
v. A final assumption is that there is a finite time interval (partial latency) l_i during which an honest node receives a message from another honest node and also a finite total time interval or total latency,

$$L = l_1 + l_2 + \ldots + l_T \tag{15}$$

in which a message reaches all honest nodes in the network. Again the honest node does not have to trust the sending node; it must simply receive a large number of messages and evaluate the validity of each message.

These assumptions allow two additional inferences about the evolution of information on the blockchain network (IV-B and IV-C).

B. EXPONENTIAL DECAY OF CANDIDATE BLOCKS

Suppose every proposer attempted to submit a candidate block to the blockchain network with each new block round. In that case, the valid new block (smallest lottery ticket) will be accepted by each honest node in a finite time interval, provided that the valid block was submitted to at least two honest nodes. The intuition for this statement comes from the exponential function [99] and the assumptions above. The number of honest nodes receiving the message with the smallest lottery ticket will grow exponentially with each partial latency interval. For example, if the rate of spread or growth rate (r) is at least two (each node messages two other nodes), then the number of nodes reached (N) after a finite

TABLE 1. Exponential spread of messages.

Latency interval (l_i)	Growth rate (r)	Number of nodes reached (N)
1	2	3
2	2	9
3	2	27
	...	
15	2	14 348 907

TABLE 2. Adversary's probability of success during successive block rounds.

Block rounds	Successes*	Probability
1	1	0,4990
2	2	0,2490
3	3	0,1243
4	4	0,0620
5	5	0,0309

* Adversary's probability of successful round = 0,499 (49,9%)

TABLE 3. Honest nodes' probability of success during successive block rounds.

Block rounds	Cumulative success*	Probability
5	1	0,1553
5	2	0,3119
5	3	0,3131
5	4	0,1572
5	5	0,0316
Cumulative probability		0,9691

* Honest nodes' probability of successful round = 0,501(50,1%)

number of partial latency intervals (L) is given by:

$$N = (1 + r)^T \tag{16}$$

Table 1 shows an example of the exponential spread of messages among honest nodes for the first 15 partial latency intervals. Note, however, that in theory, the possible number of partial latency intervals is unbounded.

It stands to reason that if the P2P network contains a finite number of nodes, all candidate blocks are eventually ignored in favour of the valid new block. The rate at which candidate blocks will decay is exponential. In practice, this idea is borne out by research from [100] that shows that new Bitcoin blocks are received by 90% of the nodes on the Bitcoin network in less than three seconds after being announced by the successful miner.

The number of candidate blocks among the network of honest nodes decreases as higher lottery tickets are ignored, and only the lowest lottery ticket is stored and forwarded. The total number of partial latency steps (L), which is also the total network latency time, required for the smallest ticket to propagate through the network is:

$$L = \log_b N, \text{ where } b = r + 1 \tag{17}$$

C. SECURITY CONCERNS

Like PoS blockchain systems, there are inherent problems associated with the low cost of block construction in PoPVR. Unlike PoW, where an adversary has to expend resources to construct a valid block, PoPVR is designed not to be resource intensive. This author [52] identifies two problems with the low cost of block construction in PoS blockchain systems that also apply to PoPVR.

The first problem is the nothing-at-stake problem, where all nodes submit candidate blocks with every new block round, even if the probability of success (where the proposer has a large lottery ticket) is negligible. It increases the number of forks on the blockchain, and the situation worsens as each fork grows. The ultimate cost to the blockchain system is that it delays consensus and creates uncertainty about the finality of transactions [52].

The previous section has shown that unsuccessful candidate blocks will decay exponentially over the network and that all the honest nodes in the blockchain network will become aware of the candidate block with the smallest lottery ticket that satisfies the lowest running mean rule in time L. While L grows with the number of nodes in the network, it does so only logarithmically to N.

The second problem identified by [52] is the double-spending attack. Imagine an attacker sending a transaction to the network representing a payment to a merchant for a physical good. Once the merchant delivers the goods, the attacker attempts to replace the block containing the transaction, thereby never relinquishing its asset.

For the attacker to be successful, it must create a block that successfully replaces the one containing its payment. If an attacker controls 49.9% of the nodes in the network, it will have a 49.9% chance of replacing the latest block in the blockchain.

Suppose the merchant was to wait for the transaction to be buried under several blocks before releasing the goods. In that case, the attacker must construct a chain of successful blocks to replace the block containing its transaction and the successive blocks.

Table 2, however, provides the merchant with the probability that the attacker can consecutively overwrite a series of blocks versus the likelihood that the honest portion of the network will prevent it.

Table 2 shows that an attacker that controls 49.9% of the nodes will have a slightly higher than 3% chance of successfully overwriting five consecutive blocks. In comparison, the probability that the honest proportion of nodes will interrupt its attempt by one or more blocks is almost 97%

TABLE 4. Results for the uniformity of p-values and the proportion of passing sequences.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-Value[3] (Uniformity)	Proportion	Statistical test
8	6	9	12	11	10	10	12	8	14	0.834308	98/100[1]	Frequency
13	9	8	15	6	8	10	8	7	16	0.289667	98/100[1]	BlockFrequency
9	7	11	12	11	13	11	9	10	7	0.935716	97/100[1]	CumulativeSums
12	6	12	5	12	7	12	10	16	8	0.304126	98/100[1]	CumulativeSums
14	9	9	8	14	14	9	8	8	7	0.616305	97/100[1]	Runs
11	13	7	12	10	7	6	11	8	15	0.554420	99/100[1]	LongestRun
10	10	5	9	16	11	13	7	10	9	0.514124	99/100[1]	Rank
12	8	14	5	8	11	10	13	10	9	0.699313	99/100[1]	FFT
7	11	15	10	12	10	8	11	9	7	0.798139	99/100[1]	OverlappingTemplate
11	6	11	9	13	12	8	11	7	12	0.834308	99/100[1]	ApproximateEntropy
2	1	1	0	4	4	1	0	2	1	0.008879	16/16[2]	RandomExcursionsVariant
2	1	1	2	1	1	3	4	1	0	0.066882	16/16[2]	RandomExcursionsVariant
1	2	1	3	1	2	0	1	4	1	0.066882	16/16[2]	RandomExcursionsVariant
0	2	3	2	3	0	1	2	1	2	0.122325	16/16[2]	RandomExcursionsVariant
1	1	3	2	1	1	2	1	3	1	0.350485	16/16[2]	RandomExcursionsVariant
1	0	2	2	2	2	0	3	3	1	0.122325	16/16[2]	RandomExcursionsVariant
0	0	2	1	5	0	0	3	1	4	0.000089	16/16[2]	RandomExcursionsVariant
0	1	1	4	0	3	1	1	0	5	0.000199	16/16[2]	RandomExcursionsVariant
1	2	3	2	2	1	1	1	0	3	0.213309	16/16[2]	RandomExcursionsVariant
0	3	0	3	0	2	4	1	1	2	0.008879	16/16[2]	RandomExcursionsVariant
0	2	1	3	2	1	2	1	1	3	0.213309	16/16[2]	RandomExcursionsVariant
0	0	1	2	4	2	3	1	2	1	0.035174	16/16[2]	RandomExcursionsVariant
0	0	0	2	0	2	3	2	4	3	0.004301	16/16[2]	RandomExcursionsVariant
0	1	1	0	0	6	0	3	2	3	0.000017	16/16[2]	RandomExcursionsVariant
0	0	1	2	2	2	2	6	0	1	0.000199	16/16[2]	RandomExcursionsVariant
0	0	2	2	2	1	1	3	4	1	0.035174	16/16[2]	RandomExcursionsVariant
0	1	1	2	0	2	0	4	5	1	0.000439	16/16[2]	RandomExcursionsVariant
0	0	1	1	2	1	4	1	3	3	0.017912	16/16[2]	RandomExcursionsVariant
4	10	13	6	9	9	19	6	12	12	0.051942	100/100[1]	Serial
7	9	7	9	10	10	11	13	15	9	0.779188	100/100[1]	Serial
9	8	15	9	10	6	9	12	8	14	0.616305	99/100[1]	LinearComplexity

1. The minimum pass rate for each statistical test (α=0.01), excluding the random excursion (variant) test, is approximately = 96 for a sample size = 100 binary sequences of 100000 bits each.
2. The minimum pass rate for the random excursion (variant) test is approximately 14 for a sample size of 16 binary sequences.
3. χ^2 Goodness of Fit Test for uniformity. Reject uniformity when $p < \alpha = 0.0001$. Else consider the sample as uniformly distributed.

* All tests performed using the US National Institute of Standards and Technology's (NIST) Statistical Test Suite for Random Number and Pseudorandom Number Generators for Cryptographic Applications (NIST Test Suite) [97]

(Table 3). The waiting time for the merchant to reach this level of certainty is five times L.

As in other blockchain systems, the probability of deleting a transaction from the blockchain becomes vanishingly tiny when buried below an increasing number of blocks.

D. FAIRNESS AND TRANSPARENCY

The Introduction (I) noted the importance of blockchain systems operating on open-source principles, as identified by [8]. It supports the purpose of transparency of the system and ensures that participants do not have to trust

TABLE 5. Results for Kolmogorov-Smirnov (KS) goodness of fit test (uniform distribution).

Sample number	1
Reject H0 - values are uniformly distributed (KS α = 0.05)	FALSE
No of seeds	3125
Mean	0,51061587
Min	0,00031693
Max	0,99978089
Variance	0,08488003
Std deviation	0,29134178
Sample number	2
Reject H0 - values are uniformly distributed (KS α = 0.05)	TRUE
No of seeds	3125
Mean	0,49097416
Min	0,00013566
Max	0,99926743
Variance	0,08530532
Std deviation	0,29207075
Sample number	3
Reject H0 - values are uniformly distributed (KS α = 0.05)	FALSE
No of seeds	3125
Mean	0,50645662
Min	0,00001484
Max	0,99997069
Variance	0,08309165
Std deviation	0,28825622
. . .	
Sample number	100
Reject H0 - values are uniformly distributed (KS α = 0.05)	FALSE
No of seeds	3125
Mean	0,50428000
Min	0,00028222
Max	0,99995265
Variance	0,08488643
Std deviation	0,29135275
Summary for 100 sample runs	
Reject H0 (Cumulative KS)	3 out of 100
Number of sample runs	100
Number of seeds per run	3125
Mean std deviation of 100 sample runs	0,28868343
Theoretical standard deviation of uniform distribution	0,28867513
Minimum seed value of all sample runs (smallest seed found)	0,00000186
Mean of minimum seeds for 100 sample runs	0,00021523

The minimum pass rate for the cumulative Kolmogorov-Smirnov test for 100 samples is 95/100 (Rejections < 5). For accommodation by the KS test, samples were converted from 256-bit hash output to decimal values between 0 and 1 and rounded to 8 decimal places, causing some loss of resolution.

TABLE 6. Minimum observations and running mean of minimum observations.

Sample number	Minimum observation in the sample	The running mean of minimum observations
1	0,0003169267	0,0003169267
2	0,0001356630	0,0002262949
3	0,0000148397	0,0001558098
4	0,0006515430	0,0002797431
5	0,0000078562	0,0002253657
	...	
96	0,0003184201	0,0003181691
97	0,0002127399	0,0003170822
98	0,0000711749	0,0003145730
99	0,0001429999	0,0003128399
100	0,0002822171	0,0003125337

so-called "black-box" components when interacting with the blockchain.

An essential feature of PoPVR is that the internal workings of every component are public knowledge. The BPRN is transparent and uses hash functions and VRFs as subcomponents. The details of hash functions and VRFs, which utilise public-key cryptographic primitives found in the RSA or elliptic curve cryptosystems, are public knowledge. Finally, the blockchain contains each block proposer's lottery ticket and the required information for public verification.

In addition to the operational transparency of PoPVR, each recipient (address holder) has an equal chance to participate in the consensus process. The only strategy for an asset holder to increase the probability of a winning lottery ticket is to redistribute its assets to multiple addresses, essentially paying itself multiple times in, for example, a cryptocurrency blockchain system. This type of behaviour will attract transaction costs to its detriment.

V. CONCLUSION

This paper proposed a method to use permissionless blockchain systems as pseudo-random number generators to produce *publicly verifiable* pseudo-random numbers that are *confidential*, *tamper-resistant*, *unpredictable* and *collision-resistant*. It also showed how to use these pseudo-random numbers to construct a consensus mechanism (PoPVR) that can operate without the disadvantages of algorithms that rely on economic tokens, for example, PoS and DPoS. PoPVR does not require large amounts of computation as with PoW and scales well, with computational complexity linear in the number of transactions per block as needed for verifying lottery tickets. Appendix D summarises the characteristics of proof-based consensus algorithms discussed in II-E and adds the properties of PoPVR.

Under reasonable operational assumptions (IV-A), security risks such as denial-of-service, nothing-at-stake and double-spending attacks, which are of concern in other consensus algorithms that also enable low-cost block creation, can be mitigated by PoPVR

The main implementation details not yet investigated revolve around participation limits, for example, requiring that a potential block proposer must source its reference transaction from a block within a specific age range. These parameters are related to the problem of block finality. For example, to ensure that a block proposer does not find a valid block and holds on to it indefinitely (presumably with the intent to create a fork in the blockchain), Bitcoin requires that miners ignore blocks older than two hours when first received [101]. It is unknown precisely how many multiples of the blockchain network's total latency intervals are required before a block can be considered final. Both these problems require a different research approach, such as the setup and live testing of PoPVR, which require additional study.

PoPVR is a generic approach to decentralised consensus, adaptable to the needs of a specific blockchain implementation. It allows the preservation of all aspects of decentralisation in permissionless blockchain systems referred to in the introduction.

APPENDIX A*

See Table 4.

APPENDIX B

See Table 5.

APPENDIX C

See Table 6.

APPENDIX D

See Table 7.

TABLE 7. Summary of proof-based consensus algorithms.

Consensus algorithm	Suitable for a decentralised environment	Worst case complexity	Note
PoW	Yes	Hash calculation has a non-deterministic computational complexity.	PoW is a proven algorithm with a substantial track record. Concerns exist over energy use and the centralisation of the computational power of the mining network in mining pools.
PoS	Yes	Finding the coin owner in the Finding the Satoshi randomisation algorithm is linear in the number of blocks in the blockchain.	The staking process is automated and does not require the active involvement of participants to vote for block producers. Concerns exist over the influence of large coin holders on the consensus process.
DPoS	Yes, but active community participation is required.	The literature is vague, but the witness signing process may have low complexity, such as linear in the number of transactions per block. More complex signing protocols may result in quadratic complexity in the number of witnesses.	More democratic than PoS as small coin holders can participate in selecting block producers. Requires participants to take an active interest in the voting process, and concerns exist that only a tiny proportion of the account holders will participate.
nlPoW	Yes	Hash calculation has a non-deterministic computational complexity.	While the expected energy consumption is lower than PoW, nlPoW is still in the non-deterministic complexity class.
PoW (Cuckoo Hash Function)	Yes	Hash calculation has a non-deterministic computational complexity.	This algorithm exchanges a lower computational load for increased RAM requirement. It may have the same runaway resource use as PoW.
PoW (Prime Chains)	Yes	Finding prime chains are non-deterministic complexity.	It still requires large amounts of computation, serving as an incentive for block producers to invest in ever-growing resources.
PoET	No	linear in the number of transactions per block.	It relies on a central authority to provide a synchronised time-stamp to all block producers.
PoL	No	linear in the number of transactions per block.	It relies on a central authority to provide a synchronised time-stamp to all block producers.
PoH	Yes	Uncertain as the consensus process requires human interaction.	Exchanges computer resources for human resources. Uncertain of practical.

TABLE 7. *(Continued.)* **Summary of proof-based consensus algorithms.**

PoB	No	Hash calculation has non-deterministic computational complexity, but the number of computations is limited to the amount of currency each participant is willing to sacrifice.	Skews the probability of successfully producing a block towards a participant that can afford to sacrifice more coins.
PoSpace	Yes	Non-deterministic space complexity.	This algorithm exchanges a lower computational load for increased storage requirements. It may have the same runaway resource use as PoW.
PoI	Yes	Calculating coin age is linear in the number of transactions per block.	Combines the number of assets with asset holding time and transaction activity to place a value on the importance of the participant. Concerns are that participants may adopt strategies to skew their ability to create blocks in their favour.
PoR	No	Hash calculation has a non-deterministic computational complexity.	Rely on a selected group of authorities to oversee the system's security.
PoA	No	Depending on implementation may be linear in the number of transactions per block or quadratic in the number of authorities.	Rely on a selected group of authorities to oversee the system's security.
Dfinity	Yes	Quadratic computational complexity in participating witnesses to establish distributed key generation.	Dfinity is a semi-permissionless blockchain system but cannot function in a completely decentralised environment.
PoPVR	Yes	Lottery verification has computational complexity linear in the number of transactions per block.	Still a theoretical consensus model and requires more research to establish practical applicability.

REFERENCES

[1] S. Squarepants, "Bitcoin: A peer-to-peer electronic cash system," Bitcoin.og, Tech. Rep. 1, 2008.

[2] C. Holotescu, "Understanding blockchain technology and how to get involved," in *Proc. Int. Sci. Conf. eLearn. Softw. Educ.*, 2018, pp. 1–22.

[3] I. Konstantinidis, G. Siaminos, C. Timplalexis, P. Zervas, V. Peristeras, and S. Decker, "Blockchain for business applications: A systematic literature review," in *Proc. Bus. Inf. Syst., 21st Int. Conf.*, vol. 320, 2018, pp. 384–399.

[4] O. Labazova, "Towards a framework for evaluation of blockchain implementations," in *Proc. Int. Conf. Inf. Syst.*, 2019, pp. 1–17.

[5] N. Nawari and S. Ravindran, "Blockchain technology and BIM process: Review and potential applications," *J. Inf. Technol. Constr.*, vol. 24, pp. 209–238, May 2019.

[6] C. Cachin and M. Vukolic, "Blockchains consensus protocols in the wild," in *Proc. 31st Int. Symp. Distrib. Comput.*, 2017, pp. 1–24.

[7] R. Rueda, E. Šaljic, and D. Tomić, "The institutional landscape of blockchain governance. A taxonomy for incorporation at the nation state," *TEM J.*, vol. 9, no. 1, pp. 181–187, 2020.

[8] R. Bezuidenhout, W. Nel, and J. Maritz, "Defining decentralisation in permissionless blockchain systems," *Afr. J. Inf. Commun. (AJIC)*, no. 29, pp. 1–24, Jul. 2022.

[9] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2017, pp. 557–564.

[10] P. Tasca and C. J. Tessone, "A taxonomy of blockchain technologies: Principles of identification and classification," *Ledger*, vol. 4, pp. 1–39, Feb. 2019.

[11] F. Glaser, "Pervasive decentralisation of digital infrastructures: A framework for blockchain enabled system and use case analysis," in *Proc. 50th Hawaii Int. Conf. Syst. Sci.*, 2017, pp. 1–10.

[12] S. Adhami, G. Giudici, and S. Martinazzi, "Why do businesses go crypto? An empirical analysis of initial coin offerings," *J. Econ. Bus.*, vol. 100, pp. 64–75, Nov. 2018.

[13] W. Dai. (1988). *B-Money, Satoshi Nakamoto Institute*. Accessed: Jun. 10, 2022. [Online]. Available: https://nakamotoinstitute.org/b-money/

[14] T. Aste, P. Tasca, and T. D. Matteo, "Blockchain technologies: The foreseeable impact on society and industry," *Computer*, vol. 50, no. 9, pp. 18–28, Jan. 2017.

[15] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Business & infomation system engineering," *Bus. Inf. Syst. Eng.*, vol. 59, no. 3, pp. 183–187, 2017.
[16] N. Chaudhry and M. M. Yousaf, "Consensus algorithms in blockchain: Comparative analysis, challenges and opportunities," in *Proc. 12th Int. Conf. Open Source Syst. Technol. (ICOSST)*, Dec. 2018, pp. 54–63.
[17] X. Xu, "A taxonomy of blockchain-based systems for architecture design," in *Proc. IEEE 6th Int. Congr. Softw. Archit.*, Apr. 2017, pp. 243–252.
[18] B. O. M. Mular, "Blockchain technology in the enterprise environment," M.S. thesis, Fac. Inform., Masaryk Univ., Brno, Czech Republic, 2018.
[19] K. John, M. O'Hara, and F. Saleh, "Bitcoin and beyond," *Annu. Rev. Financ. Econ.*, vol. 14, pp. 95–115, Nov. 2021.
[20] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. Nyang, and A. Mohaisen, "Exploring the attack surface of blockchain: A systematic overview," 2019, *arXiv:1904.03487*.
[21] F. Glaser and L. Bezzenberger, "Beyond cryptocurrencies—A taxonomy of decentralized consensus systems," in *Proc. ECIS*, 2015, pp. 1–19.
[22] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the Internet of Things: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1676–1717, 2nd Quart., 2019.
[23] T. Ali Syed, A. Alzahrani, S. Jan, M. S. Siddiqui, A. Nadeem, and T. Alghamdi, "A comparative analysis of blockchain architecture and its applications: Problems and recommendations," *IEEE Access*, vol. 7, pp. 176838–176869, 2019.
[24] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen, "The blockchain as a software connector," in *Proc. 13th Work. IEEE/IFIP Conf. Softw. Archit. (WICSA)*, Apr. 2016, pp. 182–191.
[25] A. Deshpande, K. Stewart, L. Lepetit, and S. Gunashekar, "Distributed ledger technologies/blockchain: Challenges, opportunities and the prospects for standards," *Overview Rep. British Standards Inst.*, vol. 40, p. 40, May 2017.
[26] S. Kaur, S. Chaturvedi, A. Sharma, and J. Kar, "A research survey on applications of consensus protocols in blockchain," *Secur. Commun. Netw.*, vol. 2021, pp. 1–22, Jan. 2021.
[27] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for Internet of Things: A survey," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8076–8094, Oct. 2019.
[28] I. E. Inghirami, "Accounting information systems in the time of blockchain," in *Proc. ITAIS*, 2018, pp. 1–15.
[29] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, "Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities," *IEEE Access*, vol. 7, pp. 85727–85745, 2019.
[30] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies*. Princeton, NJ, USA: Princeton Univ. Press, 2016.
[31] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2084–2123, 3rd Quart., 2016.
[32] C. Troncoso, M. Isaakidis, G. Danezis, and H. Halpin, "Systematizing decentralization and privacy: Lessons from 15 years of research and deployments," *Proc. Priv. Enhancing Technol.*, vol. 2017, no. 4, pp. 307–329, 2017.
[33] V. Buterin, "A next generation smart contract & decentralized application platform," Ethereum.org, Tech. Rep. 1, 2013.
[34] Cardanofoundation.org. (2020). *Foundation Team*. Accessed: Jun. 16, 2020. [Online]. Available: https://cardanofoundation.org/en/team/
[35] Bitcoin.org. (2020). *Bitcoin Community*. Accessed: Jun. 16, 2020. [Online]. Available: https://bitcoin.org/en/community
[36] Ethereum.org. (2022). *Ethereum Community*. Accessed: Jun. 22, 2022. [Online]. Available: https://ethereum.org/community/
[37] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," *J. Cryptol.*, vol. 3, no. 2, pp. 99–111, Jan. 1991.
[38] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Rayton, FL, USA: CRC Press, 1996.
[39] C. Paar and J. Petzl, *Understanding Cryptography*. Heidelberg, Germany: Springer, 2010.
[40] K. Rabah, "Elliptic curve ElGamal encryption and signature schemes," *Inf. Technol. J.*, vol. 4, no. 3, pp. 299–306, Jun. 2005.
[41] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," in *Proc. 17th Int. Conf. Theory Appl. Cryptograph. Techn.*, 2005, pp. 925–963.
[42] D. Stinson and R. Strobl, "Provably secure distributed schnorr signatures and a (T, N) threshold scheme for implicit certificates," in *Proc. 6th Australas. Conf. Inf. Secur. Privacy*, 2001, pp. 417–434.
[43] R. Dahlberg, T. Pulls, and R. Peeters, "Efficient sparse Merkle treescaching strategies and secure (non-)membership proofs," in *Proc. IACR*, 2016, pp. 199–215.
[44] H. Massias, X. S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," in *Proc. 20th Symp. Inf. Theory Benelux*, 2002, pp. 1–8.
[45] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," in *Proc. 1st Int. Conf. Peer-Peer Comput.*, 2001, pp. 101–102.
[46] D. Schoder, K. Fischbach, and C. Schmitt, "Core concepts in peer-to-peer networking," in *Peer-to-Peer Computing: The Evolution of a Disruptive Technology*, R. Subramanian and B. D. Goodman, Eds. Hershey, PA, USA: Idea Group Publishing, 2005, pp. 1–27.
[47] F. J. Hinzen, K. John, and F. Saleh, "Bitcoin's limited adoption problem," *J. Financ. Econ.*, vol. 144, pp. 347–369, Jan. 2022.
[48] G.-T. Nguyen and K. Kim, "A survey about consensus algorithms used in blockchain," *J. Inf. Process. Syst.*, vol. 14, no. 1, pp. 101–128, 2018.
[49] I. Bentov, A. Gabizon, and A. Mizrahi, "Cryptocurrencies without proof of work," in *Proc. Financial Cryptogr. Data Secur. Conf.*, 2016, pp. 142–157.
[50] C. Li and B. Palanisamy, "Comparison of decentralization in DPoS and PoW blockchains," in *Proc. ICBC*, 2020, pp. 18–32.
[51] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Proc. 18th Int. Conf. Financial Cryptogr. Data Secur.*, 2014, pp. 1–17.
[52] F. Saleh, "Blockchain without waste: Proof-of-stake," *Rev. Financial Stud.*, vol. 34, no. 3, pp. 1156–1190, Feb. 2021.
[53] A. Li, X. Wei, and Z. He, "Robust proof of stake: A new consensus protocol for sustainable blockchain systems," *Sustainability*, vol. 12, no. 7, p. 15, 2020.
[54] A. Hasib. (2018). *101 Blockchains*. Accessed: Nov. 12, 2019. [Online]. Available: https://101blockchains.com/consensus-algorithms-blockchain/#6
[55] F. Hinzen, F. Irresberger, K. John, and F. Saleh, "The public blockchain ecosystem: An empirical analysis," *SSRN Electron. J.*, pp. 1–43, Jan. 2019.
[56] BitShares Blockchain Foundation. *Delegated Proof-of-Stake Consensus*. Accessed: Aug. 11, 2019. [Online]. Available: https://bitshares.org/technology/delegated-proof-of-stake-consensus/
[57] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *Proc. 41st Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, 2018, pp. 1545–1550.
[58] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (PoET)," in *Proc. Stabilization, Safety, Secur. Distrib. Syst., 19th Int. Symp.*, 2017, pp. 282–297.
[59] R. Bezuidenhout, W. Nel, and A. Burger, "Nonlinear proof-of-work: Improving the energy efficiency of bitcoin mining," *J. Construct. Project Manag. Innov.*, vol. 10, no. 1, pp. 20–32, Sep. 2020.
[60] J. Tromp. (2014). *Cuckoo Cycle: A Memory Bound Graph-Theoretic Proof-of-Work*. Accessed: Nov. 14, 2019. [Online]. Available: https://eprint.iacr.org/2014/059.pdf
[61] S. King, "Primecoin: Cryptocurrency with prime number proof-of-work," Primecoin Project, Tech. Rep. 1, 2013.
[62] A. Meneghetti, M. Sala, and D. Taufer, "A survey on PoW-based consensus," *Ann. Emerg. Technol. Comput.*, vol. 4, no. 1, pp. 8–18, Jan. 2020.
[63] M. Milutinovic, W. He, H. Wu, and M. Kanwal, "Proof of luck: An efficient blockchain consensus protocol," in *Proc. IACR*, 2016, pp. 1–6.
[64] J. Blocki and H.-S. Zhou, "Designing proof of human-work puzzles for cryptocurrency and beyond," in *Theory of Cryptography*. Berlin, Germany: Springer, 2016, pp. 517–546.
[65] *Slimcoin: A Peer-to-Peer Crypto-Currency With Proof-of-Burn*, Slimcoin, 2014.
[66] T. Jenks. (2018). *Pros and Cons of Different Blockchain Consensus Protocols*. Accessed: Nov. 12, 2019. [Online]. Available: https://www.verypossible.com/blog/pros-and-cons-of-different-blockchain-consensus-protocols
[67] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, "Proofs of space," in *Proc. CRYPTO, Int. Cryptol. Conf.*, 2015, pp. 585–605.
[68] NEM. (2018). *Proof-of-Importance*. Accessed: Nov. 12, 2019. [Online]. Available: https://nemplatform.com/wp-content/uploads/2020/05/NEM_techRef.pdf

[69] Coinspace.com. (2019). *Electroneum's Revolutionary Proof of Responsibility Blockchain is Now Live*. [Online]. Available: https://coinspace.com/news/altcoin-news/electroneums-revolutionary-proof-responsibility-blockchain-now-live
[70] T. Hanke, M. Movahedi, and D. Williams, "DFINITY technology overview series, consensus system," 2018, *arXiv:1805.04548*.
[71] B. Libert, M. Joye, and M. Yung, "Born and raised distributively: Fully distributed non-interactive adaptively-secure threshold signatures with short shares," in *Proc. ACM Symp. Princ. Distrib. Comput.*, Jul. 2014, pp. 303–312.
[72] V. Gramoli, "From blockchain consensus back to Byzantine consensus," *Futur. Gener. Comput. Syst.*, vol. 107, pp. 760–769, Jun. 2020.
[73] L. Lamport, "The part-time parliament," *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, May 1998.
[74] I. Moraru, D. G. Andersen, and M. Kaminsky, "There is more consensus in egalitarian parliaments," in *Proc. 24th ACM Symp. Operating Syst. Princ.*, Nov. 2013, pp. 358–372.
[75] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX*, 2014, pp. 305–320.
[76] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. 3rd Operating Syst. Design Implement.*, 1999, pp. 173–186.
[77] M. Lokhava, G. Losa, D. Mazières, G. Hoare, N. Barry, E. Gafni, J. Jove, R. Malinowsky, and J. McCaleb, "Fast and secure global payments with stellar," in *Proc. 27th ACM Symp. Operating Syst. Princ.*, Oct. 2019, p. 17.
[78] T. Nguyen-Van, T. Nguyen-Anh, T.-D. Le, M.-P. Nguyen-Ho, T. Nguyen-Van, N.-Q. Le, and K. Nguyen-An, "Scalable distributed random number generation based on homomorphic encryption," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 572–579.
[79] M. Andrychowicz and S. Dziembowski, "Distributed cryptography based on the proofs of work," *Cryptol. ePrint Arch.*, vol. 2014, p. 796, Dec. 2014.
[80] I. Cascudo and B. David, "SCRAPE: Scalable randomness attested by public entities," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2017, pp. 537–556.
[81] B. Schoenmakers, "A simple publicly verifiable secret sharing scheme and its application to electronic voting," in *Proc. Annu. Int. Cryptol. Conf.*, 1999, pp. 148–164.
[82] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford, "Scalable bias-resistant distributed randomness," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 444–460.
[83] P. Schindler, A. Judmayer, N. Stifter, and E. Weippl, "HydRand: Practical continuous distributed randomness," *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 319, Aug. 2018.
[84] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *Advances in Cryptology—ASIACRYPT*, Berlin, Germany: Springer, 2001, pp. 514–532.
[85] M. Alturki and G. Rosu, "Statistical model checking of RANDAO's resilience to pre-computed reveal strategies," in *Proc. Int. Symp. Formal Methods*, G. Goos and J. Hartmanis, Eds. Berlin, Germany: Springer, 2020, pp. 337–349.
[86] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," in *Proc. 8th Int. Conf. Theory Pract. Public Key Cryptogr.*, vol. 3386, 1970, pp. 416–431.
[87] D. Boneh, S. Eskandarian, L. Hanzlik, and N. Greco, "Single secret leader election," in *Proc. 2nd ACM Conf. Adv. Financial Technol.*, Oct. 2020, pp. 12–24.
[88] D. Bonehy, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. Rasmussen, and A. Sahai, "Threshold cryptosystems from threshold fully homomorphic encryption," *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 10991. Berlin, Germany: Springer, 2018.
[89] D. Boneh, J. Bonneau, B. Bunz, and B. Fisch, "Verifiable delay functions," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2018, pp. 757–788.
[90] V. Fuchs-Attias, L. Vigneri, and V. Dimitrov, "Implementation study of two verifiable delay functions," in *Proc. Tokenomics*, Toulouse, France, E. Anceaume, C. Bisière, M. Bouvard, Q. Bramas, and C. Casamatta, Eds., Oct. 2020, pp. 6:1–6:15.
[91] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Proc. 40th Annu. Symp. Found. Comput. Sci.*, 1999, pp. 120–130.
[92] R. Bezuidenhout, W. Nel, and J. M. Maritz, "Embedding tamper-resistant, publicly verifiable random number seeds in permissionless blockchain systems," *IEEE Access*, vol. 10, pp. 39912–39925, 2022.
[93] R. Bezuidenhout, W. Nel, and J. Maritz, "Transient random number seeds in permissionless blockchain systems," in *The Transdisciplinary Reach of Design Science Research*. Cham, Switzerland: Springer, 2022, pp. 85–96.
[94] W. Schindler, "Random number generators for cryptographic applications," in *Cryptographic Engineering*, C. K. Koc, Ed. Boston, MA, USA: Springer, 2009, pp. 5–23.
[95] S. Goldberg, L. Reyzin, D. Papadopoulos, and J. Vcelak, "Verifiable random functions," Internet Eng. Task Force, Washington, DC, USA, Tech. Rep. draft-irtf-cfrg-vrf-15, 2022.
[96] P. B. Stark and K. Ottoboni, "Random sampling: Practice makes imperfect," 2018, *arXiv:1810.10985*.
[97] A. Rukhin, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," National Institute of Standards and Technology, Gaithersburg, MD, USA, Tech. Rep. Special Publication 800-22, Revision 1a, 2010.
[98] Accord. (2017). *Accord.NET Framework*. Accessed: Mar. 18, 2022. [Online]. Available: http://accord-framework.net/docs/html/T_Accord_Statistics_Testing_KolmogorovSmirnovTest.htm
[99] R. Larson, R. Hostetler, and B. H. Edwards, *Precalculus Functions and Graphs A Graphing Approach*, 5th ed. Boston, MA, USA: Houghton Mifflin Company, 2008.
[100] (2022). *Karlsruhe Institute of Technology (Decentralized Systems and Network Services Research Group)*. Accessed: Oct. 29, 2022. [Online]. Available: https://www.dsn.kastel.kit.edu/bitcoin/
[101] A. M. Antonopoulos, *Mastering Bitcoin*. Sebastopol, CA, USA: O'Reilly Media, 2014.
[102] T. Thorn, A. Baumann, M. Fennestad, and P. Sestoft, "A distributed, value-oriented XML store," M.S. thesis, Dept. Comput. Sci., IT Univ. Copenhagen, Copenhagen, Denmark, 2002.

RIAAN BEZUIDENHOUT was born in Qonce, South Africa, in 1970. He received the B.Com. degree, the B.Com. degree (Hons.) in statistics, the B.Com. degree (Hons.) in business management, the M.B.A. degree, the B.Sc. degree (Hons.) in computer science and informatics, and the M.Sc. degree in computer science and informatics from the University of the Free State, Bloemfontein, South Africa, in 1993, 1996, 1999, 2003, 2018, and 2020, respectively, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Informatics.

He worked in management consulting until 2012 before returning to university in 2013 to study computer science. He works as an Assistant Researcher with the Department of Computer Science and Informatics, University of the Free State. His research interest includes blockchain consensus algorithms.

WYNAND NEL was born in Humansdorp, South Africa, in 1980. He received the B.Com. degree in information technology, in 2001, the B.Com. degree (Hons.) in computer science and informatics, in 2002, the M.Com. degree in computer science and informatics, in 2007, and the Ph.D. degree from the University of the Free State, Bloemfontein, South Africa, in 2019.

From 2002 to 2005, he was a Junior Lecturer in computer science with Technicon Free State, Bloemfontein. From 2005 to 2006, he was a Lecturer in computer science at the Central University of Technology, Free State, Bloemfontein. Since 2007, he has been a Lecturer with the Computer Science and Informatics Department, University of the Free State. He has also been involved in the private sector with software, web, and app development, since 2002. His research interests include blockchain technology, cyber security, and human–computer interaction.

JACQUES M. MARITZ received the master's degree and the Ph.D. degree in astrophysics from the University of the Free State, South Africa, in 2017.

He has been a Lecturer in engineering sciences with the University of the Free State, since 2017. His research interests include physics, astrophysics, energy modeling, energy analytics, energy AI, and power systems.