IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 23, 2022, accepted March 29, 2022, date of publication April 7, 2022, date of current version April 18, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3165616

Embedding Tamper-Resistant, Publicly Verifiable Random Number Seeds in Permissionless Blockchain Systems

RIAAN BEZUIDENHOUT[1], WYNAND NEL[1], AND JACQUES M. MARITZ[2]
[1]Department of Computer Science and Informatics, University of the Free State, Bloemfontein 9301, South Africa
[2]Department of Engineering Sciences, University of the Free State, Bloemfontein 9301, South Africa

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

ABSTRACT Many blockchain processes require pseudo-random numbers. This is especially true of blockchain consensus mechanisms that aim to fairly distribute the opportunity to propose new blocks between the participants in the system. The starting point for these processes is a source of randomness that participants cannot manipulate. This paper proposes two methods for embedding random number seeds in a blockchain data structure to serve as inputs to pseudo-random number generators. Because the output of a pseudo-random number generator depends deterministically on its seed, the properties of the seed are critical to the quality of the eventual pseudo-random number produced. Our protocol, B-Rand, embeds random number seeds that are *confidential*, *tamper-resistant*, *unpredictable*, *collision-resistant*, and *publicly verifiable* as part of every transaction. These seeds may then be used by transaction owners to participate in processes in the blockchain system that require pseudo-random numbers. Both the Single Secret and Double Secret B-Rand protocols are highly scalable with low space and computational cost, and the worst case is linear in the number of transactions per block.

INDEX TERMS B-Rand, blockchain, consensus algorithm, homomorphic encryption, pseudo-random number generation, random number seeds.

I. INTRODUCTION

When Satoshi Nakamoto introduced the concept of peer-to-peer transactions using a blockchain system in 2008 [1], it is reasonable to assume that he did not intend for its electricity consumption to reach current levels. Because Bitcoin miners keep details about their operations secret, it is not possible to make a direct attempt to determine the electricity consumption of the Bitcoin network [2]. De Vries [2] does, however, make an indirect estimate of the lower bound of the electricity use by the Bitcoin network of 2.55 Gigawatt. This was done by estimating the average number of hash calculations per second (from the block target parameter) multiplied by the electricity use of the most efficient mining hardware. Using this same method on 15 September 2021, it is noted that the current lower bound on electricity consumption is 12.9 Gigawatt or 113.2TWh per year. According to the U.S. Energy Information Administration, this is on par with the electricity consumption of the Netherlands [3]. While these estimates are the lower bounds and do not account for miners using less efficient hardware or related electricity use, notably the cooling requirements of mining operations, it goes a long way to highlight the problem. It is unknown how Satoshi envisaged the long-term future of his idea, but it seems reasonable to these authors that the intention was for stakeholders who engage with the Bitcoin blockchain maintenance process on the same basis as they would in creating transactions, by existing hardware and without additional investment in computational or electrical supply infrastructure.

Ma, Gans and Tourky [4] established that the Bitcoin energy consumption problem stemmed from the proof-of-work (PoW) consensus algorithm used by the Bitcoin network. In any blockchain system, the consensus algorithm is critical in establishing agreement over the final correct ver-

The associate editor coordinating the review of this manuscript and approving it for publication was Mehdi Sookhak.

sion of the blockchain. PoW is also the consensus algorithm used by Ethereum [5]. Ma, Gans and Tourky [4] postulated that the solution to Bitcoin's energy use lies in reducing the number of competing miners that attempt to solve each block of transactions as it is appended to the blockchain. The process of devising alternative blockchain consensus algorithms is ongoing, and new ideas are constantly coming to light [6].

Some alternatives, such as proof-of-stake (PoS) and proof-of-importance, exhibit the possibility for the consensus mechanism to be skewed in the direction of a minority of stakeholders in the blockchain network [7], [8]. Proof-of-elapsed time, proof-of-luck, and proof-of-responsibility, on the other hand, require intervention by a centralised authority, which opposes the primary ethos of a peer-to-peer transaction system [9]–[11]. Another strategy is to shift the use of energy to another resource, such as PoW with a cuckoo hash function (random access memory), proof-of-space (data storage capacity), and proof-of-burn (blockchain tokens) [12]–[14]. There have even been attempts to save resources by sacrificing automation in favour of increased stakeholder involvement. Examples of these are delegated proof-of-stake (DPoS) and proof-of-human work [15], [16]. Despite all these attempts, and the list above is certainly not exhaustive, PoW remains the most important blockchain consensus algorithm. This is given by the fact that more than 82% of the top ten cryptocurrency market capitalisations (US$1.4 trillion) belong to cryptocurrencies that utilise the PoW consensus mechanism [17].

Currently, the most promising alternative to PoW is the family of PoS and DPoS consensus protocols. They select block proposers based on their stake (value of assets/number of coins). This may lead to a minority of participants exercising control over the system. Since each participant has a weighted (non-equal) probability of participation, some protocols allow smaller stakeholders to delegate their 'voting right' to a pool. This allows broader participation but also opens the door for collusion by groups of stakeholders to the exclusion of others [7]. PoS type systems also require that the value of digital assets is objectively comparable, i.e., comparing the number of coins.

The above synopsis begs the question of whether it is possible to design a more practical, less wasteful, permissionless blockchain system. While this paper does not provide a complete solution, the question provides a fundamental perspective on one of the elements that will surely be part of the solution. A critical starting point is to take a step back and consider the primary purpose of blockchain consensus algorithms. Glaser [18] summarised it well when he said that the purpose of blockchain consensus algorithms is to select the party that adds a block to the blockchain at random. En-route to putting Glaser's advice into practice (selecting a party at random), the problem of generating pseudo-random numbers must be solved.

Pseudo-random numbers, produced by pseudo-random number generators, also known as deterministic random number generators, require seed values as input. Each seed produces the same pseudo-random number when used as an input in the same pseudo-random number generator [19]. It stands to reason that the first step in selecting the party that adds a block to the blockchain at random requires a seed. Specifically, for use in a blockchain consensus algorithm, the seed must be tamper-resistant and publicly verifiable. Tamper-resistant, in this case, refers to tampering, first by any would-be proposer of a new block and second by any stakeholder who may wish to assist or hinder that proposer. Furthermore, because the block addition process in any permissionless blockchain is subject to verification by all stakeholders, the seed must be publicly verifiable.

The use of tamper-resistant, publicly verifiable random number seeds may also be used elsewhere on the blockchain. This may include input values for smart contracts (determining the ordering of future events), operation of distributed applications (gaming, for example), or for the construction of entirely new blockchain applications (such as publicly verifiable distributed random number generators).

In this paper, we propose a *confidential*, *tamper-resistant*, *unpredictable*, *collision-resistant*, *publicly verifiable*, and *scalable* protocol (B-Rand) to assign random number seeds to transaction owners for use in blockchain applications where the use of pseudo-random numbers may be required. It is a mechanism that allows participants to produce provable pseudo-random numbers for participation in, for example, block proposer selection that is randomly distributed among all participants and where the value of digital assets may not be objectively comparable, for example, supply chain or intellectual property records.

In addition to the characteristics above, B-Rand is also self-contained within the blockchain system and does not require the use of any outside oracle or services. The protocol has two variants, the first, Single Secret B-Rand, which requires only two parties to implement, namely the transaction owner where the seed is embedded and the verifier. The second variant, Double Secret B-Rand, requires three parties: the transaction owner of the transaction where the seed is embedded, the proposer (in some blockchain systems referred to as the miner) of the block where the transaction is recorded, and the verifier. In both cases, no party can employ any meaningful strategy to manipulate the seed. B-Rand adds exceedingly low computational complexity to the blockchain system and space complexity to the data structure.

This paper is organised into six parts. The five parts to follow start with a general background discussion of blockchain systems, focusing on how transactions and transaction blocks are created, before briefly reminding the reader of key aspects of public-key cryptography, homomorphic encryption and pseudo-random number generation (Section II). Distributed random number generators are not a new idea, and the most important work to date on the subject is given in Section III. A description of the two B-Rand protocols is laid out in Section IV, and in the analysis (Section V), the claims of the

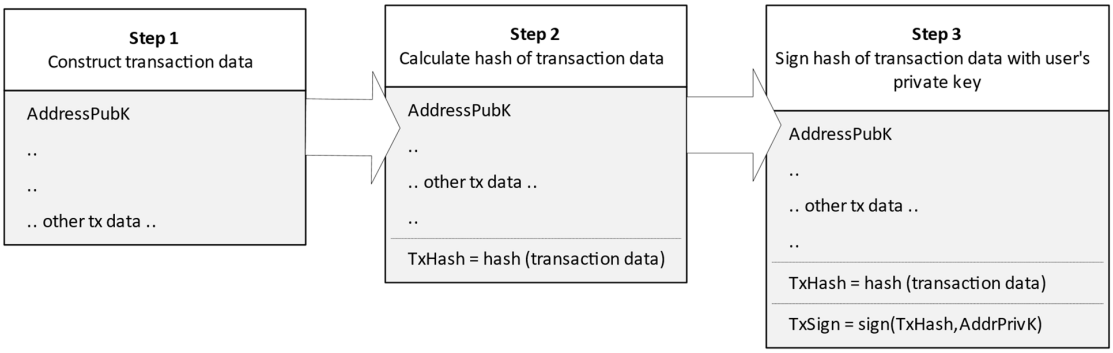

FIGURE 1. A generalised blockchain transaction construction process.

paper are revisited, and each one is addressed individually regarding the details of the protocol.

II. BACKGROUND

Permissionless blockchain systems are distributed systems that employ a combination of technologies such as distributed ledgers, cryptography, and consensus algorithms so that untrusted parties can agree on the state of decentralised transaction data [20]–[22]. One of the components of a blockchain system is a distributed ledger or blockchain that conforms to a cryptographically linked data structure, serving as a sequential record of transactions. The characteristics of the distributed ledger are designed to enable parties to agree on a single correct version of the transaction record, without having to trust each other and without the intervention or oversight of a central authority [23], [24].

A blockchain system's purpose is to record transactions on a distributed ledger that are immutable and cannot be repudiated, secure, transparent, and accessible [20], [24]. Permissionless blockchain systems do not restrict the participation of any stakeholder, and no central authority exercises control over them. They function on a peer-to-peer basis, and a decentralised consensus mechanism is required for participants to reach an agreement on the ultimate correct state of the blockchain [6], [20], [21].

Permissionless blockchain systems require ways to designate one participant with the right to act on behalf of other participants [25]. A good example is the execution of a consensus algorithm, such as PoW [1] or PoS [6]. In these instances, the designated participant is selected by an algorithm based on a random process. Permissionless blockchain systems allow participants to employ any strategy allowed by the consensus algorithm and, as such, can act with self-interest. This is currently the case with, for instance, Bitcoin mining. These self-interest strategies may have unintended or undesirable consequences, such as inefficient resource use [26].

A. THE BLOCKCHAIN TRANSACTION CONSTRUCTION PROCESS

Any stakeholder who desires to interact with a blockchain system does so by initiating a transaction on the blockchain. The process starts by constructing a transaction for transmission to a blockchain network and can broadly be divided into three steps (Fig. 1). Note that this study treats the mechanics of permissionless blockchain systems in their most general sense, as the research is independent of the specific intricacies of any specific blockchain implementation.

Step 1 consists of constructing the transaction data. For simplicity, it is useful to imagine that each transaction contains a blockchain address of the user initiating the transaction (the transaction owner, for brevity, referred to as the owner). In practice, it may be that a transaction contains multiple addresses belonging to the owner, but this does not matter for illustration purposes. The address is the owner's public key in a public-key encryption scheme. Step 2 involves calculating the hash of the transaction data. This hash, denoted as TxHash, is appended to the transaction data and serves as a unique identifier for the transaction. Step 3 requires that the user sign TxHash using their private key. Again, the signature is appended to the transaction data. The reason for this final step is to allow any party, for example, a proposer, to verify that first, the transaction was signed by the rightful owner of the address (AddrPubK) and second, that the transaction data has not been tampered with while in transit. Transactions are propagated across the blockchain network, where they are eventually recorded on the blockchain.

B. THE TRANSACTION BLOCK

Blockchain transactions are recorded on the blockchain in a Merkle tree. In a Merkle tree, the leaf nodes contain the transaction data, with each transaction identified by its hash (TxHash), as illustrated in Fig. 2.

The leaf nodes are grouped into pairs (siblings), and the TxHash of each sibling pair is hashed to produce a parent hash. This process is repeated recursively until a single root hash, the Merkle root, is produced [1], [27], [28].

C. THE BLOCKCHAIN BLOCK

Each block on a blockchain can be divided into two parts or sub-blocks: the header block and the transaction block [29].

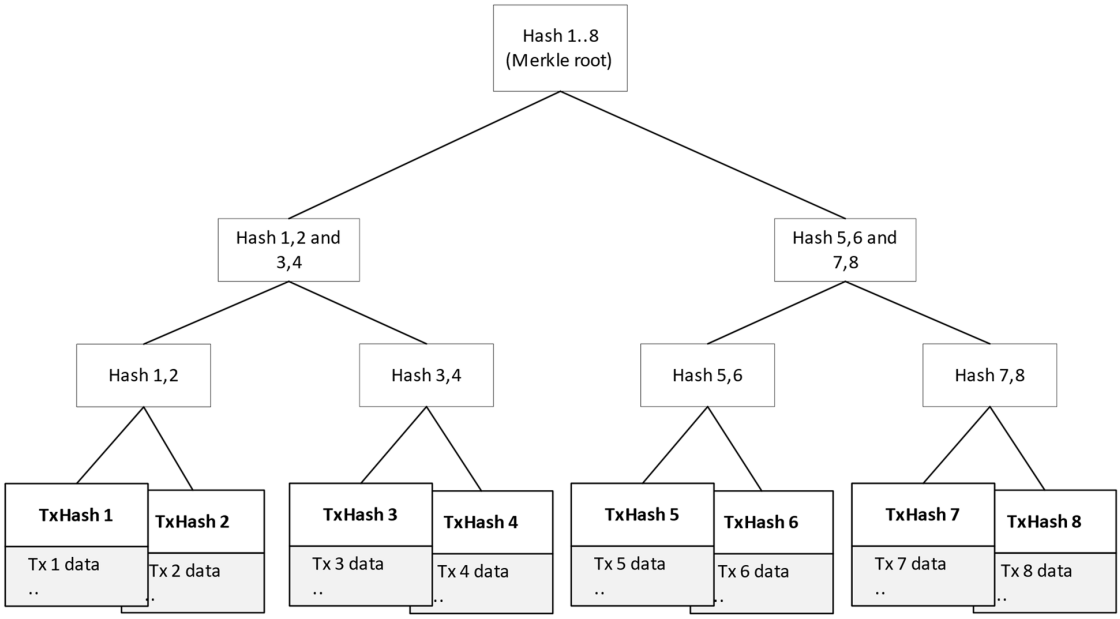

FIGURE 2. Transaction Merkle tree.

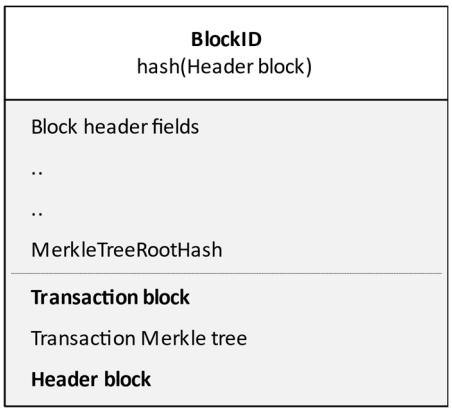

FIGURE 3. A simple blockchain block.

For the explanations to follow, a very simple block structure is shown in Fig. 3.

Each block in the blockchain carries a block identifier (BlockID), which is calculated as the hash of the data in the block header. Merkle root (MerkleTreeRootHash) was included in this set of data. This construct effectively encodes all the transaction data into the block hash, providing tamper resistance across all the data in the block [30].

D. HOMOMORPHIC ENCRYPTION

RSA, Elgamal and Elliptic Curve Cryptography are public-key cryptography schemes [31]. Public-key cryptography provides five essential services for data and communication security. These services are key generation, encryption, decryption, message signature generation, and signature verification [32]. Table 1 summarises the five public-key cryptography services.

TABLE 1. Essential services of public-key cryptography.

Service	Description
Key Generation	Creates a private and public key (Key_{priv}, Key_{pub}) for use in the other four functions and provides protocols for exchanging keys between parties
Encryption	$Message_{encrypted}$ = Encrypt ($Message_{plain\text{-}text}$ with Key_{pub})
Decryption	$Message_{plain\text{-}text}$ = Decrypt ($Message_{encrypted}$ with Key_{priv})
Message signature	Signature = Sign ($Message_{plain\text{-}text}$ with Key_{priv})
Signature verification	Verification = Verify ($Message_{plain\text{-}text}$, Signature with Key_{pub})

Encryption takes place with the public key; for instance, Bob encrypts a message to Alice with Alice's public key, and Alice decrypts it with her private key. This allows Alice and Bob to share information known only to them. Message signing takes place with the private key; for instance, Alice signs a message with her private key and sends both the message and the signature to Bob. Bob can verify two things using Alice's public key. First, the message signature originated from Alice (and not someone pretending to be Alice), and the message was not changed or tampered with in transit [31], [33].

The RSA and Elgamal cryptographic schemes are partially homomorphic encryption schemes. Partially homomorphic encryption schemes allow a single type of operation (addition or multiplication) over encrypted data. For the B-Rand protocol, homomorphic encryption means that messages encrypted with the same public key can be summed to give the encrypted sum of the unencrypted messages. When decrypted, this sum yields the sum of unencrypted

messages (1) [33]:

$$\begin{aligned} If\ A &= encrypt\ (a) \\ B &= encrypt\ (b)\ , \\ then\ C &= A + B \\ &= encrypt\ (a + b) \\ and & \\ decrypt\ (C) &= a + b \end{aligned} \tag{1}$$

This is an important property that plays an instrumental role in the Double Secret B-Rand protocol.

E. PSEUDO-RANDOM NUMBER GENERATION

The reader is reminded that this paper proposes a method for embedding a seed for pseudo-random number generation and does not suggest how the eventual pseudo-random number is ultimately calculated. This detail is left to be determined by the details of whatever blockchain implementation is required. However, pseudo-random number generators are deterministic and depend on the value of the seed to produce a pseudo-random number algorithmically. It is therefore necessary that the method for producing seed values must exhibit a high degree of entropy and that they are produced using a true random number generator [19]. Schindler (2009) suggested the construction of a non-physical true random number generator from non-deterministic system sources such as system time, cursor position, thread numbers, and the hash over a specified RAM area, to produce a raw bit string that may be post-processed via a hash function to a fixed length. B-Rand uses this idea as the basis for producing embedded seed values.

III. RELATED WORK

The problem of producing publicly verifiable random numbers, also referred to as beacons or random beacons, has been studied in a range of applications [33]. Andrychowicz and Dziembowski [34] devised a protocol that uses the properties of cryptographic hash functions in PoW blockchain systems to produce an unpredictable public beacon in a peer-to-peer environment. This beacon can be used in multi-party computation to produce genesis blocks for new blockchain systems or enhance security in cryptocurrencies. It differs from the B-Rand protocol in that it provides the same beacon to all parties. They proposed a post-process method for differentiating random numbers between parties, but raised concerns that this method could be open to manipulation. PoW algorithms complete in non-deterministic polynomial time and scale particularly badly.

Scrape [35] is a protocol that uses a distributed ledger and publicly verifiable secret sharing [36] to enable a set of participants to generate a random beacon. Again, the beacon is the same for all participants during each round as opposed to B-Rand, which provides individualised seeds for individualised random numbers. In their work, [35] identify that Scrape does not scale well with cubic computational complexity.

RandHound and RandHerd are two protocols by [37]. RandHound provides individualised, publicly verifiable random numbers to requesting clients, using a commit/reveal scheme, publicly verifiable secret sharing [36], and threshold signatures [38]. RandHerd extends the RandHound protocol to produce a stream of publicly verifiable random numbers which are not individualised. Both RandHound and RandHerd have been reported by the authors to exhibit quadratic computational complexity. RandHound and RandHerd were not intended to produce random numbers for blockchain systems specifically, but they required multiple participants (a client or requester and multiple servers) to operate. B-Rand requires only two participants in a blockchain system, namely the transaction owner or requester, and the block proposer, to produce an individualised random number seed.

HydRand [39] is a protocol aimed at permissioned environments where the number of participants is fixed. It also employs publicly verifiable secret sharing and consecutive rounds of random leader selection to produce a continuous publicly verifiable random beacon. HydRand claims to guarantee randomness, even in the presence of unreliable participants. The protocol has a quadratic computational complexity [39]. HydRand differs from B-Rand in its requirement for a permissioned environment.

Dfinity [40] is a blockchain consensus protocol with a built-in pseudo-random number generator that forms the basis for selecting new block proposers. It uses distributed key generation and a threshold signature scheme [41] based on Boneh-Lynn-Shacham signatures [42] to produce a random beacon. Dfinity allows participants to freely join or leave the network but does require that for each epoch (a fixed number of consecutive block additions), all the participants are identified. This is achieved with a special opening block for each epoch containing transactions where participants register their intention to participate or leave. This is an interesting idea because it creates a semi-permissionless blockchain system that utilises the security advantages of permissioned blockchain systems while striving for the open participation of permissionless blockchain systems. The distributed key generation protocol at the beginning of each epoch requires quadratic computation [43], but the repeated signing process for generating random numbers during the epoch is linear [40]. B-Rand differs from Dfinity in that it is not a blockchain consensus algorithm per se, but it may be used as a source of randomness for one. Furthermore, B-Rand operates in a completely decentralised manner, without the need for participants to be identified before participating. Since B-Rand is designed to function within permissionless blockchain systems, the term decentralised means that it operates between disintermediated parties [44], [45] over a peer-to-peer network and relies solely on algorithmic trust [46], [47] to decide the validity of claims. For example, the value of a random number calculated by a participant for use in a blockchain process can be checked by any verifier from publically available information encoded on the blockchain and computation alone.

Randao [48] is a commit-reveal type random beacon that utilises the Ethereum blockchain to enable public verifiability. Participants share the hash values of their locally generated seeds on the blockchain, and once all seed hashes have been registered, participants reveal their seeds, which are combined to produce a random value. Randao has linear computational complexity but is vulnerable to look-ahead attacks. B-Rand utilises the idea of registering the hash of a locally generated secret on a blockchain so that the secret itself becomes publicly verifiable in the future.

Ginar [49] allows individual participants to request a random number in cooperation with a set of disintermediated participants on a blockchain system. Each participant establishes their eligibility to participate in a request by using a verifiable random function [50] to determine their eligibility threshold. Eligible participants then encrypt a secret value with the requester's public key and submit it to the blockchain. Using the homomorphic property of Elgamal encryption (Section II D), the requester can decrypt the sum of all the encrypted secrets on the blockchain to receive its random number. The computational complexity of Ginar is linear [33]. The innovative use of homomorphic encryption described in [33] was used in the B-Rand protocol.

Single Secret Leader Election (SSLE) is a protocol devised by [51] to enable the selection of a random block proposer (leader) in a blockchain system. SSLE uses threshold fully homomorphic encryption [52] to obfuscate the leader (only the leader knows that it is the leader) until it reveals itself. This protects against a denial-of-service attack by a malicious party on the leader. SSLE relies on public randomness beacons, the choice of which is left to the designer of the final SSLE implementation. SSLE requires participants to register to participate in the election process during the setup phase, where repeated elections are possible until there is a change in participants [51]. Time complexity has not been reported by the authors, but they state that most of the computational load goes into registering the participants during the setup phase, presumably with quadratic complexity. This initial setup investment is then amortised over many election rounds. The random number seeds provided by B-Rand may provide a useful source of randomness for SSLE. B-Rand continues the convenient n-secret naming convention in [51].

Verifiable Delay Functions (VDF's) [53] address the problem inherent in multi-party commit/reveal schemes where an adversary or group of adversaries attempt to manipulate the outcome of a pseudo-random output or force a failure of the process by not revealing their secrets. It uses an evaluation algorithm that requires an arbitrary number of steps to calculate the output, of which the correctness can be verified easily. The sequential nature of the calculation means that an adversary cannot speed up the result through parallelisation. VDF's consist of three algorithms, namely, the setup algorithm, which determines the environment in which the VDF will be evaluated. The evaluation algorithm consists of sequentially computing the output and proof of its correctness, and the verification algorithm allows a verifier to confirm that the prover completed the evaluation algorithm correctly. Through simulation of VDF performance, [54] has found that VDF evaluation algorithms are linear in the security parameter specifying the bit-length of the output and verification algorithms exhibit constant time complexity. VDF's show promise as a source of randomness for blockchain consensus, but it is not clear how to run the setup algorithm efficiently in a distributed manner. Setup may therefore require a trusted source that does not participate in the consensus process [54].

IV. PROPOSED APPROACH

The B-Rand protocol proposes a method whereby a seed for random number generation can be appended to each transaction when a proposer adds the transaction to the blockchain. Note that the seed, once recorded on the blockchain as part of a transaction, is known to and intended for use by, each transaction owner. The owner decides when to use the seed. If the seed is intended for selecting a new block proposer, the owner must decide the opportune time to employ the seed (for example, when it maximises its chances of success). This is made clearer in the example in Section IV C. Each seed has five important properties.

The seed is *confidential*, meaning that it is known only to the owner of the transaction until the time that they choose to expose it.

The seed is *tamper-resistant* and *unpredictable* because no party, including the owner and block proposer, can employ any meaningful strategy to manipulate it.

Seeds are made *collision-resistant* by using suitable cryptographic hash functions. Suitable in this context refers to a hash function that complies with current standards, such as the SHA-2 family of hash functions specified by the Internet Engineering Task Force (IETF) for TLS protocol 1.3 [55].

The seeds are *publicly verifiable*. Once an owner utilises the seed in a public process, any party can verify that the seed was not manipulated.

The seed generation process is *scalable* as it adds little computational and space complexity to the operation of the blockchain system. The B-Rand protocol has two variants: Single Secret and Double Secret B-Rand. These two variants are discussed in Sections IV A and IV B, respectively.

A. SINGLE SECRET B-RAND

Single Secret B-Rand requires that the transaction owner insert a secret into the transaction at the time of creation which, once confirmed on the blockchain, serves as an immutable and publicly verifiable seed generator in the future. The protocol consists of three sub-protocols: transaction construction, seed retrieval, and verification.

1) SINGLE SECRET B-RAND TRANSACTION CONSTRUCTION

The Single Secret B-Rand transaction construction consists of three steps. These three steps, executed by each owner during transaction creation, are listed in Table 2.

TABLE 2. Steps in Single Secret B-Rand transaction creation.

Step	Operation
1	OwnerSeed = Locally generated pseudo-random output by transaction owner
2	OwnerSeedHash = hash (OwnerSeed)
3	Append to transaction data: OwnerSeedHash

The owner creates a high-quality pseudo-random number (OwnerSeed) using the technique of Schindler (2009), or more practically, the built-in cryptographic service available in some programming languages. An example of such a service is the cryptographic service available in.Net [56]. The owner then appends the hash of OwnerSeed to the transaction data. Because OwnerSeedHash is part of the transaction data, it is included in TxHash. Fig. 4. shows the structure of the transaction created by the owner.

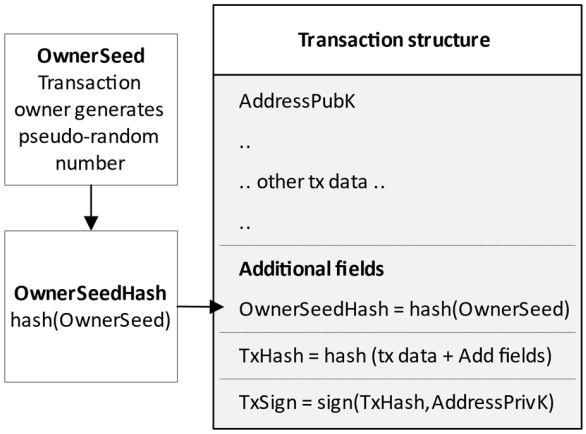

FIGURE 4. Single Secret B-Rand transaction construction.

Transaction construction can be performed by each owner in constant time and adds a linear space requirement to the block (O(n), where n is the number of transactions in the block).

2) SINGLE SECRET B-RAND SEED RETRIEVAL PROTOCOL

When the transaction is recorded on the blockchain, the owner retrieves FinalSeed using the steps in Table 3.

TABLE 3. Steps in Single Secret B-Rand seed retrieval.

Step	Operation
1	MerkleSeed = hash (MerkleTreeRootHash, TxHash)
2	FinalSeed = hash (OwnerSeed, MerkleSeed)

Seed retrieval allows the owner to generate a pseudo-random number for use in a blockchain process, for example, block proposal, an input to a smart contract, or any functionality that requires tamper-resistant, publicly verifiable random numbers.

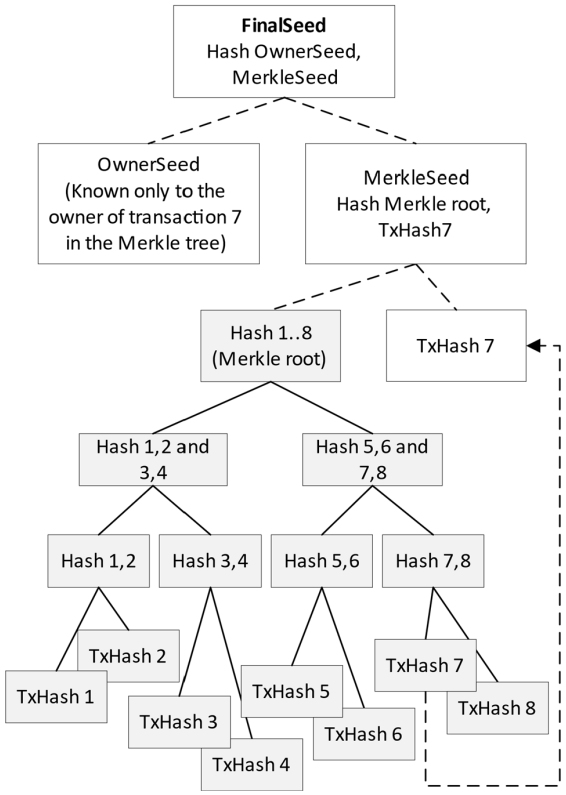

FIGURE 5. Single Secret B-Rand seed retrieval from the transaction Merkle tree.

The owner first calculates a second seed (MerkleSeed) from the transaction Merkle tree data. This requires hashing the Merkle root (MerkleTreeRootHash) with a transaction identifier (TxHash). Second, the owner calculates the hash of OwnerSeed and MerkleSeed to produce FinalSeed. Fig. 5 illustrates the hypothetical situation in which the owner of transaction number seven calculates their MerkleSeed (The original Merkle tree is shown in grey).

The motivation for the MerkleSeed calculation is five-fold. First, the Merkle root contributes the entropy of all the transactions in the transaction block. Second, the TxHash differentiates each owner's Merkle seed from all the other owners in the block. Third, the calculation is efficient as the seed can be retrieved by the owner in constant time. Fourth, since the proposer does not know the value of OwnerSeed, it cannot manipulate the block data in any meaningful way to influence FinalSeed, provided that the owner and the proposer are not the same entity. Finally, MerkleSeed is publicly verifiable from data on the blockchain.

3) SINGLE SECRET B-RAND VERIFICATION PROTOCOL

When the owner uses FinalSeed in a process that requires public verification, they must expose both OwnerSeed and FinalSeed (Fig. 6).

This may be required where the owner, for example, proposed a new block based on the authority of the random number generated from the FinalSeed. In this case, all the

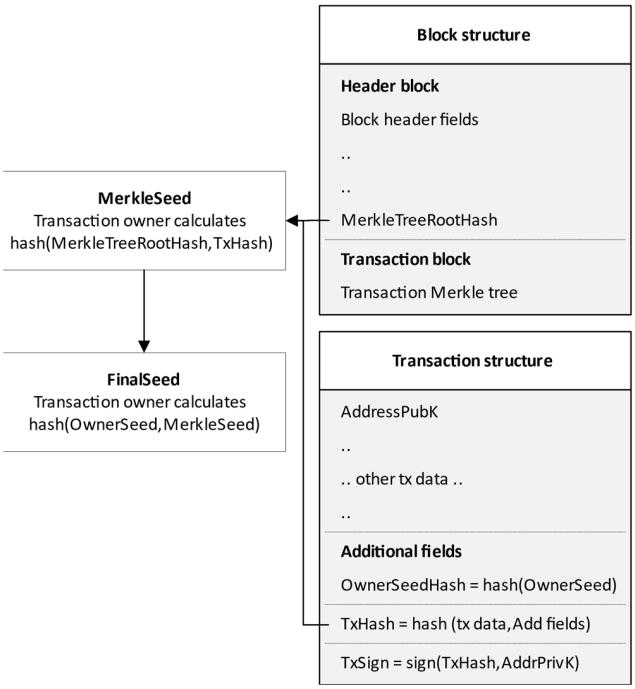

FIGURE 6. Single Secret B-Rand seed retrieval.

nodes in the network verify the validity of the random number, and thus the authority of the proposer. The steps of the verification process are presented in Table 4.

TABLE 4. Steps in Single Secret B-Rand verification.

Step	Operation
1	Verifier checks: hash (OwnerSeed) = OwnerSeedHash
2	Verifier checks: FinalSeed = hash (OwnerSeed, MerkleSeed)
3	Verification pass if both 1 and 2 are TRUE, else verification fails

A verifier first checks if OwnerSeedHash is indeed the hash of OwnerSeed, and then if FinalSeed is the hash of OwnerSeed and MerkleSeed (Fig. 7). The verification process can be performed in linear time by any verifier.

The Single Secret B-Rand protocol places a part of the information for generating the seed, namely the MerkleSeed, in the public domain. Therefore, only the OwnerSeed portion remains secret until exposed by the owner. The confidentiality of seed information can be further enhanced by the Double Secret B-Rand protocol, discussed in the next section.

B. DOUBLE SECRET B-RAND

While the OwnerSeed is private in the Single Secret B-Rand protocol, the MerkleSeed part is in the public domain. The privacy of the FinalSeed generation process can be further enhanced using the Double Secret B-Rand protocol. Double Secret B-Rand requires four sub-protocols: transaction construction, block construction, seed retrieval, and verification.

1) DOUBLE SECRET B-RAND TRANSACTION CONSTRUCTION PROTOCOL

When an owner constructs a new transaction under the Double Secret B-Rand protocol, they add a small amount of additional detail to the transaction. Table 5 summarises the process to create the additional details.

TABLE 5. Steps in Double Secret B-Rand transaction creation.

Step	Operation
1	OwnerSeed = Locally generated pseudo-random output by transaction owner
2	OwnerPubK, OwnerPrivK = Public-private key pair from a partially homomorphic encryption scheme
3	OwnerSeedEncr = encrypt (OwnerSeed with OwnerPubK)
4	Append to transaction data: OwnerSeedEncr, OwnerPubK

The owner creates a high-quality pseudo-random number (OwnerSeed) in the same manner as described in Single Secret B-Rand. The owner then generates a new public-private key pair (OwnerPubK and OwnerPivK) using a partially homomorphic encryption scheme and, using OwnerPubK, encrypts OwnerSeed to produce OwnerSeedEncr. Two fields are now appended to the transaction: OwnerSecretEncr and OwnerPubK. Additional fields added during the transaction construction process are shown in Fig. 8.

Note that since the two fields (OwnerSeedEncr and OwnerPubK) appended to the transaction are now part of the transaction data, the TxHash and TxSign. Recall from Section II A that TxSign is the signature of TxHash where the private key corresponds to the owner's address (AddressPubKey). The Double Secret B-Rand transaction construction protocol can be executed by the owner in constant time, adding constant space complexity to each transaction and linear space complexity (number of transactions) to the transaction block.

2) DOUBLE SECRET B-RAND BLOCK CONSTRUCTION PROTOCOL

When a block proposer incorporates the transaction into a transaction block, it follows the four steps listed in Table 6.

TABLE 6. Steps in Double Secret B-Rand block creation.

Step	Operation
1	ProposerSeed = Locally generated pseudo-random output by the block proposer
2	ProposerSeedEncr = Encrypt (ProposerSeed with OwnerPubK)
3	SumEncr = OwnerSeedEncr + ProposerSeedEncr
4	Append to transaction data: SumEncr

For each transaction, the block proposer produces a high-quality pseudo-random number (ProposerSeed) in the

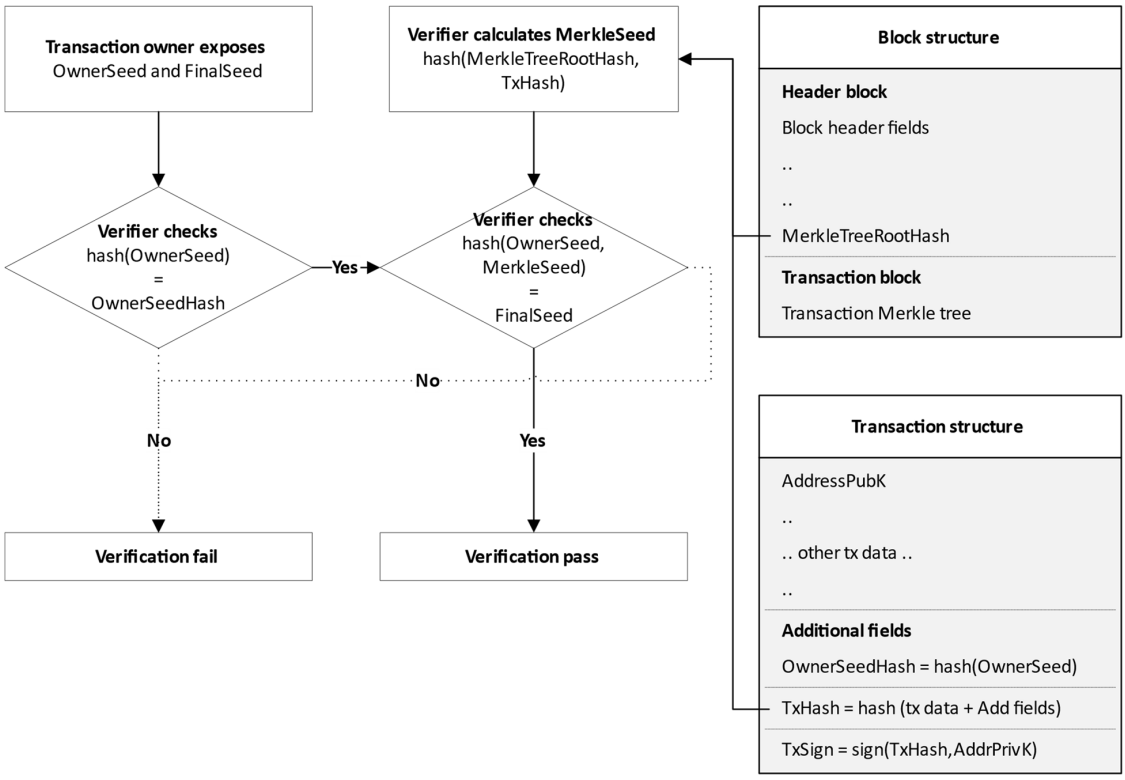

FIGURE 7. Single Secret B-Rand seed verification.

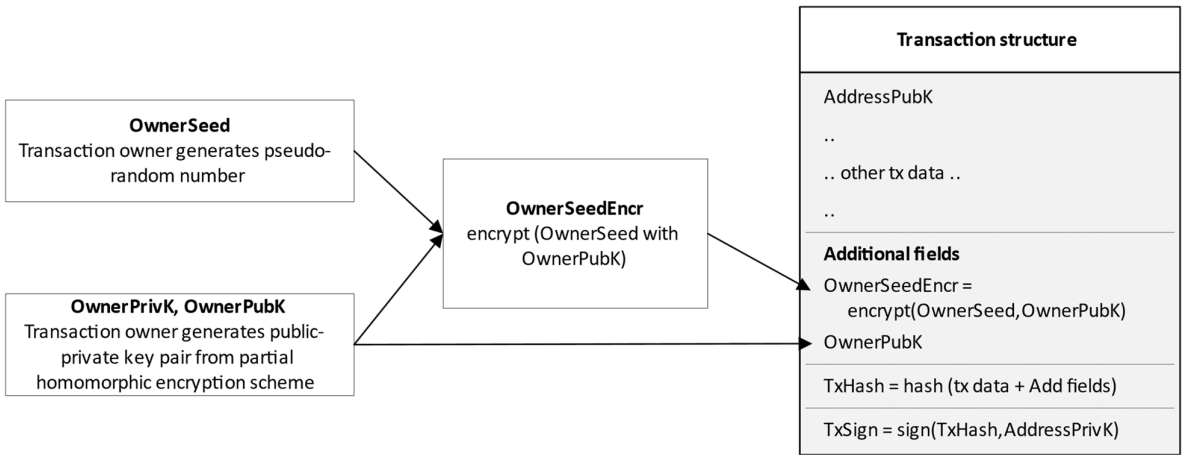

FIGURE 8. Double Secret B-Rand transaction construction.

same manner as described for Single Secret B-Rand. It then encrypts ProposerSeed with the owner's public key (OwnerPubK, the same key used to encrypt OwnerSeed), denoted as ProposerSeedEncr. The proposer then calculates the sum of OwnerSeedEncr and ProposerSeedEncr, denoted as SumEncr. Finally, the proposer appends SumEncr to the transaction.

The computational and space complexity is O(1) for each transaction and O(n) for the transaction block. Fig. 9 shows how the final transaction structure is stored in the blockchain by the block proposer.

Note that the SumEncr field is situated outside of the TxHash data. This means that these fields are not included in the Merkle tree hash process and therefore do not affect the eventual block hash. There is more than one way to overcome this problem. All the SumEncr fields may be hashed together in a new SumEncrHash field in the block header, or the SumEncr fields may be stored in a separate Merkle tree with

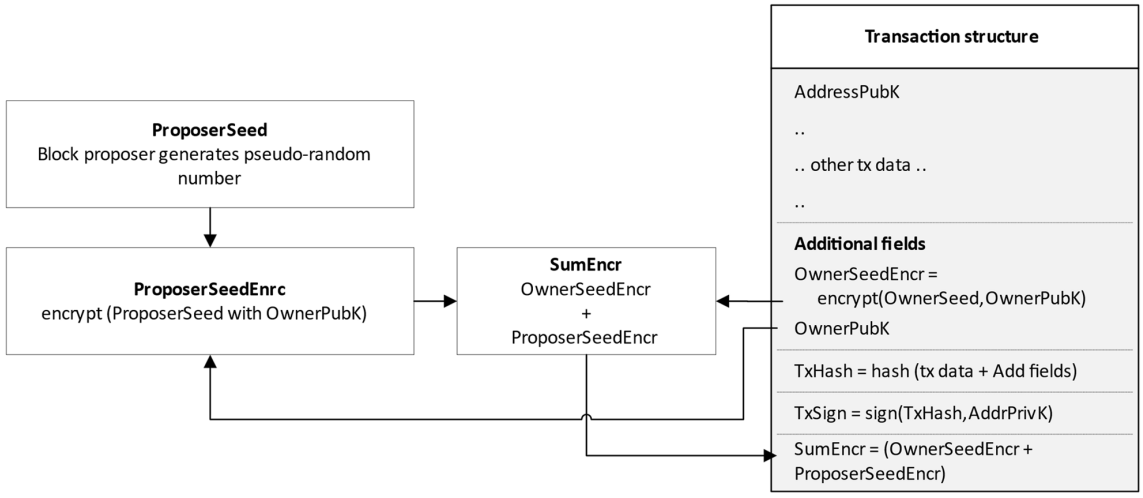

FIGURE 9. Double Secret B-Rand final transaction structure.

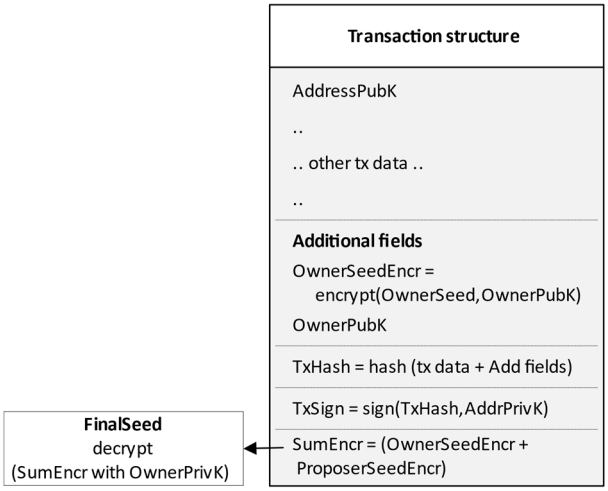

FIGURE 10. Double Secret B-Rand seed retrieval.

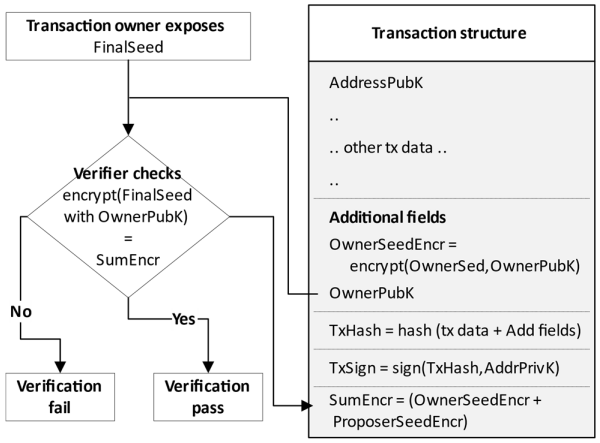

FIGURE 11. Double Secret B-Rand seed verification.

a root hash. The individual design of a blockchain system dictates the ultimate solution.

3) DOUBLE SECRET B-RAND SEED RETRIEVAL PROTOCOL

Once a transaction is recorded on the blockchain, each owner can retrieve its seed (FinalSeed), which is the decrypted value of SumEncr using the owner's private key (refer to Table 1 for a summary of encryption/decryption services). Table 7 summarises the FinalSeed retrieval process.

TABLE 7. Steps in Double Secret B-Rand seed retrieval.

Step	Operation
1	FinalSeed = decrypt(SumEncr with OwnerPrivK)

The homomorphic property of the encryption scheme makes it possible for the owner to calculate the sum of OwnerSeed and ProposerSeed by decrypting SumEncr with OwnerPrivK to yield FinalSeed (2).

$$\begin{aligned} SumEncr &= OwnerSeedEncr + ProposerSeedEncr \\ &= encrypt\,(OwnerSeed) \\ &\quad + encrypt\,(ProposerSeed) \\ and & \\ FinalSeed &= decrypt\,(SumEncr) \\ &= OwnerSeed + ProposerSeed \end{aligned} \tag{2}$$

The seed retrieval process is shown in Fig. 10 and can be achieved in constant time.

4) DOUBLE SECRET B-RAND VERIFICATION PROTOCOL

When the owner decides to use FinalSeed, for example, to generate a random number for participating in the block proposal process or as input required by a smart contract,

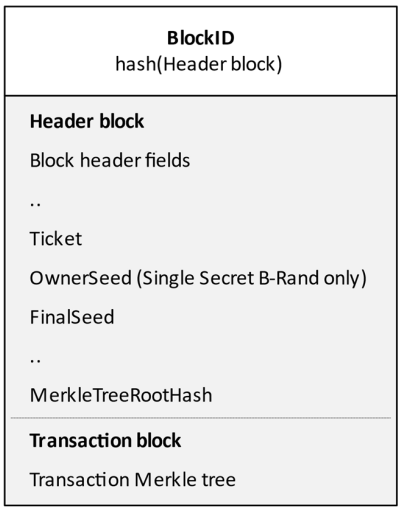

FIGURE 12. Candidate block.

FinalSeed must be exposed. A verifier can then check the validity of FinalSeed in the two steps listed in Table 8.

TABLE 8. Steps in Double Secret B-Rand verification.

Step	Operation
1	Verifier checks: encrypt (FinalSeed with OwnerPubK) = SumEncr
2	Verification pass if TRUE, else verification fails

The verifier encrypts the FinalSeed provided by the owner with OwnerPubK on the blockchain and compares it with SumEncr. The verification process is shown in Fig. 11, and it is possible to calculate in linear time.

C. EXAMPLE

We illustrate the use of B-Rand through a simple example where a new block proposer is selected by way of a lottery. This can be done where candidate proposers generate a lottery ticket (Ticket) from their B-Rand random number seeds. Each candidate proposer then constructs a new transaction block and publishes its ticket information in the block header. The network then follows the consensus rule of accepting the block with the largest ticket as the next valid block to extend the blockchain.

Candidate proposers are those owners who have previously embedded the hash of its OwnerSeed (OwnerSeedHash) in a transaction. Since OwnerSeed is locally produced, the network cannot control how each owner produces it. In practice, the Rivest method for producing a pseudo-random number from a hash function should be sufficient for honest participants, as hash functions are a good source of pseudo-randomness [57]. Owners who follow an insecure method for producing the owner seed, for example, by hashing a predictable value, compromise the confidentiality of their own seed, but it has no impact on the seeds of other honest owners. OwnerSeed creation is shown in Table 9.

TABLE 9. Creating OwnerSeed locally.

Step	Operation
1	Roll a die with many sides (i.e 10 sides), many times (i.e. 100 times) to generate a random string S
2	Calculate OwnerSeed = Hash(S,t), where t is the Unix timestamp when OwnerSeed is created

The transaction can now be constructed using the B-Rand transaction creation protocols. Double Secret B-Rand requires an additional step during block creation. It requires the block proposer to generate ProposerSeed for each transaction, which can be done by an honest proposer in the same way as described in Table 9. Proposers that use insecure methods to produce ProposerSeed do not impact the Owner's FinalSeed as long as OwnerSeed has not been compromised. Once recorded on the blockchain, the owner can retrieve its FinalSeed using the B-Rand seed retrieval protocols.

Candidate proposers generate a lottery ticket at the time it constructs a new candidate block. The Ticket for proposer i, T_i is the hash of $OwnerSeed_i$ and the hash of the preceding block (H_{n-1}).

$$T_i = hash(OwnerSeed_i, H_{n-1}) \tag{3}$$

Each candidate proposer records Ticket, OwnerSeed and FinalSeed in the case of Single Secret B-Rand or Ticket and FinalSeed in the case of Double Secret B-Rand by adding the fields to the proposed block's header. Each node in the blockchain network evaluates each candidate block to ascertain the validity of OwnerSeed, FinalSeed and Ticket and accepts the valid candidate block with the largest Ticket as the new block. Since the Ticket is tied to the previous block hash, each new block round will produce a new ticket for each owner. The consensus algorithm can be adapted to place restrictions on the age of seeds that may be used, for example, a seed may need to be buried by a minimum number of blocks or must be used within a certain number of block rounds. The layout of a candidate block is shown in Fig. 12.

V. ANALYSIS

In the introduction, the authors claimed that the B-Rand protocol encodes random number seeds on a permissionless blockchain system that is *confidential*, *tamper-resistant*, *unpredictable*, *collision-resistant*, *publicly verifiable*, and *scalable*.Each of these aspects is now addressed individually.

A. CONFIDENTIAL

The information required to complete the construction of each seed is available only once the transaction is recorded on the blockchain. At this point, the owner is the only party that possesses all the required information to know the seed. It is only when the owner makes the seed known that it comes into the public domain.

B. TAMPER-RESISTANT AND UNPREDICTABLE

Since the owner has no information about the MerkleSeed (Single Secret B-Rand) or ProposerSeed (Double Secret B-Rand) before the block is published, it has no information about how its choice of the OwnerSeed will ultimately influence its FinalSeed. Likewise, since the block proposer where the transaction is recorded does not know the value of the OwnerSeed, it cannot know how its choice of the ProposerSeed (Double Secret B-Rand) will affect the final value of the seed.

However, this does not preclude a situation in which the owner and block proposer collude or are the same entity. Therefore, it may be necessary to harden FinalSeed with additional information at the time of use. For example, if the seed is used in a pseudo-random number generator to determine the owner's eligibility to propose a new block, the pseudo-random number generator may combine the owner's seed with the previous block hash of the proposed new block. The choice of this type of seed hardening is entirely dependent on its type of use in the blockchain system.

C. COLLISION-RESISTANT

Enforcing the requirement that the initial random numbers OwnerSeed and ProposerSeed, are the outputs from cryptographic hash functions (suitable hash functions are described in Section IV), negates the probability that two transactions share the same seed. This provides the possibility that unique selection criteria can be constructed for participants in blockchain functions controlled by pseudo-random number generation.

D. PUBLICLY VERIFIABLE

The verification protocols established for both Single and Double Secret B-Rand make it impossible for owners to alter their choice of the OwnerSeed once the transaction is recorded on the blockchain. By designing the blockchain system in a manner that forces the owner to publicly expose the OwnerSeed and FinalSeed (Single Secret B-Rand) or the FinalSeed (Double Secret B-Rand) as is the current case with, for instance, the nonce value in proof-of-work [1], enables unambiguous public verification.

E. SCALABLE

The B-Rand protocol adds minimal additional computational and space complexity to the blockchain system. The computational and space complexities associated with B-Rand are summarised in Table 10.

F. NOVELTY, COMPETITIVENESS AND OPEN QUESTIONS

The example in Section IV-C, set out a simple method whereby B-Rand can be used as part of a consensus algorithm for new block selection. It must be noted that the example is not meant to demonstrate a new full-scale consensus algorithm, as it is not the purpose of the paper. Several aspects must still be investigated in this regard.

TABLE 10. Computational and space complexity of B-Rand protocols.

Protocol	Type of complexity	Singe Secret B-Rand	Double Secret B-Rand
Transaction construction	Computation	O(1)	O(1)
	Space	O(1)	O(1)
Block construction	Computation	n/a	O(n)
	Space	O(n)*	O(n)*
Seed retrieval	Computation	O(1)	O(1)
	Space	n/a	n/a
Verification	Computation	O(n)*	O(n)*
	Space	n/a	n/a

* n = transactions per block

First, by evaluating the history (mean and standard deviation) of previously successful tickets, owners may decide on opportune rounds to propose candidate blocks. The same applies to nodes that receive and then propagate new blocks to the network. A probabilistic extinction model is still needed to evaluate the tradeoff between limiting network traffic and ensuring timely block addition.

Second, it may be necessary to restrict FinalSeed until it is buried under a minimum number of blocks and also limit its lifespan to a maximum number of blocks. These parameters must still be determined.

The block selection example is, however, useful in evaluating the novelty and competitiveness of a B-Rand based application. Since the maximum computational complexity of B-Rand is O(n) where n represents the number of transaction per block. This suggests that candidate blocks can be produced and evaluated at a very low cost. This improves on Randao [48] and Ginar [33], of which Ginar is the only protocol designed to be self-contained in a blockchain system. Randao requires the use of an outside service on the Ethereum network.

A particularly novel aspect of B-Rand in the context of block selection is that it does not require the assets on the blockchain to be objectively comparable. For example, the PoS consensus model compares the value of assets owned by stakeholders to decide which party has the right to produce a new block [7]. If the blockchain were to contain other types of assets, for example, intellectual property rights or logistical information instead of a cryptocurrency, this type of value comparison may not be possible, but it will not impact the B-Rand protocol.

G. THREATS

There are four main security risks related to the B-Rand protocol. First, since participation is computationally cheap, it may result in the network being flooded with information, for example, when it is used in the construction of candidate blocks during each block round. This poses the risk of a denial of service attack on the network. Mitigation strategies such

as probabilistic extinction models may be useful in the block addition example (Section IV C), where each node evaluates the probability of a candidate's success based on the historical distribution of successful blocks.

Second, proposers may withhold potentially successful blocks in an attempt to later launch a long-range, post-hoc fork on the blockchain by submitting the withheld blocks later. This vulnerability has been identified in PoS type blockchain systems where a majority of stakeholders who have already divested from the blockchain attempt to rewrite a large portion of the blockchain history [53].

Third, is the possibility of owners and block proposers colluding to manipulate random number seeds (For the sake of the example in IV-C, it was assumed that all parties are honest). This threat can be addressed by hardening the FinalSeed with new information on the blockchain when it is used to generate a pseudo-random number. In the example, the seed was hardened with the previous block hash to produce the Ticket.

Last, owners or block proposers may also generate trivial OwnerSeed or ProposerSeed values by, for example, hashing empty strings. It is not known how these actions may impact the robustness of the solution.

VI. CONCLUSION

In this study, the authors proposed two methods for embedding random number seeds in permissionless blockchain systems. The Single Secret B-Rand protocol is simpler to implement than the Double Secret B-Rand protocol but leaves a portion of the seed information in the public domain. While more complicated, the Double Secret B-Rand protocol provides enhanced seed confidentiality. In both cases, B-Rand provides an efficient way to prepare a blockchain system for future participation by stakeholders based on random selection. It does so by seeding the blockchain with *confidential*, *tamper-resistant*, *unpredictable*, *collision-resistant*, *publicly verifiable* random number seeds in a way that scales efficiently. The paper illustrates the use of B-Rand random number seeds in an example of a block addition lottery, but how each blockchain designer chooses to implement functionality that utilises random number seeds is up to them. Exploring these opportunities provides many avenues for future research, and the detailed description provided for each of the two variants of B-Rand should simplify the task for interested investigators.

In addition to studying the utilisation of B-Rand random number seeds, there are also other open questions. The most notable of these is the possible attack vector on seed generation where the transaction owner and block proposer collude or are the same entity. This may require the development of additional rules about, for example, seed hardening, seed expiry and seed maturation. These rules do not have to be static and dictated by the B-Rand protocol but may be adapted to suit individual implementation requirements.

As the use of blockchain systems continues to grow and more blockchain systems come into existence, the requirement for a better foundational blockchain infrastructure becomes paramount. B-Rand aims to strengthen these foundations to enable the imagining of newer and better permissionless blockchain systems.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, p. 21260, Oct. 2008.
[2] A. de Vries, "Bitcoin's growing energy problem," *Joule*, vol. 2, no. 5, pp. 801–805, May 2018.
[3] U. S. Energy Information Administration. (2019). *International Electricity Consumption*. [Online]. Available: https://www.eia.gov
[4] J. Ma, J. S. Gans, and R. Tourky, "Market structure in bitcoin mining," Nat. Bur. Econ. Res., Cambridge, MA, USA, Tech. Rep., 24242, 2018.
[5] V. Buterin, "A next generation smart contract & decentralized application platform," Ethereum Found., White Paper, 2013.
[6] 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 (BigData Congress)*, Jun. 2017, pp. 557–564.
[7] I. Bentov, A. Gabizon, and A. Mizrahi, "Cryptocurrencies without proof of work," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2016, pp. 142–157.
[8] NEM. (2018). *Proof-of-Importance*. NEM Technical Reference. Accessed: Nov. 12, 2019. [Online]. Available: https://nemplatform.com/wp-content/uploads/2020/05/NEM_techRef.pdf
[9] A. Hasib. (2018). *101 Blockchains*. Accessed: Nov. 12, 2019. [Online]. Available: https://101blockchains.com/consensus-algorithms-blockchain/#6
[10] M. Milutinovic, W. He, H. Wu, and M. Kanwal, "Proof of luck: An efficient blockchain consensus protocol," in *Proc. 1st Workshop Syst. Softw. Trusted Execution*, 2016, pp. 1–6.
[11] 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
[12] 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
[13] *Slimcoin: A Peer-to-Peer Crypto-Currency With Proof-of-Burn*, Titan, Bengaluru, India, 2014.
[14] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, "Proofs of space," in *Proc. Int. Cryptol. Conf.*, 2015, pp. 585–605.
[15] J. Blocki and H.-S. Zhou, "Designing proof of human-work puzzles for cryptocurrency and beyond," in *Theory of Cryptography*. Cham, Switzerland: Springer, 2016, pp. 517–546.
[16] BitShares Blockchain Foundation. *Delegated Proof-of-Stake Consensus*. Accessed: Aug. 11, 2019. [Online]. Available: https://bitshares.org/technology/delegated-proof-of-stake-consensus/
[17] Binance Capital Mgmt. (2021). *CoinMarketCap*. [Online]. Available: https://coinmarketcap.com/
[18] 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.
[19] W. Schindler, "Random number generators for cryptographic applications," in *Cryptographic Engineering*, Ç. K. Koç, Ed. Boston, MA, USA: Springer, 2009, pp. 5–23.
[20] P. Tasca and C. J. Tessone, "A taxonomy of blockchain technologies: Principles of identification and classification," 2019, *arXiv:1708.04872*.
[21] F. Glaser and L. Bezzenberger, "Beyond cryptocurrencies—A taxonomy of decentralized consensus systems," in *Proc. 23rd Eur. Conf. Inf. Syst. (ECIS)*, 2015, pp. 1–19.
[22] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. Nyang, and D. Mohaisen, "Exploring the attack surface of blockchain: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1977–2008, 3rd Quart., 2020.
[23] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," *Bus. Inf. Syst. Eng.*, vol. 59, no. 3, pp. 183–187, 2017.
[24] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A taxonomy of blockchain-based systems for architecture design," in *Proc. IEEE 6th Int. Congr. Softw. Archit.*, Apr. 2017, pp. 243–252.

[25] 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.
[26] R. Bezuidenhout, W. Nel, and A. Burger, "Nonlinear proof-of-work: Improving the energy efficiency of Bitcoin mining," *J. Construct. Project Manage. Innov.*, vol. 10, no. 1, pp. 20–32, Sep. 2020.
[27] R. Dahlberg, T. Pulls, and R. Peeters, "Efficient sparse Merkle trees caching strategies and secure (non-)membership proofs," in *Proc. 21st Nordic Workshop Secure Comput. Syst.*, 2016, pp. 1–16.
[28] H. Massias, X. S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirement," in *Proc. 20th Symp. Inf. Theory Benelux*, 2002, pp. 1–8.
[29] B. O. Mulár, "Blockchain technology in the enterprise environment," M.S. thesis, Dept. Inform., Masaryk Univ., Brno, Czechia, 2018.
[30] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies*. Princeton, NJ, USA: Princeton Univ. Press, 2016.
[31] K. Rabah, "Elliptic curve ElGamal encryption and signature schemes," *Inf. Technol. J.*, vol. 4, no. 3, pp. 299–306, Jun. 2005.
[32] C. Paar and J. Petzl, *Understanding Cryptography*. Heidelberg, Germany: Springer, 2010.
[33] 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.
[34] M. Andrychowicz and S. Dziembowski, "Distributed cryptography based on the proofs of work," *IACR Cryptol. ePrint Arch.*, vol. 2014, p. 796, Dec. 2014.
[35] I. Cascudo and B. David, "SCRAPE: Scalable randomness attested by public entities," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2017, pp. 537–556.
[36] 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.
[37] 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.
[38] 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.
[39] P. Schindler, A. Judmayer, N. Stifter, and E. Weippl, "HydRand: Practical continuous distributed randomness," *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 319, Aug. 2018.
[40] T. Hanke, M. Movahedi, and D. Williams, "DFINITY technology overview series, consensus system," 2018, *arXiv:1805.04548*.
[41] B. Libert, M. Joye, and M. Yung, "Born and raised distributively: Fully distributed non-interactive adaptively-secure threshold signatures with short shares," in *Proc. Annu. ACM Symp. Princ. Distrib. Comput.*, 2014, pp. 303–312.
[42] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *Advances in Cryptology—ASIACRYPT*. Cham, Switzerland: Springer, 2001, pp. 514–532.
[43] 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.
[44] C. Holotescu, "Understanding blockchain technology and how to get involved," in *Proc. 14th Int. Sci. Conf. eLearn. Softw. Educ.*, Bucharest, Romania, Apr. 2018, pp. 1–22.
[45] I.-C. Lin and T.-C. Liao, "A survey of blockchain security issues and challenges," *Int. J. Netw. Secur.*, vol. 19, no. 5, pp. 653–659, Sep. 2017.
[46] 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.
[47] C. Cachin and M. Vukolic, "Blockchains consensus protocols in the wild," in *Proc. 31st Int. Symp. Distrib. Comput.*, Oct. 2017, pp. 1-1–1-16.
[48] M. Alturki and G. Roşu, "Statistical model checking of RANDAO's resilience to pre-computed reveal strategies," in *Formal Methods. FM 2019 International Workshops*, G. Goos J. Hartmanis, Eds. Princeton, NJ, USA: Springer, 2020, pp. 337–349.
[49] T. Nguyen-Van, T.-D. Le, T. Nguyen-Anh, M.-P. Nguyen-Ho, T. Nguyen-Van, M.-Q. Le-Tran, Q. N. Le, H. Pham, and K. Nguyen-An, "A system for scalable decentralized random number generation," in *Proc. IEEE 23rd Int. Enterprise Distrib. Object Comput. Workshop (EDOCW)*, Oct. 2019, pp. 100–103.
[50] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," in *Proc. 8th Int. Workshop Public Key Cryptogr.*, vol. 3386, 1970, pp. 416–431.
[51] 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.
[52] D. Bonehy, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai, "Threshold cryptosystems from threshold fully homomorphic encryption," *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 10991. Cham, Switzerland: Springer, 2018, pp. 565–596.
[53] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *Advances in Cryptology—CRYPTO*. Cham, Switzerland: Springer, 2018, pp. 757–788.
[54] V. Fuchs-Attias, L. Vigneri, and V. Dimitrov, "Implementation study of two verifiable delay functions," in *Tokenomics*. Wadern, Germany: Dagstuhl, 2020, 2021.
[55] *The Transport Layer Security (TLS) Protocol Version 1.3*, Internet Engineering Steering Group, Internet Eng. Task Force, Fremont, CA, USA, 2018.
[56] Microsoft. (2021). *Cryptographic Services*. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/standard/security/cryptographic-services
[57] P. B. Stark and K. Ottoboni, "Random sampling: Practice makes imperfect," 2018, *arXiv:1810.10985*.

RIAAN BEZUIDENHOUT was born in King William's Town, 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, and the B.Sc. (Hons.) and M.Sc. degrees 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 is also an Assistant Researcher with the Department of Computer Science and Informatics, University of the Free State. He worked in management consulting, until 2012, before returning to university, in 2013, to study computer science. 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, South Africa. From 2005 to 2006, he was a Lecturer in computer science with 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, Bloemfontein. 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 in physics and the Ph.D. degree in astrophysics from the University of the Free State, South Africa, in 2014 and 2017, respectively. He has been a Lecturer in engineering sciences at the University of the Free State, since 2017. His research interests include physics, astrophysics, energy modeling, energy analytics, energy AI, and power systems.