Journal of Cellular Automata, Vol. 0, pp. 1–15
Reprints available directly from the publisher
Photocopying permitted by license only

RandomBlocks: A Transparent, Verifiable Blockchain-based System for Random Numbers

KLITOS CHRISTODOULOU[1,2,*], SAVVAS A. CHATZICHRISTOFIS[3], GEORGIOS CH. SIRAKOULIS[4] AND PANAYIOTIS CHRISTODOULOU[2,5]

[1]*Institute For the Future (IFF), University of Nicosia, Cyprus*

[2]*Thriller Web Solutions Ltd, Cyprus*

[3]*Intelligent Systems Laboratory (ISLab), Department of Computer Science, Neapolis University Pafos, Cyprus*

[4]*Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece*

[5]*Department of Electrical Eng., Computer Eng. and Informatics, Cyprus University of Technology*

Received: June 12, 2019. Accepted: July 1, 2019.

Games-of-chance require high-levels of trust between participants that is often uncertain and difficult to enforce. The unique characteristics introduced by the *blockchain* technology can be leveraged to inform a verifiable but transparent *pseudo-random* generation process that at the same time eliminates the need of an external trustee service or oracle to guarantee the fair execution of the process. In this paper we propose a process that builds on a one-dimensional *Cellular Automaton (CA)* where the evolution of the CA aligns to the evolution of a blockchain. Given an initial seed defined as the point in time where the evolution of the CA is triggered, the set of transition rules, along with the finite number of evolution steps, any external oracle is able to verify and backtrack (but not predict) the outcome of the process. The effectiveness of the implemented system is confirmed by the use of various statistical testing suites that confirm the existence of a set of statistical properties required to produce sufficient pseudo-random number sequences.

Keywords: Blockchain, pseudo-random number generation, cellular automata

1 INTRODUCTION

The blockchain protocol with its first application, the Bitcoin peer-to-peer (p2p) payment system [24] has demonstrated a mechanism of global consensus across many distributed nodes introducing a distributed way of

* Contact author: E-mail: christodoulou.kl@unic.ac.cy

performing financial transactions, without the need of a centralized entity. This idea motivated a recent explosion of interest towards distributed p2p networks proposed as solutions to the so called *Byzantine Fault Tolerance* problem discussed in the literature of distributed and decentralized systems [22]. Decentralization is only one of the main characteristics introduced by the blockchain technology. Examining the anatomy behind this technology, the blockchain protocol enables a secure mechanism of issuing transactions that are recorded on a public distributed database (known as the *ledger*) in a way that is secure, immutable, anonymous but transparent* , and *trustless*.

Although the absence of a central authority may considered to be a vulnerability of the technology, the protocol ensures that transactions between parties are secured, even though by nature parties are distributed and minimum trust is assumed between them. This is made possible by the use of cryptographic rules that ensure the authorization, and validity of the transactions issued. In addition to the aforementioned key characteristics, the capability of issuing self-executable scripts (i.e., *smart contracts*) that exist on the blockchain and encouraged by a variety of p2p networks, such as, the Ethereum Virtual Machine [44], enables the execution of automated workflows on the network.

This new dimension along with the unique characteristics introduced by the blockchain technology attracted the interest of the industry and academia in a wide spectrum of applications, ranging from: finance, to healthcare, and from the governmental sector [7] to various online games of chance (e.g., prize competitions, and lotteries) [12]. The emergence of online gambling and prize competition platforms is increasing; however, they are subject to a variety of challenges due to regulatory restrictions, unfair processes, and fraud. In more detail, such platforms are facing difficulties with regards to ensuring the transparency and fairness in the process of determining the winner.

In general, participants to a *game-of-chance* (e.g., prize competition) exchange their probability of winning by buying a number of tickets of some fixed price from the issuing authority. Assuming that the rules of the competition are met, a *pseudo*-random process is utilized to determine the winning ticket electronically. For this random process, these platforms often employ external regulatory procedures and protocols or rely entirely to external third-parties (or centralized authorities) to ensure fairness of the competition, and for providing a degree of trust to the participants. Ultimately the goal of such platforms is to gain the trust of the participants.

In this paper, we argue that participants should not rely trust entirely to a centralized authority to prevent fraud. Instead, the random generation

* by publicly exposing cryptographic transactions the Blockchain protocol enables any third party to verify the issuance of a transaction and examine its validity in a *pseudo*-anonymous manner.

process should be fundamentally highly transparent, at the same time, providing sufficient evidence that elevate the process itself to be *verifiable*, thus, ensuring its intended execution, and at the same time eliminating the risk of any potential external interference. To meet these requirements we propose a methodology that leverages the key characteristics introduced by the blockchain technology with an alignment to one-dimensional (`1-d`) Cellular Automaton (CA) [43] to enable a transparent and a verifiable *pseudo*-random generation process.

The effectiveness of the implemented system is demonstrated by an instantiation that harvests different sources of entropy from a transparent blockchain protocol enhanced with additional publicly available uncertain processes (e.g., monitoring daily cryptocurrencies price fluctuations). The pseudo-random sequences generated by the current instantiation of the system are challenged against both the *Dieharder* ver. 3.31.1 [3] and *NIST SP* 800-22 [29] statistical testing suites. For the proposed system data that are made publicly available beforehand are utilized as sources of entropy to inform the evolution of a CA. At the same time, their publishable nature enables any external oracle to verify the process, and reproduce the outcome using the seed state and the algorithm configuration alone. Although, the algorithm has a deterministic nature towards rendering pseudo-random number sequences any obvious patterns are practically challenging to predict.

The remainder of this paper is structured as follows. Sections 2 and 3 provide a brief introduction to random processes in Blockchain as well to CA pseudo-random generators. Section 5 illustrates how verifiable data (Section 4) from the evolution of a distributed ledger can be utilized as seeds to trigger a pseudo-random process that builds on a 1-d CA, while the specificities of the CA used by the proposed system are described in Section 6. Finally, experimental details, results, as well as, tests in different suites are presented in Sections 7 and 8, respectively. Conclusions are drawn in Section 9.

2 RANDOM PROCESSES IN BLOCKCHAINS

The implementation of pseudo-random processes on a Blockchain-based protocol is a challenging task due to its deterministic nature. This characteristic introduces challenges for gambling applications in maintaining fairness in their underlying processes; an important factor to gain users' trust. Although the deterministic dimension of various Blockchain-based protocols (e.g., Bitcoin, Ethereum) is considered important to ensure that the execution of the same sequence of instructions (given the same input data) will always produce the same outcome, it does create several flaws that can compromise the random generation process. One major flaw is that it prohibits the

introduction of enough entropy to a true-random generation process. Thus, providing no guarantees that the outcome cannot be predicted.

Since Bitcoin does not have any seed mechanism in place, Smart Contracts[†] deployed on the Ethereum blockchain attempt to mitigate the risk of predicting the outcome of a random process by introducing a variable that holds a large "secret" number. Notwithstanding that each contract is translated to bytecode, data are recorded in storage, therefore running the risk of revealing that "secret" number. In our work, the initial seed used for the evolution steps enforced by the CA is generated by a combination of data using a series of blockchain variables (e.g., timestamp of blocks, blockhash, difficulty etc.), enhanced with external sources of entropy (e.g., crypto-currency prices fluctuations [6]). We note that the instantiation of the CA algorithm is not fixed.

Another known issue is to use information from the current block as a source of entropy. This causes issues, since the blockhash value returned by the Ethereum Virtual Machine (EVM) is obvious[‡], thus can be easily predicted. A safer alternative choice is the use of an external off-chain service provider such as Oraclize [27]. The hardware device used by Oraclize provides no transparent evidence of its usage, and thus it goes against the whole blockchain philosophy.

Bonneau *et al.* describe a formalization on how Bitcoin could become a source of publicly available randomness [2]. In this paper, a methodology that can be instantiated with any source of entropy to establish enough credibility towards a process that generates random numbers, is described analytically. Similarly, it is mentioned that self-verification of random processes could be enabled by evidence recorded on public ledgers. BitcoinMegaLottery [1] attempts to build a provably-fair process for a lottery application using data from Bitcoin's blocks combined with information from the IDs that are assigned to Tweets. The approach of introducing Tweet IDs as a source of entropy, can be disputed without providing any guarantee that these data are generated before the chain's data, or how to prevent a malicious actor from tampering process for generating the IDs. `Random.org` is a third party service that generates and publishes 512-bit of random hashes every minute, according to [10,14]. However, the services provided by Random.org are centralized and data published on the Website are buffered. Thus, the services do not provide enough transparency on the publicly available information. In our implemented system, evidence is recorded in an immutable ledger to provide publicly available data to any external oracle that requires self-confirmation of the outcome. The evidence recorded on a public ledger is unlikely to be

[†] Smart contracts are programs that execute autonomously on a blockchain.
[‡] returns a zero value.

predicted or reversed engineered by a malicious actor. Section 8 provides evidence that the proposed system presents adequate randomness when challenged against various statistical testing suites.

On another note, it is observed that many blockchain implementations propose an alternative mode to Nakamoto Consensus for block propagation that is based on a *proof-of-stake* mechanism. Such a consensus mechanism relies upon a leader election for block coordination and propagation (e.g., [19]). The election decision is often based on a randomized election process. Similarly, Milutinovic *et al.* elaborate on a *proof-of-luck* consensus protocol [23] where the election of a leader is chosen based on the protocol's build-in random number generation mechanism. Our design of a pseudo-random generator can be used to enable additional consensus schemes based on fair random decisions.

3 CELLULAR AUTOMATA AND PSEUDO-RANDOM NUMBER GENERATION

Cellular Automata (CA) as a powerful modeling tool with unique characteristics like inherent parallelism, ubiquitous and emerging computation, have been successfully utilized for numerous application fields and, as a result, they have attracted the interest of several researchers from spanning various disciplines, e.g., from the field of robotics [15], image processing [4], [37] up to environmental modelling [11, 26] and real time applications [9, 38, 39, 45], just to name a few. Furthermore, and owing to the fact of easy hardware and parallel implementation and plentiful computational properties, CA also been applied to the design of pseudo-random number generators (PSNRs). It is also well known that pseudo-random number sequences produced by CA are found in communication, computing and digital signal processing practical applications, for example cryptography, D-A conversion, coding, VLSI testing, etc. [5, 20, 25, 32] but also on computer modeling where many simulation methods rely on random numbers, such as Brownian dynamics and Monte Carlo techniques, as well as, stochastic optimization methods like simulated annealing [36]. From a historical point of view, CA were originally applied for the generation of pseudo-random numbers by Stephen Wolfram [40], who also introduced the main advantages of CA PRNGs in terms of randomness of CA generated patterns, when compared with other well known methods cited in literature, like for example linear feedback shift registers (LFSRs) [42]. In these early works, it was shown that although CA consist of many identical components, each with a simple description, their combination has shown evidence of generating complex sequences of random patterns that experience a

diversity of unpredictable behaviors. In more recent years, there were other, also promising, attempts to combine CA principles with the development of efficient PRNGs. As a result, one-dimensional (`1-d`) and two dimensional (`2-d`) CA PRNGs have been proposed recently [8, 13, 18, 30, 34–36]. Additionally, hybrid CA have been also proposed as PSNGs, like non-linear CA in [16], or by using different implementation/substrate medium such as DNA in [31] or memristive devices in [17]. Referring to `1-d` and `12-d` implementations of CA PRNGs, the argument is between the proposed algorithmic complexity and the quality of the produced randomicity. Moreover, CAs have been evaluated as pseudo-random number generators and have been found to perform well on statistical tests [21, 28, 31].

The proposed system builds on a binary `1-d` CA defined as a grid of cells where each cell can be of two states i.e., $S = (0, 1)$ for each position as originally conceived by Wolfram. Primarily, the initial state of the CA is evolving deterministically in discrete time steps t according to a definite set of evolution rules that consider the values of its nearest neighboring cells [41, 43]. Thus, each cell is connected to r local neighbors on either side according to the size of a neighborhood n (i.e., each cell has $n = 2r + 1$ neighbors), where r defines its radius. The local transition rule defined here is as follows, $f : \{0, 1\}^n \rightarrow 0, 1$ such that, $s_1(t + 1) = f(s_{i-r}(t), \ldots, s_i(t), \ldots, s_{i+r}(t))$. The configuration of the proposed system considers $r = 1$, thus, the size of the neighborhood is $n = 3$ and the function f is transformed to $s_1(t + 1) = f(s_{i-r}(t), s_i(t), s_{i+r}(t))$. With this configuration the domain of f is the set of 2^3 triples that allows the configuration of the CA with $2^8 = 256$ distinct elementary rules [43]. Finally, we further assume that boundary conditions are applied.

Building on independent sources of entropy with data drawn from the blockchain, further enhanced with additional publicly available data from uncertain processes, the proposed system evolves a population of potential seed data that are then used as input to the CA. As discussed in Section 5 the seeding process evolves dynamically and includes verifiable and immutable data written on a blockchain. In Section 7 we argue that using data from the blockchain for seeding a CA is proven effective, especially when such data provide publicly verifiable proofs of the functions' correctness.

Figure 1 illustrates different patterns that are observed by the evolution of the implemented CA that assumes `11000000010010010 01` as the initial seed, the same number of evolution steps, but a different rule in each case (see (a) – (h) from Figure 1). Successive lines from each time-space diagram indicate the configuration obtained on each evolution step. It is observed that the simple structure of a CA can result to various complex behaviours (i.e., results) that are difficult to be predicted in detail, even in cases of assuming that the initial seed could be compromised by an attacker. Random

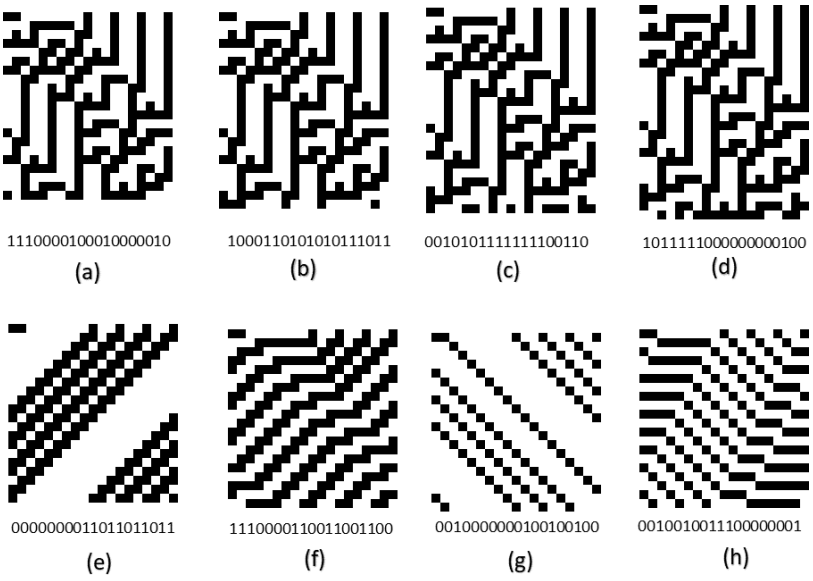

FIGURE 1
Time-space diagrams generated by the implemented 1-d CA, two states are represented; white squares represent state 0, and black-squares represent state 1. Evolution of states for each time step increases downwards.

number sequences produced by the CA exhibit an inherent complexity that is introduced by the configuration state that can yield to extremely complicated transformations even with the simple case configurations and a deterministic number of evolution iterations.

4 BLOCKCHAIN: VERIFIABLE DATA

By its definition the outcome of a pseudo-random process is not *verifiable* unless the seed or information related to the pseudo-random function is made publicly available. But even if the seed is made publicly available to a verifier there is no guarantee that the actual data were not manipulated to serve a specific purpose beforehand. On another node, the attacker can even manipulate the seed generation process; therefore the trustworthiness of the entire process is challenged. In this paper, we argue that the inherent unpredictability of a blockchain protocol produces data that can be leveraged as sources of high entropy to inform a pseudo-random generation process. At the same time, the immutability of the recorded data introduces another layer of trustworthiness to the process. The data are recorded and made publicly available therefore

any external oracle or autonomous actor can judge, but unlikely to predict, the outcome of the process.

5 POPULATION OF RNGBLOCKS

Random processes should provide uniformity in the outcomes with sequences ensuring unpredictability, without revealing any obvious patterns from the process. To ensure a high level of randomness, algorithms for pseudo-random generation utilize a variety of seeds with high entropy, thus aiming to inherit the properties of natural stochastic processes. To meet the requirement of a high degree of entropy the proposed pseudo-random generation algorithm leverages publicly verifiable information from the evolution of an existing distributed ledger[¶] (e.g., Ethereum). In more detail, as the distributed ledger evolves over time, data extracted from each *block* (verified by a consensus algorithm) are utilized as seeds to the generation of a *population* of `RngBlocks`, denoted by $\mathcal{RB}$.

An `RngBlock`, $V_{t_i} \in \mathcal{RB}$, where $i = 1, \ldots, n$, is a sequence of `bits` of length k, with t_i denoting the *timestamp* based on when the `RngBlock` is generated. Each `RngBlock` is instantiated with a set of statistically independent evidence $E = \{e_1, e_2, \ldots, e_n\}$ from the blockchain used. Note that the configuration of the `RngBlocks` is dynamic and any kind of evidence can be utilized, as long as this evidence can be represented in a binary form.

Considering the evolution of the blockchain as a continuous sequence of blocks, we align the population of `RngBlocks` in a bounded time window $[L, A]$ of a block stream S consisting of a set of $(V_{t_i}, block.hash_i)$ pairs: $S = \{(V_{t_0}, block.hash_0), (V_{t_1}, block.hash_1), \ldots (V_{t_n}, block.hash_n)\}$, where $block.hash$ is the result of the hash function that uniquely identifies each block[§] . During each successive block, as confirmed by the hash function, a new `RngBlock` is generated as depicted in Figure 2.

The generation of the final *genesis* `RngBlock` pair $(V_{t_n}, block.hash_n)$ is triggered by an external process. This external process encapsulates the logic and data with regards to a specific game-of-chance (e.g., a draw for a prize competition). For the purposes of this work, we envision that the execution logic for the random generator is encapsulated in a *smart-contract*, $\mathcal{C}_{RNG}$, annotated with a deployment `Date` that marks the creation point in time i.e., L in the bounded window. The contract $\mathcal{C}_{RNG}$ self-executes on a specific point in time (assuming that the conditions of the smart-contract are met) that bounds the generation of $(V_{t_n}, block.hash_n)$ on the blockchain time-window to A (as shown in Figure 2).

[¶] the proposed algorithm is generic and not limited to the `Ethereum` blockchain.
[§] for `Bitcoin` this is the result of the *proof-of-work* algorithm.

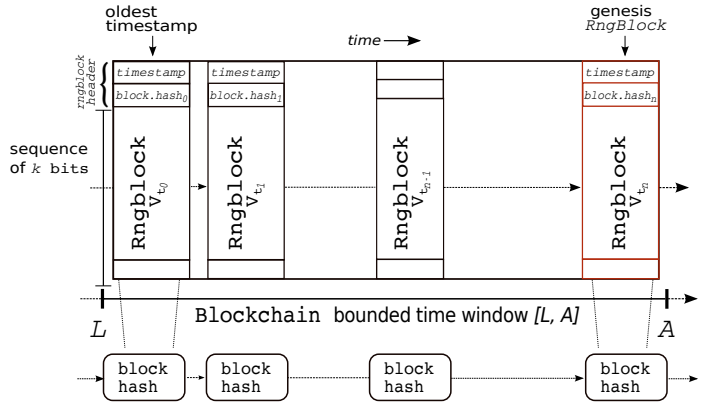

FIGURE 2
Successive generation of `RngBlocks` in alignment to the evolution of a *blockchain* and bounded within a specific time window.

6 THE PROPOSED CELLULAR AUTOMATA BLOCKCHAIN ALGORITHM

The genesis `RngBlock` individual, $(V_{t_n}, block.hash_n)$, is selected as the *initial seed* used later to inform the evolution of a `1-d` CA. The proposed CA consists of a Boolean cellular array (grid) for which the cellular state is $s \in \{0, 1\}$. In more detail, the configuration of the proposed CA in conjunction with the earlier description of `1-d` CA is as follows: the individual $(V_{t_n}, block.hash_n)$ is used as the *initial-seed* of the CA, the *length* of the CA cellular array is set to k which is the same as the `RngBlock` length, and the *transition-rule* is specified in the form of a `3-bit` rule-table with each entry capturing every possible neighborhood state configuration i.e., $2^3 = 8$. To complete the configuration of the proposed CA the *time of execution* set to evolve the initial seed of the CA is set to a finite number of steps and a periodic boundary condition is used.

At each step of the evolution the proposed methodology instantiates a CA using the genesis `RngBlock` individual as the *initial seed* of the CA. The selection of the transition rule that contributes to the next evolution state of the CA is dynamic and is set according to the *milliseconds* (`ms`) derived from the execution `Date` of the smart contract $\mathcal{C}_{RNG}$. This value is normalized in the domain $[0 - 255]$ which represents the total number of rules that can be involved at each evolution step. The normalized value derived by $y = (255 \times ms)/1000$ is rounded-down to the nearest integer and then translated to a binary representation and used to enable the rules that are participating to the CA evolution.

To determine the time of execution (i.e., evolution steps) for the CA we define the variable q. As an example, and to ensure a high degree of entropy, q is instantiated with the *blockhash* value of some block determined as follows `bhash = block.blockhash(block.number-offset)`, where `block.number` is the height of the current block, and `offset` is a normalized value bounded in the domain $[0-255]$[||] . The last z-digits from the hexadecimal value returned by the `blockhash()` are casted to a decimal form that determines the number of evolution steps (i.e., the value of q).

When the evolution of the CA reaches a finite number of steps as of q, a binary number of length k is produced as the final outcome of the proposed pseudo-random generator process.

7 EXPERIMENTAL DETAILS

For experimentation purposes the methodology was instantiated as follows. Each `RngBlock` individual is a sequence of bits of length k = 19 consisting from data retrieved from the last confirmed block i.e., `block.number - 1` assuming the Ethereum blockchain. In particular, the cellular array is filled with binary data as follows[#] :

- index [0-4] from the `uint(block.difficulty);`
- index [5-10] from the height of the block i.e., `block.number`;
- index [11-15] from `block.nonce`; and
- index [16-18] from the size of the transactions array from the block `block.transactions`.

Additionally, we relate the `offset` variable, required to determine the value of q, to Ethereum's price fluctuations obtained dynamically with the help of `Coinmarketcap.com` API.

8 EXPERIMENTAL RESULTS

Truly random sequences can only be obtained from stochastic physical phenomena thus random numbers generated algorithmically are classified as pseudo-random. Based on this inevitable assertion the aim is to inject high

[||] assuming the Ethereum blockchain, one can only access the hashes of the most recent 256 values, other values are set to zero.

[#] we assume that the data from each block are casted to a binary representation

Test's name	p-value	result
diehard birthdays	0.93481163	PASSED
diehard operm5	0.77251059	PASSED
diehard rank 32x32	0.47198340	PASSED
diehard rank 6x8	0.16963500	PASSED
diehard bitstream	0.63628175	PASSED
diehard opso	0.86983038	PASSED
diehard oqso	0.94570945	PASSED
diehard dna	0.45052643	PASSED
diehard count 1s str	0.94823989	PASSED
diehard count 1s byt	0.25156061	PASSED
diehard parking lot	0.13186651	PASSED
diehard 2dsphere	0.96572359	PASSED
diehard 3dsphere	0.40422396	PASSED
diehard squeeze	0.47236249	PASSED
diehard sums	0.22757837	PASSED
diehard runs	0.28411048	PASSED
diehard runs	0.56608951	PASSED
diehard craps	0.24641008	PASSED
diehard craps	0.73056728	PASSED
marsaglia tsang gcd	0.88529256	PASSED
marsaglia tsang gcd	0.21698468	PASSED
sts runs	0.28820268	PASSED
sts serial	0.06370033	PASSED
rgb bitdist	0.05300485	PASSED
rgb minimum distance	0.19279478	PASSED
rgb permutations	0.42801173	PASSED
rgb lagged sum	0.06228273	PASSED
rgb kstest test	0.23739296	PASSED
dab bytedistrib	0.91016351	PASSED
dab dct	0.64809168	PASSED
dab filltree	0.27301099	PASSED
dab filltree2	0.09113271	PASSED
dab monobit2	0.69835025	PASSED

TABLE 1
Result of Dieharder 3.31.1 test suite on the extracted random numbers. In the case of multiple tests in a category, the smallest have been reported.

sources of entropy in the process that enable the approach to behave as stochastic as possible. A series of well-known statistical (empirical) test suites are utilized to assert that the random sequences of numbers produced have a degree of unpredictability. Thus, generated numbers were tested using *Dieharder* and *NIST* test suites [3, 29]. The findings of our experimentation, as shown in Tables 1 and 2, confirm our initial intuition and present adequate

Test's name	p-value	result
Frequency	0.350485	PASSED
BlockFrequency	0.213309	PASSED
CumulativeSums	0.534146	PASSED
Runs	0.008879	PASSED
LongestRun	0.000199	PASSED
Rank	0.739918	PASSED
FFT	0.911413	PASSED
NonOverlappingTemplate	0.004301	PASSED
OverlappingTemplate	0.739918	PASSED
LinearComplexity	0.534146	PASSED

TABLE 2
Result of NIST SP 800-22 test suite on the extracted random numbers. In the case of multiple tests in a category, the smallest have been reported.

evidence that the random sequences generated by the proposed technique are predicted to by random enough. Although passing the statistical tests does not always guarantee randomness, however it can be concluded that no obvious patterns are present in the analyzed data. Furthermore, it is shown that data from the evolution of a blockchain protocol can be used as sources of sufficient entropy to inform the evolution of CA for producing sequences of pseudo-random numbers.

9 CONCLUSIONS

The integration of fair random numbers can enable a wide spectrum of distributed ledger applications. Such applications e.g., gambling, often rely on external third party processes that are often difficult to audit. Public blockchains have the potential to leverage transparency and to enable public judgments to random processes. This paper makes the case that a process for generating pseudo-random numbers from using blockchain data should, at least, exhibit the following properties: (i) provide enough data that enable self-verification of the process; (ii) auditing the result is open to the general public; (iii) independent from any third-party; and (iv) capable of enough uncertainty that leads to diverse and complex behaviour that models nature random processes. According to the above minimum requirements, this paper presents a methodology that builds on CA to associate high sources of entropy including data from public blockchain, and other uncertainty processes for the generation of verifiable pseudo-random numbers. At the same time, with the use of smart contracts `OP_RETURN` [2], evidence used for audit and self-verification is appended on an immutable public ledger

and exposed to the general public. This process enables any external oracle to cross-validate the fairness of the outcome. Applying an extensive set of statistical testing suites we demonstrate significant evidence of the effectiveness of the methodology as a pseudo-random generation process with a high degree of entropy. In addition, we suggest that the implemented solution could enable alternative consensus schemes and signature mechanisms. As a continuation of this work we will be observing the effects of adding chaos-theory [33] to our verifiable random numbers generation process.

REFERENCES

[1] BitcoinMegaLottery, (2014). Bitcoin mega lottery - provably fair. https://www.bitcoinmegalottery.com. Accessed: 2018-10-23.

[2] Joseph Bonneau, Jeremy Clark, and Steven Goldfeder. (2015). On bitcoin as a public randomness source. *IACR Cryptology ePrint Archive*, 2015:1015.

[3] Robert G. Brown. (2019). Dieharder: A random number test suite. https://webhome.phy.duke.edu/ rgb/General/dieharder.php.

[4] Savvas A. Chatzichristofis, Oge Marques, Mathias Lux, and Yiannis Boutalis. (2012). Image encryption using the recursive attributes of the exclusive-or filter on cellular automata. In Georgios Ch. Sirakoulis and Stefania Bandini, editors, *Cellular Automata*, pages 340–350, Berlin, Heidelberg. Springer Berlin Heidelberg.

[5] Savvas A. Chatzichristofis, Dimitris A. Mitzias, Georgios Ch. Sirakoulis, and Yiannis S. Boutalis. (2010). A novel cellular automata based technique for visual multimedia content encryption. *Optics Communications*, 283(21):4250–4260.

[6] Jeremy Clark and Urs Hengartner. (2010). On the use of financial data as a random beacon. In *Proceedings of the 2010 International Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'10, pages 1–8, Berkeley, CA, USA. USENIX Association.

[7] Michael Crosby, Pradan Pattanayak, Sanjeev Verma, and Vignesh Kalyanaraman. (2016). Blockchain technology: Beyond bitcoin. *Applied Innovation*, 2:6–10.

[8] Sourav Das and Dipanwita Roy Chowdhury. (2011). Cryptographically suitable maximum length cellular automata. *J. Cellular Automata*, 6(6):439–459.

[9] N. I. Dourvas, G. C. Sirakoulis, and A. Adamatzky. (June 2017). Cellular Automaton Belousov-Zhabotinsky Model for Binary Full Adder. *International Journal of Bifurcation and Chaos*, 27:1750089–478.

[10] Michael J. Fischer, Michaela Iorga, and René Peralta. (2011). A public randomness service. In *Security and Cryptography (SECRYPT), 2011 Proceedings of the International Conference on*, pages 434–438. IEEE.

[11] Ioakeim G. Georgoudas, Georgios Ch. Sirakoulis, Emmanouil M. Scordilis, and Ioannis Andreadis. (2007). A cellular automaton simulation tool for modelling seismicity in the region of xanthi. *Environmental Modelling & Software*, 22(10):1455–1464.

[12] Stéphane Grumbach and Robert Riemann. (2017). Distributed random process for a large-scale peer-to-peer lottery. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 34–48. Springer.

[13] Peter D. Hortensius, Robert D. McLeod, and Howard C. Card. (Oct 1989). Parallel random number generation for vlsi systems using cellular automata. *IEEE Transactions on Computers*, 38(10):1466–1473.

[14] NIST Randomness Beacon. (2019)https://www.nist.gov/programs-projects/nist-randomness-beacon. [Online; accessed 12-June-2019]

[15] Athanasios Ch. Kapoutsis, Savvas A. Chatzichristofis, Georgios Ch. Sirakoulis, Lefteris Doitsidis, and Elias B. Kosmatopoulos. (2015). Employing cellular automata for shaping accurate morphology maps using scattered data from robotics missions. In *Robots and Lattice Automata*, pages 229–246. Springer.

[16] Ioannis Karafyllidis, Ioannis Andreadis, Philippos Tsalides, and Adonios Thanailakis. (1998). Non-linear hybrid cellular automata as pseudorandom pattern generators for vlsi systems. *VLSI Design*, 7(2):177–189.

[17] Rafailia-Eleni Karamani, Vasileios Ntinas, Ioannis Vourkas, and Georgios Ch. Sirakoulis. (Sep. 2017). 1-D memristor-based cellular automaton for pseudo-random number generation. In *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 1–6.

[18] Sandip Karmakar, Debdeep Mukhopadhyay, and Dipanwita Roy Chowdhury. (2012). Cavium - strengthening trivium stream cipher using cellular automata. *J. Cellular Automata*, 7(2):179–197.

[19] Sunny King and Scott Nadal. (2012). Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19.

[20] Ioannis Kokolakis, Ioannis Andreadis, and Philippos Tsalides. (1997). Comparison between cellular automata and linear feedback shift registers based pseudo-random number generators. *Microprocessors and Microsystems*, 20(10):643–658.

[21] Leonidas Kotoulas, Dimitrios Tsarouchis, Georgios Ch. Sirakoulis, and Ioannis Andreadis. (May 2006). 1-d cellular automaton for pseudorandom number generation and its reconfigurable hardware implementation. In *2006 IEEE International Symposium on Circuits and Systems*, pages 4.

[22] Leslie Lamport, Robert Shostak, and Marshall Pease. (July 1982). The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401.

[23] Mitar Milutinovic, Warren He, Howard Wu, and Maxinder Kanwal. (2016). Proof of luck: An efficient blockchain consensus protocol. In *Proceedings of the 1st Workshop on System Software for Trusted Execution*, page 2. ACM.

[24] Satoshi Nakamoto, (2008). Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf. [Online].

[25] Sukumar Nandi, Biswajit K. Kar, and Parimal Pal Chaudhuri. (Dec 1994). Theory and applications of cellular automata in cryptography. *IEEE Transactions on Computers*, 43(12):1346–1357.

[26] Vasileios G. Ntinas, Byron E. Moutafis, Giuseppe A. Trunfio, and Georgios Ch. Sirakoulis. (2017). Parallel fuzzy cellular automata for data-driven simulation of wildfire spreading. *Journal of Computational Science*, 21:469–485.

[27] oraclize.it, (2018). A scalable architecture for on-demand, untrusted delivery of entropy. http://www.oraclize.it/papers/random˙datasource-rev1.pdf. Accessed: 2018-10-23.

[28] Lucian Petrica. (2018). FPGA optimized cellular automaton random number generator. *Journal of Parallel and Distributed Computing*, 111:251–259.

[29] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray,

and San Vo, (2010). A statistical test suite for random and pseudorandom number generators for cryptographic applications. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf.

[30] Franciszek Seredynski, Pascal Bouvry, and Albert Y. Zomaya. (2004). Cellular automata computations and secret key cryptography. *Parallel Computing*, 30(5):753 – 766. Parallel and nature-inspired computational paradigms and applications.

[31] Georgios Ch. Sirakoulis. (2016). Parallel application of hybrid DNA cellular automata for pseudorandom number generation. *J. Cellular Automata*, 11(1):63–89.

[32] Georgios Ch. Sirakoulis. (2018). Cellular automata hardware implementation. In Andrew Adamatzky, editor, *Cellular Automata: A Volume in the Encyclopedia of Complexity and Systems Science, Second Edition*, pages 555–582. Springer US, New York, NY.

[33] Toni Stojanovski and Ljupčo Kocarev. (March 2001). Chaos-based random number generators-part i: analysis [cryptography]. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 48(3):281–288.

[34] Miroslaw Szaban and Franciszek Seredynski. (2018). Designing conflict free cellular automata-based PRNG. *Journal of Cellular Automata*, 13(3):229–246.

[35] Marco Tomassini and Mathieu Perrenoud. (2001). Cryptography with cellular automata. *Applied Soft Computing*, 1(2):151–160.

[36] Marco Tomassini, Moshe Sipper, Mos Zolla, and Mathieu Perrenoud. (1999). Generating high-quality random numbers in parallel by cellular automata. *Future Generation Computer Systems*, 16(2):291–305.

[37] Dimitrios Tourtounis, Nikolaos Mitianoudis, and Georgios Ch. Sirakoulis. (2018). Salt-n-pepper noise filtering using cellular automata. *J. Cellular Automata*, 13(1-2):81–101.

[38] Anastasios Tsiftsis, Georgios Ch. Sirakoulis, and John N. Lygouras. (2017). FPGA processor with GPS for modelling railway traffic flow. *J. Cellular Automata*, 12(5):381–400.

[39] Michail-Antisthenis I. Tsompanas, Richard Mayne, Georgios Ch. Sirakoulis, and Andrew I. Adamatzky. (2015). A cellular automata bioinspired algorithm designing data trees in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 11(6):471045.

[40] S. Wolfram. (1994). *Cellular automata and complexity: collected papers*. Addison-Wesley Pub. Co.

[41] Stephen Wolfram. (Jul 1983). Statistical mechanics of cellular automata. *Rev. Mod. Phys.*, 55:601–644.

[42] Stephen Wolfram. (1986). Cryptography with cellular automata. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 429–432, Berlin, Heidelberg. Springer Berlin Heidelberg.

[43] Stephen Wolfram. (1986). Random sequence generation by cellular automata. *Advances in applied mathematics*, 7(2):123–169.

[44] Gavin Wood. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32.

[45] Ya-Hui Ye, Jia Lee, and Li-Wei Zhu. (2018). A game-of-life-based paradigm for massively parallel computing on asynchronous circuits. *J. Cellular Automata*, 13(4):287–305.