Kỷ yếu Hội nghị Quốc gia lần thứ X về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR), Đà Nẵng, ngày 17-18/08/2017
DOI: 10.15625/vap.2017.00023

FLEXIBLE PROOF-OF-WORK: A DECENTRALIZED UNPREDICTABLE NUMBER GENERATOR

Trần Anh Dũng[1], Nguyễn Đức Đình Nghĩa[1], Đặng Minh Tuấn[2], Nguyễn An Khương[3], Huỳnh Tường Nguyên[3]

[1]Infinity Blockchain Labs Co., LTD; [2]Vietkey Group; [3]Trường ĐH Bách Khoa, Tp. Hồ Chí Minh

{dungta, nghiandd}@blockchainlabs.asia; dangtuan@vietkey.vn; {nakhuong, htnguyen}@hcmut.edu.vn

ABSTRACT: *We propose a new proof-of-work protocol called "Flexible Proof-of-Work" to generate an unpredictable set of numbers. This can be seen as a component of a general framework for a decentralized random number generator. The algorithm ensures experimentally that the result is feasible and unpredictable. Moreover we demonstrate that the algorithm can be easily implemented on Ethereum blockchain system by smart contracts, and miners who have limited resource are still able to compute solutions. Meanwhile during the process, the problems are changing continuously by taking new solutions into account, making an adversary hard to attack the protocol by pre-calculating the final results.*

Keyworks: *Unpredictable number generator, decentralized system, blockchain, proof-of-work.*

I. INTRODUCTION

Random numbers have important roles in many applications, especially in lottery, statistical sampling, computer simulation, cryptography. Proving the true randomness is always significantly challenging that need to be overcome, and contains pitfalls that need to be avoided. Nowadays, with the advantages of blockchain technology, that problem can be solve by means of a decentralized system. Our ultimate is to construct a decentralized algorithm for generating random numbers (dRNG) openly and transparently, but still secure. We propose that this general dRNG framework should consists of (i) a proof-of-work (PoW) protocol for generating solution set; (ii) the guidelines in selecting the appropriate solutions; and (iii) an protocol resists against both fraud and strategically withholding secret values (selfish mining). Such a general framework has to be ensured that attempts at manipulating and predicting the result were unsuccessful. In this paper we only focus on the first module for generating solution set, called *Flexible Proof-of-Work*.

II. RELATED WORKS

Random numbers can be generated by a random bit generator which can be defined as a device or algorithm whose output is a sequence of statistically independent and unbiased binary digits [15]. Generators that produce random sequences (RNGs) can be classified into two types: Pseudo Random Number Generators (PRNGs) by using mathematical algorithms (deterministic); and True Random Number Generators (TRNGs) by using physical process (non-deterministic).

In practice, one might use the NIST's Beacon[1] as a source of public randomness. This Beacon ensures the unpredictability, autonomy, and consistency. "Unpredictability means that users cannot algorithmically predict bits before they are made available by the source. Autonomy means that the source is resistant to attempts by outside parties to alter the distribution of the random bits. Consistency means that a set of users can access the source in such a way that they are confident they all receive the same random string."

1. Pseudo Random Number Generators

PRNG uses deterministic digital process by a digital algorithm via softwares. These RNGs are based on algorithms implemented on finite-state machines to produce pseudo-random determinism sequences from initial values called seeds in mathematical processes.

PRNG are much more cost effective and thousands of times faster than hardware-based RNG. The PRNG should achieve excellent statistical properties, fast execution time, repeatability, reproducibility, and its security must be based on the difficulty to solve the related mathematical problem. However, because the output is a function of the seed state, the actual entropy of the output can never exceed the entropy of the seed. Hence, the randomness level of the pseudo-random numbers depends on the level of randomness of the seed. And the pseudo-random number has a deterministic sequence, which may be anticipated by a hacker if an initial condition of the pseudo-random number system is revealed.

[1] https://www.nist.gov/programs-projects/nist-randomness-beacon

2. True Random Number Generators

TRNG uses hardwares or non-deterministic physical nature processes such as quantum random processes, thermal noise of a resistor, short noise of a $(p-n)$ junction of a semiconductor, photon noise, atmospheric noise, free-running oscillators, frequency jitter in oscillator, and chaotic laser [20, 21] to ensure the randomness.

Because the physical nature cannot be predicted, the physical random number is more appropriate for protecting private information. Furthermore, they produce continuous time analog signals which are often called noise. However, noise intensity of the underlying physical phenomenon on which the random number is based is typically small, and thus a high voltage is required to convert the small noise to a random number. These are also called hardware-based random number generators because of the use of the randomness aspect in the hardware. They could be sampled by digitization, and post-processing techniques can be implemented to improve the randomness. TRNGs should be unpredictable, unreproducible, and statistically unbiased.

Among traditional techniques have been used for generating truly random numbers are the Integrated Circuit (IC) RNG designs [19]. In this direction, there are several approaches, i.e., by amplification of a noise source [2, 12], by jittered oscillator sampling [9, 11, 13, 19], by discrete-time chaotic maps [3, 6, 7, 8, 25, 28], or by continuous-time chaotic oscillators [20, 24]. In spite of the fact that, the use of discrete-time chaotic maps in the realization of RNG is well-known for some time, it was shown recently that continuous-time chaotic oscillators can also be used to realize a TRNG.

In comparison with RNGs based on discrete-time chaotic maps, amplification of a noise source and jittered oscillator sampling, it is seen that RNGs based on continuous-time chaotic oscillators can offer much higher and constant data rates without post-processing and generate of high fractal dimension chaos with no substantial increase on complexity.

3. Decentralized Random Number Generators

Although numerous algorithms for RNG have been developed, most of them are pseudo-random number generator (PRGN). Recently, using the leverage of blockchain technology, some random number generator (RGN) algorithms have been developed on decentralized systems (dRNGs) see [27], or based on public randomness [15]. In our opinion, the dRNGs can be seen as a subclass of the PRGNs, but with significantly high quality of the randomness due to the pools where entropy were collected, i.e., blockchain's entropy.

Most recently, Popov [23] propose an algorithm permitting a large group of individuals to reach consensus on a random number, without having to rely on any third parties. The algorithm works with high probability if there are less than 50% colluding parties in the group. The proposed algorithm has some limitations due to its assumptions that there is one group has to have at least one honest participant, and no group consist entirely of colluding parties.

In [1] the authors used Bitcoin as a public randomness source to propose two protocols generating an unpredictable beacon and constructing arbitrary multi-party computation protocols. And in [5], the author proposed a beacon from Bitcoin blockchain which requires no trusted parties. They used the advantages of Bitcoin blockchain to directly compute the financial cost of attempting to manipulate the beacon output.

However, in [20], the authors presented negative results with regard to Bitcoin-based randomness extraction by showing how an adversary could manipulate these random numbers, even with limited computational power and financial budget. Furthermore, in [4] the authors proved that no protocol can achieve an arbitrarily small bias when the adversary has an infinite budget, but they positively proposed beacon protocols that defeat a budget-restricted adversary.

As our main contribution in this work, we propose a decentralized protocol generating unpredictable sets of solutions such that no playing party will be able to predict or manipulate the results.

III. PROPOSED APPROACH

1. Problem Statement and Assumptions

Our **RNG problem** is that, for a given number with a predictable risk, a blockchain based system - the **INPUT**, we have to construct a transparent and non-deterministic process which can produce an unpredictable set of numbers (i.e., it cannot predict all these numbers)-the **OUTPUT**. In this problem, we have to make two following assumptions.

- **Assumption 1**. Blockhash (BlockId) is unpredictable;
- **Assumption 2**. Time period for creating a given number of blocks from a given block in a blockchain-based system is unpredictable.

The Assumption 1 is naturally reasonable because it is widely accepted that suitable hashing functions, i.e. Keccak, can be used as PRNGs, and Bitcoin can also be used as a randomness source. Therefore we can treat block-headers as random sources and unpredictable, and combining with hashing function might assume that *BlockIds* are unpredictable.

2. General idea of the algorithm

From a given **seed** (i.e., Blockhash, BlockID) x_0, we determine a finite set of solutions. This **solution set** will be used to determine the seed for the next iteration. More precisely, we have the following process

$$x_0 \to seed_1 \to \{x_{11}, x_{12}, \ldots.\} \to seed_2 \to \{x_{21}, x_{22}, \ldots\} \to seed_3 \to \{x_{31}, x_{32}, \ldots\} \ldots$$

We define the $seed_i$'s ($i \geq 1$) as follows. Suppose in this stage that in the $(i\text{-}1)^{\text{th}}$ round, the secret values $x_{(i-1)j}$ (solutions) have been computed successfully by some function **Find(.)** (to be specified later).

Then let

$$v_{(i-1)j} = H_1(x_{(i-1)j}), j = 1, 2, \ldots$$

be the **digest** of $x_{(i-1)j}$ by a suitable hashing function H_1 (i.e., the well-known SHA-3, Keccak hashing function,...).

We then compute

$$v_{i-1} = v_{(i-1)1} \oplus v_{(i-1)2} \oplus v_{(i-1)3} \oplus \ldots$$

We propose the (common) **fingerprint** F_i of the $x_{(i-1)j}$'s as follow

$$F_i := [(v_{i-1} >> \beta) \oplus v_{i-1}] \% 2^{\beta},$$

where "$>> \beta$" is the β-bit right shift operator, and "$\oplus$" is the Exclusive OR (XOR) operator, and initially, we take $v_0 = H_1(x_0)$.

We now define

$$seed_i := F_i, (i = 1, 2, \ldots).$$

Overall, the values are determined in the following order

$$\begin{aligned} & x_0 \to v_0 \\ \to F_1 \to \{x_{11}, x_{12}, \ldots\} \to \{v_{11}, v_{12}, \ldots\} \to v_1 \\ \to F_2 \to \{x_{21}, x_{22}, \ldots\} \to \{v_{21}, v_{22}, \ldots\} \to v_2 \\ \ldots \end{aligned}$$

Remark. In practice, the parameter β has to be specified, depending on the word-size of the computational system. In our experiment (see Section 4), we consider $\beta = 128$.

3. Description of the Algorithm

In the proposed algorithm, the key point is that all playing parties have to choose themselves suitable strategies **Find(.)** in order to find random numbers x_{ij} in the i^{th} round such that their digests have the same number $n_{ij} = e_i$ (for given e_i) of bits 1 with those in the fingerprints of solutions found in the previous round.

More precisely, let e_i be the **difficulty** of the problem at the i^{th} round. This number will be adjusted increasingly by **Era timeout**

$$e_i = \textbf{Adjust}(e_{i-1}): \text{Difficulty increased in each Era,}$$

and in the Section 4, for the experiment, we currently increase e_i by one unit per Era. Let $e_0 = \alpha$, a given difficulty level to be specified later. Then the miners on the blockchain have to find x_{ij} such that if

$$v_{ij} = H_1(x_{ij}), m_{ij} = H_2(F_i \| v_{ij}), \text{ and } n_{ij} = \textbf{NoIB}(m_{ij}),$$

then we must have

$$n_{ij} = e_i, \text{ and } v_{ij} \notin V,$$

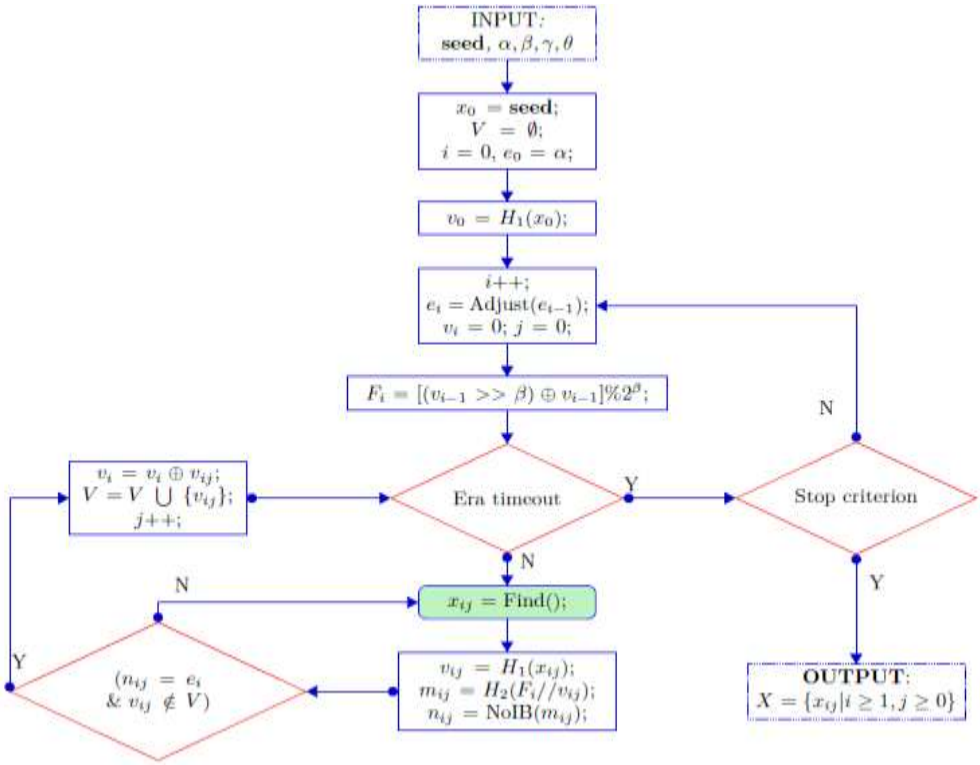
where V is the set of digests of solutions up to the previous round, and

- H_2 is also a suitable hash function (i.e., the well-known SHA-3, Keccak hashing function, ...)
- **NoIB**(m_{ij}): Number of identical bits between Fi and mij

In summary, let

- **Era timeout**: be the product of θ and *blocktimeout*, and

• **Stop criterion**: be the case that *Era timeout* is passed or we couldn't found any solution in this Era, meaning that $X_i = \emptyset$, then we have the following algorithm (see Algorithm 1.)



Algorithm 1. Flexible Proof-of-Work.

Remark. 1. It is clear that this process is non-deterministic, therefore the generated **OUTPUT** is truly random, since the **seed** is an unpredictable number (according Assumption 1), and the **Era timeout** (θ) guaranteeing the variation of number of solutions found in a given iteration (era) is also unpredictable (according Assumption 2).

2. Although function **Find(.)** can be implemented by brute-force algorithm, in practice it should use a randomized algorithm (such as OpenSSL) in order to get more trustworthy on unpredictability.

3. We can easily see that the correctness of the proposed algorithm is ensured, since the probability of successfully finding out solutions is nonzero and small enough for suitable parameters in the input. Indeed, the probability for a randomly chosen string x having e bits in common with a given $\beta-$ bit string is

$$p = \frac{\binom{\beta}{e}}{2^{\beta}}. \quad (1)$$

IV. EVALUATION

In this section we conduct an experiment to evaluate our proposed algorithm on Ethereum block chain system. In this experiment, we consider the following setting.

- **Find(.)**: The strategy to find 256-bit numbers,
- $\beta = 128$,

With $\beta = 128$, the plot of probability distribution given in the equation (1) versus e is given below, see Figure 1.

This plot suggests that in practice, for $\beta = 128$, the difficulty e should be set from $e_0 = \alpha = 64$, and each e_i should be adjusted increasingly one unit by **Era timeout**. For five difficulties

$$e \in \{69, 74, 79, 84, 89\}$$

with the further settings that the ***BlockId*** is at the block 3000100:

3b922104b70c94b592659573a65c60fe2f225e395a542e48de5dc5a8269fe94f

on Ethereum block chain[2], Era time out $\theta = 2 \times 10^6$ ns, we obtain the following plot[3] showing the running time in number of solutions found, see Figure. 2.

Thus our experimental result ensures the feasibility and the correctness of the proposed algorithm in Subsection 3.3 since it demonstrates a suitable range of difficulty degree e_i such that it is possible for finding solution(s), and moreover it also indicates a suitable value θ such that solutions have been found are hard to predict.

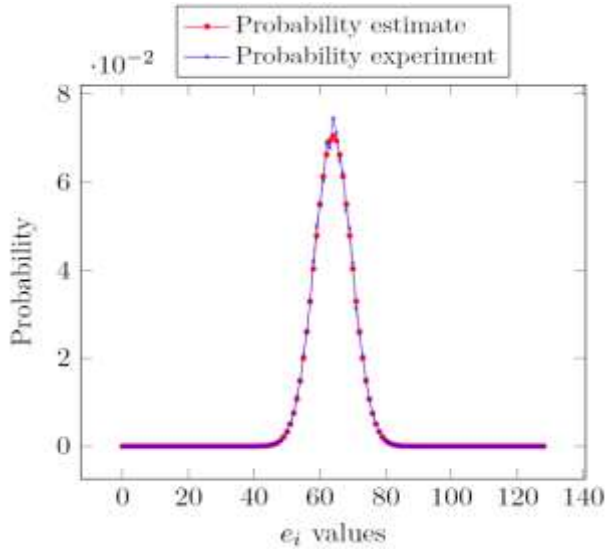


Figure 1. Probability for finding out solution x against difficulty e.

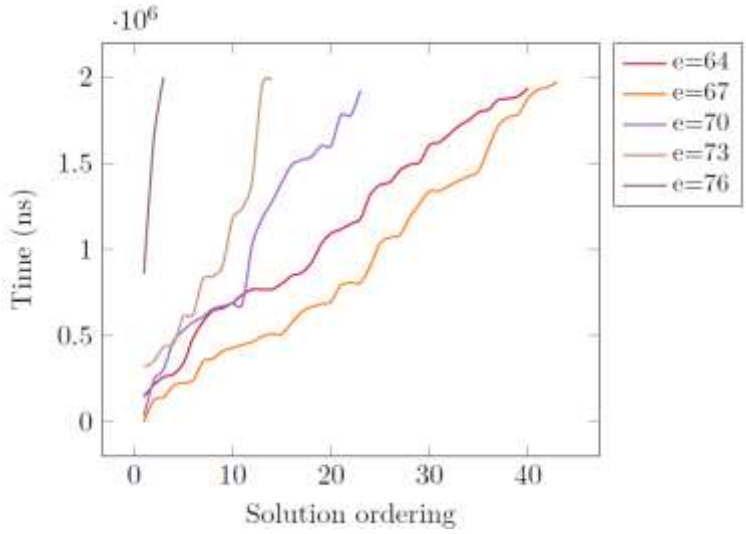


Figure 2. Running times against number of solutions

V. CONCLUSION AND FUTURE WORKS

We've proposed a so-called Flexible Proof-of-Work protocol for generating unpredictable sets of random numbers. Our experiment shows the feasibility of the algorithm and the unpredictability of the result. Moreover the experiment also shows that a party with limited computational power is still able to find out the unpredictable solution set. Meanwhile an adversary cannot neither predict the results due to the randomness of the input parameters, especially the **seed**, nor manipulate the results as the solution set constantly change during the process.

However the proposed protocol is still facing some limitations due to selfish miner's strategically withholding the solutions, or a miner can use his dominated computational power to perform an 101-Attack. Our future works are to formulate a general framework that overcome these limitations.

[2] https://etherscan.io/block/3000100
[3] We refer to https://github.com/DecentralizedRNG/resources for the complete experimental data.

ACKNOWLEDGMENTS

The work is financially supported by Infinity Blockchain Labs Co., LTD (http://www.blockchainlabs.asia).

REFERENCES

[1] Andrychowicz, M., and Dziembowski, S. "Distributed cryptography based on the proofs of work". In Crypto, 2015, http://eprint.iacr.org/2014/796.

[2] Bagini, V., and Bucci, M. "A design of reliable true random number generator for cryptographic applications." *In Proceedings of the workshop cryptographic hardware and embedded systems (CHES' 99)* (1999): 204-218.

[3] Bean, J., and Langlois, P.J. "A current mode analog circuit for tent maps using piecewise linear functions." *In IEEE ISCAS* 53(2) (1994): 125-128.

[4] Bentov, I., Gabizon, A., and Zuckerman D. "Bitcoin beacon." arXiv preprint arXiv: 1605.04559 (2016).

[5] Bonneau, J., Clark, J. and Goldfeder, S. "On Bitcoin as a public randomness source", preprint 2015, https://eprint.iacr.org/2015/1015.

[6] Callegari, S., Rovatti, R., and Setti, G. "Embeddable ADC-based true random number generator for cryptographic applications exploiting nonlinear signal processing and chaos." *IEEE Transaction on Signal Processing* 53(2) (2005): 793-805.

[7] Callegari, S, Cesaroni, A, and Setti, G. "Mixed mode unpredictable binary information source exploiting complex dynamics and adaptive median thresholding." *In Proceedings of the ninth workshop on nonlinear dynamics of electronic systems (NDES 2001)* (2001): 65-8.

[8] Delgado-Restituto, M, Medeiro, F, Rodriguez-Vazquez, A. "Nonlinear switched-current CMOS IC for random signal generation." *Electron Lett*, 29(25), (1993), 2190-1.

[9] Dichtl, M., and Janssen, N. "A high quality physical random number generator." *In Proceedings of the Sophia Antipolis forum microelectronics (SAME 2000)* (2000): 48-53.

[10] Dwork, C., Lynch, N., and Stockmeyer, L. "Consensus in the presence of partial synchrony." *Journal of the ACM (JACM)* 35(2) (1988): 288-323.

[11] Fairfield, R. C., Mortenson, R. L., and Coulthart, K. B. "An LSI random number generator (RNG)." *Advances in Cryptography: Proc. of Crypto 84, (Springer-Verlag)* (1984): 203-230.

[12] Holman, W. T., Connelly, J. A., Downlatabadi, A. B. "An integrated analog/digital random noise source." *IEEE Trans Circuits and Systems I* 44(6) (1997): 521-528.

[13] Jun, B, Kocher, P. "The Intel random number generator." *Cryptography Research, Inc. white paper prepared for Inter Corp. Apr* (1999) www.cryptography.com/resources/whitepapers/IntelRNG.pdf.

[14] Lenstra, A. K. and Wesolowski, B. "Trustworthy public randomness with sloth, unicorn, and trx", preprint 2015, http://eprint.iacr.org/2015/366.pdf; to appear in *Int. J. Appl. Cryptogr.*

[15] Menezes, A. J., Oorschot, P. C. and Vanstone, S. A. *Handbook of applied cryptography.* (1997).

[16] Marton, K., and Suciu, A. "Towards a methodology for randomness assessment: Challenges and pitfalls." *Proceedings of The Romanian Academy Series A-Mathematics Physics Technical Sciences Information Science 16* (2015): 385-394.

[17] Miller, A., and Bentov, I. "Zero-Collateral Lotteries in Bitcoin and Ethereum." arXiv preprint arXiv:1612.05390 (2016).

[18] "NIST Removes Cryptography Algorithm from Random Number Generator Recommendations." *National Institute of Standards and Technology*. 21 April 2014.

[19] Ozoguz, S., Ates, O., Elwakil, A. S. "An integrated circuit chaotic oscillator and its application for high speed random bit generation." *In Proceedings of the international symposium on circuit and systems (ISCAS 2005)*, (2005), 4345-8.

[20] Petrie, C. S., Connelly, J. A. "A noise-based IC random number generator for applications in cryptography." *IEEE T. Circuits*, I47, 615-621, 2000.

[21] Petrie, C. S., Connelly, J. A. "Modeling and simulation of oscillator-based random number generators." *In Proceedings of the IEEE international symposium on circuits and systems (ISCAS '96)*, (4), (1996), 324-7.

[22] Pierrot, C., and Wesolowski, B. "Malleability of the blockchain's entropy", preprint 2016, https://eprint.iacr.org/2016/370.pdf.

[23] Popov, S. "On a decentralized trustless pseudo-random number generation algorithm." *Journal of Mathematical Cryptology*, 11.1 (2017): 37-43. DOI: 10.1515/jmc-2016-0019

[24] Schindler, W., Killmann, W. "Evaluation criteria for true (physical) random number generators used in cryptographic applications." *LNCS, vol. 2523*, pp. 431-449. Springer, 2002.

[25] Stojanovski, T, Kocarev, L. "Chaos-based random number generators, Part I: analysis." *IEEE Trans Circuits Systems I*, 48(3), (2001), 281-8.

[26] Syta E., Jovanovic P., Kokoris-Kogias E., Gailly N., Gasser L., Khoffi I., Fischer M.J., and Ford B. "Scalable bias-resistant distributed randomness." *38th IEEE Symposium on Security and Privacy*, 2017.

[27] Syta, E., Tamas, I., Visher, D., Wolinsky, D. I., Gasser, L., Gailly, N., and Ford, B. "Keeping authorities "honest or bust" with decentralized witness cosigning", preprint 2015, https://arxiv.org/abs/1503.08768v2.

[28] Wiggins, S. *Introduction to applied nonlinear dynamical systems and chaos.* Berlin: Springer-Verlag, (1990).

[29] Yalcin, M. E., Suykens, J. A. K., and Vandewalle, J. "True random bit generation from a double scroll attractor." *IEEE Trans Circuits Systems I: Fund Theory Appl*, 51(7), (2004), 1395-404.