Article

Inflation propensity of Collatz orbits: a new proof-of-work for blockchain applications

Fabian Bocart [1]*

[1] 3354 83rd Street, Jackson Height, NY [1]
* Correspondence: fabian.bocart@gmail.com

Version September 25, 2018 submitted to Preprints

Abstract: Cryptocurrencies like Bitcoin rely on a proof-of-work system to validate transactions and prevent attacks or double-spending. Reliance on a few standard proofs-of-work such as hashcash, Ethash or Scrypt increases systemic risk of the whole crypto-economy. Diversification of proofs-of-work is a strategy to counter potential threats to the stability of electronic payment systems. To this end, another proof-of-work is introduced: it is based on a new metric associated to the algorithmically undecidable Collatz algorithm: the inflation propensity is defined as the cardinality of new maxima in a developing Collatz orbit. It is numerically verified that the distribution of inflation propensity slowly converges to a geometric distribution of parameter $0.714 \approx \frac{(\pi-1)}{3}$ as the sample size increases. This pseudo-randomness opens the door to a new class of proofs-of-work based on congruential graphs.

Keywords: Geometric distribution, Collatz conjecture, inflation propensity, systemic risk, cryptocurrency, blockchain, proof-of-work

MSC: 60E05, 62E17

JEL Classification: C46, C65, O39

1. Introduction

A decentralized electronic payment system relies on a ledger of transactions shared on a network. The decentralization of a transaction ledger raises the question of security and integrity of the ledger. In the original Bitcoin protocol, the problem of double-spending or alteration of the ledger is solved by the use of blockchain, a system that requires proof-of-work by a network of computers to confirm transactions. In cryptography, intensive computation as proof-of-work allows one party to verify with little computational effort that a counterparty has spent a large amount of computational effort. The concept was originally developed by [6] as a spam prevention technique. [16] used for Bitcoin a proof-of-work based on [1]. The protocol consists in finding a nonce value such that the application of the SHA256 hashing algorithm to a combination of that nonce and a block of information gives a hash starting with series of zeroes by targetting a given threshold. The idea behind the proof-of-work is that participants have an incentive to cooperate rather than to cheat because the computational power required to cheat is too large. However, as cryptocurrencies became more popular and diverse, an over-reliance on mainstream proof-of-work protocols, such as hashcash, Ethash (Wood , 2014) or Scrypt (Percival , 2009) creates a new type of systemic risk in which a cryptographic breakdown would jeopardize cryptocurrencies that rely on these standard proofs-of-work. Diversification of the proofs-of-work is a credible way to mitigate this systemic risk. Other types of proof-of-work have been designed, such as prime numbers verification (King , 2013), graph-theoretic proof-of-work (Tromp , 2015) or asymmetric proof-of-work based on the generalized birthday problem (Biryukov & Khovratovich , 2017). Attacks on proofs-of-work could prevent new transactions or alter past ones. In financial markets, exchanges have the possibility to cancel trades in case of infrastucture breakdown or malfunction. However, a systemic failure of the proof-of-work system in decentralized

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1

cryptocurrency markets could mean the destruction of the whole history of transactions. Potential risks clouding the proof-of-work system include innovation in mathematics and cryptography that could compromise the existing proofs-of-work. Further diversification of proofs-of-work could help mitigate that systemic risk and improve robustness of the nascent crypto-economy. However, additional threats to the cryptocurrency ecosystems could also come from technological innovation, such as for instance the introduction of quantum computers. Post-quantum algorithms are currently being developed in the field of security, see e.g. [2]. In particular, [10] propose a quantum-safe blockchain that utilizes quantum key distribution. The application presented in the following sections does not address the problems posed by a post-quantum world but suggests new ways to tackle immediate systemic threats by offering a proof-of-work based on properties of the Collatz algorithm. In order to describe this algorithm, consider the following function from $\mathbb{N}_0$ to $\mathbb{N}_0$:

$$T(x) = \begin{cases} x/2 & \text{if } x \text{ is even} \\ (3x+1)/2 & \text{if } x \text{ is odd} \end{cases} \tag{1}$$

Now, apply the following iterate of T:

$$\begin{cases} T^0(x) = x \\ T^{(k+1)}(x) = T(T^k(x)) \end{cases} \tag{2}$$

The Collatz conjecture states that $\forall\, x \in \mathbb{N}_0, \exists$ a finite k such that $T^k(x) = 1$. [14] uses the following terminology: the "total stopping time" is defined as $\sigma_\infty(x) = \inf\{k : T^k(x) = 1\}$. The "stopping time" $\sigma(x)$ is $\inf\{k : T^k(x) < x\}$. The "gamma value" is defined as $\gamma(x) = \frac{\sigma_\infty(x)}{log(x)}$.
For instance, let us consider the case for $x = 3$:

$$\begin{cases} T^0(3) = 3, \\ T^1(3) = (3 \times 3 + 1)/2 = 5, \\ T^2(3) = (5 \times 3 + 1)/2 = 8, \\ T^3(3) = 8/2 = 4, \\ T^4(3) = 4/2 = 2, \\ T^5(3) = 2/2 = 1. \end{cases} \tag{3}$$

In this example, the Collatz sequence[1] is $\left\{3, 5, 8, 4, 2, 1\right\}$ and $\sigma_\infty(3)$ equals to 5 while $\sigma(3) = 4$. By definition, the value of $\sigma_\infty(x)$ depends on the starting point of the algorithm. For example $\forall\, \alpha \in \mathbb{N}_0$, $\sigma_\infty(2^\alpha) = \alpha$ as

$$T^\alpha(2^\alpha) = 1. \tag{4}$$

Analyzing the total stopping time $\forall\, x \in \mathbb{N}_0$ has proven challenging: the lack of clear patterns and the absence of an analytical shortcut to estimate $\sigma_\infty(x)$ have left practitioners with numerical methods to compute it and verify the conjecture. [18] proved computationally that the conjecture holds up until $x = 20 \times 2^{58}$. Current computational capabilities have allowed confirming the conjecture for very large numbers. For example, [8] introduced a GPU-based method to verify the Collatz algorithm. The authors could verify 1.31×10^{12} 64-bit numbers per second. A probabilistic approach is also a frequent workaround to justify the validity of the Collatz conjecture: assuming function $T^k(x)$ is "random enough", [5] showed that half of the time, the next number in the sequence will be $(3x+1)/2$, then $1/4$ of the time it will be $(3x+1)/4$, then $1/8$ of the time it will be $(3x+1)/8$ and so on so that the average

[1] also called "trajectory" or "forward orbit"

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1

growth in the sequence will be $(\frac{3}{2})^{1/2}(\frac{3}{4})^{1/4}(\frac{3}{8})^{1/8}(\frac{3}{16})^{1/16}(\frac{3}{32})^{1/32}... = \frac{3}{4} < 1$. [20] demonstrated that the set of integers {x: x has stopping time $\leq$ k} has a limiting asymptotic density $F(k)$ with $F(k) \to 1$ as $k \to \infty$. These elements tend to indicate that $T^k(x)$ does not diverge to infinity as k grows. Using [15] machines, [4] showed that a problem generalizing the Collatz problem is not algorithmically decidable. [12] extended the proof to show that this generalization is Π_0^2 complete. If the problem is algorithmically undecidable, then no information about the future inflation of the Collatz map is passed from one step k to the next step $k+1$. To explore that hypothesis and the properties of this "pseudo-randomness", let us define the *inflation propensity of order K* $\xi(x, K)$ as the cardinality of the set of steps that lead to a number strictly larger than all previous numbers in the same sequence:

$$\xi(x, K) = \textbf{card}\left\{k : T^k(x) > \max(M_{x,k})\right\}, k = 1, ..., k, ...K, \tag{5}$$

where $M_{x,k} = \left\{T^0(x), T^1(x), ..., T^k(x)\right\}$. $\xi(x, \sigma_\infty(x))$ is a particular case. For the ease of notation: $\xi(x) = \xi(x, \sigma_\infty(x))$. In the above example of $x = 3$, $\xi(3) = 2$. Indeed, the set of numbers strictly larger than the previous maxima in the sequence are $\{5, 8\}$ so that $\xi(3) = \textbf{card}\{5, 8\} = 2$. In the other example presented supra with $x = 2^\alpha$, $\xi(2^\alpha) = 0 \; \forall \; \alpha \in \mathbb{N}_0$ since no number in their sequences can be strictly larger than the initial one.

This research paper investigates the distribution of $\xi(x)$, the inflation propensity as a deterministic variable that resembles a random behavior. If past maxima anywhere in the sequence are independent from new maxima later computed in that orbit, we should have that $\xi(x) \sim G(\rho)$, a geometric distribution of parameter ρ with density $f(\xi(x) = y) = \rho^y(1-\rho)$. The interests of fitting a density distribution to $\xi(x)$ are multiple: first, in absence of proof of the Collatz conjecture, numerical analysis of the problem stays relevant towards resolving the question. Second, by properly addressing the behavior of the series for large numbers, one can help anticipate the computational challenges related to exploring the orbits of the Collatz map. Third, identifying pseudo-random behaviour of Collatz inflation propensity directly leads to a new class of proofs-of-work for blockchain applications. The remainder of this document is built as follows: the next section discusses the empirical distributions of $\sigma_\infty(x)$, $\sigma(x)$ and $\xi(x) \; \forall \; x \in \mathbb{N}_0$. The third section details the observed density of $\xi(x)$. The density parameter of a geometric distribution is estimated using all natural numbers up to 1e11 as sample. The fourth section presents a new proof-of-work based on inflation propensity. The last section concludes.

2. Inflation propensity

[13] describes the $3x+1$ conjecture as "a deterministic process that simulates random behaviour" and goes further to mention that the problem seems "structureless". [22] formally proves the non-regularity of the Collatz's graph. As a visual illustration of this "structurlessness", the total stopping time for the first 1e6 natural numbers as a function of their value is presented in Figure 1. The equally "structureless" empirical distribution of the total stopping time for the same numbers is presented in Figure 2.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1

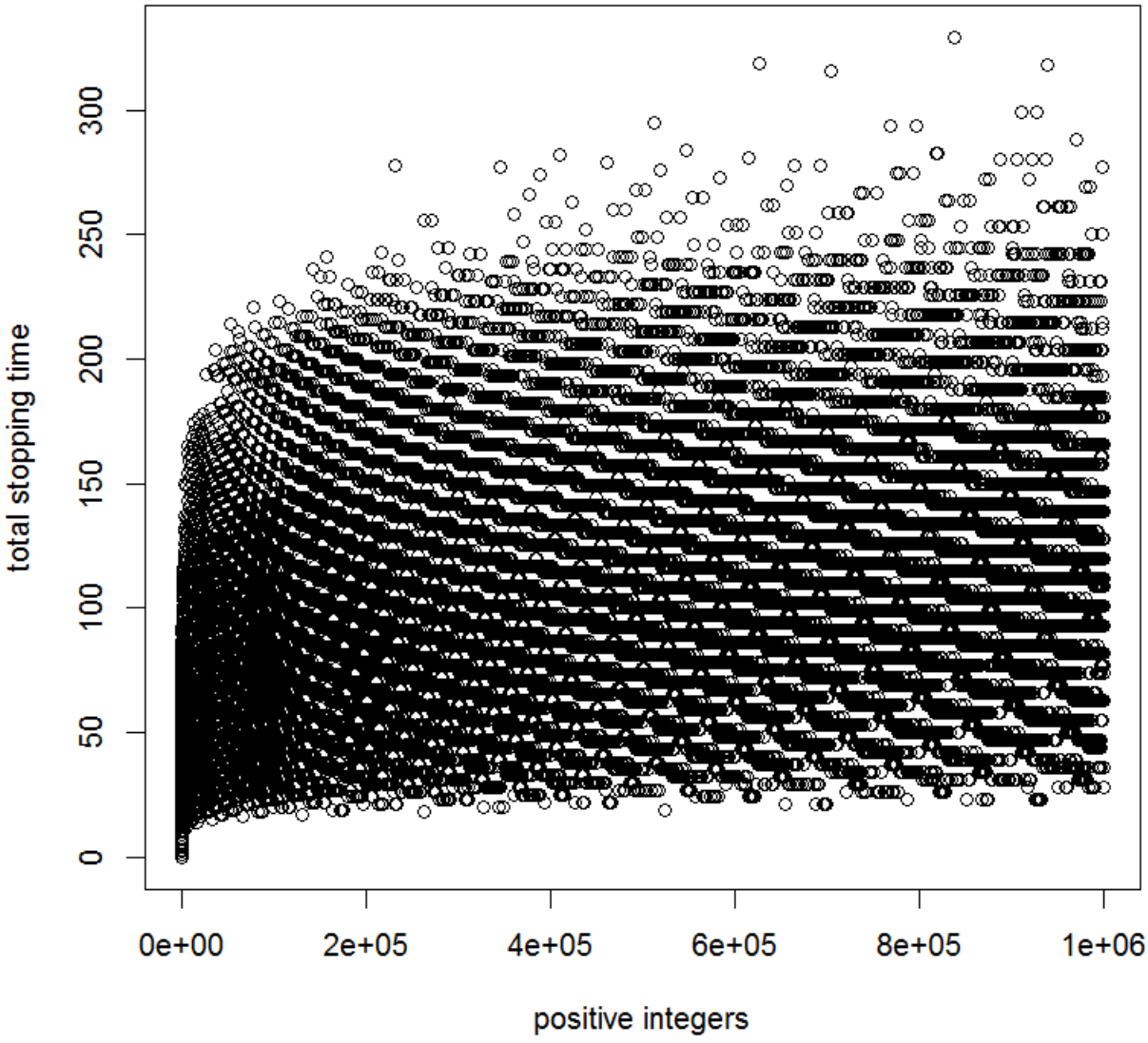

Figure 1. "Structureless" total stopping time for the first 1e6 natural numbers

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1

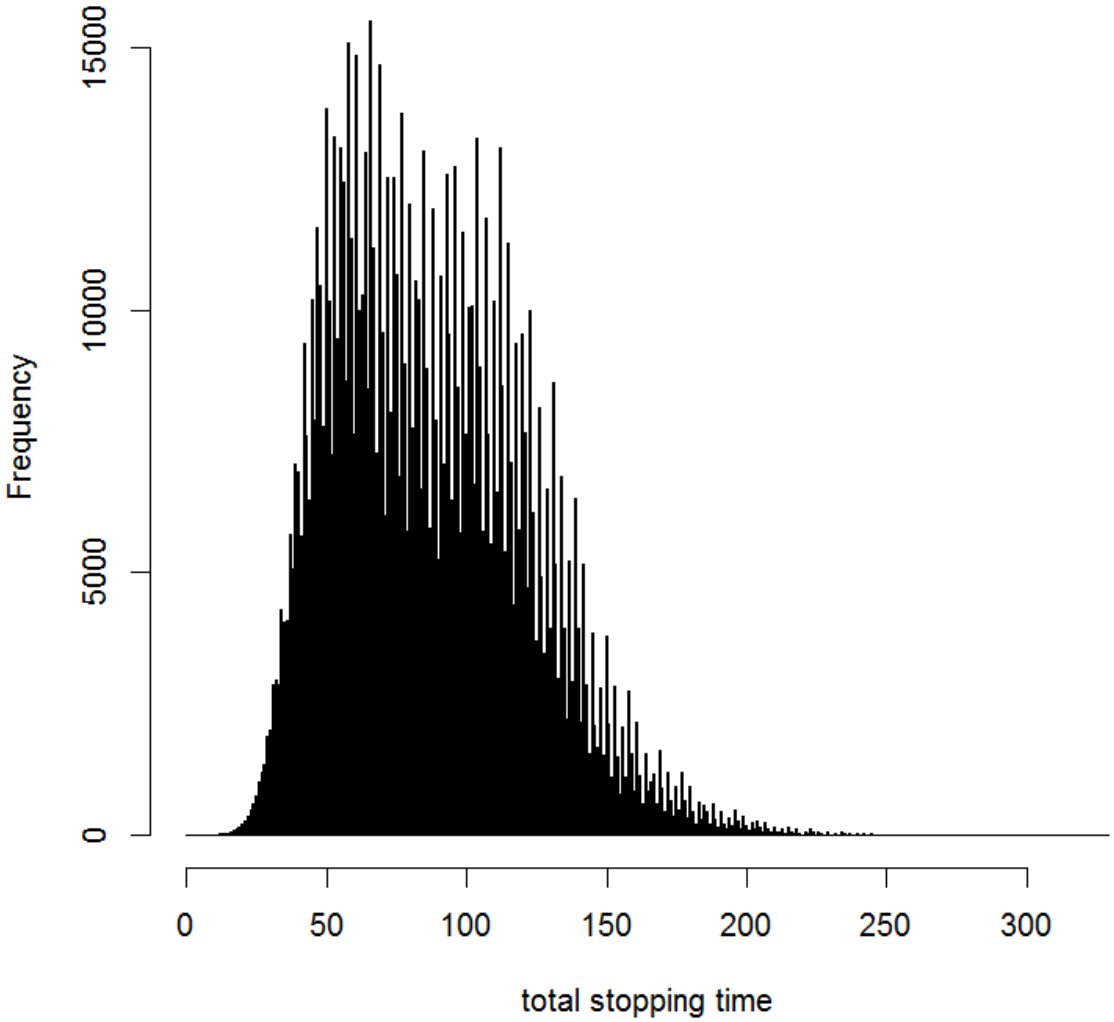

Figure 2. "Structureless"distribution of the total stopping time for the first 1e6 natural numbers

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1

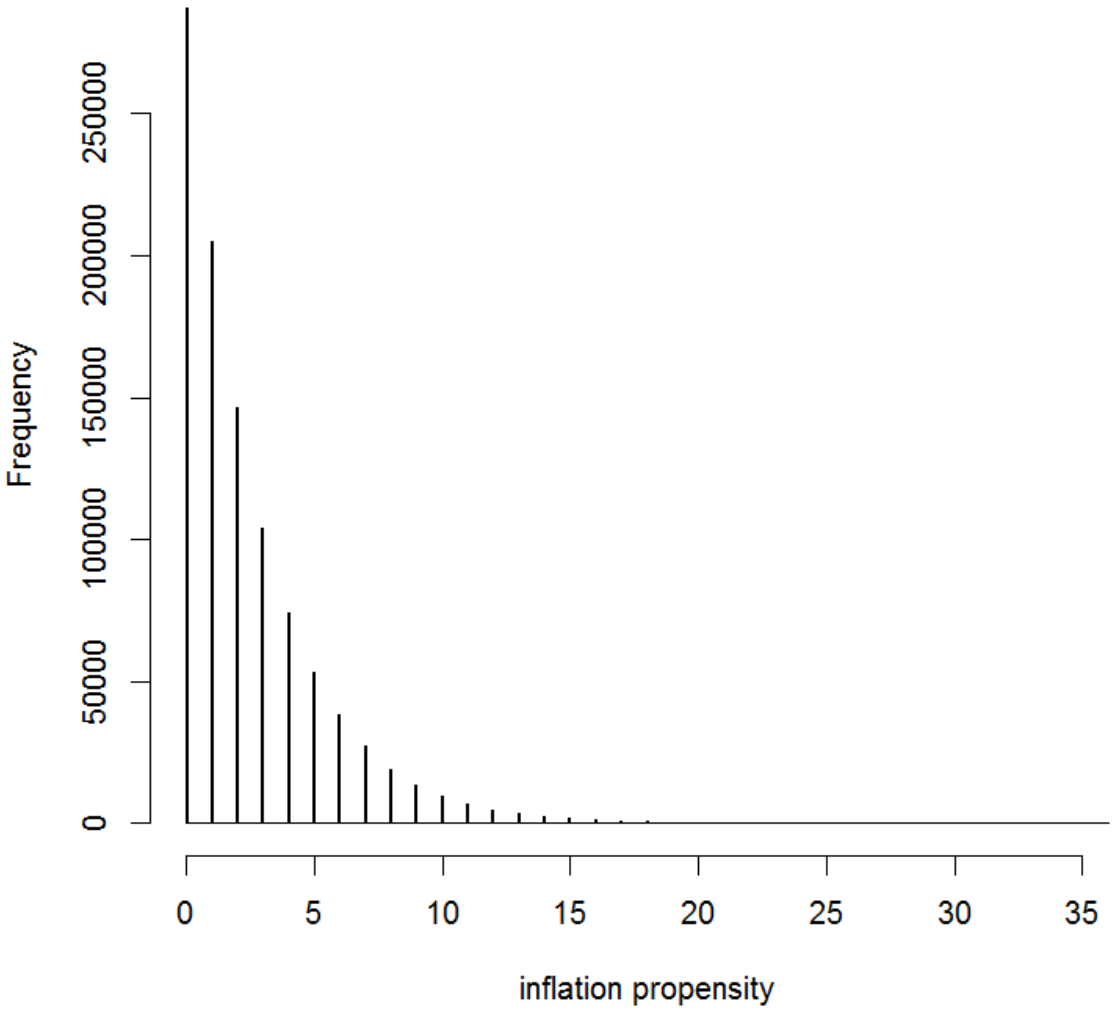

Figure 3. Quasi-geometric distribution of the inflation propensity for the first 1e6 natural numbers

100

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1

Because the Collatz graph is non-regular, its complexity gives rise to a pseudo-random behaviour. [17] and [11] explore similarities between the Collatz model and the following dynamical system:

$$\log_2 T^K(x) \approx log_2 x - K + b_3 \sum_{k=0}^{K} Y_k, \tag{6}$$

where b_3 is a constant and Y_k are IID (independant and identically distributed) Bernouilli random variables. The stochastic models predict that all orbits converge to a bounded set and that the total stopping time $\sigma_\infty(x)$ for the $3x+1$ map of random starting point x is about 6.95212 $\log x$ steps, as $x \to \infty$ have a normal distribution centered around that value. The authors point out that a suitable scaling limit for the trajectories is a geometric Brownian motion. This approach is extended in the current research: the empirical distribution of $\xi(x)$ defined in (5) is presented in Figure 3. Let us assume that the amount of new maxima in any given orbit is independent from the amount of previously found maxima in that orbit. This would mean that for any random starting point $x > 4 \in \mathbb{N}_0$:

$$P(\xi(x,K) > \xi(x,K-1) \mid \{\xi(x,K-1),...,\xi(x,1)\}) = P(\xi(x,K) > \xi(x,K-1). \tag{7}$$

Assuming such memorylessness naturally yields a geometric distribution of the inflation propensity. The longer the stopping time, the larger the orbits. Let us assume that the probability to reach a new maximum is memoryless for large orbits and that the density $f(\xi(x) = y)$ follows a geometric distribution. It would mean that

$$f(\xi(x) = y) = \rho^y(1-\rho) \tag{8}$$

with $\rho \in]0;1[$ and $y \in \mathbb{N}$. The moment generating function is

$$\mu_n = Li_{-n}(\rho) - \rho Li_{-n}(\rho), \tag{9}$$

where $Li_n(\rho)$ is the nth polylogarithm of ρ and

$$\hat{\rho} = \frac{\mu_1}{1+\mu_1} \tag{10}$$

is the corresponding estimator of ρ based on (9). It is also the maximum likelihood estimator. The next step is to test the hypothesis that $\xi(x) \sim G(\rho)$.

3. Empirical results

The samples consist in the first 1e8, 1e9, 1e10 and 1e11 positive integers. For each sample, the maximum likelihood estimator of ρ is computed, then tests are performed to see if elements of the distribution follow a geometric distribution of parameter ρ:

$$H_0 : P(\xi(x) = n) = (1-\rho)^{n-1}\rho \quad \forall n = 1,...,q \tag{11}$$

$$H_1 : P(\xi(x) = n) \neq (1-\rho)^{n-1}\rho \quad \forall n = 1,...,q \tag{12}$$

where $q \in [0, N]$ and N is the largest observed maximum in the sample. When $q = N$, the entire distribution is tested for goodness of fit with a geometric distribution of parameter $\hat{\rho}$. The tests are performed using Pearson's χ^2 test at a 10% confidence level. Table 1 summarizes the results of the tests. As the sample size increases, the hypothesis is not rejected when it comes to considering the first quantiles of the distribution. For the last sample (1e11), the hypothesis that the distribution of the inflation propensity follows a geometric distribution cannot be rejected up to the 91th percentile, compared to the 49th percentile for the 1e9 sample. Computational limitations prevent at this stage investigating larger sample sizes so that the geometric behaviour of the inflation propensity over the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1

entire domain ($\mathbb{N}_0$) needs to be conjectured. Interestingly, the estimator for ρ seems also to converge to a given value as the size of the sample increases and is very close to $\frac{\pi-1}{3}$, which is coincidentally the solution to the equation $3x + 1 = \pi$ (see Figure 4). Table A1 in Appendix indicates the distribution of inflation propensities for the first 1e11 integers.

		Sample 1e8	Sample 1e9	Sample 1e10	Sample 1e11
$\hat{\rho}$		0.7133482	0.7135956	0.713667	0.713681
q	percentile	p-value	p-value	p-value	p-value
0	29	0.01	**0.15**	**0.70**	**0.64**
1	49	0.04	**0.19**	**0.70**	**0.14**
2	64	0.09	0.00	**0.23**	**0.25**
3	74	**0.15**	0.00	**0.36**	**0.39**
4	82	0.08	0.00	**0.11**	**0.35**
5	87	0.05	0.00	0.01	**0.37**
6	91	0.07	0.00	0.00	**0.13**
7	93	**0.11**	0.00	0.00	0.03
8	95	0.00	0.00	0.00	0.04
9	97	0.00	0.00	0.00	0.03
10	98	0.00	0.00	0.00	0.00

Table 1. Pearson's χ^2 tests for goodness of fit with a geometric distribution

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1

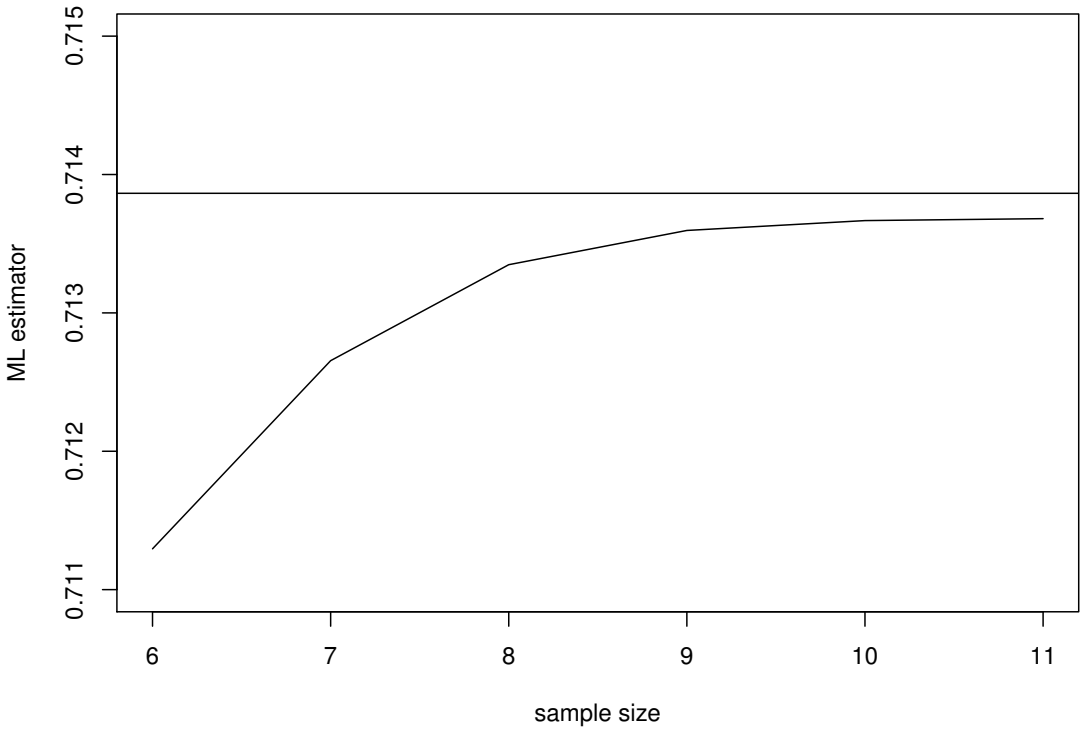

Figure 4. $\hat{\rho}$ as a function of the sample size (log10-scale)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1

4. Application

4.1. Collatz-based proof-of-work

Because the distribution of the inflation propensity of Collatz orbits can be assumed to be geometric over large samples, and that the Collatz algorithm has been proven to be undecidable, the inflation propensity can be considered as a new candidate to generate proofs-of-work. Consider the following problem: find the smallest possible value X such that the sum of X and a hashed block of information B will have an inflation propensity of a given value Q. In other terms:

$$X = \min\{x > B : \xi(x) = Q\} - B \quad x, B, Q \in \mathbb{N}_0. \tag{13}$$

Since $P(\xi(x) = Q) \approx (\frac{\pi-1}{3})^Q(1 - \frac{\pi-1}{3})$ the difficulty of the verification is tailored in a very straightforward manner: higher targets Q will be exponentially more difficult to find. A natural extension is to find a set $X = \{X_1, X_2, ..., X_n\}$ whose n elements all lead to a combination of values $Q = \{Q_1, Q_2, ..., Q_n\}$, so that the probability of occurence can be more precisely selected. However, verifying the proof Q given X and B is computationally straightforward, a desirable property for a proof-of-work. Contrary to the hashcash proof-of-work, there can exist only one solution to the proof-of-work with the Collatz approach.

At the exception of the nonce and the target Q, the remainder of blockchain application based on Collatz can be identical to the existing Bitcoin protocol. The nonce is simply the value of X. The target set Q can be selected by the network so that, similar to Bitcoin, 6 blocks are mined per hour. Every 2016 blocks, clients can compare the performance of the network and adjust the difficulty accordingly. Thanks to the geometric nature of the inflation propensity, a protocol for this adjustment is straightforward. Let us assume U_0 is the average amount of time required by the network to find any single value $\xi(x)$. Any total computational time $U_T \geq U_0$ can be easily selected by finding a set Q solving the following problem:

$$U_T = \sum_{q \in Q} \frac{1}{\rho^q} U_0 + \epsilon. \tag{14}$$

Three additional constraints must be considered for the protocol to have a unique solution and be properly defined: the set must be chosen so that $0 \leq \epsilon \leq U_0$, the cardinality of the set must be as small as possible and $\max(Q)$ should be set to an arbitrary number based on contemporary knowledge of the empirical distribution of the inflation propensity. It is suggested to set that maximum value to 50, which corresponds to an observed empirical probability of occurence 1.44e-08 in the first 1e11 integers.

4.2. Example: Bitcoin genesis hash

A new Bitcoin genesis hash is created using original inputs by [16], but exploiting inflation propensity proof-of-work instead of hashcash. The inputs are: a hash merkle root that condenses all information related to the first Bitcoin transaction, a version number, a public key, a date, a time stamp that is used as coinbase parameter, and a target for complexity. A genesis block is the first block of a blockchain. To create a genesis hash using inflation propensity as proof-of-work, only two adjustments to the Bitcoin protocol are required: first, the target for complexity is expressed with an integer, which is the targeted inflation propensity. This directly relates to a specific probability of occurence. Second, the hashcash is replaced with the inflation propensity algorithm. In practice, the block header is hashed using SHA256 then converted into an integer using hexadecimal encoding. This corresponds to B in equation (13). The target set Q is arbitrarily set to a single value of 40 for the generation of this first hash, which corresponds to a probability of occurence of $\sim$ 4e-07. The value of B given Nakamoto's other initial inputs is of $\sim$3.57e98. The X nonce is then incrementally added to the integer B and inflation propensity is computed until the target of 40 is reached. The values obtained from each iteration are hereafter named "Xis". In the python implementation of the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1

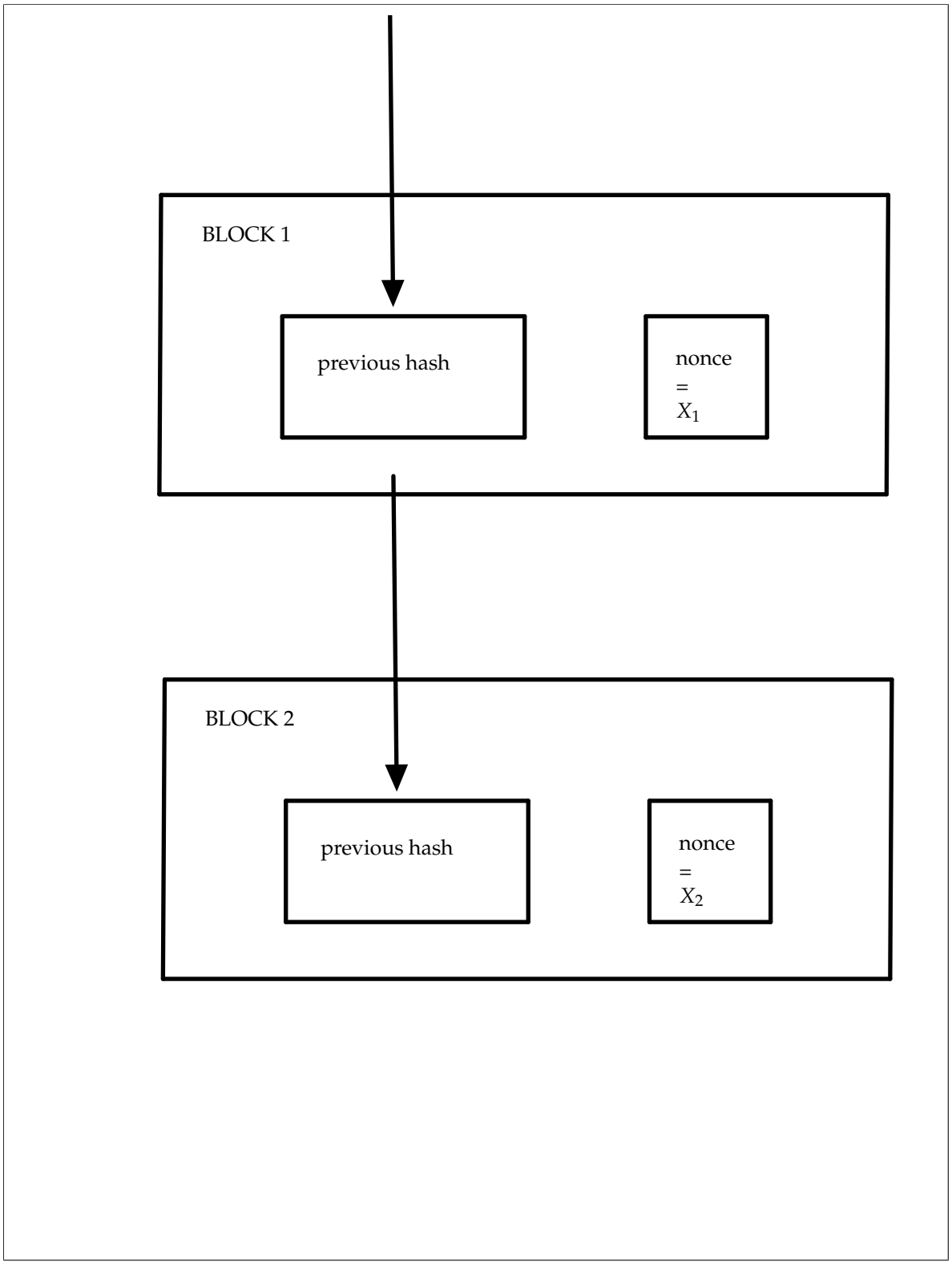

Figure 5. Proof-of-work system in the blockchain

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1

algorithm, 1248 Xis are computed per second on an Intel Core i7-4700MQ CPU with 8 x 2.40GHz. After 11 minutes of computation, the solution is found. Table 2 describes diagnostics and results of the genesis hash. Using this first instance to calibrate the computational difficulty, the smallest set Q that solves equation (14) that would yield an expected computational time of 10 minutes for the next block would be $\{2, 15, 20, 26, 30, 40\}$.

block header hash	37d25f7f472fde7bb5b84f4bb319097c580383911b45eff10e68afa06073d6c0
corresponding integer	3571253878495152629381884201940447188348330958155209882109959676568968733276070653745557479068647504
merkle hash	4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b
pszTimestamp	The Times 03/Jan/2009 Chancellor on brink of second bailout for banks
pubkey	04678afdb0fe5548271967f1a67130b7105cd6a828e03909a67962e0ea1f61deb649f6bc3f4cef38c4f35504e51ec112de5c384df7ba0b8d578a4c702b6bf11d5f
time	1231006505
inflation propensity target	40 (0x28)
nounce	848559
genesis hash	891b06db313f1ffb18c2bc46aac9515bfe83c502a3cc1e9d02a9cb576c9ac87e
computational time	11 minutes

Table 2. A genesis hash based on original Bitcoin's inputs for genesis but using inflation propensity as proof-of-work

4.3. Advantages of the Collatz-based proof-of-work

The advantages of a Collatz-based proof-of-work are many. From a practitioner perspective, the algorithm is easy to implement since the underlying problem is made of simple arithmetic operations. Also, the Collatz algorithm is known to be algorithmically undecidable, which guarantees asymmetry: it is difficult to find the targeted value but easy to verify. Furthermore, the inflation-propensity based proof-of-work has only one easily verifiable solution, which simplifies the consensus for a proof. The geometric distribution allows a very convenient tailoring of the computational complexity, by adjusting a specific targeted inflation-propensity, or a combination of targets. The same algorithm can also be indefinitely extended to meet new computational improvements since the upper bound of the orbits is infinity. In addition to this scalability, it could be possible to generalize the $3x + 1$ algorithm to other congruential graphs exhibiting the same properties (for example, the $5x + 1$ graph). Provided further research confirms this hypothesis, such a feature could allow more possibilities to generate proofs-of-work. From a purely computational perspective, computing as many orbits as possible and recording inflation propensities of integers is useful to marginally decrease the computational time of mining a block. The practical consequence of pre-computation is very clear: miners do not actually need to compute the entire orbit for proof-of-work in real-time, but have the possibility to store orbits in memory. This also offers the possibility of ASIC resistance (Application Specific Integrated Circuits) by the use of memory-hardness since holding orbits in memory gives substantial advantage over real-time computation. Finally, computing proofs-of-work by exploring Collatz orbits contributes directly to numerical verification of the Collatz conjecture for large numbers. This provides an elegant purpose to using computational power for proof-of-work.

5. Conclusion

For the classical $3x + 1$ map, it is conjectured that inflation propensity $\xi(x) = \mathbf{card}\left\{k : T^k(x) > \max(M_{x,k})\right\}, k = 1, ..., k, ...\sigma_\infty(x)$ has a geometric density distribution whose parameter's value $\rho \approx \frac{\pi-1}{3}$. This has been verified numerically for the first 1e11 integers. Standardization of proofs-of-work has led to increased systemic risk in case of cryptographic failure of the hashcash, Ethash or Scrypt protocols. The inflation propensity of Collatz orbits is a new metric that exhibits properties particularly well suited to be the base for new cryptography applications. A new proof-of-work is suggested: finding the smallest possible value X such that the sum of X and a hashed block of information B has

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1

an inflation propensity of a given value Q. Advantages of this approach are multiple including an infinite scalability and the possibility to easily tune complexity of the algorithm. While this opens the way to managing systemic risks in cryptocurrencies by diversifying the proofs-of-work, further research is needed to generalize this type of proof-of-work to a larger class of congruential graphs.

Appendix

Distribution of inflation propensity for first 1e11 integers

$\xi(x)$	Observations
0	28631964381
1	20434254718
2	14583348496
3	10407804534
4	7427954284
5	5301161512
6	3783166989
7	2699976430
8	1927052441
9	1375229862
10	981424318
11	700353911
12	499868474
13	356795944
14	254706290
15	181761315
16	129757032
17	92628127
18	66127176
19	47199172
20	33676458
21	24024158
22	17138021
23	12231945
24	8727118
25	6225787
26	4432544
27	3162432
28	2251004
29	1599248
30	1139341
31	814975
32	583455
33	416994
34	298683
35	212914
36	150443
37	106613
38	76749
39	55452
40	39947
41	28495
42	20259
43	14253
44	10396
45	7791
46	5431
47	3690
48	2640
49	1984
50	1448
51	1041
52	745
53	595
54	467
55	347
56	234
57	170
58	127
59	72
60	41
61	21
62	20
63	17
64	17
65	9
66	2
67	0
68	1
69	0

Table A1. Distribution of inflation propensity $\xi(x)$ for the first 1e11 integers

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1

Python code for genesis block

Modified from [7]).

```
import hashlib, struct, binascii, time, sys, optparse

from construct import *

def main():
  options = get_args()
  input_script = create_input_script(options.timestamp)
  output_script = create_output_script(options.pubkey)
  tx = create_transaction(input_script, output_script,options)
  hash_merkle_root = hashlib.sha256(hashlib.sha256(tx).digest()).digest()
  print_block_info(options, hash_merkle_root)
  block_header      = create_block_header(hash_merkle_root, options.time, options.bits, options.nonce)
  genesis_hash, nonce = generate_hash(block_header, options.nonce, options.bits)
  announce_found_genesis(genesis_hash, nonce)

def get_args():
  parser = optparse.OptionParser()
  parser.add_option("-t", "--time", dest="time", default=int(1231006505), type="int", help="the (unix) time
  when the genesisblock is created")
  parser.add_option("-z", "--timestamp", dest="timestamp", default=
  "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks",
                type="string", help="the pszTimestamp found in the coinbase of the genesisblock")
  parser.add_option("-n", "--nonce", dest="nonce", default=0,
                type="int", help="the first value of the nonce that will be incremented
                 when searching the genesis hash")
  parser.add_option("-p", "--pubkey", dest="pubkey", default="04678afdb0fe5548271967f1a67130b7105cd6a828e03909
  a67962e0ea1f61deb649f6bc3f4cef38c4f35504e51ec112de5c384df7ba0b8d578a4c702b6bf11d5f",
                type="string", help="the pubkey found in the output script")
  parser.add_option("-v", "--value", dest="value", default=5000000000,
                type="int", help="the value in coins for the output, full value (exp. in bitcoin 5000000000
                - To get other coins value: Block Value * 100000000)")
  parser.add_option("-b", "--bits", dest="bits",
                type="int", help="the target in compact representation, associated to a difficulty of 1")
  (options, args) = parser.parse_args()
  if not options.bits:
     options.bits = 40
  return options

def create_input_script(psz_timestamp):
  psz_prefix = ""
  if len(psz_timestamp) > 76: psz_prefix = '4c'
  script_prefix = '04ffff001d0104' + psz_prefix + chr(len(psz_timestamp)).encode('hex')
  print (script_prefix + psz_timestamp.encode('hex'))
  return (script_prefix + psz_timestamp.encode('hex')).decode('hex')

def create_output_script(pubkey):
  script_len = '41'
  OP_CHECKSIG = 'ac'
  return (script_len + pubkey + OP_CHECKSIG).decode('hex')

def create_transaction(input_script, output_script,options):
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1


```
  transaction = Struct("transaction",
    Bytes("version", 4),
    Byte("num_inputs"),
    StaticField("prev_output", 32),
    UBInt32('prev_out_idx'),
    Byte('input_script_len'),
    Bytes('input_script', len(input_script)),
    UBInt32('sequence'),
    Byte('num_outputs'),
    Bytes('out_value', 8),
    Byte('output_script_len'),
    Bytes('output_script', 0x43),
    UBInt32('locktime'))

  tx = transaction.parse('\x00'*(127 + len(input_script)))
  tx.version         = struct.pack('<I', 1)
  tx.num_inputs      = 1
  tx.prev_output     = struct.pack('<qqqq', 0,0,0,0)
  tx.prev_out_idx    = 0xFFFFFFFF
  tx.input_script_len = len(input_script)
  tx.input_script    = input_script
  tx.sequence        = 0xFFFFFFFF
  tx.num_outputs     = 1
  tx.out_value       = struct.pack('<q' ,options.value)
  tx.output_script_len = 0x43
  tx.output_script   = output_script
  tx.locktime        = 0
  return transaction.build(tx)

def create_block_header(hash_merkle_root, time, bits, nonce):
  block_header = Struct("block_header",
    Bytes("version",4),
    Bytes("hash_prev_block", 32),
    Bytes("hash_merkle_root", 32),
    Bytes("time", 4),
    Bytes("bits", 4),
    Bytes("nonce", 4))

  genesisblock = block_header.parse('\x00'*80)
  genesisblock.version       = struct.pack('<I', 1)
  genesisblock.hash_prev_block = struct.pack('<qqqq', 0,0,0,0)
  genesisblock.hash_merkle_root = hash_merkle_root
  genesisblock.time          = struct.pack('<I', time)
  genesisblock.bits          = struct.pack('<I', bits)
  genesisblock.nonce         = struct.pack('<I', nonce)
  return block_header.build(genesisblock)

#Collatz inflation propensity
def inflation_propensity(x):
    xMax=x
    stepToMaximum=0
    while x > 1:
        if x % 2 == 0:
            x = x / 2
        else:
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 September 2018 doi:10.20944/preprints201809.0472.v1


```
            x = (3 * x + 1) / 2
        if x > xMax:
            xMax=x
            stepToMaximum+= 1
    return stepToMaximum

def generate_hash(data_block, start_nonce, bits):
  print 'Searching for genesis hash..'
  nonce          = start_nonce
  last_updated   = time.time()
  header_hash = generate_hashes_from_block(data_block)
  print(binascii.hexlify(header_hash))
  orbitTrajectory=int(header_hash.encode('hex_codec'), 36)
  print(orbitTrajectory)
  timeInit=time.time()
  while True:
    xi=inflation_propensity(orbitTrajectory)
    last_updated = calculate_hashrate(nonce, last_updated, orbitTrajectory,timeInit)
    if xi==bits:
      return (generate_hashes_from_block(data_block), nonce)
    else:
     nonce     = nonce + 1
     orbitTrajectory += 1
     data_block = data_block[0:len(data_block) - 4] + struct.pack('<I', nonce)

def generate_hashes_from_block(data_block):
  header_hash = hashlib.sha256(hashlib.sha256(data_block).digest()).digest()[::-1]
  return header_hash

def calculate_hashrate(nonce, last_updated, orbitTrajectory, timeinit):
  if nonce % 10000 == 0:
    now            = time.time()
    hashrate       = round(10000/(now - last_updated))
    sys.stdout.write("\r%s Xis/s, Orbit: %s, Total time: %s minutes "
    %(str(hashrate), str(orbitTrajectory), str((now-timeinit)/60)))
    sys.stdout.flush()
    return now
  else:
    return last_updated

def print_block_info(options, hash_merkle_root):
  print "merkle hash: " + hash_merkle_root[::-1].encode('hex_codec')
  print "pszTimestamp: " + options.timestamp
  print "pubkey: "     + options.pubkey
  print "time: "       + str(options.time)
  print "bits: "       + str(hex(options.bits))

def announce_found_genesis(genesis_hash, nonce):
  print "genesis hash found!"
  print "nonce: "      + str(nonce)
  print "genesis hash: " + genesis_hash.encode('hex_codec')

main()
```

 doi:10.20944/preprints201809.0472.v1

References

1. Back, A. Hashcash-a denial of service counter-measure, **2002**.
2. Bae, Minyoung, Ju-Sung Kang, and Yongjin Yeom. A Study on the One-to-Many Authentication Scheme for Cryptosystem Based on Quantum Key Distribution. In *al Conference on Platform Technology and Service (PlatCon)*, **2017**, pp. 1-4. IEEE.
3. Biryukov, A., & Khovratovich, D. (2017). Equihash: Asymmetric proof-of-work based on the generalized birthday problem. *Ledger*, **2017**, *2*, 1-30.
4. Conway, J. H. (1972). Unpredictable iterations. In *Proceedings of the 1972 Number Theory Conference.*, University of Colorado, Boulder, **1972**, pp. 49–52.
5. Crandall, R. E.. On the 3x + 1 problem. *Mathematics of Computation.*, **1978**, *32(144)*, 1281-1292.
6. Dwork, C., & Naor, M. Pricing via processing or combatting junk mail. *Annual International Cryptology Conference*. Springer, Berlin, Heidelberg, **1992**, pp. 139-147.
7. Hartikka, L. GenesisH0. **2017**. Available online: https://github.com/lhartikk/GenesisH0. (accessed on 1 September 2018).
8. Honda, T., Ito, Y., & Nakano, K. GPU-accelerated Exhaustive Verification of the Collatz Conjecture. *International Journal of Networking and Computing*, 7(1), **2017**, 69-85.
9. King, S. Primecoin: Cryptocurrency with prime number proof-of-work, **2013**.
10. Kiktenko, E. O., Pozhar, N. O., Anufriev, M. N., Trushechkin, A. S., Yunusov, R. R., Kurochkin, Y. V. & Fedorov, A. K. Quantum-secured blockchain. *Quantum Science and Technology*, **2018**, *3(3)*, 035004.
11. Kontorovich, A. V., & Lagarias, J. C. Stochastic models for the 3x+ 1 and 5x+ 1 problems and related problems. In *The Ultimate Challenge: The 3x+ 1 Problem*, J. C. Lagarias, American Mathematical Society, **2010**, pp. 131-188, ISBN 978-0821849408.
12. Kurtz, S. A., & Simon, J. The undecidability of the generalized Collatz problem. In *International Conference on Theory and Applications of Models of Computation, May 2007*, **2007**, Springer Berlin Heidelberg, pp. 542-553.
13. Lagarias, J. C. The 3x+ 1 problem and its generalizations. *The American Mathematical Monthly*, **1985**, *92(1)*, pp. 3-23.
14. Lagarias, J. C. The 3x+1 problem: an overview. In *The Ultimate Challenge: The 3x+ 1 Problem*, J. C. Lagarias, American Mathematical Society, **2010**, pp. 3-30, ISBN 978-0821849408.
15. Minsky, M.L. Recursive unsolvability of Post's problem of "tag" and other topics in the theory of Turing machines. *Annals of Mathematics*, **1961**, *74(3)*, pp. 437–455.
16. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system, **2008**.
17. Nichols, D. Analogues of the 3x + 1 Problem in Polynomial Rings of Characteristic 2. *Experimental Mathematics*, **2018**, *27(1)*, pp. 100-110.
18. Oliveira e Silva, T. Empirical verification of the 3x+ 1 and related conjectures. In *The Ultimate Challenge: The 3x+ 1 Problem*, J. C. Lagarias, American Mathematical Society, **2010**, pp. 189-2017, ISBN 978-0821849408.
19. Percival, C. Stronger key derivation via sequential memory-hard functions **2009**.
20. Terras, R. A stopping time problem on the positive integers. *Acta Arithmetica*, **1976**, *30(3)*, pp. 241-252.
21. Tromp, J. Cuckoo cycle: a memory bound graph-theoretic proof-of-work. In *International Conference on Financial Cryptography and Data Security*, **2015**, Springer, Berlin, Heidelberg, pp. 49-62.
22. Urvoy, T. Regularity of congruential graphs. In *Mathematical Foundations of Computer Science 2000. MFCS 2000. Lecture Notes in Computer Science, vol 1893*, Nielsen M., Rovan B. (eds), **2001**, Springer, Berlin, Heidelberg.
23. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, **2014**, *151*, 1-32.