A New Proof of Work for Blockchain Based on Random Multivariate Quadratic Equations

Jintai Ding(✉)

University of Cincinnati, Cincinnati, USA
Jintai.Ding@gmail.com

Abstract. In this paper, we first present a theoretical analysis model on the Proof-of-Work (PoW) for cryptocurrency blockchain. Based on this analysis, we present a new type of PoW, which relies on the hardness of solving a set of random quadratic equations over the finite field GF(2). We will present the advantages of such a PoW, in particular, in terms of its impact on decentralization and the incentives involved, and therefore demonstrate that this is a new good alternative as a new type for PoW in blockchain applications.

Keywords: Proof-of-Work · Multivariate · Quadratic · NP-hard · Decentralization · Blockchain · Cryptocurrency

1 Introduction

The idea of Proof-of-work was invented in 1992 by Dwork and Naor [12] and it was initially invented to combat the spam attacks and the denial of service attacks by making such attacks economically unviable. Proof-of-Work can be defined as a protocol, which requires certain (from minimum to maximum) amount of computations in order to finish a task. The computation performed should produce something, which can be used to verify that the required amount of computations are indeed accomplished. The concept of Proof of Work has since found application. In 1997, Adam Back invented a protocol: HashCash, where Proof-of-Work is built upon a hash function. The term "PoW" was coined by M. Jakobsson and A. Juels later.

In October of 2008, Satoshi Nakamoto published his Bitcoin whitepaper [16], where Proof of Work is a key element of the Bitcoin protocol. The white paper stated that

"We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work."

Satoshi cited the work of HashCash. Therefore, however, the application of the PoW in Bitcoins serves a purpose very different from the original intention.

J. Zhou et al. (Eds.): ACNS 2019 Workshops, LNCS 11605, pp. 97–107, 2019.
https://doi.org/10.1007/978-3-030-29729-9_5

A key innovation in Bitcoin is that it uses a Proof of Work to build a competitive process called mining, which, with the help of digital signature system, help to solve three key problems:

1. prevention of the unlawful modification of the record or high cost of unlawful modification of the record;
2. synchronization of a decentralized system.
3. prevention of double spending;

In terms of our understanding, the main function of PoW is actually the property (1) and (2), since double spending is actually detected through the digital signature and therefore the stability of the blockchain due to the functionality of PoW enable us to easily solve the problem of double spending. Therefore, in bitcoin, Proof of Work systems is used to provide stability and security to an entire decentralized network, where we do not request trust on any speciifc player but the trust of the overall whole set of players. Proof of Work is used mainly to build a stable consensus mechanism, namely if there are enough mining nodes participating to perform the PoW, then the computational PoWer needed to control or attack the network becomes unattainable for any single entity. Also mining was really a great promotion tool to bring people participating in the process to therefore build a truly trustworthy decentralized system. However, as we all know, due to the appearance of the ASIC mining machines, the mining PoWer is increasingly controlled by big business players, and it does not make any sense for an ordinary user to doing mining on his or her PC on the side.

Later people invented many new PoW algorithms for various new altercoins, which we will not mention here, since none of them give a solid scientific base for their choices. In this paper, what we would like to do is to perform a complete theoretical analysis of PoW in the context of PoW for cryptocurrency, namely what are the theoretical properties we would like to have for a good PoW for a cryptocurrency. There is some initial work done before [15] in this direction but it is rather incomplete.

Then we will present our new PoW system, which is based on solving a set of random multivariate quadratic equations over GF(2), and we will show advantages of such a PoW system.

Remark 1. Here we would like to remark that the new PoW in bitcoin invented something that we humans never had before namely we can produce a document, which can be destroyed but can not be altered without incurring a tremendous cost. From this perspective, we believe PoW is much better choice than Proof of Stake (POS), since in a POS system, if someone controls the system, they can do whatever they want with essentially no cost, but in the case of PoW system for bitcoin, even if someone controls the system, if they want to do something illegitimate, to change the record, they still must pay a high prize, which is the best deterrence against the corruption of the system.

2 Theoretical Analysis of PoW in Bitcoin and Its Theoretical Model

The definition of PoW first requires certain amount of computations in order to finish a task and the computation performed should produce something, which can be used by anyone to publicly verify that the required amount of computations are indeed accomplished. From the mathematical perspective, we can view PoW as a task to solve certain mathematical problem, which can be verified by anyone. This requires clearly two properties of this mathematical problem:

1. This mathematical problem requires a certain amount computation (or a certain level of computational complexity) and no one should be able (easily) to find a new way to solve the problem that substantially reduce the computation complexity, which otherwise allow certain people to cheat. We call such a property the intrinsic hardness (**IH**) property of the problem.
2. Anyone can easily verify the solution is indeed a solution, We call this the solution public verifiability (**SPV**) property of the problem.

The IH property clearly indicates that we should use some problem that is a historicalluy very well-studied hard problem and we know very well computation complexity to solve this problem. Clearly in the case of bitcoin, Satoshi chose a very good problem, which we call the partial invertibility (PI) problem of hash functions. A simplified way to define the problem can be presented as the problem to find a string x of fixed length, such that

$$H(B, x) = (0, 0, ..., 0, *, ...*),$$

where B is the block to be mined, H is a Hash function and $*$ means values we do not care. Namely the problem is to find a preimage of any element whose first fixed number of entries are zeroes. This problem is hard to solve due to the Non-invertibility property of the hash functions, namely we can not find the preimage of a Hash function for a randomly chosen elements in the image space.

But one thing he missed, we believe, is that he did not expect that the ASIC machines can gain so much advantage compared with our ordinary PCs (scale of million), which, in some way, causes some kind of centralization of the mining PoWer being in the hands a few big miner and mining pools. However from the recent history of development of Hash function, we should also know this problem is not one of the historically well-studied problem since hash functions has a short history, and as we know the Non-invertibility hash function (like MD5) can be broken [11,18].

In the case of bitcoins, since it is a decentralized system, it means the mining problem that needs to be solved for each block in any node can be easily set up by anyone who has the information about the block, and the specific problem itself can also be easily verified. We call this the easy set-up and public verifiability (**ESUVP**) property of the problem. Since the Hash function is standardized and widely used, it clearly satisfies this property.

Also we should require that one can not easily find a block such that the problem associated with that block is substantially easier than the usual mining problem. This implies that in this family of the mathematical problems, it is computationally impossible to find a problem which is substantially easier than the hardest cases. We call such a property the homogenous hardness (**HH**) of the problem. We can see that if the problem is not HH, it is possible for someone to find an easier case and mine on such a case to gain advantage over others. In the case of bitcoin, it means that we can not easily find a meaningful B (a valid new block) such that it is easy to find x such that

$$H(B, x) = (0, 0, ..., 0, *, ...*).$$

Surely one may say that that if we find already a solution for the problem, $H(B, x) = (0, 0, ..., 0, *, ...*)$, we can repartition (B, x) to derive another solution, but this is impossible due to the fact that we want x to be of the fixed length.

In the case of bitcoin, to ensure the timing of mining of each blocks to be stable, the hardness of the problem of mining is adjusted accordingly if the mining time is substantially higher or lower than 10 min. This means we can actually adjust hardness in a controlled manner. In this case of bitcoin, the hardness is adjusted by increasing or decreasing the number of bits of zeroes, namely by increasing or decreasing the number of zeroes in the R.H.S. of

$$H(B, x) = (0, 0, ..., 0, *, ...*).$$

The hardness is basically either doubled or halfed when we increase or decrease the number of zeroes of the Hash image. In the case of Bitcoin, though we claim we have more precise adjustment, the nature of searching algorithm decides that the mining time has substantial fluctuations. Surely it will be much better to have ways to make more precisely controlled, for example, reduce hardness by a fixed percentage, however such a problem is clearly not easily to find. We will call this the difficulty adjustability (**DA**) property [15].

One more thing the bitcoin system wanted is that it should be a decentralized system that anyone can participate and make meaning contributions in the mining process. This means that the algorithm to solve the problem can be distributed independently to many independent users. We call this the independent distributability (**ID**) property of the problem. In the case of mining for bitcoin, we actually do a brutal force search on x for all $H(B, x)$, which can be easily distributed. If a problem does not have this property, then the participants will be very limited and it will not be a true decentralized system.

One more key property for PoW problems is that we should make sure that the work done for one block can not be reused substantially for another block, otherwise it will make it very hard to control the hardness since anyone can gain advantage in mining or even performing attacks. One such extreme situation is that, for any mined block, if someone can easily find a new block, which is associate to exactly the same hard problem and therefore has the same solution, then miners could reuse this problem and the solution to cheat or perform an attack by replaced published blocks with new and different blocks. In general,

it is better to have the property that the mathematical problem associated with a given block to be uniquely determined, and for any given two blocks whose associated problems are totally independent and therefore no work done in one problem can be reused in another. Overall, the key is that work done for one block can not be (meaningfully) reused for another block, and this is called non-reusability (**NR**) properties [15].

Overall our conclusion is that a good PoW problem should have the following properties:

1. the intrinsic hardness (**IH**) property,
2. the solution public verifiability (**SPV**) property,
3. the easy set-up and public verifiability (**ESUVP**) property,
4. the homogenous hardness (**HH**) property,
5. the r difficulty adjustability **DA**) poperty,
6. the independent distributability (**ID**) property.
7. the non-reusability (**NR**) property.

From the above analysis, we can see that it is not easy to find such a problem, in particular, the properties IH and HH. It is clear to us many of the new PoW algorithms in altercoins are very risky choices since we know not so much about the hardness of these problems. Also it is very clear that Satoshi made a very good choice with what we knew at the time.

In [15], Kim presented a new PoW where he pointed out clearly the NR and DA property, but he did not state clearly the rest of the properties, in particular, the ID property. He presented a new POW using prime numbers, but it is not clear that his PoW satisfies all the properties we have here and we will address it in a subsequent paper.

3 A New Proof of Work Based on Random Multivariate Quadratic Equations

In this section, we propose a new idea to build a new family of PoW algorithms. The basic objects of this section are systems of quadratic polynomial equations in several variables over a finite field $\mathbb{F} = \mathbb{F}_q$ with q elements (see equation (1)).

$$\begin{aligned}
p^{(1)}(x_1,\ldots,x_n) &= \sum_{i=1}^{n}\sum_{j=i}^{n} p_{ij}^{(1)} \cdot x_i x_j + \sum_{i=1}^{n} p_i^{(1)} \cdot x_i + p_0^{(1)} \\
p^{(2)}(x_1,\ldots,x_n) &= \sum_{i=1}^{n}\sum_{j=i}^{n} p_{ij}^{(2)} \cdot x_i x_j + \sum_{i=1}^{n} p_i^{(2)} \cdot x_i + p_0^{(2)} \\
&\vdots \\
p^{(m)}(x_1,\ldots,x_n) &= \sum_{i=1}^{n}\sum_{j=i}^{n} p_{ij}^{(m)} \cdot x_i x_j + \sum_{i=1}^{n} p_i^{(m)} \cdot x_i + p_0^{(m)}.
\end{aligned} \tag{1}$$

A well-known hard mathematical problem is the MQ Problem:

Problem MQ: Given m quadratic polynomials $p^{(1)}(\mathbf{x}), \ldots, p^{(m)}(\mathbf{x})$ in the n variables $x_1, \ldots, x_n$ as shown in equation (1) with the coefficients of the equations uniformly and independently random chosen, find a vector $\bar{\mathbf{x}} = (\bar{x}_1, \ldots, \bar{x}_n)$ such that

$$p^{(1)}(\bar{\mathbf{x}}) = \ldots = p^{(m)}(\bar{\mathbf{x}}) = 0.$$

The MQ Problem is proven to be NP hard including quadratic polynomials over the field GF(2) [13] when m and n are of roughly the same size and this problem is known to be hard on average. In this paper we will now only concentrate on the case of GF(2).

The hardness of the MQ problem is the security foundation of the multivariate public key cryptosystems [6], a new family of post-quantum cryptosystems that can resist quantum computer attacks. Several such algorithms including Rainbow signature [9] are selected in the second round of the National Institute of Standardization of Technology post-quantum crypto standardization process [17].

The idea of using MQ problem for PoW is inspired by the work [10], where the MQ problem is used to build a Hash functions. However that Hash function has some security issues due to the collision resistance but **NOT** non-invertibility.

We will now present the construction of the new PoW algorithm based on the MQ problem.

First we will select

$$n = m + 8,$$

namely we will have 8 more variables than the number of equations.

Let us now count the number of bits of coefficients of the all the polynomials $p(i)$, which is

$$N = m \times (n(n-1)/2 + n + 1) = (n-8)(n^2/2 + n/2 + 1).$$

Suppose that we have a block B which needs to be mined. The node will then compute

$$h_i = H^i(B) = H(H(..H(B))),$$

for $i =, 1, 2..., \lceil N/256 \rceil$, if H is a 256 bit Hash function (or $i =, 1, 2..., \lceil N/512 \rceil$, if H is a 512 bit Hash function).

Then we will use h_i one by one to assign them in a fixed order to be the coefficients of the multivariate polynomials $p^{(j)}(x_1, \ldots, x_n)$, and dump any leftover bits.

Then the mining task is to find a vector

$$\bar{\mathbf{x}} = (\bar{x}_1, \ldots, \bar{x}_n)$$

such that

$$p^{(1)}(\bar{\mathbf{x}}) = \ldots = p^{(m)}(\bar{\mathbf{x}}) = 0.$$

Here we would like use the random oracle mode [2] to claim that we can view these polynomials are indeed random polynomials since we can treat a hash function as a random oracle.

Practically, the first question one may ask is that if there is indeed a solution. The answer is positive with extremely high probability.

Theorem 1. *For a MQ map with m random polynomials and $m+8$ variables, the probability that*

$$p^{(1)}(\bar{\mathbf{x}}) = \ldots = p^{(m)}(\bar{\mathbf{x}}) = 0$$

has no solution is approximately e^{-256}, when m is big enough.

Here we will give a sketch of proof assuming the that the quadratic map is a random function. Assume that we have a map from $GF(2)^{m+8}$ to $GF(2)^m$, due to the randomness of the hash functions, we can assume that this is a random functions. Then this becomes a map from a set of size 2^{m+8} to a set of size 2^m. We would like to find out the probability that there is no vector being mapped to $(0, \ldots, 0, 0)$. It is easy to see that the probability is given as

$$(1 - 2^{-m})^{2^{m+8}} = ((1 - 2^{-m})^{2^m})^{2^8}.$$

We know that if m is big enough (like 30), we have that

$$((1 - 2^{-m})^{2^m}) \approx e^{-1}.$$

Therefore the probability will be roughly

$$((1 - 2^{-m})^{2^m})^{2^8} \approx e^{-256}.$$

This more or less guarantees that a miner can always find a solution.

Remark 2. In Bitcoin there is also no guarantee that for every block B there is a string x such that H(B——x) is of the required form. However, each miner works on a slightly different block (with different transactions) and every block in Bitcoin also contains a so called coinbase transaction (this transaction contains the Bitcoin address of the miner who wants to earn the reward money of creating the new block as well as a nonce randomly chosen by the miner.) Therefore, there exist many slightly different versions of the search problem which are worked on in parallel by the different miners. One can be relatively sure that some of these problem is actually solvable.

From the construction, it is clear that the solution public verifiability (**SPV**) property and the easy set-up and public verifiability (**ESUVP**) property are satisfied.

Due to the usage of hash functions, which can be viewed as a random oracle, to produce the problem, we can see that two different blocks can never have the same mining problems due to the collision resistance of hash functions and they should be essentially independent of each other. There the property of **NR** is satisfied.

Let us then look that the intrinsic hardness (**IH**) property, which is actually guaranteed by the NP-Hardness of the random MQ problem, and similarly we can promise the homogenous hardness (**HH**) property due to again the NP-Hardness of the random MQ problem.

Here we would like to remark that the hardness of partial inversion of a hash function in general has a much higher theoretical risk in the sense we can not actually prove the hardness of such a problem for the existing hash function, which is evident by the fact that in the past hash functions like MD5 were broken pretty badly.

Now, we would like to discuss the independent distributability (ID) property of the MQ problem. In terms of human history, more than four thousand years ago, Babylonian mathematicians could solve the problem of single variable quadratic equations. For multivariate quadratic equations, the first "smart" algorithm appeared in 1965 by Buchberger [4]. However for the problem we proposed for mining, the state of the art of the best algorithm is actually again the brutal force algorithm, which is clearly indicated in the Fukuoka MQ challenge, where there is a public challenge to find solutions for such problems (https://www.mqchallenge.org).

Due to the situation above, we also know that we can easily adjust the difficulty of the problem for each block by adjusting the number of equations where adding one more equation and one more variable means essentially doubling the hardness and reducing one equation and one variable means halfing the hardness. Therefore our PoW satisfies the **DA** property.

In addition, the new mining algorithm has other clear advantages.

1. The first one is the property we call ASIC resistance. Namely due to the following properties:
   - the simplicity of calculating the value of multivariate quadratic polynomials, which involves **very few** number of simple addition and multiplication in GF(2) after checking the first element, by this we mean that in the fast implementation of multivariate quadratic polynomial solving, we use the so called Gray code, where we search solutions in the order that each time we check if a new element is indeed a solution, it has only one bit difference from the previous element checked, to speed up tremendously the computation [3].
     More precisely, for a quadratic function $f(x_1, ..., x_n)$ over GF(2), it can be written as

$$f(x_1, ..., x_n) = \sum_{i<j} a_{ij} x_i x_j + \sum b_i x_i + c.$$

     We will look at the case where there only 1 bit change on x_1. Assume that we already know the value of $f(x_1, ..., x_n)$ and we would like to calculate $f(x_1 + 1, ...x_n)$. We know that

$$f(x_1+1,...x_n) - f(x_1+1,...,x_n)$$
$$= \sum_{j>1} a_{1j}(x_1+1)x_j + b_1(x_1+1) - \sum_{j>1} a_{1j}(x_1)x_j + b_1(x_1)$$
$$= \sum_{j>1} a_{1j}x_j + b_1.$$

This means that

$$f(x_1+1,...x_n) = f(x_1+1,...x_n) + \sum_{j>1} a_{1j}x_j + b_1.$$

Therefore we only need to calculate

$$\sum_{j>1} a_{1j}x_j + b_1$$

to derive the value of $f(x_1+1,...x_n)$ from the value of $f(x_1,...x_n)$. Similar formula applies to the case of change of value of any variable x_i.

$$f(x_1,.,x_i,..,x_n) = f(x_1,.,x_i,..,x_n) + \sum_{i\neq j} a_{ij}x_j + b_j.$$

For such a simple calculation, GPU can achieve already great efficiency that ASIC should not be able to improve too much, while this is not the case at all in the case of PoW using hash functions.

- the polynomials for each block actually changes all the time (almost like we use a different hash function for every block), which make it hard and costly for ASIC implementations.

we do not think that ASIC implementation can gain that much advantage compared to the usual GPUs. Surely this is based on our current technology. We believe that ASIC surely will gain advantages but it will not be more than a scale of 10 while the advantage of the case of bitcoin is about the scale of 5000. Due to the high initial cost in ASIC production, we think this design should greatly discourage the development of ASIC machines for do such a mining and therefore make it viable for small miners to do mining independently.

2. There is some work recently on attacks on PoW using quantum computers [1,8,14] due to the large key sizes of the coefficients of the MQ polynomials, the MQ-based PoW will be much harder to attack using quantum computers since it will require much more qubits for finding solutions and each time a different new set of multivariate quadratic functions has to be reloaded into the quantum system.
3. Due to the long history (thousands of years) of study of solving polynomials equations and the NP hardness of the problem, we expect that to attack our new PoW is much harder.
4. In the case of mining in bicoins, the mining is used solely to support the decentralized network. But in the MQ-based mining, the system actually rewards any progress made on solving a NP-hard problem, which is a much

more meaningful task compared to the case of hash-based mining. For example, many of the problems to attack various cryptosystems, namely many of the cryptanalysis problem, can be reduced to solving a hard seeming random quadratic systems over GF(2) and any breakthrough in this area could have tremendous effect in cryptography.

The new ideas presented in this paper is already implemented in a new cryptocurrency called ABC (www.ABCMint.org) [5,7] and it has worked very well. The research work in this paper was essentially finished in 2017. The public chain for ABC was launched on June 18, 2018. However this paper is the first publication to explain exactly the mathematics theory behind the PoW in ABC.

Surely there are many well-studied NP-hard problems, for example, the shortest vector problem (SVP) for a lattice. However if we want to use the SVP problem for mining, it is not a good choice due to the property SPV, and ESUPV. It is very hard to set up a SVP problem such that everyone can publicly verify, it is indeed a hard problem and no one can cheat in the set up process. This is why, unlike the MQ challenges, it is hard to set up pubic SVP challenges. Namely if it is indeed a random lattice, it is very hard to verify a given vector is indeed a shortest vector or not. Therefore it is actually better to use NP-complete problems, where we can easily verify the answers. A good example of a NP-complete problem is the Knapsack problem, but the Knapsack problem does not satisfy the HH property while the hardest cases of Knapsack problem is very hard but a random case is often easy to solve. To build the hard for PoW for cryptocurrency is not an easy task. From what we know by now, we believe nearly all the new PoW algorithms for altercoins needs much more careful scrutiny and they all look rather risky.

4 Conclusion

It is clear that PoW in bitcoin is different from usual PoW used in other applications and requires additional properties. We present a theoretical study of those properties required and propose a new PoW algorithms based on the MQ problem an NP-hard problem. We show the advantages of this new PoW. We hope to use this work to stimulate the research direction in PoW for cryptocurrencies.

Acknowledgment. We would like to thank Johannes Buchmann, Albrecht Petzolt, Lei Hu, Hong Xiang, Peter Ryan, Tsuyoshi Takagi, Antoine Joux, Ruben Niederhagen, Chengdong Tao, Chen-mou Cheng, Zheng Zhang, and Kurt Schmidt for useful discussions. We would like to thank the anonymous referees for useful comments. We also would like to thank the ABCMint Foundation, in particular, Jin Liu for support.

References

1. Aggarwal, D., Brennen, G.K., Lee, T., Santha, M., Tomamichel, M.: Quantum-proofing the blockchain. Quantum attacks on Bitcoin, and how to protect against them. arXiv:1710.10377 (2017)

2. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security, CCS 1993, pp. 62–73. ACM, New York (1993)
3. Bouillaguet, C., et al.: Fast exhaustive search for polynomial systems in $\mathbb{F}_2$. In: Mangard, S., Standaert, F.X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 203–218. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15031-9_14
4. Buchberger., B.: Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. Ph.D. thesis, Innsbruck (1965)
5. Ding, J.: Quantum-proof blockchain. In: ETSI/IQC Quantum Safe Workshop 2018 (2018). https://www.etsi.org/events/1296-etsi-iqc-quantum-safe-workshop-2018#pane-6/
6. Ding, J., Gower, J.E., Schmidt, D.S.: Multivariate Public Key Cryptosystems. Springer, Boston (2006). https://doi.org/10.1007/978-0-387-36946-4
7. Ding, J., Liu, J.: Panel on quantum-proof blockchain. Money20/20 Hanzhou China (2018). https://www.money2020-china.com/portal/index/people/id/247.html
8. Ding, J., Ryan, P., Sarawathy, R.C.: Future of bitcoin (and blockchain) with quantum computers. Preprint of University of Cincinnati, 10.2016. Submitted to Bitcoin 2017 under Financial Cryptography 2017
9. Ding, J., Schmidt, D.: Rainbow, a new multivariable polynomial signature scheme. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_12
10. Ding, J., Yang, B.-Y.: Multivariates polynomials for hashing. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) Inscrypt 2007. LNCS, vol. 4990, pp. 358–371. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79499-8_28
11. Dobbertin, H.: The status of MD5 after a recent attack. CryptoBytes (2016)
12. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_10
13. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York (1979)
14. Gheorghiu, V., Gorbunov, S., Mosca, M., Munson, B.: Quantum-proofing the blockchain, November 2017. https://www.evolutionq.com/assets/mosca_quantum-proofing-the-blockchain_blockchain-research-institute.pdf
15. Kim, S.: Primecoin: cryptocurrency with prime number proof-of-work, March 2013. assets.ctfassets.net
16. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system, October 2008. academia.edu
17. NIST. Post-quantum cryptograhic standardization, January 2019. https://www.nist.gov/news-events/news/2019/01/nist-reveals-26-algorithms-advancing-post-quantum-crypto-semifinals
18. Sasaki, Y., Aoki, K.: Finding preimages in full MD5 faster than exhaustive search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_8