[[1]](#footnote-1)

New Class of Cryptocurrency Proof-of-Work Algorithms

First A. Author, *Fellow, IEEE*, Second B. Author, and Heung-No Lee, Member, IEEE

*Abstract*—These instructions give you guidelines for preparing papers for IEEE Transactions and Journals*.* Use this document as a template if you are using Microsoft *Word* 6.0 or later. Ensure that your abstract reads well and is grammatically correct.

*Index Terms*—Enter key words or phrases in alphabetical order, separated by commas. For a list of suggested keywords, send a blank e-mail to [keywords@ieee.org](mailto:keywords@ieee.org) or visit <http://www.ieee.org/organizations/pubs/ani_prod/keywrd98.txt>

# INTRODUCTION

## Blockchain and bitcoin

- 요소: Block(transaction, header ~), miner, PoW

- Bitcoin PoW의 양

The amount of work is 16 years of work stored in a single block of bitcoin blockchain. That is, the current hashrate required for the bitcoin network is 30 Exa hashes/sec. One block time is 10 mins which is 600 secs. 30 Exa hashes/sec \* 600 secs. The required hash cycle to mine a block is ~~ cycles. For example, if one uses a 10 Tera-hashes/sec miner, it takes 30 Exa / 10 Tera = 3\*10^6 miners. This is a number of miners we need to mine a single block within 10 mins.

- What is PoW?

The amount of work needed to mine a single block is defined as the amount of Proof-of-Work (APoW). PoW is defined as the process of providing a proof by miner which has successfully solved a crypto-puzzle. For a particular blockchain system, a crypto-puzzle is given.

## Contributions

New class of crypto puzzle을 만들고, based on error-correction coding scheme which satisfies all the crypto puzzle properties with addition to time variance property

## 활용방안

Usefulness 여러 개의 crypto puzzle을 연속적으로 만듦 -> 블록체인을 여러 개 만들 수 있음.

# Literature survey

## Requirements of hash function

Cryptographic hash functions take an input of arbitrary length and produce an output of a fixed length. The output is called message digest, hash, or fingerprint of the input. Given a function *h*:X→Y, the function *h* is cryptographically secure hash function when it satisfies such requirements:

(*One-way function*) If  it is computationally infeasible to find a value , s.t. .

(*Weak collision resistance*) Given a particular message digest, it should be very difficult to find an input that has the same message digest. That is, given , it is computationally infeasible to find an input value , s.t.  and .

(*Strong collision resistance*) It should be very difficult to find two inputs that have the same message digest. That is, it is computationally infeasible to find two distinct values , s.t. .

## Hash functions of current PoW algorithms

(One way)

(Collision almost free)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Hash function | One-way | Weak collision | Strong collision |
| Bitcoin | SHA256 |  |  |  |
| Ethereum | Ethash |  |  |  |
| Litecoin | Scrypt |  |  |  |
| Monero | CryptoNight |  |  |  |
| DASH | X11 |  |  |  |

# Problem formulation

Goal is to develop a new class of PoWs (Block dependent PoW)

## Necessary conditions for any valid puzzles

One way

No attack

No cheating

Adjustable difficulty

Openness

They are not sufficient enough variations.

One way is to diversify the number of puzzles and change the puzzle over time.

## Necessary conditions for good puzzle

Goodness

(Time variant) Provable? 매 블록마다 previous hash 값에 의해서 문제가 변형되는 것.

( ) 다른 miner들이 변형된 문제를 받아 들이고 풀 수 있는가.

Time variant -> independent PoW -> Infinite cryptocurrency 가능

Scaling Problem 해결 가능

용어 hash algorithm 으로 통일 functio이 아님

TABLE I. Pseudo code

|  |  |
| --- | --- |
| Notation | Definitions |
| **Step 1** | The previous block hash |
| **MaxIter** | The maximum number of iteration of the decoding function. |
| **I***n* | The *n* × *n* identity matrix |
| **O***n* | The *n* × *n* matrix of zeros |
| **f**(*i*) | The *i*th element of a column vector **f** |
| **f**[*i*:*l*] | The column vector constructed by collecting elements of a column vector **f** from the *i*th element to *l*th element |
|  | The matrix is constructed by collecting columns of **F** corresponding to indices of a set |

SHA => Secure Hash Algorithm

# Proposed solution (theorem)

Hash-Decoder Theorem. The defined decoder satisfies all PoW requirements. Thus, Legitimate PoWs. DEC (crypto (nonce, S), F), Proposed MAP(nonce, S,F) = estimate\_c.

Our aim is to introduce a new class of PoW algorithms..

What is the content of the proof that a prover need to report for verification? – Nonce, number of iterations

Hash functions – SHA

# A specific proposed solution (example)

We aim to propose a new class of *PoW* problems. The idea behind this proposed new class is to combine SHA256 with one-way functions in inverse problems in which encoding is easily made, but decoding is time-consuming. These inverse problems generally include the error correction coding [A], sparse signal recovery [B] and image speech recovery [C]. Among these above problems, we select the problem of error correction coding (ECC) in which there is an encoder-decoder pair. In ECC, the decoder aims to correct an error vector to find an original codeword vector, i.e.,



where **e** is an error vector and **H** is a parity-check matrix.

The idea proposed here is to use an output of SHA256 as an input of the decoder. The output of the decoder is used to determine whether a crypto puzzle problem generated by the proposed *PoW* is successfully completed or not. That is, we aim to find *Nonce* such that



where is a given condition set.

< 정답 조건 정의> - 작성 중

An example of H whose size is 8 by 16.



The idea proposed here is to use an output of the SHA256 function as an input of the decoder. The output of the decoder is used to determine whether a crypto puzzle problem generated by the proposed *PoW* is successfully completed or not. That is, we aim to find *Nonce* such that Geometrical explanation.

<Decoder> 정의 - 작성 중

DEC Definition. Report the first codeword found in a given number of iterations. What is the number of iterations that the puzzle has to give? No need to give the number of iterations? Let us include the maximum number of iterations. Why? Without the maximum number of iterations defined, the verifier will have a difficulty if the proof given by the prover is true or not. What happens when the verifier fails to find a codeword within the maximum number of iterations? A miner has an option to change the merkle tree, i.e., order of transactions, addition of transactions, timestamps.

<작성 완료된 부분>

In the proposed frameworks, we adopt the message passing algorithm as our decoding function. The details regarding this algorithm are well-known in the literature. In this manuscript, we change its termination condition.

In general, the message passing algorithm is terminated after the iteration passes a given number, i.e., MaxIter. If we use only this termination condition, verifiers conduct the algorithm until the iteration passes MaxIter. This can be inefficient, i.e, the message passing algorithm can find a codeword within in a small number of iteration. Hence, we instantly terminate this algorithm once it finds a codeword. Then, we emphasize that a miner reports the number of iterations required for finding a codeword to verifiers. These verifiers run the algorithm during the given number of iteration.

So far, we have overall explained the proposed frameworks. We have defined the decoder and given the decoding algorithm and its termination conditions. Indeed, we have explained how we construct the parity check matrix depending on the previous hash value.

Now, we aim to show the proposed framework can satisfy the conditions discussed in the subsection X. For this purpose, let us define the following definition.

**Definition 1**: Let *h* be the number of cycles for conducting the SHA256 hash function. Let *p* be the successful probability that a given error vector satisfies a given condition set. Let *d* be the number of cycles cost for conducting the decoding function.

Based on Definition 1, we show the one-wayness property of the proposed framework as follows:

**Proposition 1**: The proposed *PoW* framework has the one-way property, i.e., a puzzle should be time-consuming to solve but very easy to check.

**Proof**: The number of total cycles required by a miner is



and that required by a verifier is



It should be noted that for any values of *p*, we always have



which implies that the times for mining a block is larger than that for verifying a mined block which completes the proof.

Next, we suppose that there are malicious miners who aim to deceive verifiers. As an example, they rely on chances to mine a block by reporting a previous solution. Also, another malicious miners aim to mine a block without actually running either SHA256 or the decoding function. We prevent the presence of these miners, as shown in Propositions 2 and 3, respectively.

**Proposition 2**: In the proposed *PoW* frameworks, any solutions to previous crypto puzzle problems cannot be reusable.

**Proof**: The found value of nonce is dependent upon each block header. If the block header is changed, the value of nonce has to be found again. Thus, no one can use the previous nonce value unless the current block header is the same to the previous one.

**Proposition 3**: In the proposed *PoW* frameworks, every miner must solve a crypto puzzle problem for mining a block.

**Proof**: First, SHA256 is known to be a one-way function [A]. No one can expect a hash value generated by this function even its input can be known in advance. It is concluded that SHA256 is conducted. Next, a parity check matrix used in the decoding function changes depending on the previous hash value. Indeed, its size is large. No one makes a look-up table for the decoding function a priori. It is also concluded the decoding function has to be conducted.

Next, we present Proposition 4 that shows that the difficulty of the proposed framework is adjustable

**Proposition 4**: In the proposed *PoW* frameworks, the difficulty can be adjustable.

**Proof**: As we have stated, a codeword yielded by the decoding function can be considered as a correct solution if the hamming distance of this codeword is less than a given number *l*. Then, let  be a set of codewords with *l* hamming weight. For a fixed number *l*, the successful probability is defined as follows:



If we change *l*, the successful probability *p* changes. Therefore, the average iteration for solving a crypto puzzle problem also changes. This is concluded that the difficulty can be adjustable depending on the value of *l*.

So far, we have shown that the proposed frameworks satisfy the four necessary conditions defined in the subsection X.Y. What we show is now that the proposed frameworks can satisfy the time-variant condition defined in the subsection X.B. This can be shown in Proposition 5 as follows

**Proposition 5**: A crypto puzzle problem in the proposed frameworks is time-variant.

**Proof**: A parity check matrix is constructed using the previous hash values and is used to formalize a crypto puzzle problem. Thus, the *i*th crypto puzzle problem solved for mining the *i*th block can be completely different to that solved for mining the *j*th block. This is concluded that our crypto puzzle problem is time-variant

[A] G. C. Kessler, An overview of cryptography, published in Handbook on Local Area Networks, 1998, Available at: <http://www.garykessler.net/library/crypto.html>.

# Simulation results

String – Output

## One way

A puzzle should be difficult to solve but very easy to check.

Change the code parameters and show one-wayness of the proposed PoW. 푸는데 걸린 시간, 체크하는데 걸린 시간, Nonce를 2^32 까지 돌렸는데, codeword를 못 찾는 경우 (set size 를 바꿔가면서 condition check를 바꿨을 때, 결과가 어떻게 나오는지).

## No cheating

A puzzle should be resistant to cheating. Proof reuses shall be detected and preventable. - Verifiability

Given F, is decoder output unique for all e?

Given F and e, is decoder output reproducible for all other nodes?

이전 블록의 답을 현재 블록의 답이라고 주장했을 때, 검증자가 쉽게 찾을 수 있는지

임의의 단계에서 변조가 일어났을 때, 다음 블록들의 답이 달라져야 함

Miner-Verifier relation. Need to be discussed. Why? 부당한 이득을 miner가 챙겨갈 수 있음. - How? 이렇게 저렇게 할 수 있음. 이런 문제를 막아야함. 제안하는 puzzle은 이 문제를 막을 수 있음. Why? - ~~ 때문에. 시뮬레이션 결과를 제시.

## Adjustable difficulty

Puzzle difficulty should be adjustable. Set size, block length, M, N을 바꿨을 때, difficulty level (hash cycle, expected hash power needed)

난이도 조절을 얼마나 미세하게 할 수 있는지

## Openess

Anyone with a cpu wishes to participate should be able to so.

# Conclusion

Impact of innovation

For ASIC resistant, directed acyclic graphs, use of multiple SHA funcitons, use of scripts. But they are not sufficit enough varaitions.

Acknowledgment

The preferred spelling of the word “acknowledgment” in American English is without an “e” after the “g.” Use the singular heading even if you have many acknowledgments. Avoid expressions such as “One of us (S.B.A.) would like to thank ... .” Instead, write “F. A. Author thanks ... .” In most cases, sponsor and financial support acknowledgments are placed in the unnumbered footnote on the first page, not here.

References

1. [↑](#footnote-ref-1)