

블록체인 기반 사용자 참여 게임 업데이트: 미로탈출게임 사례

전민규, 황지연, 나현숙
송실대학교 컴퓨터학부

{mkjyong, ghkdwldus98}@naver.com, hsnaa@ssu.ac.kr

Blockchain-Based User-Participatory Game Update: Maze Escape Game Case

Mingyu Jeon, Chiyeon Hwang, Hyeon-Suk Na
School of Computer Science and Engineering, Soongsil University

요 약

이 논문에서는 블록체인을 이용해 게임 변수들 (게임 난이도, 아이템 강화확률 등)을 사용자들이 직접 공정하고 투명하게 업데이트하는 게임 시스템을 제안한다. 일례로 소개하는 이더리움 기반 미로 탈출 게임은, 최초의 블록체인 기반 미로 게임이며 블록체인을 통해 난이도를 조절하는 최초의 시도이다. 이 시스템에서는 난이도 함수가 사용자들이 매매한 미로 벽의 수량을 반영해 난이도를 계산한다. 게임 관리에 블록체인을 사용하는 것은 인적 경제적 자원을 절약할 뿐 아니라, 난이도 및 아이템 강화확률을 일방적으로 조작하는 게임 개발자의 불공정한 관행도 방지된다. 누구도 비밀스럽게 미로, 승자, 난이도를 조작할 수 없기 때문이다.

ABSTRACT

In this note, we propose a game system to fairly and transparently control and update game variables using blockchain. As an example, we present an Ethereum-based maze escape game, being the first blockchain-based maze game and the first attempt to control game level through blockchain. In this system, the level function computes maze level reflecting the amount of maze walls purchased/sold by users. Using blockchain for game management saves human and economic resources and prevents unfair practice of game developers unilaterally manipulating game level or item enhancement rate.

Keywords : Blockchain(블록체인), Ethereum(이더리움), Maze Escape Game(미로탈출게임)

Received: Jun. 05. 2019 Revised: Jun. 27. 2019
Accepted: Jul. 16. 2019
Corresponding Author: Hyeon-Suk Na(Soongsil University)
E-mail: hsnaa@ssu.ac.kr

ISSN: 1598-4540 / eISSN: 2287-8211

© The Korea Game Society. All rights reserved. This is an open-access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0>), which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Introduction

Blockchain is a decentralized public ledger that allows all participants in the network to read freely and verify records independently. Due to the democratic and novel way of data management offering security and immutability, blockchain is predicted to be the core infrastructure for the fourth industrial revolution. But its technology is still in the incubation step in the sense that it is being tested for applicability and usefulness in a few industries. According to the State of the DApps[1], as of May 2019, the number of decentralized applications(abbreviated as dApps) is 2,667 and the number of daily active users is about a hundred thousands over all blockchain platforms. Statistics in the same site tell that the three most active categories of dApps in Ethereum are game, exchange of cryptocurrencies, and gambling, each dominating about 30% of all transactions in the network. Since game is the category that can serve users for non-financial and recreational purposes, blockchain developers have been interested in blockchain-based games. In Section 2.3 we summarize the status and limitations of current blockchain-based games.

In this paper, as another attempt to apply blockchain to the game, we propose a game system to fairly and transparently control and update game variables (game level, item enhancement rate, etc.) using blockchain. For illustration, we present an Ethereum-based maze escape game, which is the first blockchain-based maze game and the first attempt to control game level through

blockchain, to the best of our knowledge. In this system, the winner and maze information of each game is recorded in the blockchain, and its level adjusting algorithm computes the maze level reflecting the amount of maze walls purchased/sold by users in blockchain network. By using blockchain for game management, not only human and economic resources are saved, but also anyone as well as the developers cannot manipulate the winner, the maze, and the level in a secret manner, which prevents unfair practice of game developers unilaterally manipulating game level or item enhancement rate (in order to lead users to buy items). Moreover, as will be seen in Section 3.1(2), users' trading and play, motivated by economic interests and fun, respectively, take a key role in adjusting the level of mazes and the value of tokens (offered as escape-rewards) over time.

All the functionality of our system might be implemented in a centralized system of game developer's server in a more efficient and sophisticated manner; the winner, rewards, and the maze information can be announced for each game, and a level adjusting algorithm can control maze level in a more interesting and fascinating way. But, as long as these game information and variables are managed in the developer's server, even if they appear to be fairly controlled by user participation, the possibility of unilateral and secret manipulation of the developer cannot be ruled out. Therefore a purpose of this study is to seek for a method to guarantee fairness and transparency of game control and management with help of immutability of blockchain.

In Section 2, we briefly summarize the basic

concepts of blockchain technology and the status of current blockchain-based games, and in Section 3 describe our game system in details.

2. Blockchain and Game

2.1 Basic concepts of blockchain

Since every node in the network shares all entries in the blockchain, if a node wants to alter something in the database, it has to create and broadcast a so-called transaction which comes into effect through the following process: Transactions are bundled into a block by a miner, and as soon as the block is mined (registered into the network), the transactions therein are executed and shared by all participating nodes[2,3,4].

Every blockchain network needs to adopt a consensus algorithm (e.g. Proof of Work(PoW)[4,5], Proof of Stakes(PoS)[6]) for solving inherent problems of peer-to-peer network: two different blocks simultaneously mined at a distance (called a fork), or malicious double spending. With help of consensus algorithm, the network can share a linear sequence of blocks of transactions in time, called blockchain. A blockchain system can get network power and participants' trust only when it can solve successfully many issues including the above-mentioned problems, scalability, and security.

2.2 Ethereum

Ethereum is the most popular blockchain platform containing 93% of all dApps[1]. Such

popularity is due to its built-in programming language called Solidity[3]. This language is simple, and open to fully democratized and decentralized situations; programmers can create any smart contract or rules of ownership and transactions by writing a few lines of code.

A node participating in Ethereum, called Ethereum client runs the so-called Ethereum Virtual Machine(EVM) and copies all block information including accounts list, transactions, and smart contracts. So, once a smart contract is accepted in the blockchain, its binary code is copied to all nodes' EVM and lives forever in the network as an autonomous agent, being executed whenever invoked by a message or transaction and having control over its own Ether balance and key/value storage. Further details of Ethereum and its applications, refer to the white paper of Ethereum or Solidity documentation[2,3].

2.3 Blockchain-Based Games

Game is considered as one of the key measures of the potential for the development of blockchain technology. But blockchain-based games developed so far are in a very primitive and poor stage in terms of game genre and game function. According to Review of Dapp[7], as of May 2019, nearly two-thirds of the total 739 blockchain-based games are registered in Ethereum. Of the 485 Ethereum-based games, less than twenty games have over 15 active users within 24 hours, and only nine games listed in [Table 1] have over 100 active users within 24 hours.

[Table 1] Ethereum-based games with over 100 users within 24 hours[7](May 13, 2019)

Title	Genres	24H users
My Crypto Heroes	Collectible	2,145
OxUniverse	Collectible	940
HyperDragons	Collectible	466
Blockchain Cuties	Collectible	464
CryptoKitties	Collectible	260
Axie Infinity	SIM/SLG	226
CryptoDozer	Collectible	158
Chibi Fighters	SIM/SLG	142
Ethermon	SIM/SLG	137

As can be seen in [Table 1], most blockchain-based games developed so far are either Collectible or SIM/SLG games. ‘Collectible’ is a game to collect, crossbreed, and trade various items and virtual pets (each regarded as a unique coin) with value and scarcity. “CryptoKitties”[8] in the fifth row is one of the best-known among them, being released by Axiom Zen in November 2017. Smart contract of CryptoKitties generates and releases a new “Gen0” cat, called Genesis, every 15 minutes, and users can purchase, collect, crossbreed and sell them. ‘SIM/SLG’ games in the table are incremental or idle simulation games allowing one additional function: to train the pets (Axie, Chibi Fighter, Ethermon) by a simple action like clicking a button. In summary, most of blockchain-based games developed so far are of collectible games to collect, crossbreed, train and trade virtual pets and items. Blockchain-based game system uses blockchain as a means of recording the value/owner of coins (pets) and items, and smart contract as an automatic generator of new coin (pet).

Such poverty of blockchain-based games in

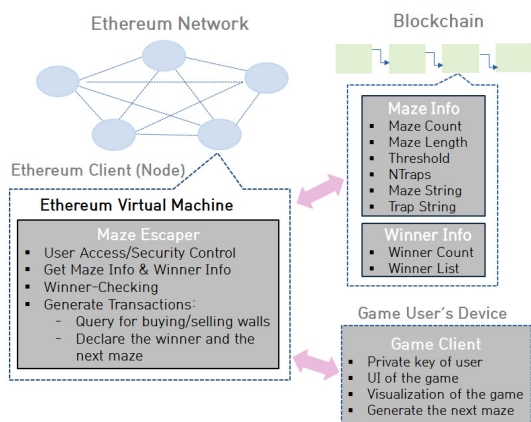
genre and functionality is due to immaturity of the blockchain technology and an inherent weakness of decentralized network to overcome; User’s trading of items and the ownership of pets can be validated by a transaction registered in a blockchain, which means that the system needs pay the transaction fee and wait until the block containing this transaction is mined, e.g., at least 10 seconds in Ethereum. Most efforts to overcome this status seem currently concentrated on developing better blockchain platforms: either being game-specific platform (e.g., Engine Coin for game items market and easy-to-use sdk) or guaranteeing higher speed (e.g., EOS, Loom network).

In the next section, we describe our attempt to apply blockchain to the game: a blockchain-based maze game system such that the winner and maze information of each game is recorded in the blockchain and the level of maze is adjusted by reflecting the amount of maze walls purchased/sold by users through the blockchain. Level of maze and value of token (given as escape-rewards) can be balanced over time by two types of users’ participation: trading maze walls and escaping mazes, motivated by economic interests and fun, respectively.

3. Maze Escape Game

A schematic of our Ethereum-based maze escape game system is illustrated in [Fig. 1]. Most blockchain-based dApps must control three parts: the network, the blockchain, and the user’s device. In our case, as seen in [Fig.

1], such controller is encoded into the smart contract (Maze Escaper) to be executed by EVM in every Ethereum client. Maze escaper contains all commands and definitions on interaction with the user's device, on which information to update by accessing the network, and on what kinds of transactions of which form in which moment to generate for changing the blockchain data and handling user's queries such as buying/selling walls and paying escape rewards. In Section 3.1, we describe the game and its smart contract to be executed by EVM more precisely, and in the following two subsections the data maintained in Ethereum and the game client to be executed in user's device, respectively.



[Fig. 1] Overview of Maze Escape Game System

3.1 The Game and Smart Contract

3.1.1 Maze's era and the winner

In Ethereum network at any moment there exists the only “current” maze which has not yet been conquered (escaped) by any user. If a user succeeds in escaping this maze, then the

Ethereum client connecting this user to the network announces two transactions: the birth of the winner (which means the end of this maze's era) and the information of the next maze generated randomly in the winner's device (which means the opening of new maze's era). As soon as these transactions are registered to the blockchain (i.e., the block containing them is mined), the new maze becomes the only one alive in the network to be conquered by the users.

3.1.2 Escape-rewards and trading a wall

The winner gets escape-rewards (in our implementation, simply fixed number ‘15’ of tokens) and is recorded in the blockchain as the winner of this maze. The winner can use such obtained tokens for buying/selling a wall in the future. So, a user can enjoy the game in two ways: either by playing the game and earning tokens as escape-rewards, or by economic activities of buying/selling a wall. For simplicity we implement buying/selling a wall as just one increment/decrement of maze length, a simplest 1-dimensional realization of making the difficulty of a maze higher/lower.

Since our level adjusting algorithm `NextMazeParameters()` (described in the next subsection) increases the maze level by the difference between the number of walls purchased and the number of walls sold, these users' activities take a key role in balancing the level of maze and the value of token, as follows: if the level of current game is too low, then the supply of tokens newly issued for escape-rewards increases and the value of token collapses quickly. In this case,

users feel more motivated to spend tokens on buying walls than to earn tokens by selling walls or playing the game. This results in the level of the next mazes computed in `NextMazeParameters()` going up, thus the amount of tokens newly issued as rewards decreases and the value of token rises up. Conversely, if the level of current game is too high, then the value of token increases and users become eager to collect tokens either by winning the game or by selling walls than to pay expensive tokens for buying walls. This results in the level of the next mazes computed in `NextMazeParameters()` going down, thus the level of maze and the value of token become reasonably stable.

3.1.3 Three major elements of a maze



Every maze has three major game elements: forked roads, traps, and Threshold. The number of these elements affect the degree of difficulty (i.e. level) of a maze. Here we focus on the roles of these elements and explain technical details in the next subsections.

(i) Forked road is a road with two branches. Maze length L is defined as the number of forked roads in a maze. Instead of centralized policies for maze generation and security of its solution path, we take the following decentralized policies: the solution path of any maze with maze length L is fixed to be 1^L , and which of the two branches of a forked road is '1' (and the other is '0') is determined randomly in the player's device, at the first moment that the player makes a choice in front of it. Such random determination changes across time and devices,

thus choosing '1's in all L forked roads is a matter of luck of the player; every player knows the answer, but exhaustive searching is the only way to achieve it, just like seeking a nonce in mining a block. Previous winners owning tokens can affect the level of maze by buying a wall (incrementing the maze length), with intent to make the value of tokens higher. But manipulating the solution of a maze or preventing the others from escaping a maze are impossible.

(ii) For a maze, traps are obstacles to overcome and `NTraps` is the number of traps. We designed three kinds of monsters and four kinds of Physical traps listed in [Table 2]. Overcoming monsters 1, 2 and Physical traps of Types 1~3 requires the player's hand-eye coordination and motor skills as in action games and each is of nearly equal level, but the others need just good luck.

[Table 2] Items, Monsters, and Physical traps, contained in a maze

Object	Type	Attribute	Symbol
Item	1	20% reduction of the maze length	
	2	40% increase of health	
Monster	1	slowly moving monster; a hit causes 50% loss of health	1
	2	arrow shooter; an arrow-hit causes 10% loss of health	2
	3	3 rock-paper-scissors games; 3/2/1/0 wins of the player yield Item 1/Item 2/random addition of Monsters 1 or 2/30% loss of health, respectively	3
Physical trap	1	iron bar periodically sweeping from left to right	4

	2	huge rock quickly rolling down to the player	5
	3	fire unpredictably attacking the player	6
	4	random event of blank, Item 2, random addition of Physical traps of Types 1~3, and 30% loss of health.	7

(iii) Threshold H is the number of forked roads the system counts before giving a notice to the player that the current path is wrong, i.e., at least one of the previous choices at forked roads were '0', not '1'. If $H=1$, then right after the player takes '0' branch in a forked road the system blocks the way, so the player can move backwards and flip the choice just made. If $H \geq 2$, then right after a player passes the $(H-1)$ -th forked road in the first '0' branch, the system blocks the way to warn that the player is in the wrong way, which is the case even when the $(H-1)$ choices following the first '0' are all correct '1's. Therefore the best strategy of a player would be, as soon as being blocked, to move backwards step by step while checking which of the previous choices is the first '0'. Once knowing H , the player can escape the maze by repeating the following: to proceed forward, if blocked, to move backwards exactly to the H -th forked road, and to flip the choice.

We use threshold H as another maze-level controller other than maze length and the number of traps in the following reason: Assume that the maze length and the number of traps are the only maze-level controllers for some fixed H . If $H = \infty$ (meaning that the system gives no notice until the end of the maze) then the difficulty of the maze is too high, because the player cannot have any clue

about the first forked road at which the choices of the system and the player were different, and thus the player needs exhaustive search of all cases. On the other hand, if $H=1$ (meaning that the system gives notice right after the wrong choice) then the difficulty becomes too low, no matter how long the maze length is, thus the game becomes boring. So, a reasonable value of H is necessary for the game to be interesting. However, even with a fixed reasonable value of H , if the maze length and the number of traps are the only controllers to increase the maze level, then the maze length will grow too fast and the probability of being lucky decreases exponentially. So, incrementing H is a way to increase the game level slowly without increasing the maze length too fast.

The following `NextMazeParameters()` shows our level adjusting algorithm for the next maze's parameters $(L, H, NTraps)$, where the initial and final mazes are of $(1,1,0)$, $(L_{max}, 0.5L_{max}, 0.5L_{max})$, respectively. In Lines 3~4, we set the maze of $(L_{max}, 0.5L_{max}, 0.5L_{max})$ to be the final game, because moving backwards half-length of the longest maze ($H=0.5L_{max}$) and having as many traps as half the maximum number of forked roads ($NTraps = 0.5L_{max}$) is difficult enough to be the final game. In the rest of the algorithm, to increase the game level slowly without increasing the maze length too fast, we cut the next maze length into $0.7L_{max}$ with $H++$, and if H reaches half of L_{max} then the number of traps starts increasing. To justify these adjustment of maze parameters, we performed some level test of mazes given in [Table 3]; setting $L_{max} = 10$, we measured the

average play time of 5 users. Comparing the average play time of (10,2,0) with that of (7,3,0) shows that cutting $L = L_{\max}$ into $0.7L_{\max}$ with $H++$ is a reasonable way of levelling up with shorter mazes. Comparing the average play time of (10,5,0) with those of (7,5,1) and (7,5,2) shows that the number of traps is another maze-level controller that can increase the difficulty of maze even with shorter maze length.

[Table 3] Level test of some mazes by measuring average play time of 5 users

(Maze Length, Threshold, NTraps)	Average of 3 play-times of 5 users(sec)					
	user 1	user 2	user 3	user 4	user 5	average
(6,1,0)	31	26.4	26.6	23.8	26	26.76
(7,1,0)	36.3	28.9	30.5	28.9	33.3	31.58
(7,2,0)	51.6	40.5	33.6	37.6	54.3	43.52
(10,2,0)	80.3	49	48.7	50	86.3	62.86
(7,3,0)	84.6	52.6	52.9	54.8	91.3	67.24
(10,5,0)	219.3	187.6	206.3	247.3	223.6	216.82
(7,5,1)	228	195.6	211.6	243	225.6	220.76
(7,5,2)	229	196.3	213.3	246	229	222.72

```

NextMazeParameters()
    L ← L + BuyingWall - SellingWall
    if(L ≥ Lmax) then
        if(H = NTraps = 0.5Lmax) then
            Maze Escaper ends and exit
        else
            L ← 0.7Lmax
            if(H = 0.5Lmax) then
                NTraps++ and exit
            else
                H++
    
```

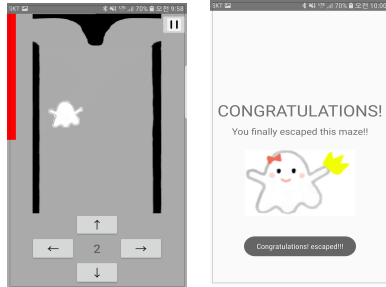
3.2 Maze Info and Winner Info

Now we explain the game data to be maintained/updated in the blockchain. Refer to [Fig. 1].

Maze count is the number of the current maze and winner count is the number of winners. They are basic information of the game used everywhere. For instance, the Ethereum client (i.e., node) connecting a player who has just escaped the current maze and declared being the winner, needs to check if this player is really the winner, i.e., the first one who escapes the current maze, before generating transactions. For this winner-checking, the node checks if this moment's winner count is one less than maze count of the maze the player just escaped.

Maze length, Ntraps, and Threshold were explained in Secion 3.1, so we proceed to maze string and trap string. Type of forked road is symbolized as '0' if left and straight, '1' if straight and right, and '2' if left and right. Maze string is defined as the sequence of types of forked roads, i.e., a string with alphabet {0, 1, 2} and length of maze length. Trap string is defined similarly; it is the sequence of types of traps, represented as a string with alphabet {0, 1, ..., 7} and length of maze length, where blank is symbolized as '0' and seven traps as the numbers from '1' to '7'. Maze string and trap string are randomly generated in the winner's device of the previous maze and are deployed into the blockchain as part of the transaction of winner declaration.

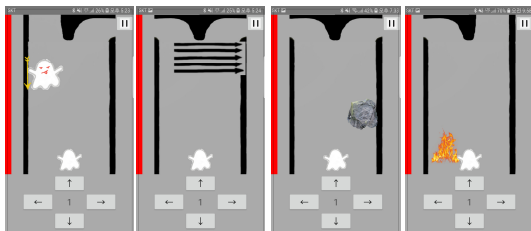
3.3 Game Client in the device



[Fig. 2] Game UI (left) and the screen for the winner (right)

The game client that a user downloads from a node, contains all programs to communicate with the node, to display all game objects including all types of forked roads, traps, items and the health-bar (displayed as a red bar in the left-side as in [Fig. 2]), and to interact with the player through the input/output devices. It conveys all information and queries between the node and the user, including maze information, buying/selling a wall and declaration of the winner and the next maze.

Since game UI and game objects are beyond the scope of this note, we pay little efforts on designing them; a brief summary of our game objects are given in [Table 2], and [Fig. 3] shows screen-shots of Monster 2 and Physical traps of Types 1~3 in some maze.



[Fig. 3] Monster 2(leftmost) and Physical traps of Types 1~3

3.4 Experiments

As mentioned before, we performed the level test of mazes to check if our adjustment of maze parameters described in the pseudocode of Section 3.1 is reasonable; setting $L_{\max} = 10$, we measured average play time of 5 users as shown in [Table 3]. To save efforts and test time, we skipped many intermediate mazes, but the table shows that the level of mazes are raised up in a reasonable rate and our three ways of levelling up work well: from (10,2,0) to (7,3,0) by cutting $L = L_{\max}$ into $0.7L_{\max}$ with $H++$, or from (10,5,0) to (7,5,1) by cutting $L = L_{\max}$ into $0.7L_{\max}$ with $NTraps++$, or from (7,5,1) to (7,5,2) by $NTraps++$.

We executed another test in the testnet “Ropsten”: to check if the game client in the winner’s device generates successfully a random maze string and a random trap string, each being of maze length, and if the smart contract in Ethereum client receives this new maze information and distributes it into the blockchain network successfully. Each row of [Table 4] shows the maze string and the trap string, each being of maze length and randomly generated in the winner’s device, and the blocknumber of the block containing the transaction that registers this maze’s information into Ropsten. This test shows that every process in both the game client and the smart contract are successful and the communication between them works well.

[Table 4] Maze string and Trap string of a maze, and its deployment into Testnet “Ropsten”

(Maze Length, Threshold, NTraps=0)	Maze String	Trap String (NTraps=0)	Block Number
Initial maze (1,1,0)	"0"	"0"	5077407
(6,1,0)	"021001"	"000000"	5094016
(7,1,0)	"0111020"	"0000000"	5094029
(7,2,0)	"1201221"	"0000000"	5094546
(10,2,0)	"0012100121"	"0000000000"	5094946
(7,3,0)	"0012110"	"0000000"	5094964

4. Conclusions

We have presented Ethereum-based maze escape game that is managed and updated in the blockchain network, with the game level adjusted by the amount of users' trading. This is the first blockchain-based maze game and the first blockchain-based user-participatory game updating system to fairly and transparently maintain game level.

This study raises an interesting question as to what else game variables can be managed through blockchain, besides the maze level discussed here. To keep the balance between game stability and purchase invigoration, careful design of which game variables to be forbidden or admissible to the public is necessary, with consideration of the characteristics of games. Once deployed, none of game environments and elements can be controlled by the game developer or a limited number of users secretly or maliciously because: changing blockchain data cannot be done retroactively without the alteration of all subsequent blocks and thus it is recorded and calls massive costs.

ACKNOWLEDGMENTS

The work was supported by the National Research Foundation of Korea(NRF) grant (No. 2016R1D1A1B03931879).

REFERENCES

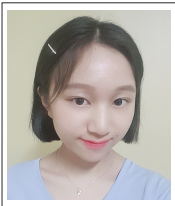
- [1] <https://www.stateofthedapps.com/ko/stats>.
- [2] Vitalik Buterin, “A Next-Generation Smart Contract and Decentralized Application Platform”, <https://github.com/ethereum/wiki/wiki/White-Paper>.
- [3] Introduction to Smart Contracts, <https://solidity.readthedocs.io/en/v0.4.24/introduction-to-smart-contracts.html#>.
- [4] Satoshi Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System”, <http://bitcoin.org/bitcoin.pdf>, 2009.
- [5] Jakobsson, Markus and Ari Juels, “Proofs of work and bread pudding protocols”, Secure Information Networks, pp. 258-272, 1999.
- [6] Sunny King and Scott Nadal, “Ppcoin: Peer-to-peer crypto-currency with proof-of-stake”, 2012.
- [7] <https://dapp.review/explore?category=1>.
- [8] CryptoKitties, <https://www.cryptokitties.co/>.



전 민 규 (Mingyu Jeon)

약 력 : 2017-현재 숭실대학교 컴퓨터학부 재학 중

관심분야: 블록체인, 4차산업, 경영학



황 지 연 (Chiyeon Hwang)

약 력 : 2017-현재 숭실대학교 컴퓨터학부 재학 중

관심분야: 블록체인, 빅데이터



나 현 숙 (Hyeon-Suk Na)

약 력 : 1993 서울대학교 수학과 이학사
1995-2002 포항공과대학교 수학과 이학석사 및 박사
2001-2003.2 프랑스 INRIA, HK UST Post Doc.
2003.3-현재 숭실대학교 컴퓨터학부 교수

관심분야: 알고리즘, 계산기하학, 컴퓨터그래픽스
