HAL open science

Decentralized Application Using Ethereum Blockchain on Performance Analysis Considering E-Voting System

B. Sriman, S. Ganesh Kumar

To cite this version:

B. Sriman, S. Ganesh Kumar. Decentralized Application Using Ethereum Blockchain on Performance Analysis Considering E-Voting System. 4th International Conference on Computational Intelligence in Data Science (ICCIDS), Mar 2021, Chennai, India. pp.187-199, 10.1007/978-3-030-92600-7_18 . hal-03772950

HAL Id: hal-03772950

https://inria.hal.science/hal-03772950

Submitted on 8 Sep 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

This document is the original author manuscript of a paper submitted to an IFIP conference proceedings or other IFIP publication by Springer Nature. As such, there may be some differences in the official published version of the paper. Such differences, if any, are usually due to reformatting during preparation for publication or minor corrections made by the author(s) during final proofreading of the publication manuscript.

Decentralized Application using Ethereum Blockchain on Performance Analysis considering E-voting system

B.Sriman Γ[1] S.Ganesh Kumar[2]

[1] Department of Computer Science and Engineering, SRMIST, Kattankulathur
srimanb@srmist.edu.in
[2] Department of Computer Science and Engineering, SRMIST, Kattankulathur
ganeshk1@srmist.edu.in

Abstract. This paper investigated how the technology behind many of the cryptocurrencies and the Bitcoin is Blockchain Technology. Bitcoin Blockchain was the first blockchain technology in 2008 to provide as the public digital payment with no other third party or the middleman interface. The result showed that technology replaced the cash exchange method (i.e., Cryptocurrency, Digital Assets, and Distributed Ledger). Furthermore, the key main characteristics of blockchain and their potentials to technology (e.g., Decentralization, Immutability, Transparency, Persistency, Auditability, Security, and Privacy). Most of the application was built by using Blockchain Technology. Blockchain-as-a-Service (BAAS) for the enterprise (e.g., Ethereum, Cloud Storage, Supply Chain Communication, smart contract and E-Voting System). In this paper, we developed a democracy E-Voting System, that replaces the ancient mode of voting system. We tested and implemented our E-Voting system that runs with a smart contract by using the E-Wallets. After the voting is confirmed, it holds all the data of voters, ballots and transaction data. Performance analysis of the E-Voting System is theoretical and practical implications of the result are discussed.

Keywords: Bitcoin · Blockchain · Ethereum · Proof of work · Smart contract · E-wallets · E-voting.

1 INTRODUCTION

Blockchain is a distributed decentralized application, where the nodes are connected through a peer-to-peer network and agreed to follow the rules that are mentioned in the smart contracts [1]. Where, the smart contracts [2],[3] are the codes that are written in the solidity programming language. In our paper, we used the Ethereum blockchain platform [4] to build a decentralized application for the e-voting system. In blockchain, we used a decentralized network instead of using the centralized network. In the peer-to-peer [P2P] network, all nodes share their records to every node in the distributed network [18]. A copy of one node's data will be shared with all the other nodes in the blockchain.

Here, no central authority is given to any of the nodes. As it is decentralized, anyone in the network can verify the other node whether it is valid or not. The node which is verified by the participant is called the miner. The miner is the participant, who undergoes for mining work. Here, mining is the process of solving mathematical crypto-puzzle [1]. Let us look deeply into the networks in which we are talking about. In the peer to peer [P2P] network [1], the shared data across the nodes are stored as the bunch of documents known as the blocks. Within, all nodes are together to construct the new registry publicly. Here, the data are secured under the cryptographic hash and are validated by using the consensus algorithm. In this paper, we are implementing our E-voting application to ensure that our vote is counted and secured [16]. As discussed above, the node ensures the sample of data beyond the web are the same. This is to the motive why we are constructing our E-voting System on the Ethereum blockchain [8]. In the existing system, when we use the web application, we will utilize a gateway to connect with the database among network and the data are shared through the central servers or the central database. If any transaction is encountered, the information will be shared through the central servers. So, our information may be accessed or been modified by any of the users. In voting system with the central servers, we may get a few problems as follows:

1. Mutability may occur in the database
2. The source code on the can be modified by any of the intruders

In our paper, we are going to build the E-Voting System which is decentralized and the nodes are connected to cast their votes. This is to ensure that our vote is casted and they are counted or not. Let us see the process deeply in the upcoming sections. Blockchain allows developers to build many of the Dapps by using different platforms. Here, we are going to build an application using the Ethereum plat- form [3]. One of the most important platforms to execute the arbitrary codes to run on the programs in Ethereum. With the help of smart contracts, the dis- tributed infrastructure could be able to complete the different types of projects in Ethereum. Ethereum allows us to build the decentralized and the fault-tolerant applications [4], which removes the intermediaries and provides transparency to all the nodes in the network. A new tradable token can be created by using the Ethereum that can be used as the cryptocurrency.

The wallets that are compatible with the Ethereum blockchain are used as token coin as a standard API [12]. To build a blockchain-based organization, the nodes to be added in a network should be agreed for the constraints[14]. Smart contracts will execute the contracts automatically only when all the constraints are satisfied. A clear concept of Ethereum should be known before entering into it. Now, we are going to learn few things about the Blockchain –[3] Ethereum platform and the working of smart contracts. Smart contracts [3] are the codes to execute the Ethereum blockchain with the help of Ethereum Virtual Machine [EVM] [5]. This is where all our business applications work. The smart contracts play an important role in reading, writing, and executing the data on the blockchain. The code is written in a solidity programming language, similar to the JavaScript [9]. The same type of things can be done by using these codes

for different applications. The smart contracts on the blockchain are related to the resource on the web. Therefore, our smart contract communicates with the business logic that executes with all our data and then the database layer of the blockchain is represented by public ledger [7].

2 PRELIMINARIES WORK OF E – VOTING SYSTEM IN ETHERUM BLOCKCHAIN

2.1 Motivations

- When we are traveling aboard or not available in our home town, the E-Voting system should be a traceable back end and transparent to the voter by using the Dapp in Ethereum platform.
- The E-Voting System uses a mobile app or computer to install the Dapp and the internet is plays major role in the whole voting procedure.

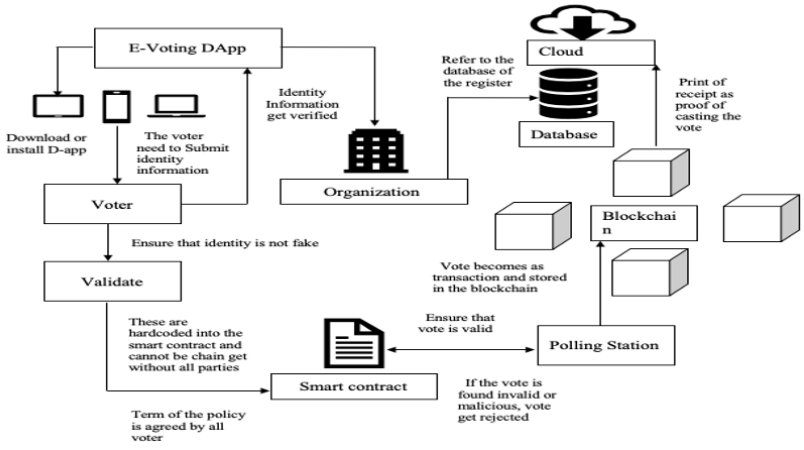

Fig. 1. Blockchain Architecture for Democracy E-Voting System

2.2 Challenges faced by E-Voting system

- Possibility of manage activities by a hacker who might straight infect and hack the servers.
- The occurrences of constraints or to modify voting procedures can be large number [18][23].
- No method to prove if your vote has been added in the E-Voting System[5][12].

3 RESEARCH METHODOLOGY

3.1 Ethereum Smart Contract

A smart Contract is a digital-based contract or digital version of a contract. it acts as a per contract, which runs on the blockchain node and executes the requirement of parties. The smart contract asset of a parameter in blockchain to become a trustless network[12]. If we design the right, a smart contract will surely be successful as it is driven by a computer system without human intervention [18][3].

3.2 Step to design a smart contract in Ethereum Blockchain

Step1. Understanding the Business problem – The Smart contract has set of parameter, it includes the business problem statement or understanding the business problem statement.
Step 2. Defining the blockchain network and its participants – Creating the blockchain network and giving them access permission to parties[20].
Step 3. Accessing involved parties of the contract – It is not possible to all the participants to be part of the smart contract. it must be only a small subset based on the rule and conditions, this small subset participant is called as involve parties.
Step 4. Exploring Prerequisites for the smart contract – It needs extended data, input from users. Etc[22].
Step 5. Examining trigger event for the smart – it means that smart contract work on the set of rule and condition, once this condition is met, the smart contract is executed successfully[17].
Step 6. Evaluating business logic for the smart contract – "if-else" statement and condition. "if " the conditions are fulfilled or true statement, then what course of action will be followed, "else" what will happen? In smart contracts.

4 INSTALLATION

For developing our DApp, we need to install few of the dependencies.

4.1 NPM (Node Package Manager)

The initial dependency is the npm that comes along with the node.js [6]. Through the integration of npm, the truffle comes with standard. It means that we can distribute and use the contracts, Dapps and even more Ethereum enabled libraries through npm. To check the version of the node, we use the command: node -v

4.2 Truffle framework

The truffle framework[5] is our second dependency, which allows us to build the DApps in the Ethereum blockchain. It helps us to interact through the smart contract. We use the npm package to install the truffle. To install the truffle, we use the command: npm install -g truffle. Make sure the truffle[5] is compiled and deployed before interacting with the smart contracts and then to interact with them by using the Web3. Here, we suggest that truffle The contract library is much easier and robust.

4.3 Metamask

The simplest way to interact with the DApps is by using the metamask. To connect with the Ethereum network, we use an extension in Chrome or Firefox without running the whole node on the browsers. The metamask is capable of connecting with the main network and any test networks like ropsten, rinkeby, kovan, or a local Ethereum development called Ganache or Truffle Develop. [23][14][19].

4.4 Ganache

The local in-memory blockchain is our third dependency and it is called Ganache [5]. It can be downloaded by using the website from the truffle framework. After installation, ganache gives 10 free accounts within our local Ethereum blockchain. This means, initially the original account holds 100 ethers. A solidity programming language is a language that is highly recommended for Ethereum blockchain [20]. Here, we are using the sublime text and we have installed the Ethereum package which supports solidity language.

5 IMPLEMENTATION

5.1 Building the Ethereum Smart contract

For a smart contract, the language used is the Solidity [10]. The programming language that is developed on top of the Ethereum Virtual Machine [EVM] [5]. The main purpose of solidity is to store all the states of the token, which are sent and received by the nodes. The programmers who are experienced in JavaScript, C++, Python can easily understand the solidity language [10]. If the solidity language is known well, then the program for Ethereum smart contracts can be written easily. The election file is written with the solidity code. Initially, we declare the solidity language with its pragma version as required. Then, the name for our contract will be given by the contract name in the smart contract with the keyword "contract"[24]. The state variable of the candidate name will be stored and allows us to store it in the blockchain. We declared it as a variable and that variable is a string.

5.2 Setting up a smart contract in solidity language

Pragma version 0.6.0;

{
Step 1. Struct keyword keep track the number of candidate in an election
// Struct candidate
Step 2. Set the unique function identifier id for each candidate of various size
//Uint id;
//String name
Step 3: Number of vote count to the candidate
// Uint votecount;

}

This string will be visible publicly in the blockchain. Because it allows us to access the value of our smart contract [25]. The function is a constructor and it is known by the solidity codes. Now, we have created our base for our smart contract. This is how we will code for the candidate model and it is given below as solidity code for storing multiple candidates and attributes

5.3 Code Snippet Mapping the candidate

Step 1: Calling the function Election contract contain a persistent data in state variable and function
// Contract Election {
Step 2: set the function identifier id
// struct candidate
Uint id;
String name ;
Uint votecount;
}
Step 3:Mapping is seen as hash tables which refers the type as arrays and struct to read and write data as mapping (uint ¯¡ candidate) Public view candidate. Here, we modified the code for a candidate in the solidity. The solidity [10] allows us to create our structure type as we have already done in our code. By declaring the struct, we can't simply give the candidate [13]. First of all, we should instantiate it and assign the variables, before we write the codes to store it in the blockchain. The next step is to store the candidate. For that, we need some space for structure type. We can simply do solidity by mapping for this. Here, mapping is to resemble an array or the hash, that associates with the key pair values. And the Solidity code for mapping a candidate structure, unsigned integers are explained.

5.4 Code Snippet Storage the Candidate count

Step 1: Calling the function state variable and function
// Contract Election {
Step 2: set the function identifier id
// struct candidate
Uint id;
String name;
Uint votecount;
}
Step 3: Mapping is seen as hash tables
// mapping(uint => candidate) Public view candidate
Step 4: Possible to store UTF-8encoded data in string variable Candidate count
Uint Public candidatecount:

}

5.5 Code Snippet Using Counter Cache

Step 1: Calling the function state variable and function
// Contract Election {
Step 2: function add Candidate or remove the candidate
Remove lsit candidate name from array and struct
function addcandidate (srting name) private
{
Candidat count++
Step 3: number responses every time are display a counter cache keep a separate counter
Candidate(camdidatescount, _name,0):
}
}

To write data in the blockchain, we assign the key-value pairs to the state variable for mapping. To get a better function, we set the function as previously with the candidate name in the test [13]. Let's how the candidate's vote are gets counted in solidity code for reading/write, store candidates in the codes.

5.6 Testing Smart Contract in Ethereum Main Network

Now, we are going to test our E-voting smart contract in the ethereum main network by using the Metamask and Ganache. We need to create the new test file in the command line from the root of our paper by using the following command: touch test/election.js To bundle the truffle files, we are using the JavaScript file with the mocha testing framework[5]. Here, we write the JavaScript file to simulate the client-side interaction with a smart contract. The truffle test

command is given to test that our contracts are running perfectly or not. we use the Client-side application to interact with the smart contracts. Let us see how it works and how it is helpful for us. In the truffle package, we get the HTML and the JavaScript file. We just want to modify for our paper. Here, these files keep us to write any of the CSS as we wish to modify. We use the lite server in our paper to develop our e-voting decentralized application. For all these, we no need to worry about the codes. It will keep the HTML and JavaScript codes simple and focuses on it. Let us now focus on how to develop smart contracts as a part of our decentralized application.

1. Web3 is a JavaScript library that allows us to interact with the client-side application to the blockchain. We configured the Web3 inside a function as "initWeb3"[11]
2. Initialize the contracts: we have to initialize some value inside the contract function to interact with it.[14]
3. Render function: from the smart contract, it renders all the function layouts on the content page with the data.

we have created a smart contract to list the candidates. Now, looping is passed through each candidate in the mapping and rendering it to the table and the current account is also fetched and displayed on the page[8]. Initially, we have to check whether we have migrated our contracts through truffle or not. To do migrate function, we should use the command truffle migrate.

5.7 Lit server – installation and configuration

we start our web server and start using DApp[14]. The lite server.

Initiating the webserver

Open the bs-config.json in the text editor and examine the contracts

```
{
"server": {
"baseDir":["./src", "./build/contracts" ]
}
}
```

The above codes tell the Lite-server[13] to which and where the files to be included in the directory. Here, we added ./src directory for the website files and contracts, we added ./build/contracts directory. The dev command to check script in package.json file in the root directory of our paper. The script code looks as follows:

5.8 Code Snippet for testing the server

```
"script": {
"dev": "lite-server",
```

"test": "echo " error: no test specified " && ext1"
}

The above script code tells the npm to run the lite server. When we give the command "npm run dev", the server starts running and the local host for our paper will be displayed. After the npm command[21], it should automatically open the web browser with our client-side application. Here, the core functionality is to increase the candidate's vote count by reading the struct out of the "candidates"mapping and increasing the "vote count" by 1 with the increment operator.

Two things needed to be tested:

1. Whether the function increments the candidate's vote count or not
2. Whether the voter is added to the mapping whenever they vote or not

Now,[13] we can ensure that if any transaction during the vote fails, when an error message is returned. In that message, we are able to revert that the error message contains the "revert" substring. Later, we could finalize that our contract state is unaltered and that the candidate who participated didn't receive any vote. To prevent double voting, we ensure to write the test code. Now, we query for the candidate id in the form. It will go to the smart contract and pass the id to call the vote. This is called an "asynchronous call". When it gets over, the page will be shown as the content to the user as shown in figure 7. After submitting, we will be directed successfully to casted a vote from the page. Then, ensure that the lite server is running. Because our assets are being served to the browser and proceeded to the browser page. There, we see the list of candidates to vote, which are connected to the ethereum blockchain main network. And it will be connected by the client-side application and it is shown in figure 8.

A popup message will come from the Metamask to confirm that we want to submit the transaction to the ethereum blockchain and to cast our vote. From the metamask account address, some amount of gas fee, a gas limit will be charged. It will be equal to 2.16 USD dollars [12]. After submitting the transaction, the user will be able to cast their vote and submit it successfully on the Ethereum blockchain main network. Finally, we have successfully casted our vote with the client-side e-voting in the distributed application as shown in figure 10.

Fig. 2. Screenshot for Election Results

Fig. 3. Screenshot for Client-side interact in E-Voting System

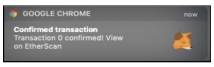

Fig. 4. Screenshot of Metamask Confirmation

Fig. 5. Dapp Vote successfully

6 DISCUSSION

- **Performance analysis:** As per the literature survey of the blockchain voting systems listed below are completely 13 seconds faster than the traditional voting systems. For multiple number of transactions, the same blocks are being created by inconsistent system process. Due to high scalability issues, only few of the nodes were tested across the distributed network with 10 nodes as shown in the figure 6

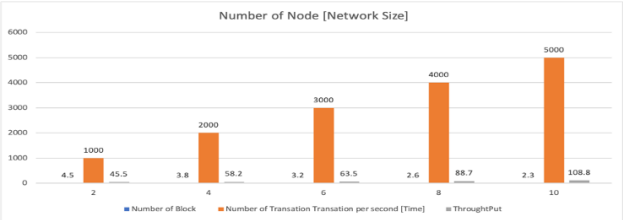

Fig. 6. Number of TPS vs Network size

- **Measured Throughputs:** The throughput is analyzed by comparing it with the network size, number of transactions per second (in time). Based on the network size, the time will be needed to generate a block in the network. More than the public Ethereum network, the private Ethereum network has faster block propagation. It is observed that the performance does not changes significantly with different network sizes for POW consensus. It is shown in the table that the observed output drift with an increasing.
- **Network Configuration:** Where the computational power in the network is the bottleneck problem and it highly requires computational power to solve the mathematical crypto puzzle for verifying the transactions around 10 minutes and add the blocks into the blockchain.
- **Size of the network:** Due to the limitations in the network like size of a block, number of blocks and the transactions per second are some of the restrictions for the performance that are to be reduced.
- **Scalability/Throughput:** Speed of the network. The time is taken to do the transaction and mined it onto the network. In the Ethereum, it takes 10 min to confirm the transaction in the network [9]. Because of our transaction in the network wait up till 6 more blocks to be added in the network. So, throughput is a big problem in the Ethereum. The number of Transactions

Table 1. Measured Throughput

Number of Node [Network size]	**Number of Block**	**Number of Transaction Per Second [Time]**	**Throughputs [Tps]**	**Measured Throughput**
2	4.5	1000	45.5	4.5(89%)
4	3.8	2000	58.2	5.8(98%)
6	3.2	3000	63.5	6.3(75%)
8	2.6	4000	88.7	8.8(70%)
10	2.3	5000	108.8	10.8(66%)

Per Second (TPS) in Ethereum is much lower than 2000 Transaction Per sec (TPS) for other payment alternatives like Visa take 21000 Transaction Per Sec (TPS).

- **Cost:** As we saw earlier, the mining process in Ethereum is expensive and the variable value of the Gas price adds to the trouble. The Gas price is unpredictable and suddenly the price of Gas will increase because of the volatile consumer building level of production application on Ethereum extreme challenge.
- **Decentralization:** In Bitcoin, 5 miner groups get together to control over 51% of the network. This violates the base of the Blockchain, which is decentralized because of these 5 miner groups can together administer or control the entire system. The same can happen in Ethereum. Though the algorithms for mining were designed in such a way that the chances of this happening are less.
- **Size:** The size of the Ethereum Blockchain is increasing, and nodes have to be highly equipped to store Blockchain of this size.
- **Ethereum:** Based DApps are not user-friendly yet.

7 CONCLUSION

In this paper, we developed a dapp as novel and optimistic web application so called "blockchain application in E-Voting System". We conclude that the main goal of our dapp development is to do E-Voting System Securely [14] [15] by using the Ethereum blockchain in the main network. We successfully implemented our dapp for voting to the candidates transparently by the voters and in a transparent and verifiable manner. For future work, the time complexity and the constraints for the node scalability [17] of the users to cast their vote from the registered network can be improved efficiently also the scalability issues.

References

1. S. Nakamoto, "Bitcoin: a peer to peer electronic cash system": https://bitcoin.org

2. V. Buterin, "Ethereum white paper: a next-generation smart contract decentralized application platform," [Online]. Available: https://github.com/Ethereum/wiki/wiki/White-Paper, 2013.
3. G. Wood, "Ethereum: A secure decentralized generalized transaction ledger," Ethereum Project Yellow Paper, vol. 151, 2014
4. N. Kshetri, J. Voas, Blockchain-Enabled E-Voting. IEEE Softw. 35, 95–99
5. (2018). https://doi.org/10.1109/MS.2018.2801546
6. M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, L. Jia-Nan, Y. Xiang, R. H. Deng, CrowdBC: A Blockchain-Based Decentralized Framework for Crowdsourcing. IEEE Trans. Parallel Distrib. Syst., 1 (2018). https://doi.org/10.1109/TPDS.2018.2881735
7. Dhulavvagol, Praveen Desai, Abhilash & Ganiger, Renuka. (2017). Vehicle Tracking and Speed Estimation of Moving Vehicles for
8. Traffic Surveillance Applications. 373-377. 10.1109/CTCEEC.2017.8455043.
9. S. Shafeeq, M. Alam, A. Khan, Privacy-aware decentralized access control system, Future Gener. Comput. Syst. 101 (2019) 420–433, http://dx.doi.org/10.1016/j.future.2019.06.025, URL http://www.sciencedirect.com/ science/article/PII/S0167739X18332308.
10. Counterparty, 2019, https://counterparty.io/. (Accessed 30 Nov 2019).
11. J.P. Cruz, Y. Kaji, E-voting System Based on the Bitcoin Protocol and Blind
12. Signatures, 2016.
13. J.P. Cruz, Y. Kaji, N. Yanai, RBAC-SC: Role-based access control using smart
14. contract, IEEE Access 6 (2018) 12240–12251.
15. Democracy.earth, 2019, http://democracy.earth/. (Accessed 30 Nov 2019).
16. D. Di Francesco Maesa, A. Lunardelli, P. Mori, L. Ricci, Exploiting
17. blockchain technology for attribute management in access control systems, in International Conference on the Economics of Grids, Clouds, Systems, and Services, Springer, 2019, pp. 3–
18. D. Di Francesco Maesa, P. Mori, L. Ricci, Blockchain-based access control, in IFIP International Conference on Distributed Applications and Interoperable Systems, Springer, 2017, pp. 206–220.
19. D. Di Francesco Maesa, P. Mori, L. Ricci, Blockchain-based access control services, in 2018 IEEE International Conference on Internet of Things, iThings, and IEEE Green Computing and Communications, GreenCom, and IEEE Cyber, Physical and Social Computing, CPSCom, and IEEE Smart Data, SmartData, IEEE, 2018, pp. 1379–1386.
20. D. Di Francesco Maesa, P. Mori, L. Ricci, A blockchain-based approach for the definition of auditable access control systems, Comput. Security. 84 (2019) 93–119.
21. The European Parliament and the Council of the European Union, regulation (EU) 2016/679 of the European Parliament and of the council of 27 April 2016, Off. J. Eur. Union, 4.5.2016.
22. The path to self-sovereign identity, 2019, http://www.lifewithalacrity.com/2016/04/the- path- to- self- soverereign- identity.html. (Accessed 30 Nov 2019)
23. Secure.vote, 2019, https://secure.vote/. (Accessed 30 Nov 2019).
24. Shitang Yu,Kun Lu,Zhou Shao,Yingcheng Gou,Bo Zhang(2018)," A High-Performance Blockchain Platform for Intelligent Devi" IEEE International Conference on Hot Information-Centric Networking.
25. John Domingue; Michelle Bachler ; Kevin Quick," Smart Blockchain Badges for Data Science Education"2018 IEEE Frontiers in Education Conference (FIE)