Register 6 (2) 2020 pp. 74-85
Jurnal Ilmiah Teknologi Sistem Informasi (Scientific Journal of Information System Technology)

ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

Available online to www.journal.unipdu.ac.id

Unipdu

S2-Accredited – SK No. 34/E/KPT/2018
Journal page is available to www.journal.unipdu.ac.id:8080/index.php/register

Two factor authentication framework based on ethereum blockchain with dApp as token generation system instead of third-party on web application

Marsha Chikita Intania Putri [a], *Parman Sukarno* [b], *Aulia Arif Wardana* [c]

[a,b,c] *Department of Informatics Engineering, Telkom University, Bandung, Indonesia*

email: [a] *marshachikita@student.telkomuniversity.ac.id* , [b] *psukarno@telkomuniversity.ac.id* , [c] *auliawardan@telkomuniversity.ac.id*

ARTICLE INFO

Article history:
Received 1 May 2020
Revised 27 May 2020
Accepted 28 May 2020
Published 3 June 2020

Keywords:
authentication
blockchain
ethereum
third-party
web

IEEE style in citing this article:
M. C. I. Putri, P. Sukarno and A. A. Wardana, "Two factor authentication framework based on ethereum blockchain with dApp as token generation system instead of third-party on web application," *Register: Jurnal Ilmiah Teknologi Sistem Informasi*, vol. 6, no. 2, pp. 74-85, 2020.

ABSTRACT

Authentication is a method for securing an account by verifying the user identity by inputting email with a password. Two factor authentications is an authentication system that combines the first-factor authentication with the second factor. General two factor authentication by entering an email or username with a password are similar. However, two factor authentication requires additional information that must be inputted by the user. Additional information can be in the form of tokens or one-time passwords (OTP). Two factor authentications generally still uses third-party services to generate token or OTP still have vulnerable because can attacked from tokens steal through MITM and found that the generated tokens with the same value. Therefore, we propose a two-factor authentication framework based on ethereum blockchain with dApp as token generation system. Firstly, outcome from the analysis of the system, next succeeded in creating a two-factor authentication system without using third-parties. Second, token system generate up to 3164 different tokens in one second and has been collisions tested. Third, security method to protect token from MITM attack. The attacker unable to get access caused all the checking are done by dApp user authentication.

1. Introduction

Authentication is usually a combination of email and password enable access to the account. Web application are platforms that uses authentication system to check an account is really accessed by own user. Authentication has three factors: something that you know, something that you have, and something that you are. Authentication usually encounter in web applications is one-factor authentication or password-based authentication refers to the category something that you know [1, 2, 3, 4]. Password-based authentication has weaknesses, that are easy to brute force attacks, so the users data are vulnerable of being stolen [5, 6]. Users usually use the same password on many platforms to make it easier to remember [4, 7]. Such method is vulnerable to attacks. Hackers who succeed in getting the user passwords from one platform can also stole the user data on another platforms. Therefore, password-based authentication cannot accommodate the security and confidentiality of account data [8, 9, 10].

To resolve the weakness of password-based authentication in web applications by adding a layer of security and combining two authentication factors together called two factor authentications (2FA).

ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

2FA is an authentication system that combines the first-factor authentication (something that you know) with the second factor (something that you have). Performance as same as with 2FA password-based authentication, but 2FA is requiring to enter additional information in the form of tokens or one-time passwords (OTP). The token is generated by a third-party then be sent to the user via SMS or mailing system.

According to paper [11], tokens generate are done by the users and stored into the blockchain. Tokens can be used every time will do 2FA. In this way, the token used for each authentication has the same value and token only used once. Refers to [2], Tokens generate system handled by a smart contract and sent to the user via OTP-SMS, but They are still use third-parties to send tokens and is not secure for the 2FA system because tokens can be stolen by attackers through MITM (Man-In-The-Middle) [2, 12].

We propose a 2FA framework that is different from the existing system and develop with dApp as a token generating system. DApp is a decentralized application that runs on peer-to-peer networks [3]. Another difference is that the general 2FA system will send the generated token via third-party to the user to be inputted on. However, the user does not need to manually input the token because checking will be done automatically by the system. Implementing Ethereum blockchain technology to avoidance using third-party and to develop this system because Ethereum can modify centralized systems into distributed systems with programming capabilities. This system can counter MITM and 2FA attacks systems from third-parties.

2. Related Work

Longfei Wu et al. in paper [13] proposed a two factor out-of-band authentication scheme method for blockchain infrastructure-based IoT devices to prevent collusion on a centralized server. The proposed authentication on this system works by checking the IoT device that was previously registered in the blockchain with the closest blockchain node. Then the node will send a code to grant access or activate the device. It is also mentioned in the paper that the system can prevent attacks from malicious devices even though the attacker can steal tokens which means the token security of the system is still not safe.

Table 1. Summarize of the related works

Researcher	Strength	Weakness
Alharbi et al. [2]	The system is safe form third-party attacks. The system is safe from MITM attacks	This system still uses OTP-SMS to distribute the token via SMS. If the handphone is lost and the attackers manage to take over the handphone then they can get the token to enter the system.
Amrutiya et al. [11]	The advantage of this system is that it uses a private blockchain which does not require large costs and memory sizes	The token in this system does not suit the characteristic of token which should be a one-time password that can only be used once.
Wu et al. [13]	The system can prevent attacks from malicious devices even though the attacker can steal tokens	- Failure possibility on the first-factor authentication - In the paper is written that the system can prevent attacks from malicious devices even though the attacker can steal tokens which means the token security of the system is still not safe.
Park et al. [14]	The system guarantees high level authentication.	- Tokens that are sent from authenticator to the application are plain text so they are vulnerable to token theft by MITM attacks. - Token collision can happen because the token is based on the time

Park et al. in the Park et al. [14] proposed a two factor authentication framework with hyperledger fabric as a solution to the private blockchain problem. The token used on this system is TOTP (Time-based One Time Password) generated by including the time information when the token is generated at that time. The way this system works is that after the user enters login data, the authenticator will

M. C. I. Putri et al./Register 6 (2) 2020 pp. 74-85
ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

generate a token which will then be sent directly to the application without the need to send it first to the user. If the tokens match, the user will be given access to the system. The system guarantees high level authentication. However, tokens that are sent from authenticator to the application are plain text so they are vulnerable to token theft by MITM attacks.

Alharbi et al. [2] proposed a 2FA framework method with OTP-SMS built on the blockchain network to overcome various kinds of attacks including MITM and third-party attacks. The paper explained that the OTP is generated by a smart contract and would be encrypted and hashed before being sent to the user to avoid snooping. But the system in this paper still uses OTP-SMS where OTP generated will be sent to the user via SMS. The disadvantage of OTP-SMS is that if the handphone is lost and the attackers manage to take over the handphone then they can get the token to enter the system.

The problem raised in Amrutiya et al. [11] is that tokens on 2FA are generated by centralized third-parties where usually trusted third-parties are always under security threats. Therefore, the author proposed a 2FA method to use blockchain without using a third-party by adding an openSSH extra security layer. The advantage of this system is that it does not require third-party so it does not need to be integrated with SMS or mailing systems. Because of this system does not use third-parties, we propose system tokens for 2FA are generated by own user. The token which consists of six digits will then be saved to the blockchain, so if later at another time the user wants to log into the system, the user must enter the six-digit token that has been entered before. It does not the characteristic of token where is a one-time password only that can be use. The related works above can be summarized in Table 1 below.

3. Research Method

After obtaining the conclusion that weaknesses were still found in the existing system, we took the initiative to try to build an authentication system by combining the strengths and fixing the weaknesses of the existing authentication system.

In this paper, we propose a two factor authentication system without third-party on a web application using blockchain. We use blockchain because blockchain has several advantages in terms of data security and integrity making it suitable to be applied to authentication systems [15, 16, 17]. The blockchain that we use is Ethereum blockchain which can be used to change all centralized systems into distributed systems with unique programming language [18, 19, 20, 21]. The 2FA system that we made is different from other pre-existing systems. In the pre-existing 2FA system that don't use blockchain, they use third-party to generate and distribute tokens. Meanwhile in the pre-existing 2FA system that already use blockchain, they use smart contract to generate tokens and SMS system to distribute tokens to the user. On the system that we made, we generate tokens from dApp (distributed application) in this case is the web server and does not require third-parties and this system does not need to distribute the tokens to the user because all the checkings have been done automatically by dApp.

```
const min = 1;
const max = Math.pow(10, 15);
function getRandomInt() {
  return Math.floor(Math.random() * (max - min + 1)) + min;
}

let newToken = getRandomInt();
while(previous == newToken) {
  newToken = getRandomInt();
}
return newToken;
```

Figure 1. Generating token

The system in Amrutiya et al. [11] still does not meet the characteristics of tokens which should only be used once. In that paper, the token that is used every time doing authentication has the same value because the token is generated directly by the user. Therefore, in the system that we made, we pay attention to the token generate system so that it can provide as many tokens as possible with different values. The token generated by our system consists of 15 digits which the algorithm can be

ISSN 2502-3357 (Online)
M. C. I. Putri et al./Register 6 (2) 2020 pp. 74-85 ISSN 2503-0477 (Print)

seen in Figure 1. The reason we made the token consists of 15 digits is to minimize the possibility of the tokens having the same value. Although the token has 15 digits which can be considered too long, the user does not need to worry because the user does not need to input the token manually.

The security of tokens from MITM attacks on Alharbi et al. [2] has been proven safe, it's just that the system still uses SMS to send tokens to the users so that if the handphone is lost the account security is not guaranteed. MITM attack is one of the attacks that has the greatest success rate. The attacker will secretly position themselves between two end-points that communicate with each other so that the attacker can change, capture, or steal information sent by both end-points [22, 23, 24, 25]. MITM attacks are carried out with targets to damage integrity, confidentiality, and availability [26, 27, 28]. The system that we made applies the same token security as in Alharbi et al. [2] by hashing the generated token using SHA-256 algorithm and then will be stored temporarily in the session struct. Session struct contains email, the user Ethereum address, role, token, and token verification. Verification tokens are set to false by default. To perform two factor authentication, the user must click on the authenticate button, then the hashed token is sent to the blockchain using the Ethereum address of the user to be stored. The web server will compare whether the Ethereum address that made the transaction or saves the token into the blockchain has the same address as the address that is temporarily stored in a session struct. If the user Ethereum address match, then the token verification value will set to true and access to the website is granted. So that our system does not need to send tokens to the user first, but will be checked automatically by system.

3.1. System overview

An overview of the system can be seen in Figure 2 where the user accesses a distributed application (dApp) which has web3js, registration smart contract, and 2FA smart contract while dApp is connected to the Ethereum blockchain.

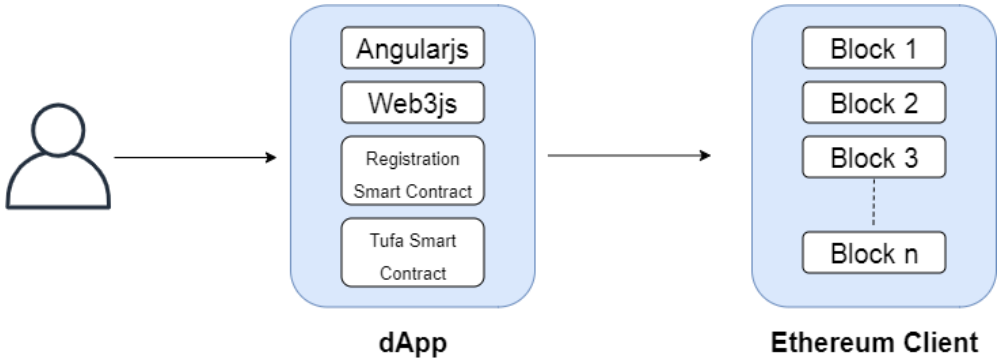


Figure 2. 2FA with ethereum blockchain

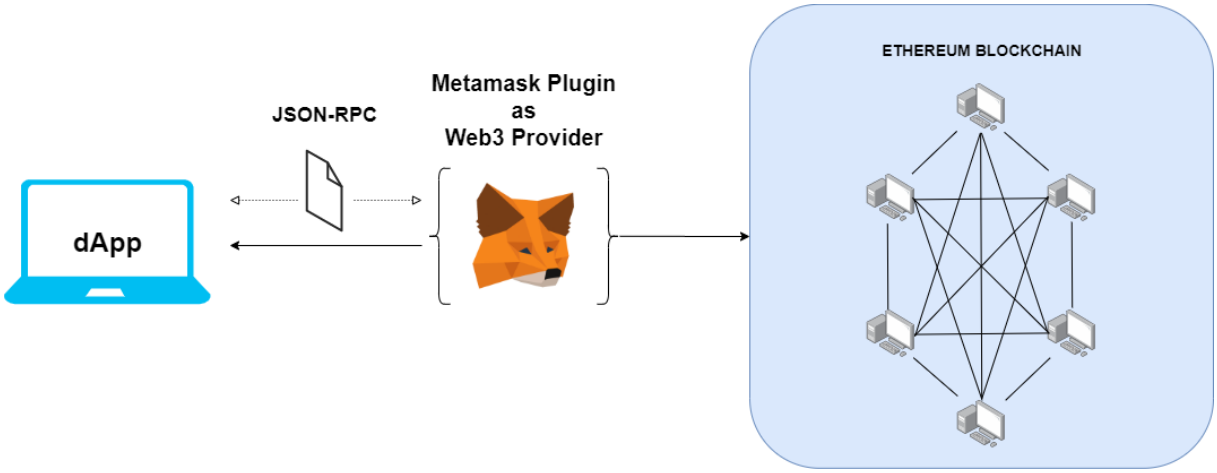
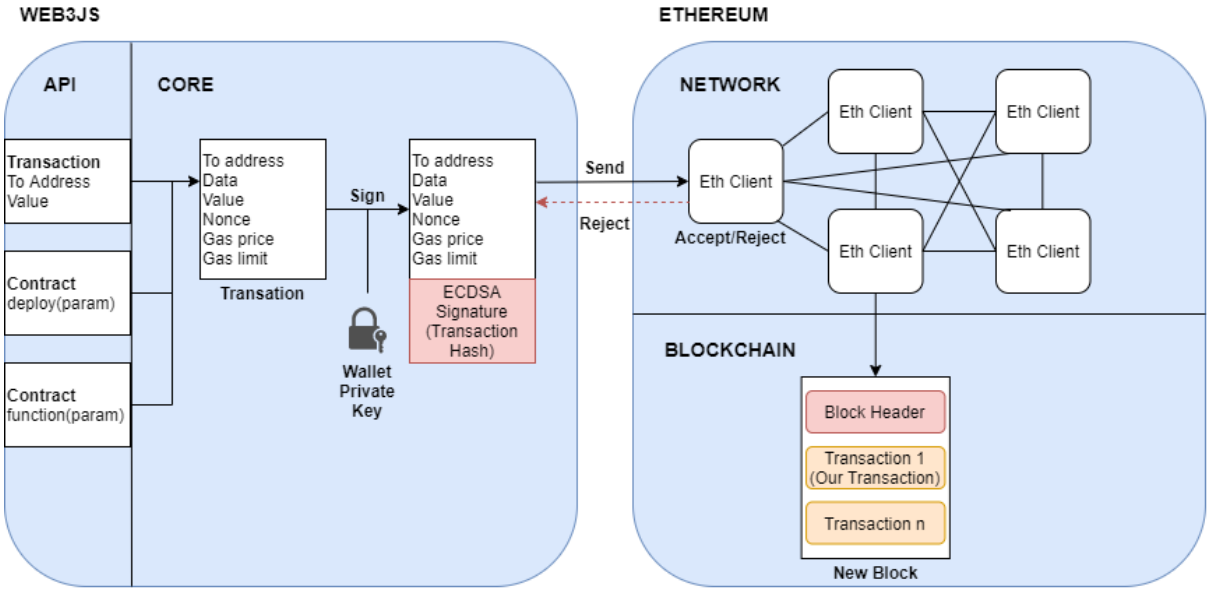


Figure 3. Web3 provider for blockchain-based 2FA

Distributed application (dApp) is a component that is used to create an Ethereum blockchain-based application. Dapp consists of a collection of smart contracts needed to build a system. First smart contract is named "registration smart contract" that used for registering or creating an account, and the second smart contract is named "tufa smart contract" that used for doing all the checkings related to

ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

two factor authentication. We used solidity language programming to build the smart contracts. Besides smart contract, there are also support by web3js on dApp, a web library that is used to work on smart contracts and is integrated with nodes on the Ethereum network. Blockchain-based two factor authentication uses web3 provider as a bridge for dApp and Ethereum blockchain to communicate. To see more detail how dApp can communicate with the Ethereum blockchain can be seen in Figure 3.
To find out in more detail how web3js works with Ethereum blockchain can be seen in Figure 4. Web3js is divided into two parts named Application Programming Interface (API) and core. Blockchain-based two factor authentication on a web application use API to make and carry out transactions according to the functions exist in the smart contract. Then the transaction will be executed in core by signing on the transaction that has been made using a private key wallet owned by each user. The signed transaction



will then be forwarded to the blockchain network and recorded in a block in the node.

Figure 4. Transaction between Web3js and ethereum blockchain

Before transactions are recorded in blocks on each node connected in the Ethereum blockchain network, signed transactions must move from web3js to the blockchain network. Next, signed transactions will be forwarded to the blockchain network using an object notation called JavaScript Object Notation (JSON). In Ethereum blockchain-based applications, there is an object notation that has a function like JSON called JSON-RPC uses to distribute data or information from front-end to back-end or reverse [29, 30, 31].

3.2. How the system work

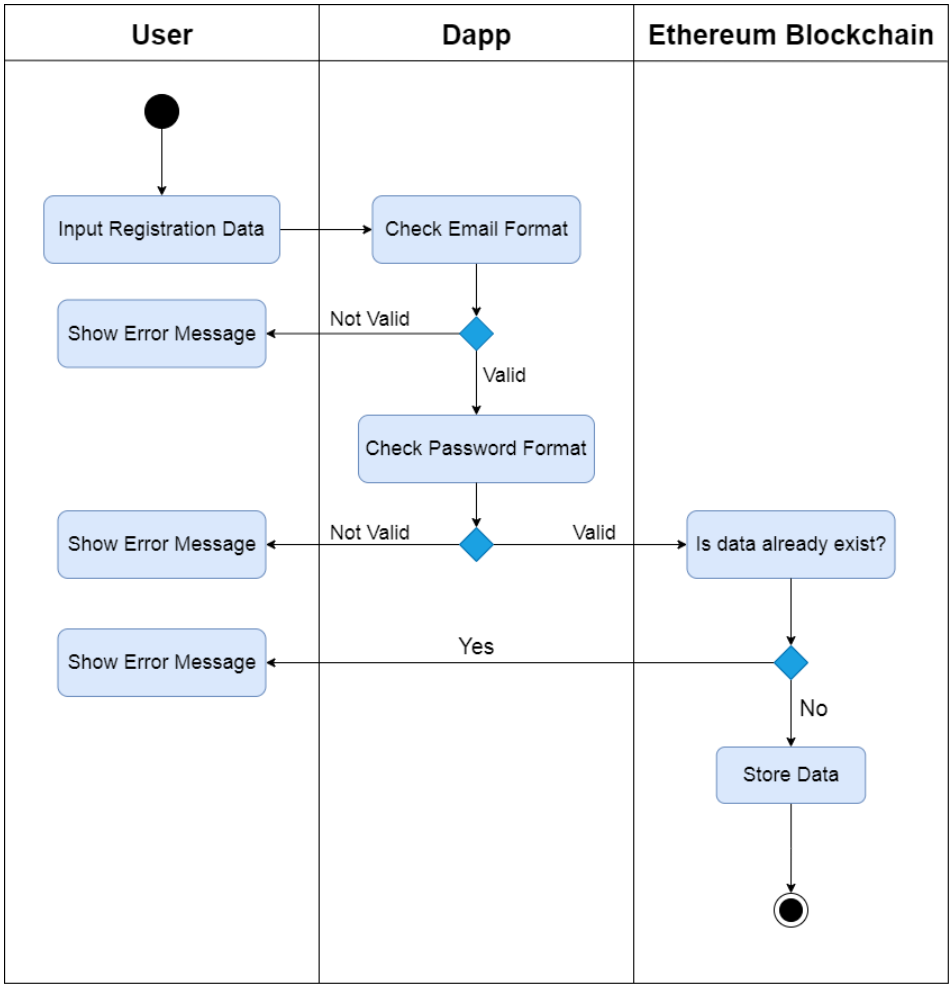
Authenticate process, user should create an account by registering email and password and will be check before data is stored to the blockchain, e-mail and password are checked. Emails are checked to have the appropriate format. If an invalid, error message will be displayed. Conversely, if the email is valid, then proceed with checking the password. Registered password must be at least eight characters long, have uppercase letters, symbol, and numbers. If password that to be registered does not meet these conditions, then error message will be displayed. If valid, then check again whether the email to be registered already exists on the blockchain. If email is already registered in blockchain, user should to register with another email and the account is successfully created. More details can be seen in Figure 5. The account registration flow is illustrated in activity diagrams.

The login flow with two factor authentications shown by Figure 6. The user enters the email and password, then dApp will check whether the entered email already registered in the blockchain. If it is not registered yet, an error message will be displayed. Conversely, if the email has been registered, dApp will create a session and generate token followed by a page move to the 2FA verification page. On the 2FA verification page, user must click the authenticate button to do two factor authentications. After that, the token that has been generated by dApp will be stored to the blockchain. DApp will check if the token stored on the blockchain is the same as the generated session token. If they are not the same

ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

an error message will appear, if true then the user access is granted.

Figure 5. Registration activity diagram



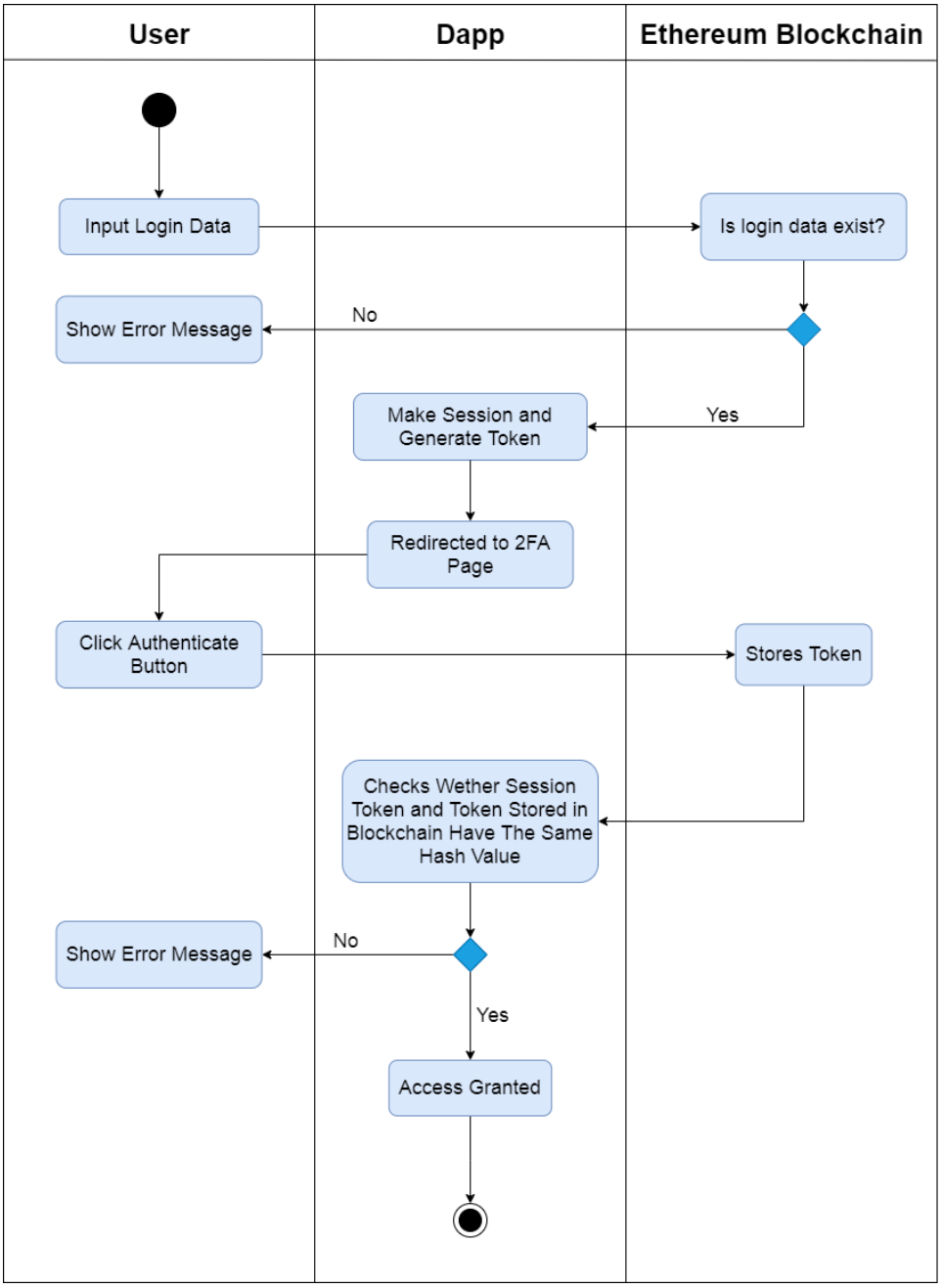
4. Experiment and Results

The system is developed shown in Figure 7. The application has a role-based access control where only the super admin and admin can create a new account. Accounts that have super admin roles can create other super admin and admin accounts. Accounts that have admin roles can only create staff accounts. Meanwhile staff accounts can not create any accounts. Super admin can login for the first time by using default email and password. In this section, we will discuss more on the authentication system rather than the registration system.

The two-factor authentication works in this system is user enters email and password that have been registered on the registration menu. Next, checking the format of the email and password. If the format and requirements entered does not match, error message will appear. If the format and requirements match, the dApp or web server will generate a token consisting of 15 digits then the token hashed using SHA256. Hashed-tokens will be temporarily stored in a session struct. Session struct also stores some attributes needed to be authenticate, user email, Ethereum address, roles, tokens, and token verification. Token verification is set by default (false). Then the user will be directed to the verification page which has an "authenticate" button to do the two-factor authentication process. On this page, the user only needs to click on the button and dApp will send a hashed-token to be saved on the blockchain. Hashed-tokens are stored in an array that has an index in the form of a user Ethereum address on the blockchain. After the hashed-token has been successfully stored in the blockchain, dApp will call a function that has the user Ethereum address parameter in the smart contract. The function will return the token value in the array with an index that matches the user Ethereum address in the session struct. Then dApp will check whether the token obtained from the function call on the smart contract has the same value as the token stored in the session struct, then the user will be given access rights. Conversely, if the token value is not the same then an error message will appear. The 2FA system that we made does

M. C. I. Putri et al./Register 6 (2) 2020 pp. 74-85
ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

not require third parties to generate and distribute tokens. Users who authenticates in system also do not need to enter tokens because all checking have been done automatically by dApp. The



authentication proccess can be found on web server terminal in Figure 8.

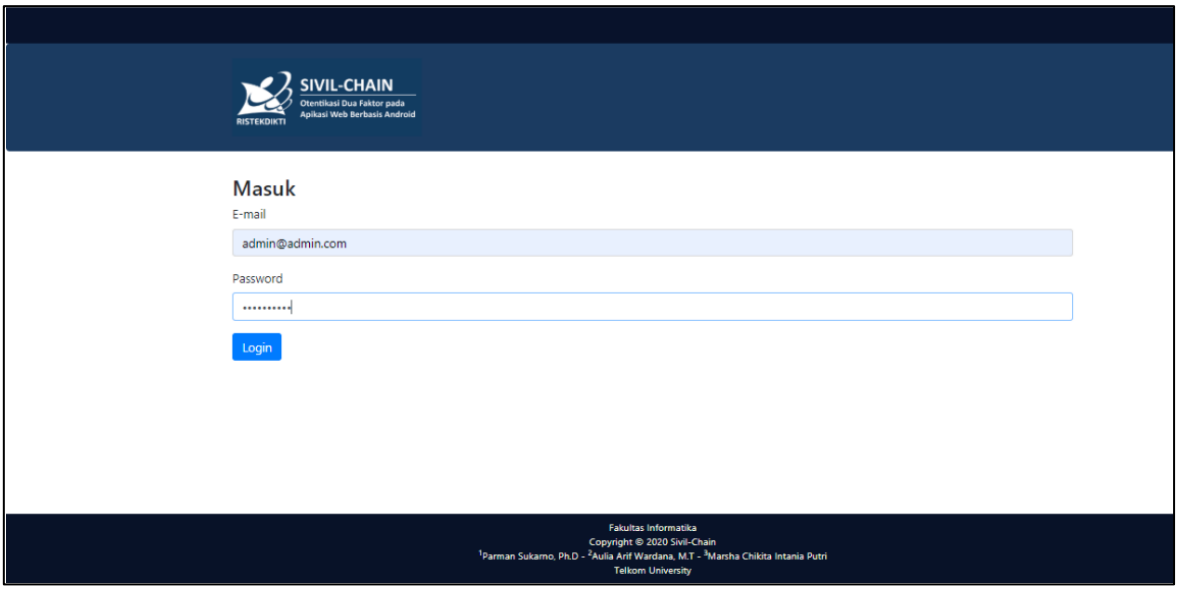
Figure 6. Login activity diagram

As explained above, the 2FA framework that we made use dApp as a token generating system. We tested dApp as a token generating system to determine whether the generated token algorithm there is still possible to have collisions. Collision is a condition where two or more inputs will produce the same value. This system is tested by 10 users logging in almost simultaneously in a span of 5 seconds, 10 seconds, and 15 seconds. The results of the test in Table 2. Tokens generate algorithm can produce up to 3164 tokens in a second. The possibility of the users that logging in at the same second and getting token with the same value is impossible. Taking 5 sample token values that were successfully generated by system before entering the hashing stage in Figure 9. That way can be concluded that the token generated from the dApp can solved collision and security issue are better than the token generating system previous [10] and [13].

4.1. Credential protection

ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

There are two authentication factor password and token. The first-factor is minimum conditions for the password they will use. The password must consist of at least 8 characters and must have an upper case, number, and symbol to avoid of brute force attack. Second factor by hashing the generated token. So,



what is sent and stored in the blockchain is not plain text but in the form of a hash that is safe from MITM attacks.

Figure 7. Login page web

Figure 8. Two factor authentication process

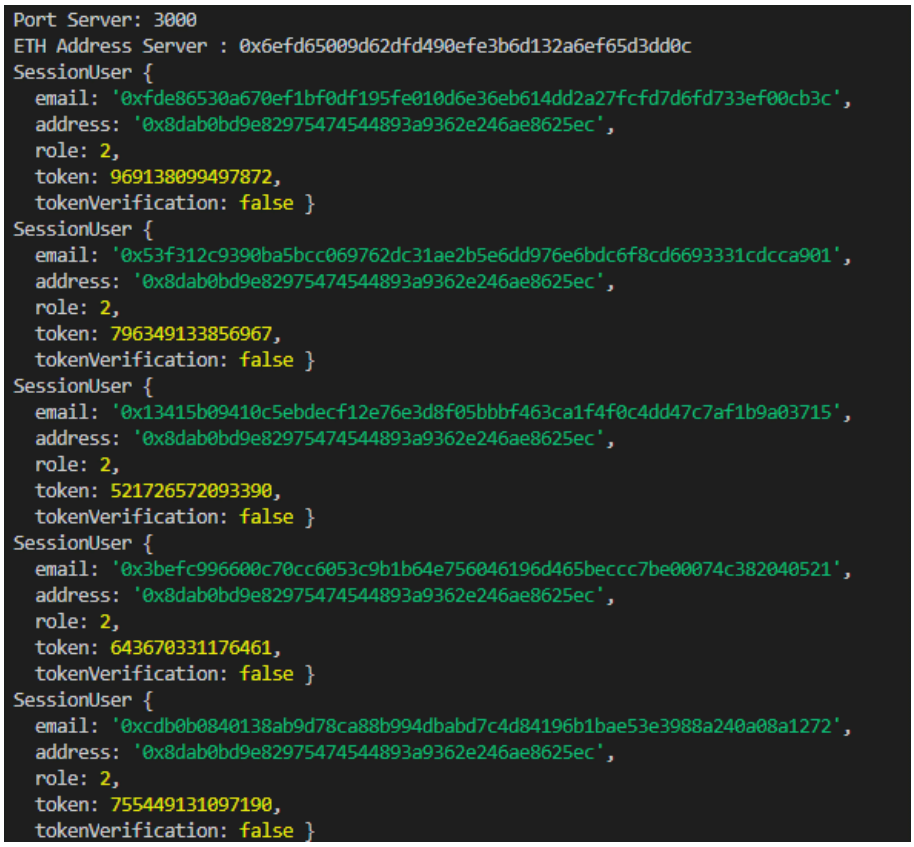
Table 2. Collision testing result

No	User	Time Span		
		5 seconds	10 seconds	15 seconds
1	User1	366134138927268	198891348747564	175924906629819
2	User2	657159696694923	899831056556082	153206935510941
3	User3	112382625936340	633482357102994	610940384091053
4	User4	398174291792390	501206394930417	322138298187288
5	User5	956865233570482	537155351584300	700189954136554
6	User6	738408563523303	540511375182516	350947647115600
7	User7	833158974100208	186592308515453	715848278654690
8	User8	159463384425952	567343759214490	987695726248925
9	User9	513310856849926	873156427393222	927261996621611
10	User10	281088250422206	132727436933244	321874185276144

```
Port Server: 3000
ETH Address Server : 0x6efd65009d62dfd490efe3b6d132a6ef65d3dd0c
Email hash: 0x53f312c9390ba5bcc069762dc31ae2b5e6dd976e6bdc6f8cd6693331cdcca901
Password hash: 0x1fdef9e03445cf2de1ecdc95a44809f7b120cbde333bc778bdbf63ea0f8a4a64
Checking email on blockchain...
Email is registered on blockchain with:
Email hash              : 0x53f312c9390ba5bcc069762dc31ae2b5e6dd976e6bdc6f8cd6693331cdcca901
Password hash           : 0x1fdef9e03445cf2de1ecdc95a44809f7b120cbde333bc778bdbf63ea0f8a4a64
Ethereum address        : 0x8dab0bd9e82975474544893a9362e246ae8625ec
Email and password match!
SessionUser {
  email: '0x53f312c9390ba5bcc069762dc31ae2b5e6dd976e6bdc6f8cd6693331cdcca901',
  address: '0x8dab0bd9e82975474544893a9362e246ae8625ec',
  role: 2,
  token: 741106513878914,
  tokenVerification: false }
Checking token...
Login success! User ETH address : 0x8dab0bd9e82975474544893a9362e246ae8625ec
```

M. C. I. Putri et al./Register 6 (2) 2020 pp. 74-85
ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

MITM attack testing by trying to capture or sniff out packages sent from dApp to the blockchain. When a user login into the system, other actors will try to peek on the token using sniffing tools. The result is that the packet can be seen, however the token value cannot be known because the one sent to the blockchain is not a plain token, but a token that has been hashed using SHA-256. In addition, the



advantage of the system is that users do not need to manually input tokens. So, if an attacker can decode the hash to get the token value, the attacker will not be able to get access rights because all checkings are carried out by dApp which is only owned by the user.

Figure 9. Tokens generated by web server before being hashed

4.2. Secure communication

Blockchain is a collection of several blocks that are connected to each other. A block consists of the hash block itself, the data, and the previous hash block. When a block is created, it automatically calculates the block hash value itself. The hashing system on the blockchain can detect changes that occur in a block. If the block hash value changes, then the block can no longer be considered as the same block. Blocks can be connected because each block stores hash values from the previous block. For example, suppose block A has the hash value of block A (the first block has no previous hash block value). Block B has the hash value of block B and block A. Block C has the hash value of block C and block B. The three blocks are connected because block B stores the hash value of block A and block C stores the hash value of block B. If there is an attacker want to damage block A, then the block A hash value will change. That way, the block A hash value that is owned by block B is invalid which causes block C and subsequent blocks to be invalid. That means, changing one block in the blockchain will cause the next blocks to be invalid.

Hash system alone is still not enough to avoid attacks. Today's computers are so sophisticated that the attacker can recalculate the hash values of all blocks to get a valid blockchain. Therefore, the term "proof-of-work" appears on the blockchain. Proof-of-wok on the blockchain is used to slow down the creation of a block. Usually the proof-of-work calculation takes about 10. That way, the attacker is difficult to damage the block because the attacker takes a long time to recalculate the subsequent proof-of-work blocks.

ISSN 2502-3357 (Online)
M. C. I. Putri et al./Register 6 (2) 2020 pp. 74-85 ISSN 2503-0477 (Print)

As explained in the previous section, a blockchain is a collection of nodes that are connected to each other. Anyone can join the blockchain network and they will get a full copy of the data stored in the blockchain. If someone creates a block, then the block will also be sent to all nodes that are connected in the network to verify and determine whether the block is tampered by the attacker or not. The tampered block will be rejected. So, the chance of the attacker to successfully damage the blockchain is very small because the attacker must damage all the blocks in the blockchain then recalculate the proof-of-work of each block. In addition, the attacker must at least be able to take control at least 51% of the nodes connected in the blockchain network. Therefore, we use this blockchain technology to secure authentication framework.

5. Conclusions

Secure authentication system is combined strengths and improve the weaknesses of existing systems. The conclusions are, tokens are generated directly by dApp and there is no need to distribute tokens to users because checking the first factor and the second factor is done automatically by the system. Therefore, system succeeded in creating a two-factor authentication system without using third-parties. The token generation system on the framework has been collisions tested. Our system can generate 3164 different tokens in one second for minimize of several users logging in at the same time with a token the same value. The result of token sniffing is to get tokens in the form of hashes, it is difficult for the attacker to get the original value of the token, even if the attacker can get the token value, they will not be able to get access because all the checkings by dApp and own user.

The developer desire to change the smart contract code are possible, it requires re-deployment which causes the data that already exists on the Ethereum blockchain lost. But re-deployment can be avoided if the developer already understands and familiar using smart contract programming languages. So, the developer can implement the smart contract without any change in the future. Therefore, we recommend understanding the smart contract programming language and analyze code before deploying the smart contract to avoid re-deployment.

6. Acknowledgement

Thank you to Telkom University Forensic and Security (Foresty) Laboratory which has become a place for researchers to build the systems and develop this research article. Also, we would like to express our gratitude to Telkom University for funding this research.

7. References

[1] D. DeFigueiredo, "The Case for Mobile Two-Factor Authentication," *IEEE Security & Privacy,* vol. 9, no. 5, pp. 81-85, 2011.

[2] E. Alharbi and D. Alghazzawi, "Two Factor Authentication Framework Using OTP-SMS Based on Blockchain," *Transactions on Machine Learning and Artificial Intelligence,* vol. 7, no. 3, pp. 17-27, 2019.

[3] R. Gupta, Hands-On Cybersecurity with Blockchain: Implement DDoS protection, PKI-based identity, 2FA, and DNS security using Blockchain, Birmingham, UK: Packt, 2018.

[4] J. Song, D. Wang, Z. Yun and X. Han, "Alphapwd: A Password Generation Strategy Based on Mnemonic Shape," *IEEE Access,* vol. 7, pp. 119052-119059, 2019.

[5] H.-M. Sun, Y.-H. Chen and Y.-H. Lin, "oPass: A User Authentication Protocol Resistant to Password Stealing and Password Reuse Attacks," *IEEE Transactions on Information Forensics and Security,* vol. 7, no. 2, pp. 651-663, 2012.

[6] J. Yan, A. Blackwell, R. Anderson and A. Grant, "Password Memorability and Security: Empirical Results," *IEEE Security & Privacy,* vol. 2, no. 5, pp. 25-31, 2004.

[7] M. Shirvanian, N. Saxena, S. Jarecki and H. Krawczyk, "Building and Studying a Password Store that Perfectly Hides Passwords from Itself," *IEEE Transactions on Dependable and Secure Computing,* vol. 16, no. 5, pp. 770-782, 2019.

[8] A. D. Yulianto, P. Sukarno, A. A. Warrdana and M. A. Makky, "Mitigation of Cryptojacking Attacks Using Taint Analysis," in *4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, Yogyakarta, Indonesia, 2019.

ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

[9] A. A. Wardana and R. S. Perdana, "Access Control on Internet of Things based on Publish/Subscribe using Authentication Server and Secure Protocol," in *10th International Conference on Information Technology and Electrical Engineering (ICITEE)*, Kuta, Indonesia, 2018.

[10] A. Rauf, M. Faiqurahman and D. R. Akbi, "Secure random port list generator pada mekanisme autentikasi dengan menggunakan Port Knocking dan Secure Socket Layer," *Register: Jurnal Ilmiah Teknologi Sistem Informasi,* vol. 4, no. 2, pp. 103-113, 2018.

[11] V. Amrutiya, S. Jhamb, P. Priyadarshi and A. Bhatia, "Trustless Two-Factor Authentication Using Smart Contracts in Blockchains," in *International Conference on Information Networking (ICOIN)*, Kuala Lumpur, Malaysia, 2019.

[12] S. M. Danish, M. Lestas, W. Asif, H. K. Qureshi and M. Rajarajan, "A Lightweight Blockchain Based Two Factor Authentication Mechanism for LoRaWAN Join Procedure," in *IEEE International Conference on Communications Workshops (ICC Workshops)*, Shanghai, China, 2019.

[13] L. Wu, X. Du, W. Wang and B. Lin, "An Out-of-band Authentication Scheme for Internet of Things Using Blockchain Technology," in *International Conference on Computing, Networking and Communications (ICNC)*, Maui, HI, USA, 2018.

[14] W.-S. Park, D.-Y. Hwang and K.-H. Kim, "A TOTP-Based Two Factor Authentication Scheme for Hyperledger Fabric Blockchain," in *Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, Prague, Czech Republic, 2018.

[15] A. Shahnaz, U. Qamar and A. Khalid, "Using Blockchain for Electronic Health Records," *IEEE Access,* vol. 7, pp. 147782-147795, 2019.

[16] C. Lin, D. He, N. Kumar, X. Huang, P. Vijayakumar and K.-K. R. Choo, "HomeChain: A Blockchain-Based Secure Mutual Authentication System for Smart Homes," *IEEE Internet of Things Journal,* vol. 7, no. 2, pp. 818-829, 2020.

[17] R. Shrestha and S. Y. Nam, "Regional Blockchain for Vehicular Networks to Prevent 51% Attacks," *IEEE Access,* vol. 7, pp. 95033-95045, 2019.

[18] S. Sayeed, H. Marco-Gisbert and T. Caira, "Smart Contract: Attacks and Protections," *IEEE Access,* vol. 8, pp. 24416-24427, 2020.

[19] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng and V. C. M. Leung, "Decentralized Applications: The Blockchain-Empowered Software System," *IEEE Access,* vol. 6, pp. 53019-53033, 2018.

[20] S. R. Niya, F. Schupfer, T. Bocek and B. Stiller, "A Peer-to-peer Purchase and Rental Smart Contract-based Application (PuRSCA)," *it - Information Technology,* vol. 60, no. 5, pp. 307-320, 2018.

[21] Q. Xu, Z. He, Z. Li, M. Xiao, R. S. M. Goh and Y. Li, "Chapter 8 - An effective blockchain-based, decentralized application for smart building system management," *Real-Time Data Analytics for Large Scale Sensor Data,* vol. 6, pp. 157-181, 2020.

[22] A. Esfahani, G. Mantas, J. Ribeiro, J. Bastos, S. Mumtaz, M. A. Violas, A. M. D. O. Duarte and J. Rodriguez, "An Efficient Web Authentication Mechanism Preventing Man-In-The-Middle Attacks in Industry 4.0 Supply Chain," *IEEE Access,* vol. 7, pp. 58981-58989, 2019.

[23] C. Li, Z. Qin, E. Novak and Q. Li, "Securing SDN Infrastructure of IoT–Fog Networks From MitM Attacks," *IEEE Internet of Things Journal,* vol. 4, no. 5, pp. 1156-1164, 2017.

[24] M. Agarwal, S. Biswas and S. Nandi, "Advanced Stealth Man-in-The-Middle Attack in WPA2 Encrypted Wi-Fi Networks," *IEEE Communications Letters,* vol. 19, no. 4, pp. 581-584, 2015.

[25] Y. Zheng and W. Wu, "Security of Khudra Against Meet-in-the-Middle-Type Cryptanalysis," *Chinese Journal of Electronics,* vol. 28, no. 3, p. 482–488, 2019.

[26] M. Conti, N. Dragoni and V. Lesyk, "A Survey of Man In The Middle Attacks," *IEEE Communications Surveys & Tutorials,* vol. 18, no. 3, pp. 2027-2051, 2016.

[27] G. Oliva, S. Cioabă and C. N. Hadjicostis, "Distributed Calculation of Edge-Disjoint Spanning Trees for Robustifying Distributed Algorithms Against Man-in-the-Middle Attacks," *IEEE Transactions on Control of Network Systems,* vol. 5, no. 4, pp. 1646-1656, 2018.

[28] F. Ahmad, F. Kurugollu, A. Adnane, R. Hussain and F. Hussain, "MARINE: Man-in-the-Middle Attack Resistant Trust Model in Connected Vehicles," *IEEE Internet of Things Journal,* vol. 7, no. 4, pp. 3310-3322, 2020.

[29] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli and M. H. Rehmani, "Applications of Blockchains in the Internet of Things: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials,* vol. 21, no. 2, pp. 1676-1717, 2019.

[30] Y. Hu, A. Manzoor, P. Ekparinya, M. Liyanage, K. Thilakarathna, G. Jourjon and A. Seneviratne, "A Delay-Tolerant Payment Scheme Based on the Ethereum Blockchain," *IEEE Access,* vol. 7, pp. 33159-33172, 2019.

[31] S. Guo, X. Hu, S. Guo, X. Qiu and F. Qi, "Blockchain Meets Edge Computing: A Distributed and Trusted Authentication System," *IEEE Transactions on Industrial Informatics,* vol. 16, no. 3, pp. 1972-1983, 2020.