User-Managed Access Delegation for Blockchain-driven IoT Services

Chun-An Lin
Dept. of Computer Science
National Chengchi University
Taipei, Taiwan
107753020@nccu.edu.tw

Chun-Feng Liao
Dept. of Computer Science
National Chengchi University
Taipei, Taiwan
cfliao@nccu.edu.tw

2020 International Computer Symposium (ICS) | 978-1-7281-9255-0/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICS51289.2020.00097

***Abstract*—**With the advancements in network bandwidth and hardware capability, the Internet of Things (IoT) has now become more and more prevalent. Nevertheless, due to the increment of the data value, incidents of malicious attacks are also spreading. Therefore, the robust access control mechanism plays an important role when designing an IoT system. Regrettably, most of the IoT devices have the limitation of the power and computation capacity. To tackle this issue, it is general for the IoT system to delegate the access control service to the third party. However, centralized access control services may bring about some significant disadvantages such as lack of transparency and reliability. Moreover, it is difficult for users to track authorization history and manage authorization policies. Fortunately, blockchain is regarded as a promising technology that can cover the shortage of the centralized system and improve the security of IoT. This paper proposes a Blockchain-assisted User-Managed Access (B-UMA) schema base on the IoT scenario. To prove the feasibility of the proposed schema, we also build a prototype system adopting the "Smart Factory" use case. Finally, the qualitative comparison among B-UMA and related access control frameworks are also presented.

Keywords—Blockchain; Internet-of-Things; Access Control; User-Managed Access

I. INTRODUCTION

Recently, IoT (Internet of things) services quickly become popular. Cisco [1] reported that there are already 50 billion smart things interconnected over the network. Generally, an IoT service consists of some devices with limited computing and networking capabilities [2]. The IoT service can be exposed to the Internet and thus the devices within the service can be easily accessed and controlled by various commodity user devices. However, without proper access control mechanisms (a.k.a. authentication and authorization), the IoT service usually suffers from a lot of security and privacy issues. For instance, the devices can be tampered by a malicious party.

To reduce the cost of building in-house authentication and authorization solutions for IoT services, a common way is to use a trusted third-party authentication service following an open standard such as OAuth [3]. Because most of the existing third-party authentication service lacks transparency, one open issue of this approach is whether these third-party services are trustable. For example, these thirty-party services may make profits by selling or utilizing the sensitive data of IoT services. A profile of OAuth called the User-Managed Access (UMA) [4] is proposed to deal with the problem mentioned above. For example, after a user is authenticated with UMA, the user is also empowered to manage the corresponding policy (rule of authorization) of the protected resources [5]. Also, the user can delegate the access right to the authorization server to address the access requests asynchronously.

However, several issues appear when designing the access control mechanisms for IoT services based on UMA: The first issue is the *Availability* of the access control service. UMA relies on a centralized authorization server. Thus, it suffers from the risk of a single point of failure; The second issue is *Transparency* and *Traceability*. In a typical third-party access control service, most of the history of the authorization process cannot be traced or inquire easily; The final issue is the *Maintainability* access control service. Once the centralized system has been deployed, it takes considerable time and cost to update a new version of the service.

Based on these observations, this paper proposes a decentralized approach based on the blockchain. Blockchain is recognized as one of the most promising technologies that complement the IoT [6]. The underlying technology of the blockchain is a distributed ledger, which is transparent, reliable, immutable and traceable, and a distributed consensus mechanism that ensures the consistency of the distributed ledger. The recent development of the blockchain platform (i.e., Ethereum), has brought about a new idea called Smart contract, namely a piece of executable programming logic that allows the performance of credible transactions, where the results of the transactions are verified by peers. Many research papers focus on exploring the ways of combining blockchain and IoT (also known as the Blockchain-driven IoT services, B-IoT) recently. This paper aims to investigate how to realize UMA on top of B-IoT. The benefits of such a combination are listed below:

- *Decentralization*. Traditionally, UMA is supported by a dedicated and centralized Authorization Server (AS). However, based on Blockchain technology, the AS and the overall UMA protocol be realized in a decentralized way [7].
- *High maintainability*. To implement UMA on a blockchain network, UMA entities (see Table 1) are realized as Smart Contracts on the blockchain network.

978-1-7281-9255-0/20/$31.00 ©2020 IEEE
DOI 10.1109/ICS51289.2020.00097

Authorized licensed use limited to: National Cheng Chi University. Downloaded on April 01,2021 at 07:20:16 UTC from IEEE Xplore. Restrictions apply.

In this way, the UMA entities can be designed and maintained in a uniform way. Thus, the maintainability is higher than the traditional approach.

- *Reusable security infrastructure*. Most of the blockchain platforms (e.g., Ethereum) provides a public-key-cryptography-based account system. Participants in the blockchain are allowed to prove their identity by providing the account managed by the blockchain platform. Furthermore, the blockchain provided a secure billing layer so that it is straightforward to build a protected resource sharing marketplace among the peers in the blockchain network.

To sum up, the objective of this paper is to propose a decentralized UMA access control management mechanism for B-IoT based on blockchain. The rest of the paper is organized as follows. In Section II, we present backgrounds of UMA, Blockchain, and IoT. After that, the design issues about B-UMA are presented. Then, the detailed descriptions of B-UMA are presented in both static and dynamic ways. In order the ensure the security of the proposed framework, the security analysis is performed in section IV. Lastly, the conclusion and future work are discussed in section V.

II. RELATED WORK

A. User-Managed Access (UMA)

Among the widely recognized authorization frameworks, OAuth2 is the most popular. However, the scenario of the party-to-party access right delegation does not cover in OAuth2. For instance, Alice can authorize the resource from application A to B that control by herself. However, she can't give access permission to Bob. Unfortunately, this problem will become more prominent in IoT applications. Take car rental for example. It is apparent that the resource (car) must be rented from the third party. UMA, developed by Kantara Initiative [8], has been proposed specifically to fill the gap of access right delegation. The UMA specification draft has been submitted to IETF [9]. TABLE I shows the main entities defined in the UMA specification.

TABLE I. COMPONENTS IN SMART FACTORY

UMA entities	*Description*
Requesting Party (RqP)	The third-party attempt to access protected resources.
Resource Owner (RO)	The owner of protected resources takes charge of defining the authorization policy.
Client	An application or server on behalf of RqP to request protected resources owned by RO.
Resource Server (RS)	A server keeps protected resources and provides Application Programming Interfaces (API) for RqP to access.
Authorization Server (AS)	A server which is delegating by RO to realize the RS protection and issue the access request in an asynchronous way.

UMA proposed a list of key requirements that enable party-to-party access control. These requirements are: (1) User-driven policies. RO can customize the authorization policy for protected resources. Anyone can request authorization from the AS through these policies; (2) Support for claims-based access control. In the authorization process, AS may require RqP to provide more *claims* (i.e., the statement of values of identity attributes) to verify the identity of the RqP; (3) User management of access control. RO doesn't need to involved in the authorization process directly but defines the authorization policy in the AS. On the other hand, RO can also modify the policy and terminate the access control service at any time.

B. Blockchian and Internet of Things

Blockchain is regarded as the core underlying technology of Bitcoin [10]. The decentralization feature of blockchain is regarded as a promising solution to problems (e.g., single point of failure, lack of transparency) of the centralized system. Besides, the recent development of blockchain technology, such as Ethereum [11], is supported to run some Turing-complete programming languages, known as "Smart Contract." Thus, the developer is enabled to run the "Decentralized application (DApp)" in the blockchain through smart contracts.

There are two general transmission methods in the blockchain system, namely on-chain and off-chain. The on-chain method implies dispatching and transmitting data using the blockchain. In Ethereum blockchain, a logging mechanism, called Event, is provided to retrieve and filter the state change of smart contracts. Furthermore, Event plays an important role in communication between blockchain platforms and programs (by emitting event to programs once the state change happens in the smart contract); On the other side, off-chain represents all the method dispatching and transmitting data without blockchain.

To join the blockchain network, the component has to arm with a blockchain endpoint. (e.g., Go-ethereum [12] or Parity) When running a fully-functional blockchain endpoint, known as *Full node*, considerable computation and storage loads are burdened. Unfortunately, most of the IoT devices own low-capacity. To allow devices with capacity limitations directly participate in blockchain to maintain high-security assurance, some blockchain platforms also provide lightweight blockchain endpoint, called *Light client*. Hence, the IoT device serves as the light client is possible to join the blockchain network directly with delegating the works of mining (or called block validating) and the block data synchronization to the full node.

III. BLOCKCHAIN-ASSISTED USER-MANAGED ACCESS

This section describes the design issues that must be addressed when designing UMA mechanisms for B-IoT. Then, we present a schema called Blockchain-assisted *User-Managed Access* (B-UMA) and how B-UMA deals with the issues.

A. Design Issues

Due to the distinguished features of the blockchain (transparency and immutability), when designing UMA in an IoT environment with blockchain, there are several issues to be considered:

- The cost, scalability, and security could be considerably affected when considering different transmission methods. The on-chain transmission method (dispatching and transmitting the data using the blockchain) can take full use of the feature of blockchain.

Authorized licensed use limited to: National Cheng Chi University. Downloaded on April 01,2021 at 07:20:16 UTC from IEEE Xplore. Restrictions apply.

On the contrary, the off-chain transmission method (dispatching and transmitting the data without blockchain) could have better performance on the cost and scalability.

- The data transmitted using the on-chain transmission method may be exposed owing to the transparency feature of the blockchain. Therefore, the confidentiality of the data should be considered when transmitting authorize or private data. (e.g., claim, access token) In practice, a privacy way of on-chain data transmission should be introduced.
- The programming logic of the smart contract can't be modified once being deployed to the blockchain owing to the immutability feature of the blockchain. If the policies implemented in the smart contract change, a new smart contract should be deployed again. The reason mentioned above makes the B-UMA system inflexible. Hence, the developer should consider manageability when designing smart contracts.
- The transmission protocol of the blockchain platform is incompatible with the typical protocol of UMA. In practice, when the transmission is considering utilizing the on-chain method, the specific format of the typical UMA should be transferred to the compatible one.

B. System Architecture

The overview of the B-UMA system is shown in Fig. 1. The components and their responsibilities are designed according to the UMA specification. The difference is that some components are now realized using the smart contract mechanism supported by the underlying blockchain. In B-UMA, every participant owns at least one blockchain account (a public/secret key-pair) which is used to represent the unique ownership of a given protected resource. In this way, the components (smart contracts) can identify RqP or RO by verifying their blockchain accounts. As mentioned, to take advantage of the decentralized feature of the blockchain, the functionalities originally provided by AS in UMA is realized using smart contracts. Each component is supported by one blockchain endpoint. The RS, namely, the IoT device in the B-IoT, is serving a "lightweight endpoint (light client)" because of the limitation of capacity. A light client is a blockchain lightweight endpoint without mining capability (i.e., the capability of creating new blocks), which must depend on a normal endpoint. As shown in Fig.1, all components except the Iot device (resource server) run full node (the normal fully functional blockchain endpoint). As a result, all the components can participate in the blockchain network directly to avoid data be tempered by or exposed to the malicious third party sniffing the off-chain network link. However, as can be observed from Fig.1, the connections from RqP and RO to the components are off-chain, which are not protected by the blockchain security mechanism. To ensure the security of the off-chain method, the use of the point-to-point transportation layer secure mechanism such as HTTPS is needed.

In UMA, there are two main tasks for AS: managing the resource set as well as authorizing the access request base on the policy defined by RO. Because these tasks are realized by the smart contract in B-UMA, the flexibility and cost of modification of deployed smart contracts should be considered. Since the programming logic of the smart contract can't be modified once it has been deployed, the design pattern of decoupling of static data and programming logic into separate smart contracts will be a great solution[13]. In B-UMA, we design two smart contracts, called the Resource Management Contract (RMC) and the Authorization Contract (AC), segregating the storage of resource set from authorization logic. In this way, the considerable cost of migrating the resource information already stored in the original smart contract to the new version can be reduced whenever the policy of B-UMA has been changed. In practice, RO can not only register and manage the resource in RMC but define and modify the policy in AC. Generally speaking, RO pays the role of the owner of both RMC and AC.

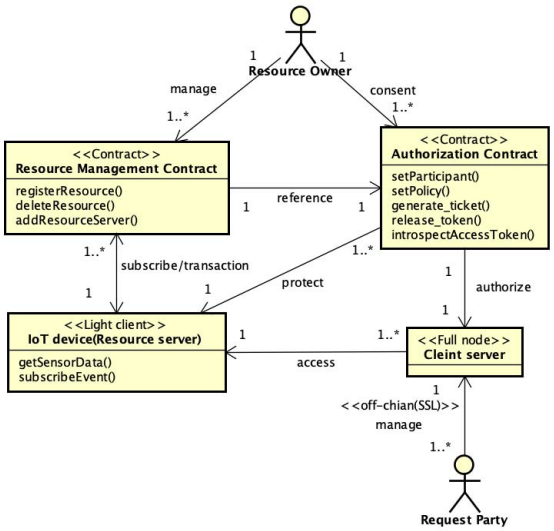

Fig. 1. Overview of B-UMA

C. Execution Flow

As described in the UMA specification, the access control process has been divided into three phases ("Protecting a Resources", "Getting Authorization" and "Accessing a Resource"). This section presents the flows of each phase work in the B-UMA. The detailed flow (see Fig. 2) of the first phase (Protecting a Resources) is listed and explained below.

(1) *RO deploys the RMC and AC*: The RO should be the creator of the RMC and AC. When developing the smart contract, RO has to set himself/herself as the administrator that has permission to invoke all the function of the smart contract. Moreover, from the UMA point of view, RS has to be verified by AS by providing a Protection API Token (PAT). In the B-UMA, the PAT is replaced by a unique blockchain account. Therefore, RO has to allow the RS's blockchain account to invoke specific functions (e.g., registerResource) by setting the permission in RMC in advance. On the other hand, AC needs to check the information about protected-resource in RMC. Hence, RO has to set the deployed RMC's reference (i.e., contract address) when deploying AC.

Authorized licensed use limited to: National Cheng Chi University. Downloaded on April 01,2021 at 07:20:16 UTC from IEEE Xplore. Restrictions apply.

(2) *RO registers the protected resource to the RMC though RS*: In the B-UMA, RO is in charge of registering and managing the protected resource's information in RMC through RS. When sending a transaction to invoke the function (registerResource) of RMC, the *resource name* (name of the protected resource) and *scope* (the scope of the protected resource that allowed to access) should be included as the parameter.

(3) *RO sets the policy corresponding to the specific protected resource to the AC*: After registering the protected resource to the RMC, RO can set the corresponding policy to AC. According to the UMA specification, the RqP should provide the claim to satisfy the policy defined by RO. In our proposed schema, the policy including claim and hint. In practice, RO has to set the claim which expects the RqP to provide while requesting the authorization. Meanwhile, the hint to prompt the RqP which information has to provide should be accompanied.

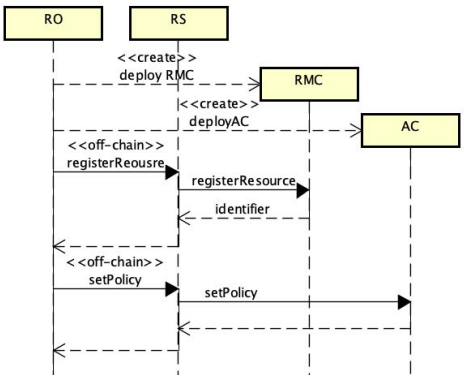

Fig. 2. Flow of Resource Protection

After finishing the first phase, RqP can start to request to access the resource of the RS through the client server. Before the request allowed, RqP needs to obtain the access token from the AC. The detailed flows (see Fig. 3 and Fig. 4) of the second phase (Getting Authorization) is listed and explained below.

(1) *Client Server Attempts to Access Protected Resource*: In B-UMA, when attempting to access the protected resource (e.g., when the RqP clicks the button of web service provided by the client server to request for the resource), the client server has to get authorized from AC in advance. First, once receiving a request without accompanied with the access token, RS registers request permission with the AC. Then, AC has to check whether the related resource has been registered in RMC or not. Besides, AC should verify the blockchain account of RS to avoid the malicious attack. Finally, AC would return unique request identifier, permission ticket and hint which has been defined in the first phase. Lastly, RS responds to the permission ticket, hint and AC's address through HTTPS protocol. (The response format corresponding to the UMA specification is recommended)

(2) *Client Server Seeks Authorization for Access*: After receiving the response from AC, the client server will ask the RqP to provide claims (e.g., phone number) related to the hint. Then, the client server will start to request the access token from sending a transaction to AC accompany the parameters including claim and permission ticket. It is worth mentioning that the plaintext of the parameters is passed to the local blockchain endpoint and be hashing in the smart contract. Therefore, the plaintext of the parameters won't be exposed to the third party. When the transaction has been sent to the blockchain, AC will start to verify the claim and permission ticket. If the verification succeeded, a unique access token will be generated. By the way, each access token should have its own expiration date. Meanwhile, the access token will be mapping to the request identifier and RqP's account through the data structure (mapping) of the smart contract. On the other hand, if the verification failed (e.g., invalid permission ticket), AC would stop and rollback the transaction.

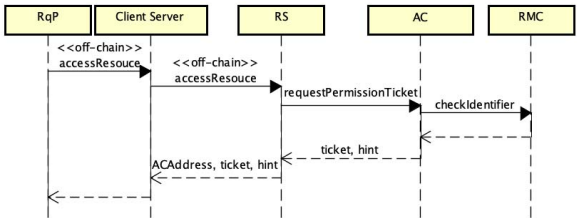

Fig. 3. Follow of Client Attempts to Access Protected Resource

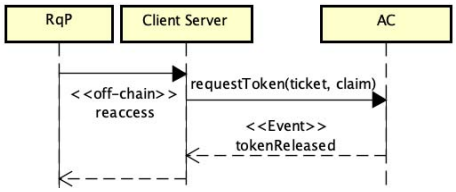

Fig. 4. Flow of *Client Server Seeks Authorization for Access*

Finally, when the client received an access token, RqP can start to access the protected resource. The detailed flow (see Fig. 5) of the third phase (Accessing a Resource) is listed as follows.

(1) *Client Server Sends a Resource Request*: To access the resource, the client server can send and token-accompanied resource request to the RS by means of the HTTPS protocol. To check whether the token provided is valid, according to UMA specification, the *token introspect* flow has been defined. However, in B-UMA, considering the transparent feature of blockchain, the access token generated by the smart contract is possible to be exposed to the third party. (The third party may use the access token to access protected resource) Therefore, we introduce a signature recover mechanism (i.e., ecrecover [14]) provided by the smart contract to ensure that the access token is sent by the authorized user (by verifying the blockchain account) as follow. First, before sending a resource request, the client should sign the access token by the blockchain private key owned by RqP. Then the client sends the resource request accompanied both signature and access token to RS. When receiving the request, RS invoke the introspect function provided by AC to validate the access token and the blockchain account which signed the token.

(2) *RS Responses for the Resource Request*: If the validation succeeds, RS would return the requested resource to the client server by means of the HTTPS protocol. On the contrary, if the

Authorized licensed use limited to: National Cheng Chi University. Downloaded on April 01,2021 at 07:20:16 UTC from IEEE Xplore. Restrictions apply.

validation failed, the failure message (e.g., invalid_token) would be returned to the client server by RS.

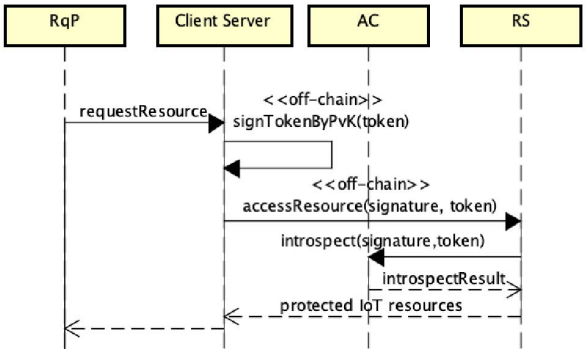

Fig. 5. Flow of Client Server Requests Resource with access token

IV. Evaluation

A. Feasibility

To verify the feasibility of the proposed B-UMA mechanism, we adopt the use case of the "Smart factory" and implement the prototype system accordingly. Fig. 6 shows the structure of the prototype system. The supplier (RO) of the smart device (RS) authorizes the factory member (RqP) to access the service (e.g., temperature and humidity data) from smart devices. Besides, to implement the authorization process and manage the smart devices, the supplier has to realize through the cloud server. On the other side, to access the service, the factory member has to send a request thought the edge server (client server) in the factory.

In the prototype system, we use Ethereum as the underlying blockchain platform. The JavaScript-based Koa2 [15] framework is used to implement the backend process providing RESTful API. TABLE II. shows the components of the prototype system. To construct the private chain network, every component is armed with a blockchain endpoint (i.e., Go-ethereum). Due to the limitation of the low-capacity environment, the smart device is serving as a light client that needs to synchronize the block data with the full node on the edge server. Besides, we employ web3.js (an JavaScript module) to connect the backend process with the blockchain endpoint and adopt Solidity [16] to support as the smart contract language.

The supplier can deploy the RMC and AC to the blockchain through the web page (see Fig. 7-A) provided by the cloud server. Then the supplier can register the resource information (i.e., name, scope) of the smart device to the RMC. After registration, the identifier of the resource responded by the RMC can be checked on the web page. Lastly, the supplier can set the policy of the corresponding resource. On the other side, the factory member can synchronize registered resources and further send an access request to the smart device through the web page (see Fig. 7-B) provided by the edge server. During the authorization process (the Getting Authorization phase), the web page will alert the factory member to provide the member ID (i.e., claim). After getting authorized, the factory member can continuously access the resource of the smart device until the access token expired.

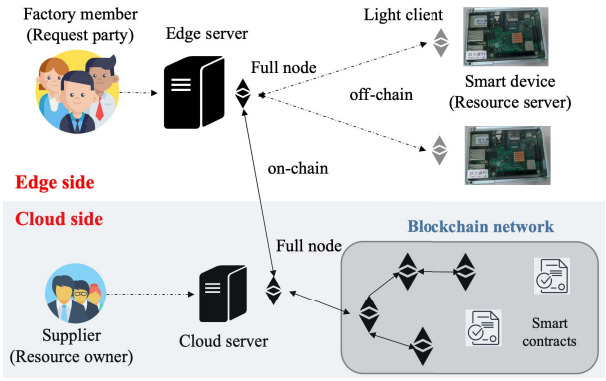

Fig. 6. Structure of the Prototype System

TABLE II. Components in Smart Factory

Component	*OS*	*Blockchain client mode*	*Role in use case*
Raspberry Pi B3+	Raspbian GNU/Linux 9.4	light client	Smart device
PC1	Ubuntu 18.04	full node	Edge server
PC2	macOS 10.15	full node	Cloud server

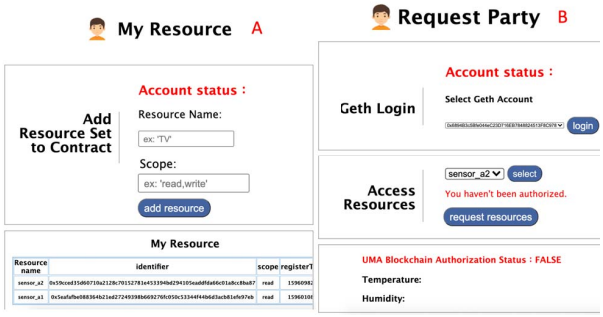

Fig. 7. The prototype system. (A) Resource owner view; (B) Request Party view

B. Cost

To measure the cost when invoking the function of the smart contract (i.e., RMC, AC), the consumption of gas (transaction fee of Ethereum) of each phase is provided. Developers can evaluate whether they should spend real money to build the B-UMA system underlaying the public chain or not. Generally speaking, the advantage of using services on the public chain is that the cost of the equipment and maintenance of constructing the private chain can be saved. Moreover, in February 2020, there are about 7,000 Ethereum nodes distributed around the world [17]. Compared to constructing a private chain within the organization, it can better achieve the ideal decentralized mechanism. However, the TPS (transaction per second) of the public chain could be slower than the private chain (TPS of the private chain can be lower by modifying the difficulty of the consensus). According to [18] , in February 2020, the standard speed to perform a transaction of the Ethereum public chain is about 23-second. (The gas price is about 5 gwei which means that it cost about $0.02 each 21,000 gas is consumed).

Authorized licensed use limited to: National Cheng Chi University. Downloaded on April 01,2021 at 07:20:16 UTC from IEEE Xplore. Restrictions apply.

TABLE III. illustrates the cost in each B-UMA phase. Apparently, most of the gas is consumed in the "resource protect" phase because of the operation of smart contract deployment. However, all the operation has only need to be done once. On the other hand, the operations in the "request authorization" phase are cheap. Most of the gas is consumed due to the generation of authorization data and emission of the event. However, the more time the access requests are issued, the more gas is spent. Lastly, in "resource access" phase, no GAS should be spent since the invocation of smart contract cause no state change (query without sending any transaction).

TABLE III. GAS CONSUME FOR EACH PHASE

Phase	*Gas consume*	*Price in USD*	*Percentage*
resource protectionx (deploy RMC and AC)	1813053	$1.97	82.6%
resource protection (registerResourceSet and setPolicy)	227384	$0.25	10.4%
request authorization	154528	$0.17	7.0%

Price in USD = 5(gwei) * Gas consume * 0.00000021;
Percentage = Gas consume/Total Gas consume

C. Discussion

To compare the qualitative performance of B-UMA with other related access control frameworks discussed previously, we present an evaluation metric (see TABLE IV), which compares the performance between 4 types of access control frameworks. Decentralization indicates both the transparency of the system and whether the problem of a single point of failure can be solved; Maintainability means the flexibility of system once the authorization policy or hardware needs to be updated; Complexity explains the difficulty and the time consumption to design or build a system; Economy implies the cost (e.g., electronic, computation, money) when running the system.

TABLE IV. A SUMMARY OF B-UMA

Features/Mechanism	*Basic*	*OAuth2*	*UMA*	*B-UMA*
Decentralization	+	++	+++	++++
Maintainability	+	++	+++	++++
Complexity	++++	+++	++	+
Economy	++++	+++	++	+

Performance: ++++= best, +++= well, ++= normal, += worst

V. CONCLUSION

Currently, it is still common to rely on third-party services to realize the access control mechanism in the IoT system. However, a centralized system may face challenges such as low reliability, transparency, and maintainability. Fortunately, the blockchain is regarded as a technology to solve the related problem of a centralized system. This paper, in a nutshell, proposed a "Blockchain-assisted User-Managed Access" schema in the IoT field base on the UMA specification. The proposed schema alleviated the reliance on centralized authorization server. Moreover, the transparency and traceability of the authorization process were improved since all the components in the proposed schema can interact with the smart contract by joining the blockchain directly. To increase the maintainability of UMA, the resource owner can not only own the smart contracts but manage the authorization policy he/herself. Last, the cost of operating the smart contract is provided, the developer can evaluate to construct the system underlying the public or private blockchain. On the other hand, since the blockchain technology is in the development stage, there are still several issues to employ blockchain to the IoT system, such as scalability and stability. Furthermore, developers should consider the limitation of complexity and economy when designing the B-UMA system. In the future, we expect more feasible blockchain consensuses for IoT will be proposed and will try to consider implementing proposed schema in different blockchain platforms.

ACKNOWLEDGMENT

This work is partially supported by Ministry of Science and Technology, Taiwan, under grant 109-2221-E-004 -004 -.

REFERENCES

[1] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper,* vol. 1, no. 2011, pp. 1-11, 2011.
[2] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks,* vol. 54, no. 15, pp. 2787-2805, 2010.
[3] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi, and K. Salah, "A user authentication scheme of iot devices using blockchain-enabled fog nodes," in *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*, 2018: IEEE, pp. 1-8.
[4] Kantara Initiative. "User-Managed Access (UMA) Core Protocol draft-hardjono-oauth-umacore-00." https://tools.ietf.org/html/draft-maler-oauth-umagrant-00 (accessed August 8, 2020).
[5] M. P. Machulak, E. L. Maler, D. Catalano, and A. Van Moorsel, "User-managed access to web resources," in *Proceedings of the 6th ACM workshop on Digital identity management*, 2010: ACM, pp. 35-44.
[6] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *Ieee Access,* vol. 4, pp. 2292-2303, 2016.
[7] A. Z. Ourad, B. Belgacem, and K. Salah, "IOT Access control and Authentication Management via blockchain."
[8] Kantara Initiative, "Kantara Initiative." [Online]. Available: https://kantarainitiative.org/.
[9] Kantara Initiative. "User-Managed Access (UMA) Core Protocol draft-hardjono-oauth-umacore-00." https://tools.ietf.org/html/draft-maler-oauth-umagrant-00 (accessed February 4, 2020).
[10] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[11] V.Buterin. "A Next-Generation Smart Contract and Decentralized Application Platform." https://github.com/ethereum/wiki/wiki/White-Paper (accessed February 4, 2020).
[12] F. L. Viktor Trón. "Go-ethereum." https://github.com/ethereum/go-ethereum (accessed February 4, 2020).
[13] M. Wöhrer and U. Zdun, "Design patterns for smart contracts in the ethereum ecosystem," 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)*, 2018: IEEE, pp. 1513-1520.
[14] "Security Considerations in Solidity." https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#abstraction-and-false-positives (accessed August 8, 2020).
[15] "Koa.js." https://github.com/koajs (accessed August 8, 2020).
[16] "Solidity." https://solidity.readthedocs.io/en/v0.5.13/ (accessed August 8, 2020).
[17] "Etherscan." https://etherscan.io/ (accessed February 4, 2020).
[18] "Eth Gas Station." https://ethgasstation.info/ (accessed February 4, 2020).

Authorized licensed use limited to: National Cheng Chi University. Downloaded on April 01,2021 at 07:20:16 UTC from IEEE Xplore. Restrictions apply.