Multi-authority Decentralized Attribute-Based Authorization Framework

Sok, Kimheng; COLIN, Jean-Noel

Published in:
Advanced Information Systems Engineering Workshops. CAiSE 2022.

DOI:
10.1007/978-3-031-07478-3_2

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

Multi-authority decentralized attribute-based authorization framework

Kimheng SOK[1][0000−0002−2152−9590], Jean Noël COLIN[1][0000−0003−4754−7671], and Kimtho PO[2]

[1] University of Namur, NaDI Research Institute
Namur, Belgium
kimheng.sok@student.unamur.be, jean-noel.colin@unamur.be
[2] Institute of Technology of Cambodia, Phnom Penh, Cambodia
pokimtho@itc.edu.kh

Abstract. The advancement in collaborative work among multiple institutions results in tremendous data sharing and, at the same time raises concerns about data security and privacy. Current systems are mostly built upon a centralized model, so the system requires one authority to concede another authority or mitigate trust to a trusted third party.

In this paper, we propose a multi-authority decentralized attribute-based authorization framework illustrated in an eHealth scenario that provides resilience and transparency, enforces data privacy, and allows each authority to enforce their access policy on the shared data. The framework operates on Blockchain technology and multi-authority attribute-based encryption. Our framework protects against malicious single authority who aims to grant permission to users to the shared data. Moreover, we optimize the user key structure that could reduce the key size of the key set by a factor of up to n! where n is the number of attributes in the user key to eliminate brute-force attacks. We implement the framework, show the system performance, and prove the efficiency of our proposed framework.

Keywords: Security · Privacy · Authorization Framework · Blockchain · D-MA-ABE.

1 Introduction

1.1 Motivation

The reason we choose eHealth as a scenario for proof of concept is that the health care sector is a multi-authority industry that includes: hospitals, laboratories, radiologists, pharmacies, patients, insurance companies, financial institutions, government agencies, research institutions, partners, other referral health care providers, donors, suppliers, auditors, and regulators. Health care data is one of the valuable and sensitive assets to be protected. That is why health care providers are required to comply with certain regulations depending on

their regions such as Health Insurance Portability and Accountability Act 1996 (HIPAA) [1], General Data Protection Regulation 25 May 2018 (GDPR) [2].

Most of the designs of the system so far are using a centralized governance approach to manipulate access to data. There is a central authority that creates user accounts, sets user roles, and manages access policy by granting access rights to each user, resulting in some drawbacks such as centralized decision making and management overhead. Moreover, due to the centralized nature of the system, it is subject to a single point of failure. Therefore, we aims to build a system in a decentralized way to answer the problems above by using Blockchain technology [3]. Blockchain technology provides many benefits, firstly, it maintains the integrity of the data due to its immutable feature. Secondly, historical data inside the Blockchain could also be served as an audit log. Thirdly, Blockchain allows multi-party to join the network for collaborative work and data sharing. Lastly, a smart contract that deploys inside Blockchain nodes could serve as access control. In addition, we also use encryption techniques to preserve data privacy.

1.2 Our contributions

In this paper, we propose a multi-authority decentralized attribute-based authorization framework. The main contributions of this paper are summarized as follows:

1. **Design multi-authority decentralized access control architecture**, which allows multi-authority to set a joint access policy on the shared data.
2. **Propose decentralized multi-authority attribute-based encryption (D-MA-ABE) algorithms** for privacy enforcement and protection.
3. **Optimize user key structure** to reduce the brute-force attack vector.
4. **Implement the proposed framework** for proof of concept and performance evaluation.

1.3 Paper organization

The rest of the paper is organized as follows: section 2 presents background information; Section 3 provides an overview of related work; section 4 shows our approach, and overall architecture; section 5 demonstrates our implementation and performance evaluation; finally, section 6 deliberate the conclusion and future works.

2 Background information

2.1 Notations

Symbols	Description	Symbols	Description
AA	Attribute Authority	$Uattr$	User attributes
DO	Data Owner	UK_i	User sub key from AA_i
DU	Data User	UK	User Key
CSP	Cloud Service Provider	PKs	All public key of AAs
λ	Security Parameter	K	AES Key
PP	Public Parameter	AP	Access Policy
SK_i	Secret Key of authority i	M	Message
PK_i	Public Key of authority i	CK	Cipher Key
Uid	User identity	CT	Cipher Text

There are four main actors in the system:
AA (Attribute Authority): An AA is in charge of issuing a user subkey that contains certain attributes for data users (DU).
DO (Data Owner): A DO is an entity that owns and controls the data to be shared. A DO specifies access control policies for the data it shares, there could be multiple DOs in our system. A DO could also be an AA, but AA is not necessary to be a DO.
DU (Data User): A DU is an entity that wants to access the data shared by DOs.
CSP (Cloud Service Provider): A CSP is an entity that hosts the encrypted files of DOs.

In order to understand how we design our system; we need to have a basic understanding of Attribute-based encryption and Blockchain technology.

2.2 Attribute-Based Encryption

There are several types of attribute-based encryption (ABE) systems, one is Key-Policy (KP-ABE) [12] [13] , and another one is Ciphertext-Policy (CP-ABE) [14]. For example, we encrypt a message with an access policy *"user.role=doctor and user.org=hospital1 and user.group=emergency_staff"*, so the user needs to request a user key from their attribute authority. In this case, the hospital if their user key contains the attributes {*hospital1, doctor, emergency_staff*}, user can access the data, otherwise not. Most ABE scheme relies on a centralized approach of using Key Generation Center (KGC), while others rely on a decentralized approach such as multi-attribute authorities ciphertext-policy attribute-based encryption (MAA-CP-ABE) [15]. Our attribute-based encryption scheme is extended from the work in [15]; we will discuss the extension of the algorithm in section 4.2. Our encryption scheme is applied in a decentralized way called by the user application and smart contracts in the Blockchain network, we called it Decentralized Multi-Authority Attribute-Based Encryption (D-MA-ABE).

2.3 Blockchain and smart contract

Blockchain [3] is a peer-to-peer distributed ledger technology that has immutability, transparency, and distributed property which could be used to ensure the integrity and availability of the system. Smart contract [4] is a computerized transaction protocol that executes the terms of a contract; it is the computer programs business logic that executes when the condition is verified, and it can be used to set an agreement between multi-authority for access control privilege without a trusted third party in the middle. To understand more about Blockchain technology opportunities and challenges please refer to [16]. There are different types of Blockchain. Public Blockchain where everyone is allowed to join the network, private Blockchain which only authorized members are allowed to join. We choose Hyperledger Fabric [7], as it is a private Blockchain, which some people consider as a consortium Blockchain. It is built for the enterprise by design that allows multiple enterprises to work together as a consortium. We also choose the attribute-based access control (ABAC) model suitable for defining a fine-grained access policy as well as the complex access control rules [17].

3 Related Work

Blockchain technology has become popular; the use of smart contracts as an access control has gained more attention from research communities. There are several papers targeting the same use case for secure data sharing [5] [6] both papers are using Hyperledger Blockchain [7] with the eHealth use case. The idea in [5] is to create a Medichain that acts as another trusted third party storage and policy enforcement service between patient, caregiver, and medical clinic by granting full control to the patient, who is the data owner. It is a patient-centric health system that allows patients to be the owner of data and set access policies to their own data. If the patient is unable to perform the task by him/herself, the patient can designate the role to the caregiver to become the data owner. [6] is also using Hyperledger Blockchain together with the edge node for storage. Hyperledger access control list is used to verify the user identity, and attribute-based access control (ABAC) is used to enforce data on the edge node. The patient is the data owner who sets the policy on their data, and the doctor is a data user who requests to read the data. Both papers were using Hyperledger composer to create an access control policy that was already deprecated and declared the end of life in August 2021 by the Hyperledger community. None of the maintainers are actively developing new features. None of the maintainers are actively providing support via GitHub issues. In [8] [9], both papers use Ethereum Blockchain [10] and attribute-based access control (ABAC) for policy enforcement, but they have distinct approaches for creating their access control model; in [8] was built on the eXtensible Access Control Markup Language (XACML) Components that operate together with Ethereum smart contract. Most of the research works focus only on a single data owner to protect the confidentiality of their own data, but there have not been many works on multiple

data owners controlling the shared data. On the other hand, [9] there is a concept of multi-authority whose roles are to attest the attributes from data users then provide attribute tokens back. Enough attribute tokens could be utilized to exchange for the encryption key, which is used to decrypt the requested data stored in the shared database. This paper also uses ciphertext-policy attribute-based encryption (CP-ABE) [11] to encrypt the data.

Table 1 shows the comparison between our framework we named it DRACO, and the other related work. We can notice there are different kinds of Blockchains such as Hyperledger [7] and Ethereum [10] that have been used to create authorization frameworks. Real data or encrypted data is usually stored off-chain, in edge node, private resource server (Private RS), share database (share DB), or outsourced to a cloud service provider (CSP). ABAC is the most well-known access control model to be used. Some works did not focus on encryption at rest, while some used different kinds of attribute-based encryption schemes for data encryption such as CP-ABE. In most of the works, there is only a single attribute authority (AA), while the use of multi-authority has grown in attention due to the increasing amount of collaborative work. None of the related work has ever optimized the user key structure.

Table 1. Related work comparison.

Paper	Blockchain	Storage	AC Model	Encryption	AA	Optimize UK
[5]	Hyperledger	Off-chain	DAC	YES	Single	NO
[6]	Hyperledger	Edge node	ABAC	NO	Single	NO
[8]	Ethereum	Private RS	ABAC	NO	Single	NO
[9]	Ethereum	Share DB	ABAC	CP-ABE	Multiple	NO
DRACO	Hyperledger	CSP	ABAC	D-MA-ABE	Multiple	YES

4 Our approach

4.1 Scenario

In order to understand the process of a decentralized authorization framework in a multi-authority environment, let us converge toward an eHealth scenario. The scenario begins when a patient is seeking health care services, with a doctor, who is an employee inside the hospital, and the insurer, who needs to access the billing information of the patient in order to reimburse the insurance fee in which the patient has subscribed. In this simple scenario, there are two big institutions: the hospital and the insurance company, which act as an attribute authority (AA) in our system to provide necessary attributes to their employees. In the patient-centric system, a patient is considered a data owner (DO), and at the same time an attribute authority (AA) too. In the hybrid system there could be multiple data owners as such both the health care institution and the patient

are data owners (DOs), who can enforce their personal preference of the access policy on the patient records. Insurance companies might assign their employee to access patient's billing record so their employee acts as data user (DU) in the system that requests to read the billing information. As information is encrypted using D-MA-ABE, the data user (DU) needs to request all the subkeys from the correspondent attribute authorities to access the information of the patient. The business process could be extended to have multiple roles in each organization, with more than two organizations wishing to join the system.

4.2 System architecture

Our authorization framework composes of five main components: Client Application, Blockchain network, D-MA-ABE Module (Crypt Module), Smart contracts, and Storage as shown in Figure.1. All users in the system such as AA, DO, DU use client applications to communicate with the Blockchain network. First, they need to create an account in the Blockchain to establish the relationship between them. Once they are in a relationship, DU can request user subkeys from AA and use them to request data by calling the smart contract (chaincode) in the Blockchain network. To enforce and protect data privacy, users need to follow the D-MA-ABE protocol, so the D-MA-ABE module (Crypt Module) is implemented and embedded in the client applications and also inside each network peer. Due to the length of the paper, we will only discuss two main components here, which are the smart contract and the D-MA-ABE Module.

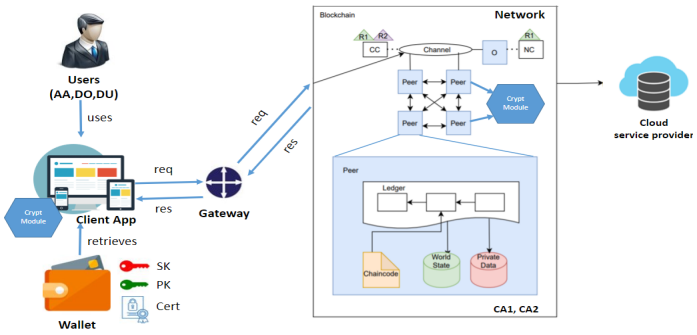

Fig. 1. Multi-authority decentralized authorization framework architecture.

Smart contract In Hyperledger Blockchain, a smart contract is also called chaincode, which is the program running on the Blockchain network. In Figure.1. we illustrate four peers in which the smart contracts are installed, instantiated, and run. We create three smart contracts called AdminCC, AuthzCC, CryptCC.

AdminCC is responsible for administrative tasks such as account creation, relationship establishment (ex. care circle, employment), system initialization, and setup phase. AuthzCC is responsible for authorization tasks such as key generation, merging user subkeys, and decryption phase, while CryptoCC is responsible for data encryption. The way we design smart contracts is for software modularity and separation of functionality and data sensitivity. The interaction of smart contracts could be found in Figure.2.

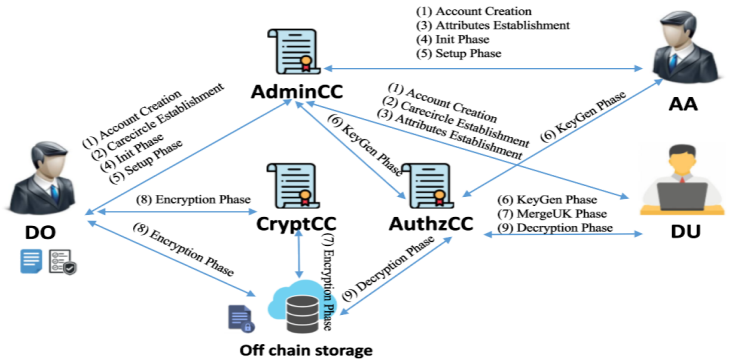

Fig. 2. Smart contracts architecture.

D-MA-ABE Module decentralized multi-authority attribute-based encryption scheme (D-MA-ABE) follows the construction from the MAA-CP-ABE scheme [15] which consists of five main phases: GlobalSetup, AuthSetup, KeyGen, Encrypt, Decrypt. The scheme was not intended to use with Blockchain Technology. Therefore, in order to adapt to the decentralization workflow, we have extended the scheme in many aspects. Firstly, we separated the KeyGen phase into two different phases. Secondly, we modified the input and output of some algorithms in the scheme, Thirdly, we modified the encryption and decryption scheme so that we will get the constant size of cipher key. Fourthly, we optimized the user key structure. Lastly, we implemented smart contracts to interact with all the algorithms in the D-MA-ABE module.

Our D-MA-ABE scheme consists of six main phases: Init, Setup, Key Generation, Merge User subkeys, Encrypt and Decrypt. The details of the six-phase will be described in section 5.1.

5 Implementation

5.1 D-MA-ABE Algorithms

Below are the D-MA-ABE six phases algorithms:
Suppose that we have a set of multi-authority $A = \{AA_i, AA_j ..., AA_n\}$, a message M is the shared data that all attribute authorities want to set access policy on. P_i is the policy of AA_i on message M and P_j is the policy of AA_j on message M. $AP = \{P_i, P_j, ..., P_n\}$ is the access policy set of all attributes authorities.
i/ Init:

$$Init(\lambda) : PP \tag{1}$$

Init algorithm takes a security parameter (λ) which is the size of the order in bits and outputs Public Parameter (PP). One of the attribute authorities is called the init function and broadcasts output to all the relevant attribute authorities.
ii/ Setup:

$$Setup(PP, i, \alpha, y) : SK_i, PK_i \tag{2}$$

Each time a new authority is introduced, setup will generate a random number α and y to create the secret key (SK) then produce the corresponding public key (PK) which also includes the authority (AA_i) with identity (i) in it, so that we can verify the public key belongs to which authority. Then, each authority can generate a subkey embedding relevant user attributes.
iii/ KeyGen_i:

$$KeyGen\_i(PP, SK_i, Uid, Uattr_i, r\_arr) : UK_i \tag{3}$$

The KeyGen stands for key generation, which is an algorithm that takes a Public Parameter (PP), the Secret Key (SK_i) of a particular authority (AA_i), User identity (Uid) that want to access information, user attributes ($Uattr_i$) that attribute authority provides to the user and a random seed for cryptographic uses (r_arr) and output the User subkey (UK_i). Users that want to access the restricted data need to request a User Key from the attribute authority. In case there is a multi-authority enforced access control policy on the shared data, the user needs to request a user subkey (UK_i) from each attribute authority.
iv/ Merge_UK:

$$Merge\_UK(\{UK_i, UK_j, ..., UK_n\}) : UK \tag{4}$$

Merge_UK is a function that takes all the user subkeys from each authority which belongs to the user and merges them together to generate a single user key (UK). Each user subkey UK_x where $x = \{i, j, ..., n\}$ is in the form of:
UK_x ={'GID':gid, 'keys':{attr1:{'K':'val', 'KP':'val'},attr2:{'K':'val', 'KP':'val'}}
each UK_x contains user identity gid and multiple attributes in random order $attr1, attr2$ with their respective cryptographic value K and KP. once merge together the final user key also contains multiple attributes from all the attribute authorities in random order as well. All possible positions of each attribute inside the user key creates a valid key set by a factor of n! where n is the number of

attributes inside the user key. We optimize user key structure by sorting all key attributes in alphabetic order, so only a single key is valid during key validation. By doing so, we reduce the key size of the key set by a factor of n!.

v/ Encrypt:

$$Enc(PP, PKs, K, AP, M, s, w, tx\_r\_arr, path) : CK, CT \tag{5}$$

Encrypt is an algorithm that encrypts messages (M) with access policy (AP) where $AP = \{P_i,P_j,...,P_n\}$ and produces ciphertext (CT). In theory, we want to encrypt (M) the sensitive data such as patient personal information, patient health information, or financial information with the access policy (AP) we want to enforce on that data. But in practice, encrypting the real data with access policy from different organizations consume a lot of processing time and storage depending on the size of the message, for this reason, we divide our encrypt algorithm into two steps, step one we use symmetry key (K) which is used to encrypt the message (M) to produce ciphertext (CT), Ciphertext will be store off-chain mentioned by the variable ($path$). Then, step two we encrypted the key (K) with a multi-authority attribute-based encryption scheme with an access policy (AP) of multi-authority so that our encrypt algorithm always take a constant time as the size of the key to be encrypted is always the same. To encrypt the key (K) we need the public parameter (PP), an access policy (AP), the public key of all authority (PKs), the symmetry key (K) that we want to encrypt, (s) secret share, (w) zero share, and some random seed will be used for the cryptographic purpose (tx_r_arr) and output the Cipher Key (CK) that we store on the blockchain.

vi/ Decrypt:

$$Dec(PP, UK, CK, CT) : K, M \tag{6}$$

Decrypt is an algorithm that decrypts (CT) to get the message (M). We also divide our Decrypt algorithm into two steps. Step one the algorithm decrypts the cipher key (CK) by using the public parameter (PP) and the user key (UK) to obtain key (K), and then step two it uses the key (K) to decrypt cipher text (CT) that retrieved from the off-chain storage to get the real message (M).

5.2 Evaluation

In this section, we want to evaluate the strength of the algorithm with the threat model that we will define next; we prove that the system is preserving the security and privacy of information in the smart health system, and show the strength of the system in keeping malicious users away from revealing the secret data, let us see some attack scenarios below:

Threat model

i. Honest-but-curious cloud storage provider When we upload data (patient health records) or other sensitive information in the cloud, we might trust our cloud storage provider but they might be honest-but-curious to see, share, or sell our data without our consent. These issues could be solved by encrypted data at rest with our decentralized multi-authority attribute-based encryption.

Thus, the data has been encrypted before being sent to the storage; without a user key from multi-authority, no one could be able to decrypt the data and view it.
ii. Malicious single authority Single attribute authority could maliciously generate a user sub key for any unauthorized user. In this case, the shared data is protected by the joint-access policy. Malicious authority could only generate the key which contains attributes that he/she is responsible for but not the attributes that belong to other authorities.
iii. Malicious attacker wants to modify access policy To decrypt the message, we need to have a user key with enough attributes or modify the policy, so that the enforcement is not too restricted. In that case, the malicious attacker could not do it because the hash of the encrypted message and the cipher key is stored in the Blockchain for transparency. Modifying encrypted messages or access policy will result in invalidating the hash value stored in the Blockchain ledger that maintains the integrity of the data, which is enforced by the consensus algorithm during the validating process.
iv. User key collision Inside the user key, there is a list of attributes. Attributes in the user key are used as the decryption criteria. The position of those attributes could be in any random order. In our system, we optimize user key structure by sorting all the attributes before generating the final user key. This technique has never been done so far in any works that we have read. Why it is important to sort those attributes? Because if we did not there might be multiple valid user keys which increases the valid key set that provides a bigger chance for the attacker to compromise the user key.

5.3 Performance

In this section, we want to measure and evaluate the speed of the algorithm in two different cases. The first case runs the algorithm in a centralized model without using Blockchain, and the second case uses Hyperledger Blockchain. To measure the performance of our implementation, we used a laptop with Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz and 16.0GB RAM.We run Hyperledger farbic stable version 1.4 first network as docker containers in virtual machines with Ubuntu operating system version 18.04, with pythons version 3.6 and a standard SS512 pairing group for attribute-based encryption. We show the comparison of the computing time without and with using a Hyperledger Blockchain network.

Figure. 3. shows the computing time in milliseconds of the six algorithms without using the Blockchain network. Init, setup, and merge user sub key algorithms take only 2 to 5 milliseconds, while key generation, encryption, and decryption algorithms take exponential time depending on the number of attributes used. We show the computing time using 5, 10, 20, and 40 attributes.

Figure.4. shows that for all the six algorithms it takes approximately 4 to 5 seconds in addition to running on Hyperledger Blockchain. This additional com-

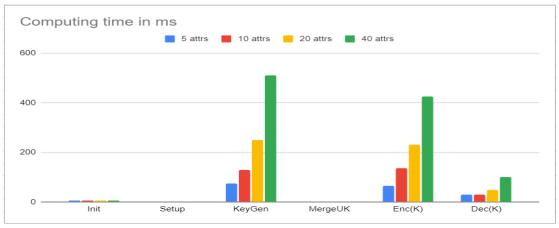

Fig. 3. Computing time without Blockchain.

Time(ms)	PP	Setup	Keygen	MergeUK	Enc	Dec
Without Blockchain	5	2	73	0	64	29
With Blockchain	5155	4817	4418	4671	5906	4842

**Blockchain (hyperledger fabric)*

Fig. 4. Computing time using blockchain.

puting cost is spent on the communication between peers in the Blockchain network. It is the cost that we need to pay for security such as system availability, data integrity, transparency, traceability, and multi-authority system.

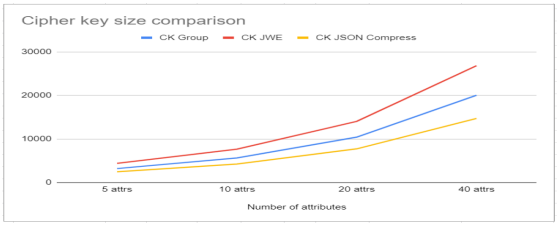

Fig. 5. Cipher key size comparison.

Cipher Key (CK) could be stored using different types of formats as shown in Figure.5. The JSON compressed version consumes less memory than the pairing group operation; and the JSON Web Encryption (JWE) consumes higher memory.

6 Conclusion

In this paper, we presented a decentralized multi-authority attribute-based encryption framework and built a prototype, and implemented our D-MA-ABE algorithm and wrote three smart contracts AdminCC, AuthzCC, CryptCC for administration, authorization, and for data encryption, respectively. By using

Blockchain technology and smart contract, we can eliminate central authority from the system, and enable multi-authority to work collaboratively. The integrity of data is maintained by the Blockchain consensus algorithm. Distributed ledger serves as the source of truth for transparency and audit. We also presented the threat models and performance of our framework. Our framework allows multi-authority to have equal control over the shared data, which is absent in many current systems. We have optimized the user key structure to protect against brute-force attacks which have never been done before. For our future work, we want to reduce network communication costs by performing operations in a localized way. All the source code will be available in our Github repository https://github.com/KimhengSOK. We also explore the possibilities of using Self-Sovereign Identity to present claims which involve additional stakeholders. Thanks to ARES, UNAMUR, and ITC for support.

References

1. Health Insurance Portability and Accoutnability Act HIPAA. https://www.hhs.gov/hipaa/index.html. Last accessed 7 Sep 2021
2. General Data Protection Regulation. https://www.gdpr.eu. Last accessed 7 Sep 2021
3. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. (2008)
4. Szabo, N.: The idea of Smart Contracts. (1994)
5. Adam, S. Sara, R. Luke, B.: MediChain: A Secure Decentralized Medical Data Asset Management System. IEEE (2018) https://doi.org/10.1109/Cybermatics_2018.2018.00258
6. Mark, N. Chien-Chung, S. Hao, G. Wanxin, L.: Access Control for Electronic Health Records with Hybrid Blockchain Edge Architecture. IEEE (2019)
7. Hyperledger homepage. https://hyperledger.org. Last accessed 7 Sep 2021
8. Aura, R. Damiano, P.M.: Blockchain Based Access Control Service. IEEE (2018). https://doi.org/10.1109/Cybermatics_2018.2018.00237
9. Chien-Chung, S. Hao, G. Ehsan, M.: Multi-Authority Attribute-Based Access Control with Smart Contract. ICBCT (2019). https://doi.org/10.1145/3320154.3320164
10. Ethereum homepage. https://ethereum.org. Last accessed 7 Sep 2021
11. Yinghui, Zh. Dong, Zh.: Computationally efficient ciphertext-policy attribute-based encryption with constant-size ciphertexts. In International Conference on Provable Security, pp. 259-273. Springer (2014).
12. Sahai, A. Waters, B.: Fuzzy Identity Based Encryption. pp. 457–473. Springer 3494 of LNCS (2005)
13. Waters, B. Sahai, A.: Attribute Based Encryption for Fine-Grained Access Control of Encrypted Data. ACM (2006)
14. Bethencourt, J. Sabai, A. Waters, B.: Ciphertext-Policy Attribute-Based Encryption. IEEE (2007)
15. Water, B. Rouselakis, Y.: Efficient Statically-Secure Large-Universe Multi-Authority Attribute-Based Encryption. pp. 315–332. (2015)
16. SOK, K. COLIN, J.N. PO, K.: Blockchain and Internet of Things Opportunity and Challenges. pp.150–154. ACM (2018). https://doi.org/10.1145/3287921.3287933
17. COLIN, J.N. Laurent, E.: A Flexible and Centralized Approach for Access Control in Heterogeneous IoT Environment. (2019) https://doi.org/10.4018/IJHIoT.2019010102