SCIS 2019 2019 Symposium on Cryptography and Information Security
Shiga, Japan, Jan. 22 - 25, 2019
The Institute of Electronics, Information and Communication Engineers

Decentralizing the Role of Root Authority in CP-ABE by the Help of Blockchain

Dongyeon Hong[*] Kwangjo Kim[*]

Abstract: Attribute-based encryption (ABE) is classified as an extended public key cryptosystem which replaces the public key of a user with his/her attributes in order to reduce the management overhead of their public keys. Another advantage of ABE is to handle monotonic access control with attributes. Nevertheless, ABE has some disadvantages like a fully trustness on the central component. In this paper, we propose a novel way of Ciphertext-Policy Attribute-based encryption (CP-ABE) by applying features of the decentralized ledger in the blockchain to overcome the current disadvantages of CP-ABE. We provide the role distribution of the central component to achieve a more reliable system and prevent a single point of failure from strong adversaries.

Keywords: Blockchain, Attribute-based Encryption, Ciphertext-Policy, Key Revocation, Key Management, Key Issuing, Access Control.

1 Introduction

1.1 Motivation

Public key cryptography (PKC) is well known and powerful method to share data between users to reduce the load of key management in secret key cryptography using two keys: one is a private key and the other is a public key. Anyone can send data with encryption using a recipient's public key to provide confidentiality of the sending data. Only those who have the secret key corresponding to the public key can decrypt the encrypted data. It looks very simple, efficient and well-fitted. Every time we want to send encrypted data, however, the sender has to check the intended recipient's certificate to inspect that public key belongs to the recipient. If we send or share data with a small group, it is not hard to check all certificates one-by-one. But as the size of the group increases, it gets harder to check. For example, in a university, we want to send clinical data to students or professors. There are a huge number of students or professors and it takes a long time to send the encrypted data. So the overhead comes from sending data to multiple people. Especially sending data to someone who satisfies some conditions like having a driver's license, teaching assistant, *etc.*, becomes more difficult.

Attribute-based encryption (ABE) addresses this weakness easily. Even though ABE is one of PKC, the main difference between typical PKC and ABE is how to generate keys. In ABE, keys are replaced with attributes to make different consequences. Anyone who has some attributes setting previously can recover original data from encrypted data. It is easy to send data to an unspecified number of recipients by determining what attributes are required.

However, ABE still has some problems such as (1) revocation issues, (2) fully trustness on the main server and (3) single point of failure. The problems (2) and (3) happen because of the centralized structure of PKC. In this paper, we will address them by applying a decentralized feature of blockchain to distribute the strong power of central authorities and to make our approach more reliable.

1.2 Outline of the Paper

In Section 2, we introduce the background of ABE and blockchain in brief. Section 3 describes preliminaries for understanding ABE and our model. Then in Section 4, we describe the distributed ABE and define our approach in Section 5. In Section 6, we discuss some issues and then we suggest future work and conclusion in Section 7.

[*] Graduate School of Information Security, KAIST. 291, Daehak-ro, Yuseong-gu, Daejeon, South Korea 34141. { *decenthong93, kkj* }@kaist.ac.kr

2 Background and Related Work

2.1 Attribute-Based Encryption

In 1984, Shamir introduced identity-based encryption [19], which makes any users communicate without key exchange protocol. The message is encrypted using a recipient's identity, like an IP address. Then only the user who has a secret key corresponding to identity can decrypt. The important benefit is minimizing leakage of keys.

Sahai and Waters suggested the first attribute based encryption in 2005 [18]. They combined the encryption and access control schemes. After [18], developing Boneh and Franklin's ideas [2], Goyal *et al.* [11] and Bethencourt *et al.* [1] proposed an improved ABE as a cryptographic fine-grained access control scheme. ABE is divided into three types along access control policy which are Threshold Policy, Key-Policy (KP) and Ciphertext-Policy (CP) access control. In this paper, we focus on the access control on CP.

2.1.1 Ciphertext-Policy ABE

Attributes in ciphertext-policy are associated with keys while access structures are embedded into the ciphertext [21]. In this manner, the data owner can determine who can decrypt. Moreover, if the policy needs to be updated frequently, ciphertext-policy can be more flexible since the data owner only needs to update the access structure in the ciphertext. In general, there are two approaches to construct CP-ABE: (1) bilinear pairing and (2) lattice. In general, bilinear pairing is often used so there are several works [1], [8], [10], [6], [16], [13], [5], [21] that have suggested improved CP-ABE. Later in 2012, Zhang *et al.* suggested novel CP-ABE using lattice [22]. Wang introduced a CP-ABE in the standard model using lattice [20]. Some works use Learning with Error approach [9], [7].

2.2 Blockchain

The concept of blockchain becomes very popular from bitcoin made by Nakamoto [15] in 2008. 3 years after bitcoin was published, th improvement of blockchain is quite explosive. Numerous kinds of cryptocurrencies like Ethereum and many platforms like EOS are designed. In addition, block chain have become pervasive into finance, supply chains and the Internet of things, *etc.* with steep growth.

2.3 Related Work

2.3.1 Distributed ABE

Distributed ABE [14] proposed a distributed version of CP-ABE while multiple attribute authorities carry out operations independently and create secret attribute keys. Furthermore, they give the first construction of a distributed ABE scheme with supporting an access policies written in Disjunctive Normal Form (DNF). The ciphertexts grow linearly as we include conjunctions more and more.

2.3.2 Decentralizing ABE

Decentralizing ABE [13] proposed a novel multi-authority attribute-based encryption scheme. In this scheme, any party can become an authority with no requirement for any global coordination except the creation of an initial common reference parameters. A party can act as an authority by creating a public key and issuing private keys to each user corresponding to their attributes. Authorities need not be aware of each other but just carry out their jobs independently. Lewko and Waters use the multi-authority [5] concept of global identifiers to connect private keys that are issued by different authorities to the same user. A user can encrypt data with any Boolean formula expressed by a Linear Secret Sharing Scheme (LSSS) for attributes. They did not require any central authority.

2.3.3 FairAccess

FairAccess by Ouaddah *et al.* [17] is a fully decentralized pseudonymous and privacy-preserving authorization management framework. They designed and implemented a new distributed access control framework based on blockchain. Unlike transactions in many cryptocurrencies, they use new types of transactions that are grand, get, delegate and revoke access. They follow some principles they defined: (1) User driven and transparency: user has full control on own data and access control, (2) Fairness: nobody can handle loss control on user's own data, and (3) Distributed structure and no central authority: without additional progress, data owner and other nodes can communicate directly. In this paper, they use a blockchain as a distributed ledger for access control management decisions but do not provide confidentiality of the data.

2.3.4 BDABE

Blockchain-based Distributed Attribute based Encryption (BDABE) by Bramm *et al.* [3] improved the idea of FairAccess approach. But BDABE is not for the Internet of Things, they focus on ABE. From [14], they improve ABE scheme by adding some more components to use blockchain. They use blockchain similar to FairAccess but they proposed solution that provides complete confidentiality of the data by utilizing a cryptographic access control approach.

3 Preliminaries

3.1 Bilinear Pairing

Let $G_1,\ G_2,\ G_T$ be cyclic groups of prime order $p > 3$ with generator $g_1,\ g_2,\ g_3$, respectively. Define a map $e : G_1 \times G_2 \longrightarrow G_T$ and call e as a *bilinear map* if e satisfies followings:

(1) Bilinearity: For any $\alpha \in G_1,\ \beta \in G_2$ and $a,\ b \in \mathbb{Z}_p$, we have $e(\alpha^a,\ \beta^b) = e(\alpha,\ \beta)^{ab}$.

(2) Non-degeneracy: $e(g_1,\ g_2) \neq 1$, (1 denotes the identity of G_T).

(3) It is efficient to compute e.

3.2 Access Structure

Let $X = \{P_1,\ P_2, \cdots,\ P_n\}$ be a set of parties. A collection $\mathbb{A} \in \mathcal{P}(X)$ is *monotone* if for any $B,\ C \in \mathcal{P}(X)$, $B \in \mathbb{A}$ and $B \subset C \implies C \in \mathbb{A}$. An access structure (respectively, *monotone access structure*) or policy is a collection (respectively, *monotone collection*) $\mathbb{A}$ of non-empty element of $\mathcal{P}(X)$. The $\mathcal{A} \in \mathbb{A}$ is called an *authorized set*, and $\mathcal{A}' \notin \mathbb{A}$ is called an *unauthorized set.* The role of the parties is taken by the attributes. We write the access policy $\mathbb{A}$ with DNF as following:

$$\mathbb{A} = \bigvee_{i=1}^{n} \left(\bigwedge_{\mathcal{A} \in S_j} \mathcal{A} \right)$$

where S_j is a set of attributes.

3.3 Blockchain

We can consider blockchain into three different kinds of features. First, blockchain is just a chain of blocks like the linked list, one of data structures. The block consists of block number, hash of block and some transactions. The transaction contains information on attributes or users that we discuss in Section 5. Blocks are connected to each other by a hash function and the hash value of the block is included in the next block. Secondly, it looks like a distributed ledger. The first type of blockchain does not spread to any other users. But now many users can share the same blockchain. Because of the distribution feature, we prevent a single point of failure so that we enhance availability. The last is adding a consensus algorithm in the second view. The consensus algorithm is a way of agreement on the latest block. The last is a general definition of blockchain and we will use it after remaining sections.

3.3.1 Permissioned Blockchain

There are two types of blockchain in general, one is public and the other is private. The big difference between them is to allow participants to have authorities of reading, writing and verifying transactions or blocks. In public, any user does not need to get authority in reading, writing and verifying, but in private some organization gives authority to users in order to read, write and verify transactions or blocks. From this point of view, the private chain seems suitable for ABE because some authorities have to authorize other authorities to operate.

3.4 Consensus

In blockchain, consensus is the most important part of blockchain. Consensus is a way of agreement determining whether to share the block with all users. We can easily share old blocks which were generated a long time ago because old blocks are generated by the consensus algorithm previously and nobody can change the transactions in those. But a problem occurs in dealing with the newest block. Users cannot know which block contains the correct block number, valid hash value, and transactions. The consensus algorithm gives users confidence about which block is trustable. After consensus, we assume that users believe the block is reliable and agree on sharing this block. In this paper, we use the Practical Byzantine Fault Tolerance (PBFT) algorithm proposed by Miguel Castro and Barbara Liskov [4].

PBFT is one of the optimizations to solve Byzantine Generals Problem. We can rethink Byzantine Generals Problem as to how dealing with a distributed computer network works correctly as we intend, even though malicious components of the system try to disturb. Fig.(1) shows the workflow of PBFT. It consists of 4 phases.

(1) A client sends a request to the leader D to invoke a service operation.

(2) The leader D broadcasts the request to the backup nodes.

(3) The nodes (R0, R1, R2, R3 in Fig.(1)) carry out the request and send a reply to the client.

(4) The client waits until getting $f + 1$ answers from different nodes (f represents the maximum number of nodes that may be faulty).

4 BDABE Scheme

In order to design a system, we describe the construction: [3]

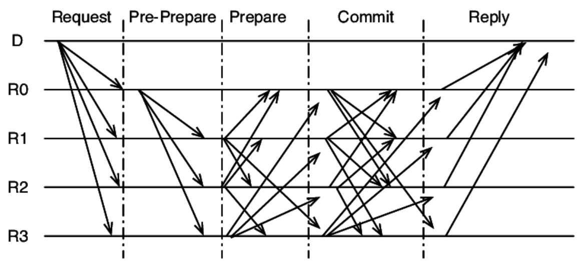

Figure 1: Workflow of PBFT

(1) Root Authority (RA): RA is the first component of our system. It takes charge of creating and revoking attribute authorities by generating private keys of AU and subtracting AUs from consensus. In our system, RA initializes and verifies blockchain.

(2) Attribute Authority (AU): AU is one of the cores in ABE. AU communicates with RA, each other AUs and users. AU handles a domain consisting of a set of users and a set of attributes. Each AU registers or revokes users in its domain and generates a public key and private key of users. But the main job of AU is to create attributes and issue attributes to users.

(3) Data Owner (DO): It could be anyone, even an unauthorized user. DO is a user who uploads or shares own data. DO decides an access policy $\mathbb{A}$ for data. Therefore a user who wants to access the data can read only if he or she satisfies the determined access policy $\mathbb{A}$.

(4) Data Reader (DR): DR is an authorized user who can access to encrypted data. If one of DR's attribute sets satisfies an access policy $\mathbb{A}$ associated with ciphertext, DR will be able to decrypt and acquire the plaintext.

Table 1 indicate notations of various keys that we will use in this paper.

Table 1: Notation of Keys

PK	public key of a root authority RA
MK	master key of a root authority RA
SK_{AU}	secret key of an attribute authority AU
PK_u	public key of a user u
SK_u	secret key of a user u
$PK_{\mathcal{A}}$	public key of an attribute $\mathcal{A}$
$SK_{\mathcal{A},u}$	secret key of an attribute $\mathcal{A}$ for a user u

4.1 Distributed ABE

4.1.1 Setup

The Setup algorithm is run by RA. It chooses two bilinear groups G_1 and G_2 of prime order $p > 3$ and a bilinear pairing $e : G_1 \times G_2 \longrightarrow G_T$. It chooses two generators $g_1 \in G_1$ and $g_2 \in G_2$, two random elements $P_1 \in G_1$ and $P_2 \in G_2$, a random exponent $y \in \mathbb{Z}_p$, and a global hash function $H : \{0,\ 1\}^* \longrightarrow \mathbb{Z}_p$. The public key of the system is given by:

$$PK = \left\{ \begin{array}{c} G_1,\ G_2,\ H,\ e, \\ g_1,\ g_2,\ P_1,\ P_2, \\ e(g_1,\ g_2)^y \end{array} \right\}$$

You can regard PK as a global parameter. The secret master key of the system is given by:

$$MK = \{y\}$$

4.1.2 CreateAuthority(PK, MK, a)

The algorithm CreateAuthority is also run by the RA and takes PK, MK and Authority address a. CreateAuthority chooses a random value $r_{\mathsf{AU}} \in \mathbb{Z}_p$ for randomizing the result of the hash function H. The MK is split into two components by selecting two random exponents $\alpha,\ \beta \in \mathbb{Z}_p$, such that $\alpha + \beta = y \equiv MK \pmod p$. Then a secret key of AU, SK_{AU}, is

$$SK_{\mathsf{AU}} = \left\{ \begin{array}{c} SK_{\mathsf{AU},\mathrm{I}} = g_1^{\alpha},\ SK_{\mathsf{AU},\mathrm{II}} = g_2^{\beta}, \\ SK_{\mathsf{AU},\mathrm{III}} = r_{\mathsf{AU}},\ SK_{\mathsf{AU},\mathrm{IV}} = a \end{array} \right\}$$

4.1.3 CreateUser(PK, SK_{AU}, u)

The algorithm CreateUser, run by AU takes inputs PK, SK_{AU} and an user address u. It chooses a secret value $r_u \in \mathbb{Z}_p$ for each u and returns the public and secret user key of u as follows:

$$PK_u = \{PK_{u,\mathrm{I}} = g_1^{r_u},\ PK_{u,\mathrm{II}} = g_2^{r_u},\ PK_{u,\mathrm{III}} = u\}$$

$$\begin{aligned} SK_u &= \{SK_{u,\mathrm{I}} = SK_{\mathsf{AU},\mathrm{I}} \cdot P_1^{r_u}, SK_{u,\mathrm{II}} = SK_{\mathsf{AU},\mathrm{II}} \cdot P_2^{r_u}\} \\ &= \{g_1^{\alpha} \cdot P_1^{r_u},\ g_2^{\beta} \cdot P_2^{r_u}\} \end{aligned}$$

4.1.4 RequestAttributePK($PK, SK_{\mathsf{AU}}, \mathcal{A}$)

Both users and AUs can request public attribute key $PK_{\mathcal{A}}$ but their requests have different purpose and acceptance authority also different. In the users' view, they need a set of $PK_{\mathcal{A}}$s during encryption and we handle this later. A set of attributes $\{PK_{\mathcal{A}}\}_{\mathcal{A} \in S_j}$ where S_j is a set of attributes is related to an access policy $\mathbb{A}$ so that users request only required $PK_{\mathcal{A}}$ following $\mathbb{A}$. Users send this request to AU where they belong to. In AU's view, AU send a request to other AU which really creates $\mathcal{A}$ so that users who belong to some AU's domain can request attributes in other AUs' domain. RequestAttributePK verifies that AU is a AU which creates $\mathcal{A}$. If not, the algorithm returns $\perp$. Otherwise, using SK_{AU}, calculate an exponent $\epsilon(\mathcal{A}, \mathsf{AU}) = H(\mathcal{A}) \cdot SK_{\mathsf{AU},III} \cdot H(SK_{\mathsf{AU},IV}) = H(\mathcal{A}) \cdot r_{\mathsf{AU}} \cdot H(a)$ and return $PK_{\mathcal{A}}$, which consists of three parts:

$$PK_{\mathcal{A}} = \left\{ \begin{array}{c} PK_{\mathcal{A},\mathrm{I}} = g_1^{\epsilon(\mathcal{A},\mathsf{AU})}) \\ PK_{\mathcal{A},\mathrm{II}} = g_2^{\epsilon(\mathcal{A},\mathsf{AU})}) \\ PK_{\mathcal{A},\mathrm{III}} = e(g_1,\ g_2)^{y \cdot \epsilon(\mathcal{A},\mathsf{AU})}) \end{array} \right\}$$

where $\epsilon(\mathcal{A},\ \mathsf{AU}) = H(\mathcal{A}) \cdot SK_{\mathsf{AU},\mathrm{III}} \cdot H(SK_{\mathsf{AU},\mathrm{IV}}) = H(\mathcal{A}) \cdot r_{\mathsf{AU}} \cdot H(a)$.

The PK can be requested from AU by any users but to get $PK_{\mathcal{A}}$, $\mathsf{AU}_{\mathcal{A}}$ has to be involved because of a random value $r_{\mathcal{A}}$.

4.1.5 RequestAttributeSK($PK, SK_{\mathsf{AU}}, \mathcal{A}, PK_u$)

The AU runs RequestAttributeSK algorithm when a DR requests the secret attribute key $SK_{\mathcal{A}}$. RequestAttributeSK takes $PK, SK_{\mathcal{A}}, \mathcal{A}$, and PK_u as inputs. Firstly, check that AU where a DR belongs to has an authority to handle $\mathcal{A}$ and then the authority AU figures out whether the DR with address u is eligible for the attribute $\mathcal{A}$. If the DR with u is not eligible to hold $\mathcal{A}$, RequestAttribute SK returns $\perp$, otherwise RequestAttributeSK returns the secret key of attribute $\mathcal{A}$ for address u, $SK_{\mathcal{A},u}$ defined as following:

$$SK_{\mathcal{A},u} = \left\{ \begin{array}{c} SK_{\mathcal{A},u,I} = (PK_{u,I})^{\epsilon(\mathcal{A},\mathsf{AU})}, \\ SK_{\mathcal{A},u,II} = (PK_{u,II})^{\epsilon(\mathcal{A},\mathsf{AU})} \end{array} \right\}$$

$$= \left\{ \begin{array}{c} g_1^{r_u \cdot \epsilon(\mathcal{A},\mathsf{AU})}, g_2^{r_u \cdot \epsilon(\mathcal{A},\mathsf{AU})} \end{array} \right\}$$

where $\epsilon(\mathcal{A}, \mathsf{AU}) = H(\mathcal{A}) \cdot SK_{\mathsf{AU},III} \cdot H(\mathsf{AU}) = H(\mathcal{A}) \cdot r_{\mathsf{AU}} \cdot H(\mathsf{AU})$

4.1.6 Encrypt($M, PK, \mathbb{A}, PK_{\mathcal{A}_1}, \cdots, PK_{\mathcal{A}_l}$)

In order to encrypt a plaintext, an Encrypt needs a message M, PK, a set of $PK_{\mathcal{A}}$s in the access policy $\mathbb{A}$. The resulting ciphertext may be decrypted by DR with a sufficient authorized set of attributes, bound to his address u. The access policy $\mathbb{A}$ must be written in DNF. A policy in DNF consists of n different sets of conjunctions $\{S_1,\ S_2,\ \cdots,\ S_n\}$. The encryption algorithm chooses a random value $r_j \in \mathbb{Z}_p$ for each S_j. Afterwards, the ciphertext CT_j is calculated as:

$$CT_j = \left\{ \begin{array}{c} E_{j,\mathrm{I}} = M \cdot \left(\prod_{\mathcal{A} \in S_j} PK_{\mathcal{A},\mathrm{III}} \right)^{r_j}, \\ E_{j,\mathrm{II}} = P_1^{r_j}, \\ E_{j,\mathrm{III}} = P_2^{r_j}, \\ E_{j,\mathrm{IV}} = \left(\prod_{\mathcal{A} \in S_j} PK_{\mathcal{A},\mathrm{I}} \right)^{r_j}, \\ E_{j,\mathrm{V}} = \left(\prod_{\mathcal{A} \in S_j} PK_{\mathcal{A},\mathrm{II}} \right)^{r_j}, \end{array} \right\}$$

The resulting ciphertext CT is finally obtained as:

$$CT = \{CT_1,\ CT_2,\ \cdots,\ CT_n\}$$

4.1.7 Decrypt($CT, PK, \mathbb{A}, SK_u, \{SK_{\mathcal{A}_i,u}\}_{i \in [n]}$)

Whenever DR wants to read a data from encrypted data, the DR runs Decrypt. To decrypt a CT, Decrypt first verifies that for a given policy $\mathbb{A}$, a DR with $\{SK_{\mathcal{A}_i,u}\}$ for $1 \leq i \leq n$ satisfies the $\mathbb{A}$. If this is not the case, the algorithm outputs $\perp$. Otherwise, it returns the plaintext message M by calculating as follow:

$$M = E_{j,\mathrm{I}} \cdot \frac{e(E_{j,\mathrm{II}}, \prod_{i \in S_j} SK_{\mathcal{A},u,\mathrm{II}}) \cdot e(E_{j,\mathrm{III}}, \prod_{i \in S_j} SK_{\mathcal{A},u,\mathrm{I}})}{e(E_{j,\mathrm{IV}}, SK_{u,\mathrm{II}}) \cdot e(E_{j,\mathrm{IV}}, SK_{u,\mathrm{I}})}$$

5 Our Approach

Now, we will show how we apply the blockchain to Distributed ABE. First, indicate what data is stored in blockchain and secondly, indicate how the algorithms in the Distributed ABE use block chain. Lastly, we will show our blockchain model.

(1) Store attributes

We use $\mathcal{B}$ as a database (DB) stores attributes and what AU generates attributes. This is for getting grant when user request attribute to some AU which does not generate the attribute. If store the secret key of attribute, one of malicious AU will bring the collapsing our model. We write $\mathcal{B}$ to denote blockchain.

(2) Store user's information

We will store users' information, for example what attributes belong to a user and a revocation bit. This is for verifying that AU works correctly and issued attribute is valid. But we have to use hash function when store the user's ID.

Now, we see how algorithms in Distributed ABE are connected with $\mathcal{B}$.

5.1 CreateAuthority

As mentioned in Section 4.1.2, RA runs CreateAuthority right after CreateBlockchain algorithm we see later. At this moment, RA takes a value f which is the maximum number of malicious AU in $\mathcal{B}$ and choose a number k larger than $3f+1$ which denotes how many AUs make. After creation, AUs participate in $\mathcal{B}$ as nodes.

5.2 CreateUser

When we operate CreateUser in Distributed ABE, AU stores a user's information on $\mathcal{B}$. User's information is used when revoking attribute key and verifying $\mathcal{B}$. The precaution when store user's information, we must hash the user's ID to prevent tracking.

5.3 RequestAttributePK

When AU request a $PK_{\mathcal{A}}$, AU has to contact with $\mathsf{AU}_{\mathcal{A}}$ to get permission for $PK_{\mathcal{A}}$. Now, AU can skip this process and takes $PK_{\mathcal{A}}$ on $\mathcal{B}$.

From now, we consider algorithms that does not belong to Distributed ABE.

5.4 CreateBlockchain(f)

CreateBlockchain algorithm operates while Setup algorithm is running and initializes a blockchain. Specifically we use a permissioned blockchain $\mathcal{B}$ as Section 5.1 because we have to restrict access to credential data. CreateBlockchain generates a genesis block storing PK, *etc.* and takes a value f. CreateBlockchain determines a number $k(k \geq 3f+1)$. k is related to a PBFT protocol. Security of our model is depending on the number of AU. After CreateBlockchain, it operates CreateAuthority k times.

5.5 CreateAttribute(AU, $\mathcal{A}$)

When AU decide to create a attribute $\mathcal{A}$, run CreatAttribute. There are two case to function CreateAttribute one is AU wants to make $\mathcal{A}$ and the other is a users request public attribute key $PK_{\mathcal{A}}$ for $\mathcal{A}$. Anyone who wants to encrypt data can access to this $PK_{\mathcal{A}}$. After creating $PK_{\mathcal{A}}$, AU uploads the $PK_{\mathcal{A}}$ on $\mathcal{B}$ so that any AUs can access to this. AU publishes an transaction including $\mathcal{A}$, $PK_{\mathcal{A}}$, and AU's address and broadcasts to near AUs to do consensus. Depending on a consensus, $\mathcal{A}$ is registered or not. When the other AU′ gets a RequestAttributePK from a user u, AU′ does not need to communicate with $\mathsf{AU}_{\mathcal{A}}$ just reads $PK_{\mathcal{A}}$ from $\mathcal{B}$ and sends it to u.

5.6 IssueAttribute(AU, $\mathcal{A}, u$)

IssueAttribute issues an attribute $\mathcal{A}$ to a user u if u is eligible for $\mathcal{A}$. While RequsetAttributeSK, AU determine whether a user u really has an attribute $\mathcal{A}$ and if in that case, AU updates a user's information on the $\mathcal{B}$. AU makes a transaction and broadcasts to other $\mathcal{A}$ for consensus. If the transaction is valid after consensus, IssueAttribute send the secret key of $\mathcal{A}$ related to a user u.

5.7 RevokeAttribute($\mathcal{A}, u$)

We can classify RevokeAttribute into two types. One is revocation just a user's attribute and other is revocation attribute itself, in other words, revoke attribute from all user have $\mathcal{A}$. When AU finds that a user u does malicious behaviour or get a revocation request from a user u, AU revokes the public and secret attribute key $PK_{\mathcal{A}}$, $SK_{\mathcal{A},u}$ from u. It is possible because AU know the secret key of a user. If AU wants to revoke an attribute $\mathcal{A}$, AU requests that revoke public and secret attribute key $PK_{\mathcal{A}}$, $SK_{\mathcal{A},u}$ in other AUs' domain. After requesting and revocation, AU or all AUs have to update whether to revoke or not.

5.8 VerifyChain($\mathcal{B}$)

If a value f denoting the maximum number of malicious AU in $\mathcal{B}$ is large enough, $\mathcal{B}$ is concrete against some deliberate failures. But as f increases, consensus process takes a long time so that decreases the performance of the whole block chain system. It is necessary to determine a proper f and keep in check AU's behaviour and we revoke a malicious AU to maintain $\mathcal{B}$. RA runs VerifyChain to find out a malicious AU if RA detects some strange transaction or update for example, a user's attribute public key has been revoked abruptly.

5.9 RevokeAuthority(AU)

After running VerifyChain algorithm, RA has to revoke a malicious AU if RA finds some malicious AU. RA broadcasts a transaction for revocation of a malicious AU near AUs and carries out a consensus algorithm. The reason that it is not revoked immediately is that RA can be corrupted. After consensus, AU would be revoked or not.

5.10 RevokeRoot(RA)

Unlike RevokeAuthority, AUs can revoke RA if they detects that RA is corrupted. So AU executes consensus and decides whether revoke RA or not. If RA is revoked, some authority takes the RA's role. Some AU can take charge of RA's role or some other RA can take on.

To make trustable blockchian and prevent against some issues, RA and AUs have to keep in check each other. The algorithms RevokeAuthority and RevokeRoot allow us to do.

6 Discussion

6.1 Comparison

We compare our model with Decetralizing ABE [13] and BDABE [3]. Firstly, Decentralizing ABE does not apply blockchain but uses Global Parameter (GP). BDABE and our model apply permissioned blockchain. BDABE is implemented by Multichain [12] which is private blockchain platform using modified Proof-of-Work (PoW) for consensus algorithm. In our model, we use the PBFT protocol for a consensus algorithm to increase performance. In decentralizing ABE, a single AU can verify a user's validity, but in BDABE and our model, several AUs have to participate in verification. However, in BDABE, all AUs do not need to participate in consensus just 75% of AUs take part in. In our model, we need all of AUs for operating consensus. We summarize comparisons of other approaches in Table 2.

6.2 Delegation of Attributes

Many CP-ABE schemes like [1] and [10] support the delegation algorithms that allow a user to make up a subset of own attributes. But in our scheme, we do not support delegation algorithm because it would occur the collusion attack, which means that a party of users pretend to be a new user not belonging to a party. In other words, let S_{u_1} and S_{u_2} be two sets of attributes of users u_1 and u_2, respectively. If we allow delegation they can make a new set of S which $S \nsubseteq S_{u_1}$ and $S \nsubseteq S_{u_2}$ but $S \subseteq (S_{u_1} \cup S_{u_2})$ so that u_1 and u_2 can pretend to a new user. To prevent collusion attack, delegation of attributes is not allowed.

6.3 Transmission of a Secret Attribute Key of User

When a user u executes RequestAttributeSK for an attribute $\mathcal{A}$, u sends a request a secret attribute key $SK_{\mathcal{A},u}$ to AU which contains u in its domain and AU determines whether a user is eligible for $\mathcal{A}$. If u is enough to get $\mathcal{A}$, AU requests a $SK_{\mathcal{A},u}$ with PK_u to $\mathsf{AU}_{\mathcal{A}}$. Then $\mathsf{AU}_{\mathcal{A}}$ creates $SK_{\mathcal{A},u}$ and sends it to u. Here, we have a problem how we send privately. If follow the reveres way of request, $SK_{\mathcal{A},u}$ has to be passed by AU and this gives an opportunity to store $SK_{\mathcal{A},u}$ for AU. So we consider another way to send $SK_{\mathcal{A},u}$ directly to u. But it leaks some information to track u.

Table 2: Comparions with other approaches

	Decentralizing ABE [13]	BDABE [3]	Our Model
Blockchain	X	O	O
Style^{*_1}	GP^{*_2}	Permissioned	Permissioned
Consensus	X	modified PoW	PBFT
Verifier	Single AU	not all AU	all AU

$*_1$ indicates how to achieve decentralization.
$*_2$ works like common reference string (CRS)

7 Conclusion and Future work

In this paper, we improve the concept of applying blockchain to Distributed ABE from BDABE protocol [3]. In our scheme, we can reduce the absolute trust on central server RA because RA and AU hold each other in check frequently and even there are some malicious components, by PBFT, valid transactions are stored in $\mathcal{B}$. By creating multiple AUs, we prevent a single point of failure threat. AUs can share valid data rapidly through $\mathcal{B}$ which is verified by RA often so that they always store reliable data. It increases trust of our scheme. We expect to apply our model to another policy beyond CP, for example, KP, threshold policy.

Later, we will implement our scheme and compare performance with other CP-ABE schemes using LSSS and lattice, *etc.* Since we have not prove our security formally, we will show detailed security proof.

Acknowledgement

This work was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00555, Towards Provable-secure Multi-party Authenticated Key Exchange Protocol based on Lattices in a Quantum World).

References

[1] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 321–334. IEEE, 2007.

[2] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.

[3] Georg Bramm, Mark Gall, and Julian Schütte. BDABE - blockchain-based distributed attribute based encryption. In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications, ICETE 2018 - Volume 2: SECRYPT, Porto, Portugal, July 26-28, 2018.*, pages 265–276, 2018.

[4] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[5] Melissa Chase. Multi-authority attribute based encryption. In *Theory of Cryptography Conference*, pages 515–534. Springer, 2007.

[6] Melissa Chase and Sherman SM Chow. Improving privacy and security in multi-authority attribute-based encryption. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 121–130. ACM, 2009.

[7] Zehong Chen, Peng Zhang, Fangguo Zhang, and Jiwu Huang. Ciphertext policy attribute-based encryption supporting unbounded attribute space from r-lwe. *KSII Transactions on Internet & Information Systems*, 11(4), 2017.

[8] Ling Cheung and Calvin Newport. Provably secure ciphertext policy abe. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 456–465. ACM, 2007.

[9] Tan Soo Fun and Azman Samsudin. Lattice ciphertext-policy attribute-based encryption from ring-lwe. In *Technology Management and Emerging Technologies (ISTMET), 2015 International Symposium on*, pages 258–262. IEEE, 2015.

[10] Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute based encryption. In *International Colloquium on Automata, Languages, and Programming*, pages 579–591. Springer, 2008.

[11] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. Acm, 2006.

[12] Gideon Greenspan. Multichain private blockchainwhite paper. *URl: http://www. multichain. com/download/MultiChain-White-Paper. pdf*, 2015.

[13] Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 568–588. Springer, 2011.

[14] Sascha Müller, Stefan Katzenbeisser, and Claudia Eckert. Distributed attribute-based encryption. In *International Conference on Information Security and Cryptology*, pages 20–36. Springer, 2008.

[15] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[16] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 195–203. ACM, 2007.

[17] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Fairaccess: a new blockchain-based access control framework for the internet of things. *Security and Communication Networks*, 9(18):5943–5964, 2016.

[18] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473. Springer, 2005.

[19] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the theory and application of cryptographic techniques*, pages 47–53. Springer, 1984.

[20] Yongtao Wang. Lattice ciphertext policy attribute-based encryption in the standard model. *IJ Network Security*, 16(6):444–451, 2014.

[21] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *International Workshop on Public Key Cryptography*, pages 53–70. Springer, 2011.

[22] Jiang Zhang, Zhenfeng Zhang, and Aijun Ge. Ciphertext policy attribute-based encryption from lattices. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 16–17. ACM, 2012.