Symmetry **2015**, *7*, 269-283; doi:10.3390/sym7010269

OPEN ACCESS

symmetry

ISSN 2073-8994

www.mdpi.com/journal/symmetry

Technical Note

Study on User Authority Management for Safe Data Protection in Cloud Computing Environments

Su-Hyun Kim and Im-Yeong Lee *

Department of Computer Software Engineering, Soonchunhyang University, 646, Eupnae-ri, Sinchang-myeon, Asan-si, Chungcheongnam-do 336-745, Korea; E-Mail: kimsh@sch.ac.kr

* Author to whom correspondence should be addressed; E-Mail: imylee@sch.ac.kr; Tel.: +82-41-530-1323.

Academic Editors: Young-Sik Jeong, Laurence T. Yang and Stefanos Gritzalis

Received: 15 January 2015 / Accepted: 10 March 2015 / Published: 19 March 2015

Abstract: In cloud computing environments, user data are encrypted using numerous distributed servers before storing such data. Global Internet service companies, such as Google and Yahoo, recognized the importance of Internet service platforms and conducted self-research and development to create and utilize large cluster-based cloud computing platform technology based on low-priced commercial nodes. As diverse data services become possible in distributed computing environments, high-capacity distributed management is emerging as a major issue. Meanwhile, because of the diverse forms of using high-capacity data, security vulnerability and privacy invasion by malicious attackers or internal users can occur. As such, when various sensitive data are stored in cloud servers and used from there, the problem of data spill might occur because of external attackers or the poor management of internal users. Data can be managed through encryption to prevent such problems. However, existing simple encryption methods involve problems associated with the management of access to data stored in cloud environments. Therefore, in the present paper, a technique for data access management by user authority, based on Attribute-Based Encryption (ABE) and secret distribution techniques, is proposed.

Keywords: cloud computing; attribute-based encryption; user authority

1. Introduction

Recently, as interest in data has increased at home and abroad, many related studies have been conducted. Based on the growth of IT technologies, many firms are interested in data that can expand to diverse areas and allow efficient use of computing power. Recently, numerous Internet service companies have recognized the importance of Internet service platforms, and have conducted in-house research and development to create and utilize large cluster-based cloud computing technologies based on low-priced commercial nodes [1]. In such cloud computing environments, user data are stored and maintained using numerous distributed servers. In such distributed computing environments, distributed management that can manage user high-capacity data is emerging as a major issue. However, because of the diverse forms of using high-capacity data, such as storage and saving, there are no appropriate countermeasures against security invasion and data loss by malicious attackers or internal users. Among users or firms, systems to maintain important data in external places that cannot be controlled by such users or firms are spreading because of issues, such as management costs. In this case, the diverse insecurity factors of cloud services, such as information exposure, information manipulation, and information loss, occur because of cloud service providers.

As the most basic method of solving such problems, stored data can be managed through encryption. However, existing simple encryption methods involve problems associated with the management of access to data stored in cloud environments. That is, users want multiple user access to data stored in cloud servers, or require diverse functions such as access control by user class. However, existing public-key cryptography or symmetric-key algorithm techniques cannot solve problems in key management or satisfy requirements such as access control.

To solve problems in existing encryption techniques, encryption methods suitable for distributed storage servers have recently been studied actively. Characteristically, Sahai *et al.* proposed the concept of Attribute-Based Encryption (ABE) as an expanded form of the concept of ID-based encryption (IBE) [2]. The IBE method that has become the basis of ABE is a method for solving certificate problems in public-key cryptography, which was first proposed by Shamir in 1984 [3]. ABE makes public-keys that use user attributes instead of their IDs. The user attributes can be composed of multiple attributes. For instance, data for a particular user can be decrypted only when the attributes of such user are the department of computers and professor as the user's department and title.

In this paper, the existing ABE technique is used together with the secret distribution technique to propose a new data access management technique. Although the existing ABE technique provides a method for multiple users to access data using diverse attributes, the ABE technique alone cannot control data access in diverse cloud environments. The property-based technique alone cannot control the detailed data access management in the diverse cloud computing environments that require the classification of the users who have the same properties. Therefore, in this paper, a technique is proposed to allow only those users with authorization not lower than a given threshold among all users that satisfy attributes, to finally access data through the secret distribution technique.

This paper is composed as follows. In Section 2, existing studies are introduced to help understand the technique proposed in this paper; in Section 3, basic security requirements with which cloud computing environments should be equipped are examined; and in Section 4, the proposed method is explained. In Section 5, the safety of the proposed method is analyzed, and finally, in Section 6, this paper is finished with conclusions and future study directions.

2. Related Studies

In this chapter, related technologies and previously proposed methods are introduced to help understand the technique proposed in this paper.

2.1. Google File System (GFS)

Google File System (GFS) used by Google was developed to be suitable for high-capacity data, and has been optimized for core data storage and search engines [4]. GFS uses numerous low cost storage servers to distribute high capacity to servers for storage. GFS is composed of a master server and chunk servers, and data are distributed to multiple chunk servers in units of 64 MB. The master server assigns unique 64-bit labels when individual chunk servers are created, and maintains connections with the chunk servers using logical mapping. However, GFS has a limitation in that it requires measures to respond to disorders that could occur frequently because multiple chunk servers are used.

2.2. Apache Hadoop Distributed File System (HDFS)

Apache Hadoop Distributed File System (HDFS) is a file system made to be executable in existing hardware, and it is quite similar to existing distributed file systems. However, HDFS also shows many differences from existing distributed file systems, such as its good fault recovery functions, and it is designed to be applicable to low-priced hardware. HDFS is most commonly used with the cloud computing platforms of diverse IT companies, such as Amazon's Elastic Compute Cloud (EC2) and Yahoo [5].

The design and structure for implementation of HDFS are almost the same as those for GFS, although the fact that it has excellent portability to diverse platforms because it uses Java is a significant difference from GFS. The use of Java language with high portability provides HDFS with the advantage that it can be driven in diverse servers that support Java [6].

An HDFS client divides user data into 128 MB blocks, and requests NameNode for storage. NameNode finds the addresses of DataNodes where the data blocks will be stored, and sends the addresses to the client. On receipt of the three DataNode addresses, the client directly transmits data to the first DataNode, and the first DataNode transmits a copy of the data to the next DataNode on receipt of the data. As such, each block is backed up and stored in a total of three DataNodes to be prepared for DataNode failure or errors.

Existing Hadoop systems do not apply access control to data blocks stored in data nodes. In this case, unauthenticated clients can access data blocks to read or write. To prevent this, Hadoop version 1.0.0 was added with a concept-termed block access token to confirm the authority of users (authorities) that access data blocks.

The name node and data nodes share one secret key. The secret key is renewed periodically by the name node, and it is shared through the heartbeats frequently exchanged with data nodes. This secret key is used to encrypt block access tokens requested by clients. If the secret key transmitted between the name node and data node is exposed to an attacker, data blocks stored in the data nodes could be exposed to the attacker as well.

The structure of these block access tokens is shown in Figure 1. Each block access token is composed of a tokenID and a TokenAuthenticator, and the detailed structure is as follows:

- expirationDate: expiration date
- keyID: identification of the secret key used to make the token authenticator
- ownerID: identification of the token owner
- blockID: identification of blocks permitted by the token
- accessModes: consists of a combination of block authorities; authorities to read, write, and copy

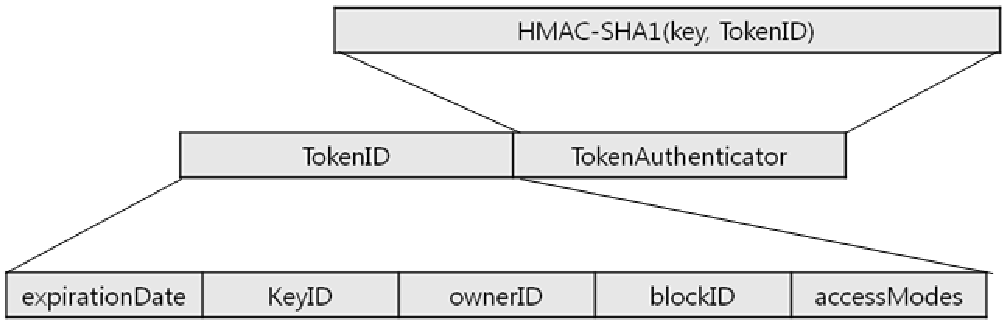

Figure 1. Structure of a block access token.

2.3. Attribute-Based Encryption (ABE)

Attribute-Based Encryption (ABE) is a technology to provide control mechanisms for access to encrypted data based on decrypting keys and attributes assigned to encrypted data or access policies. ABE types include Ciphertext-Policy ABE (CP-ABE) that determines the access structures when documents are encrypted, and Key-Police ABE (KP-ABE) that determines access structures when user keys are created.

2.3.1. CP-ABE

CP-ABE determines the access structures when documents are encrypted. Each ciphertext is associated with an access structure, and each decryption key is associated with attributes. Each user has attributes, and receives the key that corresponds to the attributes.

2.3.2. KP-ABE

KP-ABE determines the access structures when user keys are created. Each ciphertext is associated with attributes, and each decryption key is associated with an access structure. Encrypting parties cannot determine those that can decrypt data.

2.4. Bilinear Mapping

When Cyclic Groups G1 and G2 have the same fraction q, computable bilinear mapping e: G1 × G2 = GT satisfies the following conditions [7]:

(1) Computability: efficient algorithms that calculate $e(P,Q)$ for all $P,Q \in G_1$ exist.

(2) Bilinearity: $e(aP,bQ) = e(P,Q)^{ab}$ is valid for all $P,Q \in G_1$ and $a,b \in Z_p^*$.

(3) Non-degeneracy: $e(P,Q) \neq 1$ is valid for all pairs P and Q of G.

The following arithmetic operation is valid because of the above bilinearity.

$$\begin{aligned} e(aP,bQ) &= e(P,bQ)^a = e(aP,Q)^b = e(P,Q)^{ab} \\ &= e(abP,Q) = e(P,abQ) \\ e(P,Q+R) &= e(P,Q)e(P,R) \\ e(P+Q,R) &= e(P,R)e(Q,R) \end{aligned}$$

Many encryption protocols that use bilinear mapping as a cryptological tool are based on the difficulties of the following problem.

Definition 1. Bilinear Diffie-Hellman Problem (BDHP): refers to the problem of calculating $e(P,P)^{abc}$ when G1 elements P, *aP*, *bP*, and *cP* have been given.

2.5. Analysis of Existing Scheme

2.5.1. Yu *et al.* [8]

Yu *et al.* forecasted that existing business environments would shift to cloud computing environments, and proposed a system model that can guarantee data confidentiality in those environments. They utilized the concept of KP-ABE to encrypt data as sets of attributes, and to allow those with the secret key related to the access structure that satisfies the set of encryption attributes to access the data. Data owners define system attribute sets and create and set keys related to the system. Data owners encrypt secret keys made by encrypting data using ABE, and transmit the results to the cloud service provider. Encrypted keys and ciphertexts are managed solely by the cloud service provider, and data confidentiality can be invaded through collusion between withdrawn users and the cloud service provider.

2.5.2. Zhu *et al.* [9]

Zhu *et al.* proposed a new attribute-based file sharing technique. They provided a systematic definition of attribute computing in cloud computing environments, and proposed a safe and practical ABE technique that does not require pairing.

2.5.3. Jin *et al.* [10]

In 2013, Jin *et al.* analyzed data security issues on the Hadoop platform and proposed a design of reliable file system for Hadoop. This method was designed by using fully homomorphic and authentication agent technology. Homorphic cryptography uses encrypted data to protect data security and efficiency of application programs. Authentication agent technology provides various access control rules based on combination of access control mechanism. These two technologies are used for providing privilege separation and security inspection mechanism to ensure safe storage of data in the Hadoop file system.

2.5.4. Zhang *et al.* [11]

In 2011, Zhang *et al.* proposed flexible extension technique for data storage by using Linux clustering, distributed file system, and cloud computing framework based on Hadoop. While the conventional HDFS file system structure has high throughput to build a large-capacity file storage system, it is not suitable for processing and storing small size files. To improve this problem, the study proposed an extension technique that is flexible for data storage, by managing the files based on HBase on the entity metadata storage level and using entity metadata generation query.

3. Security Requirements

One of core mechanisms of data is the efficient management of high-capacity data. Given that security vulnerability and privacy invasion by malicious attackers or internal users can occur because of the diverse forms of using high-capacity data, the following security items become necessary [12].

- *Confidentiality*: data communicated between data storage servers and client terminals should be readable only by legitimate entities. That is, unauthorized users should not be capable of obtaining such data.
- *Authentication*: data storage servers should be capable of verifying whether users are legitimate entities, and allow only legitimate users to access data.
- *Availability*: to guarantee that, when high-capacity data are transmitted, authentication and confidentiality occur.
- *Calculation efficiency*: in order to reduce the overhead of client terminals and cloud servers during frequent data transmissions, only minimum calculations should be ensured.
- *Collusion resistance*: even if many users collude, they should be able to obtain only the data permitted to each individual user.
- *Forward secrecy*: users for whom certain attributes have been withdrawn should not be capable of obtaining the data required by the relevant attributes.
- *Backward secrecy*: users that have obtained new attributes should not be capable of obtaining, with the current authority, data that could be obtained previously.

4. Proposed Method

4.1. System Model and Assumption

In general, cloud-computing environments were designed based on the HDFS of Apache. The basic concept of our proposed method is CP-ABE. A user that satisfies certain attributes in a group obtains the authority to obtain the decryption key. In this case, the user obtains such authority to decrypt data based on his/her authority class under the agreement of the manager group. Therefore, the proposed method can perfectly guarantee resistance to collusion, which is a problem of the existing ABE, in addition to data confidentiality through class management by subdivided authority (Figure 2).

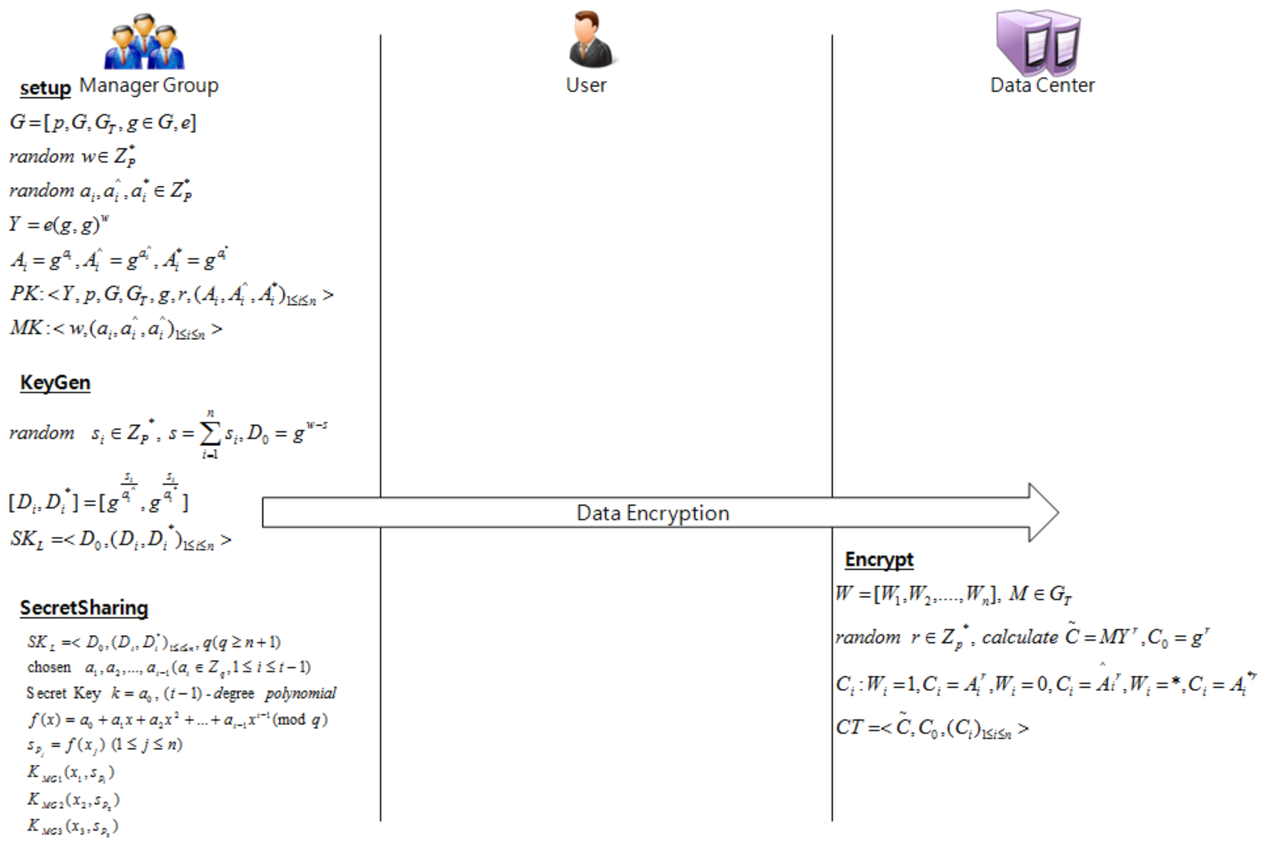
setup Manager Group
User
Data Center
$G=[p,G,G_T,g\in G,e]$
random $w\in Z_P^*$
random $a_i,a_i^{\wedge},a_i^*\in Z_P^*$
$Y=e(g,g)^w$
$A_i=g^{a_i},A_i^{\wedge}=g^{a_i^{\wedge}},A_i^*=g^{a_i^*}$
$PK:<Y,p,G,G_T,g,r,(A_i,A_i^{\wedge},A_i^*)_{1\le i\le n}>$
$MK:<w,(a_i,a_i^{\wedge},a_i^{\wedge})_{1\le i\le n}>$
KeyGen
random $s_i\in Z_P^*$, $s=\sum_{i=1}^{n}s_i$, $D_0=g^{w-s}$
$[D_i,D_i^*]=[g^{\frac{s_i}{a_i^{\wedge}}},g^{\frac{s_i}{a_i^*}}]$
$SK_L=<D_0,(D_i,D_i^*)_{1\le i\le n}>$
SecretSharing
$SK_L=<D_0,(D_i,D_i^*)_{1\le i\le n},q(q\ge n+1)$
chosen $a_1,a_2,\ldots,a_{t-1}(a_i\in Z_q,1\le i\le t-1)$
Secret Key $k=a_0$, $(t-1)$-degree polynomial
$f(x)=a_0+a_1x+a_2x^2+\ldots+a_{t-1}x^{t-1}(\bmod q)$
$s_{P_j}=f(x_j)\ (1\le j\le n)$
$K_{MG1}(x_1,s_{P_1})$
$K_{MG2}(x_2,s_{P_2})$
$K_{MG3}(x_3,s_{P_3})$
Data Encryption
Encrypt
$W=[W_1,W_2,\ldots,W_n]$, $M\in G_T$
random $r\in Z_p^*$, calculate $\tilde{C}=MY^r,C_0=g^r$
$C_i:W_i=1,C_i=A_i^r,W_i=0,C_i=\hat{A_i}^r,W_i=*,C_i=A_i^{*r}$
$CT=<\tilde{C},C_0,(C_i)_{1\le i\le n}>$

Figure 2. Attribute-Based Encryption (ABE) and secret sharing.

n	number of participants
P	set of participants $P_i(1\le i\le n)$ in a secret distribution
q	fraction
k	secret information $\in Z_q$
K	set of secret information k
s_{P_i}	pieces of secret $\in Z_q$
S_{P_i}	set of secret pieces s_{P_i} possessed by individual participants P_i
i	attribute value
L	attribute set
$a_i,a_i^{\wedge},a_i^*\in Z_P^*$	correspond to attribute i
$W=[W_1,W_2,\ldots,W_n]$	access structure
M	plain text
$r\in Z_p^*$	random value

4.2. User Authority Management Scheme

4.2.1. Setup

Enter security parameter k to output the public key PK and master key MK that correspond to the value of the parameter.

(1) $G=[p,G,G_T,g\in G,e]$.

Create random values $w\in Z_P^*$.

(2) Select random values $a_i,\hat{a_i},a_i^*\in Z_P^*$ that correspond to attribute $i,(1\le i\le n)$.

(3) Calculate $Y=e(g,g)^w$ and $A_i=g^{a_i},\hat{A_i}=g^{\hat{a_i}},A_i^*=g^{a_i^*}$.

(4) PK is $<Y,p,G,G_T,g,r,(A_i,\hat{A_i},A_i^*)_{1\le i\le n}>$ and MK is $<w,(a_i,\hat{a_i},\hat{a_i})_{1\le i\le n}>$.

4.2.2. KeyGen

This is an algorithm to enter master key MK and attribute set L in order to output secret key SK_L that corresponds to the access structure.

(1) Enter attribute set $L=\lfloor L_1,L_2,...,L_n\rfloor$ to create a secret key.

(2) Select $s_i\in Z_P{}^*$ randomly and calculate $s=\sum_{i=1}^{n}s_i,D_0=g^{w-s}$.

(3) If $L_i=1$, calculate $[D_i,D_i^*]=[g^{\frac{s_i}{\hat{a_i}}},g^{\frac{s_i}{a_i^*}}]$.

(4) The secret key is $SK_L=<D_0,(D_i,D_i^*)_{1\le i\le n}>$.

4.2.3. Secret Sharing

The created secret key SK_L is distributed to individual members of the manager group using the secret distribution technique. Shamir *et al.*'s secret distribution method was applied to the proposed method [13]. In this scenario, the number of members of the manager group is assumed to be three.

$$q(q\ge n+1)$$
$$\text{chosen } a_1,a_2,...,a_{t-1}(a_i\in Z_q,1\le i\le t-1)$$
$$\text{Secret Key } k=a_0,(t-1)\text{-degree } polynomial$$
$$f(x)=a_0+a_1x+a_2x^2+...+a_{t-1}x^{t-1}(\text{mod}q)$$
$$s_{P_j}=f(x_j)(1\le j\le n)$$
$$K_{MG1}(x_1,s_{P_1})$$
$$K_{MG2}(x_2,s_{P_2})$$
$$K_{MG3}(x_3,s_{P_3})$$

4.2.4. Encrypt

This is an algorithm to enter public key PK, access structure W, and plain text M in order to output ciphertext CT that corresponds to the plain text.

(1) Encrypt access structure $W=[W_1,W_2,....,W_n]$ and plain text M.

(2) Calculate random values $r\in Z_p{}^*$ and $\tilde{C}=MY^r,C_0=g^r$.

(3) Calculate $C_i:W_i=1,C_i=A_i^r,W_i=0,C_i=\hat{A_i}^r,W_i=*,C_i=A_i^{*r}$ that satisfies the following.

(4) The ciphertext is $CT=<\tilde{C},C_0,(C_i)_{1\le i\le n}>$.

4.2.5. (*k*,*n*) Threshold

This is the stage in which a user with access authority requests the manager group for the secret key. If the requester has a legitimate access class to the data, he/she can receive n secret pieces from the manager group to restore the secret key.

$$f'(0) = \sum_{i=1}^{t} s_{P_{Ai}} \prod_{j=1, j\neq i}^{t} \frac{x_{Ai}}{(x_{Aj} - x_{Ai})} (\bmod q)$$
$$SK_L =< D_0, (D_i, D_i^*)_{1\le i\le n} > \quad (1)$$

4.2.6. Decrypt

This is an algorithm to enter the restored secret key SK_L and ciphertext CT in order to output the plain text that corresponds to the ciphertext (Figure 3).

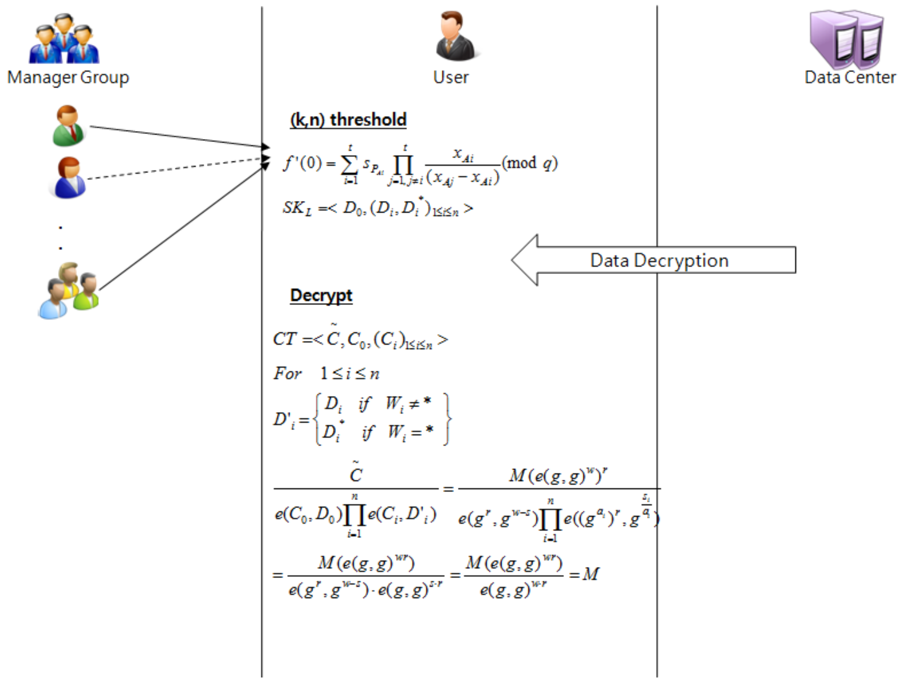
Manager Group
User
Data Center
(k,n) threshold
$f'(0) = \sum_{i=1}^{t} s_{P_{Ai}} \prod_{j=1, j\neq i}^{t} \frac{x_{Ai}}{(x_{Aj} - x_{Ai})} (\bmod q)$
$SK_L =< D_0, (D_i, D_i^*)_{1\le i\le n} >$
Data Decryption
Decrypt
$CT =< \tilde{C}, C_0, (C_i)_{1\le i\le n} >$
$For \quad 1 \le i \le n$
$D'_i = \left\{ \begin{matrix} D_i & if & W_i \neq * \\ D_i^* & if & W_i = * \end{matrix} \right\}$
$\frac{\tilde{C}}{e(C_0, D_0)\prod_{i=1}^{n} e(C_i, D'_i)} = \frac{M(e(g,g)^w)^r}{e(g^r, g^{w-s})\prod_{i=1}^{n} e((g^{a_i})^r, g^{\frac{s_i}{a_i}})}$
$= \frac{M(e(g,g)^{wr})}{e(g^r, g^{w-s}) \cdot e(g,g)^{s \cdot r}} = \frac{M(e(g,g)^{wr})}{e(g,g)^{w \cdot r}} = M$

Figure 3. Data decryption.

Decrypt ciphertext $CT =< \tilde{C}, C_0, (C_i)_{1\le i\le n} >$ using secret key $SK_L =< D_0, (D_i, D_i^*)_{1\le i\le n} >$.

$$For \quad 1 \le i \le n$$

$$D'_i = \left\{ \begin{matrix} D_i & if & W_i \neq * \\ D_i^* & if & W_i = * \end{matrix} \right\}$$

$$\frac{\tilde{C}}{e(C_0, D_0)\prod_{i=1}^{n} e(C_i, D'_i)} = \frac{M(e(g,g)^w)^r}{e(g^r, g^{w-s})\prod_{i=1}^{n} e((g^{a_i})^r, g^{\frac{s_i}{a_i}})}$$

$$= \frac{M(e(g,g)^{wr})}{e(g^r, g^{w-s}) \cdot e(g,g)^{s \cdot r}} = \frac{M(e(g,g)^{wr})}{e(g,g)^{w \cdot r}} = M$$

4.3. Data Block Management Scheme

4.3.1. Distribution Process of the Block Access Token

When clients ask NameNode for data storage, NameNode secures DataNode for data storage, and then a block access token is created for user authentication. The block access token is divided into three pieces based on the shamir (2, 3) secret sharing to be stored by the clients, NameNode, and DataNode (Figure 4).

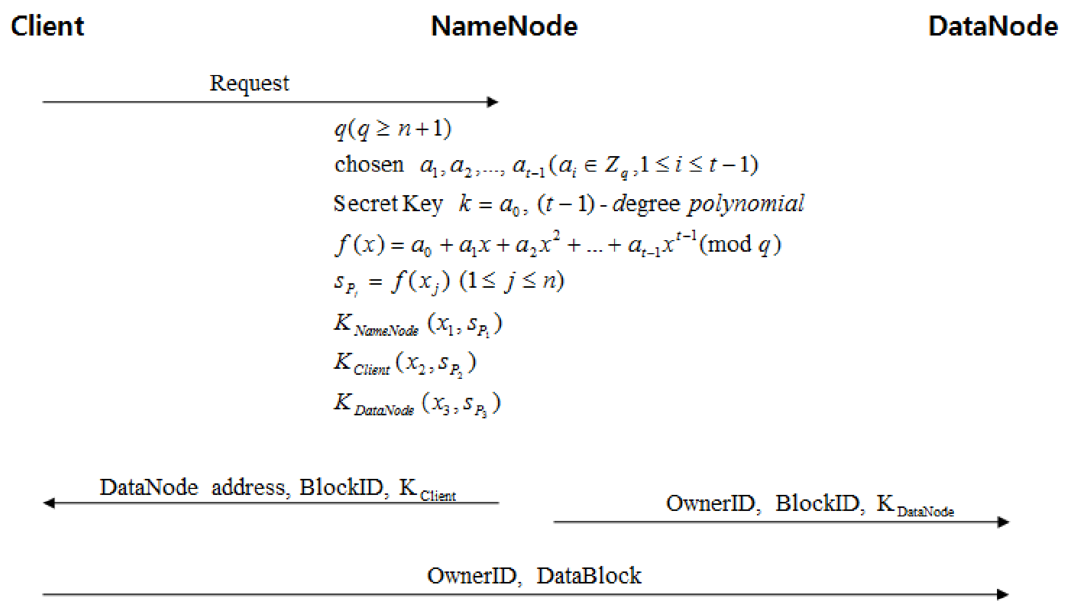
Client
NameNode
DataNode
Request
$q(q \geq n+1)$
chosen $a_1, a_2, \ldots, a_{t-1} (a_i \in Z_q, 1 \leq i \leq t-1)$
Secret Key $k = a_0$, $(t-1)$-degree polynomial
$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1} (\mathrm{mod}\ q)$
$s_{P_i} = f(x_j)\ (1 \leq j \leq n)$
$K_{NameNode}(x_1, s_{P_1})$
$K_{Client}(x_2, s_{P_2})$
$K_{DataNode}(x_3, s_{P_3})$
DataNode address, BlockID, K_{Client}
OwnerID, BlockID, $K_{DataNode}$
OwnerID, DataBlock

Figure 4. Secret distribution of block access token.

4.3.2. User Authentication Process

When data are recovered, clients receive information necessary for the block access token from NameNode (Figure 5). The details are as follows:

Step 1. When clients ask NameNode for data, NameNode sends the DataNode address where the data are stored, the data block ID, and its own single share to the clients.

Step 2. Upon receiving NameNode's single share, the clients create a block access token using their own single share.

Step 3. The clients send the created access token and NameNode's single share to the DataNode address received from NameNode.

Step 4. Using its own single share and NameNode's single share received from the clients, DataNode creates a block access token and compares it with the block access token received from the clients to verify if the clients are normal users.

Step 5. In the created block access token structure, Datablock Checksum is added when TokenAuthenticator is created, and accordingly, it is used for verifying the integrity of the stored data block (Figure 6).

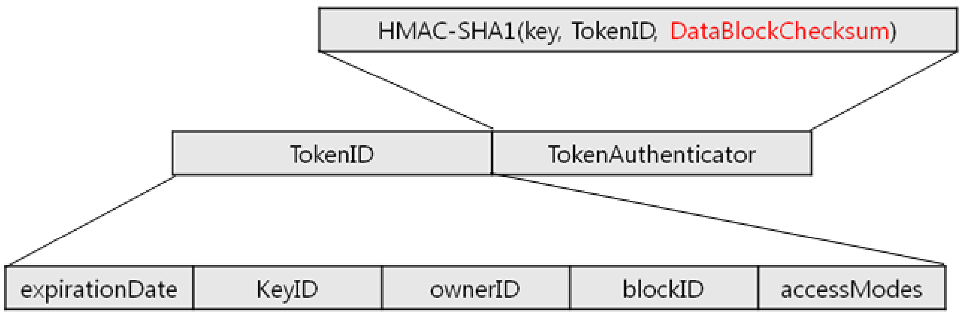

Figure 5. User authentication using block access token.

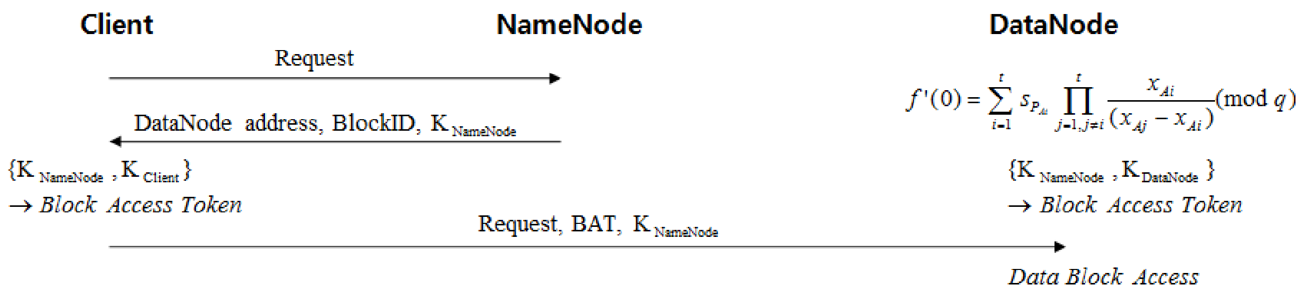
Client
NameNode
DataNode
Request
DataNode address, BlockID, K NameNode
{K NameNode , K Client }
→ Block Access Token
Request, BAT, K NameNode
$f'(0)=\sum_{i=1}^{t} s_{P_{Ai}} \prod_{j=1,j\neq i}^{t} \frac{x_{Ai}}{(x_{Aj}-x_{Ai})} (\mathrm{mod}\ q)$
{K NameNode , K DataNode }
→ Block Access Token
Data Block Access

Figure 6. Improved structure of block access token.

5. Analysis of Proposed Method

5.1. User Authority Management Scheme

5.1.1. Resistance to Collusion

The existing ABE has a limitation in that two or more users can collude using their attributes to arbitrarily access data for which they have not been authorized. For instance, users A and B can use user A attributes (A,B) and user B attributes (B,C) to access the ciphertext that consists of access structure (A,C). However, the proposed method can fundamentally block illegal data access through collusion because the manager group directly manages user classes with secret distribution. That is, the user classes can be adjusted to (1,3) threshold, (2,3) threshold, or other, according to system policy by distributing the secret key to secret pieces, such as $K_{MG1}(x_1,s_{P_1}),K_{MG2}(x_2,s_{P_2}),K_{MG3}(x_3,s_{P_3})$.

5.1.2. Data Availability

The availability of data stored in distributed servers is provided by all methods. HDFS reproduces and stores at least three of each data block to respond to single point errors first. In addition, this system maximally increases data availability by frequently inspecting the errors or loss of data blocks stored in data nodes.

5.1.3. Forward Secrecy

Forward secrecy is a feature that ensures that when certain user attributes have been withdrawn, the relevant users cannot obtain the data required by the relevant attributes. The proposed method provides forward secrecy because individual pieces x in the expression $f'(0)=\sum_{i=1}^{t} s_{P_{Ai}} \prod_{j=1,j\neq i}^{t} \frac{x_{Ai}}{(x_{Aj}-x_{Ai})} (\bmod q)$ calculated with secret pieces gathered based on secret distribution are created considering the access structures of attributes. That is, when the attributes of encrypted data change, illegal data access can be prevented, unless user attributes also change.

5.1.4. Reliability by Secret Distribution Technique

The method proposed in this study was based on the conventional CP-ABE and used secret distribution technique. By using the secret distribution technique, it was possible to overcome the shortcoming that, when the size of the attribute set $L=\lfloor L_1, L_2, \ldots, L_n \rfloor$ increases, the computational burden increases. However, in the process of encrypting data, each time, the secret key is segmented and shared. The distribution process is based on a polynomial such as $f(x)=a_0+a_1x+a_2x^2+\ldots+a_{t-1}x^{t-1} (\bmod q)$ and it is much more efficient than the computational burden generated by increase of attributes.

5.2. Data Block Management Scheme

5.2.1. Exposure of the Block Access Token Information

In the typical Hadoop structure, the secret key is frequently renewed between the NameNode and DataNode, and upon request from the users, the block access token is encrypted and transferred using an appropriate secret key. The technique using a hash chain is the concept of adding a disposable token to the block access token. Data are exposed without the separate encryption process, so blockID, userID, and keyID may be used by attackers. In other typical methods, the typical Hadoop environment is still used, and accordingly, the problems of Hadoop remain.

In this proposal, the block access tokens created during the communication process of single shares are transferred using the hash function. Only the rightful users or DataNode can create a block access token to take the hash function, and verify themselves as rightful users.

5.2.2. Man-in-the-Middle (MITM) Attack

In the typical methods, when a secret key between NameNode and DataNode is renewed, attackers can intercept the key to temporarily create a block access token. The hash chain technique and this proposal are suggested to prevent these Man-in-the-Middle (MITM) threats. As other typical methods use the typical Hadoop environment, the Hadoop problems remain.

5.2.3. Operation Amount

In the typical Hadoop system, the secret key is frequently renewed, and when a block access token is created, an operation based on a symmetric-key algorithm is conducted. Various symmetric-key algorithms, such as Advanced Encrypted Standard (AES) and Data Encryption Standard (DES), can be used. In the hash chain technique, each data block uses random numbers for n times of hash operation. In this proposal, however, the block access token is transferred together with the single share only once after taking the hash function, and after the initial single share is created and distributed. Therefore, this method is efficient.

5.2.4. Prevention of False Attacks

In the typical methods [10], user access control is conducted using Access Control List (ACL), but the Access Control Module is not enough for preventing false user data access. In this proposal, the block access token needed for data access is created only by the rightful constructors, and as such, false attacks can be prevented.

5.2.5. Data Integrity Test

In the typical Hadoop system and the method in [14], the data block integrity and the data storage server's errors are frequently tested through the Heartbeat message. The Heartbeat message is frequently transferred by DataNode to NameNode. In the method of [10], the data block integrity is also frequently confirmed through checksum, but the operation amount increases accordingly. In the method in [11], the effectiveness is confirmed using the index information created in the object metatable. In this proposal, verification can be conducted through single share, which is necessary for the creation of the block access token, and no frequent confirmation through the Heartbeat message is necessary. Therefore, this method is efficient.

5.2.6. Data Availability

All the methods have the availability of the data stored in the distributed server. In the Hadoop distributed storage system, three or more data blocks are copied and stored to cope with the single-site errors. In addition, error and loss tests on the data block stored in DataNode are frequently conducted to enhance data availability.

Table 1. Analysis of proposed scheme.

Security Requirements	**[15]**	**[16]**	**[8]**	**[17]**	**[9]**	**Proposed Scheme**
Data confidentiality	○	○	○	○	○	○
Collusion resistance	×	×	△	○	×	○
Forward secrecy	×	×	△	×	×	○
Backward secrecy	×	×	△	×	×	×
Data sharing	○	○	○	○	○	○
Calculation to restore secrets	–	–	–	–	–	Polynomial
Number of times of distribution of secret pieces	–	–	–	–	–	n

6. Conclusions

In this paper, to solve the problem where various sensitive data stored in cloud servers could be leaked because of external attackers or internal user poor management, a technique was proposed that is related to data access management by user authority based on ABE and the secret distribution technique. To solve the problem where user data access authorities cannot be managed using existing simple encryption methods, the ABE method that encrypts data based on diverse user attributes was applied, and to control access by user class among users with legitimate attributes, access control by class through secret distribution was proposed. The problem of managing the secret key shared by DataNode and NameNode, and of NameNode management and MITM risks, were solved. This method is more efficient than the typical methods, which can reduce the operation overhead concentrated to NameNode.

Sensitive information stored in cloud servers can be managed safely by allowing minor user access control in cloud environments. The proposed protocol is of a structure that can safely and efficiently control access to diverse high-capacity data, including personal user information of highly confidential data, and it is expected to be used efficiently in cloud computing environments. However, our proposed method has a limitation where more calculations in polynomial expressions are added compared with the existing ABE method because it provides more functions. In the future, more efficient and safer methods should be studied based on the proposed method.

Acknowledgments

This research was supported by Ministry of Science, ICT & Future Planning (MSIP), Korea, under the Information Technology Research Center (ITRC) support program (NIPA-2014-H0301-14-1010) supervised by National IT Industry Promotion Agency (NIPA).

This work was supported by the Soonchunhyang University Research Fund.

Author Contributions

Su-Hyun Kim and Im-Yeong Lee both designed research and wrote the paper. Both authors have read and approved the final manuscript.

Conflicts of Interest

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

1. O'Malley, O.; Zhang, K.; Radia, S.; Marti, R.; Harrell, C. *Hadoop Security Design*; Yahoo Inc.: Santa Clara, CA, USA, 2009.
2. Sahai A.; Waters, B. Fuzzy Identity-based Encryption. In *Advances in Cryptology—EUROCRYPT 2005*; Springer Berlin Heidelberg: Heidelberg, Germany, 2005; pp. 457–473.
3. Shamir, A. Identity-based Cryptosystems and Signature Schemes. In *Advances in Cryptology*; Springer Berlin Heidelberg: Heidelberg, Germany, 1985; pp. 47–53.
4. Ghemawat, S.; Gobioff, H.; Leung, S.-T. The Google File System. In Proceedings of the Nineteenth ACM symposium on Operating Systems Principles, Bolton Landing, NY, USA, 19–22 October 2003; pp. 29–43.
5. Hadoop Wiki. Available online: http://wiki.apache.org/hadoop/PoweredBy (accessed on 10 March 2015).
6. Apache Hadoop. Available online: http://hadoop.apache.org/ (accessed on 10 March 2015).

7. Kar, J.; Majhi, B. A Novel Deniable Authentication Protocol based on Diffie-Hellman Algorithm Using Pairing Technique. In Proceedings of the 2011 International Conference on Communication, Odisha, India, 12–14 February 2011; pp. 493–498.
8. Yu, S.; Wang, C.; Ren, K.; Lou, W. Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing. In Proceedings of 2010 IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010; pp. 1–9.
9. Zhu, S.; Yang, X.; Wu, X. Secure Cloud File System with Attribute Based Encryption. In Proceedings of 2013 5th International Conference on Intelligent Networking and Collaborative Systems (INCoS), Xi'an, China, 9–11 September 2013; pp. 99–102.
10. Jin, S.; Yang, S.; Zhu, X.; Yin, H. Design of a Trusted File System Based on Hadoop. In *Trustworthy Computing and Services*; Springer Berlin Heidelberg: Heidelberg, Germany, 2013; pp. 673–680.
11. Zhang, D.-W.; Sun, F.-Q.; Cheng, X.; Liu, C. Research on Hadoop-based Enterprise File Cloud Storage System. In Proceedings of 2011 3rd International Conference on Awareness Science and Technology (iCAST), Dalian, China, 27–30 September 2011; pp. 434–437.
12. Hubbard, D.; Sutton, M. Top Threats to Cloud Computing V1.0. Available online: https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf (accessed on 19 March 2015).
13. Shamir, A. How to Share a Secret. *Commun. ACM* **1979**, *22*, 612–613.
14. Park, S.-J.; Kim, H. Improving Hadoop Security Through Hash-chain. *J. Korean Inst. Inf. Technol.* **2012**, *6*, 56–73.
15. Cheung, L.; Newport, C. Provably Secure Ciphertext Policy ABE. In Proceedings of 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 29 October–2 November 2007; pp. 456–465.
16. Goyal, V.; Pandey, O.; Sahai, A.; Waters, B. Attribute-based Encryption for Fine-Grained Access Control of Encrypted Data. In Proceedings of the 13th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 30 October–3 November 2006; pp. 89–98.
17. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-Policy Attribute-based Encryption. In Proceedings of IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 20–23 May 2007; pp. 321–334.