JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 33, 695-712 (2017)
DOI: 10.6688/JISE.2017.33.3.6

Computation Efficient Attribute Based Broadcast Group Key Management for Secure Document Access in Public Cloud

PERUMAL PANDIARAJA[1], PANDI VIJAYAKUMAR[2,*],
VARADARAJAN VIJAYAKUMAR[3] AND RAMAN SESHADHRI[4]
[1]*Department of Computer Science and Engineering*
Arunai Engineering College
Thiruvannamalai, 606603 India
[2]*Department of Computer Science and Engineering*
University College of Engineering Tindivanam
Melpakkam, Tamilnadu, 604001 India
[3]*School of Computing Science and Engineering*
VIT University
Chennai Campus, 600127 India
[4]*GAVS Technologies Pvt. Limited*
Chennai, 600119 India
E-mail: {sppandiaraja; vijibond2000; seshadhri.raman21}@gmail.com; vijayakumar.v@vit.ac.in

An efficient fine-grained encryption-based access control scheme for documents stored in the public cloud network proposed by Mohamed *et al*. utilizes more computational and storage complexities. Although, Mohamed *et al.*'s broadcast key management scheme communicates group key securely, it consumes massive computational and storage resources for frequently changing the group key. In order to avoid these problems, we have proposed a new broadcast group key management scheme that takes reduced computational resources by performing minimal cryptographic operations in the data owner and cloud user's to update and recover the group key using a matrix called access control vector. The computation complexity of the data owner is reduced by performing simple arithmetic operations for updating the group key. Moreover, the computational complexity of the cloud user is also minimized by performing only one addition and two subtraction operations whenever there is a change in the group or access control policy. In addition, the proposed scheme also minimizes the storage complexity of cloud users, but maintains the same communication complexity as that of Mohamed *et al.*'s scheme. The performance results show that the proposed scheme is computationally efficient in the data owner and the cloud users.

Keywords: access controls, cryptographic controls, data encryption, data sharing, computation time

1. INTRODUCTION

Many organizations use XML as an important markup language for information distribution and data sharing among various user communities. In such a scenario, the organization would place a large amount of documents to be shared among the users on a Cloud Service Provider (CSP). However, such cloud providers are not always trusted and hence may not maintain confidentiality of the documents placed in the CSP by the organization. Therefore, maintenance of confidentiality is a major concern for many or-

Received July 8, 2016; accepted August 21, 2016.
Communicated by Balamurugan Balusamy.
* Corresponding author.

ganizations that utilize the facility of access control mechanism in a public cloud. In order to provide access control policy mechanisms in public cloud for restricting unauthorized users from accessing the documents, various key management schemes are used [1-3, 22]. However, in all the key management schemes, the provision of access control facility is a challenging task. This is due to the fact that handling the process of key generation and distribution are more complex when the subdocuments are distributed to a group of users in a secure way from the cloud servers. Moreover, the document dissemination is performed in a dynamic way and hence the users are allowed to join or depart service at any time. When a new user joins into the service, it is the responsibility of the data owner to prevent the new user from having access to previous documents to provide backward secrecy. Similarly, when an existing group user leaves from any group, such a user should not have further access to the documents to achieve forward secrecy. In order to handle the issues of forward and backward secrecy, the keys are updated frequently whenever a user joins or leaves the service.

The data owner takes the responsibility of generating a new group key subsequent to user join and leave operations. After a user joins the group or leaves the group, a new group key is generated and is shared to the group users in a secure way. In order to do that, when membership changes in the dynamic group, it computes common public information and the access control vector at the data owner and it is sent as a broadcast message to all the cloud users. In order to compute and to distribute the access control vector, many researchers have worked on broadcast key management schemes [1, 2]. However, most of these schemes consume more key computation time and memory [1, 2, 4, 5]. In order to tackle these issues, it is necessary to propose a new key management scheme. Therefore, in this paper, a new computation efficient group key management scheme is proposed.

In all the existing Broadcast Group Key Management (BGKM) schemes [1, 2, 4, 5], initially the data owner creates various access control policies for various subdocuments user communities to define which subject can access which objects under which method. These Access control policies are created using attribute conditions. An attribute condition [3] is an expression of the form "attrib op val", where attrib is the identity attribute, op is a comparison operator such as $=, <, >, \leq, \geq, =, \neq$ and val is a value that can be used for the attribute (*eg*., "*role* = *doctor*"). Thus, an Access Control Policy (ACP) is a tuple (A, S, D) where A denotes attribute conditions, S denotes subdocuments and D denotes document. A is a conjunction of attribute conditions ($con_1 \wedge con_2 \ldots \wedge con_n$) that must be satisfied by a user to have access to a subdocument (S). There will be only one main document (D) and there can be more than one subdocument (S) for an organization. After creating ACPs, these ACPs are implemented by an access control mechanism. While implementing ACPs, it is necessary to consider various cases [4]. First, the same access control policy could be applied to a set of subdocuments. Second, different access control policies could be applied to different subdocuments within the same document. Finally, the ACPs must include various conditions to access the documents from the cloud. Moreover, the number and type of subject are not known before implementing ACPs and hence it is not possible to create ACPs based on the user IDs. Therefore, it is necessary to create the ACPs based on the user IDs, roles, age and qualifications. These properties are called as Identity attributes of a subject. However, it is very difficult to create ACPs for selectively distributing the documents among users.

Consider, for example, the case of an Electronic Health Records (EHRs) [3] in which different level of hospital employees (users) is allowed to view different subdocuments by encrypting and storing them on the cloud. The hospital employees are divided into various categories such as receptionist, cashier, doctor, nurse, pharmacist, system administrator (data analyst) and non-employees such as patients. The EHR document is divided into various subdocuments such as Billing Information, Contact Information, Medication, Physical Exam, Lab Records and so on. In this example, various ACPs are created to allow different hospital users to view different encrypted sub-documents. For example, an ACP for a cashier is created in such a way that he/she should not have access to any other documents except for the Billing Information document. In contrast to this, a doctor or a nurse should not have access to Billing Information. However, they can access other documents related to a patient with disease and medication history. The following are some of the ACPs that are created for the "EHR.xml" that contains various subdocuments related to health records.

1. *ACP*1 = ("*role = receptionist*", {*Contact Information*}, "*EHR.xml*")
2. *ACP*2 = ("*role = cashier*", {*Billing Information*}, "*EHR.xml*")
3. *ACP*3 = ("*role = doctor*", {*Clinical Records*, *Lab Records*, *Physical Exams*, *Plan*, *Medication*}, "*EHR.xml*")
4. *ACP*4 = ("*level* ≥ 59" ∧ "*role = nurse*" {*Contact Information*, *Medication*, *Physical Exams*, *Lab Records*, *Plan*}, "*EHR.xml*")
5. *ACP*5 = ("*role = data analyst*", {*Contact Information*, *Lab Records*}, "*EHR.xml*")
6. *ACP*6 = ("*role = pharmacist*", *Billing Information*, *Medication*, "*EHR.xml*"

Similarly, "*EHR.xml*" is divided into various subdocuments based on these access control policies defined above:

- *Contact Information*: *ACP*1, *ACP*4, *ACP*5
- *Billing Information*: *ACP*2, *ACP*6
- *Medication*: *ACP*3, *ACP*4, *ACP*6
- *Physical Exams*: *ACP*3, *ACP*4
- *Lab Records*: *ACP*3, *ACP*4, *ACP*5
- *Plan*: *ACP*3, *ACP*4
- {*Clinical Records*}: *ACP*3
- *Others*: *none*

After the creation of ACPs, the data owner has to generate various group keys to encrypt various documents ac-cording to the ACPs. However, the encryption alone is not sufficient for an organization to enforce fine-grained access control on their data. Because, fine-grained access control is based on information such as role or project of users on which the users are working in the organization. Moreover, a single subdocument can be accessed by different credentials of users in an organization. In order to support fine-grained access control and multiple access, Policy Configurations are specified in an expressive way. The set of access control policies that can be applied for a subdocument is called as Policy Configuration (PC). There can be multiple subdocuments that fall under same policy configuration. The policy configurations and their associated subdocuments are:

- ✓ *PC*1 = *Contact Information*: *ACP*1, *ACP*4, *ACP*5
- ✓ *PC*2 = *Billing Information*: *ACP*2, *ACP*6
- ✓ *PC*3 = *Medication*: *ACP*3, *ACP*4, *ACP*6
- ✓ *PC*4 = *Physical Exams*: *ACP*3, *ACP*4
- ✓ *PC*5 = *Lab Records*: *ACP*3, *ACP*4, *ACP*5
- ✓ *PC*6 = *Plan*: *ACP*3, *ACP*4
- ✓ *PC*7 = {*Clinical Records*}: *ACP*3
- ✓ *PC*8 = *Others*: *none*

The organization (Data owner) creates ACPs and PCs for many subdocuments enclosed within a main document (*e.g.*, EHR.xml) and store them in the CSP. In order to do that, the organization divides the documents into subdocuments based on the PC and stores them in the cloud. When $user_1$ submits his/her original *id* and attribute conditions, the data owner finds a match for the supplied attribute conditions with an access control policy. After finding the match, the data owner delivers a set of secrets based on the attribute conditions to $user_1$. In a similar way, $user_2$, …, $user_n$ obtains secret keys when attribute conditions are matched with the access control policies. After that, the data owner chooses a group key for a policy configuration and encrypts the group key using all the entire user's secret key which is a coarse grained encryption to inform the group keys to users in a secure way. To encrypt the group keys, various group key management algorithms are available [1-3]. In all the group key management algorithms, the group key is informed to the group members by creating an Access control Vector (ACV) for each subdocument. After that, one (or) more subdocuments coming under same policy configuration are encrypted with a same group key. The encrypted subdocuments are stored in the cloud along with its ACV and the hash value which is computed using the ACVs and group key values. The cloud users use the ACV to derive the group key. In order to do this, each cloud user receives the ACV in a secure way from the data owner. After receiving the ACV, the users first decrypt the group key with his/her set of secrets using key recovery procedure which is a fine grained decryption. Finally, the users can decrypt one (or) more subdocuments downloaded from the CSP using the group key which falls under same policy configuration. In this scenario, the security of the documents being communicated between the data owner and the cloud users is preserved.

However, the privacy of the users is not preserved since each user submits their own identity to the data owner for accessing the documents. In order to preserve the privacy of each user, a Trusted Third Party (TTP) is introduced in the existing approaches [1, 2]. At first, the users submit their identity attributes to TTP that issues identity tokens to the users. An identity token contains the user's pseudonym, a tag that identifies the identity attribute, cryptographic semantically secure commitment of a user's identity attribute value and a digital signature. Identity tokens are used by the users in registering them to the data owner. Each user submits their identity tokens and receives a set of secrets from the data owner based on each identity attribute that matches with the access control policy condition. However, the computation complexity of data owner and cloud users in the existing ACV-BGKM [1] approach is high. In order to improve this, in this paper, a new computation efficient key management scheme is developed. A key advantage of the proposed broadcast key management scheme is that adding/revoking users can be performed in a computation efficient way. The major objectives of the proposed work are as follows:

- To propose an effective broadcast group key management scheme in order to minimize the computational complexity from $O(n^3)$ to $O(n)$ for updating the group key in the data owner when group membership changes.
- To propose a technique in order to minimize the computational complexity of the key recovery process used in the cloud user from $O(n)$ to $O(3)$.
- To propose a technique in order to reduce the storage complexity from $O(n)$ to $O(3)$ at the cloud user's for recovering the group key.
- To maintain the same communication complexity as given in Mohamed *et al.*'s ACV-BGKM scheme by sending only minimum amount of information from the data owner to cloud user.

The remainder of this paper is organized as follows: Section 2 provides the literature survey of some of the past related works. Section 3 depicts the complete architecture of our proposed work. Section 4 gives a detailed explanation of our proposed work. Section 5 explains the security strength of our proposed work. Section 6 analyzes the performance of our proposed work with the other existing works. Section 7 gives the concluding remarks.

2. LITERATURE SURVEY

Cloud services are deployed in the cloud servers using four methods, namely private, public, community and hybrid cloud. In public cloud, the services are available to the general public users and are controlled by data owner who stores the documents and a third party Cloud Service Provider (CSP) who maintains the documents. Since the documents are public in the public cloud, many users can access the documents located in the cloud service provider by getting access rights from the data owner. To enhance the confidentiality of the documents, an access control mechanism is to be implemented in public cloud networks. Access control mechanisms are used for restricting unauthorized users from accessing the documents. Many previous works on access control based security mechanisms are present in the literature which are used to secure the data stored in cloud servers. Among them, a Java based system that addresses the security issues of access control was proposed by Elisa *et al.* [6] based on policy design for XML documents. This system supports the specification of policies at various granularities and considers the trust level of users to enforce access control. Generally, a Role Based Access Control (RBAC) model consists of four basic components; a set of users, a set of roles, a set of permissions and a set of sessions. Roles have several advantages since roles represent an organizational function. A role based access control model can directly support an organization's security policy so that it helps the administration. The RBAC model is widely used for access control management, both in closed and open systems (James *et al.* [7]) where authorizations are specified with respect to roles and not with respect to individual users. Each user can have more than one privilege since they can play more than one role at a time. Based on these roles, privileges are assigned to each of the roles, since managing fewer roles is much more efficient than managing most individual users. Because of its relevance, RBAC has been individually investigated by researchers (Elisa *et al.* [6], James *et al.* [7] and Barker [8]). Although RBAC has been

thoroughly explored, there are still significant application requirements which are not addressed by current RBAC models. To overcome this issue, a generalization of the RBAC model, called the Action Status based Access Control (ASAC) was proposed by Barker [8]. A key feature of the ASAC model is that a decision on an agent's request to access resources is determined by considering the agent's ascribed status. Another important criterion for security in distributed database systems is the location constraints. Therefore, an access control system must not only consider temporal constraints, but also considers the spatial constraints.

Group access control can be achieved by encrypting a message (document) using an encryption key with a large size. This key is dynamically generated for each session of the communication session. Therefore, this dynamically generated key which is developed using an effective key management scheme is known as the Group Key (*Gk*) that is shared to all legitimate users of a group to access a common data from the cloud server. This is necessary since the group membership in a group key management scheme is most likely to change dynamically. Whenever a new user join or an existing user leave from the group, the encryption key must be updated in order to prevent the leaving or joining user from accessing the data or messages from the future or prior communications as proposed by Poovendran *et al.* [9]. The issues of establishing and updating the group keys have been addressed by various Group Key Management schemes present in the literature (Kim *et al.* [10], Drira *et al.* [11] and Naranjo *et al.* [12]).

The process of generating, distributing and maintaining the keys is taken care by key management schemes. There are many key management schemes that are available in the literature (Vijayakumar *et al.* [13, 25-30], Yoon-Su Jeong *et al.* [14], and Jung-Yoon Kim *et al.* [15]). The two main types of key management schemes are centralized and distributed key management scheme and are used currently for providing security in group communication. In the centralized scheme, a trusted third party is used to control the activities of group members. Moreover, the trusted third party called key server (or) group center is used for interacting with the group users and to control them in the centralized key management scheme. This key server is responsible for generating and distributing the keys. In contrast to centralized key management scheme, the keys in a distributed key management scheme are computed and maintained with the coordination of group users.

Attribute Based Encryption (ABE) is a technique which is mainly used for managing the overlapping users with different credentials. Initially, the concept of ABE has been introduced by A. Sahai and B. Waters [16]. This ABE system is limited only to fixed policies with less number of common attributes. M. Pirretti *et al.* [17] overcame the drawback by choosing a very large value where number of common attributes is big in number. However, this scheme was not significant for the privacy of users was not preserved. V. Goyal *et al.* [18] introduced the idea of key-policy ABE (KP-ABE) systems and J. Bethencourt *et al.* [19] introduced the idea of ciphertext-policy ABE (CP-ABE) systems. These approaches fulfilled privacy, security and large number of overlapping users. However, these schemes did not handle group dynamics where users join and leave are frequent. Therefore, proxy based encryptions encryption schemes [20, 21] were adopted for the public cloud. These systems handled group dynamics efficiently. One of the important limitations of this approach is that duplicate encryptions are performed and hence the computational complexity is increased.

Di Vimercati *et al.* [22] identified this problem and proposed a solution to handle the data hosted in the public cloud. However, this approach does not support significant attribute based policies with overlapping users. Attribute Based Encryption (ABE) is a technique which is mainly used for managing the overlapping users with different credentials. Recently, Mohamed Nabeel *et al.* [1, 2] proposed an approach in the year 2013 and 2014 to support privacy and selectively disseminating documents in a secure way from the public cloud. However, the computation complexity of data owner in their ACV-BGKM approach is high and it is $O(n^3)$ where n is the number of cloud users. The computation complexity of (ACV-BGKM) scheme [1] is $O(n^3)$ because the data owner uses Gaussian-Jordan elimination method [23] to find the new access control vector whenever there is a change in the group. The computation complexity of cloud user is $O(n)$ because each cloud user computes a row using which n multiplication operation is performed to get the group key. Moreover, the storage complexity is also $O(n)$ both in the data owner and the cloud users. In addition to this, the communication complexity is $O(n)$.

The improved ACV-BGKM scheme has also become inefficient when bucketization and subset cover approaches are used. In bucketization approach, as the bucket size increases, improved ACV-BGKM becomes computationally inefficient. Moreover, in subset cover approach, the group dynamics are handled only for revoking cloud users. When numbers of joining cloud users are large, it makes use of basic ACV-BGKM for key generation and hence subset cover approach also becomes inefficient. Therefore, we propose a computation, storage and communication efficient group key management scheme in this paper. The main idea of proposed group key management scheme is to give some secrets to the cloud users based on the identity attributes from which it allows the users to recover the group key. The major advantage of this proposed group key management scheme is that it minimizes the computation complexity for updating the group key in the data owner side. In addition to this, it also minimizes the computation complexity of the cloud users.

3. SYSTEM ARCHITECTURE

The system architecture shown in consists of four components, namely, Data Owner, Cloud Service Provider (CSP), Token Generator, and cloud User. The data owner is the one who places the original documents in the public cloud that are accessed by the cloud users. A CSP operates a single or a collection of servers used to maintain the data owner's data. The token generator is used to generate a token which should be given to each cloud user to get a secret key from the data owner.

A cloud user is a person (or application acting on behalf of this user) who wants to access the data from the CSP. Initially, each cloud user must send their identity attributes to the token generator to get a token. The token generator receives the identity attributes from the cloud user and generates an identity token. After generating this token for a cloud user, based on their identity attributes, it gives the newly generated identity token to the cloud user and then sends the same identity token to the data owner for verification. The verification should be performed after giving the token to the cloud user, since the token generator has to send the identity token of a cloud user to the data owner only if it

is correctly delivered to that particular cloud user. After receiving confirmation from that cloud user only, the token generator will send the identity token to the data owner. All these processes are used as explained in the existing approach [1]. In this paper, we use the existing Pedersen commitment scheme [24] to preserve the privacy of the cloud user to hide the cloud user's identity from the data owner.

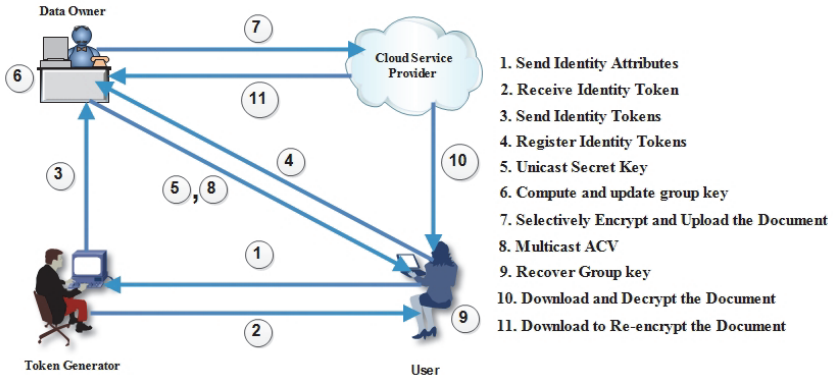

Fig. 1. Privacy preserving group key management architecture.

The cloud user registers their identity token with the data owner to get a secret key. The data owner gives secret keys to the cloud users based on their identity token. After providing the cloud user with a secret key, the data owner generates a Group Key (GK) for each group of users. After that, the data owner encrypts the group key based on their individual secret key values. Then, the data owner broadcast the encrypted group key in an ACV format to the cloud users along with their index value. After that, the data owner encrypts the subdocuments using group key and then uploads the encrypted subdocuments to the CSP. Each cloud user can derive (recover) the GK using their secret key and thus can use this GK to decrypt the encrypted documents placed in the cloud. When group membership changes, it is the responsibility of the data owner to change the GK. The data owner may also change the GK periodically. For example, when a user leaves or joins the group, the data owner downloads the corresponding document from the cloud service provider and re-encrypts the document with the new GK. Then, the re-encrypted document is uploaded into the cloud. In this paper, we mainly concentrate on developing a new broadcast group key management that takes care of the job of generating the secret and group keys and also updating them when group membership changes. In addition to this, we have also introduced a new key recovery process in this paper to allow the cloud users for finding the group key from ACV value. Hence, our proposed work concentrates from Steps 3 to 9 in Fig. 1.

4. PROPOSED BROADCAST GROUP KEY MANAGEMENT SCHEME

In this section, we provide a detailed explanation about our proposed broadcast group key management scheme (PBGKM). The PBGKM scheme provides forward secrecy, backward secrecy, and collusion resistance. In addition to these facilities, our

proposed scheme is also efficient in terms of computation and storage complexities. The proposed broadcast group key management (PBGKM) works in four phases. The first phase is the Data owner initial setup phase and user join phase, where a multiplicative group is created by the data owner. In this phase, the cloud users send join requests to the data owner and obtain all the necessary keys for computing the group key in order to download the documents from the cloud. The second phase is Group key computation phase, where the data owner generates and encrypts the group key in ACV format. The third phase is called as Group key recovery phase that deals with the key recovery process used in the cloud users. The final phase is Group key updating phase that deals with updating of the group key when there is a change in the dynamic group in order to provide forward/backward secrecy.

4.1 Data Owner Initial Setup Phase and User Join Phase

In this phase, the data owner selects a large prime number (p) to define a multiplicative group z_p^* which is used to generate the secret keys (δ_i)($1 \le i \le n$) and random seed keys (r_i)($1 \le i \le n$) for n number of users with respect to the prime number (p). In addition to this, it also generates an arbitrary value 'ρ' The secret keys (δ_i), random seed keys (r_i) and arbitrary value 'ρ' are used for updating the group key at the data owner. The secret and seed keys corresponding to legitimate cloud users are used in the cloud users to recover the updated group key. Initially, when the cloud user sends join request to the data owner, the data owner informs the secret keys (δ_i) and its corresponding seed keys (r_i) to all the existing cloud users in a secure way using Secure Socket Layer (SSL). For example, when a new user 'i' is authorized to join a group to access the documents stored in the public cloud for the first time, the data owner sends a secret key (δ_i) and its corresponding random seed key (r_i) with respect to their identity (pseudonym value). This secret key (δ_i) and random seed key (r_i) are known only to the cloud user (Usr_i) and to the data owner. All the cloud users of the group store the secret key (δ_i) and the random seed key value (r_i) in their respective storage area.

4.2 Group Key Computation Phase

In this phase, the data owner generates and encrypts the group key in the following ways to create the ACV for a particular subdocument and broadcast it to the cloud users of the group.

1. Initially, data owner creates a column vector 'A' for n number of users. If there are 'n' users in a group, it constructs (n+1) column vector in which the first element of the column vector is a group key γ and remaining elements $a_{(i,1)}$($1 \le i \le n$) are calculated with respect to Eq. (5) and are placed in the column vector.

$$A = \begin{pmatrix} \gamma \\ a_{1,1} \\ a_{2,1} \\ a_{3,1} \\ \vdots \end{pmatrix} \tag{1}$$

2. After creating 'A', the data owner constructs an arbitrary numbered column vector 'B' equal to the size of 'A' that consist of zeros followed by an arbitrary number 'ρ' as shown in Eq. (2).

$$B = \begin{pmatrix} \rho \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{2}$$

In this column vector, the arbitrary value 'ρ' is randomly chosen from the group z_p^*.

3. Then, the data owner adds the column vector 'A' with the column vector 'B' to obtain a new column vector 'C' as shown in Eq. (3) and it is the initial process of encrypting the group key.

$$(A) + (B) = (C) \tag{3}$$

The result of this process can be expressed as shown in Eq. (4).

$$\begin{pmatrix} \gamma \\ a_{1,1} \\ a_{2,1} \\ a_{3,1} \\ \vdots \\ a_{n,1} \end{pmatrix} + \begin{pmatrix} \rho \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} \beta \\ a_{1,1} \\ a_{2,1} \\ a_{3,1} \\ \vdots \\ a_{n,1} \end{pmatrix} \tag{4}$$

In general, data owner obtains value of $(a_{i,1})$ using the formula shown in Eq. (5).

$$(a_{i,1}) = (\gamma + \beta - \delta_i) \text{ where } (1 \le i \le n) \tag{5}$$

4. Next, the data owner creates a column vector 'R' and places all the random seed key values (r_i) in that column vector followed by a value zero as shown in Eq. (6).

$$R = \begin{pmatrix} 0 \\ r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{pmatrix} \tag{6}$$

5. This column vector 'R' is added with the vector 'C' to produce the access control vector ACV_i of a subdocument S_i as shown in Eq. (7).

$$\begin{pmatrix} \beta \\ a_{1,1} \\ a_{2,1} \\ a_{3,1} \\ \vdots \\ a_{n,1} \end{pmatrix} + \begin{pmatrix} 0 \\ r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{pmatrix} = \begin{pmatrix} \beta \\ x_{1,1} \\ x_{2,1} \\ x_{3,1} \\ \vdots \\ x_{n,1} \end{pmatrix} = ACV_i \tag{7}$$

6. Finally, the data owner broadcasts the ACV_i value along with their index value to all the cloud users present in the group. The index value is used by the cloud users in order to extract a particular value according to their positions from the ACV_i value.

4.3 Group Key Recovery Phase

In this phase, the group key encrypted by data owner is decrypted by the authorized cloud users using their own random seed key (r_i) and secret key (δ_i) in the following ways:

1. Each cloud user of the group receives the broadcast message and extracts β value and $x_{i,1}$ alone from the ACV_i according to their positions indicated by the index value. Using these two values, each cloud user subtracts their random seed key from $x_{i,1}$ in order to get $a_{i,1}$ which is the first step in finding the group key as shown in Eq. (8).

$$(x_{i,1}) - (r_i) = (a_{i,1}) \tag{8}$$

2. In the second step of finding the group key, each cloud user adds their secret key (δ_i) with ($a_{i,1}$) and subtracts β from the resultant value to obtain the group key as shown in Eq. (9).

$$(a_{i,1} + \delta_i) - \beta = \gamma \tag{9}$$

3. Finally, each cloud users can decrypt the subdocument S_i downloaded from the public cloud using the corresponding group key (γ).

The proposed scheme does not change privacy preserving scheme of Mohamed *et al.*'s scheme. The proposed approach focuses only on enhancing efficiency of the group key management scheme used in their approach so that the group key is updated both when a user joins and leaves a group with minimum computational and storage complexities.

4.4 Group Key Updating Phase

The final phase of this scheme is called as key updating phase where the data owner updates (changes) the group key when a user joins/leaves the dynamic group. The data owner also updates the group key when the cloud user's credentials have to be updated dynamically from time to time for various reasons such as promotions, change of re-

sponsibilities, and so on. If a user updates his/her credentials with the data owner, then the data owner needs to update the column vector A and B in order to update the ACV_i value only for the subdocuments involved.

The information to be updated by data owner in the column vector A and B are old group key (γ) to new group key (γ'), ($a_{i,1}$) to ($b_{i,1}$), arbitrary number (ρ) to (ρ') and β to β' respectively. The data owner obtains the value of ($b_{i,1}$) using the formula shown below:

$$(b_{i,1}) = (\gamma' + \beta' - \delta_i).$$

The updated information is sent as a broadcast message to the cloud users, from which all the existing cloud users of the group obtain the new group key (γ') using (β') and ($b_{i,1}$) respectively. When the data owner sends the updated group key in ACV_i format to the existing cloud users, it also includes the updated index value along with the new ACV_i value.

Theorem 1: The proposed key management scheme is correct.

***Proof*:** The correctness of proposed scheme can be easily proved as shown below:

$$(\gamma) = (a_{i,1} + \delta_n) - \beta$$
$$(\gamma) = (a_{i,1} + \delta_n - (a_{i,1} + \delta_n - \gamma))$$
(Since, $(\beta) = (a_{i,1} + \delta_n - \gamma)$)
$$(\gamma) = (a_{i,1} + \delta_n - a_{i,1} - \delta_n + \gamma) = (\gamma)$$

5. SECURITY ANALYSIS

This paper analyzes the proposed scheme for forward secrecy, backward secrecy and collusion resistance. The assumption of the implemented scheme is that an adversary might be a group member in prior and the data owner keeps the cloud users entire key values secretly which is also kept secret by the users.

5.1 Forward/Backward Secrecy

Computing the newly updated group key (γ') to break forward or backward secrecy in the PBGKM scheme depends on the method used to calculate the members secret key (δ_i) and random seed key value (r_i) in a particular amount of time. In the proposed work, the data owner broadcast ACV matrix to all the cloud users present in the group. Hence, an attacker (old or new user) will try to capture this matrix and by using his/her secret key and random seed key values, the attacker can try to find the value of updated group key (γ'). This updated group key (γ') can be computed only by using any one the existing user's secret and random seed key value. If the attacker is not an active adversary (*i.e.*, not a member of existing group), then the attacker can use brute force attack to learn about any one member's secret key δ_i and seed key value (r_i) If the size of δ_i is w bits, then the attacker has to use the total number of trials of 2^w to learn about secret key. Sim-

ilarly, the attacker has to use another 2^w trials to find the random seed key value and hence the attacker has to perform $2^{w+w} = 2^{2w}$ trials in total. The time taken to derive δ_i and r_i values can be increased by choosing large δ_i and r_i for each user's secret and seed key values. Therefore, when large size δ_i and r_i are used, it is infeasible for an adversary to find the value of γ' and hence it provides forward and backward secrecy.

5.2 Collusion Resistance

Collusion resistance is the one in which two or more adversaries outside the group cannot cooperatively compute the updated group key after leaving the group. Since, the group key and $a_{i,1}$ values are updated in the column vector A and arbitrary value is updated in column vector B after the leaving operation is performed, any number of previous users collision will not be used to derive the updated group key γ'. The following scenario describes a kind of collusion attack in which two adversaries act as legitimate members. Assume that u_1 is an adversary 'a' who knows the secret key value δ_1 and seed key value r_1 and u_3 is an adversary 'b' who knows the δ_3 and seed key value r_3 and group key γ at time $(t - 2)$. In time $(t - 1)$, the adversary 'a' leaves the group with the key values (δ_1, r_1) and γ. 'b' receives the rekeying message from the data owner at the time (t) and computes γ'. In time $(t + 1)$, 'b' leaves the group with the two key values (δ_3, r_3) and γ'. Both of these adversaries exchange their known key values and they have δ_1, r_1, δ_3, r_3, γ and γ'. Using these known values, the adversaries 'a' and 'b' cannot cooperatively find the updated group key $(\gamma')'$ which is will be broadcast at time $(t + 2)$ in a feasible amount of time.

6. PERFORMANCE ANALYSIS

In this section, we evaluate the performance of our PBGKM scheme with the existing ACV-BGKM scheme. Table 1 shows the computation, storage and communication complexities of ACV-BGKM scheme and our proposed broadcast group key management (PBGKM). From Table 1, it is clear to understand that the existing ACV-BGKM approach takes $O(n)$ hashing operations for computing the row vectors to be placed in the matrix.

In addition to this, it also takes $O(n^3)$ computational complexity for solving the matrix using the Gauss Jordan elimination method to find the inverse matrix. However, our PBGKM approach takes (n) additions in the data owner for adding the column vector 'R' with the vector 'C'. Moreover, it also takes (n) subtractions for computing $a_{i,1}$ values as shown in Eq. (5) during the group key updating in ACV format. Therefore, our modified PBGKM scheme takes less computational complexity in the data owner. It also takes only two subtractions and one addition operations in the cloud users in finding the group key, which is better than ACV-BGKM approach. With regard to the storage complexity, our PBGKM scheme stores only 3 values in the cloud users to recover the group key. Therefore, our proposed approach takes less computational and storage complexity by maintaining the same communication complexity.

Table 1. Comparison between ACV-BGKM scheme and PBGKM scheme.

Parameters	ACV-BGKM	PBGKM
Data owner's computation complexity	a) $O(n)$-Hash value computing time.	a) $O(n)$ additions
	b) $O(n^3)$-Gauss Jordan elimination time.	b) $O(n)$ subtractions for computing $a_{i,1}$ values.
	c) 1 addition.	c) 1 addition
Cloud user's computation complexity	a) $O(n)$-Hash value computation.	a) 2 subtractions.
	b) $O(n)$ multiplications	b) 1 addition.
	c) $O(n)$ additions.	
Storage complexity of data owner	$O(n\times(n+1))$ for storing the $n\times(n+1)$ matrix.	$O(n)$
Storage complexity of cloud user	$O(n)$	$O(3)$
Communication complexity	1 broadcast consists of ACV and public key values.	1 broadcast consists of ACV value and index value of users.

Table 2. Computation time of key updating and key recovery process.

No. of users	Key Size (bits)	Key updating Process (ms)		Key recovery process (ms)	
		ACV-BGKM	PBGKM	ACV-BGKM	PBGKM
1000	64	21773498	196	4564800	1
2000	64	44893209	246	9150123	1
3000	64	64320789	336	12989789	1
4000	64	86759987	396	18789985	1
5000	64	109123103	472	21999987	1
6000	64	129789569	526	27459789	1
7000	64	151456989	596	32123009	1
8000	64	175012412	706	36789899	1
9000	64	195769120	902	40899789	1
10000	64	209878978	1025	44759219	1

The proposed method has been implemented in JAVA (Intel Core i5-2450M CPU @2.50 GHz, 2GB RAM, 500 GB Hard disk, Microsoft's Windows 7 Professional 64-bit operating system) for a group of 10000 users and we have compared the group key computation time and recovery time for various key sizes in our proposed approach. For implementing this proposed approach, a private key and random seed key values are selected from the z_p^*. For generating large integers as secret key values in our program, we use BigInteger class that supports various methods for handling large positive integers. The methods multiply(), add() and subtract() supported by Biginteger class are used to perform multiplication, addition and subtraction operation in the large integer values. TABLE 2 shows the measured computation time in milliseconds (ms) for performing key updating process in the data owner and key recovery process in the cloud user's. It is evident from the values that the computation time of our PBGKM scheme is found to be better both in the data owner and the cloud user's than ACV-BGKM scheme.

7. CONCLUDING REMARKS

In this paper, a new computational and storage efficient broadcast group key management scheme has been proposed to improve the efficiency of Mohamed *et al.*'s ACV-BGKM scheme for accessing the selected documents from the CSP to the cloud users in a secure way. Even though, Mohamed *et al.*'s ACV-BGKM scheme is a secured one, the computation and storage complexity are extremely large. The proposed scheme has two dimensional focuses, namely minimal computation complexity and minimal storage complexity. The computation complexity of the data owner is $O(n)$ and cloud users computational complexity is $O(3)$. With respect to the communication complexity, the communication complexity of our proposed scheme is $O(1)$ which means that our proposed scheme takes only one broadcast message as that of Mohamed *et al.*'s scheme to inform the ACV value to the cloud users for finding the group key. The storage complexity of data owner is $O(n)$ and the cloud users storage complexity is $O(2)$. The further extension of this work is to devise a technique to handle document overlapping.

REFERENCES

1. M. Nabeel, N. Shang, and E. Bertino, "Privacy preserving policy-based content sharing in public clouds," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 25, 2013, pp. 2602-2614.
2. M. Nabeel, N. Shang, and E. Bertino, "Privacy preserving delegated access control in public clouds," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 26, 2014, pp. 2268-2280.
3. N. Shang, M. Nabeel, F. Paci, and E. Bertino, "A privacy-preserving approach to policy-based content dissemination," in *Proceedings of IEEE 26th International Conference on Data Engineering*, 2010, pp. 944-955.
4. E. Bertino and E. Ferrari, "Secure and selective dissemination of XML documents," *ACM Transactions on Info and System Security*, Vol. 5, 2002, pp. 290-331.
5. E. Bertino, S. Castano, E. Ferrari, and M. Andmesiti, "Specifying and enforcing access control policies for XML document sources," *World Wide Web Journal*, Springer, Vol. 3, 2001, pp. 139-151.
6. E. Bertino, P. A. Bonatti, and E. Ferrari, "TRBAC: A temporal role-based access control model," *ACM Transactions on Information and System Security*, Vol. 4, 2001, pp. 191-233.
7. J. B. D. Joshi, R. Bhatti, E. Bertino, and A. Ghafoor, "Access-control language for multidomain environments," *IEEE Transactions on Internet Computing*, Vol. 8, 2004, pp. 40-50.
8. S. Barker, "Action-status access control," in *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies*, 2007, pp. 195-204.
9. R. Poovendran and J. S. Baras, "An information-theoretic approach for design and analysis of rooted-tree-based multicast key management schemes," *IEEE Transactions on Information Theory*, Vol. 47, 2001, pp. 2824-2834.
10. Y. Kim, A. Perrig, and G. Tsudik, "Group key agreement efficient in communication," *IEEE Transactions on Computers*, Vol. 53, 2004, pp. 905-921.

11. K. Drira, H. Seba, and H. Kheddouci, "ECGK: An efficient clustering scheme for group key management in MANETs," *Computer Communications*, Vol. 33, 2010, pp. 1094-1107.
12. J. A. M. Naranjo, J. A. Lopez-Ramosz, and L. G. Casado, "A suite of algorithms for key distribution and authentication in centralized secure multicast environments," *Journal of Computational and Applied Mathematics*, Vol. 236, 2012, pp. 3042-3051.
13. P. Vijayakumar, S. Bose, and A. Kannan, "Chinese remainder theorem based centralized group key management for secure multicast communication," *IET Information Security*, Vol. 8, 2014, pp. 179-187.
14. Y.-S. Jeong, K.-S. Kim, Y.-T. Kim, G.-C. Park, and S.-H. Lee, "A key management protocol for securing stability of an intermediate node in wireless sensor networks," in *Proceedings of IEEE 8th International Conference on Computer and Information Technology*, 2008, pp. 471-476.
15. J.-Y. Kim and H.-K. Choi, "Improvements on Sun *et al.*'s conditional access system in pay-TV broadcasting systems," *IEEE Transactions on Multimedia*, Vol. 12, 2010, pp. 337-340.
16. A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proceedings of Eurocrypt*, LNCS 3494, 2005, pp. 457-473.
17. M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure attribute-based systems," in *Proceedings of the 13th ACM Conference on Computer and Communication Security*, 2006, pp. 99-112.
18. V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference of Computer and Communication Security*, 2006, pp. 89-98.
19. J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of IEEE Symposium on Security and Privacy*, 2007, pp. 321-334.
20. X. Liang, Z. Cao, H. Lin, and J. Shao, "Attribute based proxy re-encryption with delegating capabilities," in *Proceedings of the 4th International Symposium Information*, *Computer*, *and Communication Security*, 2009, pp. 276-286.
21. G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information System Security*, Vol. 9, 2006, pp. 1-30.
22. S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption: Management of access control evolution on outsourced data," in *Proceedings of the 33rd International Conference on Very Large Databases*, 2007, pp. 123-134.
23. D. Dummit and R. Foote, "Gaussian-Jordan elimination," *Abstract Algebra*, 2nd ed., Wiley, 1999, p. 404.
24. T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, 1992, pp. 129-140.
25. P. Vijayakumar, M. Azees, A. Kannan, and L. J. Deborah, "Dual authentication and key management techniques for secure data transmission in vehicular ad-hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 17, 2016, pp. 1015-1028.

26. P. Vijayakumar, S. Bose, and A. Kannan, "Centralized key distribution protocol using the greatest common divisor method," *Computers and Mathematics with Applications*, Vol. 65, 2013, pp. 1360-1368.
27. P. Vijayakumar, S. Bose, A. Kannan, and L. J. Deborah, "Computation and communication efficient key distribution protocol for secure multicast communication," *KSII Transactions on Internet and Information Systems*, Vol. 7, 2013, pp. 878-894.
28. P. Vijayakumar, S. Bose, and A. Kannan, "Rotation based secure multicast key management for batch rekeying operations," *Networking Science*, Vol. 1, 2012, pp. 39-47.
29. P. Vijayakumar, K. Anand, S. Bose, V. Maheswari, R. Kowsalya, and A. Kannan, "Hierarchical key management scheme for securing mobile agents with optimal computation time," *Procedia Engineering*, Vol. 38, 2012, pp. 1432-1443.
30. S. Audithan, T. S. Murunya, and P. Vijayakumar, "Anonymous authentication for secure mobile agent based internet business," *Circuits and Systems*, Vol. 7, 2016, pp. 1421-1429.

Perumal Pandiaraja received Bachelor of Engineering in Computer Science and Engineering degree from Anna University, Chennai, India, in 2006 and the Master of Engineering degree in Computer Science and Engineering from Anna University, Chennai, India in 2009. He is working as a part time Research Scholar in the Department of Computer Science and Engineering at University College of Engineering, Tindivanam. His research areas are key management, cloud security.

Pandi VijayaKumar completed his Ph.D. in Computer Science and Engineering in Anna University Chennai in the year 2013. He Completed Master of Engineering in the field of Computer Science and Engineering in Karunya Institute of Technology affiliated under Anna University Chennai in the year 2005. He completed his bachelor of Engineering under Madurai Kamarajar University, Madurai in the year 2002. He was working as a Senior Lecturer in All Nations University Ghana, West Africa from 2007 to 2008. He is presently working as a Dean at University College of Engineering Tindivanam which is a Constituent College of Anna University. His main thrust research areas are key management in network security and multicasting in computer networks.

Varadarajan Vijayakumar is a Professor in the School of Computing Science and Engineering at Vellore Institute of Technology Chennai campus. His primary research interest is in cloud computing, grid computing, big data and security. He received the BE and ME degree from Madras University and Ph.D. degree from Anna University and MBA degree from Periyar University. He has more than 14 years of experience in teaching and industry. He has more than 20 publications which includes Journals and Conferences. He is also the reviewer for Springer's Journal of Super Computing and many other journals.

Raman Seshadhri has completed his bachelor of Engineering in the field of Computer Science and Engineering department at University College of Engineering Tindivanam (A Constituent College of Anna University, Chennai). His research adviser is Dr. P. Vijayakumar. His research interests include computer networks, cryptography and network security. At present he is working as E4-Trainee-Systems in GAVS Technologies Pvt. Limited, Chennai.