International Journal of Latest Engineering and Management Research (IJLEMR)
ISSN: 2455-4847
www.ijlemr.com || Volume 03 - Issue 08 || August 2018 || PP. 58-64

An Effective Key Management Scheme for Secure Data Sharing in Public Cloud

Sam Silva. A[1], Jeya Praise. J[2]
[1,2](Computer Science and Engineering, Rajas Engineering College, Anna University, India)

Abstract: Designing a computation and communication efficient key management to selectively share the documents based on fine-grained attribute-based access control policies in public cloud is a challenging task. The data owner allow the cloud users to access the documents placed in the cloud service provider based on the user's access control vector provided to the cloud users by the data owners. In such type of scenario, it is important to maintain the confidentiality of the documents exchanged between the cloud service provider and the cloud users. The existing approaches used to provide this facility are not computation and communication efficient for performing key updating in the data owner side and the key recovery in the user side. Therefore, a computationally efficient key management scheme that allows the data owner to place the documents in a secure way in the cloud service provider. In addition to that, the proposed scheme also minimizes the storage complexity of data owner and cloud users. To show the effectiveness of the scheme simulations is performed on various key sizes that are used to encrypt the data.

Keywords: Multiplicative Inverse, Data Owner, Computation Time, Access Control Vector, Group Key Management.

I. Introduction

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 data and a third party Cloud Service Provider (CSP) who maintains the data. Services are provided to various cloud users by a vendor on the basis of pay-per-user policy. Since the data are public in the public cloud, many users can access the data located in the cloud service provider by getting access rights from the data owner. However, such cloud providers are not always trusted and hence may not maintain the confidentiality of the data placed in the CSP by the data owner. Therefore, maintenance of confidentiality is a major concern for many organizations that utilize the facility of accessing the data from a public cloud. To enhance the confidentiality of the data, an access control mechanism is to be implemented in public cloud networks. Access control mechanisms are used for restricting unauthorized users from accessing the data. Many previous works on access control based security mechanisms are present in the literature, which are used to secure the data stored on cloud servers. Among them, a Java based system that addresses the security issues of access control was proposed by Elisa et al. [1] based on policy design for XML documents.

This system supports 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. [2]) 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 few roles is much more investigated by researchers Elisa et.al. [1], James et. al. [2] and Barker et. al. [3]). 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 Action Status based Access Control (ASAC) was proposed by Barker et. al. [3]. 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 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 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 joins 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 proposed by Poovendran et. al. [4]. 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 [5], Drira et. al. [6] and Naranjo et. al. [7].

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 Vijayakumaret. al. [8], Yoon-suJeong et. al. [9] and Jung-Yoon kim et. al. [10]. There are two types of key management schemes, namely centralized and distributed key management schemes that are used at present for providing security in group communication. In the centralized scheme, a trusted third party is used to control the activities of group members. These activities include user registration, key generation, key distribution and group management. 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 managementscheme. 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 schemes are computed and maintained with the coordination of group users.

The provision of access control facility in centralized key managements is a challenging task. This is due to the fact that the handling of key generation and distribution are more complex when the messages are distributed to a group of users from the cloud servers where the number of users joining or leaving the group is more and dynamic. Therefore, in such a dynamic group communication, it is necessary to allow the users to join or depart the 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 data in order to provide backward secrecy in the secure group communication. Similarly, when an existing group user leaves from any group, such a user should not have further access to data which is available only to the existing group users in the cloud server to achieve forward secrecy. In order to handle the issues of forward and backward secrecy, the keys are updated whenever a user joins or leaves to or from the database service. The data owner takes the responsibility of generating a new group key after users join and leave operations are carried out. After a user joins the group or leaves the group, the new group key is generated and is shared between the data owner and group users in a secure way. In order to do that, when a membership changes in the dynamic group, it computes common public information and the access control vector in the data owner side in broadcast group key management (BGKM) scheme with massive computational complexity [11] and it is sent as a broadcast message to all the cloud users.

The major objectives of this paperwork are as follows:

- To propose an effective group key management scheme in order to enhance the security of multicast group communication performed in a public loud network.
- To propose a computationally efficient group key running algorithm in order to allow the cloud users to perform less mathematical operations to find the group key in a secure way.
- To propose a communication efficient group key management.
- To propose a technique to reduce the memory requirements of the group by making the group members to store less amount of information which is needed for finding the group key.

The major assumptions made in this paperwork are:

- The system can support several thousands of cloud users.
- The data owner keeps all cloud users private keys secret and each user keeps his or her own private keys secret.
- User joining and leaving behavior is independent of other cloud users.

The remainder of this paper is organized as follows: Section II discusses the proposed broadcast group key management scheme and a detailed explanation of the proposed work. Section III discusses the security analysis of the proposed work. Section IV analyzes the performance of the proposed key management scheme. Section V gives the concluding remarks.

II. PROPOSED VALUABLE KEY-RUNNING SCHEME (PVK-RS)

The proposed key running (key management) scheme focuses on enhancing security with minimum computation and communication time for the group-key running scheme, where the group key is updated both when a member joins and leaves a group without affecting forward and backward secrecy. The Proposed Valuable Key-Running Scheme (PVK-RS) scheme works in four phases. The first phase is the data owner initial setup, where a multiplicative group is created from which a secret key for each user and its multiplicative inverse value with respect to a prime number $^{(p)}$ are constructed at the data owner side. In this phase, the data

owner also generates a random seed key for each user. In the second phase called user join phase, the cloud users send join requests to the data owner and obtain a secret key and a random seed key for participation through secure channel. The third phase of this scheme is a key derivation phase in which the group key is encrypted and it is sent to a group of cloud users that can be used as an access control vector of a particular document. Each user derives the group key using their own secret key, a random seed key and encrypted information in a computationally efficient way. The final phase of this scheme is known as key updating phase that deals with updating of group key in the data owner side when user joins and leave operations are performed in the dynamic group in order to provide forward and backward secrecy.

1. Data Owner Initial Setup Phase

In this phase, the data owner selects a large prime number $^{(p)}$ to define a multiplicative group ^{Zp}, which is used to generate a secret keys $^{(\delta i)\ (1\le i\le n)}$ and its multiplicative inverse numbers $^{(\delta' i)\ (1\le i\le n)}$ with respect to prime number $^{(p)}$ for number of cloud users. In addition to this, the data owner also generates a random seed key $^{(ri)}$ $^{(1\le i\le n)}$ for each user, which will be used, for updating and recovering the group key in this approach. After that, the data owner informs the secret key value $^{(\delta i)}$ and its corresponding seed key value $^{(ri)}$ to all the existing cloud users in a secure way using Secure Socket Layer (SSL). After generating these values, the data owner executes the following steps in the initial setup phase.

1. It constructs a column vector (A) for number of cloud users

$$(A) = \begin{bmatrix} \delta'1 \\ \delta'2 \\ . \\ . \\ \delta'n \end{bmatrix}$$

where,
$^{(\delta i)}$ denotes the multiplicative inverse of the secret key values chosen form the group.

2. After generating the column vector and the secret key values of individual cloud users, the data owner stores them in its storage area.

2. User Join and Key Derivation Phase

Whenever a new user (i) is authorized to join the group to access the data stored in the public cloud for the first time, the data owner sends secret key value $^{(\delta i)}$ and its corresponding seed key value $^{(ri)}$ with respect to their identity in the group using a Secure Socket Layer (SSL). The user $^{(Usri)}$ and the data owner only know this secret key. All the cloud users of the group store the secret key$^{(\delta i)}$ and the seed key value $^{(ri)}$ in their respective storage area. The secret key and the random seed key value $^{(ri)}$ sent by data owner is used to find the group key $^{(\gamma)}$ in the cloud user side. After providing this information through SSL, the data owner computes the group key in the following ways and broadcast it to the users of the group

1. Initially, the data owner selects a random element $^{(\gamma)}$ as group key from the multiplicative group $^{(Zp)}$.
2. After selecting the group key $^{(\gamma)}$, the data owner multiplies the group key $^{(\gamma)}$ with the column vector (A) to produce a column vector (B) as shown in the below equation.

$$(\gamma)\times(A)=(B)$$

$$(\gamma) \times \begin{bmatrix} \delta'1 \\ \delta'2 \\ . \\ . \\ \delta'n \end{bmatrix} = \begin{bmatrix} \beta1 \\ \beta2 \\ . \\ . \\ \beta n \end{bmatrix}$$

3. After computing column vector (B), the data owner adds individual users seed key values (ri) to get the encrypted block is called as Access Control Vector $^{(ACVi)}$ as shown in the below equation.

$$\begin{bmatrix} \beta1,1 \\ \beta2,1 \\ . \\ . \\ \beta n,1 \end{bmatrix} + \begin{bmatrix} r1 \\ r2 \\ . \\ . \\ rn \end{bmatrix} = \begin{bmatrix} a1 \\ a2 \\ . \\ . \\ an \end{bmatrix} = ACV_i$$

4. The data owner broadcasts the $^{(ACVi)}$ value along with their index value to all the cloud users present in the group. The cloud users use the index value in order to extract a particular value from the $^{(ACVi)}$ value.

5. Upon receiving $^{(ACVi)}$ value from the data owner, each authorized cloud users $^{(Usri)}$of the current group can obtain the group key $^{(\gamma)}$ by doing simple subtraction and multiplication with respect to their index in the ^{ACV}, as shown in the below equation.

$$\gamma = (a_i - r_i) \times \delta_i mod p$$

The $^{(\gamma)}$ value obtained in this way must be equal to the value generated in step 1 of user join phase.

3. Key Updating Phase

When a user joins or leaves the group, the data owner runs the key updaing algorithm and outputs the newly updated group key as $^{(\gamma)}$. For example, if a user $^{(Usri)}$ wants to leave from the group, the data owner must update the group key $^{(\gamma)}$ in order to prevent the user $^{(Usri)}$ from viewing in the future data uploaded in the CSP. The data owner updates the group key $^{(\gamma)}$ as shown below:

1. The data owner eliminates n^{th}position as $\delta_{(n-1)}$, $\delta'_{(n-1)}$and multiplies a new group key $^{(\gamma')}$ with $^{(A')}$ to get a column vector $^{(B')}$.

$$(\gamma') \times (A') = (B')$$

$$(\gamma') \times \begin{bmatrix} \delta'1 \\ \delta'2 \\ \cdot \\ . \\ \delta'(n-1) \end{bmatrix} = \begin{bmatrix} \beta'1 \\ \beta'2 \\ \cdot \\ . \\ \beta'(n-1) \end{bmatrix}$$

2. After computing this value, it also updates the $^{ACVi'}$ value as shown below:

$$\begin{bmatrix} \beta'1 \\ \beta'2 \\ \cdot \\ . \\ \beta'(n-1) \end{bmatrix} + \begin{bmatrix} r1 \\ r2 \\ \cdot \\ . \\ r(n-1) \end{bmatrix} = \begin{bmatrix} a'1 \\ a'2 \\ \cdot \\ . \\ a'(n-1) \end{bmatrix} = ACV_i'$$

3. Upon receiving the updated $^{ACVi'}$ value from the data owner, each authorized users $^{(Usri)}$ of the current group can obtain the new group key by doing simple subtraction and multiplication with respect to the index $^{ACVi'}$.

$$\gamma = (a'_i - r_i) \times \delta_i mod p$$

III. Security Analysis

This paper analyses 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.

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 broadcasts *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 key and random seed key values. If the attacker is not an active adversary (i.e not a member of existing multicast 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 trial of $2^w$to learn about secret key. Similarly, the attacker has to use 2^w trial 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 drive δ_i and r_i values can be increased by choosing the 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.

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 number is updated in column vector B after the leaving operation is performed, any number of previous cloud 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 form 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 (γ)' which is broadcast at time (t+2) in a feasible amount of time.

3. Theorem: The proposed key management scheme (PVK-RS) is correct.
Proof: The correctness of PVK-RS can be easily proved as shown below:
$(\gamma) = (a_i - r_i) \times \delta_i$
$(\gamma) = ((\beta_i+r_i)-r_i) \times \delta_i$ (Since $a_i=(\beta_i+r_i)$)
$(\gamma) = ((\delta'_i \times \gamma+r_i)-r_i) \times \delta_i$ (Since $\beta_i= \delta'_i \times \gamma$))
$(\gamma) = (\delta'_i \times \gamma) \times \delta_i$
$(\gamma) = (\gamma)$ (Since $(\delta'_i \times \delta_i=1)$

IV. Performance Analysis

The proposed method has been implemented in JAVA(Intel Core i5 processor, 2 GB RAM, 500 GB Hard disk, Windows XP Operating System) for a group of 1500 users and have compared the group key computation 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 $Z^*_p$and the multiplicative inverse of the secret key values is placed in the matrix to compute the group key value. For generating large integers as secret key values, BigInteger class is used that supports various methods for handling large positive integers. The method multiply () supported by BigInteger class is used to multiply large integer values.

The method modInverse() is used in the implemented Java program to find the multiplicative inverse of a given element with respect to the size of the multiplicative group. Table I compares the data owners group key comutation time of the proposed approach with various key management schemes, namely Chinese Remainder Group Key (CRGK) [8], Fast-chinese Remainder Group Key (FRGK) [8], Key-tree Chinese Remainder Theorem (KCRT) [9] and Eluer'sTotient Function based group key management (ETF) [10]. Table I compares the measured computation time in milliseconds (ms) for updating the group key value in the data owner side various key sizes. For measuring the computation time, the program is executed for 10 trails and the average of the 10 trails is included in the table. It is evident from the values that the computation time of the proposed algorithm is found to be better in the data owner side for computing the group keys than the other algorithms.

TABLE. I Data Owner Group Key Computation Time For Various Key Distribution Approaches

Number of users	Prime Number Size (bits)	CRGK (ms)	FRGK (ms)	ETF (ms)	KCRT (ms)	PVK-RS(PROPSED) (ms)
5	1024	3022	2922	3311	2222	1511
10	1024	4638	4538	4119	3838	2319
50	1024	6692	6592	6986	5892	3346
100	1024	7009	6909	7382	6209	4843
200	1024	8789	8689	7900	7989	5618
400	1024	11987	11887	9701	11187	6027
500	1024	13902	13902	14801	13102	6205

Similarly, the analysis of the cloud users computation time is also done and the results are tabulated in Table II. From the result analysis, it is clear that the cloud users group key derivation time taken by our proposed algorithm is found better than the other algorithms. For measuring the computation time both in data owner and cloud user side, 1024 bits prime number has been used (*p*=120275242554787488859562207937345121287333878036820754336538999839551798509887978998691469008091316111533468170508320960221601463663463918124709871054152 33) to define a multiplicative group. The computation complexity for the data owner is the proposed algorithm is O(n) since the new group

key is multiplied with n user shares. The computation complexity of the cloud user in the proposed algorithm is O(3) because three operations (multiplication, subtraction and modulo division) are performed in the user side to derive the group key. The communication complexity of the proposed algorithm is 1 broadcast. The storage complexity of data owner is O(2n+1), where 2n represents n users secret and seed key value and 1 represents the group key value stored by the data owner. Similarly, the storage complexity of the cloud user is O(3).

TABLE. II Cloud Users Key Derivation Time For Various Key Distribution Approaches

Number of users	Key Size (bits)	CRGK (ms)	FRGK (ms)	ETF (ms)	KCRT (ms)	PVK-RS (PROPSED) (ms)
500	64	1.7	1.5	2.3	0.4	0.23
500	128	1.9	1.6	3.4	0.8	0.53
500	256	2.9	2.6	4.6	2.7	1.5
500	512	7.8	7.3	9.8	5.9	5.3
500	1024	9.4	8.6	11.9	7.4	6.4

The graphical result shown in Fig.1 is used to compare the group key computation time of the proposed method with the existing methods. From Fig.1, it is observed that when the group consists of 500 users, the group key updating time is found to be 6205ms in the proposed approach.

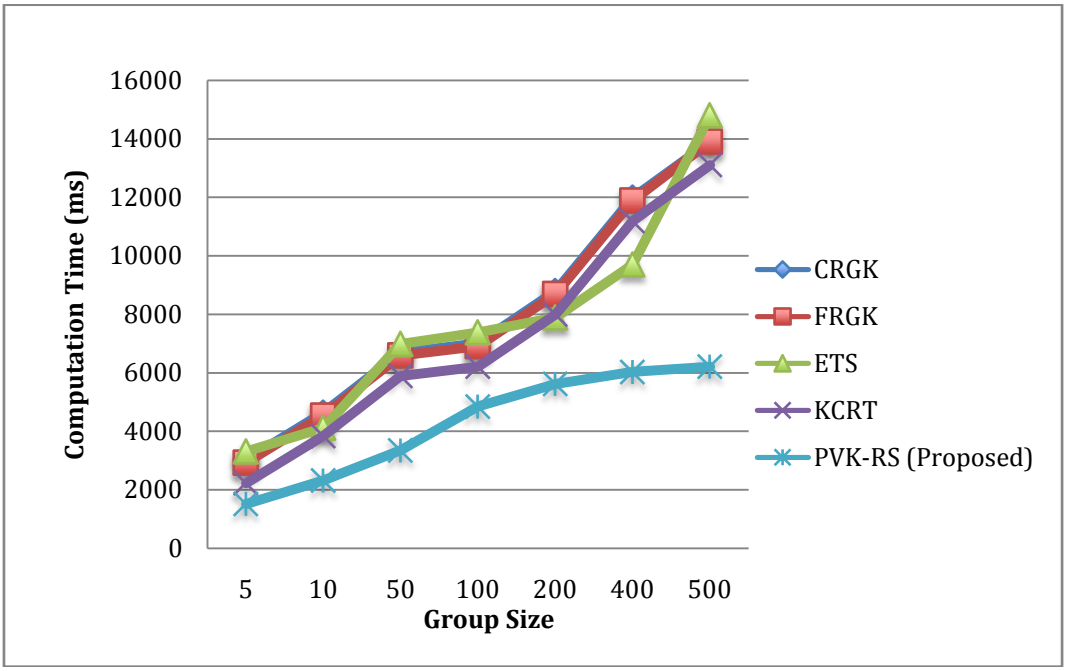

Fig. 1 Data owner group key computation time

Fig.2 shows the key recovery time to recover the group key sent from the data owner side of the proposed method. It is very clear to see from the Fig.2 when the key size is 1024 bits, the key recovery time of a cloud user is found to be 6.4ms in our proposed approach.

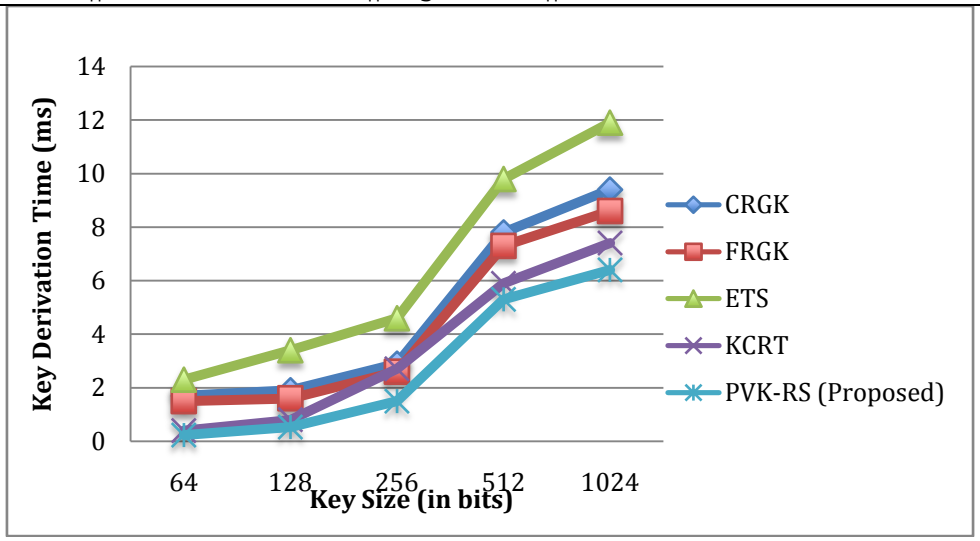

Fig. 2 Cloud users key derivation time

V. Conclusion

In this paper, a new computation and communication efficient key management technique has been proposed to tackle challenges in providing the selected documents form the CSP to the cloud users in a secure way. The proposed algorithm has two dimensional focuses namely minimal computation complexity and minimal communication complexity. With respect to the communication complexity, the communication complexity of the proposed scheme takes only one broadcast message to inform the ACV values to the cloud users to make them to find group key. A further extension to this work is to devise a technique to handle document overlapping in an efficient way.

References

[1]. Elisa Bertino, Piero. A, Bonatti and Elena Ferrari, "TRBAC: A temporal Role-Based Access Control Model", *ACM Transaction Information and System Security,* vol.4, no. 3, pp. 191-233, 2001.

[2]. James, B.D. Joshi, RafaeBhatti, Elisa Bertino and ArifGhafoor, "Access-Control Language for Multidomain Environments", *IEEE Transactions on Internet Computing*, vol.8, no.6, pp.40-50, 2004.

[3]. Barker. S, "Action-Status Access Control", *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies",* pp.195-204, 2007.

[4]. Poovendran. R and Baras. J.S, "An Information-Theoretic Approach for Design and Analysiis of Rooted-Tree-Based Multicast Key Management Schemes", *IEEE Transacgions on Information Theory*, vol.4, pp.2824-2834, 2001.

[5]. Kim. Y, Perrig. A and Tsudik. G, "Group Key Agreement Efficient in Communication", *IEEE Transactions on Computers*, vol. 53, number.7, pp. 905-921, 2004.

[6]. Drira. K, Seba. H and Kheddouci. H, "ECGK:Aneffiecient clustering scheme for group key management in MANETs", *Computer Communication*, Elsevier, vol.33, number.9, pp. 1094-1107, 2010.

[7]. Naranjo, Lopez-Ramosz J.A and Casado L.G., "A Suite of Algorithms for Key Distribution and Authentication in Centralized Secure Multicast Environments", *Journal of Computational and Applied Mathematics*, Elsevier, vol. 236, no.12, pp.3042-3051, 2012.

[8]. Vijayakumar. P, Bose. S, Kannan. A, "Chinese Remainder Theorem based Centralized Group Key Management for Secure Multicast Communication", *IET Information Security*, IET, vol.8, no.3, pp.179-187,2014.

[9]. Yoon-Su Jeong, Ki-Soo Kim, Yong-Tae Kim, Gil-Cheol Park and Sang-Ho Lee, "A Key Management Protocol for Securing Stability of an Intermediate Node in Wireless Sensor Networks", *IEEE 8th International Conference on Computer an d Information Technology,* pp.471-476, 2008.

[10]. Jung-Yoon Kim and Hyoung-Kee Choi, "Improvements on Sun et. al.'s Conditional Access System in Pay-TV Broadcasting Systems", *IEEE Transacitons on Multimedia*, vol.12, no.4, pp. 337-340,2010.

[11]. Mohamed Nabeel, Ning Shang and Elisa Bertino, "Privacy Preserving Policy-Based Content Sharing in Public Clouds", *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no.11, pp. 2602-2614, 2013.