International Journal of Security and Its Applications
Vol. 9, No. 4 (2015), pp. 217-228
http://dx.doi.org/10.14257/ijsia.2015.9.4.20

A User Access Control Scheme for Reducing Authentication Keys in Cloud Systems

Seungtae Hong, Hyeongil Kim, Taehoon Kim and Jaewoo Chang*

Dept. of Computer Engineering
Chonbuk National University
Jeonju, Republic of Korea
{dantehst, melipion, taehun3718, jwchang}@jbnu.ac.kr

Abstract

With the development of IT and internet services, the cloud computing system has attracted much interest in processing the big data efficiently. Because services using big data on the cloud computing environment consider a lot of users, an efficient user access control scheme is required. However, the existing schemes have a critical problem that the cost of the key management for the user access control is high. To solve the problem, we propose a user access control scheme for reducing authentication keys in cloud systems. The proposed scheme can reduce the number of keys based on a resource set tree by applying the minimum spanning tree. Finally, we show from the performance analysis that the proposed scheme outperforms the existing scheme in terms of key generation cost.

Keywords: *User Access Control; Optimization of authentication key; Cloud Computing*

1. Introduction

With the development of IT and internet services (e.g., social network services), the researches on big data have been actively studied [1]. Big data requires a large cluster of computing devices to process the huge volumes, high velocity, and varied formats of big data. Therefore, the cloud computing system has attracted much interest to manage the big data efficiently. In the cloud computing system, a service provider virtualizes computing resources and provides them to user as a service. A user borrows the computing resources as much as what he/she wants from the host and pays a rental fee based on the usage of the resources. A data owner can outsource the data to the service provider, so that he/she can reduce the costs of data management and system maintenance [2, 3]. For example, a real estate enterprise has to manage a large amount of data which include customer information, real estate information, market price information, and so on. For this, the real estate enterprise can outsource the real estate data to the service provider. By doing so, the enterprise can reduce the costs for purchasing hardware equipment, managing enterprise data, and maintaining the system. For protecting his/her enterprise data, the real estate enterprise wants to allow only authorized users to have accesses to the outsourced database [4].

Thus we should consider the following requirements when outsourcing the database in the cloud computing environment [5, 6]. First, the contents of the database should be hidden from the service provider and malicious attackers. For this, the encryption of the original data is required for database outsourcing [7-10]. Secondly, the service provider should verify whether a query is issued by an authenticated user or not. For this, a user authentication method is required for providing a decryption key [11-15].

* Corresponding Author

ISSN: 1738-9976 IJSIA

Based on the requirements, the research on the user access control for the cloud computing environment has been actively performed. However, the existing user access control schemes have two main problems due to the use of hierarchical tree structure. First, the number of keys for each user drastically increases as the depth of a tree is increased. Secondly, the key update cost is very high when a user's permission to access data is changed. This is because it is necessary to modify all the keys corresponding to the changed permission in the hierarchy.

In order to efficiently perform the user access control for the cloud computing environment, we propose a user access control scheme for reducing authentication keys in cloud systems. The proposed scheme can reduce the number of keys for each user by generating a resource set tree (RST). Thus, it is possible to perform the user access control efficiently for a large number of users in the cloud computing environment. Our contributions can be summarized as follows:

- We present a framework to provide the user access control for the data outsourced to a cloud system.
- Our scheme can reduce the key generation cost by using RST-based key generation algorithm.
- We also present an extensive experimental analysis of our scheme by comparing it with the existing work. It is shown that our scheme provide better performances on key generation cost.

The rest of the paper is organized as follows. In section 2, we introduce the existing methods for user access control. In section 3, we describe the proposed user access control scheme. In section 4, we do performance analysis of our scheme. Finally, we conclude this paper with future work in section 5.

2. Related Work

The user access control schemes which do not consider the outsourced database environment are as follows. First, C. K. Wong et al. [11] proposed a user access control scheme which can efficiently manage authentication keys. The scheme generates several user groups with the same number of users. Each user in a group holds both a group key and his/her private key. When a new user is assigned to a user group, the scheme updates the group key by using a logical key graph. However, when the number of users is large, the scheme has two problems. First, the key generation cost is high because all users should hold their own private keys. Second, the key update cost is high because the scheme sends the updated keys to all users in the corresponding group. Secondly, to solve these problems, Q. Zhang et al. [12] proposed a Hierarchical Access Control (HAC) scheme based on a directed acyclic graph, which represents the relations between user groups and data groups. By linking two groups of the graph, the HAC generates authentication keys for the user access control.

On the other hand, user access control schemes on the outsourced database environment have recently been proposed. First, Miklau and Suciu [13] present a framework for executing user access control on published XML documents. The framework applies different cryptographic keys over different parts of the documents. Secondly, Damiani et al. [14] proposed a user access control scheme by using selective encryption and hierarchical key assignment concepts. To efficiently perform the user access control, they also proposed an algorithm that minimizes the number of secret keys in users' key rings. Finally, C. Blundo et al. [15] proposed a heuristic scheme which greatly reduces the number of authentication keys for the user access control. To reduce the cost for the user authentication, the heuristic scheme performs a key optimization by using a Minimum Spanning Tree (MST). For this, it obtains the common users of two

nodes and adds a new parent node to manage the common users. As a result, the scheme can form the user tree with the minimum weights. However, the heuristic scheme cannot update user keys when the user's access permission is changed. This is because they only focus on the minimizing the number of the user authentication keys.

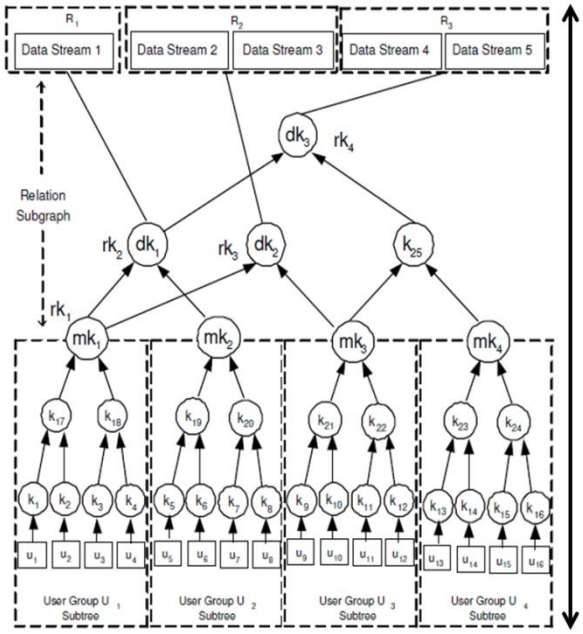


Figure 1. Overall Architecture of the HAC [12]

3. A User Access Control Scheme for Reducing Authentication Keys

3.1. Motivation

The HAC scheme proposed by Q. Zhang et al. uses a hierarchical tree to provide users with their authentication keys. However, the number of the authentication keys is drastically increased as the depth of the tree increases. In addition, when a user's access permission is changed, the key update cost is very high because HAC should modify all the keys corresponding to the changed permission in the hierarchy. Fig. 2 shows two user groups UserGroup1 = {u_1, u_2, u_3, u_4} and UserGroup2 = {u_5, u_6, u_7, u_8}. Here, each user holds 3 kinds of keys; i) his/her own authentication key, ii) a shared key for his/her neighboring nodes, iii) the group key(mk_i). For example, assume that u_8 holds 3 keys, i.e., k_8, k_{12}, and mk_2, and u8 becomes a neighbor of u_1 by moving from UserGroup2 to UserGroup1. To update the access permission of u8 in UserGroup1, i) HAC has to generate a new shared key k_{13} by using k_1 and k_8. ii) The algorithm creates a new node for k_{13}, which becomes a new parent node of both the node of k8 and that of k_1. iii) It modifies k_9 using k_2 and k_{13}, and changes the relationship between the node for k_9 and the new node. iv) HAC updates the group key mk_1 because k_9 has been modified. In this case, because all nodes in UserGroup1 share the group key mk_1, they should receive the updated mk_1. In the UserGroup2, HAC also modifies the shared key k_{12} and the group key mk_2. Then, it sends the modified keys to their corresponding nodes. Thus, the update cost for changing the access permission of u_8 is high.

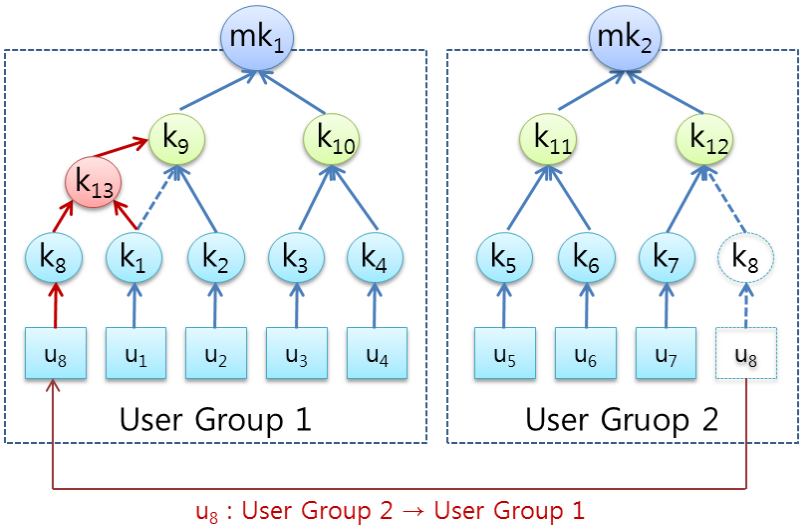


Figure 2. Problem of HAC Scheme

To solve the problems of the existing work, we propose a user access control scheme for reducing authentication keys in cloud systems. To efficiently manage the authentication keys, we design two indices: user access matrix (ACM) and resource set tree (RST). ACM and RST are defined as definition 1 and definition 2, respectively.

Definition 1 (user access matrix) ACM is a matrix which represents users' access permission. In ACM, each row and column represents a user group and a data resource, respectively. When a user has a permission to a data resource, the corresponding cell of the matrix is set as 1; otherwise 0.

Definition 2 (resource set tree) RST is a tree consisting of resource set, where a resource set is a group of cells which have the same access permission in the ACM by performing the column based grouping.

To minimize the number of authentication keys, our scheme first generates resource sets based on the users' access permissions. Secondly, our scheme merges the resource sets by using Minimum Spanning Tree (MST), so it generates a minimum-weighted RST. Finally, it assigns an authentication key to each node. Because the tree can be minimized based on MST, our scheme can reduce the key generation cost.

Fig. 3 shows the system architecture of our scheme, which consists of a data owner (DO), a service provider (SP), and authenticated users (AU). The DO contains a key optimization module and a key management server to efficiently manage the authentication keys. When a user requests an authentication key, the DO identifies the user. If the user is an authenticated user, the DO generates an authentication key and sends the key to the AU. Meanwhile, the SP has a responsibility to provide services to the AU based on the encrypted database sent from the DO. The AU can send a service request to SP. When the AU receives an encrypted result from SP, the AU can decrypt the result by using the key given by the DO.

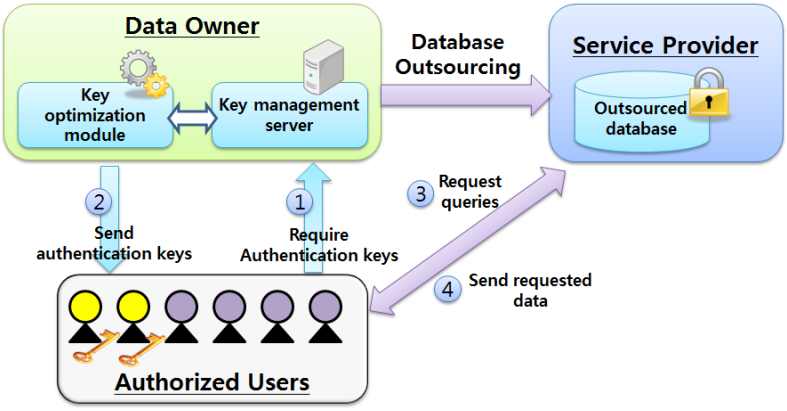


Figure 3. Overall System Arhitecure

The proposed scheme consists of two phases; resource set tree construction phase and resource set tree optimization phase.

3.2. Resource Set Tree Construction

First, a data owner generates a user access matrix which represents the users' access permissions to data resources. In the matrix, each row indicates a user class, rather than a user. This is because IT services are commonly provided based on the user's class. For example, a user class named 'Gold' can enjoy all kinds of services whereas the 'Basic' user class can only enjoy politic and weather services. Fig. 4 shows an example of user access matrix where there are four user classes (i.e., A, B, C, and D), and five data resources (i.e., r1, r2, r3, r4, and r5). When a user class has a permission to a data resource, the corresponding cell of the matrix is set to 1; otherwise 0. Secondly, our scheme generates resource sets by grouping cells with the same access permissions in the user access matrix. For this, our scheme finds cells with value 1 in the matrix and then performs column grouping. By generating authentication keys based on the resource sets, we can greatly reduce the number of required authentication keys. Fig. 4 shows an example of the resource set generation. Because r_1 is set to 1 for the user classes A and B, we can generate a resource set R1 for A and B. Because the user classes A and C have permissions to access r_2 and r_3, we can generate a resource set R_2.

	r1	r2	r3	r4	r5
A	1	1	1	1	1
B	1	0	0	1	1
C	1	1	1	1	1
D	0	0	0	1	1

R_1 R_2 R_3

Figure 4. Example of Resource Set Generation

Finally, our scheme generates a resource set tree using the user access matrix. Each leaf node of the resource set tree contains a list of user classes that has the access permission to its resource set. An edge being incident to a leaf node has an edge weight which means the number of user classes that can access the leaf node. The edge weight and a total authentication cost are defined as definition 3 and definition 4, respectively.

Definition 3 (edge weight) An edge of the tree has a weight that is calculated by subtracting the number of users in the child node from that of the parent node. When arg denotes the accessible resource group, the edge weight is calculated by Equation (1).

$$w(v_i, v_j) = |v_j.arg \backslash v_i.arg| \quad \ldots (1)$$

Definition 4 (authentication cost) The authentication cost is defined as the sum of all weights in the RST, as shown in Equation (2).

$$weight(T) = \sum_{(v_i,v_j)\in E} w(v_i, v_j) \quad \ldots (2)$$

where E means the set of all edges in the RST.

Fig. 5 shows a resource set tree generated from the user access matrix in Fig. 4. In the figure, v_1[AB] indicates that user classes A and B have access permissions to the resource set R_1.

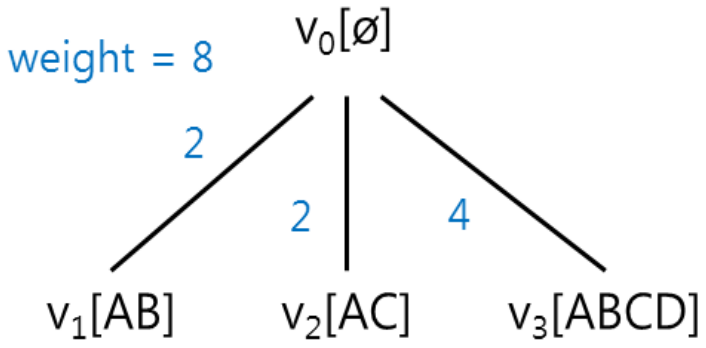


Figure 5. Example of User GroupTree

A RST construction algorithm to generate a resource set tree from the user access matrix is as follows. First, the algorithm generates a user access matrix using the users' access permissions to data resources (line 1~2). Secondly, it generates resource sets by grouping cells which have the same access permissions in the user access matrix (line 3~4). For each generated resource set, the algorithm adds a node to the resource set tree (line 5~6). Thirdly, it calculates the weights of edges of the resource set tree and computes the total user authentication cost as the sum of all weights in the tree (line 7~9). Finally, the algorithm returns the generated resource set tree (line 10).

Algorithm 1: RST construction()
Input : ACM : User access matrix, UC : User class
Output : RST : Resource Set Tree
1: ACL = init_ACL (ACM)
2: RST = initTree()
3: for(each column c_i in ACM)
4: for(each class g_{j-1} in UC)

```
5:      if(ACL [g_i]==ACL [g_{i+1}])
6:          RST = addNode(c_i)
7: for(each e_i in RST)
8:    cost_i = weight(e_i)
9:    totalcost += cost_i
10: return RST(totalcost)
```

3.2. Resource Set Tree Optimization

First, a data owner generates a RST which minimizes the weights of edges in the tree. For this, our scheme uses the sibling based algorithm proposed in [15]. The algorithm of RST optimization is as follows. i) The algorithm finds all pairs of sibling nodes in RST (line 1~2). ii) It calculates the weight of each pair and inserts the pair into a candidate list (line 3~4). iii) The algorithm sorts the candidate list in a descending order based on the weights (line 5). iv) The algorithm selects the pair with the largest weight in the list. Then it creates a new node which contains the shared user classes from the selected pair (line 7). The new node becomes a parent node of the two sibling nodes in the pair. The node also becomes the child node of the previous parent node (line 8~10). v) The algorithm calculates the weights of newly created edges and inserts them into the candidate list (line 11). vi) If two nodes selected in the step iv do not have common user classes, the algorithm deletes the pair from the candidate list (line 12~13). vii) The algorithm repeats the steps from iv to vi until the candidate list has no members. viii) The algorithm returns the minimum-weighted RST (line 14). Fig. 6 shows a minimum-weighted RST which is derived from the resource set tree in Fig. 4.

Algorithm 2: RST optimization()

```
Input : RST : Resource set tree
Output : MinRST : Minimum-weighted Resource set tree
 1: for(each n_i in RST)
 2:    sibling_pair_i = FactorizeSibling()
 3: for(each sibling_pair_i)
 4:    candidate = weight(sibling_pair_i)
 5: sort(candidate)
 6: while(candidate is not null)
 7:   max_pair = getFirst(candidate)
 8:    sharedGroup = check(max_pair)
 9:    if(shared is not null)
10:       MinRST = updateTree(sharedGroup)
11:       totalcost = updatecost(RST, candidate)
12:   else
13:       deleteFromCandidate(max_pair)
14: return MinRST(totalcost)
```

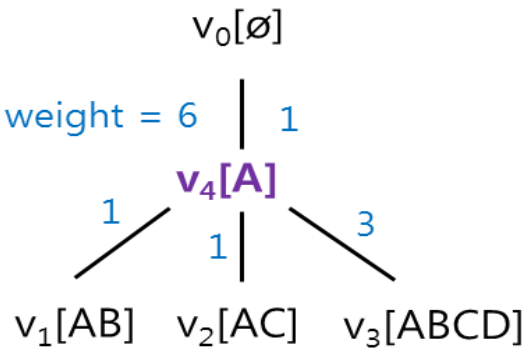


Figure 6. Example of Minimum-weighted Resource Set Tree

Secondly, a data owner assigns authentication keys to user classes based on our minimum-weighted RST. By using a logical graph concept, each resource set is connected to a user class which has the access permission to the resource set. Fig. 7 shows an example of authentication key assignment using the minimum-weighted RST. In Fig. 7, resource sets R1, R2, and R3 are connected to their unique authentication keys, i.e., RK1, RK2, and RK3. Because the node v_4 is newly created in the minimum-weighted RST, our key assignment scheme generates a new authentication key RK4 for v_4. Then, RK1, RK2, and RK3 are connected to RK4. Finally, all the authentication keys RK1, RK2, RK3, and RK4 are assigned to their corresponding user classes. For example, user classes B, C, and D have the access permission to R3, so RK3 is assigned to B, C, and D. As a result, the authentication keys of user classes A, B, C, and D are {RK4}, {RK1, RK3}, {RK2, RK3}, and {RK3}, respectively.

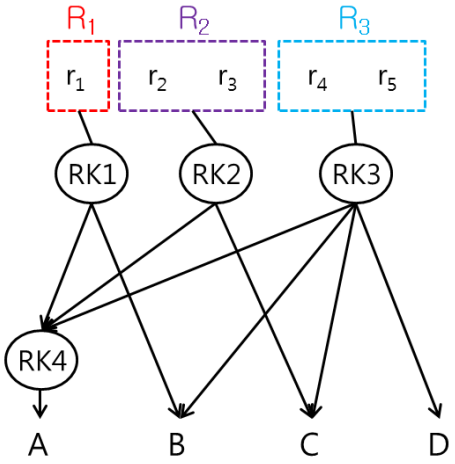


Figure 7. Example of Authentication Key Assignment

4. Performance Analysis

In this section, we present the performance analysis of our user access control scheme. We compare our scheme with the Heuristic approach [15] proposed by C. Blundo et al. We do our performance analysis under the environment setting as shown in Table I.

Table 1. Experimental Environment

CPU	Intel i3-2100 CPU 3.10GHz
Memory	2GB
OS	Windows 7 (64 bit)
Compiler	Microsoft Visual Studio 2010

We evaluate the performance on the key generation cost varying the number of both user groups and resources. We measure the key generation cost by counting the number of generated keys. In addition, we generate a user access matrix under the assumption that each user class of the matrix shares at least one resource with other user classes. For our performance analysis, we use both RST of our scheme and a user tree of the existing work as completely height-balanced trees with 4 child nodes. The parameter settings are illustrated in Table II.

Table 2. Simulation Parameter Settings

parameter	values	default
# of user classes	5, 10, 20	5
# of resources	20, 40, 60	80

Fig. 8 shows the key generation cost, with varying the number of user classes from 5 to 20. When the number of user classes is 20, our scheme requires about 95 keys while the existing work requires about 320 keys. Thus our scheme shows about 3~4 times better performance than the existing work. This is because our scheme assigns authentication keys by using the minimum-weighted RST. As a result, we can eliminate the redundant authentication keys in user classes.

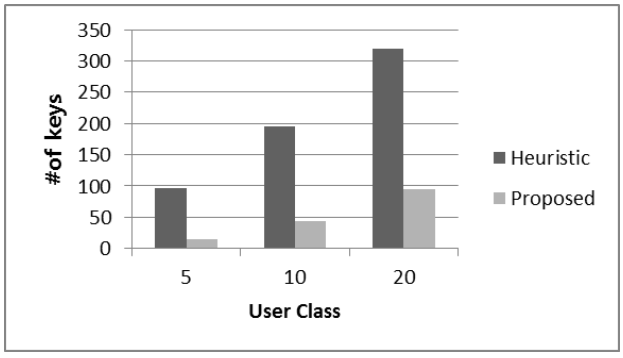


Figure 8. Key Generation Cost Varying the Number of User Classes

Fig. 9 shows the key generation cost, with varying the number of resources from 20 to 80. When the number of resources is 80, the number of authentication keys of our scheme and the existing work are 14 and 96, respectively. Thus our scheme shows about 7 times better performance than the existing work. In the Heuristic approach, the number of user classes with the same access permissions to specific resources is increased as the number of resources increases. As a result, the key generation cost of the existing work is high because unnecessary authentication keys are increased. Meanwhile, our scheme shows the better performance than the existing work in all cases because it can minimizes the number of redundant authentication keys by using the minimum-weighted RST.

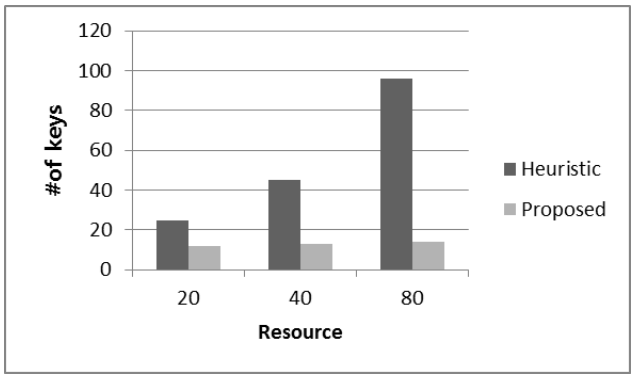


Figure 9. Key Generation Cost Varying the Number of Resources

5. Conclusion

Recently, outsourced databases have attracted much interests with the development of the cloud computing. Because services using big data on the cloud computing environment consider a lot of users, an efficient user access control scheme is required. For this, we proposed a user access control scheme for reducing authentication keys in cloud systems. Our scheme can reduce the key generation cost by generating the minimum-weighted RST. Through our performance analysis, it was shown that our scheme outperforms the existing scheme, in terms of key generation costs under the different number of user classes and the resources. As a future work, we have a plan to expand our work to reduce the key update cost.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2014065816).

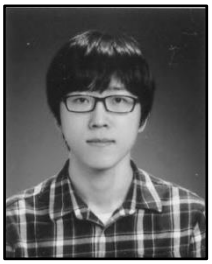
References

[1] J. H. Lim, "System proposal and CRS model design applying personal Information protection for BIG DATA analysis", International Conference on Big Data and Smart Computing, **(2014)**.
[2] M. L. Yiu, G. Ghinita, C. S. Jensen and P. Kalnis, "Enabling search services on outsourced private spatial data", VLDB Journal, vol. 19, no. 3, **(2010)**, pp. 363-384.
[3] H. I. Kim, "Hilbert-curve based cryptographic transformation scheme for protecting data privacy on outsourced private spatial data", International Conference on Big Data and Smart Computing, **(2014)**.
[4] H. Hacigümüş, "Executing SQL over encrypted data in the database-service-provider model", Proceedings of the ACM SIGMOD international conference on Management of data. ACM, **(2002)**.
[5] J. Brodkin, "Gartner: Seven cloud-computing security risks", Network World, **(2008)**.
[6] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing V2.1", **(2009)**.
[7] A. Khoshgozaran and C. Shahabi, "Private buddy search: enabling private spatial queries in social networks", Proceedings of the IEEE International Conference on Computational Science and Engineering, **(2009)**.
[8] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg and D. Boneh, "Location privacy via private proximity testing", NDSS, **(2011)**.
[9] M. Yie, I. Assent, C. Jensen and P. Kalnis, "Outsourced similarity search on metric data assets", IEEE Transactions on knowledge and data engineering, vol. 24, no. 2, **(2012)**, pp. 338-352.
[10] R. Agrawal, J. Kiernan, R. Srikant and Y. Xu, "Order-PreservingEncryption for Numeric Data", Proceedings of the ACM SIGMOD international conference on Management of data, **(2004)**.
[11] C. K. Wong, M. Gouda and S. S. Lam, "Secure group communications using key graphs", IEEE/ACM Transactions on Networking, vol. 28, no. 4, **(2000)**, pp. 16-30.

[12] Q. Zhang, "A Key Management Scheme for Hierarchical Access Control in Group Communication", International Journal of Network Security, vol. 7, no. 3, **(2008)**, pp. 323–334.
[13] G. Miklau and D. Suciu, "Controlling access to published data usingcryptography", Proceedings of the 29th VLDB Conference, **(2003)**.
[14] E. Damiani, S. D. C. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi and P. Samarati, "Selective data encryption inoutsourced dynamic environments", Proceedings of VODCA, **(2006)**.
[15] C. Blundoa, "Managing key hierarchies for access control enforcement: Heuristic approaches", Computers & Security, vol. 29, no. 5, **(2010)**, pp. 533–547.

Authors

Seungtae Hong, he received the B.S and M.S degrees in computer engineering from Chonbuk National University, Korea, in 2008 and 2010 respectively. He is currently in a Ph.D. course in Chonbuk National University. His research interests include security and privacy of database, sensor network and cloud computing.



Hyeong-Il Kim, he received the B.S and M.S degrees in computer engineering from Chonbuk National University, Korea, in 2009 and 2011 respectively. He is currently in a Ph.D. course in Chonbuk National University. His research interests include security and privacy of database, query processing algorithm, and cloud computing.

Tae-hoon Kim, he received the B.S degrees in computer engineering from Chonbuk National University, Korea, in 2013. He is currently in a Master course in Chonbuk National University. His research interests include cloud computing and Hadoop.

Jae-Woo Chang, he received the B.S. degree in computer engineering from Seoul National University, Korea, in 1984, the M.S. and Ph.D. degrees in computer engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, in 1986 and 1991, respectively. During 1996~1997, he stayed in University of Minnesota as a visiting scholar. He joined the faculty of the Department of Computer Engineering at Chonbuk National University in 1991. His research interests include spatial network database, context awareness and storage system.