International Journal of Security and Its Applications
Vol. 10, No. 8 (2016), pp.121-130
http://dx.doi.org/10.14257/ijsia.2016.10.8.10

A Group Key based Security Model for Big Data System

Heeyoul Kim[1]

[1]*Department of computer science, Kyonggi Univ., Republic of Korea*
heeyoul.kim@kgu.ac.kr

Abstract

Recently Big data is in the spotlight and several NoSQL systems have been appeared in order to process large scale data. Among them Cassandra provides high scalability and availability with internal cluster structure. However, it does not provide enough security functionalities, especially transferred messages between internal cluster nodes are easily exposed by outside adversaries. In this paper a group key based security model for Cassandra is proposed. Cluster membership authentication and message confidentiality is provided by using of the group key, and the key is efficiently updated by decentralized approach where the cluster is divided into several subgroups considering Cassandra structure. Our model contributes preventing Cassandra cluster from illegal outside access attempts.

Keywords*: NoSQL systems, Cassandra system, security model, group key management*

1. Introduction

There have been presented many technologies for future computing, and recent keywords in the spotlight are cloud computing and Big data. Cloud computing is a phrase used to describe a variety of computing concepts that involve a large number of computers connected through a real-time communication network such as the internet [11]. Cloud computing enables service providers to have virtually everything available for rent, which can significantly reduce both establishment cost and management cost [5]. The major models of cloud computing service are known as SaaS, PaaS and IaaS. Despite these benefits, however, cloud computing has many issues to be solved and one of them is security [15-17], which makes many companies to hesitate about the adoption of the cloud computing.

Big Data, which has become very popular phrase just like cloud computing, refers to incredibly large datasets that can reveal hidden patterns and relationships. Traditional procedures have difficulty in processing this large datasets, so distributed systems with multiple servers are essential. Cloud computing is a good clue to bring Big Data to the masses without having to add permanent resources [1]. There have appeared various NoSQL systems such as Cassandra, BigTable, and MongoDB in order to process Big Data where SQL-style querying is not the crucial objective [3]. However, the research on security of NoSQL systems has not been covered in depth enough, thus a lot of concerns have been encountered [10].

Cassandra, one of famous NoSQL systems, is designed to handle Big Data across multiple nodes [8]. The nodes are dispersed in several data centers, and they are connected to the network. Though physically they are dispersed in multiple machines, they are logically seen as a single storage and provide a single instance to the end users. Cluster is the most important internal structure of Cassandra. All nodes constitute a ring cluster and data is distributed among all nodes in the cluster. Each node exchanges cluster metadata across the cluster every second. For scalability a new node can be added into the cluster, and at that time one of seed nodes is used as a contact point and it inform the

ISSN: 1738-9976 IJSIA

topology of ring cluster to the newcomer node. Multiple seed nodes can be designated, for example, per a data center [4].

Unfortunately, Cassandra does not support enough security functionalities currently. Though Cassandra provides client-to-node encryption based on SSL and authentication of users based on password, it lacks security support for inter cluster communication and cluster node authentication. Thus the messages between cluster nodes are easily exposed by outside attackers. One possible suggestion is using a private certificate authority [10]. After certificate-based authentication, each node can communicate securely with encryption. However, this suggestion requires relatively large overhead, and node authentication occurs very frequently. Especially Cassandra uses gossip protocol to share and state information about cluster for every second [12]. Therefore, it is inefficient to authenticate each node every time.

Our main idea is that just confirming legitimate cluster membership is sufficient instead of peer authentication, and it can be performed efficiently with current group key. In this paper a decentralized group key management model is presented for this purpose. This model is designed in consideration of the characteristic of Cassandra system, and it blends both centralized approach and distributed approach. With this model the cluster membership of a node is easily confirmed by other nodes, which significantly enhances performance of Cassandra system.

This paper is organized as follows. Section 2 explains the architecture and operations of the proposed model. Section 3 analyzes both security and performance after applying our model to Cassandra. Finally, Section 4 presents our conclusion.

2. Proposed Model

The purpose of group key management is to share a secure group key among legitimate group members [2]. In this section we present a group key management model suitable for Cassandra to improve security. The notations in *Table* 1 are used to explain the model.

Table 1. Notations and Definitions Used

Notation	Definition
SD_x	seed node
n_x	member node
S_x	subgroup
(a,b)	b-th node at level a in a tree
$K_{(a,b)}$	$n_{(a,b)}$'s private key
$BK_{(a,b)}$	$n_{(a,b)}$'s blinded key
p	large prime number
α	primitive root mod p

In the aspect of Cassandra, there are some issues in applying group key properly. First, security functionality should not degrade performance of Cassandra seriously. Second, in Cassandra addition and deletion of cluster nodes occur very often as it is a large-scaled distributed system. Finally, Cassandra should preserve fault tolerance even after applied.

A centralized group key management approach is not suitable for Cassandra due to its distributed nature. Otherwise, a distributed group key management approach induces

critical performance degradation. Therefore, we adopt a decentralized approach [6] which blends both centralized and distributed approaches. *Figure* 1 explains our decentralized approach. It is assumed that one seed node is assigned in a data center. Each seed node manages a subgroup where all member nodes in the same data center are included. An LKH scheme [14], one of centralized approach, is applied to the subgroup and the seed node plays the role of key distribution center. All seed nodes themselves participate in constructing and managing group key with TGDH scheme [7], one of distributed approach [13].

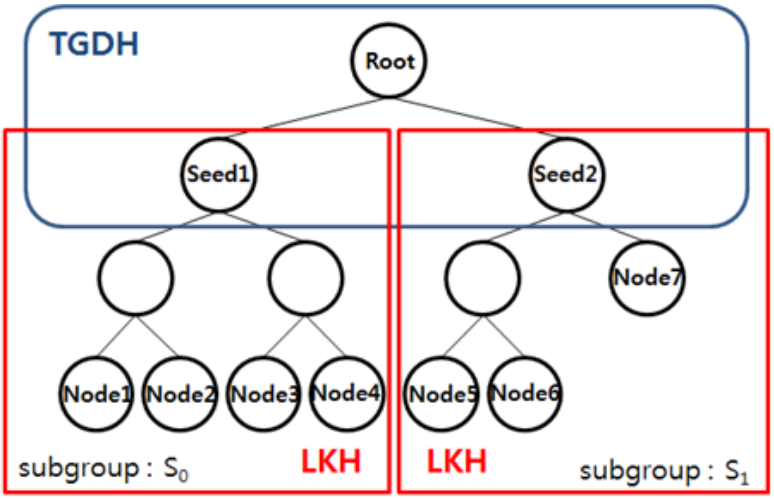


Figure 1. Overview of Proposed Model

2.1. Initialization

In the initial phase, all seed nodes of the cluster together construct a key tree and generate a group key following TGDH scheme. *Figure* 2 shows the way four seed nodes generate a group key. The key of each node in the tree can be computed with a private key of one child node and a blinded key of the other child node as following equation:

$$\begin{aligned} K_{(a,b)} &= (BK_{(a+1,2b+1)})^{K_{(a+1,2b)}} \ \ mod \ \ p \\ &= (BK_{(a+1,2b)})^{K_{(a+1,2b+1)}} \ \ mod \ \ p \\ &= \alpha^{K_{(a+1,2b)}K_{(a+1,2b+1)}} \ \ mod \ \ p. \end{aligned} \tag{1}$$

After computing the group key each seed node has to deliver it to the subgroup member nodes. The seed node constructs a key tree for its subgroup, assigns keys including subgroup key, and distributes keys following LKH scheme. Then the group key above is securely delivered to member nodes by encrypting it with the subgroup key.

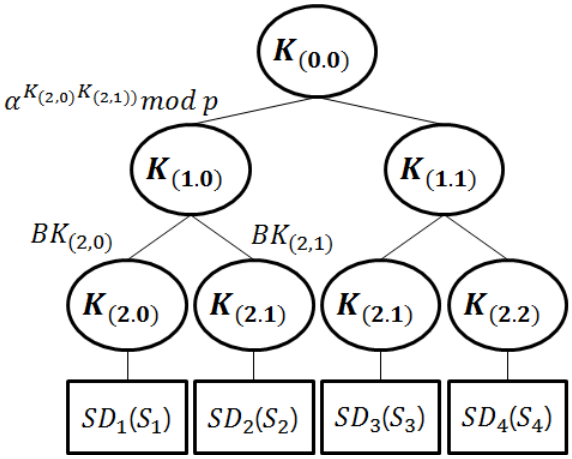


Figure 2. Generation of a Group Key by Seed Nodes

2.2. Join Operation

When a new node joins Cassandra cluster for scalability, its logical location in the key tree is determined by its physical location. The nearest seed node (usually in the same data center) accepts join request after proper authentication, and the key tree is expanded for the new node. Then, current group key has to be changed for backward secrecy, *i.e.*, the new node cannot discover any previous group keys. *Figure* 3 shows expanded key tree after join, and corresponding key update protocol is as follows.

1. $SD_2 \rightarrow \{n_5, n_6, n_7\} : \{K'_{(1,1)}\}_{K_{(1,1)}}$
2. $SD_2 \rightarrow \{n_7\} : \{K'_{(2,3)}\}_{K_{(3,6)}}$
3. $SD_2 \rightarrow \{n_8\} : \{K'_{(1,1)}, K'_{(2,3)}\}_{K_{(3,7)}}$
4. $SD_2 : K'_{(0,0)} = \left(BK_{(1,0)}\right)^{K'_{(1,1)}} \ mod\ p$
5. $SD_2 \rightarrow \{SD_1\} : \{BK'_{(1,1)}\}$
6. $SD_1 : K'_{(0,0)} = \left(BK'_{(1,1)}\right)^{K_{(1,0)}} \ mod\ p$
7. $SD_1 \rightarrow \{n_1 \sim n_4\} : \{K'_{(0,0)}\}_{K_{(1,0)}}$
8. $SD_2 \rightarrow \{n_5 \sim n_8\} : \{K'_{(0,0)}\}_{K'_{(1,1)}}$

In 1~3 steps, as n_8 was freshly added, the seed node SD_2 delivers changed keys $K'_{(1,1)}$ and $K'_{(2,3)}$ to its member nodes securely by using existing keys. In 4~6 steps, as SD_2's key is changed, both SD_1 and SD_2 compute a new group key $K'_{(0,0)}$. In 7~8 steps, the new group key is delivered to all member nodes by each seed node.

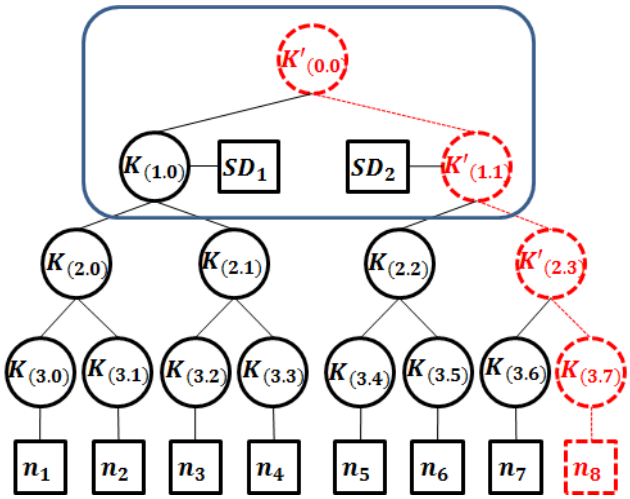


Figure 3. Key Tree after n_8 has Joined

2.3. Leave Operation

When an existing node leaves Cassandra due to scaling down, system error, malicious attack, and so on, current group key has to be changed for forward secrecy, *i.e.*, the leaving node cannot discover any future group keys. *Figure* 4 shows the key tree after n_8 leave, and corresponding key update protocol is as follows.

1. $SD_2 \rightarrow \{n_7\} : \{K'_{(1,1)}, K'_{(2,3)}\}_{K_{(3,6)}}$
2. $SD_2 \rightarrow \{n_5, n_6\} : \{K'_{(1,1)}\}_{K_{(2,2)}}$
3. $SD_2 : K'_{(0,0)} = \left(BK_{(1,0)}\right)^{K'_{(1,1)}} \ mod \ p$
4. $SD_2 \rightarrow \{SD_1\} : \{BK'_{(1,1)}\}$
5. $SD_1 : K'_{(0,0)} = \left(BK'_{(1,1)}\right)^{K_{(1,0)}} \ mod \ p$
6. $SD_1 \rightarrow \{n_1 \sim n_4\} : \{K'_{(0,0)}\}_{K_{(1,0)}}$
7. $SD_1 \rightarrow \{n_5 \sim n_7\} : \{K'_{(0,0)}\}_{K_{(1,1)}}$

In 1~2 steps, n_8's seed node SD_2 updates keys $K'_{(1,1)}$ and $K'_{(2,3)}$, then it delivers them to its member nodes securely except for n_8. In 3~5 steps, as SD_2's key is changed, both SD_1 and SD_2 compute a new group key $K'_{(0,0)}$. The new group key is delivered to all existing member nodes by each seed node in 6~7 steps.

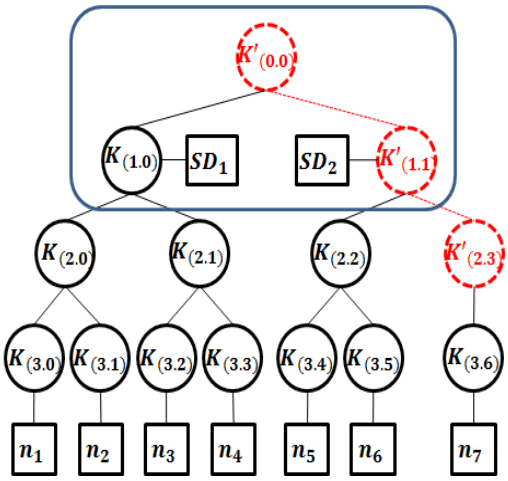


Figure 4. Key Tree after n_8 has Leaved

3. Analysis of Proposed Model

3.1. Security

The proposed model satisfies forward secrecy, backward secrecy, and key independence. Particularly, since the proposed model is based on TGDH scheme and LKH scheme, it provides an equivalent security level to above schemes.

For backward secrecy, when a new node joins, the seed node delivers only the changed keys in the path from subgroup root to the node, *i.e.*, $K'_{(1,1)}$ and $K'_{(2,3)}$ in *Figure* 3. Then no other keys except for the new group key $K'_{(0,0)}$ are delivered to the node. Therefore, the new node cannot discover any information about previous group keys or other keys not allowed.

For forward secrecy, when a node leaves, the keys known to the node - $K'_{(1,1)}$ and $K'_{(2,3)}$ in *Figure* 4 - are changed by the seed node. The group key is also changed, and it is securely delivered to only existing nodes by using the freshly changed keys. Therefore, the leaving node cannot discover any information about current and future group keys even if it knows the information about the former keys.

In addition, the changed keys are randomly generated regardless of the previous keys to guarantee key independence. And each node knows only current group key and the keys in the path from seed node to itself. Therefore, each node cannot analogize other keys which aren't allowed through the known keys.

3.2. Performance

There is usually a tradeoff between security and performance [9]. Thus adding security functionality to Cassandra inevitably induces performance degradation. The proposed model made an effort to find efficient way while enhancing security. During communication, each node can confirm the cluster membership of the other node with current group key, which is much more efficient than authenticating each other individually.

Our model is a decentralized model that blends centralized approach and distributed approach to obtain advantages of both sides together. Our model has better performance than distributed approach because seed nodes reduce redundant works and efficiently control their subgroups. Meanwhile our model is more fault tolerant than centralized approach because even the shutdown of one seed node doesn't cause the breaking of

entire system. *Table* 2 compares performance of our model with LKH and TGDH assuming that the heights of the key tree are the same. The number of transferred keys, which is one of major factors in performance, is $O(\log N)$ in our model, and it is similar to the existing group key management models.

Table 2. Comparison of the Proposed Model

		LKH	TGDH	Proposal
# of Stored Keys	**Seed or key server**	$2N-1$	-	$2(N/s+\log s)$
	Member	$\log N+1$	$2\log N+1$	$\log(N/s)+1$
# of Transferred Keys	**Join**	$\log N+1$	$\log N+1$	$\log N+2$
	Leave	$\log N$	$\log N$	$\log N$
Performance		High	Low	High
Fault-Tolerance		Low	High	High
Type		Centralized	Distributed	Decentralized

N : Number of cluster nodes, s : Number of subgroups

Generally, in the group key management, it is efficient to keep the balance of the tree. However, in Cassandra the network cost varies according to whether the nodes are in the same data center or not. Thus in the presented model a node's subgroup is determined by its physical location instead of the way to keep the balance of the key tree. Of course it can happen that the cluster nodes are not evenly distributed over data centers and the key tree becomes unbalanced. However, in this case the loss of performance is not so big. Let N be the number of cluster nodes, and s be the number of seed nodes. In an extreme case, let's assume all nodes are in only one seed node's subgroup. The communication cost of the seed node is $O(\log N)$ for node join/leave operation. On the other hand, if the nodes are evenly distributed, each seed node has N/s subgroup members. In this case the communication cost of the seed node is $O(\log N/s)=O(\log N)-O(\log s)$. That is, the increased cost is $O(\log s)$ in worst case. As s is quite smaller than N and the number of seed nodes is not so big in practice, the performance degradation due to unbalanced tree is endurable. Therefore, it is efficient to manage the key tree considering physical location of the nodes due to the nature of Cassandra.

3.3. Stability

Centralized approaches suffer from SPOF (single point of failure) when the key server crashes due to critical reasons such as network failure and power failure. Our model does not suffer from SPOF because there is not centralized key server. However, there arises a problem if one of seed nodes crashes because our model partially adopts centralized approach to manage subgroups. Our suggestion to overcome this problem is adding replication of seed node. One cluster node in the same data center is chosen, and it receives subgroup metadata from the seed whenever changed. Later this node becomes a seed node when the original seed node crashes.

4. Conclusion

In this paper we presented the way to improve Cassandra security with group key. When the nodes communicate to maintain the cluster, transferred messages are protected from outside adversaries and cluster membership can be confirmed efficiently. Our group key management model is decentralized in consideration of Cassandra structure. Therefore, it is more scalable than distributed models and more available than centralized models.

The presented model solves the security issue in managing inside cluster network. However there still remain other security issues to be solved. Especially an efficient authentication method for cluster nodes and Cassandra users is necessary. Combining this method and our model will protect Cassandra more securely.

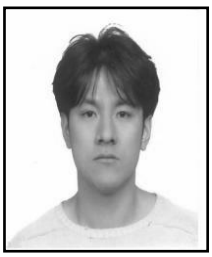
Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the ministry of Education, Science and Technology (2011-0013757).

References

[1] D. Agrawal, S. Das and A.E. Abbadi, "Big Data and Cloud Computing: Current State and Future Opportunities", 14th International Conference on Extending Database Technology, **(2011)**, pp. 530-533.
[2] F. B. Degefa and D. Won, "Extended key management scheme for dynamic group in multi-cast communication", Journal of Convergence, vol. 4, no. 4, **(2013)**, pp. 7-13.
[3] K. Grolinger, W. A. Higashino, A. Tiwari and M. AM Capretz, "Data management in cloud environments: NoSQL and NewSQL data stores", Journal of Cloud Computing, vol. 2, no. 22, **(2013)**.
[4] E. Hewitt, "Cassandra : The Definitive Guide", O'REILLY Media, Inc., **(2010)**.
[5] R. Hussain, "Cooperation-Aware VANET Clouds: Providing Secure Cloud Services to Vehicular Ad Hoc Networks", Journal of information processing systems vol. 10, no. 1, **(2014)**, pp. 103-118.
[6] D.-W. Kwak and J.-W. Kim, "A Decentralized Group Key Management Scheme for the Decentralized P2P Environment", IEEE Communications Letters, vol. 11, no. 6, **(2007)**, pp. 555-557.
[7] Y.-D. Kim, A. Perrig and G. Tsudik, "Tree-based Group Key Agreement", ACM Transactions on Information and System Security, vol. 7, no. 1, **(2004)**, pp. 60-96.
[8] A. Lakshman and P. Malik, "Cassandra : A Decentralized Structured Storage System", ACM SIGOPS Operation Systems Review, vol. 44, no. 2, **(2010)**, pp. 35-40.
[9] M.I. Malkawi, "The art of software systems development: Reliability, Availability, Maintainability, Performance (RAMP)", Human-centric Computing and Information Sciences, vol. 3, no. 1, **(2013)**, pp. 1-17.
[10] L. Okman, N. Gal-Oz, Y. Gonen, E. Gudes and J. Abramov, "Security Issue in NoSQL Database", International Joint Conference of IEEE TrustCom-11, **(2011)**, pp. 541-547.
[11] B. P. Rimal and E. Choi, "A Taxonomy and Survey of Cloud Computing Systems", Fifth International Joint Conference on INC, IMS and IDC, **(2009)**, pp. 44-51.
[12] R. V. Renesse, D. Dumitriu, V. Gough and C. Thomas, "Efficient reconciliation and flow control for anti-entropy protocols", ACM LADIS'08 Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware, **(2008)**.
[13] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication", ACM Computing Survey, vol. 35, no. 3, **(2003)**, pp. 309-329.
[14] D. Wallner, E. Harder and R. Agee, "Key management for multicast : Issue and architectures", RFC 2627, **(1999)**.
[15] A. Waleed and L. Chunlin, "User privacy and security in cloud computing", International journal of Security and Its Applications, vol. 10, no. 2, **(2016)**, pp. 341-352.
[16] S. A. Alhumrani and J. Kar, "Cryptographic protocols for secure cloud computing", International journal of Security and Its Applications, vol. 10, no. 2, (2016), pp. 301-310.
[17] X. Li, H. Liand and D. Jia, "Research on privacy protection approach for cloud computing environments", International journal of Security and Its Applications, vol. 9, no. 3, **(2015)**, pp. 113-120.

Author



Heeyoul Kim, He received the B.E. degree in Computer Science from KAIST, Korea, in 2000, the M.S. degree in Computer Science from KAIST in 2002, and the Ph.D. degree in computer science from KAIST in 2007. From 2007 to 2008, with the Samsung Electronics as a senior engineer. Since 2009 he has been a faculty member of Department of Computer Science at Kyonggi University. His main research interests include cryptography & security such as secure group communication and clod computing security.