Secure Group Communication for Cassandra

Heeyoul Kim

1 Department of Computer Science, Kyonggi Univ., Republic of Korea
heeyoul.kim@kgu.ac.kr

Abstract. Cassandra is a distributed NoSQL system providing high scalability and availability. However, it has many security vulnerabilities. Especially the information about internal cluster structure and transferred data can be exposed by outside attackers. In this paper, we present a novel way to enhance Cassandra security by means of group key. The group key model is designed considering Cassandra’s physical structure, and it is a decentralized model where the nodes are divided into several subgroups. The proposed model helps protecting internal cluster from outside by confirming the node’s cluster membership.

Keywords: Cassandra, security, group key, decentralized model

1 Introduction

There have appeared various NoSQL systems in order to process Big Data where SQL-style querying is not the crucial objective. However, the research on security of NoSQL systems has not been covered in depth enough, thus a lot of concerns have been encountered [1]. Cassandra, one of popular NoSQL systems, works as a distributed storage where several nodes constitute a cluster [2]. Since there has been less consideration of security, the transferred data and metadata for cluster management can be exposed by outside adversaries.

In this paper, we present a novel way to enhance Cassandra security by means of group key. Only legitimate group members know the group key, and the key is used to encrypt transferred messages in the group. Also, the key is used to confirm cluster membership for Cassandra. We present a decentralized group key management model suitable for Cassandra structure, which blends both centralized approach and distributed approach.

2 Cassandra Security

Cassandra is designed to handle Big Data across multiple nodes. 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 one 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 topology of ring cluster to the newcomer node. Multiple seed nodes can be designated, for example, per a data center [3].

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 eavesdropped by attackers. Especially Cassandra uses gossip protocol to share and state information about cluster for every second. Therefore, it is inefficient to authenticate each node every time. Our suggestion is that just confirming legitimate cluster membership is sufficient instead of peer authentication, and it can be performed efficiently with the presented group key management model below.

3 Secure Group Communication for Cassandra

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. On the other hand, a distributed group key management approach induces critical performance degradation. Therefore, we adopt a decentralized approach [4] 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. LKH scheme [5], 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 [6], one of distributed approach.

Let $n_a$ denotes a member node and $SD_b$ denotes a seed node. Let $(a,b)$ means b-th node at level $a$ in a tree, and let $K_{(a,b)}$ and $BK_{(a,b)}$ denote $n_{(a,b)}$‘s private key and blinded key.

3.1 Initialization

In the initial phase, all seed nodes of the cluster construct a key tree and generates a group key following TGDH scheme. 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 where \( p \) is a large prime number and \( \alpha \) is a primitive root modulo \( p \).

\[
K_{(a,b)} = (BK_{(a+1,b)})^{\alpha \mod p} \mod p = (BK_{(a,1,2)})^{\alpha \mod p} \mod p = \alpha^{\alpha \mod p} \mod p .
\] (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.

### 3.2 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. Fig. 1 shows both the expanded key tree after join and corresponding key update protocol.

![Fig. 1. Join operation for new member \( n_i \)](image)

1. \( SD_2 \rightarrow \{n_3, n_6, n_7\} \rightarrow \{K'_{(1,1)}\}_{K_{(1,1)}} \)
2. \( SD_2 \rightarrow \{n_7\} \rightarrow \{K'_{(2,3)}\}_{K_{(2,3)}} \)
3. \( SD_2 \rightarrow \{n_6\} \rightarrow \{K'_{(1,1)}, K'_{(2,2)}\}_{K_{(3,2)}} \)
4. \( SD_2 \rightarrow \{K'_{(0,0)}\}_{K'_{(0,0)}} \rightarrow \{K'_{(1,0)}\}_{K_{(1,1)}} \)
5. \( SD_2 \rightarrow \{SD_1\} \rightarrow \{BK'_{(1,1)}\}_{K_{(1,1)}} \)
6. \( SD_1 \rightarrow \{K'_{(0,0)}\}_{K'_{(0,0)}} \rightarrow \{K'_{(1,1)}\}_{K_{(1,1)}} \)
7. \( SD_1 \rightarrow \{n_1, n_4\} \rightarrow \{K'_{(0,0)}\}_{K_{(1,1)}} \)
8. \( SD_2 \rightarrow \{n_3, n_6\} \rightarrow \{K'_{(0,0)}\}_{K'_{(1,1)}} \)

### 3.3 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. Fig. 2 shows both the key tree after \( n_i \) leave and corresponding key update protocol.
Fig. 2. Leave operation after \( n_i \) leaves

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.

Acknowledgments. This work was supported by the GRRC program of Gyeonggi province.

References