International Journal of Engineering Innovation & Research
Volume 2, Issue 4, ISSN: 2277 – 5668

Explicit Data Encryption Architecture for Cassandra

Ramanjaneya Swamy
Department of CNE
Siddaganga Institute of Technology, Tumkur Karnataka, India
Email: ramanjaneyaswamy@gmail.com

Mr. Srinivasa K.
Department of CNE
Siddaganga Institute of Technology, Tumkur Karnataka, India
Email: srinivas.karur@gmail.com

***Abstract* – The recent advances in distributed database applications has created the need for large storage which should be scalable and should not be restricted to schema constraints. In this move many companies are moving from relational database systems to Non-relational database systems, particularly known as NoSQL databases. As opposed to relational database systems, these new database systems lagging in providing security in order to achieve high performance and scalability. Cassandra is one among the several existing NoSQL databases whose primary concern is scalability, high availability and no single point of failure for retrieving the stored data. As mentioned earlier, for Cassandra also the security is not a primary concern. Here we proposed a method for storing the data into Cassandra securely using explicit data encryption scheme. This paper gives a detailed overview of Cassandra database with its data model, architecture, security issues and finally the proposed architecture for storing the data into Cassandra securely using explicit data encryption method.**

***Keywords* – Scalability, High Availability, Reliability, Big Table, Dynamo, Symmetric, Consistent, Flexible Portioning, Replication and NOSQL.**

I. INTRODUCTION

In today's world e-commerce and social media occupy large portion of web usage, as a result there is a growing need for technology to handle large amount on data related to users. This particular need has been so far meet by using technologies like cloud computing and distributed web applications. But there is a huge draw in terms of scalability and reliability as it involved very large amount of data. To overcome these many companies have adopted various types of non-relational databases, normally called as NOSQL databases [1]. Different NOSQL databases take different approaches. One such NOSQL database is Cassandra which is developed by Prashant Malik and Avinash Laxman for Facebook [2]. The main purpose of Cassandra is to have high availability with no single point of failure. Cassandra is based on the combination of concepts of Google's Big Table [3] and Amazon's Dynamo. Further to add to this, symmetric, consistent hashing, flexible partitioning, replication, membership and failure detection have been major features of Cassandra. Its major drawbacks include Cassandra data files, inter cluster communication, and denial of service. In this paper we discuss these drawbacks of Cassandra one by one. Most importantly we will concentrate on Cassandra data files which do not provide any mechanism to automatically encrypt the data thus making it vulnerable for hacking by someone who has direct access to the file system. After discussing the drawbacks we will provide solution for problem pertaining to Cassandra data files by making appropriate architectural changes.

II. OVERVIEW

In this section Cassandra architecture, data model and its features will be discussed.

A. Data Model

Cassandra is one of the NOSQL databases [6] which are designed to handle large amount of data which is coming from cloud computing and distributed applications and which are spread across number of different servers. So in any point of time the data should be available with no failure and it should not also limit any users.

Cassandra data model[4] consists of the following terms.

- Column – it is the smallest unit of data which holds the users data. This is normally a triplet that holds name, value and a timestamp.
- Super column – it is similar to column but it will be having nesting structure in it. In the simple words super column are the ones whose values are again columns.
- Row – it is collection of columns which are identified by unique key. In Cassandra each row can have different columns with respect to other columns.
- Column family – it is the container for the columns but it can be analyzed to table format of the relational databases. It holds the order list of the columns which are referenced by unique column name. Each row in a column family can be referenced by its key.
- Key space -- A key space is the top level dimension of the Cassandra hash, and is the holder for column families. Column families are subordinate to exactly one key space.

B. Architecture

Considering the performance of the database, it has the architecture to support necessary features like scalability, high availability, fault tolerance, concurrency and robustness [5].

Cassandra is shaped by two systems like Google's Big Table and Amazon's Dynamo. Big Table uses the distributed file system called Google file system and it has single master which stores Meta information regarding servers which are located. And the servers which are residing will store the actual data, whereas Dynamo is based on the distributed storage system and eventually consistent and in fact it provides high availability service. Describing the details of each the solutions is out of the paper so discussing only the core features as follows.

B.1 Partitioning:

Cassandra provides partitioning feature to handle the large amount of data being inserted into database.

Cassandra maintains a list of nodes and as the data comes, it distributes it among nodes in either consistent or inconsistent manner. In the former approach system evaluates the given data using hash function and given key. Then data is assigned to node corresponding the value generated by hash function. Hash function is designed to treat cluster of nodes in circular fashion where largest node value wraps to the value of lowest node. Partitioning takes care of choosing coordinator for nodes where all the requests for that node are routed

B.2 Replication:

Cassandra is responsible for providing high data availability by replicating data across remaining N replicas by quorum protocol to acknowledge read/writes spread to all the replicas. Whenever a node is updated/wrote by an application, coordinator spreads relevant updates across remaining N-1 nodes. User has an option to specify how this replication should take place across all N replicas.

Types:

- In Rack Unaware Strategy: replicas are always placed on the next (in increasing Token order) nodes along the ring.
- In Rack Aware Strategy: a replica is placed in the first node along the ring that belongs in another data center than the first; the remaining replicas, if any, are placed on the first nodes along the ring in the same rack as the first.

B.3 Membership:

Cassandra does not introduce new mechanism of its own to maintain membership among all system nodes[9].It uses Scuttlebutt gossip protocol to maintain membership among participating nodes in the system. It is responsible to carry out frequent checks for membership information and spread out relevant information to all the other nodes so that all the nodes within system boundary know about existing, live and communicable nodes. Besides transmitting membership information, scuttlebutt can also be used to spread out system related control information.

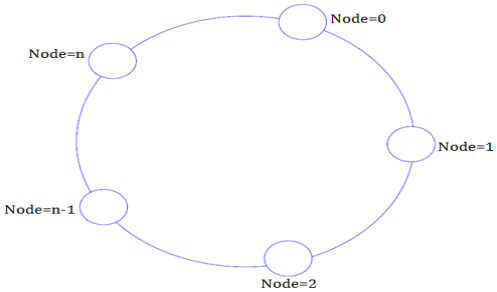

Fig.1. Membership Diagram.

Membership diagram for N nodes arranged in circular fashion.

Hash function is out value = (Key) % (N) where data is assigned to the node with sequence number=out value.

B.4 Failure detection:

It involves the task of checking whether any of the system nodes is up or down. Unlike traditional deterministic failure detection mechanism, Cassandra uses more efficient probabilistic model to check if node is down or up and running. This approach is based upon the accrual failure detection algorithm which uses the threshold parameter denoted as Φ. Its value is set according to local network conditions and load present at particular node for which we want to detect failure. As a part of failure detection, module emits the value known as suspicion level value which reflects the fact that current node is down. Now probability that this statement is false (due to late reception) depends upon the value of set threshold Φ. More the value of Φ more is the probability that suspected node is actually down (And decision won't be affected by late message reception).

B.5 Load Balancing mechanism:

When new data node is added to system, hash function evaluates and assigns a token in such a way that it can load balance heavily loaded node. Load balancing heavily loaded node is also called as Bootstrapping. The process of Bootstrapping can be initiated either through command line or Cassandra dashboard user interface. Whenever bootstrapping occurs, specific ranges of nodes get transferred from old node to newly introduced node. Copy operation is performed in data stream replication mechanism which takes place in kernel-kernel copy operation. However this example shows only single replica participating in Bootstrapping. It is also possible to deploy more replicas to perform parallel Bootstrapping on the behalf of participating nodes.

B.6 Read and Write Operations:

Write:

Cassandra writes data in the persistent local file system. Data is stored in the format such that it would allow efficient and reliable read results. However before storing in the file system, it writes data in the commit log for durability in order to recover from any possible system failovers. After system's confirmation that desired data has been successfully written to commit log it actually performs write to the file system residing on system disk. Data is stored according to index defined on each key in a row for efficient search and query processing. Each write is maintained as separate file. Frequently a process called as merge is ran which unifies these files into single big file.

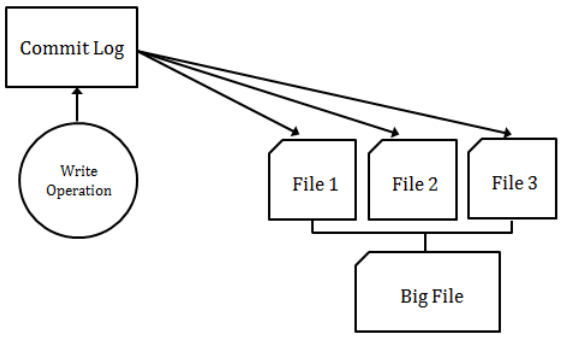

Fig.2. Cassandra write operation.

Read:

Read operation takes help from in memory data structure which helps in locating key indices corresponding to relevant rows. Each row is uniquely

identified and indexed by the key column associated with it. While looking for the key, it is possible to perform lookup in a file which does not have relevant key. This overhead can be eliminated by using bloom filter which makes use of another file which contains list of keys associated with that particular data file. This mechanism avoids look up in the data file which does not have specific key. As Cassandra offers this facility, column pair (Key, timestamp) are sorted in descending order of timestamps so that latest record always appears first. This is in accordance with the fact that most users are in need of latest version of available data.

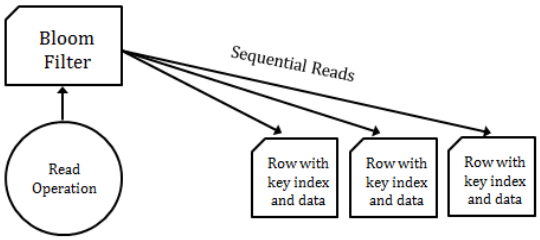

Fig.3. Cassandra read operation.

III. Cassandra Security Issues

Security was not a primary concern of Cassandra's designers. As a result there are few issues in its design. Following are the main issues which are identified.

- Cassandra Data Files. The data in Cassandra is kept unencrypted and Cassandra does not provide any mechanism to automatically encrypt the data in storage. This shows that any attacker with access to the file-system can directly retrieve the information from the files. In order to solve this, the application must explicitly encrypt any sensitive information before writing it to the database.
- Inter-cluster communication. In general, nodes on a cluster can freely communicate and no encryption method or authentication is applied.
- Denial of Service problem. Cassandra uses a Thread-Per-Client model in its network code. During setting up a connection requires the Cassandra server to start a new thread on each connection. An attacker can prevent the Cassandra server from accepting new client connections by causing the Cassandra server to allocate all its resources to fake connection attempts.

IV. Proposed Work

In this section we are proposing a solution to one of the issues which we have identified in the previous section.
The brief introduction to the selected issue is as follows
Cassandra Data Files: The data which is stored in the Cassandra database is in unencrypted form and it does not provide any mechanism to automatically encrypt the data in storage [1].

If any application is in need of storing any information in encrypted manner then it has to specify explicitly to get the job done.

The solution has been carried out in section wise as follows

- User creation section.
- Data insertion section using explicit data encryption
- Data retrieval section using explicit data decryption.

Each of the section is described with a proper diagram.

I. User Creation Section

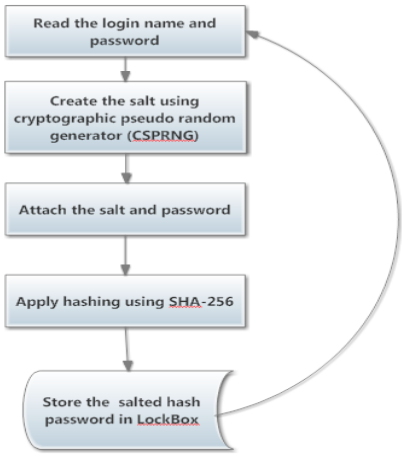

Fig.4. User Creation Phase.

The above diagram depicts the first level of security that can be given to the Cassandra database.

After the installation phase it should prompt for user creation where the person who is installing will act as an admin and his credentials can be stored. Hence a first level of security is achieved.

It does the security functionality as follows

- It should prompt a window for creating user and it should read the credentials of the user.
- Generating the cryptographic secure pseudo random generator.
- Adding the password and salt which is created in the second step.
- Applying the hashing method using SHA-256 for better security purpose.
- Storing created salted hash password in USERS column family.

By this technique we can assure of storing the password of an admin in secure manner. The USERS Column family will be having a following format

Username	Salt used	Salted hash password

Accessing of this column family will be restricted to only admin.

Validation Procedure can be followed as follows.

- Read the password.
- Retrieve the Salt used from USERS column family.
- Attach the salt to the password and apply the same Hashing algorithm.
- Now compare the generated salted password with the stored one.
- Give access only for the matched ones.

In this the admin can create the extra users which can access the database and creating a secure environment for only trusted users. The list of users can be added to USERS column family and this column family can be accessed only by admin.

II. Explicit Data Encryption Architecture:

It helps in encrypting sensitive and confidential data such as credit card numbers, social security numbers etc stored in the column family. Encrypted data is transparently decrypted for the users which have been created.

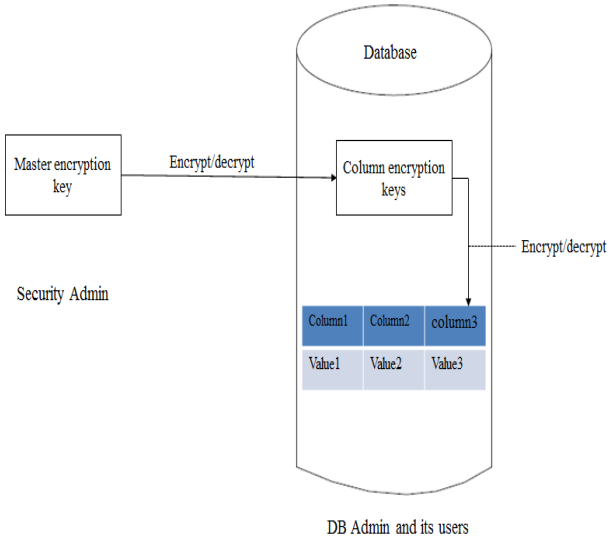

Fig.5. Explicit Data Encryption Architecture

This architecture has the following entities namely

- Master Encryption key --> This is stored in an external security module also called as wallet which is outside of the database and accessible only to security administrator.

This key is used to encrypt column encryption key.

- Column Encryption Key --> This key is used for encrypting the data which is going to be inserted into the database.
- Database admin --> This admin will create the users who are part of the Cassandra database.
- Security Admin --> This admin will create the external security module which will act as a wallet and there he will set the master encryption key.

Using an external security module separates the general functions from encryption operations, making it possible to divide the tasks between database administrators and security administrators.

Here the Security is enhanced because wallet password is unknown to database administrator requiring security administrator to provide the password.

Before applying the encryption or decryption to the database columns, security admin should set the master encryption key and this key will encrypt the column encryption key.

Then the encryption to the column family can be applied.

Interactions during the INSERTION phase involve:

- Whenever insertion query occurs in any of the column family, then the explicit encryption should occur.
- It should read the column encryption key.
- Apply the encryption using symmetric encryption algorithms such as Blowfish, 3DES or AES.
- Store the ciphered values into the database.

Interactions during the RETRIEVAL phase involve:

- Whenever select operation occurs, it should apply the reverse process of explicit encryption.
- Again it should read the column encryption key.
- Apply the decryption using the same algorithm.
- Display the original values to the user and hence proper way of displaying the content.

V. Conclusion

The two level of security which takes place during installing the Cassandra database and applying the explicit data encryption to the data which is going to insert into the database really enhances the security feature. And hence the stated issue can be resolved.

This level of security creates a secure environment and gives way of accessing only to the legitimate users.

References

[1] Lior Okman,Nurit Gal-OZ, Yaron Gonen, Ehud Gudes, Jenny Abramov," Security Issues in NoSQL Databases, 2011 International Joint Conference of IEEE.
[2] Avinash Lakshman, Prashanth Malik, "Cassandra – A Decentralized Structured Storage system".
[3] Venkata Poonam, "A survey on Cassandra".
[4] Sumit Goyal, "Survey paper B534 Distibuted system".
[5] Jiang Wu, "Decentralized storage system- Cassandra" , 26 march 2011.
[6] Guoxi Wang, Jianfeng,"The NoSQL Principles and Basic Application of Casandra Model", 2012 International Conference on Computer Science and service system.
[7] Cassandra Wiki, http://wiki.apache.org/cassandra/
[8] Apache Cassandra Documentation, "http://solvehow2. blogspot. com".
[9] Jayesh Kawali," Apache Cassandra Distributed Database Mangaement System".

Author's Profile

Ramanjaneya Swamy

Ramanjaneya Swamy, pursuing his M. Tech in Computer Network and Engineering from the department of Computer Science & Engineering in Siddaganga Institute of Technology, Tumkur under Vishveshvaraya Technological University (VTU), and received his B.E in 2010 from B. V.Bhoomraddi College of Engineering, Hubli under VTU, Belgaum, Karnataka. His major field of study is on Information Security.

Mr. Srinivasa K.

Srinivasa.K working as an Assistant Professor in the department of Computer science and Engg, at SIT Tumkur. He received his M.Tech in 2010 from NITK surathkal and BE in 2004 from VEC,Bellary,Karnataka. His area of interest covers big data Analysis, Data Stream management and data structures.