Secure Authentication in Document Oriented Database MongoDB

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

in

Information Security

Submitted By

Palvi Aggarwal

(801233014)

Under the supervision of:

Dr. (Mrs.) Rinkle Rani

Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

July 2014

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "***Secure Authentication in Document Oriented Database MongoDB***", in partial fulfillment of the requirements for the award of degree of Master of Engineering in ***Information Security*** submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of ***Dr. Rinkle Rani*** and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Palvi

(Palvi Aggarwal)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Rinkle 07/07/14

(Dr. Rinkle Rani)

Assistant Professor,
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

(Dr. Deepak Garg)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

The successful completion of any task would be incomplete without acknowledging the people who made it possible and whose constant guidance and encouragement secured the success.

First of all I wish to acknowledge the benevolence of omnipotent God who gave me strength and courage to overcome all obstacles and showed me the silver lining in the dark clouds.

With the profound sense of gratitude and heartiest regard, I express my sincere feelings of indebtedness to my guide **Dr. Rinkle Rani**, Assistant Professor, Computer Science and Engineering Department, Thapar University for his positive attitude, excellent guidance, constant encouragement, keen interest, invaluable co-operation, generous attitude and above all his blessings. He has been a source of inspiration for me.

I am grateful to **Dr. Deepak Garg**, Head of Department, and **Dr. Jhilik Bhattacharya**, P.G. Coordinator, Computer Science and Engineering Department, Thapar University for the motivation and inspiration for the completion of this thesis. I will be failing in my duty if I do not express my gratitude to **Dr. S. K. Mohapatra,** senior Professor and Dean of Academics affair in Thapar University, for making provisions of infrastructure such as library facilities, computer labs equipped with internet facility, immensely useful for the learners to equip themselves with latest in the field.

Last but not the least I would like to express my heartfelt thanks to my parents and my friends who with their thought provoking views, veracity and whole hearted co-operation helped me in doing this thesis.

(Palvi Aggarwal)

Roll No. 801233014

Abstract

Today organizations almost in every industry are being showered with quantities of new information. Along with traditional sources, many more data channels and categories exist. Collection of this vastly large information volume and new assets are known as big data. Tradition Relational databases deals with the structured data, but now a day's data is in form of graphs, images, audio, videos and text which is either partially structured or completely unstructured. To deal with such type of data NoSQL databases are introduced. MongoDB is document oriented NoSQL database which stores documents as databases. This is schema less databases, which does not need to follow a particular structure. Horizontal scaling, sharding, replica set, high performance capabilities enhance its demand. But there are some security issues in MongoDB. Security is not major concern of 10gen. In this thesis, research has been carried out to find out the security loopholes and to provide secure authentication in database.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Introduction

Efficient storage and retrieval of data has always been an issue due to the growing needs in industry, business and academia. So, organized storage solutions are required to handle massive amount of data which is generated by a large number of transactions and experimentation. Therefore, to store and retrieve the data in an organized manner, databases were created. Since their origin in the 1960's different types have emerged, each database uses its own representation of data and technology to handle the transactions. They began with navigational databases which were based on linked-lists, moved on to relational databases, afterwards object-oriented and in the late 2000s NoSQL emerged and has become a popular trend [1].

Relation databases have been used for a long time for storing, retrieving and processing data for industries. Relation database management systems (RDBMs) are widely used for storing structured data in web and business applications. Since few years, with the growth of data, demand of database management systems increased. Relation databases provide variety of features and strict data consistency. Relational databases uses separate tables in which column specify field name and row specify record [2]. These tables are connected using foreign keys. The schema design of relational database is shown in Figure 1.1.

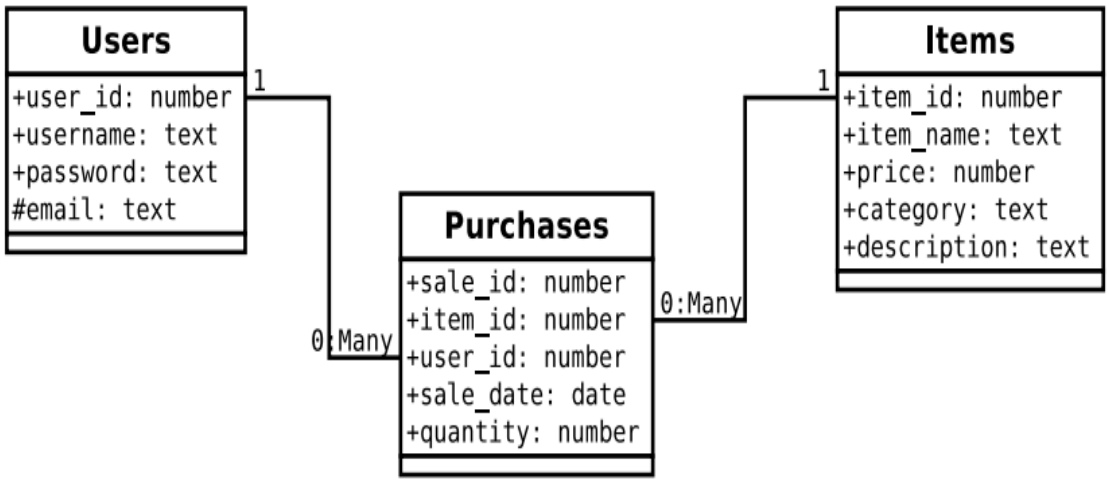

Figure 1.1: Relational Database Schema

1.2 Normalization

An important aspect of relational database is normalization. The normalization [3] involve following steps.

- **First Normal Form (1NF):** First normal form removes repeating data in fields by creating new tables.
- **Second Normal Form (2NF):** In 2NF, non-prime attribute should not be dependent on any proper subset of candidate key.
- **Third Normal Forma (3NF):** Every non-prime attribute should non-transitively dependent on every super key of table.

1.3 ACID Properties

An important feature provided by relational databases to provide reliability of transaction is ACID properties [4]. ACID properties for database are explained below:

- **Atomicity:** Atomicity ensures that either all the parts of transactions are completed or none. If any part of transaction fails, the whole transaction should fail. Hence the state of the transaction remains unchanged.
- **Consistency:** Consistency ensures that any transaction will bring the database in valid state. Everything that is being written in database is according to the rules defined.
- **Isolation:** Isolation ensures that inconsistency of one transaction would not be visible to another transaction. Isolation property gives concurrency control.
- **Durability:** This property ensures that once the transaction is complete, the changes will be preserved.

NoSQL databases were introduced to provide a more flexible, efficient and economical way of managing data. NoSQL was first introduced by Carlo Strozzi in 1998 [5]. The name signifies non-relational databases. The motivation behind these databases is simplicity of design, horizontal scaling and good performance. NoSQL encircled with variety of database management technologies and was basically designed to handle large amount of data, objects and products, the frequency, performance and processing needs. On the other hand relational databases were not designed and implemented to handle the

challenges that face today's advanced applications and also they were not able to take advantage of the low cost storage and high processing power available today.

1.4 CAP Theorem

In 2000 Eric Brewer had proposed the idea that there are many difficulties to maintain consistency, availability and partition tolerance at the same time in the distributed systems. This idea is known as CAP theorem [6]. The three attributes of CAP theorem are:

- **Consistency:** All the data must be seen same to all nodes at the same time**.** There should not be any change in data on any node.
- **Availability:** The availability of data should be managed in different situations and it should guarantee that every request which is made by its users will get a response about its query whether it was successful or failed.
- **Partition Tolerance:** This is the very big factor the system can be able to continue in all the circumstances like data loss and other damage.

1.5 Relaxing ACID

ACID is the abbreviation which consists of basic elements of any transaction: atomicity, consistency, isolation and durability. These terms are surely familiar to anyone who deals with databases. The basics of any transaction are defined by these terms. To ensure that transaction meet ACID property, strict rules are defined. It has been observed that in order to provide scalability, some of these terms should be relaxed or redefined. NoSQL is the solution provided to relax these qualities. In a distributed environment, it is a big deal for communication system to ensure consistency. To provide consistency locks are used, due to which systems are forced to wait for mutually shared data items.

Even in cases where multiple systems are generally not operating on the same piece of data, there is a great deal of overhead that prevents systems from scaling. To solve this, most of the NoSQL offer something known as "eventual consistency [7]". This relaxes the complete consistency feature of relation databases. Using this concept, each system is

allowed to make data updates and learn other updates within small amount of period. Now there is no need of becoming consistent every time.

Multi-version concurrency control is another approach for optimistic concurrency control. This technique allows one transaction to consistently read which is concurrent to another transaction which is writing. But it does not deal with write conflicts. It can provide a provision of more transaction retries in case of overlap or long running time of transaction.

1.6 Growth of Data

Today, the usage of NoSQL technology is increasing quickly among Internet companies and the enterprise [8]. It is gradually considered as reasonable alternate for relational databases, especially for the organizations who operate at high scale. It is more effectively achieved by running on commodity servers, standard clusters, and a schema-less data model. This approach is better to handle variety of data that is usually captured and processed.

Through third parties like facebook and twitter, data capturing becomes easy. User generated content, sensor generated data, machine logging information, personal user data, geographical location data and social graphs are some of the examples of variety of data captured. It's not astonishing that developers are using this data to enrich existing applications and to create new applications too. The collected data is becoming basis for market analysis, for sentiment analysis, stock market and various decision making applications. The collected data is also being used in shopping, entertainment, relation management and communication [9]. Applications that don't get ways to control it quickly will quickly fall behind. Big data is combination of semi structured and unstructured form of data that arrive continuously in large amount as shown in Figure 1.2. We are oversupplied in a flood of data today.

In a variety of application area, data is being collected at extraordinary scale. Previously the decisions were made on guesswork. No strong basis was there for making decisions. But now, this large volume of data is being used in the process of decision making. Big data analytics is dominant area for decision making.

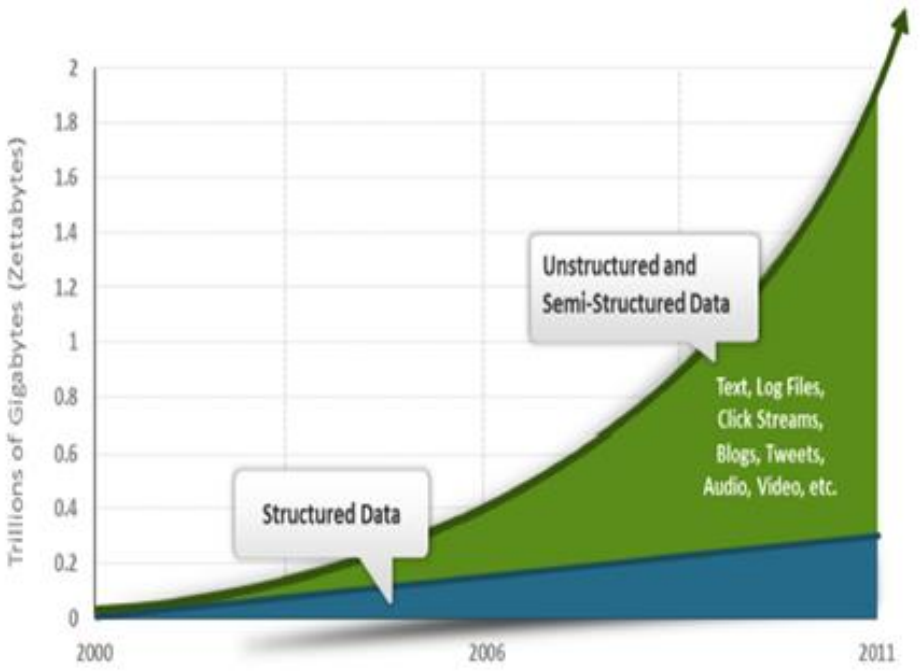

Figure 1.2: Growth of big data [9]

Developers demand a very flexible database that can easily accommodate any new type of data on which they want to work. Database is not interrupted by change in content structure from third-party data providers. New data is generally is unstructured and semi--structured, there is a need for developers to have a database that is capable for storing it efficiently. Unfortunately, schema-based approach used by relational databases is not good for such unstructured and semi structured data.

Finally, with the rise in value of processing data, developers get irritated with the "impedance mismatch" between the object-oriented approach and the schema-based tables of a relational database. Data model of NoSQL provide better mapping to the application's organization of data. It also simplifies the relations between the application and the database which results in less code to write, debug and maintain.

1.7 NoSQL Database Classification

NoSQL data bases are classified into four categories as shown in Figure 1.3 [10, 11]. Document based databases, Graph databases, Key Value databases and Column databases are various classes of NoSQL. All these databases serves different role and used in different applications.

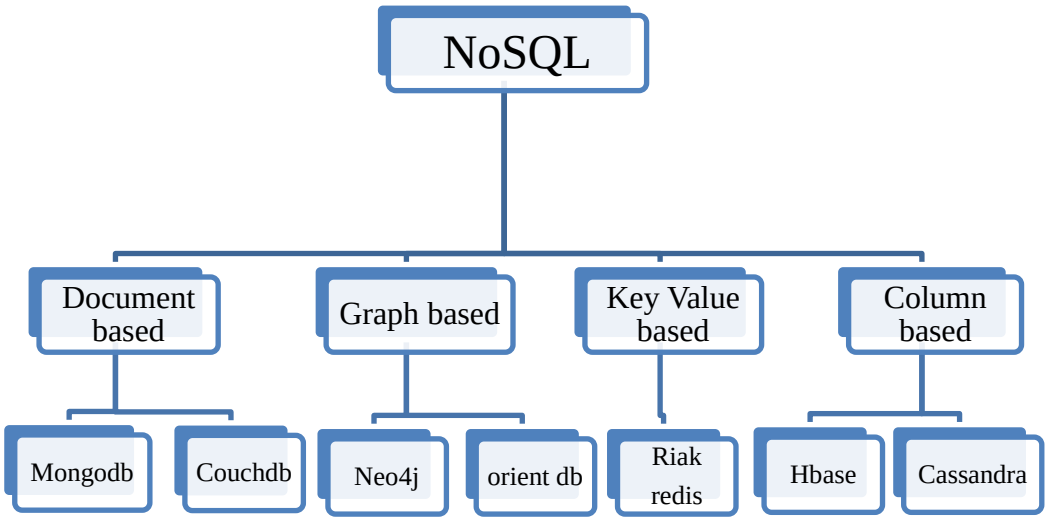

Figure 1.3: Classifications of NoSQL Databases

1.7.1 Document Databases

A document based databases are designed for storing, retrieving and managing document oriented information which is also known as semi structured information [12]. This database couples each key with a data structure known as a document. Document is stored in JSON format in different key value pairs. Examples of Document databases are MongoDB and Couchdb. The structure of document based database is given in Figure 1.4.

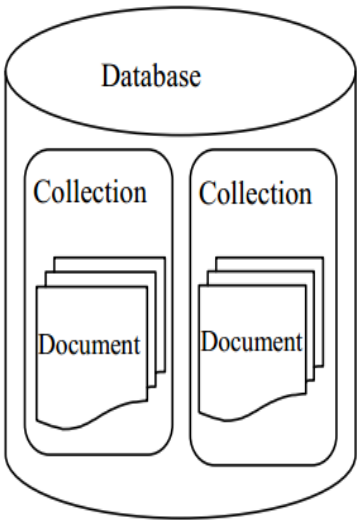

Figure 1.4: Representation of Document Store

- **CouchDB**: CouchDB (Cluster of Unreliable Commodity Hardware Data Base) is schema-less, fault-tolerant, distributed document database [12]. It is maintained by Apache Software Foundation and written in Erlang. CouchDB is schema free, it means in this data-store data is stored in arbitrary JSON format [13]. To access the CouchDB data store RESTful HTTP/JSON API is used. To manage concurrent access to the documents, CouchDB uses Multi-Version Concurrency Control (MVCC). It means documents are versioned using Sequence ID. If someone fetches the document for updates, CouchDB will notify the application. Then application combines the update or overwrites the update with its update. So the file on disk is always in consistent state. To query the database "views" are used. Views are of two types, temporary views and permanent views. Temporary views are loaded into memory and permanent views are stored on disk [14]. Views are basically map/reduce functions, implemented in JavaScript, that process the data stored and return the results. CouchDB keeps these views updated by running new inserts through map/reduce functions. CouchDB supports asynchronous replication for scaling. It has interfaces with many programming languages like JAVA, C, PHP, LISP, and Python that converts Native API calls into RESTful calls.
- **MongoDB:** MongoDB is a GPL open-source document database developed by 10gen in 2008. It is written in C++ and its query language is javascript. The word mongo in its name comes from word humongous [15]. MongoDB does not support Joins or ACID transactions. MongoDB was designed to handle growing data storage needs. MongoDB provides interactive JavaScipt shell for database management. It supports a rich, ad-hoc query language of its own. In addition, there exist bindings for many programming languages like Java, C, C++, Erlang, Python, Ruby and more. The data type of many languages is built up of key-value pairs and some languages support JSON [16]. It is easy to create documents for MongoDB using these languages. MongoDB keeps the data in RAM which is used most recently. When indexes are created for queries, all data sets fits in RAM and queries are run from memory. There is no query cache in

MongoDB. All queries are run directly from the indexes or data files. Data is physically written to the disk with in 100 milliseconds.

1.7.2 Graph Stores

Graph databases use graph data structure with nodes, edges and properties that store data [15]. It provides index free adjacency. This means every entity can directly point to its adjacent entity. These graph stores are used to store the information about networks like social connections. Graph stores consist of Neo4J and Orient DB. The graph database structure is shown in Figure 1.5.

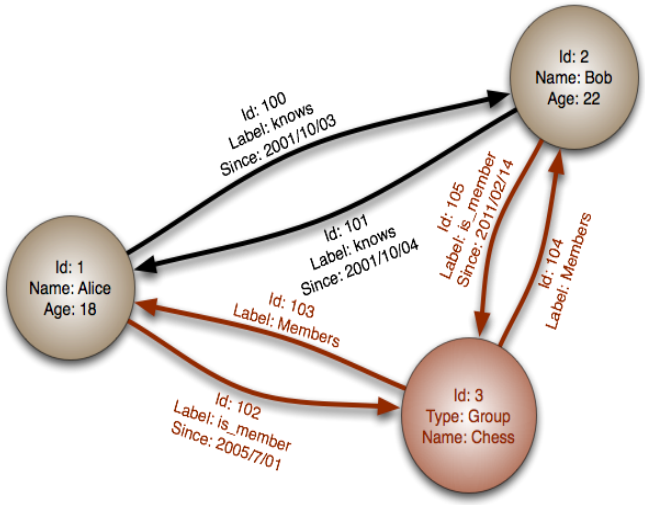

Figure 1.5: A View of Graph Store

- **Neo4j:** Neo4j is one of the leading open source graph database [18]. It is supported by Neo Technology. It stores the data in the form of graphs rather than tables. It is implemented in Java, but it has binding for many languages like Python, Ruby, Jython, Scala. It supports ACID transactions and master-slave replication. Neo4j is exceptionally scalable and mainly used in embedded applications. Cypher and Gremlin query languages can be used for graph traversals.
- **OrientDB:** OrientDB is popular open-source document-graph database having features of both document and graph database [19]. Like document databases, it stores the data in the form of documents. Like graph databases, relationships

are managed as direct connections between records. It is purely written in java. It supports ACID transactions and recovers pending documents on crash. It supports extension of SQL language that handle relationships without using Joins. It is extremely light. It uses MVRB-Tree indexing algorithm, which is derived from the Red-Black Tree and B+ Tree and has fast insertion and ultra-fast lookup. It can work in schema-less mode, schema-full and a mix of both.

1.7.3 Key-Value Stores

As in relational databases, key-value database are also consist of rows and columns [20]. But this database has only two columns, that are key and value shown is Figure 1.6. This database uses associative array for storage. Every key is associative with values. The key is unique and used to retrieve all the values associated with that key. The values associated can be objects, list or hashes. There is no fixed data model and key-value databases are schema less. These are the simplest NoSQL databases. The query retrieval time is faster than relational databases. Examples of this database are Riak, Redis and Voldemort [20].

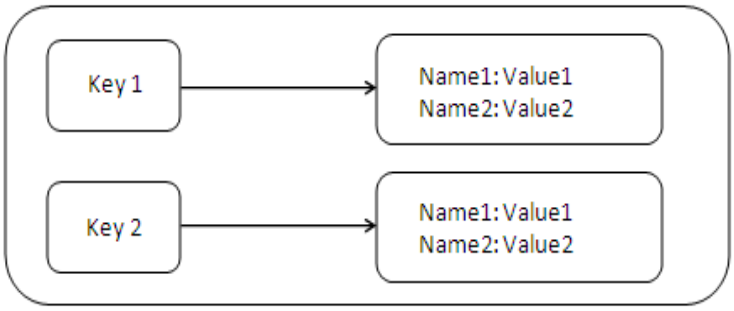

Figure 1.6: Representation of Key-Value Data Store

- **Riak**: Riak is a fully distributed, scalable key-value data store developed by Basho [21]. It was open-sourced under Apache2 license in August, 2009. Riak is written in Erlang. It provides Multi-Version Concurrency Control (MVCC) for updates. In Riak, vector clocks are used as version control mechanism, which are assigned when values are updated. Riak has both the features of key value and document databases. Like document databases, Riak objects are stored and fetched in JSON (JavaScript Object Notation) format and can have

multiple fields and can be grouped into buckets (collections in document databases). But Riak does not have query mechanism of the document database. Only primary key lookup can be done. It also supports links which are relationships between objects. Links simplifies traversal requests and it also supports MapReduce in both Erlang and JavaScript.

- **Redis:** Redis is an open source, BSD licensed key value data store [22]. Redis is written in ANSI C. It stores the data in the form of keys and values, where values associated with keys can be lists, ordered lists and sets. Redis load the whole document into main memory. So, all operations are run in memory but periodically save the data on disk asynchronously. Due to memory operations its performance is high. Redis supports master-slave replication to scale reads and Sharding to distribute writes. This data store also offers shell for simple interaction.

1.7.3 Column Stores

Column oriented stores data tables as sections of columns of data in place of rows of data [23]. These databases have benefits in data warehouses, customer relationship management systems and adhoc query systems. To query over large datasets and to store columns of data collectively, Cassandra and HBase are optimized. The structure of column family data store is shown in Figure 1.7.

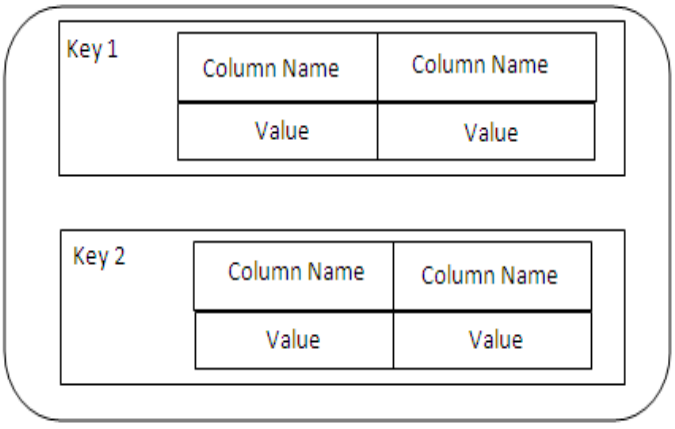

Figure 1.7: Representation of Column Family Data Store

- **Cassandra:** Facebook is one of the most popular social networking websites. It is used by millions of people to connect with friends and relatives. Facebook applications required a data store that can manage billions of writes per day. To meet these requirements, Facebook engineer Prashant Malik and Dynamo's original author Avinash Lakshman have designed a distributed, scalable, column oriented data store that combines Google Big Table's structured data model with Amazon Dynamo's eventually consistent, decentralized storage model. Cassandra is an open source data store with flexible schema [24]. Cassandra is written in JAVA and Thrift API is used to access Cassandra. Other APIs such as REST APIs can also be used. Cassandra can also have super columns with in column families that have related columns. It provides another level of grouping. Cassandra is distributed data store which is composed of lots of database nodes. Data is replicated on these nodes based on eventual-consistency. To achieve scalability only nodes need to be added. All nodes in a cluster are same. Cassandra uses an order hash index which gives benefit of both hash table and B-Tree index. Cassandra supports rich data model and powerful query language.
- **HBase:** HBase is an open-source, distributed column-oriented data store [25]. It is based on HDFS (Hadoop Distributed File System) [26]. It is written in JAVA. It supports ACID transactions. Thrift interface is used to access the HBase and it also supports REST calls. The HBase data model is based on BigTable data model. Each HBase table has a name. Rows have RowID, which is user defined array. Tables are organized into regions, which are stored separately. Each region has a contigious RowID defined by (first row, last row). Columns are organized into column family. HBase can handle both structured and semi structured data naturally. It has dynamic and flexible data model and does not constrain the data types and kind of the data to be stored. For one column it can store integer in one row and string in other row.

2. Literature Survey

2.1 NoSQL Movement

Over 30 years, Relational databases have been used around all the applications. Relational databases are group of tables that consist of rows and columns. These tables are connected through referral keys and also restricted through constraints. To retrieve data sets, queries are triggered on one or more tables. To query more than one table, joins are used to combine the tables. The concept of normalization is used to ensure the consistency of data and remove duplicity. Relational databases provide flexibility, scalability, compatibility, robustness and performance. But even then there require a change in database management paradigm.

Due to increasing number of applications and users, these parameters are also becoming critical. Major consideration is scalability. With more number of applications, workload also increases e.g. web applications. Scalability requirements changes quickly and they grow even faster [27]. Relational databases are though scalable but they are limited to single node scalability. When the scalable capacity of that node is reached, it has to distribute load to other nodes. This is very complex in case of relational databases. Another problem with relation databases was structured format and strict schema. There was so much of data that is unstructured. To store such data NoSQL databases were introduced. Another reason of NoSQL origin is speed of development and emergence of cloud computing.

Traditional Relational databases are not suitable for development if we consider the following key points [28]:

- Developers are using new type of data, i.e. structured, semi-structured, un-structured and large volumes of such data types. Relational databases don't play with such type of data.
- Relational databases work with monolithic server architecture. But now developers are moving towards commodity servers to scale out the architecture.
- Process of development is long using waterfall life cycle in relational databases. But if NoSQL databases are consider, small teams can work on different modules.

NoSQL provides scalability and better performance. That is why relational databases are being replaced by NoSQL databases.

2.2 Document Based Databases

With the requirement of change, NoSQL databases came in picture. One of the popular NoSQL database is document oriented database. In document databases data is considered as "document" [29]. With the high use of internet, storage of large volume of documents became a necessary issue. Documents can be images, un-structured data, and semi-structured data like XML. Document oriented databases provides efficient ways to store and retrieve such documents. Storing documents in document databases provide following advantages.

- Documents are considered as independent unit of database, which leads to better performance of database. It also becomes easier to distribute data among different servers.
- Document database allows storing un-structured data. This reduces the cost of transformation of data to structured form.

2.3 MongoDB

MongoDB is document based open-source NOSQL database [29]. MongoDB stores data as documents in binary representation called BSON (Binary JSON). BSON representation extends popular JSON (java script object notation) to include additional data types such as integer, long and floating point numbers. Popularity of MongoDB among developers and operation professionals is due its agile and scalable approach.

In MongoDB documents represents rows and fields represents columns of the relational database. However, when user data is searched in MongoDB, it tries to locate all the data in a single document whereas in relational database, information searched for a given record is usually located from rows in many tables. It can be said that, in MongoDB data be likely to more localized. Figure 2.1 represents the basic structure of MongoDB database.

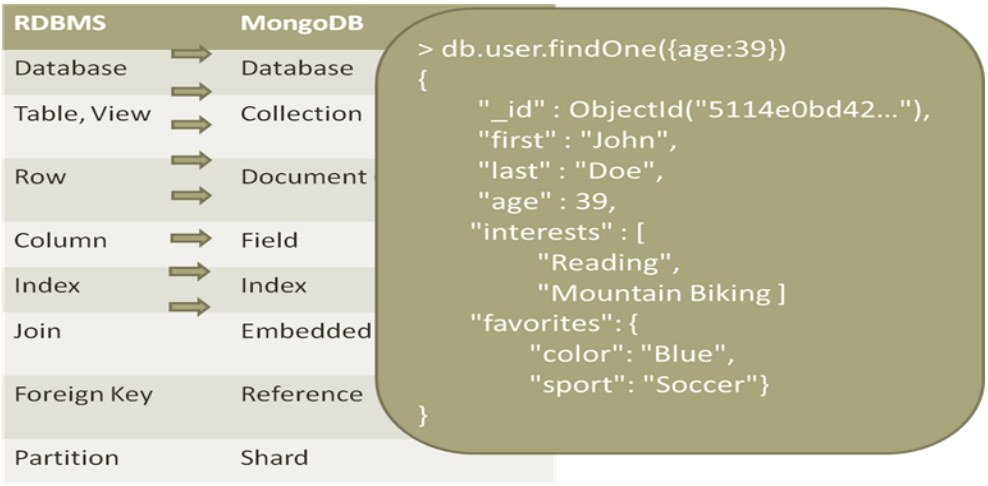

Figure 2.1: MongoDB Structure

2.3.1 Database and Collections

MongoDB databases are stored on MongoDB servers. Each server can handle multiple databases individually [29]. In database, one or more than one collections are stored. Collections are group of documents which can have different structure shown in Figure 2.2.

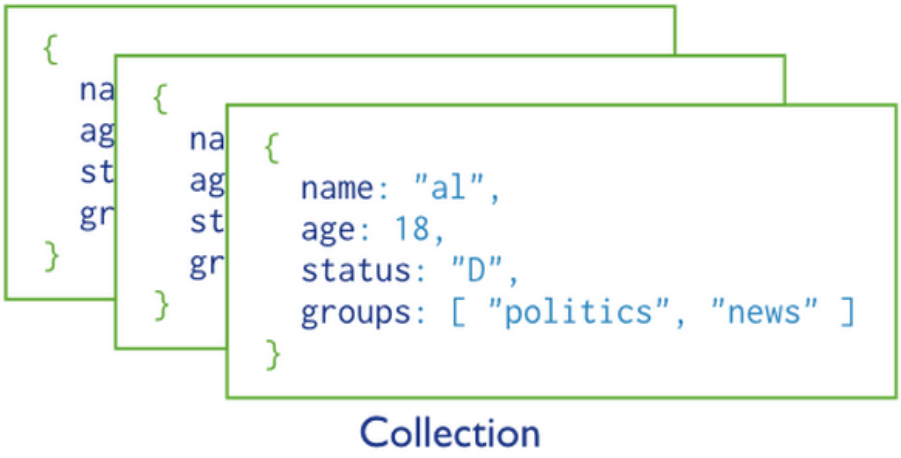

Figure 2.2: Collections in MongoDB

2.3.2 Document

The basic data unit stored in MongoDB is called document [29]. Document is a record that is stored in JSON format, which is composed of field value pairs. When documents

are stored on disk, they stored in BSON format. BSON format is binary representation of JSON format. Relationship among documents is represented in two ways.

- **References:** In references, the relations are maintained by storing the links or references in one document to another [29]. In Figure 2.3, user document is referenced by contact and access documents.

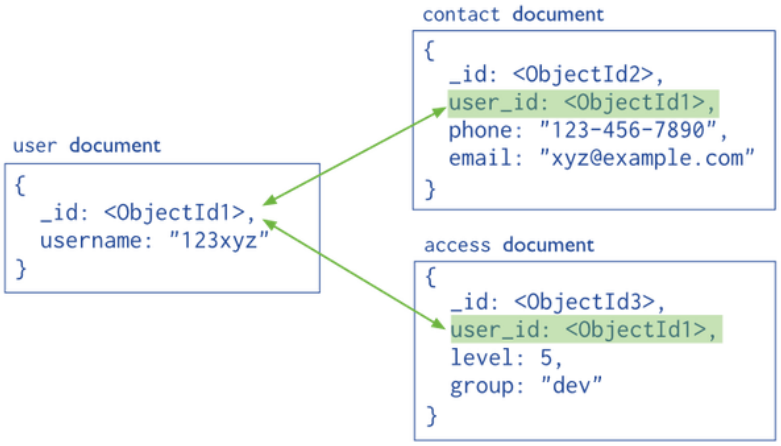

Figure 2.3: Referenced Documents

- **Embedded Documents:** Embedded documents [29] as shown in Figure 2.4 capture the relationships among different databases by storing related documents in same document structure. MongoDB documents have a feature that makes it possible to embed sub documents in document structures in a field or array within a document. This is de-normalized data model which allow applications to retrieve and process data in single database operation only.

Figure 2.4: Embedded Documents

2.3.3 Data Types

MongoDB supports variety of data types used for storing values in documents [30]. The list of data types is given below.

- **Null:** Null represent no value is assigned to the field and also signifies that field does not exist.
- **Boolean:** Boolean data type assigns true or false value to the field.
- **Integer:** MongoDB supports both 32 and 64-bit integers.
- **String:** UTF-8 characters, regular expressions and java script code are represented by using string data type.
- **Object id:** Object id is unique identifier of 12 bytes for each document. These 12 bytes represent the following things.

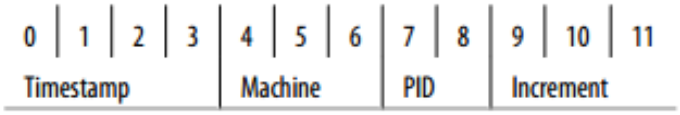

Figure 2.5: Object ID format for Documents [30]

- **Date:** Date stores the milliseconds. Time zone values are not stored in database.
- **Array:** List of values for a particular field is assigned using array data type.

2.3.4 Database Operations

Data structures like JSON and BSON are used to build queries in client libraries. Given example show queries written in JavaScript. These queries are used in the "mongo" shell. There is a mapping chart on the MongoDB website, which maps relational queries to its MongoDB equivalents queries [30].

- **Insert:** The "insert" function is used to insert a new tuple in a database[30]:

  db.myCollection.insert ({Key1: "value1", Key2: "value2"})

  It will insert some values to the database "db" and collection "mydbCollection". If the collection does not exist, it will be created automatically. For every new object, an object id will be associated as key "_id". More complex objects can be inserted using nested data structures.

db.mydbCollection.insert ({name: "Sec Armour", address: {Street: "Sec", City: "Armour"}})

- **Read:** To search and read the stored objects "find" method can be used. To get all the objects in the "myCollection", the following command can be used [30].

db.mydbCollection.find ( )

To search a particular object, limit can set on number of items and only one column can also be extracted.

db. mydbCollection. find ({lastname: "Carter"}, {city: true}). limit (15). skip (10)

The corresponding SQL query for searching a particular record is:

SELECT city FROM Tablename WHERE lastname = 'carter' LIMIT 10, 15

MongoDB allows querying the documents in very simple manners. Keyword "like" is used to search in document for words stored in array. The following query returns all persons who like either mango or banana or both.

db.persons.insert (firstname: "Carter", likes: ['mango', 'banana', 'carrot'])
db.persons.find ($or: [{likes: 'mango'}, {likes: 'banana'}])

It becomes easy to query for data if the document follows right structure. There are no joins in MongoDB. Referenced documents are used for referring a document into another. Similarly they can be dereferenced easily. This can also be done transparently using DBRefs [30]. By using the Object id type, it becomes possible to add references.

db.persons.insert ({name: "John", father: ObjectId("...")})

If the field "father" is accessed then the referenced document is loaded by the MongoDB driver. The used driver like client library must also support this functionality.

- **Update:** To update the existing document values, update command is used. The update command is used. The first argument refers to the field which need to change and second refers to the value to be assigned [30].

db.mydbCollection.update({id: 321}, {$set : {x : 2}})

- **Delete:** To delete a row from a collection 'remove' command is used. The command given below will remove all tuples whose first name is "John" [30].

db.mydbCollection.remove({firstname: "John"});

Table 2.1 SQL to MongoDB Query Mapping [31]

	SQL	**MongoDB**
Create Table	CREATE TABLE persons(id number, name varchar(10))	db.createCollection("person")
Alter Table	ALTER TABLE persons add join_date Datetime	db.persons.update({}, { $set: {join_date: new Date()}})
Drop Table	DROP TABLE persons	db.persons.drop( )
Insert Table	INSERT INTO persons(id, name) values (12, "A")	db.persons.insert({id:12, name: A})
Select Table	SELECT * FROM persons	db.persons.find( )
Delete Records	DELETE FROM persons	db.persons.remove({})

2.4 MongoDB Features

- **Document Model:** MongoDB stores data as document in BSON format. MongoDB documents store all the data for a given record in a single document but in relational databases information for a record is stored in different rows among different tables. So data in MongoDB tends to be more localized [28].
- **Dynamic Schema:** MongoDB documents need not to have the same structure. For example, one document that describes user id and marks can also contain user

address or contact information for another user. Declaration of the structure of documents in MongoDB is not required. Documents are self-describing [32].

- **Collections:** In MongoDB documents are grouped in collections. All documents in a collection are related or have similar task [32].
- **Indexes:** MongoDB uses B-tree indexes for query optimization. Indexes are defined of document fields. MongoDB supports many types of indexes which includes unique, compound, geospatial, TTL, sparse, array and text search indexes [32].
- **Idiomatic Drivers:** MongoDB supports drivers for different languages. Developers can take advantage of this by interacting with the database though native libraries. These libraries are associated with their respective environments and code repositories. This enhances the usability of MongoDB [32].
- **Horizontal Scalability:** Commodity hardware and cloud infrastructure helps to increase the capacity of MongoDB system. This feature is used by developers when data volume and throughput increases [32].

Figure 2.6: Horizontal Scalability

- **High Availability:** Redundant copies of data are stored in the native replication. Any modification to the primary data is automatically replicated to the secondary. Automatic failover to secondary nodes, racks and data centres makes it possible to achieve enterprise grade uptime without custom code and complicated tuning [32].
- **In-Memory Performance:** MongoDB makes extensive use of RAM to speed up the read operation. All the data is read and manipulated in memory mapped files. Reading from memory is extremely faster than reading from disk. This feature provides fast performance and eliminates the need of separate cache layer [32].
- **Flexibility:** Consistency, operation level availability, easy deployment provide tremendous flexibility. This is the reason MongoDB is suitable for many applications across the industries [32].

- **Replication:** To recover from database downtime or system downtime MongoDB maintains multiple copies of data at the different locations which is called replica. A replica set is a fully self-organising and automatic shard that can recover database downtime automatically and so it eliminating the need for manual interruption by network administrators [32]. A replica set consists of multiple replicas as shown in Figure 2.6. Only one replica of database act as a primary replica at a single moment and others act as the secondary. All the operations like read and write are performed on the primary replica by the MongoDB. If in any situation like (e.g. network failure, hardware failure and overheating) the primary data member fails than the secondary data members or replica automatically takes place of primary replica and begins all the process. The replica set of MongoDB is configurable means to increase data durability and availability in case of database downtime (e.g. network failure, hardware failure and overheating) number of replicas in a MongoDB replica set should be more.

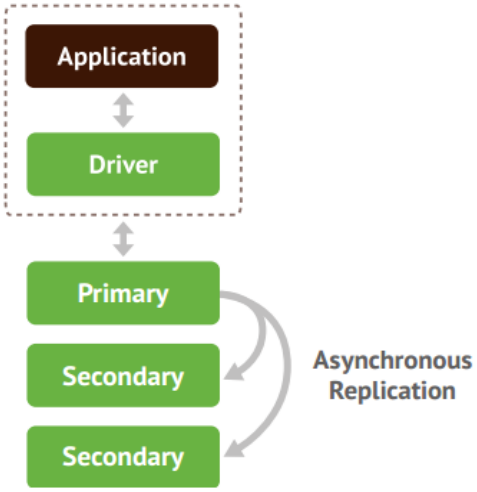

Figure 2.7: Replication of Database [32]

- **Sharding:** MongoDB provides horizontal scale-out for databases using a technique called sharding, which is transparent to applications. The data is distributed across multiple physical partitions by sharding called shards as represented in Figure 2.7. Sharding allows MongoDB deployments to address the

hardware limitations of a single server, such as bottlenecks in RAM or disk I/O, without adding complexity to the application [32].

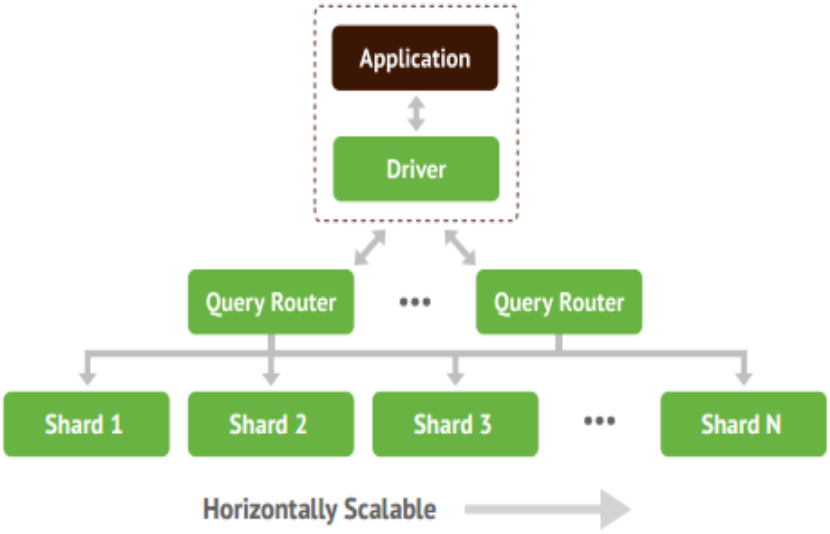

Figure 2.8: Sharding in MongoDB [32]

2.5 Security Issues in MongoDB

MongoDB has many loop-holes in its design. Here the security issues faced by MongoDB are discussed.

- **Hash Injections:** MongoDB suffers from hash attacks [33, 34]. Though hash injections are not dangerous as SQL injection. But they can attempt authentication bypass, DOS attacks and data leakage attacks. MongoDB Rails code is vulnerable to hash injections**.** There is a function param [: value] which does not check that whether it is a string or something else. This flaw is used for hash injections.
- **Clear Text Data:** MongoDB data file are stored as clear text. No encryption mechanism is applied to encrypt the data. If any unauthorized user get access to the database, can gain access over the valuable information very easily. In order to avoid this kind of security fears encryption algorithms must be applied to the sensitive information. Other than that access level for the database must be assigned and file system permissions must be implemented [35].

- **RESTful Interfaces**: MongoDB uses TCP port number 27017 by default. This port is used by binary wire level protocol which has support of all the drivers. So this protocol is good for effective communication. By default MongoDB uses port 28017 and this protocol for performing replication. This HTTP server provides many management level services, and it can also be configured using rest=true in configuration file to provide a Restful interface to the database. The internal HTTP server does not provide any support for TLS or SSL in any method, and wire-level protocol is also not provides encryption [35].
- **Java Script Injection Attacks***:* JavaScript is a scripting language used by MongoDB functions. Short java-script scripts are available as internal command for developers. Java-script functions can be stored in the database in db.system.js collection. This collection is available to the database users. So there is a high probability for attackers to inject their malicious scripts to this collection. JavaScript uses $where clause to evaluate records in collection. The result of this clause is read only, so any changes in database are not possible. But if $where clause is used without sanitizing the input values, chances of injection becomes more [36].
- **Authentication:** MongoDB supports authentication only in Standalone mode. There is no authentication mechanism when used in shared mode. In replica mode authentication is done though pre shared secret key which is provided in configuration file by the key file parameter [36].

  Administrator can decide through which password replica server will be access, and is it can be same for all replica servers. The user is configured by connecting its database in the replica system for client connections, and also the master server must have all the updates. db.addUser () method is used to add new user to the database and can have all the necessary privileges.

  The password storing mechanism is also weak in MongoDB. The password storing method is MD5 hash of the string "username : mongo : password" , and it can be easily accessible from the admin data-files, which means that if attacker get that MD5 hash than only he/she can get the actual password for the whole

database. Nonce used for encryption is static and remains same for all users. This makes the encryption weak [37].

- **Fast Password Hashing:** MD5 is a fast password hashing algorithm. This means brute forcing passwords will also become fast. Now a days processing becomes faster due to GPU processors. Fast hashing algorithms are not suitable for password hash calculation [38]. Algorithms like bcrypt or scrypt should be implemented which are slow in processing. The will prevent passwords from brute force attacks.
- **Salt Reuse**: The salt used while calculating hash of the password is "mongo". This remains same for all the users [38]. This causes a problem of hash collisions, which can add benefit to the hackers.

2.6 Overview Of Password Hashing

Hashing, in cryptography, is a one-way operation which transforms a stream of data into a more compressed form called a message digest [39]. The operation should not be invertible, meaning that recovering the original data stream from the message digest should not be possible. All the message digests or hash values generated by a given hash function have the same size no matter what the size of the input value is.

2.6.1 One Way Functions

One way functions are the mathematical functions which are computationally easy to calculate in one direction and complex in the other direction (invert direction) [39]. It might be possible the computation in forward direction is done in seconds and for backward computation it takes months or years.

A function f: $\{0, 1\}^* \rightarrow \{0, 1\}^*$ is called one way function if

- There exists a polynomial time algorithm A, so that it computes f(x)

$$A(x) = f(x)$$

- For every randomized algorithm A that runs in polynomial time, every polynomial p(n), and all sufficiently large n

$$Pr\left[f\left(A(f(x))\right) = f(x)\right] < \frac{1}{p\,(n)}$$

2.6.2 Cryptographic Hash Functions

A cryptographic hash function is a one way function that takes input of any length and converts it into a fixed length output called message digest, hash, checksum or digital finger print [40].

Formally hash function is defined as

$$F: M \rightarrow H$$

Where F is cryptographic hash function, takes message M as input and convert it into hash H of the message. Here M is infinite non-empty set and H is finite not empty set.

Properties of Good Cryptographic Hash [40]:

- **Preimage Resistant:** For all the calculated hashes H, it must be computationally difficult to find the original message, i.e. , M.
- **2nd Preimage Resistant:** It must be difficult to find another message M' that has same hash as of M, i.e. , F(M) = F(M')
- **Collision Resistant:** It must be difficult to find two inputs M and M' that results into same hash value.

2.7 MD5 Cryptographic Function

The MD5 message digest algorithm was designed by Ron Rivest in 1991 [41]. This algorithm was designed to replace the earlier hash algorithm MD4. MD5 algorithm is widely used to calculate cryptographic hash, which produce 128 bits (16 Byte) hash value. This value is represented in 32 digit hexadecimal number.

The algorithm takes input message of any length and produces output of 128-bit "fingerprint" or "message digest" of the input. The blocks of 512 bits are made. Each block is than processed in 4 rounds. In every round each block is further turned in 16 sub-blocks of 32 bit each. [41] The message block is padded if it is not up to 512-bits, as shown in Figure 2.8.

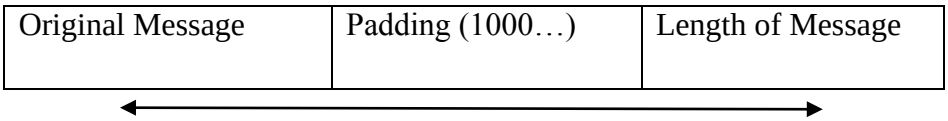

Figure 2.9: MD5 Message Format

2.8 Weaknesses of MD5

The password is stored in the form of a MD5 hash of the string "username : mongo : password" and it can be easily accessible from the admin data-files which means that if attacker get that MD5 hash than only he/she can get the actual password for the whole database [42]. Nonce used for encryption is static and remains same for all users. This makes the encryption weak.

- **Non-Variant Nonce:** Nonce "mongo" is always same for every username and password. If two users choose same credentials, they will get same hash. The non-variant nonce makes the hash insecure.
- **Collision Irresistant:** MD5 algorithm is found prone to collisions. To be a good cryptographic hash function, it should be collision resistant.
- **Dictionary Attacks:** In Dictionary attacks [42], all the possible username and password are tested. The attacker calculates hashes from the dictionary and performs binary search on compromise hashes. To speed up this process, attacker pre-calculates hashes for future reference.
- **Rainbow Attacks:** Rainbow table are pre-calculated tables made up of hash chains. These tables are used for reversing cryptographic hash function to find the plain text. Rainbow tables are comparatively more efficient than hash tables. They are slightly slow in terms of time for lookups but are storage efficient [42]. Rainbow tables use hash function as well as reduction function. Reduction functions are the reverse cryptographic hash functions that convert the hashed values into plain text.The plain text converted is not same as the plain text for which the hash value is generated. By changing the hash function with the reduction function, different hash values and passwords are generated. In the process of table creation, only first and last plain text created is stored in table. To decode a hashed password, we first process the hashed password through

reduction functions until we find a match to a chain's end point. We then take that chain's corresponding start point and regenerate the hash chain and find the original plaintext to the hashed password. Rainbow tables are very easily available on the web now. Many password cracking tools are available that uses rainbow tables, e.g., OphCrack [43]. Though, the use rainbow tables do not provide 100 % surety of successful password cracking. The bigger the character set is, longer will be hash chain length and bigger will be the rainbow table.

3. Problem Statement

With the rise of data in terms of volume, velocity and variety, traditional databases are not efficient to handle such data. Data is available in structured, unstructured and semi structured form. Relational databases cannot handle such data due to strict constraints and fixed schema. To overcome this problem, a new form of databases was developed and that are Non-Relational Databases. NoSQL databases are comparatively new and there are many improvements that are required to bring in these databases. In this report, we are discussing issues related to document based database MongoDB. MongoDB is 10Gen product which is used to store the documents in schema less form. This is a new type of paradigm; major consideration of the company was to improve its performance in terms of time and queries. Security was not the primary concern of 10Gen. By studying and analyzing the database, it was found that security of database needs serious improvements.

3.1 Research Gaps

From literature survey, the following research gaps were analyzed.

i) MongoDB database functions for adding user, changing user password and removing user were not authenticating user before performing action.

ii) MongoDB database uses default ports, the data can be visible from these ports. To provide limited access to these ports network security rules are required to define.

iii) The password for user is hashed using md5 algorithm. For calculating hash value of user a time in-variant nonce value has been used. Time in-variant nonce provides a way to easy crack for hackers.

iv) Time in-variant nonce leads to the problem of hash collision. Same user name and password leads to same hash value.

3.2 Objectives

To fulfill above research gaps, these objectives have been made.

i) To study the properties of Relational and Non-Relational databases and various classes of NoSQL databases.

ii) To install and configure the Document based database MongoDB and execute the query system.

iii) To identify the security loopholes in the document based database MongoDB.

iv) To provide proper and efficient method for securing the environment in which MongoDB is being used.

v) To compare the proposed methods with existing methods for various MongoDB operations.

3.3 Methodology

To achieve all the objectives discussed in section 3.2, the following methodology has been proposed.

i) To familiarize with the query system of MongoDB, different databases and collections has been made for query processing.

ii) To identify the security loopholes in system, unauthorized ways of entering in databases has been used like performing injection attacks, server side java script inclusion etc.

iii) To secure the periphery in which MongoDB is being used, different firewall rules has been written for windows and linux operating system.

iv) To improve the user authentication, function definitions has been changed for remove user and change user password.

v) To improve the security of stored password hash, a time variant nonce is introduced that will provide a unique hash for each user and that will be difficult to break.

4. Implementation

4.1 System Configuration

The system environment for MongoDB database is shown in Table 4.1.

Table 4.1: System Environment

SYSTEM	DELL Inspiron 15R
SYSTEM TYPE	64 BIT
RAM	8GB
HARD DISK	1 TB
OPERATING SYSTEM	WINDOWS 8 and UBUNTU 12.04
PROCESSOR	INTEL CORE™ i7
MONGODB	2.6.3

4.2 Configuring Document Database MongoDB

To configure MongoDB database, the following steps are taken.

Step1: Run the command prompt in administrator mode and run the following commands:

```
cd\
move C:\mongodb-win32 C:\mongodb
```

Step2: MongoDB requires a data folder to store its files. The default location for the MongoDB data directory is C:\data\db. Create this folder using following command:

```
md data
md data\db
```

Step3: Add the data folder path in configuration file that is mongo.conf.

Step4: Start MongoDB demon from C:\mongodb\bin\mongod.exe. Figure 4.1 represent the MongoDB demon.

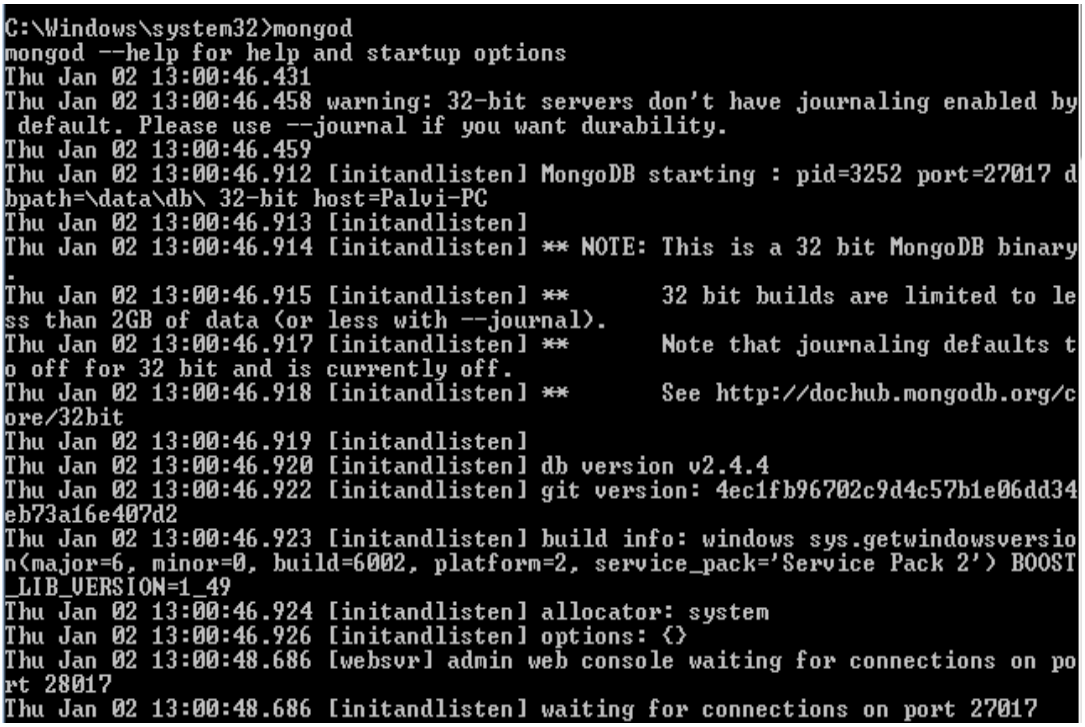

Figure 4.1: Mongodb Demon

Step5: Start MongoDB shell from C:\mongodb\bin\mongo.exe. Figure 4.2 represents how to start the MongoDB shell by command.

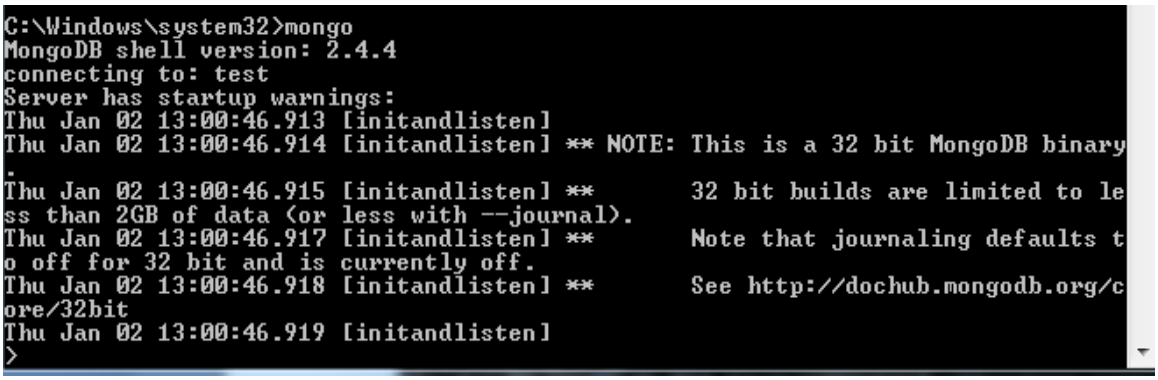

Figure 4.2: Mongodb Shell

4.3 Securing the Network for MongoDB Database

First step to secure the data in the database and its unethical use, is to secure the periphery in which database has been configured. To secure the network periphery windows as well as linux operating systems has provided firewalls. Firewalls are used to secure the network using set of rules that can be manually configured. In the following

sections rules are configured in windows as well as linux firewall for MongoDB database.

4.3.1 Configuring Windows Firewall (netsh) for MongoDB

netsh is a utility that provide methods for managing windows firewall. By using this, administrator can control which hosts can connect to the system [35]. By limiting the host's connections, risk of exposure also limits. Here we write some rules for mongodb. Windows Firewall processes rules in an ordered determined by rule type, and parsed in the following order:

1. Windows Service Hardening
2. Connection Security Rules
3. Authentication Bypass Rules
4. Block Rules
5. Allow Rules
6. Default Rules

Default policy of windows firewall allows all outgoing connections and blocks all incoming connections. MongoDB uses default ports, so networking rules can be configured by allowing only required communication between application and mongod.exe and mongos.exe. Following rules have been identified and created which explicitly allow traffic from some specific address and ports.

Rule 1: Traffic to and from mongod.exe Instances

This rule is applicable for mongd.exe instances which are either standalone or a part of replica set. This rule allows traffic coming to mongod.exe instance on port number 27017 from any application server as shown in Figure 4.3.

Figure 4.3 Rule to allow traffic on 27017

We can also set the rule to allow network access for an entire application rather than to a specific port, as in the Figure 4.4.

```
C:\Windows\system32>netsh advfirewall firewall add rule name="Allowing mongod" d
ir=in action=allow program=" C:\mongodb\bin\mongod.exe"
Ok.
```

Figure 4.4: Rule to allow access mongod.exe

Similarly we can define rule to allow all access for a mongos.exe server, with the command shown in Figure 4.5.

```
C:\Windows\system32>netsh advfirewall firewall add rule name="Allowing mongos" d
ir=in action=allow program=" C:\mongodb\bin\mongos.exe"
Ok.
```

Figure 4.5: Rule to allow access mongos.exe

Rule 2: Traffic to and from mongos.exe Instances

Mongo.exe instance provide query routing for sharded clusters. Clients connect to this instance that behaves like mongod.exe to client. We can use the same Windows Firewall command to allow traffic to and from these instances as you would from the mongod.exe instances that are members of the replica set shown in Figure 4.6.

```
C:\Windows\system32>netsh advfirewall firewall add rule name="Open mongod shard
port 27018" dir=in action=allow protocol=TCP localport=27018
Ok.
```

Figure 4.6: Rule to allow access on sharded cluster port 27018

Rule 3: Provide Access for Monitoring Systems

If monitoring system needs access to HTTP interface, the rule can be added to firewall as shown in Figure 4.7.

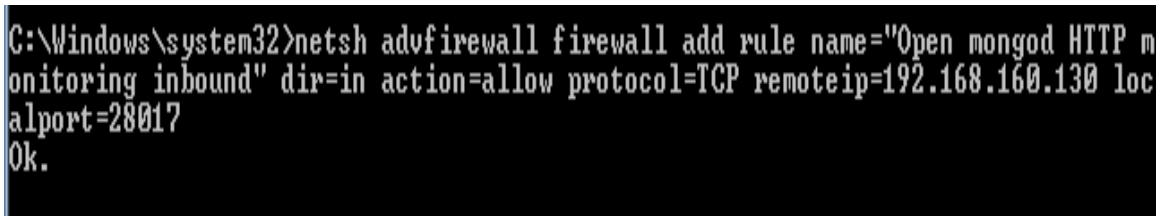

Figure 4.7: Rule to allow access HTTP interface

4.3.2 Configuring Linux iptables Firewall for MongoDB

On contemporary Linux systems, the iptables program provides methods for managing the Linux Kernel's net filter or network packet filtering capabilities [35]. These firewall rules make it possible for administrators to control what hosts can connect to the system, and limit risk exposure by limiting the hosts that can connect to a system. Rules in iptables configurations fall into chains, which describe the process for filtering and processing specific streams of traffic. Chains have an order, and packets must pass through earlier rules in a chain to reach later rules. This document addresses only the following two chains:

INPUT Controls all incoming traffic.
OUTPUT Controls all outgoing traffic.

Given the default ports of all MongoDB processes, you must configure networking rules that permit only required communication between your application and the appropriate mongod and mongos instances. The default policy of iptables is to allow all connections and traffic unless explicitly disabled. The configuration changes outlined in this document will create rules that explicitly allow traffic from specific addresses and on specific ports, using a default policy that drops all traffic that is not explicitly allowed. When you have properly configured your iptables rules to allow only the traffic that you want to permit, you can Change Default Policy to DROP.

Rule 1: Traffic to and from mongod Instances

This pattern is applicable to all mongod instances running as standalone instances or as part of a *replica set*. The goal of this pattern is to explicitly allow traffic to the mongod instance from the application server.

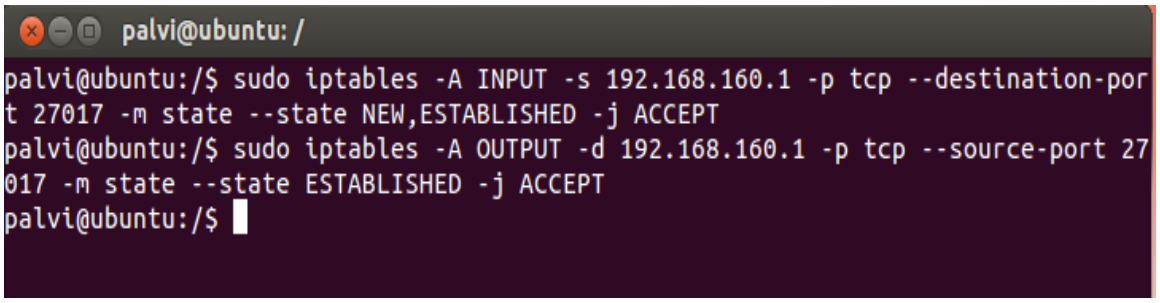

Figure 4.8: Rule to allow traffic to and from mongod

The first rule allows all incoming traffic from 192.168.160.1 on port 27017, which allows the application server to connect to the mongod instance. The second rule, allows outgoing traffic from the mongod to reach the application server. To view all the rules use command show in Figure 4.9.

Figure 4.9: List firewall rules

Rule 2: Provide Access for Monitoring Systems

If your monitoring system needs access to the HTTP interface, insert the rule defined in Figure 4.10.

Figure 4.10: Rule to allow access to HTTP interface

4.4 Attack Scenario

MongoDB provided web interface to monitor the activities. The Rest settings for mongod provide fully administrative mode to interact with MongoDB. On this interface attacker can submit malicious scripts and read sensitive information. As defined above MongoDB suffers from java script injection vulnerabilities. Along with vulnerabilities in eval function calls in node.js, JavaScript functions containing user specified parameters are also vulnerable.

In this attack scenario, victim has enabled Rest interface. The default port to run this interface is 28017. Attacker get access to the victim machine through port addresses 28017. Attacker sends malicious script to the mongodb site. When that script is executed the reply goes to hacker machine through admin browser.

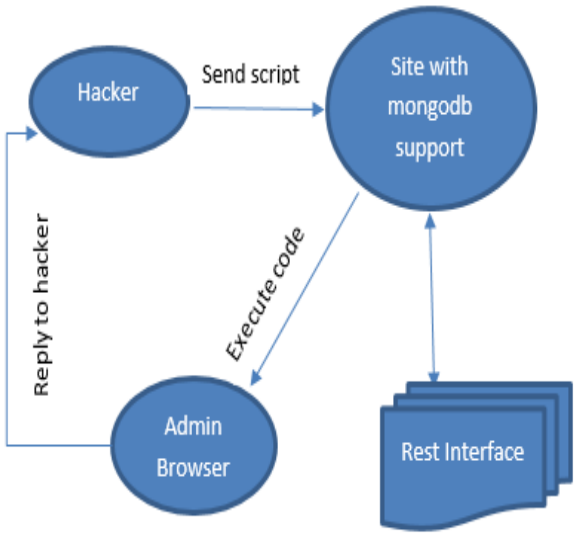

Figure 4.11: Flow diagram of attack on rest interface

Any user with ip address 192.168.160.129 is using mongodb with rest interface. Through Rest interface user data can be read. Attacker can read any information from database. As shown in Figure 4.11, attacker is reading user information from admin database. It displays all the information from user like user name, password and role of user.

Figure 4.12: User tables of MongoDB

From users table encrypted password for particular user is extracted shown in Figure 4.12. To calculate hash MongoDB use one way hash cryptographic algorithm md5. Md5 hash calculation id done by hex_md5 function which takes {username: mongo: password}. "mongo" is used as nonce here which is static. So if two users have same username and password, they will get same cryptographic hash. So problem of hash confliction arise here. This helps hackers to easily calculate password for particular user. Moreover md5 is easily cracked able by brute force attacks and rainbow tables.

4.5 Security loopholes in MongoDB and Solution

Security loopholes are basically the ways from where the attacker can penetrate into the system. Majority of loopholes are created due to poor programming issues. MongoDB java script functions also have some issues that may allow unauthorized user to execute function. These functions are described below.

4.5.1 Change User Password

MongoDB function changeUserPassword ( ) is intended to change the password of the existing user. This function takes two parameters i.e. "username" and "new password" shown in Figure 4.13.

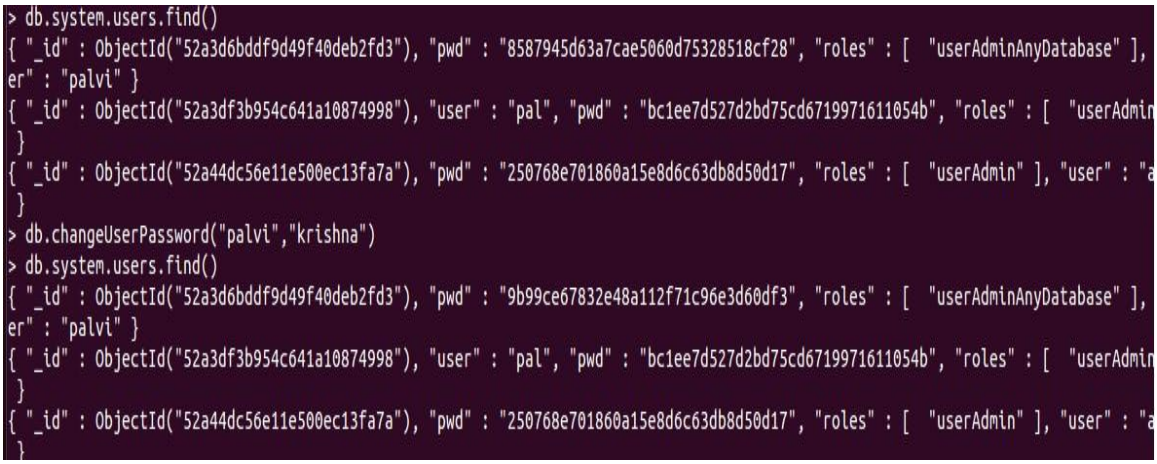

Figure 4.13: changeUserPassword function

Problem with this function is that if any unauthenticated user gets access to database, without knowing password of user he can change it. This can be done by any insider in the organisation too. By changing the definition of this function shown in Figure 4.14, we

can stop unauthorized access to some extent. If changeUserPassword ( ) function asks for old password and matches with the old password hash, risk of compromise will be reduced. Improved version of this function asks for old password and calculates its hash. Calculated hash is matched with old hash stored in database for the user. If both the hashes match then only password will be changed.

Figure 4.14: Modified changeUserPassword function

This function now matches the hash of old password. It will be matched with the database entry, if matched then will change otherwise shows error.

4.5.2 Remove User

Remove user password function take only one argument that is “user name”. By knowing only name of the user, it will remove the user. Any other user or attacker can remove that user and pretend to be same as old user. This problem can be solved by checking that whether an authorised person is performing the operation or not. This function can be improved by introducing a new argument “password”. Before performing the operation, password should be verified. If the provided password is correct, then only user should be removed. The modified function definition is shown in Figure 4.15.

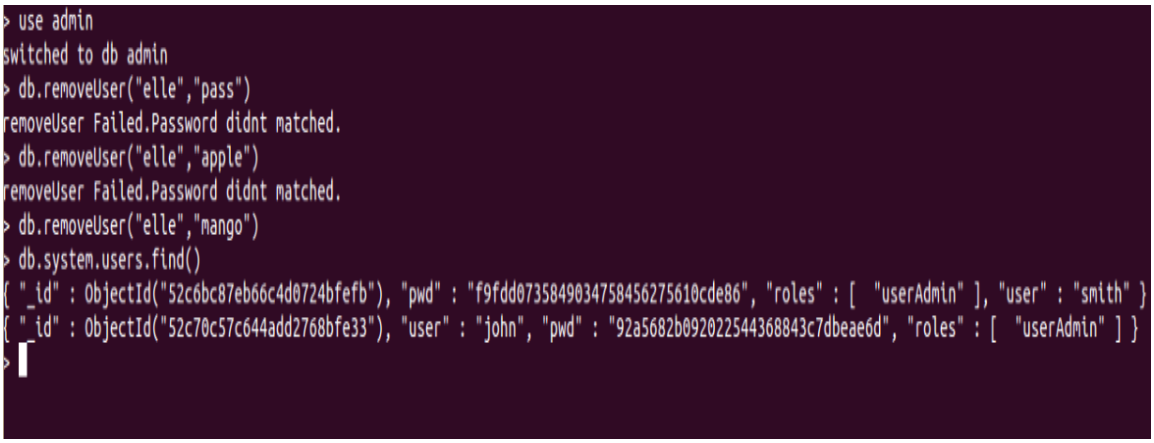

Figure 4.15: Removes user function and modification

4.6 Time Variant Nonce Addition to MD5 Hash Calculation of Password

This section explains the existing password methodology and new purposed method for password hash.

4.6.1 Existing Password Hashing Mechanism

The existing system for password hashing use static nonce value. Along with user name and password, nonce is passed to md5 function for hash calculation. The prototype for hash calculation is shown below.

hash = md5 (user_name: mongo: password)

To overcome the security breaches of MongoDB passwords, an improvement is required for hash calculation. The solution is proposed in section 4.5.2. The proposed solution calculates the randomized and unpredictable nonce using a secure cryptographic pseudo random number generator. This nonce is then prepended to the password. Password prepended with nonce value is parsed for hashing through MD5 algorithm. To provide complexity to the hash value, this process is repeated N times. The output of all the rounds is then XORed and final hash is calculated. This hash value is stored to the database along with the nonce value. Nonce of the hashed value is stored in different

location other than hash value location. This is done to add more security feature to the database.

4.6.2 Proposed Solution

Usually Attacker cracks the simple hashes using dictionary and rainbow table attacks. The proposed method uses Randomized hashing approach to provide a better solution. To randomize the hash value the concept of nonce is introduced. Nonce is a value that is generated randomly and concatenated with password before calculating its hash.

Nonce is generated using Cryptographically Secure Pseudo-Random Number Generator. CSPRNG provide high level of randomness and are completely unpredictable.

Properties of nonce:

- Nonce should be unique for every user
- Length of the nonce should be long
- Never reuse once

Existing password hashing algorithm use static and reusable nonce. The hashing algorithm works as follow in MongoDB. It takes three arguments user_name, nonce and password which are parsed to MD5 function to calculate the hash value.

4.6.3 Proposed Methodology

Step1: CSPRNG ( ) is producing nonce value which is random and unique
nonce ← CSPRNG( )
Step2:nonce is concatenated with password
pass1 ← username + nonce + password
Step3: hash is calculated using md5 hashing algorithm
hash ← md5 (pass1)
Step4: Store nonce and hash in database

The flow diagram shown in Figure 4.16 describes the working of algorithm. This algorithm works for N number of rounds. Each round takes hashed value of previous

round and password. The output of all the rounds is XORed and final hash value is calculated.

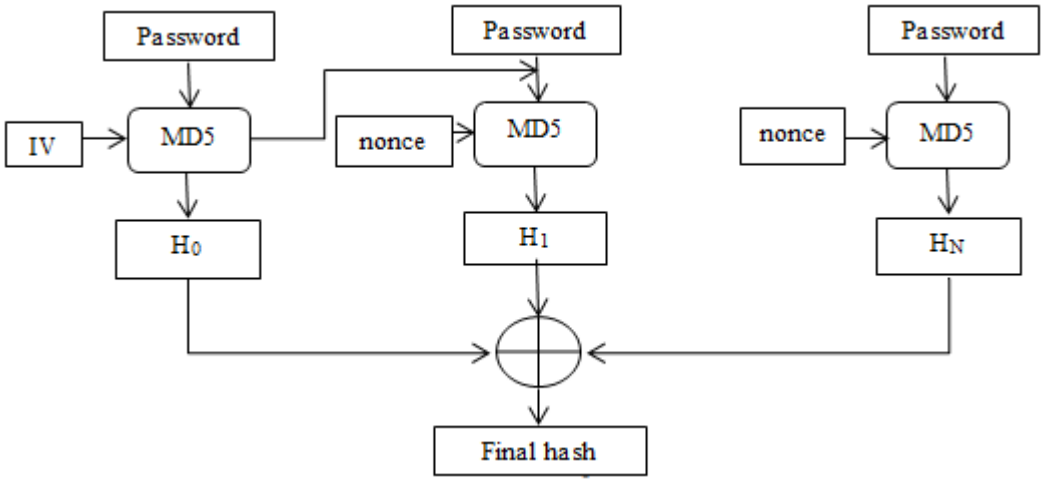

Figure 4.16: Improved MD5 algorithms in MongoDB

Figure 4.16 shows the flow of algorithm, each round will calculate hash value of password. This process will be repeated for N number of rounds. The description of each round is given below.

H_0 = MD5 (Password, IV)

H_1 = MD5 (Password, H_0, nonce)

H_N = MD5 (Password, H_{N-1}, nonce)

Final hash value = H_0 ^ H1^……..^ H_N

H: Hash Value

N: Number of iterations

^: XOR operation

IV: Initialization Vector

5. Conclusion and Future Scope

5.1 Conclusion

In this thesis, detailed study of document database MongoDB has been done. Security is not considered as major issue for MongoDB till now by 10gen. This research work reviews the security issues present in MongoDB. Although mongodb provides flexibility, scalability and good performance, but there are many security issues like authentication, authorization, weak password hashes and weak encryption at both client and server end. These can lead to serious attacks like denial of service, query injection and unauthorized user access.

- In this research work, a solution is proposed for securing the environment in of MongoDB database. Firewall rules are configured for windows and linux operating system.
- Another issues is lack of authentication while using functions like change user password and remove user. These functions do not check that whether valid user is performing the function or not. To secure these functions the function definition is changed. Now these functions check that valid user is performing the operation or not.
- In MongoDB, hash of the password is calculated using md5 algorithm. It uses a static nonce value for all the users. Due to this, there was a problem of hash collision and crackable password hash. In this research work, the problem is resolved using time variant nonce value. The variable nonce is generated using pseudo random number generator. Using pseudo random number generator, randomized and unpredictable nonce value is generated. The generated hash value is more secure and un-crackable. The use of Nonce also saves passwords from dictionary and rainbow table attacks. The problem of hash collision is also resolved due to randomized nonce value. Therefore the proposed methodology serves a better option to resolve MongoDB Password Hashing problem.

5.2 Future Scope

The proposed system works on the security of user passwords. Calculated hash value can be generated using more secure algorithm. This work can be extended to enhance the security of hash of the password. To provide more security, user data files can be encrypted and the channel through which different shards communicate with each other can be secured. In future, this thesis work can be enhanced to add more security in MongoDB.

References

[1] O. Sutinen, "NoSQL – Factors Supporting the Adoption of Non-Relational Databases", M.Sc thesis, Department of Computer Science, Tampere University, Finland, 2010.

[2] M. Widenius and D. Axmark, "MySQL Introduction", *Linux Journal*, vol. 67, 1999.

[3] H.F. Korth, and A. Silberschatz, "*Database system concepts*", vol. 6., New York: McGraw-Hill, 1991.

[4] L. Frank, "Countermeasures against consistency anomalies in distributed integrated databases with relaxed ACID properties", *in proceedings of IEEE conference on Innovations in Information Technology*, 2011.

[5] C. Strozzi, "Nosql: A Non-Sql Relational Database Management System", [Online] Available: http://www.strozzi.it/cgibin/ CSA/tw7/I/en US/nosql/Home-Page, 2013.

[6] CAP Theorem, [Online] Available: http://en .wikipedia .org/wiki/CAP_theorem.

[7] Khachana, R. Tshoganetso, A. James, and R. Iqbal. "Relaxation of ACID properties in AuTrA, The adaptive user-defined transaction relaxing approach", *Future Generation Computer Systems*, vol. 27, pp. 58-66, 2011.

[8] Jeremy Zawodny, "Lessons Learned from Migrating 2+ Billion Documents at Craigslist"
[Online] Available: http://www.10gen.com/presentation/mongosf2011/craigslist.

[9] J. Han, E. Haihong, D. Guan and D. Jian, "Survey on NoSQL Database", *in proceedings of 6th International Conference on Pervasive Computing and Applications (ICPCA)*, pp. 363-366, 2011.

[10] Why NoSQL? [Online] Available: http://info.couchbase.com/whyNoSQL-LP.html.

[11] K. Kaur and R. Rani, "Modeling and Querying Data in NoSQL Databases", *in proceedings of IEEE International Conference on Big Data*, pp. 1-7, Silicon Valley, CA, October 2013.

[12] Nosql Conference. [Online] Available: https://nosqleast.com/2009, 2009.

[13] Javascript Object Notation (JSON). [Online] Available: http://www.json.org.

[14] Couchdb Website. [Online] Available: http://couchdb.apache.org/

[15] MongoDB. [Online] Available: http://www.mongodb.org/

[16] K. Seguin, "The Little MongoDB Book" [Online] Available: http://github.com/Karlseguin/the-little-mongodb-book.

[17] I. Robinson, J. Webber and E. Eifrem, "Graph Databases", O'Reilly Media, June 2013.

[18] E. Eifrem., Neo4j homepage. [Online] Available: http://www.neo4j.org/

[19] Orientdb wiki. [Online] Available: http://www.orientechnologies.com/orientdb/

[20] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall and W. Vogels, "Dynamo: Amazon's Highly available Key-Value store", *in ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 205-220, ACM, 2007.

[21] E. Redmond, and J.R. Wilson, "Seven databases in seven weeks: a guide to modern databases and the NoSQL movement", Pragmatic Bookshelf, 2012.

[22] Redis homepage.[Online] Available: http://redis.io/

[23] P. Raichand and R. Rani, "A Short Survey of Data Compression Techniques for Column Oriented Databases", *Journal of Global Research in Computer Science*, vol. 4, no. 7, pp. 43-46, July 2013.

[24] Cassandra homepage. [Online] Available: http://cassandra.apache.org

[25] HBase homepage. [Online] Available: http://hbase.apache.org/

[26] P. Raichand and R. Rani, "Query Execution and Effect of Compression on NoSQL Column Oriented Data Store using Hadoop and HBase", *International Journal of Scientific and Engineering Research*, vol. 4, no. 9, September 2013.

[27] R. Cattell, "Scalable Sql and Nosql Data Stores" SIGMOD Rec., vol.39, no. 4, pp. 12–27, 2011.

[28] Nosql Conference. [Online] Available: https://nosqleast.com/2009, 2009.

[29] MongoDB-Manual, MongoDB, Inc., July 2013.

[30] R. Arora and R. Rani, "Modeling and Querying Data in MongoDB", *International Journal of Scientific and Engineering Research*, vol. 4, no. 7, pp. 141-144, July 2013.

[31] Getting Started with the Mongo Shell, [Online] Available: http://docs.mongodb.org/manual/tutorial/getting-started-with-themongo-shell.

[32] K. Chodorow and M. Dirolf, "MongoDB: The Definitive Guide", O'Reilly Media, 2010.

[33] OWASP, "Testing for NoSQL Injection", [Online] Available: https://www.owasp.org/index.php.

[34] B. Sullivan, "Server-Side JavaScript Injection", Adobe Secure Software Engineering Team, 2011.

[35] MongoDB Security Guide, MongoDB, Inc., September 2013.

[36] L. Okman, N. Gal-Oz, Y. Gonen, E. Gudesand and J. Abramov, "Security Issues in NoSQL Databases", *in proceedings of IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications,* 2011.

[37] NSIT, "Secure Hashing", [Online] Available:http://csrc.nist.gov/groups/ST/toolkit/secure_hasing.html.

[38] Bug Tracker for MongoDB [Online] Available: https://jira.mongodb.org/browse/SECURITY.

[39] William Stallings, "Cryptography and Network Security, Principles and Practice" 2nd Ed, Prentice Hall, 1999.

[40] B. Preneel, "Analysis and Design of Cryptographic Hash Functions", Doctoral Dissertation, 2003.

[41] R. Rivest, "The MD5 Message-Digest Algorithm", RFC 1321, 37, 1992.

[42] H. Mirvaziri, K. Jumari, M. Ismail and Z.M. Hanapi, "A New Hash Function Based on Combination of Existing Digest Algorithms", *in proceedings of 5thStudent Conference on Research and Development – SCORED 2007*, Malaysia, 2007.

[43] X. Zheng and J. Jin, "Research for the Application and Safety of MD5 Algorithm in Password Authentication", *in proceedings of 9thInternational Conference on Fuzzy Systems and Knowledge Discovery*, 2012.

List of Publications

1. P. Aggarwal and R. Rani, "Security Issues and User Authentication in MongoDB", *in proceedings of Elsevier Second International Conference on Emerging Research in Computing, Information, Communication and Applications*, NMIT Bangalore, August 1-2, 2014.

   [Accepted]

2. P. Aggarwal and R. Rani, "Improved Password Hashing in MongoDB", *in proceedings of 5th International Conference CONFLUENCE*, 2014.

   [Communicated]