IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-10, Oct 2018 E-ISSN: 2347-2693

COMPARISON OF ENCRYPTION ALGORITHMS ON NoSQL DATABASES

P. Rajesh Kannan[1*], R. Mala[2]

[1]Dept. Of Computer Science, MarudhuPandiyar College,Thanjavur, Tamilnadu, India.
[2]Dept. Of Computer Science, Alagappa University College of Arts and Science, Paramakudi, India

[*]*Corresponding Author: rajeshapril04@gmail.com*

Available online at: www.ijcseonline.org

Accepted: 20/Oct/2018, Published: 31/Oct/2018

Abstract— Security has become one of the key features of data transmission on large databases. Sensible data that are available in the form of documents or unstructured format must be shared among the users in a confidential manner. NoSQL databases are nowadays popular in handling the unstructured data that are available as open source databases such as MongoDB, Cassandra, Redis etc. This paper make a detailed study on the encryption techniques of NoSQL databases especially MongoDB which becomes popular in data management. Since encryption features are not applied on handling the data in MongoDB, the various encryption techniques proposed on securing the sensitive data at rest and at transit are compared based on different encryption algorithms.

Keywords— Data Security, NoSQL, MongoDB Encryption, encryption at rest, encryption at transit.

I. INTRODUCTION

Secure handling of private data becomes very important nowadays on the databases. Relational databases are securing their data with some additional efforts provided along with the database. Data security is maintained by the Data Base Administrators (DBA) depending on the level of security needed by the concern. With the huge storage of unstructured data on various mediums have become difficult to handle with the help of relational databases like SQL databases. In order to handle these kinds of un structured data, NoSQL(Not only SQL) databases are available as open source databases like MongoDB, Cassandra, Redis, Hypertable, CouchDb etc. Most of these open source databases are not built with complete data security. In this paper, the open source databases are studied on the basis of encryption techniques used by them to handle the security flaws that are prevalent in data security. The existing measures of data security are compared and the proposed techniques are suggested to mitigate these data security problems on the data at rest and data in transit.

Hariharan et al. [1] discusses the database encryption techniques on the databases. This author surveyed different encryption methods like A Database Record Encryption Scheme Using the RSA Public Key Cryptosystem and its Master Keys, Chip-Secured Data Access: Confidential Data on Untrusted Servers, Fast, Secure Encryption for Indexing in a Column-Oriented DBMS, The Transport Layer Security (TLS) Protocol Version 1.2 etc. These suggested techniques differ based on their performance, access time and key management.

Jef Van Loon et al. [2] discussed the engineering aspects of security and present some basic measures to protect databases from unauthorized access. In order to mitigate the dominant security issues like legal and ethical issues, policy issues, organizational and system related issues, there is a serious need for database security. They also emphasize the security challenges that arise in the context of networked environments, ubiquitous and mobile computing due to the increase in data sizes.

Mubina Malik et al [3] discussed in their paper about the techniques used for database security and they state that 84% companies feel that database security is adequate and 73% of companies predict database attacks increase day by day, 48% of attackers are authorized users. They also suggest some security measures to be adapted like access control policy, good audit trail in order to avoid privilege abuse attack and legitimate privilege abuse. To protect the unmanaged sensitive data the user has to encrypt the sensitive data and apply the required controls and permissions to the database. Denial of Service(DoS) attacks can be avoided by hardening TCP/IP stack by applying the appropriate registry settings to increase the size of the TCP connection queue, decrease the connection establishment period and employ the dynamic backlog mechanisms to ensure that the connection queue is

never exhausted. They also suggested a network intrusion system (IDS) to automatically detect and respond to SYN attacks. In order to control database threats, they suggest access control methods, Inference policies, user identification/authentication, accountability/auditing and encryption techniques. Data encryption becomes one of the important measures of data protection when the data being at rest or in transit by using SSL/TLS solutions.

Section 2 elaborates the literature behind this work as discussed by various authors; section 3 compares the various encryption techniques adapted in MongoDB database, section 4 discusses the results of various encryption measures as suggested by the authors based on the important parameters and section 5 includes the conclusion and future works to be carried out on this work.

II. RELATED WORK

In this section, the data protection techniques on open source NoSQL databases are discussed in detail and the best countermeasures are compared on the basis of performance overhead, memory usage, speed and time related parameters.

2.1 Security Aspects in MongoDB

MongoDB has become one of the top players in non-relational database providers that are available as open source document based database. MongoDB is built with low up-front operational cost and suitable for organizations across retail, financial, healthcare and government entities. Huge volumes of unstructured data are manipulated easily with MongoDB database. When user's confidential data are handled often to read, write, update and retrieve there is a demand for high security. For the enterprise editions some security measures are followed with the best practices adapted by the database administrators. If the best practices are not followed carefully, there is a chance of data loss or threats to the sensitive data of the common users.

Priyadharsini et al[4] studied a data protection system that is used for encrypting data before storing data into database repository and the NoSQL injections performed with javascript and PHP. Table 1 depicts that MongoDB does not provide data encryption which is the problem that is discussed in detail here as discussed by the authors.

Table 1 MongoDB status [4]

Category	Status	Recommendations
Data	No encryption	Encryption is critical
Authentication	Unsharded configuration	Supports proxy authentication
Authorization	Unsharded configuration	To generate complete authentication
Auditing	Not available	Support auditing and generate audit events
Injection attacks	JavaScript	Malicious user cannot modify the code

Data in MongoDB database can be encrypted when the data are at rest or in motion. MongoDB offers some solutions for encrypting data in motion and at rest. The various proposed methods of encryption are discussed in the following sections.

2.1.1 Encryption methods when data is at rest and in motion

For securing data in motion, all versions of MongoDB support TLS(Transport Layer Security) and SSL(Secure Socket Layer) to transfer the data over networks. This type of encryption technique is commonly used to secure website traffic and file sharing. While in transit, that is when the data travels from one point to other, it is unencrypted or 'in the clear'. MongoDB provides asymmetric key protocols to configure and secure the data in motion [12]. One of the challenges for MongoDB users is that when sensitive information is added to the database, users have to adapt a safe strategy of encrypting the sensitive data in the MongoDB database with a proper key management.

MongoDB Enterprise offers a storage based file symmetric key encryption called Transparent Data Encryption(TDE) to encrypt the whole database files at storage level. Version 3.2, MongoDB utilizes the Advanced Encryption Standard (AES) 256-bit encryption algorithm, an encryption cipher which uses the same secret key to encrypt and decrypt data. But data at rest encryption is only available on MongoDB enterprise and atlas editions using the required Wired Tiger storage engine.

When TDE is used to encrypt the data, a unique, private key is generated. Each encrypted database file generates a new private symmetric key, and all keys in the storage device are encrypted using a master key. MongoDB never allows the master key to be stored on the same server as the encrypted data [21]. The security admin or the database has to identify a secure storage location for the encryption key. MongoDB recommends third-party enterprise key management solutions; however, users have the option to store the key locally using a key file. But according to the best practices, storing the key locally is risky, and almost not recommended for key protection.

2.2 MongoDB Security Mechanisms

Anil Kumar et al[7] proposed an additive homomorphic asymmetric cryptosystem which is used to encrypt the users data in MongoDB(CryptMDB) and provides strong user's data privacy protection. The authors utilized a common encrypted tool proposed by Paillier et al.[8] Which is used to

achieve additive operations over encrypted data. They have also concluded that cryptMDB could provide strong privacy protection for user's data and prevent intruders from gaining access to the database.

Grim et al [9] studied the performance and security of NoSQL databases especially in MongoDB by implementing two types of encryption namely, encryption at rest which is performed at the server and en-to-end encryption done by the client. For the latter case they have extended earlier work done by Alves et al.[10] by adding functionality to their Python MongoDB connector wrapper. This research shows that enabling encryption introduces overhead in both the case of encryption at rest and the case of end-to-end encryption. MongoLabs natively supports encryption at rest since version 3.2 of their MongoDB Enterprise Advanced edition1. It uses the OpenSSL library to encrypt pages at application level using AES256-CBC. This improves performance as only modified pages need to be encrypted or decrypted [11]. Additionally this research shows that enabling end-to-end encryption prevents numerous attack vectors at the server side, with the introduction of significant overhead in performance and limiting the number of supported queries [12]. In this work using two benchmarking frameworks, YCSB and BenchmarkDB, the overhead of doing two types of encryption on a MongoDB server was estimated. These benchmarks showed a small overhead for doing encryption at rest and a relatively big overhead for using the Secure Mongo framework.

III. ENCRYPTION TECHNIQUES USED IN MONGODB

MongoDB is mostly written in C++. It is a document kind database that manages JSON-like documents. Here data can be nested in complex hierarchies and then also are indexable. Documents are stored in the form of collections in collections and are in turn stored in database. It provides high amount of unity, regularity and the ability to change size[13]. MongoDB offers a set of collection in collection .A collection can be considered as a table but the table does not have a schema here, set of fields could be defined as a document. In this every document consists of an id. MongoDB is a cluster of nodes where not every node is the same that is it lacks symmetry.

3.1 Security Measures on NoSQL databases

In this section, the security aspects of MongoDB are studied in various aspects as discussed by different research perspectives. The following are the security considerations in MongoDB:

- In MongoDB the data files are unencrypted in nature and it does not provide an implementation to encrypt the files automatically.
- It uses a binary wire level protocol which uses TCP port 27017 by default. It is at high verge of injection attacks because it uses mostly java script language. Thus, any attacker can easily acquire the passwords of the data files of user in a particular database.
- No auditing is allowed here.
- In data communication no encryption is there.

MongoDB has a de-normalized model that basically contributes in incrementing the query speed. The authors tested the performance of MongoDB on the basis of inserting 10000 notebooks information data into the database and found that MongoDB spends less time compared to MySQL and implied the query efficiency is improved in NoSQL databases [14]. But on the other side, NoSQL databases are weak in security measures compared to SQL databases [15]. In order to reduce the injection attacks, the application must be verified by reasonable input validation measures and NoSQL databases are m.uch efficient in cloud computing and more research attention is required in this line [16].

3.1.1 Authentication protocol

Priyadarsini et al[4] propose Kerberos, an authentication protocol to secure the network which provides authentication service and mutual authentication between user and server. Kerberos protocol builds on symmetric key cryptography. It requires a trusted third party, and it may use cryptography during authentication.

The Kerberos requirements are scalable, secure, reliable and transparent. It is a solution for network security problems. This protocol used the strong cryptography. Clients can identity to a server across an insecure network connection.

ALOGRITHM:

Step 1: Symmetric key cryptography is used to authentication Kerberos

Step 2: Kerberos Key Distribution Centre provides scalability.

Step 3: Secure transport of a session key can provided by Kerberos ticket.

Step 4: The session key is distribute by Kerberos KDC send to its client.

Step 5: The use of the entities master keys can be limited by the Kerberos ticket granting Ticket.

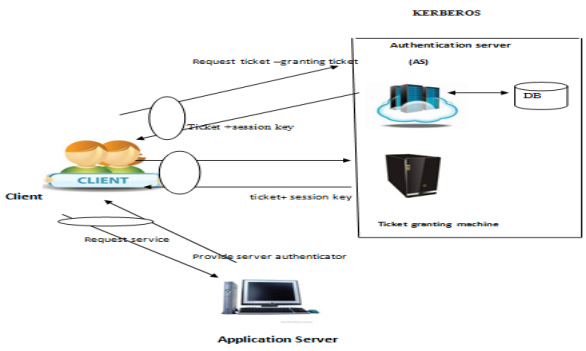


Figure 1. Kerberos Architecture[4]

The author suggests their proposed model as depicted in Figure 1, which helps to design and enhance security in NoSQL databases. It provides the tools of strong cryptography over the network to secure the information systems across the entire enterprise[5,6]. Kerberos can be extended to provide auditing service, thus securing the NOSQL database.

3.1.2 Asymmetric cryptosystem

ANIL Kumar et al[7] proposed an additive homomorphic asymmetric cryptosystem in their paper by using CryptMDB, they utilize a common encrypted tool proposed by Paillier et al. [8] which can achieve additive operations over encrypted data.

Pailier Algorithm:

Key generation:

1. Choose two large primes p and q randomly and independently of each other such that

$gcd\ (pq,(p-1)(q-1))=1$.

This property is assured if both primes are of equal length.

2. Compute $n=pq$ and $\lambda=lcm(p-1,q-1)$.

3. Select random variable g such that $g \epsilon Zn$

$2*$

4. Ensure n divides the order of g checking the existence following modular multiplicative

inverse: $\mu=(L(g\lambda \bmod n2)-1 \bmod n$, where L function

is defined as: $L(x)=$

5. The public (encryption) key is (n,g)

6. The private (decryption) key is (λ,μ)

Encryption:

1. Let m be the message to be encrypted $0 \le m \le n$

2. Select random number r, $0 \le r \le n$

3. Compute ciphertext as: $c=gm.rn \bmod n2$

Decryption:

1. Let be the ciphertext to decrypt, where: $c \epsilon Zn$

$2*$

2. Compute the plaintext message as:

$m=L(c\lambda \bmod n2).\mu \bmod n$

Homomorphic properties:

A notable feature of the Paillier cryptosystem is its homomorphic properties along with its non deterministic encryption. As the encryption function is additively homomorphic, the following identities can be described:

Homomorphic addition of plaintexts: The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts

$D(E(m1,r1) . E(m2,r2) \bmod n2)= m1 + m2 \bmod n$

The product of a ciphertext with a plaintext raising g will decrypt to the sum of the corresponding plaintexts,

$D(E(m1,r1) . gm2 \bmod n2)= m1 + m2 \bmod n$

Homomorphic multiplication of plaintexts:

An encrypted plaintext raised to the power of another plaintext will decrypt to the product of the two plaintexts,

$D(E(m1,r1)$

$m2 \bmod n2)= m1\ m2 \bmod n$

$D(E(m2,r2)$

$m1 \bmod n2)= m1\ m2 \bmod n$

as per the method suggested, the encryption will provide a secure access of the data at the database.

3.2 APPLICATION LEVEL ENCRYPTION

In MongoDB database, data can be encrypted at the application level itself as proposed by the following authors. Charmi et al[17] discusses the security features of MongoDB and proposes an encryption at application level [18]. Figure 2 depicts the application level encryption suggested by the authors. They proposed the following steps to encrypt the important fields like address, PAN number etc. The object id remains the same while updating the data that can be used for further verification of data.

Step 1: At first step client will log in the system and authenticate himself using user id and password. Application server checks for the client's access permissions and grant the access of database to the client.

Step 2: After the authentication client can access the database. He can perform insert the data, update the existing data or delete the data from database. Client will send the data to application server in plain text format. And at retrieval of the data server will provide plain data to client.

Step 3: Application server then apply AES algorithm on data which is send by the client. On Client's request for data retrieval application server fetch the encrypted data from the database and decrypt that data using AES algorithm and plain data will send back to client.

Step 4: Application server store the encrypted data into specific collection of particular database on insertion operation and retrieve the encrypted data from specific collections from particular mongo database.

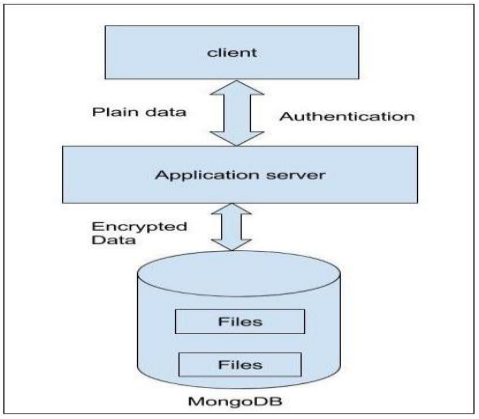


Figure 2. Encrypting data at application level [17]

Anand Shende et al [19] proposed an application level encryption methodology to secure the Adhar number of an individual. This unique Identification number can be used for secure transactions. Figure 3 is the proposed architecture used by the authors.

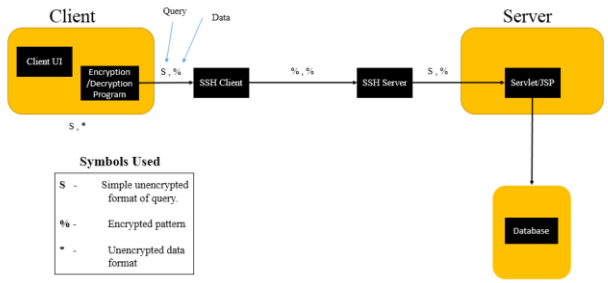


Figure 3 Client side encryption and Decryption [19]

Here SSL protocol is used to transfer the data and the server uses JSP and is connected to the database via a JDBC. Whenever the data is taken, the data it is encrypted and is represented by '%'. The unencrypted or plain text is represented by '*'. The S stands for the query.

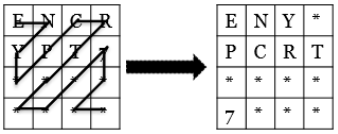
SSL protocol builds a secure tunnel based communication interface between the client and the server. Here the SSL is used in the grid network to resolve the load over the network after the authentication of both sender and receiver. The stage of SSL protocol are as follows:
i. To establish the key of safety communication.
ii. Server authentication.
iii. Client authentication.
iv. End stage
When the client tries to retrieve data, it sends the query. The query is then encrypted by SSL Client and is decrypted by SSL Server. The decrypted query is sent to the Server. The Server then returns the documents requested by the Client. The algorithm which the paper [20] proposes is an ETSFS algorithm. This algorithm is modified by this authors [19] in the following method:
Algorithm for Encryption –

1. The string entered is converted into an array of integers. This is a 2-dimensional array of 4 rows and 4 columns.

E	N	C	R
Y	P	T	7
*	*	*	*
*	*	*	*

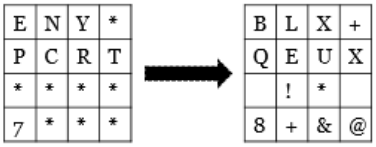
The above is the plotting of the string 'ENcrypt7' in a 2D array form.

2. Transpose.

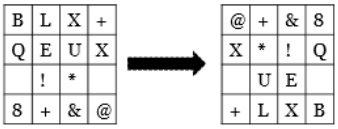


We just move around the elements of the array in the way shown by the arrows.

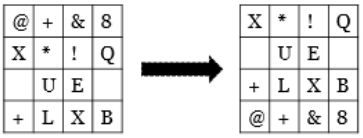
3. Substitution.
In the substitution phase, the values of the characters are substituted by other values. This adds to the security. Substitution is a three step process- 1. Characterizing 2. Keys formation 3. Operation. During the first phase, the character is categorized into small caps (a-z), capital alphabets (A-Z), digits (0-9) and symbols (! @#$%*).
Characterizing gives the value of 'M'. For alphabets (i.e. small and capital) M is 26. While for digits it is 10 and for symbols it is 7. Three user given keys are convert into 12 keys. Next the character's values are taken in its numeric form. For example, A is replaced by 1, C by 3, Y by 25. Now we add the keys and take modulus with M. Consider the character as 'E', its numeric is 5. The key element is suppose 23. Then adding 23 and 5 will give 2. Which is the character 'B'.



4. Folding.
The matrix is mirrored about the left diagonal first and then the right diagonal. All the elements along the diagonal are only interchanged. Now for the rest of the elements the matrix is mirrored about the horizontal centre and then the vertical centre. Now the rest of the elements are interchanged.



5. Shifting.
This shifts the rows upwards by 1. The upper row is added down at the end.



The above method as suggested by the authors could be used to provide encryption on the data stored in NoSQL databases which can be used to securely handle the confidential data.

IV RESULTS AND DISCUSSION

Data security becomes one of the main requirements for any type of database chosen by the users. Especially in NoSQL databases, huge volumes of unstructured, document oriented data are manipulated. Initially, security was not given importance on the built of these NoSQL databases like MongoDB, Cassandra, Redis, Hypertable etc. But, dealing with the sensitive data of the users in the domain like healthcare and financial sectors, there is a need for protecting the data in the databases. Data encryption is one of the solutions for protecting the data in the database, when the data is in the motion or at rest.

Encrypting the data at rest or in transit are some of the best measures of securing the data in MongoDB databases. Since these NoSQL databases handle unstructured data available in the form of documents, there are more chances of security breaches on the private data that are to be handled confidentially. Since MongoDB does not follow any encryption technique at the lower editions, there is a need for adapting some encryption methods either at database level or at application level. Some of the security measures for the data at rest and in transit are discussed and the proposed solutions are compared based on their performance.
Application level encryption was proposed with the help of the AES algorithm and SSL protocol. Before accessing the data at the database, encryption method starts at the application level itself. According to the authors[17], AES algorithm is the best among the other algorithms such as DES, 3DES, CASTS, MARS, IDEA, BLOWFISH and RC6 based on scalability. AES algorithm proved to be secure, utilizes less storage space, fast and effective encryption algorithm as shown in Figure 4.

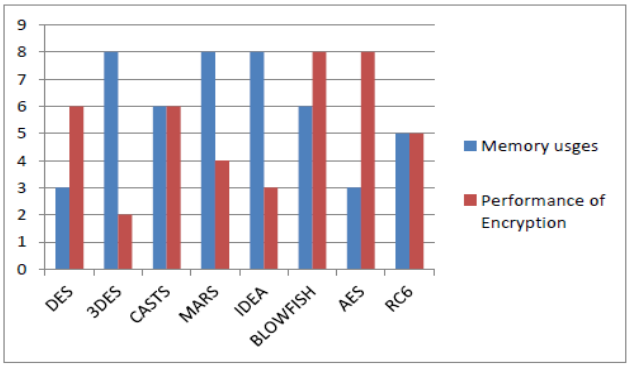


Figure 4. Comparisons of algorithms based on scalability [17]

From their work the object id remains the same while the other sensitive data like PAN number, personal address are encrypted at the application level. With the help of the proposed method, this application level encryption makes the MongoDB data secure for the users.

Another encryption technique at application level proposes an algorithm named, ETSFS algorithm which is an improved version of the older versions as suggested by the authors [19]. The three level encryption methods are shown in the figure 5.

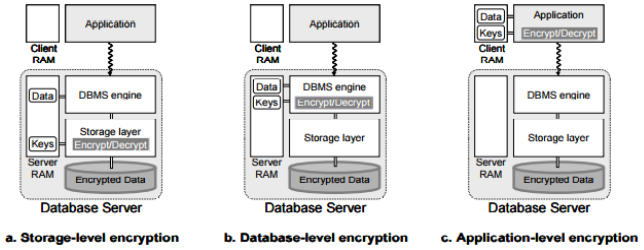


Figure 5. Three level encryption methods[19]

Their proposed algorithm encryption at application level is adapted and implemented in MongoDB database. They proposed an improved ETSFS algorithm to encrypt the adhar number of users and thus encrypting the sensitive data.

From the literature survey, there are some encryption algorithms proposed by the authors to secure the data stored in the MongoDB database at application level. There are some database level encryption methods that are proposed by the authors[7] called CryptMDB in order to achieve privacy protection of user's data. They use an additive homomorphic asymmetric cryptosystem to encrypt the data which prevents adversaries from the illegal access of the database.

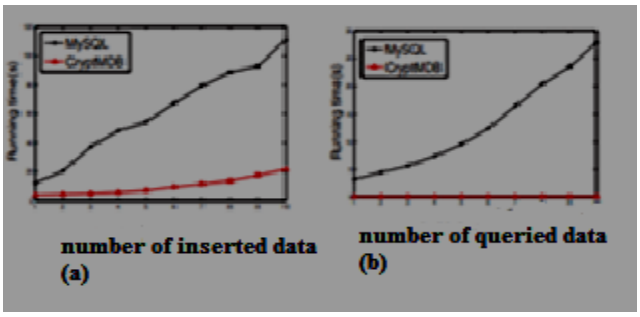


Figure 6. Total running time(s).
(a) For the different number of inserted data.
(b) For the different number of queried data [7]

From the Figure 6, user's record are inserted in MySQL and MongoDB from 10000 to 100000. As per the results, CryptMDB has high insertion speed compared to MySQL. For example, when the number of inserted data reach 100000, the CryptMDB only takes 21.513s to complete inserted operations while MySQL needs 111.025 to finish the same operations as per the proposed technique applied by the authors. Similarly, Fig. 6.(b) shows that the running times with different number of queried data, from the picture it is obvious that the CryptMDB has stronger queried ability compared with MySQL.

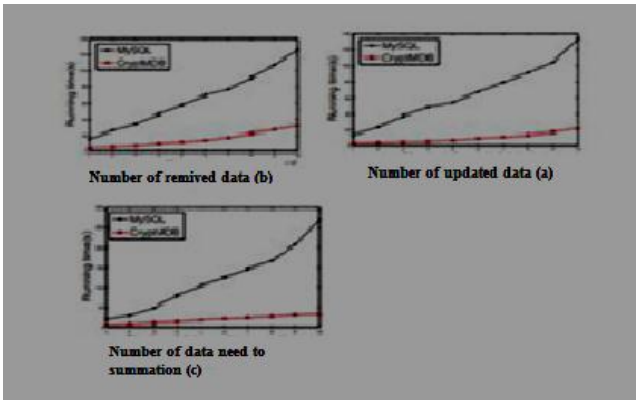


Figure 7: Running time(s).
(a) For the different number of updated data.
(b) For the different number of removed data.
(c) For the different number of data need to summation [7]

From the Figure 7, with the increase of user's data, compared with MySQL, cryptMDB achieve large scale data access, For example, when the number of updated data reach 100000, MySQL need 133.821s to execute all the sql requests but CryptMDB only takes 22.513s to achieve the same tasks. As shown in Fig. 6.(b), removal of the users' data from 10000 to 100000 orderly, it is not difficult to find that CryptMDB has higher performance to remove user's information, especially when users' data are huge. Fig. 6.(c) shows that the running times with different number of data need to summation, because the strong ability of distributed data processing, it is undoubted that the CryptMDB has lower running time to achieve same operations compared with MySQL as validated by the authors.

From the results of the above proposed mechanism as suggested by various authors, NoSQL databases need a strong encryption method to protect the sensitive data of the user's. MongoDB enterprise editions built with security measures as they improved in the later versions. Unstructured data storage requires a high data security when they are accessed in open while we use the open source databases like MongoDB. The data stored in the MongoDB are to be secured by a strong encryption techniques either at rest or in transit. The data could be secured at application level or at database level and need some more improved methods which minimize the time and increase the speed of access.

V CONCLUSION AND FUTURE SCOPE

In this paper various encryption algorithms are compared based on their methods of securing the data from the external attacks that are made intentionally or unintentionally on the NoSQL databases. MongoDB database is analyzed based on the security measures adapted at database level or at application level either the data is at rest or in transit. The proposed intermediate algorithms that are used to encrypt the data at application level and at database level meets some of the performance enhancements like in terms of speed and time. Compared to relational databases like MySQL, MongoDB manipulates the huge volumes of unstructured data in a fast and open source format. But there is a streamlined security measurement is to be adapted by the database administrators in order to save the sensitive data that are stored in the NoSQL databases. Some of the proposed algorithms improved the performance of the database secure access as suggested by the authors.

But every encryption technique has to be adapted with additional efforts that are to be strictly followed by the database administrators that makes the process tedious and could take extra checklists to be followed in a controlled manner. In future, the NoSQL databases can concentrate more on the security considerations like encryption at the built itself that makes the database as a high secured one to be adapted by the common users.

REFERENCES

[1]. P.R.Hariharan & Dr. K.P. Thooyamani , Various Schemes for Database Encryption - A Survey", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 19 (2017) pp. 8763-8767, Research India Publications.

[2]. Jef Van Loon, Prof. Dr. C-C. Kanne, Ch. Sturm, "Database Security - Concepts, Approaches", **Article** in IEEE Transactions on Dependable and Secure Computing · Seminar in Database Systems, University of Zurich, Department of Informatics, Autumn Term 2008, DOI: 10.1109/TDSC.2005.9 · Source: IEEE Xplore.

[3]. Mubina Malik and Trisha Patel, "DATABASE SECURITY - ATTACKS AND CONTROL METHODS", International Journal of Information Sciences and Techniques (IJIST) Vol.6, No.1/2, March 2016.

[4]. S. Priyadharshini, R. Rajmohan, "Analysis on Database Security Model Against NOSQL Injection", International Journal of Scientific Research in Computer Science, Engineering and Information Technology © 2017 IJSRCSEIT | Volume 2 | Issue 2 | ISSN : 2456-3307

[5]. Suna Yin, Dehua Chen, Jiajin Le,China, 2016 IEEE,"STNOSQL Creating NOSQL Database on the SensibleThings Platform.

[6]. Boyu Hou, Kai Qian, Lei Li, Yong Shi, Lixin Tao, Jigang Liu, USA, 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing ,"Mongo Database NOSQL Injection Analysis and Detection".

[7]. Anil Kumar, Harsha H L, B. Swaroop Reddy, K.Sunil Kumar Reddy, Krishna N, "Homomorphic Encrypted MongoDB for Users Data Security", International Journal of Engineering Research in Computer Science and Engineering (IJERCSE) Vol 5, Issue 6, June 2018.

[8]. https://en.wikipedia.org/wiki/Paillier_cryptosystem M.W. Grim, A.T. Wiersma, F. Turkmen, "Security and Performance Analysis of Encrypted NoSQL Databases", February 12, 2017.

[9]. Alves, Pedro. A framework for searching encrypted databases. In Anais do XVI Simpsio Brasileiro em Segurana da Informao e de Sistemas Computacionais (SBSeg 2016), 2016.

[10]. Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. "Benchmarking cloud serving

systems with YCSB", In Proceedings of the 1st ACM symposium on Cloud computing, pages 143{154. ACM, 2010.
[11]. At-test encryption in MongoDB 3.2: features and performance. https://www.mongodb.com /blog/post/at-rest-encryption-in-mongodb-3-2-features-and-performance.Accessed: 2017-01-23.
[12]. Vidushi Jain, Aviral Upadhyay, , "MongoDB and NoSQL Databases",*International Journal of Computer Applications (0975 – 8887) Volume 167 – No.10, June 2017*
[13]. Zhu Wei-ping, Li Ming-xin- Using MongoDB to Implement Textbook Management System instead of MySQL IEEE2011
[14]. Lior Okman, Nurit Gal-Oz, Yaron Gonen, Ehud Gudes, Jenny Abramov - Security Issues in NoSQL Databases IEEE2011
[15]. Tianyu Jia, Xiaomeng Zhao, Zheng Wang, Dahan Gong and Guiguang Ding - Model Transformation and Data Migration from Relational Database to MongoDB IEEE 2016.
[16]. Charmi Pariawala, and Ravi Sheth," Encrypting Data of MongoDB at Application Level", Advances in Computational Sciences and Technology, Volume 10, Number 5 (2017) pp. 1199-1205
[17]. Karan Patel, Kirti Sharma, Mosin Hasan. "Encrypting MongoDB Data using Application Level Interface". *Discovery, 2015, 46(214), 164-169*
[18]. Anand Shende1, Omkar Gurav2, Swapnil Shirode3, Piyush Govekar 4, S.N.Zaware 5, "Secure Unique Identification using Encrypted Storage in NoSQL Database", *International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 4, April 2016*
[19]. Hanan A. Al-Souly, Abeer S. Al-Sheddi, Heba A. Kurdi;Fast, Lightweight Symmetric Encryption Algorithm for Secure Database. (IJACSA) International Journal of Advanced Computer Science and Applications, Special Issue on Extended Papers from Science and Information Conference 2013.
[20]. https://docs.mongodb.com/manual/core/security-encryption-at-rest.

