Scattered,Simulteneous and Autonomous Access to Encrypted Database

Sagar.R.Jadhav
Dept.of Computer science
Vathsalya Institute of Science and Technology
Telangana, India
srjadhav6@gmail.com

Prof. Ajaykumar Kurra
Dept.of Computer Science
Vathsalya institute of science and Technology
Telangana, India
ajaykumarkurra@gmail.com

***Abstract*—**Put the important data on cloud i.e Cloud provider Storage, It should giving the guarantee of security without loss of data while data in use or not in use. Much of the option are available for providing storage services. We decelop an best architecture which integrates data over the cloud and execute multiple operation simultaneously on Encrypted cloud. We are connecting multiple client those are physically distributed. An another advantage we are eliminating the proxies for best performance .the architecture based on theoretical basis. We are providing the prototype to the different client & Network delay.

Keywords-*Security,cloud,database,secureDBaaS.*

*****

I. INTRODUCTION

In a cloud,in which important data is stored at untrusted third parties so here confidentiality of data important parameter.this required meaningful data management choices.Original data should be access by trusted parties excluding internet and cloud providers;in untrusted network information must be encrypted.here different types of cloud services define different level of complexities to satisfying these goals.in this paper ,we propose SecureDBaas that allows cloud to take full benefits of DBaaS qualities ,without showing unencrypted information to the cloud provider

The construction modeling outline was persuaded by a triple objective: to permit various, free, and topographically circulated customers to execute simultaneous operations on scrambled information, including SQL explanations that alter the database structure to safeguard information privacy and consistency at the customer and cloud level; to take out any middle of the road server between the cloud customer and the cloud supplier. The likelihood of consolidating reliability, what's more, versatility of a run of the mill cloud DBaaS with information secrecy is exhibited through a model of SecureDBaaS that backings the execution of simultaneous what's more, free operations to the remote scrambled database from numerous geologically conveyed customers as in any decoded DBaaS setup. To accomplish these objectives, SecureDBaaS coordinates existing cryptographic plans, separation instruments, and novel procedures for administration of encoded metadata on the untrusted cloud databases. In paper contains a hypothetical exchange about answers for information consistency issues because of simultaneous and free customer gets to encoded information. In this setting, we can't apply completely homomorphic encryption plans [7] as a result of their extreme compu our tational intricacy.

The SecureDBaaS construction modeling is customized to cloud stages and does not present any middle person intermediary on the other hand representative server between the customer and the cloud supplier. Wiping out any trusted middle of the server to permits SecureDBaaS to accomplish the same accessibility, irresolute quality, and flexibility levels of a cloud DBaaS. Other recommendations (e.g., [8], [9], [10], [11]) in view of middle of the road server(s) were viewed as impracticable for a cloud-based arrangement in light of the fact that any intermediary speaks to a solitary purpose of disappointment and a framework bottleneck that confines the principle advantages (e.g., adaptability, accessibility, and flexibility) of a database administration sent on a cloud stage. Not at all like SecureDBaaS, architectures depending on a trusted transitional intermediary don't bolster the most commonplace cloud situation where geologically scattered customers can simultaneously issue read/compose operations and information structure adjustments to a cloud database.

A substantial arrangement of examinations taking into account genuine cloud stages show that SecureDBaaS is early material to any DBMS in light of the fact that it obliges no adjustment to the cloud database administrations. Different studies where the proposed construction modeling is liable to the TPC-C standard benchmark for diverse quantities of customers and system latencies demonstrate that the execution of simultaneous read and compose operations not changing the SecureDBaaS database structure is equivalent to that of decoded cloud database. Workloads including changes to the database structure are likewise bolstered by SecureDBaaS, however at the cost of overheads that appear to be satisfactory to accomplish the wanted level of information classifiedness. The inspiration of these outcomes is that system latencies, which are average of cloud situations, have a tendency to veil the execution expenses of information encryption on reaction time. The general conclusions of this paper are critical in light of the fact that interestingly they exhibit the materialness of encryption to cloud database benefits as far as practicality and execution.

II. RELATED WORK

SecureDBaaS gives a few unique components that separate it from past work in the field of security for remote database administrations.

- It promises information classifiedness by permitting a cloud database server to execute simultaneous SQL operations (read/compose, as well as changes to the database structure) over encoded information.

- It gives the same accessibility, flexibility, and versatility of the first cloud DBaaS on the grounds that it does not oblige any moderate server. Reaction times are influenced by cryptographic overheads that for most SQL operations are conceal by system latencies.
- Different customers, conceivably geologically disseminated,can get to simultaneously and freely a cloud database administration.
- It doesn't oblige a trusted intermediary or a trusted intermediary in light of the fact that occupant information and metadata put away by the cloud database are constantly encoded.
- It is perfect with the most mainstream social database servers, and it is material to diverse DBMS usage in light of the fact that every single embraced arrangement are database rationalist.

Cryptographic record frameworks and secure capacity arrangements speak to the most punctual works in this field. We don't detail the few papers and items (e.g., Sporc [3], Sundr [4], Station [5]) on the grounds that they don't bolster calculations on scrambled information.

Distinctive methodologies ensure some secrecy (e.g., [12], [13]) by appropriating information among distinctive suppliers and by exploiting mystery sharing [14].In such a way, they keep one cloud supplier to peruse its segment of information, however data can be reproduced by conniving cloud suppliers. A stage forward is proposed in [15], that makes it conceivable to execute range inquiries on information what's more, to be vigorous against tricky suppliers. SecureDBaaS varies from these arrangements as it doesn't oblige the utilization of various cloud suppliers, and makes utilization of SQL-mindful encryption calculations to bolster the execution of most basic SQL operations on scrambled information.

SecureDBaaS relates all the more nearly to works utilizing encryption to secure information oversaw by untrusted databases. In such a case, a primary issue to address is that cryptographic methods can't be nai¨vely connected to standard DBaaS since DBMS can just execute SQL operations over plaintext information.

Some DBMS architecture offer the likelihood of scrambling information at the filesystem level through the supposed Transparent Information Encryption highlight [16], [17]. This element makes it conceivable to assemble a trusted DBMS over untrusted stockpiling. On the other hand, the DBMS is trusted and unscrambles information some time recently their utilization. Subsequently, this methodology is not appropriate to the DBaaS setting considered by SecureDBaas, in light of the fact that we expect that the cloud supplier is untrusted.

Different arrangements, for example, [18], permit the execution of operations over encoded information. These methodologies save information classifiedness in situations where the DBMS is most certainly not trusted; in any case, they oblige a changed DBMS engine what's more, are not good with DBMS programming (both business and open source) utilized by cloud suppliers. Then again, SecureDBaaS is good with standard DBMS engine, and permits occupants to manufacture secure cloud databases by utilizing cloud DBaaS benefits as of now accessible. Thus, SecureDBaaS is more identified with [9] and [8] that save information privacy in untrusted DBMSs through encryption strategies, permit the execution of SQL operations over scrambled information, and are perfect with regular DBMS engine. On the other hand, the building design of these arrangements is in view of a transitional and trusted intermediary that intervenes any association between every customer and the untrusted DBMS server. The methodology proposed in [9] n by the creators of the DBaaS model [6] meets expectations by encoding squares of information rather than every information thing. At whatever point an information thing that has a place with a square is obliged, the trusted intermediary requirements to recover the entire piece, to decode it, and to channel out superfluous information that fit in with the same square. As a result, this configuration decision obliges substantial alterations of the first SQL operations created by each customer, therefore bringing on noteworthy overheads on both the DBMS server and the trusted intermediary. Different works [10], [11] present streamlining and speculation that broaden the subset of SQL administrators bolstered by [9], however they share the same intermediary based structural planning and its natural issues. On the other hand, SecureDBaaS permits the execution of operations over encoded information through SQL-mindful encryption calculations. This system, at first proposed in CryptDB [8], makes it conceivable to execute operations over encoded information that are like operations over plaintext information. By and large, the question arrangement executed by the DBMS for scrambled and plaintext information is the same.

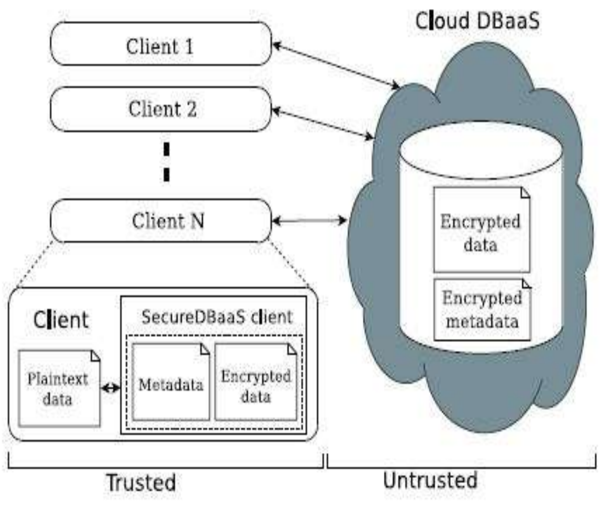


Fig. 1. SecureDBaaS architecture.

The dependence on a trusted intermediary that classifies [9] and [8] encourages the execution of a protected DBaaS, and is appropriate to multitier web applications are primary core interest. Then again, it causes a few downsides. Since the intermediary is believed, its capacities can't be outsourced to an untrusted cloud supplier. Thus, the intermediary is intended to be executed and oversaw by the cloud occupant. Accessibility, multiplicity, and flexibility of the entire secure DBaaS administration are then limited by accessibility, multiplibity, what's more, flexibility of the trusted intermediary, that turns into a solitary purpose of disappointment and a framework conjection. Since high accessibility, multithreading, and flexibility are among the preeminent reasons that prompt the selection of cloud administrations, this restriction blocks the pertinence of [9] what's more, [8] to the cloud database situation. SecureDBaaS settles this issue by letting customers unite specifically to the cloud DBaaS, without the need of any middle part also, without presenting new bottlenecks and single purposes of disappointment.

III. 3.ARCHITECTURE DESIGN

SecureDBaaS is intended to permit various and autonomous customers to unite specifically to the untrusted cloud DBaaS with no middle of the server. Fig. 1 depicts the general building design. We accept that an occupant association gets a cloud database administration from an untrusted DBaaS supplier. The inhabitant then sends one or more machines (Customer 1 through N) and introduces a SecureDBaaS ustomer on each of them. This customer allows a client to unite with the cloud DBaaS to manage it, to peruse and compose database, and indeed, even to make and change the database tables after creation.

The data oversaw by SecureDBaaS incorporates plaintext information, encoded information, metadata, and scrambled metadata. Plaintext information comprise of data that an occupant needs to store and process remotely in the cloud DBaaS. To keep an untrusted cloud supplier from disregarding classifiedness of inhabitant information put away in plain shape, SecureDBaaS receives different cryptographic systems to change plaintext information into encoded occupant information and scrambled inhabitant information structures in light of the fact that even the names of the tables and of their segments must be encoded. SecureDBaaS customers deliver likewise an arrangement of metadata comprising of data needed to encode and decode information and additionally other organization data. Indeed, even metadata are encoded furthermore, put away in the cloud DBaaS.

SecureDBaaS moves far from existing architectures that store only inhabitant information in the cloud database, and recovery metadata in the customer machine [9] or split metadata between the cloud database and a trusted proxy [8]. At the point while considering situations where different customers can get to the same database simultaneously, these past arrangements are wasteful. For instance, sparing metadata on the customers would require grand systems for metadata synchronization, and the strange possibility of permitting different customers to get to cloud database benefit freely. Arrangements of a trusted intermediary are more possible, however they present a framework bottleneck that decreases accessibility, flexibility, and multitasking of cloud database administrations.

3.1 Data Management

We are considering tenant data are saved in a relationaldatabase. We have to save the confidentiality of thestored data and even of the database infrastrcture because tableand column names may outcome information about saved data.We distinguish the types for encrypting the information structures and the inhabitant data

Encoded inhabitant information are put away through secure tables into the cloud database. To allows straightforward execution of SQL commands, each plaintext table is changed into a protected table in light of the fact that the cloud databases is untrusted. The name of a safe table is created by scrambling the name of the comparing plaintext table. Table names are encoded by method for the same encryption calculation and an encryption key that is known not the SecureDBaaS customers. Henceforth, the scrambled name can be processed from the plaintext name. Then again, section names of secure tables are haphazardly created by SecureDBaaS;hence, regardless of the possibility that distinctive plaintext tables have segments with the same names, the names of the section of the comparing secure tables are distinctive. This configuration decision enhances privacy by keeping an antagonistic cloud database from speculating relations among diverse secure tables through the distinguishing proof of segments having the same encoded name.

The information sort speaks to the kind of the plaintext information (e.g., int, varchar). The encryption sort recognizes the encryption calculation that is utilized to figure all the information of a section. It is picked among the calculations upheld by the SecureDBaaS executions. As in [8], SecureDBaaS influences a few SQL-mindful encryption calculations that permit the execution of proclamations over scrambled information. It is imperative to watch that every calculation bolsters just a subset of SQL administrators. These components are examined in Appendix C, accessible in the online supplemental material. At the point when SecureDBaaS makes a scrambled table, the information kind of every segment of the encoded table is dictated by the encryption calculation used to encode inhabitant information. Two encryption calculations are characterized good on the off chance that they create encoded information that require the same segment information sort.

The field confidentiality parameter allows a tenant to define explicitly which columns of which secure table should share the same encryption key (if any). SecureDBaaS offers three field confidentiality attributes

- Column (COL) is the default confidentiality level that should be used when SQL statements operate on one column; the values of this column are encrypted through a randomly generated encryption key that is not used by any other column.
- Multicolumn (MCOL) should be used for columns referenced by join operator, foreign keys, and other operation involving two columns; the two columns are encrypted through the same key.
- Database (DBC) is recommended when operations involve multiple columns; in instance, it is convenient to utilise the special encryption keys that is generated and implicitly shared among all the columns of the databases characterized by the same secure type.

The selection of the field confidentiality levels makes it possible to execute SQL statements over encrypted data while allowing a tenant to minimize key sharing.

3.2 Metadata Management

Metadata generated by Secure DBaaS contain all the in data that is necessary to manage SQL statements over the encrypted database in a way transparent to the users. Metadata managements strategies represent an original idea because SecureDBaaS is the first architecture storing all metadata in the untrusted cloud database together with the encrypted inhabitant data. SecureDBaaS uses two types of metadata.

- Database metadata are related with the whole databases.There is only one instance of this metadata type for each database.
- Table metadata are associated with one secure table.Each table metadata contain all information that is necessary to encrypt and decrypt data of the associated secure table.

Database metadata have the encryption keys that are used for the secure types having the field confidentiality set to database. A multiple encryption key is associated with all the possible combinations of data type and encryption type.Hence, the database metadata represents a keyring and do not contain any information about tenant data.

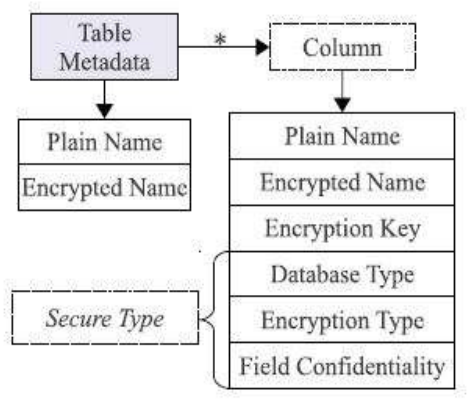


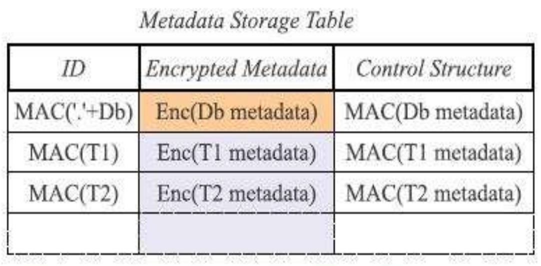
Fig. 2. Structure of table metadata.

The structure of a table metadata is shown in Fig. 2. Table metadata contain the name of the related secure table and the unencrypted name of the related plain text table. Moreover, table metadata include column metadata for each column of the related secure tables. Each column metadata contain the following information.

- Plain name: the name assign corresponding column of the plaintext tables.
- Coded name: the name allocated column of the secure table. This is the only data that links a column to the corresponding plain texts column because column names of secure tables are randomly generated.
- Secure type: the secure type of the column, as describe in Section 3.1. This allows a SecureDBaaS client to be notify about the data type and the encryption policies associated with a column.
- Encryption key: the key used to encode and decode all the information stored in the column.

SecureDBaaS stores metadata in the metadata storage table that is situated in the untrusted cloud as the database. This is a unique decision that increases adaptability, yet opens two novel issues as far as effective information recovery and information classifiedness. To permit SecureDBaaS customers to control metadata through SQL articulations, we spare database and table metadata in an even shape. Indeed, even metadata secrecy is ensured through encryption. The structure of the metadata stockpiling table is indicated in Fig. 3. This table uses one line for the database metadata, and one column for every table metadata.

Database and table metadata are encoded through the same encryption key before being commit. This encryption key is known as an master key. Just trusted customers that definitely know the master key can unscramble the metadata and gain data that is important to encode and decode inhabitant information. Every metadata can be recovered by customers through a related ID, which is the essential key of the metadata stockpiling table. This ID is figured by applying a Message Authentication Code (MAC) capacity to the name of the item (database or table) depicted by the relating line. The utilization of a deterministic MAC capacity permits customers to recover the metadata of a given table by knowing its plaintext name.


Metadata Storage Table

ID	Encrypted Metadata	Control Structure
MAC('.'+Db)	Enc(Db metadata)	MAC(Db metadata)
MAC(T1)	Enc(T1 metadata)	MAC(T1 metadata)
MAC(T2)	Enc(T2 metadata)	MAC(T2 metadata)

Fig. 3. Organization of database metadata and table metadata in the metadata storage table.

IV. OPERATIONS

In this area, we layout the setup setting operations did by a database manager (DBA), and we describe the execution of SQL operations on encoded information in two situations: a naive setting described by a client, and reasonable connections where the database administrations are gotten to by simultaneous customer

4.1 Setup Phase

We define how to initialize a SecureDBaaS architecture from a cloud database service adapted by a inhabitant from a cloud provider. We consider that the DBA make the metadata stockpiling tables that toward the starting contains only the database metadata, and is not the table metadata. The DBA introduces the database metadata through the SecureDBaaS client by using randomly generated encryption keys for any combinations of data types and encryption types, and save them in the metadata storage tables after encryption through the master key. Then, the DBA sprades the master key to the legitimate users. User access control olicies are administrated by the DBA through some standard data control language as in any unencrypted database.

For example, if the database has to support a join statement among the value of T1.C2 and T2.C1, the DBA must be use the MCOL field confidentiality for T2.C1 that references T1.C2 (solid arrow). In such a way, SecureDBaaS can be retrieve the encryption key specified in the column metadata of T1.C2 from the metadata table M1 and can be use the similar key for T2.C1. The solid arrow from M2 to M1 denotes that they explicitly share the encryption algorithm and the key.

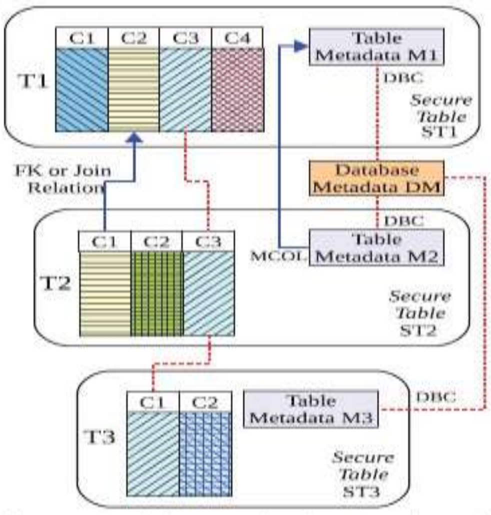


Fig. 4. Management of the encryption keys according to the field confidentiality parameter.

At the point when operations (e.g., arithmetical, request comparison)involve more than two segments, it is helpful to embrace the DBC field secrecy. This has a twofold advantage:we can utilize the unique encryption key that is created and verifiably shared among every one of the sections of the database portrayed by the same secure sort; we restrict conceivable consistency issues in a few situations described by simultaneous customers (see Appendix B, accessible in the online supplemental material). For instance, the sections T1.C3,T2.C3, and T3.C1 in Fig. 4 have the same secure sort. Hence,they reference the database metadata, as spoke to by the dashed line, and utilize the encryption key connected with their information and encryption sorts. As they have the same information and encryption sorts, T1.C3, T2.C3, and T3.C1 can utilize the same encryption key regardless of the possibility that no immediate reference exists between them. The database metadata as of now contain the encryption key K connected with the information and the encryption sorts of the three segments, in light of the fact that the encryption keys for all mixes of information and encryption sorts are made in the introduction stage. Subsequently, K is utilized as the encryption key of the T1.C3, T2.C3, and T3.C1 segments and replicated in M1, M2, and M3.

4.2 Sequential SQL Operations

The first connection of the client with the cloud DBaaS is for validation purposes. SecureDBaaS depends on standard validation and approval systems gave by the first DBMS server. After the authentication,a client interfaces with the cloud database through the SecureDBaaS customer. SecureDBaaS dissects the first operation to distinguish which tables are included and to recover their metadata from the cloud database. The metadata are retrive through the master key and their data is utilized to decipher the first plain SQL into an inquiry that works on the scrambled database.

4.3 Concurrent SQL Operations

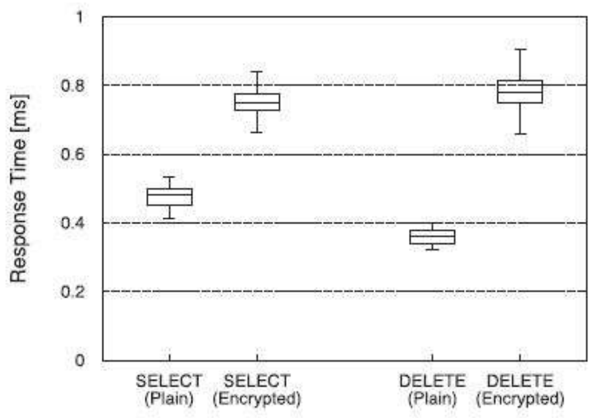
The support to simultaneous execution of SQL articulations issued by numerous autonomous customers is a standout amongst the most essential advantages of SecureDBaaS regarding best in class arrangements. Our construction modeling must ensure consistency among scrambled inhabitant information and encoded metadata on the grounds that defiled or outdated metadata would keep customers from interpreting scrambled occupant information bringing about changeless information misfortunes. An exhaustive investigation of the conceivable issues and arrangements identified with simultaneous SQL operations on scrambled occupant information and metadata is contained in Appendix B, accessible in the online supplemental material. Here, we comment the significance of recognizing two classes of explanations that are upheld by SecureDBaaS: SQL operations not bringing on adjustments to the database structure,such as read, compose, and upgrade; operations including changes of the database structure through creation, evacuation, and alteration of database tables.

V. EXPERIMENTAL RESULT

We describe the applicability of SecureDBaaS to differents cloud DBaaS outcomes by implementing and handling encrypted database operations on emulated and real cloud architecture. The present version of the SecureDBaaS prototype handles PostgreSQL, MySql, and SQL Server relational databases. As a first outcome, we can be analyse that porting SecureDBaaS to different DBMS required minor changes related to the database connector, and nominal updations of the codebase. We refers to Appendix C, available in the online supplemental materials, for an in-depth description of the prototype implementation.

Different tests are situated to confirm the operations of SecureDBaaS on distinctive cloud database suppliers. Examinations are done in Xeround [22], Postgres Plus Cloud Database [23], Windows SQL Azure [24], furthermore on an IaaS suppliers, similar to Amazon EC2 [25], that needs a manual setup of the database. The main gathering of cloud seller offer prepared to-utilize answers for occupants, however they don't permit a full access to the database framework. For e.g, Xeround gives a standard MySql interface and exclusive APIs that adaptability and accessibility of the cloud database, yet don't permit an immediate access to the machine.This confine the establishment of extra programming apparatuses, and any customization. On the positive side,SecureDBaaS utilizing simply standard SQL charges can scramble inhabitant information on any cloud database administration. Some best in class reckoning on encoded information may require the establishment of custom libraries on the cloud architecture.This is the situation of Postgres Plus Cloud that gives SSH access to enhance the database with extra capacities. We utilise the Emulab [26] testbed that provide us a controlled environment with several machines, ensuring repeatability of the experiments for the variety of scenarios to consider in terms of workload models, number of clients, and network latencies.

In first part we introduce secure DBaas & client evaluate SQL command an encrypted database via LAN.To evaluate performance overhead to encrypt SQL operations.we concentrate on most frequently executed SELECT,INSERT,UPDATE and DELETE operation statement of TPC-C benchmark.in fig 6 & 7,we compare response time.Y-axis gives knowledge about response time in ms,X-axis represent SQL operation.Each data manipulation command have its own response time.



In second part of experiment we evaluate effect of network latency and simultaneously on the utilization of cloud database from distinct client ,to this reason we identify network delay via metwork traffic shaping via Linux kernel by synthesize delay about 20 to 150 ms in client – server architecture.

VI. CONCLUSION

We propose an imaginative building design that ensures privacy of information put away out in the open cloud databases.Unlike best in class approaches, our answer does not depend on a middle of the road intermediary that we consider a solitary purpose of disappointment and a bottleneck restricting accessibility and adaptability of ordinary cloud database administrations. A substantial piece of the examination incorporates answers for backing simultaneous SQL operations (counting articulations adjusting the database structure) on scrambled information issued by heterogenous and potentially geologically scattered customers. The proposed structural engineering does not oblige changes to the cloud databases, and it is promptly appropriate to existing cloud DBaaS, for example, the tested PostgreSQL Plus Cloud Database [23], Windows Azure [24], and Xeround [22]. There are no hypothetical and down as far as possible to extend our answer for different stages and to incorporate new encryption calculations.

It merits watching that trial results of the TPC-C standard benchmark demonstrate that the execution effect of information encryption on reaction time gets to be insignificant on the grounds that it is conceal by system latencies that are run of the mill of cloud situations. Specifically, simultaneous read and compose operations that don't change the structure of the encoded database cause insignificant overhead.Dynamic situations portrayed by (conceivably) simultaneous adjustments of the database structure are upheld, however at the cost of high computational expenses. These execution results open the space to future enhancements that we are researching.

ACKNOWLEDGMENT

I feel great in expressing our deepest sense of gratitude to our guide and HOD Prof. Ajaykumar Kurra for his encouragement and enlightened comments throughout this project work. His appreciative suggestion always motivated us for putting most willing efforts on my study during project report. We are also thankful to the concerned authorities who directly or indirectly helped us in the project.

REFERENCES

[1] M. Armbrust et al., “A View of Cloud Computing,” Comm. of the ACM, vol. 53, no. 4, pp. 50-58, 2010.

[2] W. Jansen and T. Grance, “Guidelines on Security and Privacy in Public Cloud Computing,” Technical Report Special Publication 800-144, NIST, 2011.

[3] A.J. Feldman, W.P. Zeller, M.J. Freedman, and E.W. Felten, “SPORC: Group Collaboration Using Untrusted Cloud Resources,” Proc. Ninth USENIX Conf. Operating Systems Design and Implementation, Oct. 2010.

[4] J. Li, M. Krohn, D. Mazie`res, and D. Shasha, “Secure Untrusted Data Repository (SUNDR),” Proc. Sixth USENIX Conf. Opearting Systems Design and Implementation, Oct. 2004.

[5] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, and M. Walfish, “Depot: Cloud Storage with Minimal Trust,” ACM Trans. Computer Systems, vol. 29, no. 4, article 12, 2011.

[6] H. Hacigu¨mu¨ s,, B. Iyer, and S. Mehrotra, “Providing Database as a Service,” Proc. 18th IEEE Int’l Conf. Data Eng., Feb. 2002.

[7] C. Gentry, “Fully Homomorphic Encryption Using Ideal Lattices,” Proc. 41st Ann. ACM Symp. Theory of Computing, May 2009.

[8] R.A. Popa, C.M.S. Redfield, N. Zeldovich, and H. Balakrishnan, “CryptDB: Protecting Confidentiality with Encrypted Query Processing,” Proc. 23rd ACM Symp. Operating Systems Principles, Oct. 2011.

[9] H. Hacigu¨mu¨ s,, B. Iyer, C. Li, and S. Mehrotra, “Executing SQL over Encrypted Data in the Database-Service-Provider Model,” Proc. ACM SIGMOD Int’l Conf. Management Data, June 2002.

[10] J. Li and E. Omiecinski, “Efficiency and Security Trade-Off in Supporting Range Queries on Encrypted Databases,” Proc. 19th Ann. IFIP WG 11.3 Working Conf. Data and Applications Security, Aug. 2005.

[11] E. Mykletun and G. Tsudik, “Aggregation Queries in the Database-as-a-Service Model,” Proc. 20th Ann. IFIP WG 11.3 Working Conf. Data and Applications Security, July/Aug. 2006.

[12] D. Agrawal, A.E. Abbadi, F. Emekci, and A. Metwally, “Database Management as a Service: Challenges and Opportunities,” Proc. 25th IEEE Int’l Conf. Data Eng., Mar.-Apr. 2009.

[13] V. Ganapathy, D. Thomas, T. Feder, H. Garcia-Molina, and R. Motwani, “Distributing Data for Secure Database Services,” Proc. Fourth ACM Int’l Workshop Privacy and Anonymity in the Information Soc., Mar. 2011.

[14] A. Shamir, “How to Share a Secret,” Comm. of the ACM, vol. 22, no. 11, pp. 612-613, 1979.

[15] M. Hadavi, E. Damiani, R. Jalili, S. Cimato, and Z. Ganjei, “AS5: A Secure Searchable Secret Sharing Scheme for Privacy Preserving Database Outsourcing,” Proc. Fifth Int’l Workshop Autonomous and Spontaneous Security, Sept. 2013.

[16] “Oracle Advanced Security,” Oracle Corporation, http://www. oracle.com/technetwork/database/options/advanced-security, Apr. 2013.

[17] G. Cattaneo, L. Catuogno, A.D. Sorbo, and P. Persiano, “The Design and Implementation of a Transparent Cryptographic File System For Unix,” Proc. FREENIX Track: 2001 USENIX Ann. Technical Conf., Apr. 2001.

[18] E. Damiani, S.D.C. Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati, “Balancing Confidentiality and Efficiency in Untrusted Relational Dbmss,” Proc. Tenth ACM Conf. Computer and Comm. Security, Oct. 2003.

[19] L. Ferretti, M. Colajanni, and M. Marchetti, “Supporting Security and Consistency for Cloud Database,” Proc.

Fourth Int'l Symp. Cyberspace Safety and Security, Dec. 2012.

[20] "Transaction Processing Performance Council," TPC-C, http:// www.tpc.org, Apr. 2013.

[21] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil, "A Critique of Ansi Sql Isolation Levels," Proc. ACM SIGMOD, June 1995.

[22] "Xeround: The Cloud Database," Xeround, http://xeround.com, Apr. 2013.

[23] "Postgres Plus Cloud Database," EnterpriseDB, http:// enterprisedb.com/cloud-database, Apr. 2013.

[24] "Windows Azure," Microsoft corporation, http://www. windowsazure.com, Apr. 2013.

[25] "Amazon Elastic Compute Cloud (Amazon Ec2)," Amazon Web Services (AWS), http://aws.amazon.com/ec2, Apr. 2013.

[26] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An Integrated Experimental Environment for Distributed Systems and Networks," Proc. Fifth USENIX Conf. Operating Systems Design and Implementation, Dec. 2002.

[27] A. Fekete, D. Liarokapis, E. O'Neil, P. O'Neil, and D. Shasha, "Making Snapshot Isolation Serializable," ACM Trans. Database Systems, vol. 30, no. 2, pp. 492-528, 2005.

[28] A. Boldyreva, N. Chenette, and A. O'Neill, "Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions," Proc. 31st Ann. Conf. Advances in Cryptology (CRYPTO '11), Aug. 2011.

[29] "IP Latency Statistics," Verizon, http://www.verizonbusiness. com/about/network/latency, Apr. 2013.