ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Distributed, Concurrent, and Independent Access to Encrypted Cloud Databases

Mr.S.MD Ismail
Department of CSE,
Al Habeeb College of Engineering & Technology, Chevella.

Mr.Mohd Anwar Ali
Department of CSE,
Al Habeeb College of Engineering & Technology, Chevella.

Mr.Syed Tousif Ahmed
Department of CSE,
Al Habeeb College of Engineering & Technology, Chevella.

Abstract:

Storing critical information in cloud should come with the guarantee of security and availability for data at rest, in motion, and in use. Several alternatives exist for storage services, while data confidentiality solutions for the database as a service paradigm are still immature. In this paper, we propose a novel architecture that integrates cloud database services with data confidentiality and the possibility of executing concurrent operations on cipher text. This is the first solution supporting geographically distributed clients to connect directly to an encrypted cloud database, and to execute concurrent and independent operations including those modifying the database structure. Our proposed architecture has the further advantage of eliminating intermediate proxies that limit the elasticity, availability, and scalability properties that are intrinsic in cloud-based solutions. The efficacy of the proposed architecture is evaluated through theoretical analyses and extensive experimental results based on a prototype implementation subject to the TPC-C standard benchmark for different numbers of clients and network latencies.

Index Terms:

Cloud, security, confidentiality, SecureDBaaS, database.

1 INTRODUCTION:

IN a context of cloud, ensuring data confidentiality is of paramount importance. Here original plain data must be accessible only by trusted parties that do not include cloud providers, intermediaries, and Internet; in any untrusted context, data must be encrypted. In this context, we propose SecureDBaaS as the first solution that allows cloud tenants to take full advantage of DBaaS qualities, such as availability, reliability, and elastic scalability, without exposing unencrypted data to the cloud provider.

The architecture design was motivated by a threefold goal: to allow multiple, independent, and geographically distributed clients to execute concurrent operations on encrypted data, including SQL statements that modify the database structure; to preserve data confidentiality and consistency at the client and cloud level to eliminate any intermediate server between the cloud client and the cloud provider. The possibility of combining availability, elasticity, and scalability of a typical cloud DBaaS with data confidentiality is demonstrated through a prototype of SecureDBaaS that supports the execution of concurrent and independent operations to the remote encrypted database from many geographically distributed clients as in any unencrypted DBaaS setup. To achieve these goals, SecureDBaaS integrates existing cryptographic schemes, isolation mechanisms, and novel strategies for management of encrypted metadata on the untrusted cloud database. This paper contains a theoretical discussion about solutions for data consistency issues due to concurrent and independent client accesses to encrypted data.

In this context, we cannot apply fully homo morphic encryption schemes because of their excessive computational complexity. SecureDBaaS is immediately applicable to any DBMS because it requires no modification to the cloud database services (demonstrated by a large set of experiments based on real cloud platforms). Other studies where the proposed architecture is subject to the TPC-C standard benchmark for different numbers of clients and network latencies show that the performance of concurrent read and write operations not modifying the SecureDBaaS database structure is comparable to that of unencrypted cloud database. Workloads including modifications to the database structure are also supported by SecureDBaaS, but at the price of overheads that seem acceptable to achieve the desired level of data confidentiality. The motivation of these results is that network latencies, which are typical of cloud scenarios, tend to mask the performance costs ofdata encryption on response time.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The overall conclusions of this paper are important because for the first time they demonstrate the applicability of encryption to cloud database services in terms of feasibility and performance.

2 RELATED WORK:

SecureDBaaS provides several original features in the field of security for remote database services.

» It guarantees data confidentiality by allowing a cloud database server to execute concurrent SQL operations (not only read/write, but also modifications to the database structure) over encrypted data.
» It provides the same availability, elasticity, and scalability of the original cloud DBaaS because it does not require any intermediate server. Response times are affected by cryptographic overheads that for most SQL operations are masked by network latencies.
» Multiple clients, possibly geographically distributed, can access concurrently and independently a cloud database service.
» It does not require a trusted broker or a trusted proxy because tenant data and metadata stored by the cloud database are always encrypted.
» It is compatible with the most popular relational database servers, and it is applicable to different DBMS implementations because all adopted solutions are database agnostic.

SecureDBaaS relates more closely to works using encryption to protect data managed by untrusted databases. In such a case, a main issue to address is that cryptographic techniques cannot be nai¨vely applied to standard DBaaS because DBMS can only execute SQL operations over plaintext data.Some DBMS engines offer the possibility of encrypting data at the filesystem level through the so-called Transparent Data Encryption feature. This feature makes it possible to build a trusted DBMS over untrusted storage. However, the DBMS is trusted and decrypts data before their use. Hence, this approach is not applicable to the DBaaS context considered by SecureDBaas, because we assume that the cloud provider is untrusted. Other solutions, allow the execution of operations over encrypted data. These approaches preserve data confidentiality in scenarios where the DBMS is not trusted; however, they require a modified DBMS engine and are not compatible with DBMS software (both commercial and open source) used by cloud providers.

On the other hand, SecureDBaaS is compatible with standard DBMS engines, and allows tenants to build secure cloud databases by leveraging cloud DBaaS services already available. For this reason, SecureDBaaS is more related to [9] and [8] that preserve data confidentiality in untrusted DBMSs through encryption techniques, allow the execution of SQL operations over encrypted data, and are compatible with common DBMS engines. However, the architecture of these solutions is based on an intermediate and trusted proxy that mediates any interaction between each client and the untrusted DBMS server. The approach proposed in [9] by the authors of the DBaaS model [6] works by encrypting blocks of data instead of each data item. Whenever a data item that belongs to a block is required, the trusted proxy needs to retrieve the whole block, to decrypt it, and to filter out unnecessary data that belong to the same block. As a consequence, this design choice requires heavy modifications of the original SQL operations produced by each client, thus causing significant overheads on both the DBMS server and the trusted proxy. Other works [10], [11] introduce optimization and generalization that extend the subset of SQL operators supported by [9], but they share the same proxy-based architecture and its intrinsic issues. On the other hand, SecureDBaaS allows the execution of operations over encrypted data through SQL-aware encryption algorithms. This technique, initially proposed in CryptDB [8], makes it possible to execute operations over encrypted data that are similar to operations over plaintext data. In many cases, the query plan executed by the DBMS for encrypted and plaintext data is the same.

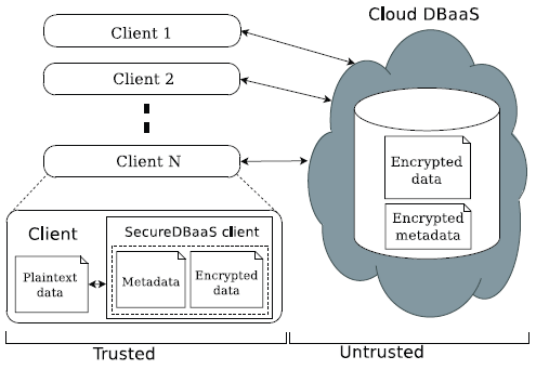

Fig. 1. SecureDBaaS architecture.

3 ARCHITECTURE DESIGN:

In this system, Multiple and independent clients can connect directly to the untrusted cloud DBaaS without any intermediate server. Fig. 1

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

describes the overall architecture. We assume that a tenant organization acquires a cloud database service from an untrusted DBaaS provider. The tenant then deploys one or more machines (Client 1 through N) and installs a SecureDBaaS client on each of them. This client allows a user to connect to the cloud DBaaS to administer it, to read and write data, and even to create and modify the database tables after creation. To prevent an untrusted cloud provider from violating confidentiality of tenant data stored in plain form, SecureDBaaS adopts multiple cryptographic techniques to transform plaintext data into encrypted tenant data and encrypted tenant data structures because even the names of the tables and of their columns must be encrypted. SecureDBaaS clients produce also a set of metadata consisting of information required to encrypt and decrypt data as well as other administration information. Even metadata are encrypted and stored in the cloud DBaaS.

3.1 Data Management:

Encrypted tenant data are stored through secure tables into the cloud database. To allow transparent execution of SQL statements each plaintext table is transformed into a secure table because the cloud database is untrusted. The name of a secure table is generated by encrypting the name of the corresponding plaintext table. Table names are encrypted by means of the same encryption algorithm and an encryption key that is known to all the SecureDBaaS clients. Hence, the encrypted name can be computed from the plaintext name. On the other hand, column names of secure tables are randomly generated by SecureDBaaS hence, even if different plaintext tables have columns with the same name, the names of the columns of the corresponding secure tables are different. This design choice improves confidentiality by preventing an adversarial cloud database from guessing relations among different secure tables through the identification of columns having the same encrypted name.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 operators, foreign keys, and other operations involving two columns; the two columns are encrypted through the same key.

» Database (DBC) is recommended when operations involve multiple columns in this instance, it is convenient to use the special encryption key that is generated and implicitly shared among all the columns of the database characterized by the same secure type.

The choice 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 management strategies represent an original idea because SecureDBaaS is the first architecture storing all metadata in the untrusted cloud database together with the encrypted tenant data. SecureDBaaS uses two types of metadata.

» Database metadata are related to the whole database. There is only one instance of this metadata type for each database.

» Table metadata are associated with one secure table. Each table metadata contains all information that is necessary to encrypt and decrypt data of the associated secure table.

Database metadata contain the encryption keys that are used for the secure types having the field confidentiality set to database. A different encryption key is associated with all the possible combinations of data type and encryption type. Hence, the database metadata represent a keyring and do not contain any information about tenant data. Figure 2 represents the structure of a table metadata and it contains the name of the related secure table and the unencrypted name of the related plaintext table. Moreover, table metadata include column metadata for each column of the related secure table. Each column metadata contain the following information.

» Plain name: the name of the corresponding column of the plaintext table.

» Coded name: the name of the column of the secure table. This is the only information that links a column to the corresponding plaintext column because column names of secure tables are randomly generated.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

- Secure type: This allows a SecureDBaaS client to be informed about the data type and the encryption policies associated with a column.
- Encryption key: the key used to encrypt and decrypt all the data stored in the column.

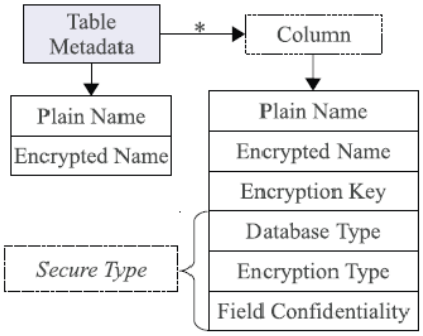

Fig. 2. Structure of table metadata.

SecureDBaaS stores metadata in the metadata storage table that is located in the untrusted cloud as the database. This is an original choice that augments flexibility, but opens two novel issues in terms of efficient data retrieval and data confidentiality. To allow SecureDBaaS clients to manipulate metadata through SQL statements, we save database and table metadata in a tabular form. Even metadata confidentiality is guaranteed through encryption.

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.

Fig. 3shows the structure of the metadata storage table. This table uses one row for the database metadata, and one row for each table metadata. Database and table metadata are encrypted through the same encryption key before being saved. This encryption key is called a master key. Only trusted clients that already know the master key can decrypt the metadata and acquire information that is necessary to encrypt and decrypt tenant data.

4 OPERATIONS

4.1 Setup Phase

Here, in this phase, We assume that the DBA creates the metadata storage table that at the beginning contains just the database metadata, and not the table metadata.

The DBA populates the database metadata through the SecureDBaaS client by using randomly generated encryption keys for any combinations of data types and encryption types, and stores them in the metadata storage table after encryption through the master key. Then, the DBA distributes the master key to the legitimate users. User access control policies are administrated by the DBA through some standard data control language as in any unencrypted database. In the following steps, the DBA creates the tables of the encrypted database. It must consider the three field confidentiality attributes (COL, MCOL, and DBC) introduced at the end of the Section 3. Let us describe this phase by referring to a simple but representative example shown in Fig. 4, where we have three secure tables named ST1, ST2, and ST3. Each table STi (i = 1, 2, 3) includes an encrypted table Ti that contains encrypted tenant data, and a table metadata Mi. (Although, in reality, the names of the columns of the secure tables are randomly generated; for the sake of simplicity, this figure refers to them through C1-CN.).

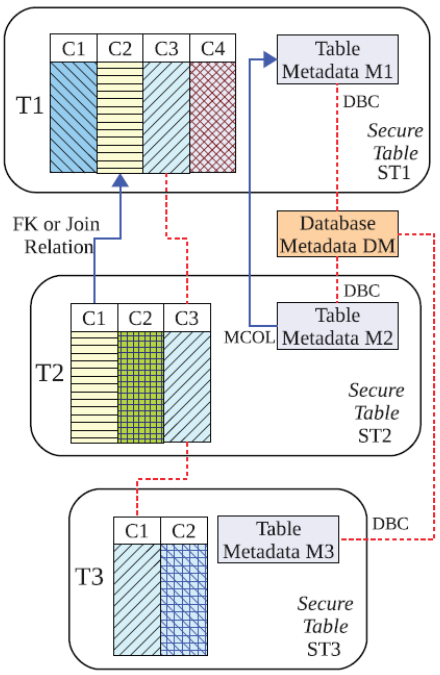

Fig. 4. Management of the encryption keys according to the field confidentiality parameter.

When operations (e.g., algebraic, order comparison) involve more than two columns, it is convenient to adopt the DBC field confidentiality. This has a twofold advantage: we can use the special encryption key that is generated and implicitly shared among all the columns of the database characterized by the same secure type, we limit possible consistency issues in some scenarios characterized by concurrent clients (see Appendix B, available in the online supplemental material).

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

For example, the columns T1.C3, T2.C3, and T3.C1 in Fig. 4 share the same secure type. Hence, they reference the database metadata, as represented by the dashed line, and use the encryption key associated with their data and encryption types. As they have the same data and encryption types T1.C3, T2.C3 and T3.C1 can use the same encryption key even if no direct reference exists between them. The database metadata already contain the encryption key K associated with the data and the encryption types of the three columns, because the encryption keys for all combinations of data and encryption types are created in the initialization phase. Hence, K is used as the encryption key of the T1.C3, T2.C3 and T3.C1 columns and copied in M1, M2, and M3.

5 EXPERIMENTAL RESULTS:

In the first set of experiments, we evaluate the overhead introduced when one SecureDBaaS client executes SQL operations on the encrypted database. Client and database server are connected through a LAN where no network latency is added. To evaluate encryption costs, the client measures the execution time of the 44 SQL commands of the TPC-C benchmark. Encryption times are reported in the histogramof the Fig. 5 that has a logarithmic Y -axis. TPC-C operations are grouped on the basis of the class of transaction: Order Status, Delivery, Stock Level, Payment, and New Order. From this figure, we can appreciate that the encryption time is below 0.1 ms for the majority of operations and below 1 ms for almost all operations but two. The exceptions are represented by two operations of the Stock Level and Payment transactions where the encryption time is two orders of magnitude higher. This high overhead is caused by the use of the order preserving encryption that is necessary for range queries.we focus on the most frequently executed SELECT, INSERT, UPDATE, and DELETE commands of the TPC-C benchmarking order to evaluate the performance overhead of encrypted SQL operations. In Figs. 6 and 7, we compare the response times of SELECT and DELETE, and UPDATE and INSERT operations, respectively. The Y -axis reports the boxplots of the response times expressed in ms (at a different scale), while the X-axis identifies the SQL operations. In SELECT, DELETE, and UPDATE operations, the response times of SecureDBaaS SQL commands are almost doubled, while the INSERT operation is as expected more critical from the computational point of view and it achieves a tripled response time with respect to the plain version. This higher overhead is motivated by the fact that an INSERT command has to encrypt all columns.

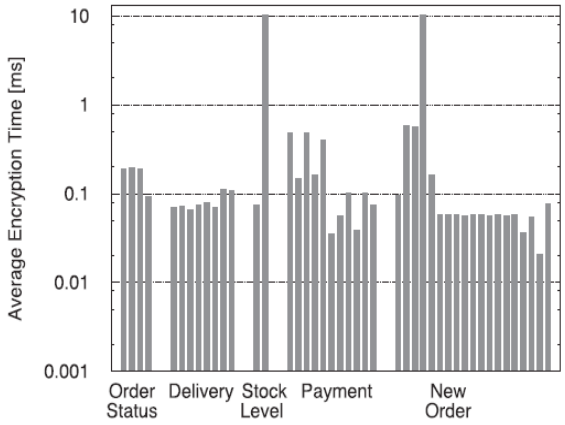

Fig. 5. Encryption times of TPC-C benchmark operations grouped by the transaction class.

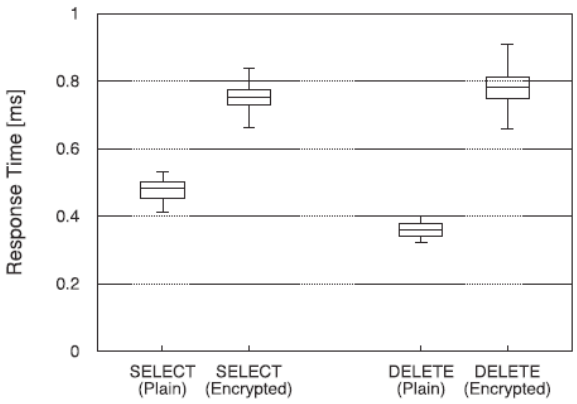

Fig. 6. Plain versus encrypted SELECT and DELETE operations.

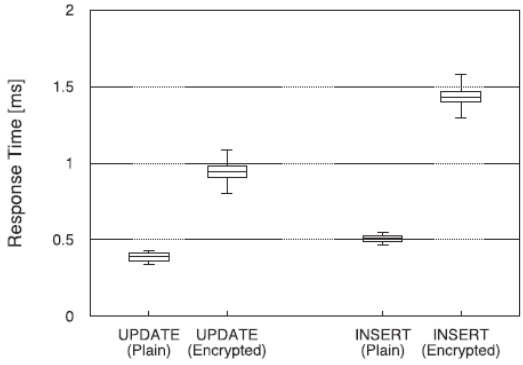

Fig. 7. Plain versus encrypted UPDATE and INSERT operations.

6 CONCLUSIONS:

Here, in this project, We propose an innovative architecture that guarantees confidentiality of data stored in public cloud databases. Our solution does not rely on an intermediate proxy that we consider a single point of failure and a bottleneck limiting availability and scalability of typical cloud database services. A large part of the research includes solutions to support concurrent SQL operations (including statements modifying the database structure) on encrypted data issued by heterogeneous and possibly geographically dispersed clients.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The proposed architecture does not require modifications to the cloud database, and it is immediately applicable to existing cloud DBaaS, such as the experimented PostgreSQL Plus Cloud Database, Windows Azure, and Xeround. There are no theoretical and practical limits to extend our solution to other platforms and to include new encryption algorithms. It is worth observing that experimental results based on the TPC-C standard benchmark show that the performance impact of data encryption on response time becomes negligible because it is masked by network latencies that are typical of cloud scenarios. In particular, concurrent read and write operations that do not modify the structure of the encrypted database cause negligible overhead. Dynamic scenarios characterized by (possibly) concurrent modifications of the database structure are supported, but at the price of high computational costs. These performance results open the space to future improvements that we are investigating.

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, "ExecutingSQL over Encrypted Data in the Database-Service-Provider Model," Proc. ACM SIGMOD Int'l Conf. Management Data, June2002.
[10] J. Li and E. Omiecinski, "Efficiency and Security Trade-Off in Supporting Range Queries on Encrypted Databases," Proc. 19thAnn. 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.

Author's Details:

Mr. S. MD Ismail, has received his M.tech degree from JNTU Hyderabad. He has been an Associate Professor for more than five years in Al Habeeb College of Engineering and Technology, Chevella, affiliated to JNTU Hyderabad. He has more than ten years of experience in the field of teaching. His areas of interesting is Cloud Computing.

Mr. Mohd Anwar Ali, has received his M.tech degree from JNTU Hyderabad. He has been an Associate Professor for more than five years, and also working as HOD in Al Habeeb College of Engineering and Technology, Chevella, affiliated to JNTU Hyderabad. He has more than ten years of experience in the field of teaching. His areas of interesting is Cloud Computing.

Mr. Syed Tousif Ahmed, pursuing an M.tech degree in Al Habeeb College of Engineering and Technology, Chevella, affiliated to JNTU Hyderabad and received his Bachelor degree from Shadan College of Engineering and Technology, Peerancheru, JNTU Hyderabad. His areas of interesting are Cloud Computing, Database Administration.