International Journal of Science Engineering and Advance Technology

Providing Distinguish Access Levels to Encrypted Cloud Databases

Mohammad Juned Ansari [1], Syed Yasin [2]
[1] M.Tech (CSE), Nimra Institute of Science and Technology, A.P., India.
[2]Head of the Department, Dept. of Computer Science & Engineering, Nimra Institute of Science and Technology, A.P., India.

***Abstract* — Outsourcing sensitive and crucial data in the hands of a cloud provider should come with the guarantee of security and seamless 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. We propose a novel architecture that integrates cloud database services with data confidentiality and the possibility of executing concurrent operations on encrypted data. 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. The 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.**

***Keywords* — Cloud, security, confidentiality, SecureDBaaS, database**

I. INTRODUCTION

Ensuring data security and confidentiality is an utmost important while placing critical information is placed in infrastructures of untrusted third parties. [1]. This requirement imposes clear data management choices: 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. Satisfying these goals has different levels of complexity depending on the type of cloud service. There are several solutions ensuring confidentiality for the storage as a service paradigm, while guaranteeing confidentiality in the database as a service (DBaaS) paradigm is still an open research area. 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 homomorphic encryption schemes [2] because of their excessive computational complexity.

The SecureDBaaS architecture is tailored to cloud platforms and does not introduce any intermediary proxy or broker server between the client and the cloud provider. Eliminating any trusted intermediate server allows SecureDBaaS to achieve the same availability, reliability, and elasticity levels of a cloud DBaaS. Other proposals (e.g., [3], [4]) based on intermediate server(s) were considered impracticable for a cloud-based solution because any proxy represents a single point of failure and a system bottleneck that limits the main benefits (e.g., scalability, availability, and elasticity) of a database service deployed on a cloud platform. Unlike SecureDBaaS, architectures relying on a trusted intermediate proxy do not support the most typical cloud scenario where geographically dispersed clients

can concurrently issue read/write operations and data structure modifications to a cloud database.

A large set of experiments based on real cloud platforms demonstrate that SecureDBaaS is immediately applicable to any DBMS because it requires no modification to the cloud database services. 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 of data encryption on response time. 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.

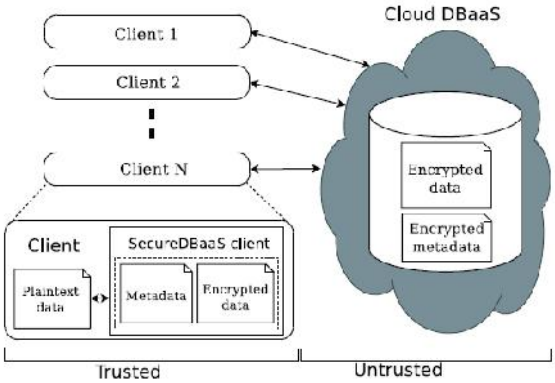

Figure 1: SecureDBaas

II. ARCHITECTURE DESIGN

SecureDBaaS is designed to allow multiple and independent clients to connect directly to the untrusted cloud DBaaS without any intermediate server. Fig. 1 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.

We assume the same security model that is commonly adopted by the literature in this field, where tenant users are trusted, the network is untrusted, and the cloud provider is honest-but-curious, that is, cloud service operations are executed correctly, but tenant information confidentiality is at risk. For these reasons, tenant data, data structures, and metadata must be encrypted before exiting from the client. A thorough presentation of the security model adopted in this paper is in Appendix A, available in the online supplemental material.

The information managed by SecureDBaaS includes plaintext data, encrypted data, metadata, and encrypted metadata. Plaintext data consist of information that a tenant wants to store and process remotely in the cloud DBaaS. 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.

SecureDBaaS moves away from existing architectures that store just tenant data in the cloud database, and save metadata in the client machine or split metadata between the cloud database and a trusted proxy [5]. When considering scenarios where multiple clients can access the same database concurrently, these previous solutions are quite inefficient. For example, saving metadata on the clients would require onerous mechanisms for metadata synchronization, and the practical impossibility of allowing multiple clients to access cloud database services independently. Solutions based on a trusted proxy are more feasible, but they introduce a system bottleneck that reduces availability, elasticity, and scalability of cloud database services.

SecureDBaaS proposes a different approach where all data and metadata are stored in the cloud database. SecureDBaaS clients can retrieve the necessary metadata from the untrusted database through SQL statements, so that multiple instances of the SecureDBaaS client can access to the untrusted cloud database independently with the guarantee of the same availability and scalability properties of typical cloud DBaaS. Encryption strategies for tenant data and innovative solutions for metadata management and storage are described in the following two sections.

Data Management

We assume that tenant data are saved in a relational database. We have to preserve the confidentiality of the stored data and even of the database structure because table and column names may yield information about saved data. We distinguish the strategies for encrypting the database structures and the tenant data.

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.

SecureDBaaS allows tenants to leverage the computational power of untrusted cloud databases by making it possible to execute SQL statements remotely and over encrypted tenant data, although remote processing of encrypted data is possible to the extent allowed by the encryption policy. To this purpose, SecureDBaaS extends the concept of data type, that is associated with each column of a traditional database by introducing the secure type. By choosing a secure type for each column of a secure table, a tenant can define fine-grained encryption policies, thus reaching the desired trade-off between data confidentiality and remote processing ability. A secure type is composed of three fields: data type, encryption type, and field confidentiality. The combination of the encryption type and of the field confidentiality parameters defines the encryption policy of the associated column.

The data type represents the type of the plaintext data (e.g., int, varchar). The encryption type identifies the encryption algorithm that is used to cipher all the data of a column. It is chosen among the algorithms supported by the SecureDBaaS implementation. As in [9], SecureDBaaS leverages several SQL-aware encryption algorithms that allow the execution of statements over encrypted data. It is important to observe that each algorithm supports only a subset of SQL operators. These features are discussed in Appendix C, available in the online supplemental material. When SecureDBaaS creates an encrypted table, the data type of each column of the encrypted table is determined by the encryption algorithm used to encode tenant data. Two encryption algorithms are defined compatible if they produce encrypted data that require the same column data type.

Metadata Management

Metadata generated by SecureDBaaS contain all the information that is necessary to manage SQL statements over the encrypted database in a way transparent to the user. 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.

This design choice makes it possible to identify which metadata type is required to execute any SQL statement so that a SecureDBaaS client needs to fetch only the metadata related to the secure table/s that is/are involved in the SQL statement. Retrieval and management of database metadata are necessary only if the SQL statement involves columns having the field confidentiality policy equal to database. This design choice minimizes the amount of metadata that each SecureDBaaS client has to fetch from the untrusted cloud database, thus reducing bandwidth consumption and processing time. Moreover, it allows multiple clients to access independently metadata related to different secure tables

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.

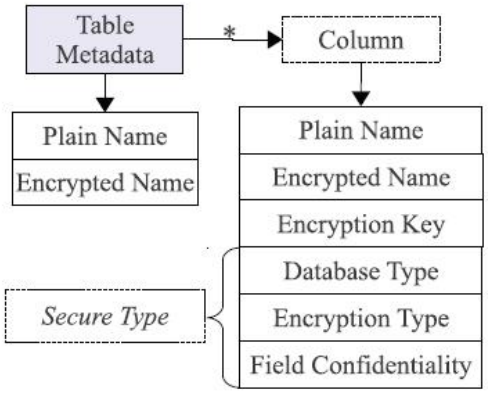

Figure 2: Structure of table metadata

The structure of a table metadata is represented in Fig. 2. Table metadata contain 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.
- Secure type: the secure type of the column, as defined in Section 3.1. 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.

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. The structure of the metadata storage table is shown in Fig. 3. This table uses one row for the database metadata, and one row for each table metadata.

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)

Figure 3 Organisation of database metadata and table metadata

III. RELATED WORK

SecureDBaaS provides several original features that differentiate it from previous work 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.

Cryptographic file systems and secure storage solutions represent the earliest works in this field. We do not detail the several papers and products (e.g., Sporc [6], Sundr [7], Depot [8]) because they do not support computations on encrypted data.

IV. CONCLUSION

We propose an innovative architecture that guarantees confidentiality of data stored in public cloud databases. Unlike state-of-the-art approaches, 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 heterogenous and possibly geographically dispersed clients. 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.

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] 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.

[9] 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.