© January 2016 | IJIRT | Volume 2 Issue 8 | ISSN: 2349-6002

Scalable architecture for multi-user encrypted SQL operations on cloud database services

A. Sheelavathi, K. Preethi
Assistant Professor/IT, Assistant Professor/CSE

***Abstract* — The success of the cloud database paradigm is strictly related to strong guarantees in terms of service availability, scalability and security, but also of data confidentiality. Any cloud provider assures the security and availability of its platform, while the implementation of scalable solutions to guarantee confidentiality of the information stored in cloud databases is an open problem left to the tenant. Existing solutions address some preliminary issues through SQL operations on encrypted data. We propose the first complete architecture that combines data encryption, key management, authentication and authorization solutions, and that addresses the issues related to typical threat scenarios for cloud database services. Formal models describe the proposed solutions for enforcing access control and for guaranteeing confidentiality of data and metadata. Experimental evaluations based on standard benchmark sand real Internet scenarios show that the proposed architecture satisfies also scalability and performance requirements.**

***Index Terms*— Database, Confidentiality, Encryption, Access Control**

I INTRODUCTION

The diffusion of cloud database services is being hindered by the perception of confidentiality risks when we store our information in cloud infrastructures. Cryptographic solutions address this issue in the context of file storage when there is no need to perform computations over encrypted data. We aim, instead, to guarantee data confidentiality and data isolation for cloud databases that represent an open research area.

There are three main related issues behind these two problems: execution of SQL operators over encrypted data; enforcement of access control mechanisms through selective encryption strategies; design of architectures not penalizing the performance and scalability that are typical of cloud-based services . Existing proposals offer partial and separate solutions to data confidentiality and isolation.

For example, architectures supporting SQL operations on encrypted data leave access control to the cloud provider or enforce it through an intermediate trusted server. Other proposed architectures solve the problem of access control without the intervention of the cloud provider, but they do not allow execution of SQL operations on encrypted data. We propose the first architecture, called *Multi-User relaTional Encrypted DataBase* (MuteDB), that guarantees data confidentiality by executing SQL operations on encrypted data and by enforcing access control policies through selective encryption methods. By combining these two approaches MuteDB is the only solution ensuring confidentiality of data stored in the cloud even in the worst threat scenario where legitimate database users collude with cloud provider employees. This result is achieved through an innovative model that translates access control policies related to a plaintext database into selective encryption strategies that are applied to the corresponding encrypted database. Our solution works even in dynamic scenarios, in which users and access control policies change over time, without the need to renew and redistribute user credentials. The proposed architecture is specifically designed for cloud database scenarios where multiple users can access the cloud database through the Internet possibly from different geographical areas.

Special attention in the architectural design is devoted to guarantee the same availability and scalability of a plaintext cloud database. For this reason, MuteDB does not rely on any intermediate trusted server that could become a system bottleneck and a single point of failure. Moreover, it adopts innovative solutions for guaranteeing efficient retrieval of database metadata that are stored in an encrypted form in the cloud database. We can consider MuteDB as the first architecture that allows enterprises to leverage cloud database services while achieving the same

confidentiality guarantees of a traditional in-house database and the same scalability of a cloud database service. The performance and scalability of MuteDB are evaluated through a prototype that is subject to different query workloads based on standard (TPC-C) and recently proposed (YCSB) database benchmarks. We highlight that, as a further contribution, this paper reports the first performance evaluation studies related to encrypted cloud database services in real distributed environments where the clients are geographically distributed over the PlanetLab platform. Experimental results show that MuteDB does not affect the scalability of the original cloud service, and its performance for geographically distributed clients are comparable to those of encrypted cloud database services.

II LITERATURE REVIEW

We propose an architecture guaranteeing confidentiality and isolation of data stored in cloud database infrastructures that are subject to two types of threats: those related to specific roles, and those deriving by the collusion of these roles. The literature focuses on the former threats, while our proposal aims to respond to both classes. Typical threat models in literature identify the possible issues related to four roles: the tenant Database Administrator (DBA), the tenant database users, the cloud provider employees, and people external to tenant's and provider's organizations. We describe our assumptions based on the four roles and then we consider collusion.

The *DBA* is the only role that has access to all tenant data. He is in charge of installing and configuring the database, implementing the *access control policies* and managing the *user's credentials*. As in related literature, our threat model assumes that the DBA is trusted. Possible measures to verify the loyalty of the DBA, such as hashed logging, continuous monitoring and supervision, are outside the scope of this paper.

External attackers have no legitimate access to the infrastructure and data of the tenant organization nor to those of the cloud provider. They can try to access tenant information through several types of attack: by eavesdropping data in motion between the tenant clients and the cloud servers, by compromising the cloud servers and/or the tenant clients.

The *cloud insiders* are employees of the cloud provider that have access to the cloud infrastructure hosting the database service of the tenant organization. Their behavior is *honest but curious*, that is, they may be interested in accessing tenant data, but they do not modify or delete them. This assumption is considered realistic in all related literature and the motivation should be clear. While reading data would remain unnoticed by a tenant, the detection of any data modification would penalize the trust and reputation of the cloud provider in the eyes of all of its customers. *Tenant insiders* refer to database users having legitimate access to a subset of the tenant data stored in the cloud database. The portion of accessible data is defined by the access control policies of the tenant organization. Tenant insiders may try to gain access to more information by escalating their privileges through a violation of the access control policies.

Guaranteeing data confidentiality in the cloud against external attackers, cloud insiders, and tenant insiders under the assumption that they do not collude can be achieved through some combinations of existing solutions. For example, best practices in the field of authentication and secure communication protocols hinder external attacks. Recent SQL-aware cryptographic strategies allow a tenant to store encrypted data thus preventing cloud insiders and external attackers from reading tenant data. Standard database access control mechanisms, such as privilege GRANTS and reference monitors, limit the operations of tenant insiders within their legitimate authorizations. Existing access control mechanisms at the database engine side guarantee confidentiality and isolation in traditional in-house deployments where the infrastructure is managed by trusted personnel, but they do not work as well for cloud database services because they do not consider the main threats posed by a collusion between a cloud and a tenant insiders when data are encrypted through a global master key.

In a cloud database scenario, the malicious operations of a tenant insider are limited by access control policies, but these policies cannot prevent the possibility that a tenant insider discloses its credentials including its decryption key(s) to a cloud insider. The latter, that has access to all the encrypted data and can bypass the access control policies enforced at the cloud side, can violate the

confidentiality of the entire database by means of the key(s) received by the tenant insider.

A second collusion scenario may happen if a cloud insider delivers some encrypted data to a tenant insider that is not authorized to access them. In this scenario the tenant insider can leverage its credentials to decrypt all encrypted data, thus violating the tenant access control policies. Let us anticipate a summary of the design choices and novel solutions that allow MuteDB to protect data against *external attackers*, *cloud insiders* and *tenant in- siders*, and against collusion between these roles. External attackers that eavesdrop network traffic cannot access any plaintext information because SQL operations issued to the cloud database are protected by using standard encryption protocols (e.g., SSL). Cloud insiders and external attackers that have breached the cloud servers cannot access confidential information, because MuteDB encrypts tenant data with SQL-aware encryption algorithms and the cloud provider never obtains the decryption keys. Tenant insiders cannot perform privilege escalation attacks on the encrypted database thanks to a novel scheme that translates and enforces the database access control policies defined by the tenant DBA on the plaintext database to the encrypted one. Even in the worst case of a collusion between tenant and cloud insiders, the proposed solution limits the data leakage to the amount of information that is accessible to the colluding tenant insider, because MuteDB does not delegate the enforcement of access control policies to the cloud provider.

III ALGORITHMS

RSA derives its security from the difficulty of factoring large integers that are the product of two large prime numbers. Multiplying these two numbers is easy, but determining the original prime numbers from the total -- factoring -- is considered infeasible due to the time it would take even using today's super computers.

Mathametical formula:

Encryption :

$$c = \text{ENCRYPT}(m) = m^e \bmod n .$$

Decryption:

$$m = \text{DECRYPT}(c) = c^d \bmod n .$$

Diffie-Hellman key exchange, also called exponential key exchange, is a method of digital encryption that uses numbers raised to specific powers to produce decryption keys on the basis of components that are never directly transmitted, making the task of a would-be code breaker mathematically overwhelming. To implement Diffie-Hellman, the two end users Alice and Bob, while communicating over a channel they know to be private, mutually agree on positive whole numbers p and q, such that p is a prime number and q is a generator of p. The generator q is a number that, when raised to positive whole-number powers less than p, never produces the same result for any two such whole numbers. The value of p may be large but the value of q is usually small.

Mathametical formula:

$$(\mathbf{g}^{\mathbf{x}} \textit{ mod } \mathbf{p})^{\mathbf{y}} \textit{ mod } \mathbf{p} = \mathbf{g}^{\mathbf{xy}} \textit{ mod } \mathbf{p}$$

$$(\mathbf{g}^{\mathbf{y}} \textit{ mod } \mathbf{p})^{\mathbf{x}} \textit{ mod } \mathbf{p} = \mathbf{g}^{\mathbf{yx}} \textit{ mod } \mathbf{p}$$

***A.*Implementation:**

Plaintext database model: Plaintext most commonly meant message text in the language of the communicating parties. Since computers became commonly available. the original definition implied that the message could be read by a human being, the modern definition emphasizes that a person using a computer could easily interpret the data. Any information which the communicating parties wish to conceal from others can now be treated, and referred to, as plaintext. Thus, in a significant sense, plaintext is the 'normal' representation of data before any action has been taken to conceal, compress, or 'digest' it.

It need not represent text, and even if it does, the text may not be "plain". Plaintext is used as input to an encryption algorithm; the output is usually termed cipher text particularly when the algorithm is a cipher. Code text is less often used, and almost always only when the algorithm involved is actually a code. In some systems, however, multiple layers of encryption are used, in which case the output of one encryption algorithm becomes plaintext input for the next. The proposed plaintext database model is a poset that extends the structure poset S, with the resources R, a structure s<S associated with a resource r< R is a parent of the resource r (s > r).

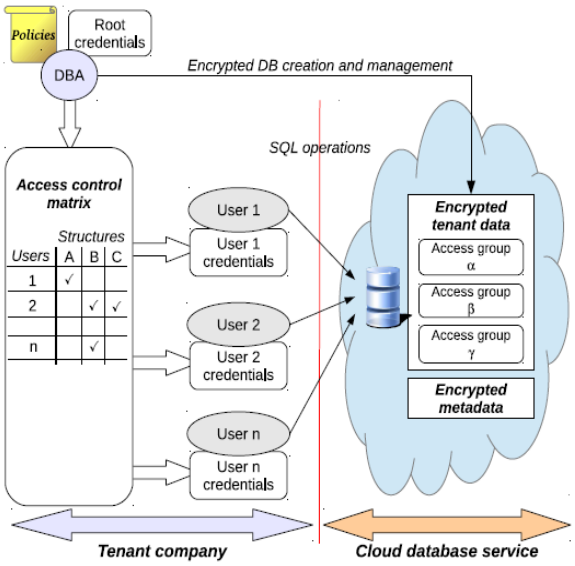

Figure 1. Architecture diagram

Access control: it is a way of limiting access to a system or to physical or virtual resources. In computing, access control is a process by which users are granted access and certain privileges to systems, resources or information. In access control systems, users must present credentials before they can be granted access. In physical systems, these credentials may come in many forms, but credentials that can't be transferred provide the most security. The management of admission to system and network resources. It grants authenticated users access to specific resources based on access policies and the permission level assigned to the user or user group. Access control often includes authentication, which proves the identity of the user or client machine attempting to access the files. the MuteDB models and schemes for combining encryption and key management to support data confidentiality and isolation in cloud data bases.

The presentation of the models related to access control in plaintext and encrypted databases, we describe how MuteDB transforms an access control matrix for the plaintext model to a matrix suitable for the encrypted database, and how it generates user credentials. Let R be the set of resources that represent plain text tenant data, S the set of plaintext database structures, E the set of encrypted tenant data, U the set of users, and K the set of encryption keys. We define A as the access control matrix where, for each user u P U and for each structure s P S, there exists a binary authorization rule a that defines whether an access to s by u is denied or allowed.

Encrypted database model: Database encryption is the process of converting data, within a database, In plaintext format into meaningless cipher text by the means of a suitable algorithm. Database decryption is converting the meaningless cipher text into the original information using keys generated by the encryption algorithms. Database encryption be provided at the file or column level. Encryption of a database is costly and requires more storage space than the original data. The steps in encrypting a database are: Determine the criticality of the need for encryption, Determine what data needs to be encrypted, Determine which algorithms best suit the encryption standard, Determine how the keys will be managed. Numerous algorithms are used for encryption. These algorithms generate keys related to the encrypted data. These keys set a link between the encryption and decryption procedures. The encrypted data can be decrypted only by using these keys.

Encrypted data are contained in encrypted tables stored in cloud database servers. For each plaintext table, the MuteDB DBA client generates the corresponding encrypted table and a unique encryption key. The name of the encrypted table is computed by encrypting the name of the plaintext table through that key. The encryption algorithm used for encrypting the table names is a standard AES algorithm in a deterministic mode (e.g., CBC with constant initialization vector). In such a way, only the users that know the plaintext table name and the corresponding encryption key are able to compute the name of the encrypted table. The deterministic scheme is preferred because it allows a correspondence between plaintext and encrypted tables and improves the efficiency of the query translation process.

Metadata management: Database metadata include all information allowing a Mute DB client to translate plaintext SQL operations into operations working on the encrypted database. We describe the original solutions adopted by Mute DB to manage metadata. Existing proposals use trusted infrastructures to store and distribute metadata information or require database users to maintain

them locally . These schemes simplify metadata management, but they limit scalability and availability of a cloud database service. The Mute DB alternative is to store metadata in the cloud database together with encrypted tenant data. This approach allows each client to access metadata directly and concurrently through standard SQL operations, thus avoiding system bottlenecks and single point of failures at the tenant side. Metadata contain sensitive information, hence it is necessary to store them in an encrypted form. Unlike the proposals of the same authors in which all users are provided with the same master encryption key , Mute DB proposes a new metadata management strategy that enforces access control policies at the encryption level, by generating a different encryption key for each user and by ensuring that each user is able to decrypt all and only encrypted tenant data on which he/she has legitimate access.

MuteDB: The Mute DB DBA client, that is the application for the creation and management of the encrypted database. All tenant database users can issue SQL operations directly to the cloud database even from geographically distributed locations by executing a Mute DB client on their machines. The entire set of *tenant data* are stored in an encrypted form in the cloud database. Thanks to the use of SQL-aware encryption strategies, the cloud database engine can execute queries on encrypted data without accessing any decryption keys. Even *metadata* that are necessary to manage encryption strategies are considered critical information, hence Mute DB stores them encrypted in the cloud database: the DBA and the tenant users can efficiently retrieve metadata through standard SQL queries. We refer to the encrypted forms of tenant data and metadata as encrypted tenant data and encrypted metadata.

Scope:Rather than run multiple database servers/VMs on the same machine, which wastes space and system resources, each node runs a single database server. Tenants can load databases onto servers, and databases can be partitioned for load balancing. The partitioning strategy is workload aware, and partitions are migrated as necessary when workloads change. Workload monitoring includes tracking resource usage, predicting combined resource requirements, and consolidating workloads to minimize the total number of machines required while not exceeding machine capacities. To protect user privacy, the authors briefly discuss CryptDB, which implements adjustable security by storing only encrypted data.

IV CONCLUSION

In this paper we propose MuteDB, a novel architecture for cloud database services that guarantees for the first time data confidentiality through SQL-aware encryption algorithms and data isolation through access control enforcement based on encryption and key derivation techniques. These solutions allow MuteDB to address threat issues that are relevant for cloud services including risks of information leakage due to collusions between cloud provider employees and tenant users.

The most important solutions are described through formal models, while the feasibility, performance and scalability of the proposed architecture are demonstrated through a large set of experiments carried out through a prototype deployed in a real Internet-based environment where cloud database services are accessed concurrently by geographically distributed clients. All results confirm that for realistic workloads, the MuteDB architecture achieves performance and scalability comparable to those of unencrypted cloud database services. Ongoing work is focused on integrating private information retrieval solutions in MuteDB with the goal of preventing information leakage caused by access pattern analyses, and novel architectural solutions for hybrid cloud environments.

V FUTURE ENHANCEMENT

In personal computing devices that rely on a cloud storage environment for data backup until the arrival of the Diffie-Hellman key exchange and RSA algorithms, governments and their armies were the only real users of encryption. However, Symmetric key cryptography, Diffie-Hellman and RSA led to the broad use of encryption in the file user to upload file. Since public-key algorithms tend to be much slower than symmetric-key algorithms, modern systems such as TLS and SSH use a combination of the two: one party receives the other's public key, and encrypts a small piece of data (either a symmetric key or some data used to generate it). The remainder of the conversation uses a (typically faster) symmetric-key algorithm for encryption. User upload file not directly store

in cloud send cloud admin it allow that time only store in cloud and download the data.

REFERENCES

[1] S. Pearson and A. Benameur, "Privacy, security and trust issues arising from cloud computing," in *Proc. 2010 IEEE Int'l Conf. Cloud Computing Technology and Science*, Nov.-Dec. 2010, pp. 693
.

[2] L. M. Vaquero, L. Rodero-Merino, and R. Buyya, "Dynamically scaling applications in the cloud," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, pp. 45–52, 2011.2168-7161 (c) 2013.

[3] L. Ferretti, M. Colajanni, and M. Marchetti, "Distributed, concurrent, and independent access to encrypted cloud databases," *IEEE Trans. Parallel and Distributed Systems*, vol. 25, no. 2, pp. 437–446, 2014.

[4] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: protecting confidentiality with encrypted query processing," in *Proc. 23rd ACM Symp. Operating Systems Principles*, Oct. 2011, pp. 85–100.

[5] E. Damiani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Key management for multi-user encrypted databases," in *Proc. ACM Workshop Storage Security and Survivability*, Nov. 2005, pp. 74 – 83.

[6] G. Wang, Q. Liu, J. Wu, and M. Guo, "Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers," *Computers & Security*, vol. 30, no. 5, pp. 320–331, 2011.

[7] M. R. Asghar, G. Russello, B. Crispo, and M. Ion, "Supporting complex queries and access policies for multi-user encrypted databases," in *Proc. 2013 ACM Workshop on Cloud computing security*, Nov. 2013, pp. 77–88.

[8] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broadcoverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, 2003.

[9] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Appli- cations*. Cambridge university press, 2004.

[10] H. Hacig¨um¨ us¸, B. Iyer, C. Li, and S. Mehrotra, "Executing sql over encrypted data in the database-service-provider model," in *Proc. 2002 ACM SIGMOD Int'l Conf. Management of data*, Jun. 2002, pp.216–227.