Scientific Journal Impact Factor (SJIF) : 1.711
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745 ISSN (PRINT) : 2393 - 8161

Privacy and Security Enhancement for Cloud Database Services

S. Priya[1]And Dr. S. Dhanalakshmi[2]
[1]*M.Phil Full Time Research Scholar, Department of Computer Science*
[2] *prof&Head of Department, Department Of Computer Science and Applications*

Abstract-Cloud computing environment provides different types of resources and services to the users. Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) are the basic cloud service models. Database services are also provided under the cloud is categorized into DataBase as a Service (DBaaS). Security and access control policies are integrated with the database services. Multiple user access is managed under the Secure DBaaS. Database elements and their details are also maintained in encrypted form. All the encryption operations are carried out with the support different Cryptography techniques. Table properties are also maintained in encrypted form. Database clients are allowed to access the database from various geographical locations. Simultaneous data access is allocated for the clients. Symmetric and asymmetric cryptography methods are applied in the system. Database data are concurrently modified by the clients.

Database structure modification is integrated with the Secure DBaaS mechanism. Data verification mechanism is merged with the system to verify the storage and transmission operations. The data query values are also protected with encryption techniques. Client privilege management model is integrated to handle the access control process.

I. Introduction

A cloud database is a database that typically runs on a cloud computing platform, such AmazonEC2, GoGrid, Salesforce, Rackspace and Microsoft Azure. There are two common deployment models: users can run databases on the cloud independently, using a virtual machine image, or they can purchase access to a database service, maintained by a cloud database provider. Of the databases available on the cloud, some are SQL-based and some use a NoSQL data model. There are two primary methods to run a database on the cloud: Virtual machine Image - cloud platforms allow users to purchase virtual machine instances for a limited time. It is possible to run a database on these virtual machines. Users can either upload their own machine image with a database installed on it, or use ready-made machine images that already include an optimized installation of a database. For example, Oracle provides a ready-made machine image with an installation of Oracle Database 11g Enterprise Edition on Amazon EC2 and on Microsoft Azure.

Database as a service (DBaaS) - some cloud platforms offer options for using a database as a service, without physically launching a virtual machine instance for the database. In this configuration, application owners do not have to install and maintain the database on their own. Instead, the database service provider takes responsibility for installing and maintaining the database and application owners pay according to their usage. For example, Amazon Web Services provides three database services as part of its cloud offering, SimpleDB, a NoSQL key-value store, Amazon Relational Database Service, an SQL-based database service with a MySQL interface and DynamoDB. Similarly, Microsoft offers the Azure SQL Database service as part of its cloud offering. A third option is managed database hosting on the cloud, where the database is not offered as a service, but the cloud provider hosts the database and manages it on the application owner's behalf. For example, cloud provider Rack space offers managed hosting for MySQL on dedicated and cloud architectures and NoSQL databases via Object Rocket's managed MongoDB service.

It is also important to differentiate between cloud databases which are relational as opposed to non-relational or NoSQL: SQL database, such as NuoDB, Oracle Database, Microsoft SQL Server and MySQL, are one type of database which can be run on the cloud. SQL databases are difficult to scale, meaning they are not natively suited to a cloud environment, although cloud database services based on SQL are attempting to address this challenge. NoSQL databases, such as Apache Cassandra, CouchDB and MongoDB, are another type of database which can run on the cloud. NoSQL databases are built to service heavy read/write loads and are able scale up and down easily and therefore they are more natively suited to running on the cloud. Most contemporary applications are built around an SQL data model, so working with NoSQL databases often requires a complete rewrite of application code.

II. Related Work

Search and queries over encrypted data. Song et al. describe cryptographic tools for performing keyword search over encrypted data, which we use to implement SEARCH. Amanatidis et al. propose methods for exact searches that do not require scanning the entire database and could be used to process certain restricted SQL queries. Bao et al. extend these encrypted search methods to the multi-user case. Yang et al. run selections with equality predicates over encrypted data. Evdokimov and Guenther present methods for the same selections, as well as Cartesian products and projections. Agrawal et al. develop a statistical encoding that preserves the order of numerical data in a column, but it does not have sound cryptographic properties, unlike the scheme we use [4]. Boneh and Waters show public-key schemes for comparisons, subset checks and conjunctions of such queries over encrypted data, but these schemes have ciphertext lengths that are exponential in the length of the plaintext, limiting their practical applicability. When applied to processing SQL on encrypted data, these techniques suffer from some of the following limitations: certain basic queries are not supported or are too inefficient, they require significant client-side query processing, users either have to build and maintain indexes on the data at the server or to perform sequential scans for every selection/search and implementing these techniques requires unattractive changes to the innards of the DBMS.

Some researchers have developed prototype systems for subsets of SQL, but they provide no confidentiality guarantees, require a significant DBMS rewrite and rely on client-side processing [9]. For example, Hacigumus et al. heuristically split the domain of possible values for each column into partitions, storing the partition number unencrypted for each data item and rely on extensive client-side filtering of query results. Chow et al. [8] require trusted entities and two non-colluding untrusted DBMSes. Untrusted servers. SUNDR uses cryptography to provide privacy and integrity in a file system on top of an untrusted file server. Using a SUNDR-like model, SPORC [6] and Depot [5] show how to build low-latency applications, running mostly on the clients, without having to trust a server. Existing server-side applications that involve separate database and application servers cannot be used with these systems unless they are rewritten as distributed client-side applications to work with SPORC or Depot. Many applications are not amenable to such a structure.

Companies like Navajo Systems and Ciphercloud provide a trusted application-level proxy that intercepts network traffic between clients and cloud hosted servers and encrypts sensitive data stored on the server. These products appear to break up sensitive data into tokens and encrypt each of these tokens using an order-preserving encryption scheme, which allows token-level searching and sorting. In contrast, CryptDB supports a richer set of operations, reveals only relations for the necessary classes of computation to the server based on the queries issued by the application and allows chaining of encryption keys to user passwords, to restrict data leaks from a compromised proxy. Disk encryption. Various commercial database products, such as Oracle's Transparent Data Encryption [3], encrypt data on disk, but decrypt it to perform query processing. As a result, the server must have access to decryption keys and an adversary compromising the DBMS software can gain access to the entire data.

Software security. Many tools help programmers either find or mitigate mistakes in their code that may lead to vulnerabilities, including static analysis tools like PQL and UrFlow and runtime tools like Resin [2] and CLAMP [1]. In contrast, CryptDB provides confidentiality guarantees for user data even if the adversary gains complete control over the application and database servers. These tools provide no guarantees in the face of this threat, but in contrast, CryptDB cannot provide confidentiality in the face of vulnerabilities that trick the user's client machine into issuing unwanted requests. As a result, using CryptDB together with these tools should improve overall application security.

Rizvi et al. and Chlipala [7] specify and enforce an application's security policy over SQL views. CryptDB's SQL annotations can capture most of these policies, except for result processing being done in the policy's view, such as allowing a user to view only aggregates of certain data [12]. Unlike prior systems, CryptDB enforces SQL-level policies cryptographically, without relying on compile-time or run-time permission checks. Privacy-preserving aggregates. Privacy-preserving data integration, mining and aggregation schemes are useful but are not usable by many applications because they support only specialized query types and require a rewrite of the DBMS. Differential privacy is complementary to CryptDB; it allows a trusted server to decide what answers to release and how to obfuscate answers to aggregation queries to avoid leaking information about any specific record in the database.

Query integrity. Techniques for SQL query integrity can be integrated into CryptDB because CryptDB allows relational queries on encrypted data to be processed just like on plaintext. These methods can provide integrity by adding a MAC to each tuple, freshness using hash chains and both freshness and completeness of query results. In addition, the client can verify the results of aggregation queries and provide query assurance for most read queries. Outsourced databases. Curino et al. advocate the idea of a relational cloud [11], a context in which CryptDB fits well.

III. Cloud Database Services and Its Issues

In a cloud context, where critical information is placed in infrastructures of untrusted third parties, ensuring data confidentiality is of paramount importance. 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 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 based on intermediate server 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 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.

Secure DBaaS is designed to allow multiple and independent clients to connect to the cloud without intermediate server. Data, data structures and metadata are encrypted before upload to the cloud. Multiple cryptography techniques are used to convert plain text into encrypted data. Table names and their column names are also encrypted in the cloud database security scheme. The system supports geographically distributed clients to connect directly to an encrypted cloud database. The client can perform concurrent query processing on encrypted databases. RSA and Advanced Encryption Standard (AES) are used in the system. Concurrent data and data structure modification are allowed on the cloud databases. The following issues are identified from the current cloud database security methods. Database structure modification increases the computational overhead. Data integrity verification is not performed. Access control mechanism is not provided. Query submission is not secured.

IV. Secure DBaaS

Secure DBaaS is designed to allow multiple and independent clients to connect directly to the untrusted cloud DBaaS without any intermediate server. We assume that a tenant organization acquires a cloud database service from an untrusted DBaaS provider. The tenant then deploys one or more machines and installs a Secure DBaaS 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. The same security model that is commonly adopted where tenant users are trusted, the network is untrusted and the cloud provider is honest-but-curious, cloud service operations are executed correctly, but tenant information confidentiality risk. 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. 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 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 [11]. When considering scenarios where multiple clients can access the same database concurrently previous solutions are quite inefficient. 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.

4.1. Sequential SQL Operations

We describe the SQL operations in SecureDBaaS by considering an initial simple scenario in which we assume that the cloud database is accessed by one client. Our goal here is to highlight the main processing steps; we do not take into account performance optimizations and concurrency. The first connection of the client with the cloud DBaaS is for authentication purposes. SecureDBaaS relies on standard authentication and authorization mechanisms provided by the original DBMS server. After the authentication, a user interacts with the cloud database through the SecureDBaaS client. SecureDBaaS analyzes the original operation to identify which tables are involved and to retrieve their metadata from the cloud database. The metadata are decrypted through the master key and their information is used to translate the original plain SQL into a query that operates on the encrypted database.

Translated operations are then executed by the cloud database over the encrypted tenant data. As there is a one-to-one correspondence between plaintext tables and encrypted tables, it is possible to prevent a trusted database user from accessing or modifying some tenant data by granting limited privileges on some tables. User privileges can be managed directly by the untrusted and encrypted cloud database. The results of the translated query that includes encrypted tenant data and metadata are received by the SecureDBaaS client, decrypted and delivered to the user. The complexity of the translation process depends on the type of SQL statement.

4.2. Concurrent SQL Operations

The support to concurrent execution of SQL statements issued by multiple independent clients is one of the most important benefits of SecureDBaaS with respect to state-of-the-art solutions. Our architecture must guarantee consistency among encrypted tenant data and encrypted metadata because corrupted or out-of-date metadata would prevent clients from decoding encrypted tenant data resulting in permanent data losses. A thorough analysis of the possible issues and solutions related to concurrent SQL operations on encrypted tenant data available in the online supplemental material. We remark the importance of distinguishing two classes of statements that are supported by SecureDBaaS: SQL operations not causing modifications to the database structure, such as read, write and update; operations involving alterations of the database structure through creation, removal and modification of database tables.

In scenarios characterized by a static database structure, SecureDBaaS allows clients to issue concurrent SQL commands to the encrypted cloud database without introducing any new consistency issues with respect to unencrypted databases. After metadata retrieval, a plaintext SQL command is translated into one SQL command operating on encrypted tenant data. As metadata do not change, a client can read them once and cache them for further uses, thus improving performance. SecureDBaaS is the first architecture that allows concurrent and consistent accesses even when there are operations that can modify the database structure. In such cases, we have to guarantee the consistency of data and metadata through isolation levels, such as the snapshot isolation that we demonstrate can work for most usage scenarios.

V. Privacy and Security Enhancement for Cloud Databases

The Secure DBaaS framework is enhanced to support concurrent database structure modification scheme with minimum overhead. Digital signature based data integrity verification mechanism is integrated with the system. Encrypted query submission model is used to secure the query values. Access control mechanism is used to allow users to grant permissions for other users.

The Secure DBaaS framework is enhanced to support concurrent database structure modification scheme with minimum overhead. Digital signature based data integrity verification mechanism is

integrated with the system. Encrypted query submission model is used to secure the query values. Access control mechanism is used to allow users to grant permissions for other users. Cloud database security scheme is enhanced with data verification mechanism. Access privilege management scheme is integrated with the system. Encrypted query processing is used to secure the user query values. The system is divided into six major modules. They are Cloud Database, Key Management, Data Upload Process, Privilege Management, Query Processing and Database Reconstruction. The cloud database module is designed to manage client data values. Key distribution process is handled under key management module. Table creation and update operations are carried out under the data upload process. User access permissions are assigned under the privilege management module. Query processing module is designed to execute encrypted queries in the cloud database. Database structure modifications are performed under the database reconstruction module.

User accounts are created and maintained under the cloud database. The system separately manages the user data and meta data values. Data values are stored in encrypted format. Query processing is performed under the cloud database environment. The key management process is used to maintain the key values. Multiple Crypto System (MCS) is used for the key management process. Separate key values are used for user data and meta data. Cloud database issues the key value to secure the query submission process.

Client application is used for the data upload process. Database tables are created and maintained in the data upload process. Meta data and records are encrypted before the upload process. Encrypted table data values are stored in the user storage area under the cloud database. User access permissions are assigned in the privilege management process. Privileges are assigned for the tables. Insert, delete, update and select permissions are used in the data sharing process. Re-encryption process is applied in the data sharing process.

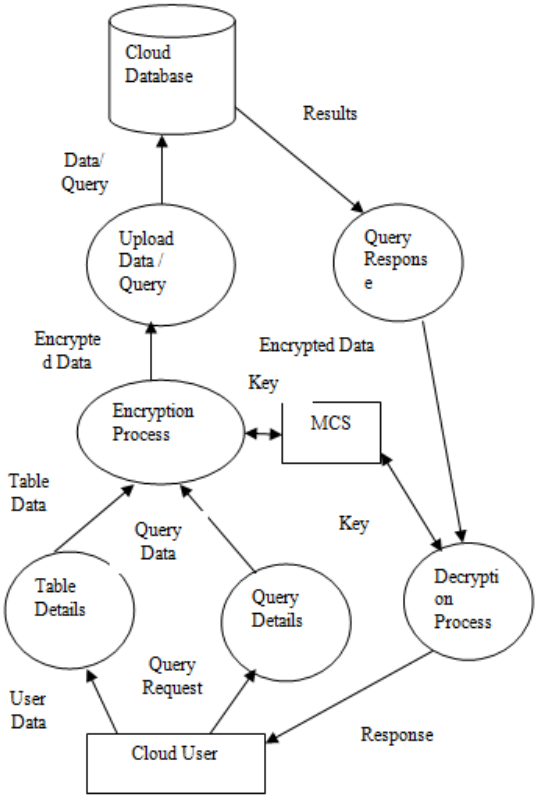

Fig. No: 5.1. Privacy and Security Enhancement for Cloud Databases

Data download operations are carried out using query submission process. Encrypted query values are submitted to the cloud database. Cloud database prepare the query response and transfer to the client. Response data values are decrypted using the client secret key. Database tables are modified in the database reconstruction process. Column addition and deletion operations are supported in the reconstruction process. Concurrent user access is allowed in the database. Table data values are reassigned in the database manipulation process.

VI. Conclusion and Future Enhancement

Cloud database services are integrated with data confidentiality and concurrent access models. Secure database as a service (Secure DBaaS) Framework is used to manage data access in encrypted cloud databases. The Secure DBaaS scheme is enhanced with data integrity features. Concurrent database structure modification and query security tasks are improved with security methods. The system eliminates the intermediate proxies in database management process. Database structure modification mechanism is adopted for multi user environment. The system improves the availability and scalability features. The response time in query processing is reduced by the system. The system can be enhanced with the following features. The Cloud Database service system can be enhanced to provide commercial database service activities. The system can be integrated with resource allocation and task scheduling schemes.

References

[1] B. Parno, J. M. McCune, D. Wendlandt, D. G. Andersen and A. Perrig. CLAMP: Practical prevention of large-scale data leaks. In Proceedings of the 30th IEEE Symposium on Security and Privacy, Oakland, CA, May 2009.

[2] A. Yip, X.Wang, N. Zeldovich and M. F. Kaashoek. Improving application security with data flow assertions. In Proceedings of the 22nd ACM Symposium on Operating Systems Principles, pages 291–304, Big Sky, MT, October 2009.

[3] Oracle Corporation. Oracle advanced security. http: //www.oracle.com/technetwork/database/options/ advanced-security/. Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich and Hari Balakrishnan, "CryptDB: Protecting Confidentiality with Encrypted Query Processing", ACM, 2011.

[4] A. Boldyreva, N. Chenette, Y. Lee and A. O'Neill. Orderpreserving symmetric encryption. In Proceedings of the 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), Cologne, Germany, April 2009.

[5] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin and M. Walfish. Depot: Cloud storage with minimal trust. In Proceedings of the 9th Symposium on Operating Systems Design and Implementation, Vancouver, Canada, October 2010.

[6] A. J. Feldman, W. P. Zeller, M. J. Freedman and E. W. Felten. SPORC: Group collaboration using untrusted cloud resources. In Proceedings of the 9th Symposium on Operating Systems Design and Implementation, Vancouver, Canada, October 2010.

[7] A. Chlipala. Static checking of dynamically-varying security policies in database-backed applications. In Proceedings of the 9th Symposium on Operating Systems Design and Implementation, Vancouver, Canada, October 2010.

[8] S. S. M. Chow, J.-H. Lee and L. Subramanian. Two-party computation model for privacy-preserving queries over distributed databases. In Proceedings of the 16th Network and Distributed System Security Symposium, February 2009.

[9] V. Ciriani, S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi and P. Samarati. Keep a few: Outsourcing data while maintaining confidentiality. In Proceedings of the 14th European Symposium on Research in Computer Security, September 2009

[10] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich and Hari Balakrishnan, "CryptDB: Protecting Confidentiality with Encrypted Query Processing", ACM, 2011.

[11] C. Curino, E. P. C. Jones, R. A. Popa, H. Balakrishnan and N. Zeldovich. Relational cloud: A database-as-a-service for the cloud. In Proceedings of the 5th Biennial Conference on Innovative Data Systems Research, pages 235–241, Pacific Grove, CA, January 2011.

[12] National Vulnerability Database. CVE statistics. http://web. nvd.nist.gov/view/vuln/statistics, February 2011.