Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 12, December 2014, pg.65 – 69

SURVEY ARTICLE

A Survey on Cloud Security in DaaS with Encryption in Metadata

Swathi.C[1], Kavitha.S.M[2], Vishnupriya.S[3]

[1]Student Computer science and Engineering, Sri Eshwar College of Engineering, India

[2]Assistant Professor Computer science and Engineering, Sri Eshwar College of Engineering, India

[3]Student Computer science and Engineering, Sri Eshwar College of Engineering, India

[1] swathicvp@gmail.com; [2] kiviktg@gmail.com; [3] smpvishnu@gmail.com

Abstract— Cloud computing becomes popular because of its provision of flexible access to the cloud databases which allows users to access and extract the sensitive information whenever they require. The storing and accessing of information through the third party server or proxy server increases the burden of the users which leads to the time complexity and memory complexity. When the information are stored and accessed through the cloud server r proxy server, security becomes the main concern, where there is a possibility collusion of data contents. The collusion occurs because the cloud servers are not fully trustable. In the existing work, the distributed concurrent independent access to the information to the users are providing without introducing the any intermediate server. The accessing of information is done by the users directly to the cloud server where the information is without the intermediate proxies. However the encryption technologies used for encrypting the data's before outsourcing to the clients is a complex process which may affects the user access to the database. And also existing work cannot provide the promising security guarantee to the users where the database administrator may collude a data that is honest but curious. This problem is overcome by proposing a novel encryption approach in which data item can be decrypted by using the one of the user passwords.

Keywords— Intermediate Proxy, Security, Encryption, Cloud Servers

I. INTRODUCTION

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 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(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.

Security could improve due to centralization of data, increased security-focused resources, etc., but concerns can persist about loss of control over certain sensitive data, and the lack of security for stored kernels. Security is often as good as or better than other traditional systems, in part because providers are able to devote resources to solving security issues that many customers cannot afford. However, the complexity of security is greatly increased when data is distributed over a wider area or greater number of devices and in multi-tenant systems that are being shared by unrelated users. In addition, user access to security audit logs may be difficult or impossible. Private cloud installations are in part motivated by users' desire to retain control over the infrastructure and avoid losing control of information security.

II. SECURITY ISUUES IN DAAS SERVICES

As people rely more and more on the Internet and Cloud technology, security of their privacy takes more and more risks. On the one hand, when data is being processed, transformed and stored by the current computer system or network, systems or network must cache, copy or archive it. These copies are essential for systems and the network. However, people have no knowledge about these copies and cannot control them, so these copies may leak their privacy. On the other hand, their privacy also can be leaked via Cloud Service Providers (CSPs') negligence, hackers' intrusion or some legal actions. These problems present formidable challenges to protect people's privacy.

III. RELATED WORK

ESS addresses two threats. The first threat who tries to learn private data (e.g., health records, personal information, financial statements, etc,..) is a curious database administrator (DBA) by snooping on the DBMS server; here, ESS prevents the DBA from learning private data. The second threat that gains an adversary of application and DBMS servers complete control. In this case, ESS cannot provide are logged into the application during an attack any guarantees for users that, but can still ensure the confidentiality of logged-out users data.

There are two challenges in combating these threats. The first lies in the confidential information revealed to the DBMS server between minimizing the amount of the ability to efficiently execute a variety of queries. The second challenge is to minimize the amount of data leaked when an adversary compromises the application server in addition to the DBMS server.

1. DBMS Server Compromise

In this threat, ESS guards other external attacker against a curious DBA with full access to the data stored in the DBMS server.

2. Arbitrary Threats

The solution is to encrypt different data items (e.g., data belonging to different users) with different keys. To determine developers annotate the application's database schema for each data item s, the key that should be used to express finer-grained confidentiality policies. A curious DBA till cannot obtain private data by snooping on the DBMS server (threat 1), and in addition, an adversary who compromises the application server or the proxy can now decrypt only data of currently logged-in users (which are stored in the proxy).

Concurrent Access Modification

It is done by k - 1 committed and one uncommitted versions of each record through our indirection mapping. By decoupling committed and uncommitted versions, we avoid clashes between readers of currently committed data and writers of newly updated/inserted records, without changing the semantics or the structure of the index.

In 2V-Indirection, the currently committed version of every record is given by cRID and the outstanding uncommitted version is given by uRID. The triplet (LID, cRID, uRID) presents a conceptual and logical connection; but, it does not dictate that the indirection mapping table must be physically extended such that it pre-allocates enough space for the uRID. The uRID could be maintained for only the active set of transactions to reduce space overhead.

There is an important subtlety that arises when combining indexes with the 2V-Indirection mapping. Suppose we update a record on column col_i, where an index is also defined on col_i. Now whenever a record value for both the old value and the new value of the coli is changed, then column are associated to the record's

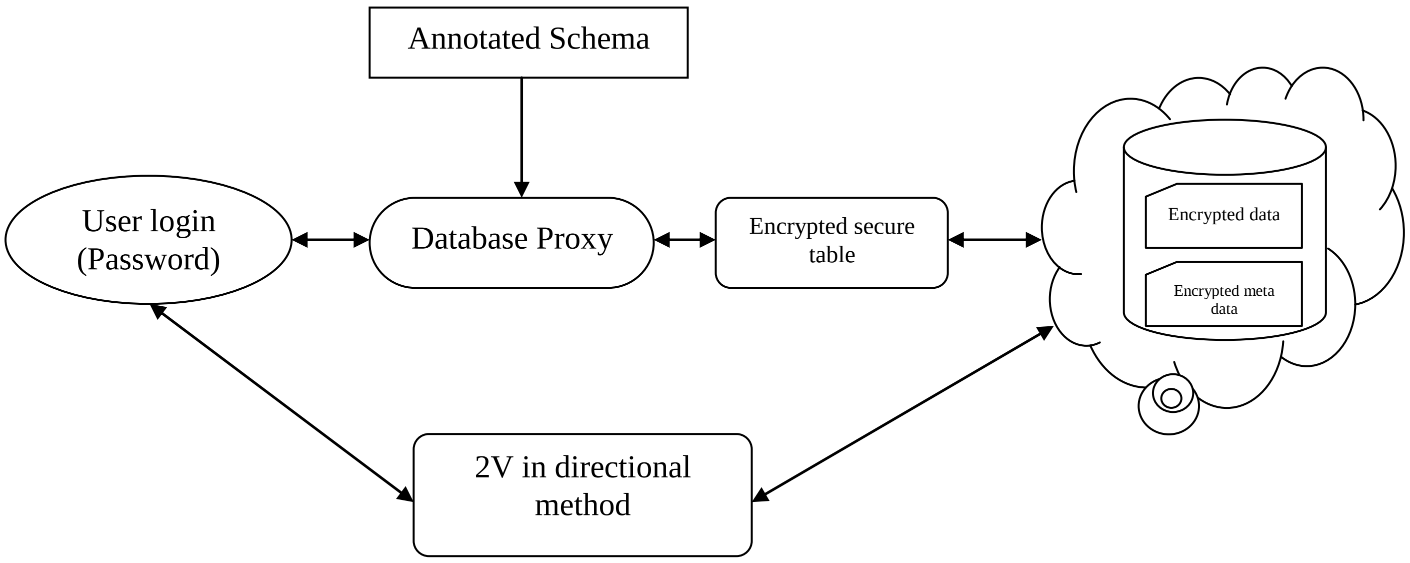
LID. This allows readers gives an option of reading either committed or uncommitted values from the index to detect that both values are referring to the same record due to the common LID. By examining, it can easily be determined whether value and LID pair is committed or uncommitted leaf page using a single bit to indicate whether an entry is committed or not.

Through 2V-Indirection, concurrent readers are able to access the currently committed version of every record without interfering with writers. Similarly, writers are able install an updated uncommitted version of a record without blocking current readers. By placing a reference to the uncommitted version of every record, it also enables the readers to speculatively read uncommitted data and ignore the committed version or vice versa. Therefore, kV Indirection seamlessly allows access to multiple versions of a record without changing the underlying structure. kV-Indirection is transient in nature because it maintains only references to at most k recent versions of the record.

Advantages:

- Better handling of threats which occurs by the honest but curious behaviours of database administrator.

Architecture Diagram



Meta Data Management

Managing the metadata in a management solution is an important step in a metadata management. It is part of the management to make sure that the metadata are completely concurrently available at any time. Managing a metadata system is also about creating sure that end-users of the system are aware of the possibilities allowed by a well-designed metadata system and how to maximize the benefits of metadata. Adequate monitoring and checking the metadata to ensure that the schema remains relevant is advised and to be secured.

Metadata produced by SecureDBaaS which may contain all the necessary information that is to manage SQL statements over the encrypted database in a way it is accessible to the user. Metadata management represent an real SecureDBaaS is the storing all metadata in the untrusted cloud database together with the encrypted tenant data. SecureDBaaS which may uses two types of metadata. Metadata databases that are related to the whole database.

Metadata are associated along with one secured hashtable. Each metadata hashtable contains all information that is necessary to encrypt and decrypt data of the associated securehash table.

Metadata database contain the encryption keys that are used for the secure types having the field of confidentiality set to secure database. A different types of encryption key is associated with all the possible combinations of data type and encryption type. Hence, the metadata secure database represent a keying and do not contain any information about original data.

Setup using COL, MCOL, and DBC

In this module we describe how to initialize a SecureDBaaS architecture from a cloud securedatabase service acquired by a tenant from a cloud provider. In case of hashtable that may contains just the metadata database that the DBA creates the metadata database for storage, and not the hashtable metadata. The DBA provider populates the securedatabase 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 securedatabase storage hashtable after encryption through the master key. Then, the DBA provider distributes the master key to the original users. Clients can able to control language through some standard data as in any unencrypted securedatabase access control policies are administrated by the DBA provider. In the following steps, the DBA provider creates the tables of the encrypted database. The three field confidentiality must consider attributes such as(COL, MCOL, and DBC).

Sequential SQL Operations

The first authentication purposes is for connection of the client with the cloud DBaaS. SecureDBaaS mechanisms provided by the original DBMS server relies on standard authentication and authorization. After the authentication, a user interacts through the SecureDBaaS client with the cloud database. SecureDBaaS analyzes which tables are involved and to retrieve their metadata from the cloud database the original operation to identify. The metadata are decrypted used to translate the original plain SQL into a query that operates on the encrypted database through the master key and their information is.

Translated nor plaintext tenant data operations contain neither plaintext database (table and column names). Nevertheless, the SecureDBaaS client can issue to the cloud database they are valid SQL operations that. Translated operations over the encrypted tenant data are then executed by the cloud database. As there is a one possible to prevent a trusted database user from accessing or modifying some tenant data by granting limited privileges on some tables to one correspondence between plaintext tables and encrypted tables as it is. User privileges and encrypted cloud database can be managed directly by the untrusted. The results are , decrypted, and delivered to the user of the translated query that includes encrypted tenant data and metadata are received by the SecureDBaaS client. The SQL statement complexity of the translation process depends on the type.

Concurrent SQL Operations

The support to multiple independent (and possibly geographically distributed) concurrent execution of SQL statements issued by clients is one of the most important benefits of SecureDBaaS with respect to recent solutions. Our architecture prevent unauthorized clients from decoding encrypted tenant data resulting in permanent data losses 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. Here, we remark two classes of statements that are supported by the importance of distinguishing SecureDBaaS: SQL operations not causing modifications to the database structure, such as operations involving alterations of the database structure through read, write, and update; creation, removal, and modification of database tables (data definition layer operators).

This concurrent SQL operations are done as follows:

reading r(x), the currently committed version of x is read; the current version is read from the cRID column of the 2VIndirection structure. For phantom detection, the range-predicate of the query is also registered.

writing w(x), a new uncommitted version of record is installed by locking *wl*(x), a write lock is set prior to modifying x.

validating reads, a read lock *rl*(x) is set prior to reading the current version of x in the readset; the current cRID value of x is read from the 2V-Indirection structure, for each x if its cRID value has not changed from when it was first read, then the validation is satisfied.

certifying writes, a certify lock *cl*(x) is set prior to finalizing the transaction on every data item x modified by the transaction (lock promotions) in order to ensure that no active transaction with repeatable read isolation or higher is currently reading the current value of records. The certification is also extended to satisfy the registered range-predicates.

commit, newly committed versions are installed, and all read and write locks are released.

IV. CONCLUSIONS

The Conclusion to show the effectiveness of the proposed work by comparing it with the existing methodology based on the parameters like Average encryption time, response time and the throughput value. A research includes large part of the 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 2V indirection method which leverages chain key encryption method an effective way of concurrent modification of database and the secure access mechanism. The experimental result shows that the proposed method is better than the existing methodologies. Our plan to provide researchers with further valuable release the current SeDBaaS system will help to experience to inform future object-based storage system designs for Cloud services.

REFERENCES

[1] Ariel J. Feldman, William P. Zeller, Michael J. Freedman, and Edward W. Felten, "SPORC: Group Collaboration Using Untrusted Cloud Resources,", Proc. Ninth USENIX Conf. Operating Systems Design and Implementation, Oct. 2010

[2] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for storage security in cloud computing," in Proc. IEEE INFOCOM, 2010.

[3] T. Cholez, I. Chrisment, and O. Festor, "Evaluation of sybil attack protection schemes in kad," in Proc. 3rd Int. Conf. Autonomous Infrastructure, Management and Security, Berlin, Germany, 2009, pp. 70–82.

[4] B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Small, J. Zelenka, and B. Zhou, "Scalable performance of the panasas parallel file system," in Proc. 6th USENIX Conf. File and Storage Technologies (FAST), 2008.

[5] S. A.Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in Proc. 7th Symp. Operating Systems Design and Implementation (OSDI), 2006.

[6] A. Acharya, M. Uysal, and J. Saltz, "Active disks: Programming model, algorithms and evaluation," in Proc. 8th Conf. Architectural Support for Programming Languages and Operating System (ASPLOS), Oct. 1998, pp. 81–91.

[7] R. Wickremesinghe, J. Chase, and J. Vitter, "Distributed computing with load-managed active storage," in Proc. 11th IEEE Int. Symp. High Performance Distributed Computing (HPDC), 2002, pp. 13–23.

[8] M. Wei, L. M. Grupp, F. E. Spada, and S. Swanson, "Reliably erasing data from flash-based solid state drives," in Proc. 9th USENIX Conf. File and Storage Technologies (FAST), San Jose, CA, USA, Feb. 2011.

[9] S. Wolchok, O. S. Hofmann, N. Heninger, E. W. Felten, J. A. Halderman, C. J. Rossbach, B. Waters, and E. Witchel, "Defeating vanish with low-cost sybil attacks against large DHEs," in Proc. Network and Distributed System Security Symp., 2010.

[10] Y. Tang, P. P. C. Lee, J. C. S. Lui, and R. Perlman, "FADE: Secure overlay cloud storage with file assured deletion," in Proc. SecureComm, 2010