International Journal of Advance Research and Innovation
Vol. 3(2), Apr-Jun 2015, pp. 59-62
Doi: 10.51976/ijari.321514
www.gla.ac.in/journals/ijari
© 2015 IJARI, GLA University

Article Info
Received: 05 Apr 2015 | Revised Submission: 30 Apr 2015 | Accepted: 20 May 2015 | Available Online: 15 Jun 2015

Efficient and Secure Data Sharing in Cloud Storage by Using Encryption Techniques

J. Jayalakshmi and P. Shanmuga Priya***

ABSTRACT

Using cloud storage, users can remotely store their data and enjoy the on-demand high-quality applications and services from a shared pool of configurable computing resources, without the burden of local data storage and maintenance. However, the fact that users no longer have physical possession of the outsourced data makes the data integrity protection in cloud computing formidable task, especially for users with constrained computing resources. Moreover, users should be able to just use the cloud storage as if it is local, without worrying about the need to verify its integrity. This paper enables security in cloud using encryption key that is generated while uploaded files by the user. The random key is generated using random key generation algorithm. The user can transfer their files with the permission of the owner. Multi-keyword ranked search over encrypted cloud data establish a set of strict privacy requirements for such a secure cloud data utilization system to become a reality. This provides high security while sharing files in cloud. The user can audit their uploaded files and verify the contents of the uploaded file in the cloud.

Keywords: *Cloud; Security; Confidentiality; Secure DBaaS.*

1.0 Introduction

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

The information managed by Secure DBaaS 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, Secure DBaaS 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. Secure DBaaS 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.

The 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 Secure DB aaS 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 Secure DB aaS; hence, even if different plaintext tables have columns with the same name, the names of the columns of the corresponding secure tables is different. This design

**Corresponding Author: Department of Computer Science Engineering, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamil Nadu, India (E-mail: praveenraj1310@gmail.com)*
***Department of Computer Science Engineering, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamil Nadu, India*

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). Secure DBaaS offers three field confidentiality attributes. 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.

2.0 Related Work

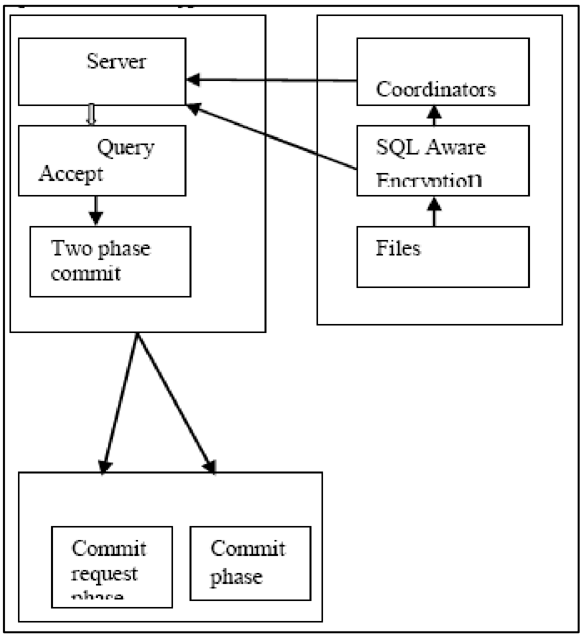
In the existing architectures that store just tenant data in the cloud database, and save metadata in the client machine or split metadata between the cloud database and a trusted proxy. When considering scenarios where multiple clients can access the same database concurrently, these previous solutions 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.

3.0 System Design

In the architecture does not require modifications to the cloud database, and it is immediately applicable no theoretical and practical limits to extend our solution to other platforms and to include new encryption algorithms 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. Proposed in, that makes it possible to execute range queries on data and to be robust against collusive providers. Secure DBaaS differs from these solutions as it does not require the use of multiple cloud providers, and makes use of SQL-aware encryption algorithms to support the execution of most common SQL operations on encrypted data.

Fig 1: System Architecture



3.1 Data dynamics

Data dynamics means after clients store their data at the remote server, they can dynamically update their data at later times. Data Dynamics provides an integrated, industry-leading storage management software platform designed to help enterprise and cloud IT service providers cost-effectively manage heterogeneous storage infrastructures, address the explosion of unstructured file data in enterprise environments, and move and manage file data between on-premise and private cloud environments.

The Data Dynamics Storage X platform is the engine behind Data Dynamics storage management solutions. Most enterprises today buy varying types of storage from multiple vendors to meet specific business needs. A typical enterprise IT environment contains storage resources from multiple vendors, including Dell, HP, IBM, Network Appliance, and EMC, each with their own proprietary device management tools.

Storage professionals face challenges with finding ways to centrally manage data in such highly mixed environments, and often resort to rudimentary tools or home-grown scripts.

These challenges include moving data across CIFS- and NFS-based file systems and efficiently managing storage and data using multiple tools from multiple vendors.

3.2 Public verifiability

Each and every time the secret key sent to the client's email and can perform the integrity checking operation. In this definition, it has two entities: a challenger that stands for either the client or any third party verifier, and An adversary that stands for the untrusted server. Client doesn't ask any secret key from third party. All content must be verifiable.

The burden to demonstrate verifiability lies with the editor who adds or restores material, and is satisfied by providing a citation to a reliable source that directly supports the contribution

3.3 Metadata generation

Separate hash key is generated for each and every file uploaded by the user. The details of the uploaded file such as file id, file name, hash key, date and time also be stored. To keep the stored data confidential against untrusted cloud service providers, a natural way is to store only the encrypted data and providing an efficient access control mechanism using a competent cipher key which is becoming a promising cryptographic solution.

3.4 Privacy against third party verifiers

The system is private against third party verifiers. If the server modifies any part of the client's data, the client should be able to detect it. Furthermore, any third Party verifier should also be able to detect it. In case a third party verifier verifies the integrity of the client's data, the data should be kept private against the third party verifier. Information privacy, or data privacy (or data protection), is the relationship between collection and dissemination of data, technology, the public expectation of privacy, and the legal and political issues surrounding them.

Privacy concerns exist wherever personally identifiable information or other sensitive information is collected and stored – in digital form or otherwise. Improper or non-existent disclosure control can be the root cause for privacy issues.

The right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures, shall not be violated, and no Warrants shall issue, but upon probable cause, supported by Oath or affirmation, and particularly describing the place to be searched, and the persons or things to be seized.

3.5 Key generation

Symmetric-key algorithms are ways of doing cryptography where the keys for decryption and encryption are exactly the same shared secret. You can generate the secret randomly, from a password, or through a secret key-exchange procedure like Diffie-Hellman. Symmetric-key algorithms are very important because they are faster on computers than public-key algorithms.

In public-key cryptography, also called asymmetric-key cryptography, it is hard to figure out the key for decryption from the key for encryption, so you can tell the key for encryption to the public with no problem, and everyone can send you secret messages. Stream ciphers encrypt a message as a stream of bits one at a time.

Block ciphers take blocks of bits, encrypt them as a single unit, and sometimes use the answer later too. Blocks of 64 bits have been commonly used; though modern ciphers like the Advanced Encryption Standard use 128-bit blocks.

Symmetric-key algorithms are a class of algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of cipher text.

The keys may be identical or there may be a simple transformation to go between the two keys. The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link.

This requirement that both parties have access to the secret key is one of the main drawbacks of symmetric key encryption, in comparison to public-key encryption.

Symmetric-key encryption can use either stream ciphers or block ciphers

1. Stream ciphers encrypt the digits (typically bytes) of a message one at a time.
2. Block ciphers take a number of bits and encrypt them as a single unit, padding the plaintext so that it is a multiple of the block size. Blocks of 64 bits have been commonly used.

4.0 Conclusion

In this paper an innovative architecture that guarantees confidentiality of data stored in public cloud databases is proposed. 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 heterogeneous 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 Postgre SQL 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, 53(4), 2010, 50-58

[2] A. J. Feldman, W. P. Zeller, M. J. Freedman, E. W. Felten, SPORC: Group Collaboration Using Untrusted Cloud Resources, Ninth USENIX Conf. Operating Systems Design and Implementation, 2010

[3] C. Gentry, Fully Homomorphic Encryption Using Ideal Lattices, 41st Ann. ACM Symp, Theory of Computing

[4] H. Hacigu¨mu¨ s¸, B. Iyer, S. Mehrotra, Providing Database as a Service, 18th IEEE Int'l Conf. Data Eng, 2002

[5] H. Hacigu¨mu¨ s¸, B. Iyer, C. Li, S. Mehrotra, Executing SQL over Encrypted Data in the Database-Service-Provider Model, ACM SIGMOD Int'l Conf. Management Data, 2002

[6] W. Jansen, T. Grance, Guidelines on Security and Privacy in Public Cloud Computing, Technical Report Special Publication 800-144, NIST, 2011

[7] J. Li, M. Krohn, D. Mazie`res, D. Shasha, Secure Untrusted Data Repository (SUNDR), Sixth USENIX Conf. Opearting Systems Design and Implementation, 2004

[8] J. Li, E. Omiecinski, Efficiency and Security Trade-Off in Supporting Range Queries on Encrypted Databases, Proc. 19th Ann. IFIP W WG 11.3 Working Conf. Data and Applications Security, 2005

[9] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, M. Walfish, Depot: Cloud Storage with Minimal Trust, ACM Trans. Computer Systems, 29(4), article 12, 2011

[10] E. Mykletun, G. Tsudik, Aggregation Queries in the Database-as-a-Service Model, 20th Ann. IFIP WG 11.3 Working Conf. Data and Applications Security, 2006

[11] R. A. Popa, C. M. S. Redfield, N. Zeldovich, H. Balakrishnan, CryptDB: Protecting Confidentiality with Encrypted Query Processing, 23rd ACM Symp Operating Systems Principles, 2011