Volume 5, Issue 4, 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

# Multi-Key Encryption in Multi-tenant Database for Data Isolation

**Panakj Kumar Khetwal**
M Tech (CS), Jamia Hamdard,
Delhi, India

**Jawed Ahmed**
Assistant Professor, Jamia Hamdard,
Delhi, India

***Abstract*— Multi-tenancy is to provide services to multiple clients on same physical resources through virtualization. But, at the same time providing security to private data of multiple tenants is big challenge. In this paper, the focus is on security and data privacy (isolation) for SaaS applications that are built and deployed on MTA (Multi-tenant Architectures). In MTA a single database is shared with all tenants. Sharing of a database can lead to unauthorized access of tenant's data by another tenant, which is a serious security flaw. To address this problem, it is proposed to store data in encrypted by using combination of special characteristically symmetric keys and PKI, so that record stored is not exposed to cloud service provider as well as to other tenants.***

***Keywords*— Multi-tenant architecture, Encryption, Public Keys, Symmetric Keys, Order preserving encryption.***

## I. INTRODUCTION

Multi-tenancy approach for a cloud infrastructure presents a lot of challenges in terms of security and privacy. The basic security issue with multi-tenancy is the very premise in which it is based; i.e., multiple tenants sharing the same resource set.

The database in multi-tenancy model has its own set of unique challenges which are very different from those of the traditional databases. The main challenge is to ensure data security. Data security addresses the problem of protecting data stored at service provider from various threats. Data are most valuable asset for organizations because it is used for making vital strategic decisions. Any compromise of data can have disastrous effects for the organization. This results in reluctance in using the SaaS model by many corporations since their data will then be stored at service provider's end and there will always be a risk that SP may view their data. Another challenge is ensuring data isolation for multiple tenants or clients. Data management is critical as several tenants are using the same database, or even same relations. So, there is sincere requirement of privacy and confidently of data in multi-tenant database environment.

These all privacy and security concerns can be greatly addressed using well-defined encryption strategies to store the data into a multi-tenant database. Encryption is especially important in situations involving privacy requirements, or when many tenants share the same set of database relations. The aim is to prevent every tenant's data so that it will remain inaccessible to unauthorized tenants.

There are basic approaches to encrypt data with single symmetric key-pair, which is shared among all tenants. This may preserve data exploitation from third party attackers, but not from one of the tenant of the same system. There are also approaches to assign each tenant a set of asymmetric key-pair for its data encryption, along with another layer of encryption embedded by the service provider. But this may additional responsibility over service provider to maintain keys, also storing and retrieving single data using asymmetric key-pair is having considerable computation overhead which drastically degrades the performance.

By keeping above facts in mind, this paper proposes an hybrid approach of using public key of tenant to encrypt the symmetric keys of that tenant, which will finally used for data encryption. The encryption techniques used are having some special characteristics for performance enhancements. It will give the tenants confidence about isolation and privacy of their valuable data, without compromising much on performance.

This paper is organized as follows. We discuss the related works done by predecessors, and analyze the strengths and weaknesses of existing solutions, in section 2. In section 3, a threat model is presented, that characterize the possible attacks that may be conducted against confidential data. In section 4, we present our database encryption scheme and show how it answers the data confidentiality and isolation questions. Finally in section 5, we conclude the research.

## II. RELATED WORK

Encrypting data is core element of the data security. So there is need of proven and efficient encryption scheme. Since long, the symmetric keys are the only way to encrypt the data. But, Diffie and Hellman [1] proposed the concept of public key cryptography in 1976, which brought revolutionary change to cryptographic world, and also added a new dimension in security of database systems. In 1978, Rivest, Shamir and Adleman [2, 3] published RSA key-pair scheme, the first public key system which is still widely used in data encryption.

In terms of encryption scheme research for relational database management system, many schemes have been proposed. Davida, Wells and Kam [4], in their 1981 paper presented a database encryption scheme with subkeys based on the Chinese Reminder theorem. This scheme is a record oriented cryptosystem which enables encryption at the level

 

of rows and decryption at the level of cells. In 1997, Hwang and Yang [5] proposed a scheme which extends the subkey encryption by supporting multilayer access control. They also introduced a twophrase algorithm for database system in another paper [6]. These schemes improved the encryption algorithms and made it more efficient, but they didn't give an effective way to manage keys and ensure that the keys are safekeeping.

In 2002, Bouganim and Pucheral [7] presented a chip secured data access principle as a solution to data confidentiality problem. This solution is quite secure and effective, but it is still too complicated and the cost is too expensive, since it requires the DBMS' migration to a smart card and modifications of a large number of queries.

Jingmin He and Min Wang [8] researched how to integrate modem cryptography technology into a relational database management system from the view of cryptographic. They introduced a security dictionary to solve the problems linked to user management and key management, which is very effective to protect security related data. But in their schemes the key pairs are stored at a directory server, so every time the database server needs a key to encrypt or decrypt data, the key must be retrieved from the directory server first, which decreases the performance of operation.

In a paper Mohamed Almorsy [9] suggests externalizing the security from the target applications. The tenants can impose security constraints to applications by an external interface provided by the service provider without knowing its background where the tenant's security lies with CSP. Kamara [10] work considered as the first contribution to encryption-based cloud storage. In public cloud infrastructure where the service provider is not completely trusted by clients/tenants, Kamara proposed secure cloud storage architectures for both clients and enterprise by using non-standard cryptographic techniques, such as attributed encryption, searchable encryption, etc.

One approach to solve the SQL performance problem in encrypted data is to use a Fully Homomorphic Encryption scheme such as one proposed by C. Gentry [11]. These schemes allow computation of any arbitrary function over encrypted data without the need to decrypt it. This approach offloads all the database processing work to the server.

Many approaches have been proposed for securing database, such as bucketization based approach [12][13], CryptDB [14][15], PhantomDB [16] etc. Bucketization based approach supports all standard SQL constructs but suffers from poor performance. It also requires the client to maintain a full-fledged database engine. On the other hand, CryptDB has good performance and does not require the client to maintain a database engine but it is able to support a very limited set of SQL constructs. These limitations prevent the use of these approaches in the industry. The PhantomDB addresses most of the issues, but suffers the privacy and separation issues within inter-tenants in Shared DB- Shared Schema model.

In this work we propose a hybrid approach, MultiKey-DEM (Multiple Keys - Data Encryption Model), to address the data security problems in multi-tenant database architecture. This scheme aims to enhance the data security and isolation problem with efficient query processing. We provide efficient key management and safe storage for security related data. Before discussing our new scheme, a threat model is to be covered, which illustrates the problems we would solve.

## III. THREAT MODEL

A typical threat models in literature identify the possible issues related to five roles: the tenant database administrator (DBA), the tenant database users, the cloud provider employees, sibling tenants on the same cloud, and people external to tenant's/provider's organizations.

The DBA is the only role that has access to all tenants' data. DBA is in charge of installing and configuring the database, implementing the access control policies and managing the users' credentials. As in related literature, our threat model assumes that the DBA is trusted. However, introducing the tenant specific public keys makes the data available to DBA only in encrypted form.

Tenant Insiders are the 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.

Cloud Insiders are employees of the cloud provider that have access to the cloud infrastructure hosting the database service. They may be interested in accessing tenant data, but they do not modify or delete them. Reading data would remain unnoticed by a tenant.

Sibling Tenants are other tenant's sharing the same database and infrastructure and/or even same application to access the data. They are interested to get the access of the other tenant's data for various reasons by escalating their privileges through a violation of the access control policies or exploiting the loophole in application.

External Attackers do not have legal access to the infrastructure/data of the tenant organization or of the cloud provider. They can try to access tenant information through several types of attack such as by eavesdropping data in motion between the tenant clients and the cloud servers.

Ensuring data confidentiality in the cloud against external attackers, cloud insiders, tenant insiders, and sibling tenants can be achieved through some combinations of existing solutions such as encryptions techniques and different access control policies.

Now we anticipate a summary of the design choices and solutions that allow MultiKey-DEM to protect data against compromised DBA, external attackers, cloud insiders and tenant insiders, sibling tenants, and even against collusion between these roles. External attackers that eavesdrops network traffic cannot access any plaintext information because SQL operations issued to the cloud database are protected by using standard encryption protocols such as SSL. Cloud insiders and external attackers that have breached the cloud servers cannot access confidential information, because MultiKey-DEM encrypts tenant data with SQL-aware encryption algorithms. Tenant insiders cannot perform privilege escalation attacks on the encrypted database due 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 data and randomized

 

relations/columns. Sibling tenants and DBA can access the encrypted data of tenants but cannot get the plain text data due to public key encryption of tenant specific encryption keys.

## IV. NEW MULTI-TENANT DATABASE ENCRYPTION MODEL

In Multitenant application data isolation of tenants is a great issue. So, for security purpose separate database grant to each tenant is ultimate solution, but it is less cost effective, time consuming and defeats the soul of MTA SaaS. So using a single database for all tenants is the best step, if we address the security concerns effectively. In this, each row stores tenant data with the tenant id, so that each row get separated by its id. In this surrounding area, security concerns dart high that any wrongly configured application code or an inaccuracy in an admittance control list may put all tenants information in the danger of theft. To recover from this insecurity we use the concept of cryptography. But using simple encryption techniques may not eliminate the possibility of data leakage of tenant to other tenants or to cloud provider, and it also slows down the entire database performance.

MultiKey-DEM is the approach that uses asymmetric key-pair (PKI) for data isolation and multiple order preserving symmetric keys for faster data encryption/decryption and efficient SQL query processing. MultiKey-DEM maintains data security by encrypting data before storing it at the server. The SQL statements issued and results returned is moved over secure layer, typically over SSL.

The encryption schemes chosen by MultiKey-DEM have some special characteristics which allow server to perform certain operations on the cipher text itself. MultiKey-DEM uses Order Preserving Encryption (supports range queries), Addition Homomorphic Encryption (supports aggregates), Multiplication Homomorphic Encryption and Search (supports word searches).

For each of the symmetric encryption, the keys used are the tenant specific. First the appropriate key is selected from the meta-table before applying any of the encryption over data. These keys are prior encrypted with the public-key of the tenant. The public keys of tenants are stored in meta-tables. There is always requirement of tenant's private key to get these symmetric encryption keys. Once the keys are known, then each column of a tuple is encrypted separately.

Each data value is encrypted simultaneously under multiple encryption schemes. This leads to multiple encrypted columns for a single plain text column. The encryption schemes used to encrypt a column depend on its data type. A numeric value is encrypted under Order Preserving Encryption (OPE), Addition Homomorphic Encryption (AHE) and Multiplication Homomorphic Encryption (MHE). A string value is encrypted under Order Preserving Encryption and Search Encryption.

While storing a relation on the server, relation name and column names are randomized. This mapping from plain text relation and column names to randomized relation and column names is stored as metadata at the client.

The MultiKey-DEM model introduces a MultiKey-DEM Engine in front of the core RDBMS. It handles the keys management, encryption/decryption and query transformation. User/application submits a plain text SQL query to the engine over SSL encryption. The query is then parsed by the engine and meta tables are consulted for tenants' symmetric keys. Once appropriate keys (in encrypted form) are selected then these are decrypted for plain text symmetric keys by applying the private key of tenants'.

The private key can be passed with each SQL request, or can be stored in temp table for a session. Even, it can be configured to generate and store asymmetric key pair for each session. So the key management is the major activity of the engine.

The encrypted query then transformed for appropriate randomized columns again consulting the meta tables. Finally this final query is submitted to database server. Database server executes this cipher text query on the data stored at database hosted by it. The result of this query execution (encrypted data) is returned to the engine, which then decrypts them to get final plain text answer. This plain text answer is then given to the user as output of his query again over the SSL layer.

Let, there are multiple tenants T1 to Tn which shares a single database over cloud, in which each record is identified by its tenant id.

There is a system catalog table, which contains the tenant info, say sys_tenant_info. The content of this is stored encrypted using the service provider's master key, in following format:

1. tenant_id
2. tenant_public_key

When any record is inserted in relation R, the tenant id is automatically inserted. Before inserting the record a value is being encrypted by the symmetric key (SQL Aware Encryption) and stored in table.

The symmetric keys are generated and stored in a system table, say sys_tenant_keys, in the following format:

1. tenant_id
2. encryption_type
3. relation_name
4. attribute_name
5. key (Encrypted using Public Key of tenant)

The randomized mapping of original columns and relation name are also stored in a system table, say sys_object_mapping, in the following format:

 

1. object_id
2. object_type
3. original_object_name
4. randomized_object_name
5. encryption_type

Access of these meta tables are only restricted to the MultiKey-DEM engine.

## V. CONCLUSION

This work presented MultiKey-DEM, which is a model for isolation and security of tenants' data with efficient query processing in multi-tenant database over very securely encrypted data. MultiKey-DEM uses encryption to ensure security of data stored at the server. A data item is simultaneously encrypted under multiple, carefully chosen encryption schemes to produce multiple cipher texts. The symmetric keys used to encrypt the data are separate for each tenant. These keys are kept stored in encrypted form in metadata tables of the database by the public key of asymmetric key combination, chosen by tenant itself.

MultiKey-DEM is an approach which is built over many earlier proposed approaches, such as CryptDB and PhantomDB. The objective is to take the advantage of these methods and design a multi-tenant database solution, either built-in or as a front layer of a RDBMS, which secures tenants' data, ensures privacy and separation of their data as well as efficient query processing.

## REFERENCES

[1] W. Diffie and M. E. Hellman, "New Directions in Cryptography", IEEE Trans. Inform. Theory, 1976, IT-22:644-654.

[2] L. A. R. Rivest, A. Shamir. "A method for obtaining digit signatures and public-key cryptosystems", Communications of the ACM, 21(2):120–126, 1978

[3] L. A. R. Rivest, A. Shamir, "On digit signatures and public key cryptosystems", Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, Location, 1979.

[4] G.I. Davida, D.L. Wells, and J.B. Kam, "A Database Encryption System with Subkeys", ACM Trans. Database Syst., 1981, pp. 312-328.

[5] Min-Shiang, H. and Wei-Pang, Y., "Multilevel secure database encryption with subkeys", Data and Knowledge Engineering, 1997, pp. 117-131.

[6] M.S. Hwang and W.P. Yang, "A Two-Phase Encryption Scheme for Enhancing Database Security", J. Systems and Software, 1995, 31(12): 257-265.

[7] L. Bouganim and P. Pucheral, "Chip-secured data access: Confidential data on untrusted servers", Proc. of the 28th International Conference on Very Large Data Bases, Hong Kong, China, 2002, pp. 131-142.

[8] Jingmin He, Ming Wang, "Cryptography and Relational Database Management Systems", Proc. IEEE Symposium on the International Database Engineering & Applications, Washington, USA, 2001.

[9] Mohamed Almorsy, John Grundy, and Amani S. Ibrahim,"TOSSMA: A Tenant-Oriented SaaS Security Management Architecture", IEEE Fifth International Conference on Cloud Computing, 2012

[10] S.Kamara, Kristin Lauter, "Cryptographic cloud storage", FC'10 Proceedings of the 14th international conference on Financial cryptograpy and data security Pages 136-149, Springer, 2010

[11] C. Gentry. Fully homomorphic encryption using ideal lattices. In STOC, 2009.

[12] H. Hacigumus, B. R. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In SIGMOD Conference. ACM, 2002.

[13] H. Hacigumus, B. R. Iyer, and S. Mehrotra. Query optimization in encrypted database systems. In DASFAA. Springer, 2005.

[14] R. A. Popa, C. M. S. Red eld, N. Zeldovich, and H. Balakrishnan. Cryptdb: processing queries on an encrypted database. In Commun. ACM, volume 55, 2012.

[15] C. Curino, E. P. C. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakr-ishnan, and N. Zeldovich. Relational cloud: a database service for the cloud. In CIDR, 2011.

[16] Akshar Kaul. Query Processing in Encrypted Cloud Databases, July 2013

[17] H. Hacigumus, S. Mehrotra, and B. R. Iyer. Providing database as a service. In ICDE. IEEE Computer Society, 2002.

[18] Jinan Fiaidhi, Irena Bojanova, Jia Zhang, Liang-Jie Zhang, "Enforcing Multitenancy for Cloud Computing Environments", IT Pro, IEEE,2012.

[19] A. Boldyreva, N. Chenette, Y. Lee, and A. Oneill. Order-preserving symmetric encryption. In Advances in Cryptology-EUROCRYPT 2009. Springer, 2009.

[20] Boicea, A, Ghita V, Radulescu, Florin, Anca D. Encryption Strategies in Database. In Annals of DAAAM International, 2010

 

[21] K.Venkataramana, Prof.M.Padmavathamma. Multi-Tenant Data Storage Security In Cloud Using Data Partition Encryption Technique. In International Journal of Scientific & Engineering Research, Volume 4, Issue 7, July-2013

[22] Iqra Basharat, Farooque Azam, Abdul Wahab Muzaffar. Database Security and Encryption: A Survey Study. In International Journal of Computer Applications (0975 – 888)

[23] Volume 47– No.12, June 2012

[24] K.Venkataramana, Prof.M.Padmavathamma. Multi-Tenant Data Storage Security In Cloud Using Data Partition Encryption Technique. In International Journal of Scientific & Engineering Research, Volume 4, Issue 7, July-2013

[25] [Website] http://en.wikipedia.org/

[26] [Website] https://msdn.microsoft.com/

 