International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No.03, April 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

Secure Database in Cloud Computing: CryptDB Overview

Nivedita W. Wasankar[1], A.V. Deorankar[2]

[1]M. Tech. Scholar, Department of Computer Science and Engineering,
Government College of Engineering, Amravati (MH) (India)
[2] Assistant Professor, Department Department of Computer Science and Engineering,
Government College of Engineering, Amravati (MH) (India)

ABSTRACT

CryptDB is allows query processing over encrypted databases. The database managed by the cloud provider, but database items are encrypted with keys that are only known by the data owner. SQL queries run over the encrypted database using a collection of operations such as equality checks and order comparisons. CryptDB uses encryption schemes that allow such comparisons to be made on ciphertexts. CryptDB represents a weak attacker model because it assumes the existence of a trusted cloud-based application server and proxy. Nevertheless, CryptDB represents an interesting position on the trade-off between functionality and confidentiality from cloud providers. In this paper, we will go into details of CryptDB.

Keywords: CryptDB, Ciphertext, Order comparisons, Proxy server, SQL query.

I.INTRODUCTION

Cloud storage enables users to remotely store their data and enjoy the on-demand high quality cloud applications without the burden of local hardware and software management and reduce the cost for both the enterprises and individual users. A cloud client, such as an IT enterprise, wants to outsource its database to the cloud, which contains valuable and sensitive information (e.g. transaction records, account information, disease information), and then access to the database (e.g. SELECT, UPDATE, etc.).

Due to the assumption that cloud provider is honest-but-curious the cloud might try his/her best to obtain private information for his/her own benefits. Even worse, the cloud could forward such sensitive information to the business competitors for profit, which is an unacceptable operating risk. Encryption or obfuscation are needed before outsourcing sensitive data - such as database system - to cloud. Data and queries of the outsouced database should be protected against the cloud service provider.

One straightforward approach to mitigate the security risk of privacy leakage is to encrypt the private data and hide the query/access patterns. CryptDB is a system which acts as a proxy to secure the communication between the database server, and the applications server. CryptDB receives queries from the application server, secures them and sends them to the database server. Then, it will receive encrypted data from the database, decrypts it and sends to application server to be sent to the requester. CryptDB enables to run SQL queries on encrypted database data as it could do on plaintext. This is done because by principle, curious DBAs, attackers, DBMS and system infrastructure are the entrusted fellows. The assumption made is that the application server and the

database server are different and a proxy can intercept their communication. CryptDB can be implemented on a range of DBMS such as MySQL and Postgres.

II.RELATED WORK

Researchers try to find out solutions to keep data on the cloud secure. Processing data securely on the cloud is very much complicated. In this section, some well-known approaches are given for secure computing on the cloud.

2.1. Homomorphic encryption: Fully homomorphic encryption, the data can be encrypted before uploading to the cloud. By using conventional encryption schemes, it is unable to processed the data on the cloud. In such case, the cloud only used for storage purposes. Homomorphic encryption overcome this limitation in certain level, which allows logical operations on ciphertexts without decryption. Homomorphic encryption schemes like Paillier or ElGamal allow only one operation, namely either addition or multiplication [7], [8]. Fully homomorphic encryption schemes allow both addition and multiplication on ciphertexts, that allows an untrusted server to carry out arbitrary computation on encrypted data without decryption. By using fully homomorphic encryption schemes, the cloud service provider can run any program of client without knowing any information about the plaintexts. Fully homomorphic encryption scheme was first invented by Gentry in 2009 [5]. But, there is no practical scheme use fully homomorphic encryption today but a lot of work is being done in this field.

2.2. MONOMI: Monomi is the first system which can execute analytical workloads over encrypted data efficiently in a secure way [6]. It is based on CryptDB's design of encryption schemes. In CryptDB, query execution is done on the server, but in Monomi, query execution of complex queries is splitted between client and server. Also Monomi improves performance with some techniques: per-row precomputation, space-efficient encryption, grouped homomorphic addition, and prefiltering. CryptDB can handle four out of 22 TPC-H queries, but Monomi executes 19 out of 22 TPC-H queries. Additional designer and planner exists in Monomi to design the physical layout and to split the query execution according to the physical design.

III.CRYPTDB PRINCIPALS AND DESIGN TECHNIQUES

CryptDB is designed to overcome the weaknesses of current solutions which are either too slow or do not provide the necessary confidentiality. CryptDB provides a proxy server and some other components into the typical structure of database backed applications, in which DBMS server and a separate application server is added, as shown in the Fig. 1 below:

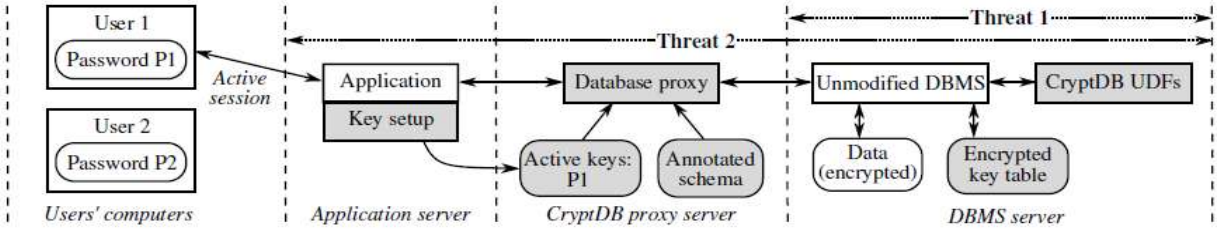

Figure 1. CryptDB's architecture

There are three approaches that CryptDB uses to solve the current approaches problems.

3.1. The SQL-aware encryption approach: SQL queries have a well-known structure such that it consists operators such as order comparisons, equality check and aggregates like sum and table joins. The SQL-aware encryption approach uses this fact. After that, CryptDB uses cryptographic methods for joins to transform the queries to a form which enable the DBMS to run them over encrypted data.

3.2. The adjustable query-based encryption: It solves the problem of certain cryptographic schemes that leaks more data than required. Just because they are still required, to adjust the queries the unions of encryption are needed which minimize the leakage of data.

3.3. Chain the cryptographic keys to user passwords : In the third approach, which is to protect users which are not logged into a system that is to chain the cryptographic keys to user passwords which enable data decryption to users with access privileges.

IV.QUERIES OVER ENCRYPTED DATA

To secure data hide any relations that can be read from the database by changing the normal database schema. and then stored it in the CryptDB proxy. Table and column names are also encrypted. Column encryption depends on the data in that column, and the type of queries to be ran by the DBMS. There are six methods of encryption depending upon the type of data in a column.

4.1. Random (RND): It produces a ciphertext from a column name by using a randomly generated initial Vector (IV). RND provides a powerful encryption. It is suitable when handling sensitive data. But it does allow running of queries which require computation e.g. MAX, SUM and ORDER BY.

4.2. Deterministic (DET): It provides a weaker security because of the leakage. Leakage caused by producing the same ciphertext for on same text. DET is a pseudo-random permutation.

4.3. Order-preserving encryption (OPE): It preserves the order of ciphertext to remain as they were in plaintext. For example, for any key K, if a < b, then $OPE_K(a) < OPE_K(b)$. OPE is comparatively weaker than DET.

4.4. Homomorphic (HOM): Homomorphic is useful for any data that requires computation. Because of it, complex mathematical computations are possible as they could be done plaintext.

4.5. JOIN and OPE-JOIN: This is used to join columns for hiding the correlation between cross-column just because of different DET keys are used. Joins are done for equality and order checks.

4.6. Word checks (SEARCH): This method is used to search encrypted words. This method is used in queries with SQL operations such as LIKE. SEARCH is a secure as Random because it does not allow the DBMS to see whether some of keyword are repeated in many rows.

The encrypted query is reaches the DBMS with the encryption keys. it runs successfully with a few User Defined Functions (UDFs). The data that it returns to the proxy is decrypted and sent to the application.

V.FEATURE

Feature of CryptDB include:

5.1. It use high standards of encryption.

5.2. A large number of query types are include as it use different type of encryption methods and query adjustments.

5.3. It is fast. This is from the results of the real tests that were carried out on phpBB, HotCRP and grad-apply.

5.4. The layered encryption provides a complex technique which send different data sets to different users.

VI.LIMITATIONS

CryptDB has some theoretical limitations such as,

6.1. Logged in users' data is at high risk.

6.2. Security of the cryptographic keys become overhead to the whole system.

6.3. The computation involved in of encrypting a query becomes intensive.

6.4. A single encryption method is not sufficient, so combing them becomes an overhead.

6.5. In some cases, CryptDB leaks data and over time, and this will help the attackers to study the layout, which will finally enable them intrude on user data.

VIII.CONCLUSION

In this paper, CryptDB is explain in a detailed way. CryptDB is the first practical Database Management System for running most standard queries on encrypted data. CryptDB does not make any changes to the DBMS. We revisited the server structure of CryptDB and pointed out the large overhead of the Proxy Server. We give a detailed analysis about the efficiency and security aspects.

REFERENCES:

[1] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, CryptDB: protecting confidentiality with encrypted query processing, in Proceedings of the 23rd ACM Symposium on Operating Systems Principles. ACM, 2011, pp. 85–100.

[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, Order preserving encryption for numeric data. In Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, Paris, France, June 2004.

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

[4] C. Curino, E. P. C. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, 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.

[5] C. Gentry, Fully homomorphic encryption using ideal lattices, In Proceedings of the forty-first annual ACM symposium on Theory of computing (STOC '09), 169-178, ACM, New York, USA, 2009. DOI=10.1145/1536414.1536440.

[6] S. Tu, M.F. Kaashoek, S. Madden, N. Zeldovich, MIT CSAIL, Processing Analytical Queries over Encrypted Data, 39th International Conference on Very Large Data Bases, Riva del Garda Trento, Italy, In Proceedings of the VLDB Endowment, Vol.6, No.5, August 26-30, 2013.

[7] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, In Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT'99), Jacques Stern (Ed.), Springer-Verlag, Berlin, Heidelberg, 223-238, 1999.

[8] T. El Gamal, A public key cryptosystem and a signature scheme based on discrete logarithms, In Proceedings of CRYPTO 84 on Advances in Cryptology, G R Blakley and David Chaum (Eds.), Springer-Verlag New York, New York, USA, 10-18, 1985.

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

[10] http://en.wikipedia.org