Overview of CryptDB

CPSC 5670 Term Paper

Dhaval Patel, Yi Jiang

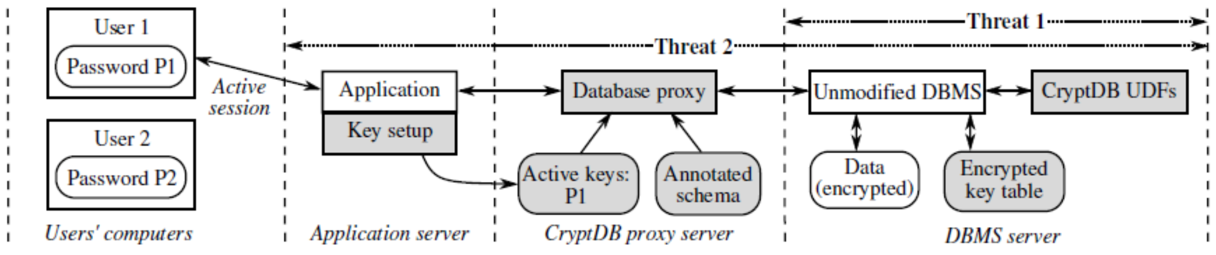
11/22/2013

Introduction

In the face of snooping Database Administrators (DBAs) and compromise from attackers, confidentiality in Database Management Systems (DBMS) should still remain. This paper comprehensively inquires, and tests a new way of securing databases in presence of the two threats mentioned above. This new technique is called CryptDB, a system which acts as a proxy to secure the communication between the database server, and the applications server. The paper mainly deals with confidentiality of information flowing between application and database servers, and not any form of information security like integrity and availability. 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. In a nutshell, CryptDB enables the BDMS 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.

CryptDB Principals and Design Techniques

According to the paper, CryptDB is designed to address the weaknesses of already current solutions which are either too slow or do not provide the necessary confidentiality. CryptDB adds a proxy server and some other components to the typical structure of database-backed applications, which consists of a DBMS server and a separate application server, as shown in the figure below:



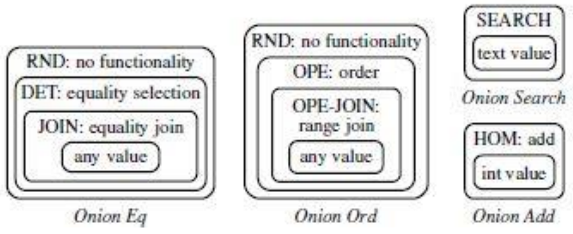
There are three approaches that CryptDB uses to solve the problems of the current approaches. The **SQL-aware encryption approach** uses the fact that SQL queries have a well known structure consisting of operators such as order comparisons, equality check and aggregates like sum and table joins. CryptDB then uses cryptographic methods for joins to transform the queries to a form that can enable the DBMS to run them on encrypted data. On the other hand, the **adjustable query-based encryption** solves the problem seen in the first techniques where certain cryptographic schemes leak more data than required. However, because they are still needed, unions of encryption are needed to carefully adjust the queries to minimize data leakage. The third approach, which is seen to protect users that are not logged in to a system, is to **chain the cryptographic keys to user passwords** to enable data decryption to users with access privileges.

Queries over Encrypted Data

To secure data the normal database schema is changed in order to hide any relations that can be read from the database, and then stored in the CryptDB proxy. In fact, table and column names are encrypted. Column encryption depends on the data held by that column, and the type of queries to be ran by the DBMS.

Depending the type of data a column stores, there are six methods of encryption. **Random (RND)** produces a ciphertext from a column name using a randomly generated initial Vector (IV). Due to the randomness, RND provides a powerful encryption and is suitable when handling sensitive data. Unfortunately, it does allow running of queries that require computation e.g. MAX, SUM and ORDER BY. The second way of encryption is the Deterministic (DET). It provides a weaker security because of the leakage caused by producing the same ciphertext for on same text. DET is a pseudo-random permutation. **Order-preserving encryption (OPE),** as the name suggests, it preserves the order of ciphertext to remain as they were in plaintext. For example, for any key k, if $x < y$, then $OPE_K(x) < OPE_K(y)$. OPE is comparatively weaker that DET because it reveal order. The fourth way of encryption is **Homomorphic (HOM**), which is handy for any data that requires computation. With it, complex mathematical computations are possible as they could be done plaintext. However, the authors of this paper never implemented this test in their experiment. Another way of encryption is (JOIN and OPE-JOIN), which is used to join columns to hide the correlation between cross-column. This is because different DET keys are used. Joins are done for equality and order checks. Lastly, word checks (SEARCH) is a method 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 repeats in many rows.

The encrypted query now reaches the DBMS with encryption keys. With a few User Defined Functions (UDFs), it runs successfully, and the data that it returns to the proxy is decrypted and sent to the application. It is worth noting that CryptDB use layered encryption to provide data secrecy, as shown in figure below. This varies and this is because of the types of queries and requirements on data access.



Multiple Principals

With the real checks that were used to evaluate CryptDB, with phpBB for example, it is quite open. In case the proxy and the infrastructure are untrusted, CryptDB can capture the application's access policy. This is where application developers have poser to annotate database schema in CryptDB to specify which principal has access to a certain subset of data items. CryptDB provides three straightforward steps that an application developer can use to annotate schema: specify the principle (such as users, groups, or messages) with the PRINCTYP annotation; specify columns with sensitive data and which principles will have access to them with ENC_FOR annotation; decide how to delegate a principle's rights to another with SPEAK_FOR annotation.

Each principal is associated with a randomly chosen key. If principal B speaks for principal A, then principal A's key is encrypted using principal B's key, which allows principal B to gain access to not only principal A's key but also his data. Each sensitive field is encrypted with the key of the principal in the ENC_FOR annotation. CryptDB encrypts the sensitive field into onions and onion keys are derived from a principal's key. The key of each principal is a combination of a symmetric key and a public-private key pair. CryptDB generally uses the symmetric key to encrypt any data and keys of other principals that are accessible to a certain principal. If a user is not online, CryptDB encrypts the data user his/her public key instead. When encrypting data in a query or decrypting data from a result, CryptDB follows key chains starting from user password until it obtains the desired keys.

Results

This paper reveals that CryptDB has few application and theoretical limitations;

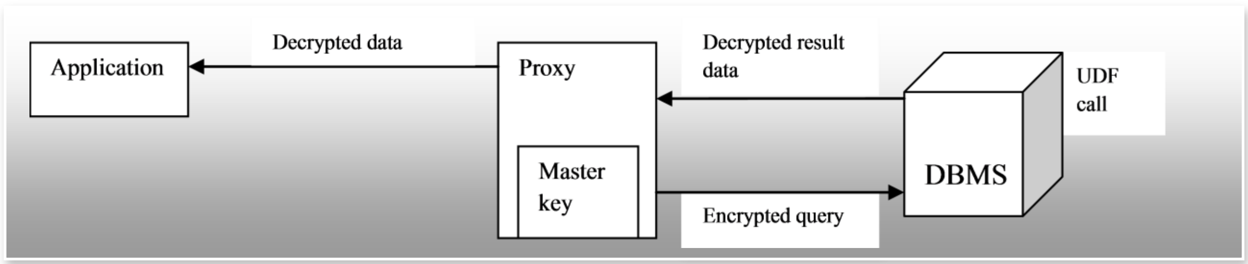
- Logged in users' data is at high risk.

- How to secure the cryptographic keys seems to be an overhead to the whole system.
- The computation involved in of encrypting a query becomes intensive, as shown in figure below.
- A single encryption method is not sufficient. Thus combing them becomes an overhead.
- In some stages, CryptDB leaks data and over time, and this will allow the attackers to study the layout, which will finally enable them eavesdrop on users data.

On the other hand, the standpoints of CryptDB that this paper digs into include;

- Use of high standards of encryption
- A large number of query types are includes because of use of a variety of encryption methods and query adjustments.
- It is fast. This is from the results of the real tests that were carried out on phpBB, HotCRP and grad-apply.
- The layered encryption can surely provide a complex technique that makes it easy to provide different data sets to different users.
- The ability to allow application developers to affect control by use annotated principles

From the paper reading, simple core architecture of CryptDB is as shown below.



Conclusion

The paper that is presented here is complete, and with logical sentiments. The theory is rich and the authors proofs that what they have done has never been done before. This is in line with their literature review. The problem that the paper addresses have been derived in a logical and the method of inquiry is amazing. Tests results have been presented substantially and statistically, and meaningful. What is encouraging is that the authors have remained in the scope that they had promised the reader. Database security is a nightmare in real world and the authors' inquiry is relevant and has added to the database knowledge.

References

Raluca, A. P., Catherine, M. S., Nickolai, Z., & Hari, B. (2012). CryptDB: Protecting Confidentiality with Encrypted Query Processing. *International Journal of Computer Applications* , *45* (8), 84-99.