MATEC Web of Conferences **226**, 05002 (2018) https://doi.org/10.1051/matecconf/201822605002
DTS-2018

The key management method for the system of end-to-end symmetric database encryption

Vasily Galushka*, *Daniil* Marshakov, *Andrey* Aydinyan, *Olga* Tsvetkova, and *Denis* Fathi
Don State Technical University, 344000 Rostov-on-Don, Russian Federation

Abstract. The article is devoted to the description of ways to manage cryptographic keys for the system of end-to-end symmetric database encryption designed to implement access control mechanisms with the aim of enhancing the capabilities for managing user rights. It describes how to use unique encryption keys for individual database tables, the associated information security risks and approaches to their elimination using cryptography. Separately, the question of implementing the method of managing access to rows of the table by creating a multi-level hierarchy of users using encryption key chains formed through irreversible transformations, as well as the exchange of data by keys. It is proposed to implement it using a crypto container, which is a set of information necessary for the operation of asymmetric encryption algorithms and modified taking into account the peculiarities of the system in question, which allows providing comprehensive information security of user data.

1 Introduction

Previously discussed methods of implementing the system of symmetric database encryption [1] have certain disadvantages. These include, in particular, the impossibility of executing a query to select data with a condition other than strict compliance, as well as the existence of a common encryption key for the entire database, which limits the flexibility of the user rights management system.

The use of end-to-end encryption [2, 3] is effective to protect against unauthorized access to information in databases stored on remote or local servers, which may be accessed by unauthorized persons. This situation, for example, takes place when a website is hosted on the Internet [4, 5] or an outsourcing organization is used to administer the information system and database as its component. At the same time, in the distribution of user rights, we must rely on built-in protection mechanisms for a particular database management system (DBMS) [6], which, although reliable enough, strongly depend on the specific implementation of the DBMS and the vulnerabilities in it [7].

To eliminate these disadvantages, it is necessary to develop a method of key management in the further development through symmetric encryption database system to improve the ability of user rights management and improve the security of the authorization data.

* Corresponding author: galushkavv@yandex.ru

2 Controlling access to tables of databases

The development of access rights management mechanisms for end-to-end symmetric database encryption is possible by modifying the key pair generation algorithm. For this, it is proposed to abandon the single encryption key for the entire database, and as necessary to generate and use such a number of unique keys that will be sufficient for each individual table, column or row. This will allow, managing the distribution of keys, to give the individual user access only to the data set that he needs.

In the first approximation, the described principle can be implemented by adding to the database schema a service table containing the results of computing the “bitwise exclusive or” for the encryption key of each table and user passwords. The value thus obtained will be called the "difference" function:

$$D^t_u = K^t \oplus P_u, \tag{1}$$

where K^t — the key of table t;

P_u — the password of user u;

D^t_u — the “difference” for user u and table t.

In this case, the key K^t is generated programmatically when creating a table using pseudo-random numbers and is an array of 16 bytes, that is

$$K^t = (K^t_1, K^t_2, \ldots, K^t_i, \ldots, K^t_{16}), \tag{2}$$

where K^t_i = Rnd (0, 255), Rnd is the function of generating pseudorandom integers in a given range.

This scheme will allow organizing the allocation of access rights separately for each table by delegating them from a user who has the information necessary to calculate the table key to one or more other users. However, this approach presents a very serious threat to information security. Assuming that the end-to-end database encryption system should prevent information from leaking in the case of bypassing the built-in security features, it should be considered that an attacker can access any information from the user's service table, including any value in the column for storing the “differences”. In addition, since the user has his own password that gives him access to any table, he has the ability to calculate its key. With this information, he can easily get the passwords of other users, as well as the keys to other tables, by simple calculations. To do this he will need to perform a series of transformations described below.

1. Calculation of the encryption key of one of the tables to which user has access:

$$K^1 = D^1_1 \oplus P_1, \tag{3}$$

2. Obtaining the difference of another user and calculating his password:

$$P_2 = K^1 \oplus D^1_2. \tag{4}$$

3. Calculation of the encryption key of another table:

$$K^2 = D^2_2 \oplus P_2. \tag{5}$$

4. Steps 2 – 3 can be repeated until there are no tables to which user can get the key, and the general scheme of such an attack, based on the formulas (3) – (5), can be represented as follows:

$$K^2 = D^1_1 \oplus D^1_2 \oplus D^2_2 \oplus P_1 \tag{6}$$

or in the general form:

$$K^m = D^n{}_n \oplus D^n{}_{n+1} \oplus D^{n+1}{}_{n+1} \oplus D^{n+1}{}_{n+2} \oplus D^{n+2}{}_{n+2} \oplus \ldots \oplus D^m{}_m \oplus P_n. \quad (7)$$

From formulas (6) and (7) it is seen that there is a transitive relationship between the arbitrary table m encryption key and the password of user n. The absence of any restrictions on the values of *n*, *m*, *i* and *j* suggests that the potential exists for any user to access any table.

It also shows from the formula (7) that the reason for the above vulnerability is the ability of any user to get the values D^i_j from the database, which according to (1) are connected by a non-cryptographic reversible conversion with passwords of other users and keys of other tables. In this connection, to eliminate the above vulnerability, it is proposed to encrypt the difference before placing it in the table, using the user's password as the key. This modification of the method does not violate the overall scheme of the database security system, because the difference is used only in the process of calculating the key, when it can be decrypted, and when changing the password when it can be encrypted.

As a result, formula (1) takes the form:

$$D^t{}_u = \mathrm{ENC}(K^t \oplus P_u)_{Pu}, \quad (8)$$

where $\mathrm{ENC}(M)_P$ — the function of symmetric encryption of the message *M* with the key *P*.

The inverse procedure for calculating the table key will then be performed as follows:

$$K^t = \mathrm{DEC}(D^t{}_u)_{Pu} \oplus P_u, \quad (9)$$

where $\mathrm{DEC}(E)_P$ — the function for decrypting the message *M*, encrypted by the symmetric algorithm with the key *P*.

It should be noted that the dependence of $D^t{}_u$ on P_u appeared, so in the formula (6), to obtain D^i_j or D^i_{j+1}, it is necessary to know the password P_j. Knowing only password P_n in this case will not be enough.

3 Controlling access to rows of tables

According to relational theory, the rows of tables are indistinguishable for the database, so standard DBMS tools do not allow separate access to specific row for different groups of users [8, 9]. In practice, this means that even if the user only needs a small amount of information stored in a certain table, he will still be given access to the entire table. This solution is not flexible and can represent a potential leak path for large tables of multi-user databases.

In this regard, it is proposed to implement a way to manage the rights of users to access information from the database by encrypting different table rows with different keys. At the same time, on the one hand, the disadvantages of the relational theory described above will be eliminated, and, on the other hand, the data security is improved.

This system can be organized on the principle of a multi-level self-organizing hierarchy, built using cryptographic keys chain. The root of this hierarchy is the user or the group that can calculate the main key of the table K^t, generating and other operations with which are described in detail in the previous paragraph. The master key must be able to perform encryption / decryption operations with the full set of rows of its table. The lower level of the hierarchy is formed by encrypting the subset of the rows of the table under consideration with the new key K^{t+1}, calculated by the formula:

$$K^{t+1} = \mathrm{md5}(K^t), \quad (10)$$

where md5 — the cryptographic hash function.

Using this or another hash function allows one-way conversion of the master key K^t to a second-level key of the hierarchy K^{t+1} while ensuring that it cannot perform an inverse transformation in an acceptable time.

The described principle can be used to further expand the hierarchy by adding to it an unlimited number of additional lower levels, during which the key of each next level is calculated as a result of applying the hashing function to the key of the previous level. Figure 1 shows the general scheme of encryption using a key chain.

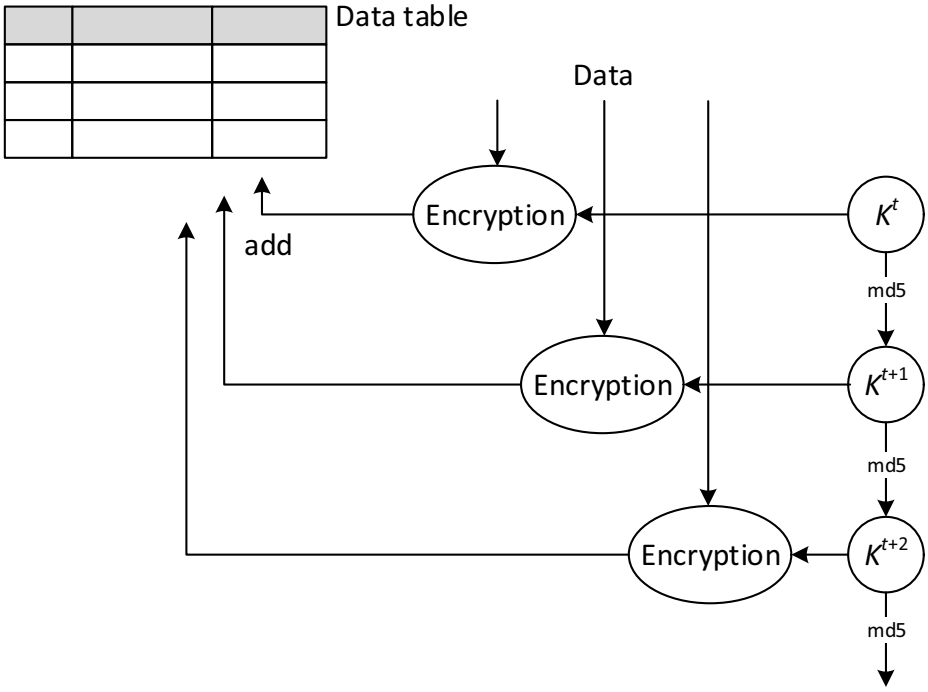

Fig. 1. The general scheme of encryption using a key chain.

With respect to the access rights management system, this means that the user who knows the master key can decrypt the encrypted rows, as well as the rows encrypted with the second level key, by first calculating it, but the user who knows the second level key can decrypt only the rows encrypted by him. It should be noted that as a result of the described method, users that are higher in the hierarchy, by default, have access to all data added by users of the lower level.

4 The method of forming a crypto container for transferring symmetric encryption keys

The method, described above, requires an efficient and safe way of transferring keys through the hierarchy of users. An important point here is the possibility of providing a key to a user who is not currently online, which means that the transfer in this case is understood not only as sending information to a remote device, but also, more importantly, organizing its long secure storage. By analogy with the previously discussed methods, all the necessary information is suggested to be stored in the database, but proceed from the

fact that the built-in data protection mechanisms can be bypassed or compromised, which leads to the need for encryption.

This task can be solved using asymmetric encryption algorithms, modifying them taking into account the features of the database protection system under consideration. Typically, asymmetric encryption involves the creation of two keys: public key to encrypt message and private key to decrypt encrypted message. In this case, the private key must be stored only on the recipient's device and not be outside its limits, and the public key can be freely distributed to senders.

In order to avoid the need to store the private key on the user's device and enable users to connect to the database and work with it from any device, it is proposed to create and use for the key exchange a secure crypto container stored directly in the database.

This crypto container is a set of information sufficient to transfer to its owner any data, their long-term storage and ensuring the inability to obtain this data by someone else. It consists of a public key, a cell for storing data encrypted with a public key and a private key encrypted with the password of the owner of the container:

$$C = <pub, E, PR_e>, \tag{11}$$

where C — the crypto container;

pub — the public key;

E — the cell for storing encrypted information;

$PR_e = \mathrm{ENC}(pr)_{Pu}$ — the encrypted private key;

pr — the private key generated by the asymmetric encryption algorithm;

P_u — the password of the crypto container owner.

A user who wants to put information in a crypto container receives from the database the public key pub, encrypts the information using it and writes the encrypted information in the field E. With respect to the system of end-to-end database encryption considered, this information is the key of the lower level of the hierarchy obtained by hashing the user's key, performing the addition. That is:

$$E = \mathrm{ASSYM\_ENC}(\mathrm{md5}(K^t))_{pub},$$

where ASSYM_ENC — any asymmetric encryption function.

To extract data from the container, its owner first decrypts the private key using its password, and then decrypts the message containing the symmetric table key of its hierarchy level with the received private key:

$$K^{t+1} = \mathrm{ASSYM\_DEC}(E)_{\mathrm{DEC}(PRe)}.$$

Next, the K^{t+1} key is used to calculate the “difference” according to the formula (1), and the message from the crypto container is destroyed by a simple query to the database to clear the corresponding cell.

5 Conclusion

The proposed methods for managing cryptographic keys for a system of end-to-end database encryption ensure the distribution of user rights regardless of the database used, based solely on symmetric and asymmetric encryption algorithms, whose crypto-stability can reliably protect user data from leaks and unauthorized access. The described method of constructing a multi-level hierarchical system of access to individual rows of a table using key chains can serve as an important addition to standard DBMS tools based on relational theory, supplementing them with the possibility of more precise adjustment of user rights.

The complex application of the methods proposed in the article allows us to ensure the protection of user data regardless of the DBMS, based only on modern methods of cryptography.

References

1. V. Galushka, A. Aydinyan, O. Tsvetkova, V. Fathi, D. Fathi, J. Phys.: Conf. Ser., **1015** 042003 (2018)
2. S. Floyd, K. Fall, IEEE/ACM Transactions on Networking **7**, 4, 458 (1999)
3. C. Wood, E. Uzun, IEEE 11th Consumer Communications and Networking Conference, Las Vegas, NV, USA, 858 (2014)
4. S. Tajalizadehkhoob, Delft University of Technology, doi:10.4233/uuid:c343a2dd-15d1-4921-9b45-f00ee38177d8
5. S. Tajalizadehkhoob, T. Goethem, M. Korczyński, ACM SIGSAC Conference on Computer and Communications Security, Dallas, Texas, USA, 533-567 (2017)
6. Shelly, K. Gurleen, Int. J. of Engineering and Management Research **7**, 3, 269 (2017)
7. M. Yunus, S. Krishnan, N. Nawi, E. Surin, Int. J. on informatics visualization **1**, 192 (2017)
8. R. Elmasri, S. Navath, *Fundamentals of Database Systems,* 7th edition (Addison Wesley, 2016)
9. J. Harrington, *Relational Database Design and Implementation* (Morgan Kaufmann, Elseiver, Cambridge, 2016)
10. W. Lu, *IEEE Transactions on Communications*, **37** (1989)