American Journal of Applied Sciences 1 (4): 327-331, 2004
ISSN 1546-9239

A Secure Database Encryption Scheme

Zongkai Yang, Samba Sesay, Jingwen Chen and Du Xu
Department of Telecommunication and Information Technology
Huazhong University of Science and Technology, Wuhan, 430074, Peoples Republic of China

Abstract: The need to protect database, would be an every growing one especially so in this age of e-commerce. Many conventional database security systems are bugged with holes that can be used by attackers to penetrate the database. No matter what degree of security is put in place, sensitive data in database are still vulnerable to attack. To avoid the risk posed by this threat, database encryption has been recommended. However encrypting all of database item will greatly degrade the performance of the database system. As an optimal solution this paper presents a database encryption scheme that provides maximum security, whilst limiting the added time cost of encryption and decryption.

Key words: Database, Database Security, Cryptography, Cryptographic Keys Encryption, Decryption, Access Control

INTRODUCTION

In today's economy databases symbolize one of the most valuable assets. They form the basis for e-business, e-commerce, Enterprise Resource Planning (ERP) and other sensitive activities. Many organizations cannot work properly if their database is down; they are normally referred to as mission-critical system. Along with the wide application of databases comes the need for its protection. Universally, huge amount of effort, time and resources are been spent in trying to make database systems meet security requirements. These security requirements include:

- Prevention of unauthorized disclosure of information
- Prevention of unauthorized modification of information
- Prevent denial of service
- Prevent system penetration by unauthorized person
- Prevent the abuse of special privileges

Designing a database that will achieve these security requirements is very difficult, since a database system processes large amount of data in complex ways. The result is that most conventional database systems have leaks that attacker can use to penetrate the database. No matter what degree of security is put in place, sensitive data in database are still vulnerable to attack. A remedy therefore is to turn to cryptographic means of storing data. Encrypting data stored in a database can prevent their disclosure to attackers even if they manage to circumvent the access control mechanism. Thus cryptographic technique can ensure excellent security for databases, by reducing the whole security process down to the protection of only few cryptographic keys. Nonetheless the time cost involved in encrypting and decrypting data items can greatly degrade the performance of the database system. A compromise solution must be found between performance and security, by encrypting only sensitive data in tables or column in the database. The objective of this paper is to propose a secure database encryptions scheme that provides maximum security, whilst limiting the added time cost for encryption and decryption. The encryption technique considered is Data Encryption Standard (DES), but the scheme is also applicable to other cryptographic techniques and standards.

[1, 2] has contributed immensely to efforts towards providing Database Security through Cryptographic means. In [1] he provides solutions to the security problems of field based protection. In his paper he presents a comparative study on implementing encryption at various database levels i.e table, attribute and field (element) levels. In [2] he tries to solve database integrity problem through cryptographic checksum.

[3] proposed a system model using two trusted modules (security kernels), and use tags to ensure integrity. [4] proposed a blend record and field techniques, based on remainder theorem. All the above approaches are different from this study in terms of structure, key management and the implementation procedures.

MATERIALS AND METHODS

The proposed scheme adopts a two-level relational database system, wherein subjects (users) are assigned to either of two levels, L_1 (low) and L_2 (high). All Subjects have access right to their own personal private data (P). And in addition, subjects in L_1 have access right only to unclassified (U) public data, whilst those in L_2 have access right to

Table 1: Elements of the Model

Set	Elements	Semantics
S	$\{s_1, s_2, s_3, \ldots\ldots\ldots, s_n\}$	Subject (Users)
O	$\{A_j, X_{ij}\}$ i = row ; j = column	Database object: {Attribute, data}
L(s)	$\{L_1, L_2\}$ $L_1 < L_2$	Clearance level of subjects {low, High}
L(o)	{[U, C], P} U < C < P	Classification of objects {[unclassified,classified], private}
K	$\{K_1, K_2, \ldots\ldots, K_r\}$	Categories: special access privileges to sensitive object.
V	$\{s\ ,p ,\times, INSERT\ , DEL, UPDATE\}$	Access operations: {select, project, join, insert, delete, update}

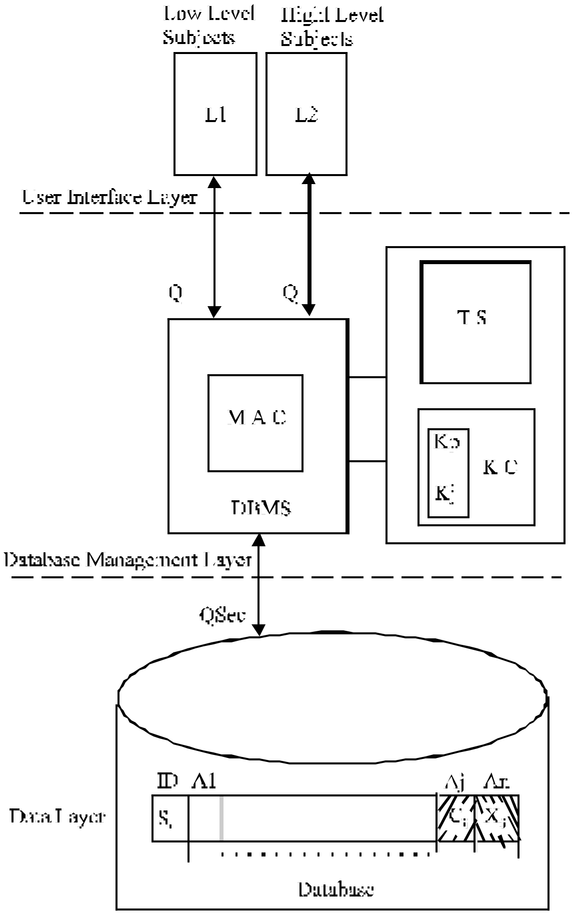

Fig.1: General Structure of Model

both unclassified and classified (C) public data. The access rights of subjects in L_2 to classified data is however limited to their "Need-to-know" sensitive data. The elements used in the scheme are defined in Table1 [5]. The database objects are classified into public [unclassified, classified] and private objects.

Unclassified Data: are non-sensitive data that forms the bulk of the database and are open to all users for access.

Classified Data: are sensitive data that has restricted access. Example salary of employees is considered a sensitive data not to be disclosed to all subjects. But some subjects such as account manager who has need-to-know salaries of all employees should be granted access privilege to employees' salary.

Private Data: are user's personal secret data like credit card number, which should only be available to them or for which others need to take direct permission from them before access.

Because database is a collection of related data, it is assumed that classified data are held under the same database attributes (column) different from those for unclassified data. Therefore classification for public [unclassified, classified] object is done at attribute level, whilst private objects are classified at data element level.

The system structure of the model is shown in Fig. 1. Basically the model was divided into three layers:

The first layer is the user interface layer, which contains two blocks, one for level1 subjects and the other for level2 subjects. All subjects posses a unique key K_P in the form of a certificate that they use when accessing their encrypted private data in the database. The second layer is the database management layer, which also contains two main blocks, one that implements the mandatory access control (MAC) to the database, and another that houses a tamper-free controller that is closely linked with a trusted subject.

The functions of the controller (KC) are:

- * To generate and safely store two sets of encryption keys. K_P for each user's private encrypted data and K_j for encrypted classified data
- * To encrypt classified and private data before being storage in the database
- * To decrypt classified and private data in the

response to users queries that satisfied security requirements

* To perform integrity check on users returned data

* To facilitate authentication of users when necessary.

The trusted-subject (TS) is in charge of:

* Registering new subjects record

* Deleting subjects and objects

* Declassifying subjects and objects

* Updating classified and private data.

The bottom layer contains the database. In order to facilitate fast retrieve of data, the database system stores unclassified data in the clear, whilst classified and private data are stored in encrypted form. The first field of every record uniquely identifies the record, and serves as its primary key (ID). Primary key should not be confused with cryptographic keys used to decrypt ciphertext.

RESULTS AND DISCUSSION

Implementation: It is described that how the model operates to provide security to the database. As mentioned earlier classified and private data are stored in encrypted form in the database. If an intruder manages to circumvent the mandatory access control mechanism and penetrate the database, sensitive data are still concealed from them as they lack the necessary decryption keys. This provides confidentiality to sensitive data stored in the database.

To control user's access to information in the database, the MAC maintains an authorized view of the database for all subjects. As the labeling of public (unclassified and classified) objects is at the attribute level, the maximum authorized view of a subject to a relation R in the database is defined as follows:

Let L (A_i) denotes the classification of attribute A_i in the relation R. The set of attributes authorized to subject 's' with clearance L(s) is defined by:

$$A.auth(s) = \{A_i \in R \mid L(A_i) \le L(s)\}$$

Thus the maximal authorize view of s on R is:

$$s.MAX\_VIEW(R) = p_{A.auth(s)}(R).$$

This implies that any query Q_i from a subject has to be augmented to a secure form Q_{sec} that reflects the authorized view of the subject. As an example let us consider three classes of increasingly complex queries: select, select-project, and select-project-join.

Select Query: For a query Q_1 = SELECT all (records) from R (relation) WHERE (constrained by) F.

$$Q_1 = s_F(R).$$

The secure version of Q_1 becomes $Q_{sec.1} = s_F(p_{A.auth(s)}(R)).$

Select-Project Query: For a query Q_2 = SELECT all records from R WHERE F, and PROJECT on attribute set A.

$$Q_2 = p_A(s_F(R)).$$

And the secure version of Q_2 is

$$Q_{sec.2} = p_A(s_F(p_{A.auth(s)}(R))).$$

Select-Project-Join Query: For a query Q_3 that joins two sub queries Q_a and Q_b, where:

$$Q_a = p_{A_a}(s_F(R_a)).$$

$$Q_b = p_{A_b}(s_F(R_b)).$$

Then $Q_3 = p_{R_aA_a,R_BA_B}(s_F(R_a \times R_b)).$

And the secure version is

$$Q_{sec.3} = p_{R_aA_a,R_BA_b}(s_F(p_{A_a.auth(s)}(R_a) \times p_{A_B.auth(s)}(R_b))).$$

In general, for all users query to the database, MAC takes as input [6]:

* User's ID

* User's clearance L(s)

* User's query Q_i

* D, a set of constraints that prevent users from making direct modifications to data and classes of subjects and objects.

If no security violation is detected, MAC provides an answer in responds to Q_i.

To maintain the integrity of the database, subjects are constrained from carrying out direct access operations that may change the state of the database. Such operations include Create, Update, Delete, and change of Classification. Request for such operations should be forward to the trusted subject who evaluates them with respect to security violations and if safe allows their execution. Further more the enciphered text also serves as checksum against which returned sensitive data are crosschecked. The restriction on the subjects can be implemented using the following two constraints

Constraint 1: For all $s \in S$, If $s \neq TS$,
S* = S, L(s)* = L(s).

Constraint 2: For all $o \in O$,
If $s \neq TS$
O* = O

L (o)* = L (o)
Content (o)* = Content (o)

The Star * in front of variables refer to their new state, whilst unstared variables their old state. Constraint1 removes the right from all subjects to create other subjects or change their clearance level. And constraint 2 removes the right from subject to directly create new object or change the classification or content of objects. In our proposed scheme Authentication and verification of query originator only applies for access to private data. Subjects are required to bind queries for private data with their special certification key as discussed below. Finally to block inference channels, all functional dependent objects (attributes) are made to have the same classification label.

Key Management: The security of enciphered sensitive data in the database depends on the protection of the keys. For this reason, management of keys is vital to the overall security of the database system. Key management includes every aspect of the handling of keys from their generation to their eventual destruction [7]. The proposed scheme makes use of three sets of encryption keys K_j, K_p and K for classified data, private data and controller's master key respectively. Their generation and distribution process are as discussed below:

Key Generation: An ideal method of key generation would be one that chose the key at random. Unfortunately, absolute randomness to key generation is difficult to achieve. A possible source of generating random key values is through pseudo-random key generators with different seed startup values. This form the bases for generating the controller master key K. Generation of the other two keys are as follows:

Generation of Classified Data Keys: With classified data labeled at attribute level. Let A_j be an attribute identifier, and K the controller master key. An attribute key to classified data is defined by $K_j = g\ (A_j, K)$ or $K_j = E_K(A_j)$, where g is a key generating function.

Generation of Private Data Keys: Users' private data elements should have unique key. Therefore classification for private data is at data element level. It is assumed that the first field (ID) in every record uniquely identifies the record; i.e, it is the primary key of the database. For every private data X_{ij}, the private key is defined by:

$K_p = g\ (ID, A_j ,T, K)$ or

$$K_p = E_K(K_t) = E_K(K_i \oplus T) = E_K[(ID \oplus A_j) \oplus T].$$

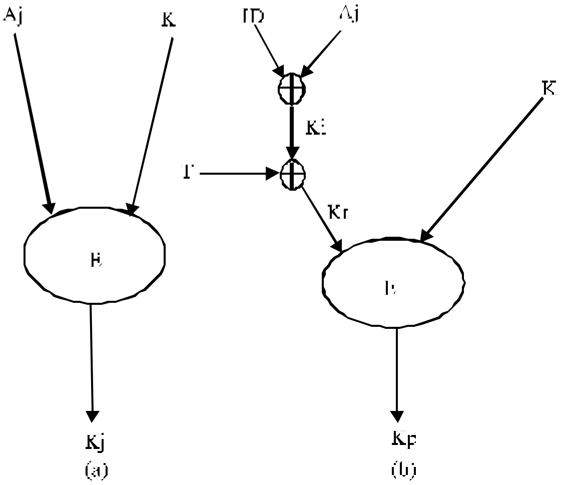

Fig. 2a and b: Hierarchy Structure for Generating Classified and Private Keys, respectively

where, T is time stamp included to ensure that K_p is unique every time it is updated and $\oplus$ is the exclusive-or operator. Using Data Encryption Standard (DES) technique, identifiers are assume to be at most 8 byte long, and are padded as necessary to fill 8 bytes (Fig. 2). Also the key generating function is chosen such that the probability of getting repeated keys is low and also be computationally infeasible to determine one element key from other element keys [8].

Key Distribution: The generated classified and private data keys are stored in a tamper free controller. A copy of a private data key K_p is sent to the owner of the private data as a form of certificate. Request for a private data should be accompanied with its certificate for such request to be honored. Subjects can exchange their certificate with others who they wish to allow view to their private data, and can later request for an update. However only a private data owner can request update to private data or it certificate. Thus the controller must authenticate the originator for update to private data. In general private data keys are refreshed more often than classified data keys as they have greater changes of being exposed.

Encryption and Decryption: The classified and private data elements X_{ij} are encrypted/decrypted using Data Encryption Standard (DES) technique. Where X_{ij} is less than the 8-byte block size of DES, X_{ij} is replicated as many times as necessary to fill the block. If X_{ij} however exceeds the block size, then the encryption is performed using cipher block chaining with initialization block [1]. Figure 3a and b illustrates the encryption and decryption of classified data and private data element, respectively. The controller relies greatly on the mandatory access control mechanism and trusted subject to control access to keys. If response to a user's secure query includes

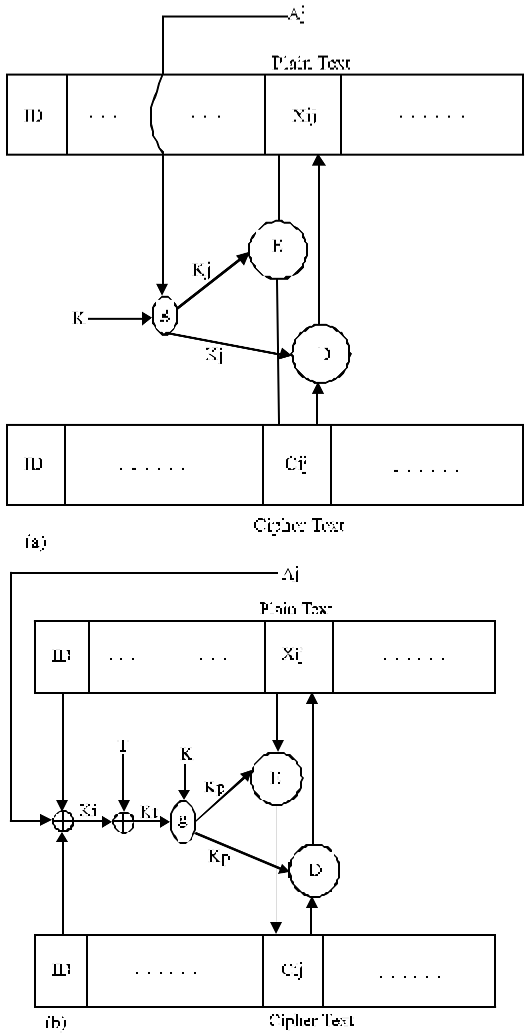

Fig. 3a and b: The Process of Encryption and Decryption of Classified and Private Data, respectively

encrypted data, the MAC sends a command to the controller to decrypt them before being forwarded to the users. The reverse process is performed when user's query requires storage or update to sensitive data in the database.

One other advantage of our proposed scheme is that the ciphertext also serves as cryptographic checksum to protect against modification to data and its classification label. This is made possible because short plaintext data are replicated many times as necessary to fill the encryption block size. Thus because of this redundancy the ciphertext can provide authenticity as well as secrecy.

Checksum $S_{ij} = C_{ij} = E_{K_j}(X_{ij})$ for classified data, and $S_{ij} = C_{ij} = E_{K_p}(X_{ij})$ for private data [1].

CONCLUSION

This paper investigates the role cryptograph can play in database security. In a database system, sensitive data stored in the clear can be vulnerable to attack. No matter the amount of security measures taken, there would always be some security leaks which attackers can use to penetrate the database. However, if sensitive data are encrypted before storage in the database, the risk from security leaks can be eliminated. And the whole database security issue can be reduced down to the protection of few cryptographic keys.

The proposed database encryption scheme is considered efficient because it provides maximum security to the database, whilst the added time cost for encryption and decryption is very minimal. All aspects of security concerned from confidentiality, access control, integrity, authentication to non-repudiation were addressed. Furthermore, the use of authorized view and a tamper free controller solves indirect threats from inference channel, insecure information flow and ciphertext searching.

To reduce the time spent on encryption and decryption, the scheme divides the database into sensitive (classified and private) and non-sensitive (unclassified) data. Non-sensitive data, which forms the bulk of the database, are stored in the clear, facilitating their fast retrieval. Although classified data (part of the public data) are stored in encrypted form, their decryption process is very fast, as only one key is needed to decrypt a whole column (attribute) of encrypted classified data. Also, although accessed encrypted private data has to be decrypted separately using their unique keys. Requests for private data are very seldom and are carried out only once on a while. This makes the time cost for their encryption and decryption to have less significance on the overall performance of the scheme.

The only downside side of the scheme is that queries such as sums, averages, counts and other statistical functions that aggregate over a range of data in the database cannot be performed directly. However users themselves at the user interface layer can do such task.

REFERENCES

1. Denning, D.E.,1983. Field Encryption and Authentication. Proc.of CRYPTO 8.9,Plenum Press.
2. Denning, D. E., 1984. Cryptographic checksums for multilevel data security. Proc. of Symp.on Security and Privacy. IEEE Computer Society, pp: 52-61.
3. Downs, D. and G.J. Popek, 1977. A Kernel Design for a Secure Data Base Management System. Proc.3rd Conf. Very Large Data Bases. IEEE and ACM, New York, pp: 507-514.
4. Davida, G.I, D.L. Wells and J.B. Kam, 1981. A Database Encryption System withsub keys. ACM Trans. On Database Systems 6:2.
5. Bell, D.E and L.J LaPadula, 1973. Secure Computer Systems: Mathematical Foundations and Mdel., M74-244. The MITRE Corp, Bedford.
6. Brodsky, A, C. Farkas and S. Jujodia, 2000. Secure Databases: Constrains, Inferences, Channels and Monitoring Disclosures. IEEE Transactions on Knowledge and Data Engineering, 12:6.
7. Davis, D.W. and W.L. Price, 1984. Security for Computer Networks. A wiley-interscience Publication.
8. William Stallings, 1999. Cryptography and Network Security Principles and Practice (2nd Edn.). Prentice-Hill Inc.