Design Application to Encrypt and Decrypt of Sensitive Columns on Client Side

Arafat Al-dhaqm, Dr. Majid Bakhtiari

Abstract— Most of the corporations these days have suffering from many threats that try to breaches the confidentially and integrity of corporations databases. Insider and outsider attacks are some examples of these threats. Encryption, authentication and authorization are some of mechanisms that the corporations have been implemented, to protect their valuables information. However, the threats still growth day by day owing to the weaknesses and flaws in these mechanisms. This paper proposed an application, which encrypt and decrypt the database column on the client side. It encrypts data in the database column with variable length key. This key is unique for each data item even in same item; the key is not fixing, it based on the length of the plaintext which sent to the database item. The data will sent and saved to the database server as encrypted form. The advantages of this novel notion are: first, nobody can decrypt the data item outside of the application easily, due to the variable length of key. Second, the data will sent to the database in encrypted form, so the man in the middle cannot compromise intercept data easily. Third, the authorized entity on the server database such as DBA cannot tamper the data outside of this application.

Index Terms --- **Encryption, Decryption, DBA, Privacy, Insider attacks, Outsider attcak;**

1 INTRODUCTION

Today, almost of the effective data processing is primal and important issue for every scientific, academic, or business corporations. Hence, the companies and corporations end up installing and managing database management systems to soakage different data processing needs. Although it is possible to buying the mandatory hardware, expand database products, create network connectivity, and assigned the professional people who run the system, as a traditional solution, this solution is expensive and impractical in the database systems which growth day by day. Different costs for the traditional solution which is mentioned above. The costs of the hardware, software, and network are decreasing constantly. People costs, however, generally, do not decrease. In the future, it is likely that computing solution costs will be dominated by people costs [1]. Solutions for database backup, recovery strategies and, reorganization to repair space or to restore the suitable arrangement of data.

Migration or mixing from one database version to the next, are an art still in its beginning [9]. Parts of a database solution, if not the entire solution usually become unavailable during version change. An organization that provides database service has an opportunity to do these tasks and offer a value proposition provided it is efficient.

Dr. Majid Bakhtiari, **Senior** Lecturer *Faculty* of Computer Science & Information System UNIVERSITY TECHNOLOGY MALAYSIA **Skudai, 81310 JohoreMALAYSIA**.E-mail: **bakhtiari.majid@gmail.com** , **bakhtiari@utm.my.**

Arafat Mohammed Rashad Aldhaqm is currently pursuing masters degree program in computer science (Information Security) in University Technology Malaysia. E-mail: arafat_aldoqm@yahoo.com

The technological aspects of developing database as service lead to new research challenges. First and foremost is the issue of data privacy. In the database service provider model, user data needs to reside on the premises of the database service provider. Most corporations view their data as a very valuable asset. The service provider would need to provide sufficient security measures to guard the data privacy. Privacy on the Internet is an issue that is of significant interest. There are two fundamental issues:

1- Privacy of data during transmission.

2- Privacy of stored data.

The first issue, privacy during network transmission, has been studied widely in the Internet area and addressed by the Secure Socket Layer protocol (SSL) [8] and Transport Layer Security (TSL) protocol [4]. The second issue, privacy of stored data in relational databases is less studied and of greater relevance to database as a service model. If database as a service is to be successful, and customer data is to reside on the site of the database service provider, then the service provider needs to find a way to preserve the privacy of the user data. There needs to be security measure in place so that even if the data is stolen, the thief cannot make sense of it. Encryption is the perfect technique to solve this problem. There are two dimensions to encryption support in databases. One is the granularity of data to be encrypted or decrypted. The field, the row and the page. The field may appear to be the best choice, because it would minimize the number of bytes encrypted. However, as we have discovered, practical methods of embedding encryption within relational databases entail a significant startup cost for an encryption operation. Row or the page level encryption consumes this cost over larger data. The second dimension is software versus hardware level implementation of encryption algorithms. Critical business data in databases is an obvious target for attackers. Although access control has been de-

ployed as a security mechanism almost since the birth of large database systems, for a long time security of a DB was considered an additional problem to be addressed when the need arose, and after threats to the secrecy and integrity of data had occurred [3]. Now many major database companies are adopting the loose coupling approach and adding optional security support to their products. You can use the encryption features of your Database Management System (DBMS), or perform encryption and decryption outside the database. Each of these approaches has its advantages and disadvantages. Adding security support as an optional feature is not satisfactory, since it would always penalize system performance, and more importantly, it is likely to open new security holes. Database security is a very big research area [6, 3] and includes topics such as statistical database security [10], and related papers in designing information systems that protect the privacy and ownership of individual information while not impeding the flow of information, include [2, 3, 4, 5]. which cause loss data for example, power outage, hardware (hard disk failure, CPU failure), fires, and system crashed earthquake, and floods, lightning and human error. Human errors (intended or unintended errors) are one of the biggest factors which cause loss data and difficult to detect it, such as authorized delete, update and overwriting data [1].

Organization of the Paper

The rest of this paper is organized as follow: in section 2 the database-layer encryption, section 3 Database Storage-Tier Encryption, section 4 Authorized Entities Management Issues, section 5 Security Dictionary Guide, Section 6 Whole Accountability, section 7 Select the Formatting of the storage of Encrypted Information, 8 Encryption of Constraints, section 9 Indexing encrypted columns, section 10 the proposed application, section 11 conclusion.

2. Database -layer Encryption

Database-level encryption allows enterprises to secure data as it is written to and read from a database. This type of deployment is typically done at the column level within a database table and, if coupled with database security and access controls, can prevent theft of critical data. Database-level encryption protects the data within the DBMS and also protects against a wide range of threats, including storage media theft, well known storage attacks, database-level attacks, and malicious DBAs. Database-level encryption eliminates all application changes required in the application-level model, and also addresses a growing trend towards embedding business logic within a DBMS through the use of stored procedures and triggers. Since the encryption/decryption only occurs within the database, this solution does not require an enterprise to understand or discover the access characteristics of applications to the data that is encrypted. While this solution can certainly secure data, it does require some integration work at the database level, including modifications of existing database schemas and the use of triggers and stored procedures to undertake encrypt and decrypt functions. Additionally, careful consideration has to be given to the performance impact of implementing a database encryption solution, particularly if support for accelerated index-search on encrypted data is not used. First, enterprises must adopt an approach to encrypting only sensitive fields. Second, this level of encryption must consider leveraging hardware to increase the level of security and potentially to offload the cryptographic process in order to minimize any performance impact. The primary vulnerability of this type of encryption is that it does not protect against application-level attacks as the encryption function is strictly implemented within the DBMS.

3.0 Database Storage-Tier Encryption

Storage-level encryption enables enterprises to encrypt data at the storage subsystem, either at the file level or at the block level. This type of encryption is well suited for encrypting files, directories, storage blocks, and tape media. In today's large storage environments, storage-level encryption addresses a requirement to secure data without using LUN (Logical Unit Number) masking or zoning. While this solution can segment workgroups and provides some security, it gives two limitations:

1- It only protects against a narrow range of threats, namely media theft and storage system attacks. However, storage-level encryption does not protect against most application- or database-level attacks, which tend to be the most prominent type of threats to sensitive data.

2- Current storage security mechanisms only provide block-level encryption; they do not give the enterprise the ability to encrypt data within an application or database at the field level. Therefore, one can encrypt an entire database, but not specific information housed within the database.

4.0 Authorized Entities Management Issues

Authorized entities such as users, process and operating systems have rights to access to the database with suitable privileges. To access database resources, a user must have a database account. User account management is the basis for the overall database system security. A DBA has the responsibility to create and maintain all DB user accounts, which Is a large portion of her/his system administration effort. During account creation, the DBA specifies how the newly created user will be authenticated, and what system resources the user can use. When a user wants to connect to a database, they must identify themselves to the server and the server will verify her/his identity using the pre-specified authentication method. Current commercial RDBMSs support many different kinds of identification and authentication methods, among them are:

1- password-based authentication [12]
2- host-based authentication [4, 12, 11],
Essentially, all methods rely on a secret known only to the connecting user. It is vital that a user should have total control over her/his own secret. For example, only she/he should be able to change her/his password. Other people can change a user's password only if they are authorized to do so. In a DB system, a DBA can reset a user's password upon the user's request, probably because the user might have forgotten her/his password. However, as we have noticed before, the DBA can temporarily change a user's password without being detected and caught by the user, because the DBA has the capability to update (directly or indirectly) the system catalogs. Design Issues in Encryption and Key Management. The most important problem in using encryption/decryption is key management implementation across all database platforms in an enterprise. When we consider incorporating encryption in a database server, there are two design issues:

1. There should be a way for a user to indicate that some data should be encrypted before
2. Storage and an option to send encrypted data to the application tier.
3. There should be a way for a user to specify (explicitly or implicitly) a key that will be used for data encryption and optional HSM (Hardware Security Module) support.

5.0 Security Dictionary Guide

A traditional data directory stores all of the information used to manage the objects in a database. A data directory consists of many catalog tables and views. It is generally recommended that users (including DBAs) do not change the contents of a catalog table manually. Instead, those catalogs will be maintained by the DB server and updated only through the execution of system commands. However, a DBA can still make changes in a catalog table if she/he wants to do so. To prevent unauthorized access to important security-related information, we introduce the concept of security catalog. A security catalog is like a traditional system catalog but with two security properties:
It can never be updated manually by anyone, and its access is controlled by a strict authentication and authorization policy.

6.0 Whole Accountability

From an administration point of view, a DBA (Database Administrator) is playing an important and positive role. However, when security and privacy become a big issue, we cannot simply trust particular individuals to have total control over other people's secrecy. This is not just a problem of trust, it is a principle. Technically, if we allow a DBA to control security without any restriction, the whole system becomes vulnerable because if the DBA is compromised, the security of the whole system is compromised, which would be a disaster. However if we have a mechanism in which each user could have control over their own secrecy, the security of the system is maintained even if some individuals do not manage their security properly. Access control is the major security mechanism deployed in all RDBMSs. It is based upon the concept of privilege. A subject (i.e., a user, an application, etc.) can access a database object if the subject has been assigned the corresponding privilege. Access control is the basis for many security features. Special views and stored procedures can be created to limit users' access to table contents. However, a DBA has all the system privileges. Because of their ultimate power, a DBA can manage the whole system and make it work in the most efficient way. However, they also have the capability to do the most damage to the system. With a separated security directory the security administrator sets the user permissions. Thus, for a commercial database, the security administrator (SA) operates through separate middle-ware, the Access Control System (ACS), used for:
1- Authentication
2- Verification,
3- Authorization
4- Audit
5- Encryption and decryption.
The ACS is tightly coupled to the database management system (DBMS) of the database. The ACS controls access in real-time to the protected fields of the database. Such a security solution provides separation of the duties of a security administrator from a database administrator (DBA). The DBA's role could for example be to perform usual DBA tasks, such as extending tablespaces etc, without being able to see (decrypt) sensitive data. The SA could then administer privileges and permissions, for instance add or delete users. For most commercial databases, the database administrator has privileges to access the database and perform most functions, such as changing password of the database users, independent of the settings by the system administrator. An administrator with root privileges could also have full access to the database. This is an opening for an attack where the DBA can steal all the protected data without any knowledge of the protection system above. The attack in this case is based on the DBA impersonating another user by manipulating that user's password, even though the user's password is enciphered by a hash algorithm. An attack could proceed as follows:-
1- The DBA logs in as themselves.
2- The DBA then reads the hash value of the user's password and stores this separately.
3- The DBA copies all other relevant user data, creating a snapshot of the user, before any alteration.
4- The DBA executes the command "ALTER USER username IDENTIFIED BY newpassword".
5- The next step is to log in under the user name "username"

with the password "newpassword" in a new session.

6- Finally the DBA resets the user's password and other relevant user data with the previously stored hash value. Thus, it is important to further separate the DBA's and the SA's privileges. For instance, if services are outsourced, the owner of the database contents may trust a vendor to administer the database. The DBA role then belongs to an external person, while the important SA role is kept within the company, often at a high management level. Thus, there is a need for preventing a DBA to impersonate a user in a attempt to gain access to the contents of the database. The method comprises the steps of: adding a trigger to the table, the trigger at least triggering an action when an administrator alters the table through the database management system (DBMS) of the database; calculating a new password hash value differing from the stored password hash value when the trigger is triggered; replacing the stored password hash value with the new password hash value. Similar authentication verification may also be implemented if VPN (Virtual Private Network) based connection and authentication is used. The first security-related component in an RDBMS (and actually in most systems) is user management. A user account needs to be created for anyone who wants to access database resources. However, how to maintain and manage user accounts is not a trivial task. User management includes user account creation, maintenance, and user authentication. A DBA is responsible for creating and managing user accounts. When a DBA creates an account for user Alice, they also specify how Alice is going to be authenticated, for example, by using a database password. The accounts and the related authentication information are stored and maintained in system catalog tables. When a user logs in, they must be authenticated in the exact way as specified in their account record. However, there is a security hole in this process. A DBA can impersonate any other user by changing (implicitly or explicitly) the system catalogs and they can do things on a user's behalf without being authorized/detected by the user, which is a security hole. A DBA's capability to impersonate other users would allow them to access other users' confidential data even if the data are encrypted.

7. Select the Formatting of the storage of Encrypted Information

Application code and database schemas are sensitive to changes in the data type and data length. If data is to be managed in binary format, varbinary can be used as the data type to store encrypted information. On the other hand, if a binary format is not desirable, the encrypted data can be encoded and stored in a 'varchar' field. There are size and performance penalties when using an encoded format, but this may be necessary in environments that do not interface well with binary formats, if support for transparent data level encryption is not used. In environments where it is unnecessary to encrypt all data within a database, a solution with granular capabilities is ideal. Even if only a small subset of sensitive information needs to be encrypted, additional space will still be required if transparent data level encryption is not used. The secure data level encryption for data at rest can be based on block ciphers. The proposed solution is based on transparent data level encryption, with Data Type Preservation that does not change ASCII Data Field Type or length. The solution provides a cost effective implementation, avoiding changes of millions of lines of business code in larger enterprise information systems. The solution offers an effective last line of defense and provides:

1- Selective column-level data item encryption
2- Cryptographically enforced authorization
3- Hardware or software key management
4- Secure audit and reporting
5- Enforced separation of duties.

The method is cryptographically strong and:

1- Works with any DBMS.
2- Can use different character sets
3- Requires application on database changes.
4- No programming language dependence
5- Is fail safe
6- Requires no DBA intervention
7- The loader and queries function normally
8- Has accelerated search capabilities based on partial encryption of data and accelerated search index.

8.0 Encryption of Constraints

Encrypted columns can be a primary key or part of a primary key, since the encryption of a piece of data is stable (i.e., it always produces the same result), and no two distinct pieces of data will produce the same cipher text, provided that the key and initialization vector used are consistent. However, when encrypting entire columns of an existing database, depending on the data migration method, database administrators might have to drop existing primary keys, as well as any other associated reference keys, and re-create them after the data is encrypted. For this reason, encrypting a column that is part of a primary key constraint is not recommended if support for accelerated index search on encrypted data is not used. Since primary keys are automatically indexed there are also performance considerations, particularly if support for accelerated index-search on encrypted data is not used. A foreign key constraint can be created on an encrypted column. However, special care must be taken during migration. In order to convert an existing table to one that holds encrypted data, all the tables with which it has constraints must first be identified. All

referenced tables have to be converted accordingly. In certain cases, the referential constraints have to be temporarily disabled or dropped to allow proper migration of existing data. They can be re-enabled or recreated once the data for all the associated tables is encrypted. Due to this complexity, encrypting a column that is part of a foreign key constraint is not recommended, if automated deployment tools are not used. Unlike indexes and primary keys, though, encrypting foreign keys generally does not present a performance impact.

9.0 Indexing encrypted columns

Indexes are created to facilitate the search of a particular record or a set of records from a database table. Plan carefully before encrypting information in indexed fields. Look-ups and searches in large databases may be seriously degraded by the computational overhead of decrypting the field contents each time searches are conducted if accelerated database indexes are not used. This can prove frustrating at first because most often administrators index the fields that must be encrypted social security numbers or credit card numbers. New planning considerations are needed when determining what fields to index; if accelerated database indexes are not used. Indexes are created on a specific column or a set of columns. When the database table is selected, and WHERE conditions are provided, the database will typically use the indexes to locate the records, avoiding the need to do a full table scan. In many cases searching on an encrypted column will require the database to perform a full table scan regardless of whether an index exists. For this reason, encrypting a column that is part of an index is not recommended, if support for accelerated index-search on encrypted data is not used.

10.0 Proposed application

Our proposed application is working on the client side. The application has two tasks, encrypt and decrypt data with same key. The application is used to encrypt data in each item on the application level for each SQL statement such as insert or update, and in same time decrypt data on application level for each SQL select statement. The functions that use to encrypt and decrypt data use the same Key. The key that used in each encryption step is a variable length based on the length of the plaintext. Therefore, the encryption function is change the plaintext to ASCII code for each character of the plaintext and add the key to each character to produce the cipher text, for example, assume that the database item (Name) carry the data such as Arafat. Therefore, the length of the plaintext (Arafat) is the key, K=6. And the plaintext (Arafat) changes to the ASCII code for each character. And then add the K to each character that converted to ASCII code to produce the cipher text in numeric mode. Also, the encrypted function will change the numeric mode to alphabetic mode to give more complexity to the cipher text and then send it to the database. The following figure.1 explains the components of the proposed mechanism.

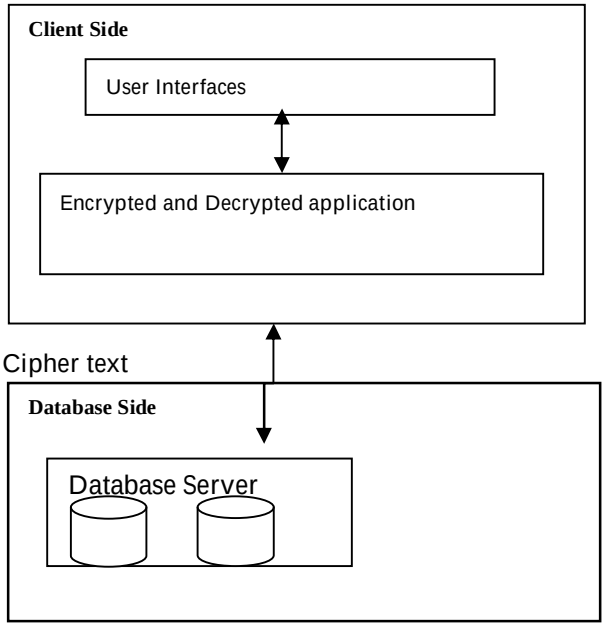

Figure.1: Proposal Application

The proposed application that encrypted and decrypted data on the client side has to functions to encrypted and decrypted data.

10.1 Encryption and Decryption Functions

The encryption and decryption functions consider the heart of the proposed application. They coded and decoded the plaintext on the client side. The core point of this application is, it can generate the key automatically from the plaintext. The key (K) is the length of the plaintext (PT). Therefore, it is not fixing, it is variable owing to, based on the length of the plaintext. The function is changing each character of the PT to the ASCII code in numerical mode, and then adds the key to produce the cipher text (CT). Also, change the CT from the numerical mode to the alphabetic mode, to give more complexity to the CT, this accruing in the encryption function. In decryption function, also the key is generating automatically, based on the length of the cipher text. The decryption function fetches the cipher text from the database and decrypted it on the client side. It generates the key from the length of the cipher text and the same other steps, which executed in the encryption function will executed again in the decryption function. The function is changing each character of the CT to the ASCII code in numerical mode, and then subtracts the key from the CT. next step; change the generated numerical data to the alphabetical data.

10.2 Strengths and Weaknesses

The proposed application has much strength that makes it a good method to encrypt data between client and server on the client side, these strengths can mention as follow:

1- Difficult to compromise or revealed a key easily, due to it is variable not fixing.
2- The authorized entities like a DBA cannot tamper or meddle the data outside of this application, even with full authorize.
3- The man in the middle cannot compromise easily the plaintext.

The weaknesses of this proposed application is suitable only for the attributes that can classified as sensitive attributes such as password, entity card and so on. It is not suitable for other attributes that have represent data as objects like picture, video.

11.0 Conclusion

The data that is traveled from the client to the database server without encryption will face many threats through outsider and insider attacks. This paper proposed an application, which encrypt and decrypt the database column on the client side. It encrypts data in the database column with variable length key. This key is unique for each data item even in same item; the key is not fixing, it based on the length of the plaintext which sent to the database item. The data will sent and saved to the database server as encrypted form. The advantages of this novel notion are: first, nobody can decrypt the data item outside of the application easily, due to the variable length of key. Second, the data will sent to the database in encrypted form, so the man in the middle cannot compromise intercept data easily. Third, the authorized entity on the server database such as DBA cannot tamper the data outside of this application.

12.0 References

[1] I. Summerville. Software Engineering. Addison-Wesley, 6th Edition, 2001.
[2] G. Davida, D.Wells, and J. Kam. A database encryption system with sub keys. ACM Transactions on Database Systems 6(2), 1981.
[3] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In Proc. of the 28th Int'l Conference on Very Large Databases, Hong Kong, China, August 2002.
[4] T. Dierks and C. Allen. The TSL protocol. Internet Draft, Nov., 1997.
[5]R.Agrawal,J.Kiernan,R.Srikant,andY.Xu.An XPathbased preference language for P3P. In Proc. of the 12th Int'l World Wide Web
Conference, Budapest, Hungary, May 2003.
[6] D. E. Denning. Cryptography and Data Security. Addi-son-Wesley Publishing Company, Inc., 1982.
[7] J. He and M. Wang. Encryption in relational database management systems. In Proc. Fourteenth Annual IFIP WG11.3 Working Conference on Database Security (DBSec'00),Schoorl, The Netherlands, 2000.
[8] P. Karlton, A. Freier, and P. Kocher. The SSL protocol v3.0.Internet Draft, Nov., 1996.
[9] D. Gupta, P. Jalote, and G. Barua. A formal framework for on-line software version change. IEEE Transactions on Software Engineering, 22(2):120–131, 1996.
[10] N. R. Adam and J. C. Wortman. Security-control methods for statistical databases.ACM Computing Surveys, 21(4):515– 556, Dec. 1989.
[11] R. Agrawal and J. Kiernan. Watermarking relational databases. In 28th Int'l Conference onVery Large Databases, Hong Kong, China, August 2002.
[12] T. F. Lunt. A survey of intrusion detection techniques. Computer & Security, 12(4), 1993