Database Security and the Role of Cipher

Anupama Chowdhary
Principal, Keen College
Bikaner (Rajasthan)
India

Abstract **–** In this digital world of e-commerce data repositories is an attraction for attackers. Generally, conventional database security systems have some lose points that can be used by attackers to penetrate the database. Database security could be enhanced by encryption algorithm or ciphers to give protection to the database from various threat or hackers who target to get confidential information. This paper analyse various ciphers used for encrypting data in the database.

Index terms **–** Cipher, encryption, decryption, symmetric, asymmetric, cryptography.

I. INTRODUCTION

Databases are the repositories where large amount of information is stored. Generally databases store sensitive data, such as credit card numbers, bank details, medical records, government top secret information and many more. Therefore this sensitive data stored in databases should be secure and protected. The objective of database security is to prevent undesired modification of data and information disclosure moreover ensuring the availability of the necessary service.

Almost all organizations are reshaping their business as a result of digital transformation such as cloud computing, mobiles, digital payments etc. and this digital transformation is reliant on data. Thales 2018 Data Threat Report says we're now at the point where we have to admit that data breaches are the new reality, with over third of organizations suffering a breach in the past year. In this increasingly data-driven world it is therefore hugely important to take steps to protect that data wherever it is created, shared or stored [1].

To protect data in databases unambiguous, consistent and multi-layered security policies must be followed. The mechanisms opt to provide security could be:

- Security features of the operating system such as authentication, logging.
- Firewalls
- Access control

These mechanisms offer protection against an intruder, but there may be an internal threat such as corrupted employees with access to the data. Furthermore probable security leaks could make feasible for someone to overcome these mechanisms.

To protect data from a malicious user such as an intruder or a corrupted employee can be achieved using database encryption. Encryption is the process of transforming plain text information using an encryption algorithm (called cipher) to make it unreadable to anyone except those have a key. With the help of cipher the data is stored in encrypted form in the database.

II. ENCRYPTION

Encryption is the method by which plaintext or any other type of data is converted from a readable form to an encoded version that can only be decoded by another entity if they have access to a decryption key [2].

For data at rest encryption can provide strong security, but developers have to consider many factors while developing a database encryption strategy. Even if the attackers somehow manage to bypass the

access control mechanism, encrypted data stored in a database can avoid their disclosure. Now, only few cryptographic keys require security in place of the whole database. But here ascend another factor; encrypting and decrypting data items can greatly degrade the performance of the database system. There must be a balance between the aspiration for outstanding performance and the requirement for security. A compromise solution must be found between performance and security, by encrypting only sensitive data in tables or column in the database [3]. So the there are some questions to be answered by the developers.

- Where sensitive data resides in the organization and classify which data to encrypt?
- Where encryption should be performed; in the storage, in the database or in the application?
- What should be the cipher and mode of operation?
- Who should have access to the encryption keys?
- How to minimize the impact of database encryption on performance?
- Train users to use encryption appropriately.

Denning [4, 5] provide database security through cryptographic means. He presents a comparative study on implementing encryption at various database levels i.e table, attribute and field level. He further tries to solve database integrity problem through cryptographic checksum.

Depending on our security and performance requirements developer have to decide where the encryption will take place, as soon as it is assessed the data to be protected. In general encryption could be implemented at any of the three levels; storage, database or application.

Storage level

Here data is encrypted in the storage subsystem therefore it is well suited for encrypting entire files and directories and protecting data at rest i.e. data is encrypted at the storage subsystem, either at the file level (NAS/DAS) or at the block level SAN. The storage subsystem has no knowledge of its users and the database scheme. The choice of data to be encrypted is limited to file granularity which may lead to increased overhead due to unnecessary data encryption. This type of encryption is well suited for encrypting files, directories, storage blocks, and tape media. Copies of sensitive data in log or temporary files may remain unencrypted.

Database level

At the time of reading from and writing to the database, the data is encrypted /decrypted. Selective encryption is possible at database level and can be done at various granularities, such as tables, columns, rows. Encryption can even be related with some logical conditions. This type of deployment is usually done at the column level within a database table and, if coupled with database security and access controls, can prevent theft of critical data. Encrypted data generally forbids the use of index so the performance of DBMS may degrade. If specific encryption algorithms or mode of operation is used so that order is preserved then the problem could be reduced. Many researchers have proposed different schemes of encryption at this level such as using Chinese remainder theorem, Newton's interpolating polynomials, RSA public-key scheme (at row and column), SPDE scheme (at each cell) [6, 7, 8, 9].

Application level

Here deployment encryption/decryption process is performed at the applications that generate the data. Encryption is performed within the application that introduces the data into the system, the data is sent encrypted, thus naturally stored and retrieved encrypted [10, 11, 12], to be finally decrypted within the application. In this method the encryption keys never have to leave the application side and thus are separated from the encrypted data stored in the database. Depending on the encryption granularity, the application may have to retrieve a larger set of data than the one granted to the actual user, thus opening a security breach. In this method too indexes are useless so there are performance overheads moreover encrypted data forbids the use of some advanced database functionalities such as stored procedures and triggers.

In storage and database level data is decrypted on the database server at runtime so keys are either transmitted to the server or are stored at server side. If some intruder takes over the administrator's identity the whole system will be at high risk.

III. ENCRYPTION MODULE DEPLOYMENT

There are three basic deployment methods namely API, plug-in, and TDE (Transparent Data Encryption).

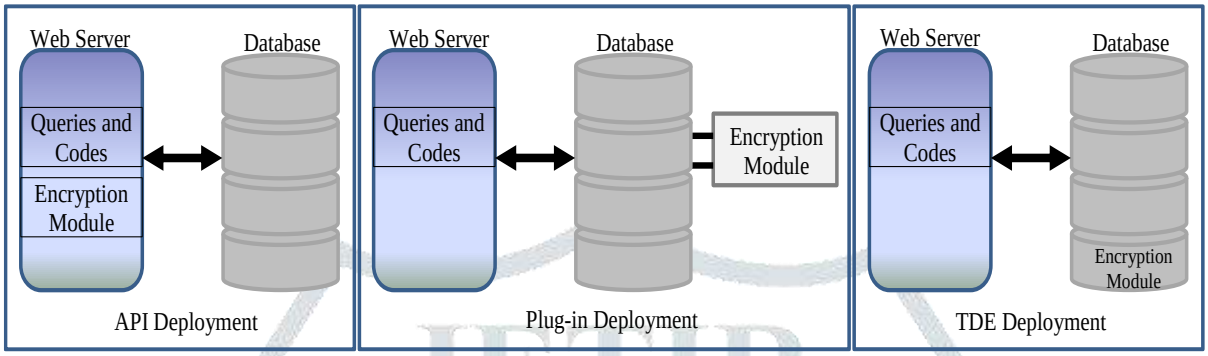

Figure 1 Encryption module Deployment

API Method

This method requires the engineer to use a provided code change function (encryption agent) to edit relevant parts in the web server in order to apply the encryption. API method is applicable regardless of the database product types such as Oracle, MSSQL etc. and does not impose any additional burden on the DBMS. The encryption process could be time-consuming when there are large volumes of data. This is due to the fact that the encryption process happens before the data enter the database.

Plug-in Method

In this method an encryption package (encryption module) is attached onto the DBMS. This encryption package works independently of the application and requires less modification to the query and code. It is easily applicable to both commercial DBMS and open source databases so is one of the most commonly used encryption method. The Plug-In method usually allows for index column-level encryption, access control, and auditing.

TDE Method

This encryption method requires the installation of an encryption/decryption engine directly into the database engine. This encryption method occurs at the lowest possible system level and requires no modification of the source code of the database environment or application. This means as actions are no longer required on the web server, administrators can easily install and manage the encryption engine in the database. But this deployment is not so secure.

IV. SYMMETRIC AND ASYMMETRIC DATABASE ENCRYPTION

Symmetric key cryptography is the most commonly used technique to encrypt data in the storage or database. The main features of this technique are:

- Ciphers use the same key for encryption and decryption of the data.
- Types of symmetric ciphers: block ciphers and stream ciphers.
- Stream ciphers are fast but require unique keys.
- Minimum key length for all symmetric key ciphers is 128 bits.
- A block cipher transforms a fixed length block of plaintext data into a block of cipher text data of the same length.

- Well known symmetric ciphers are Data Encryption Standard (DES), Advanced Encryption Standard (AES) and Blowfish.

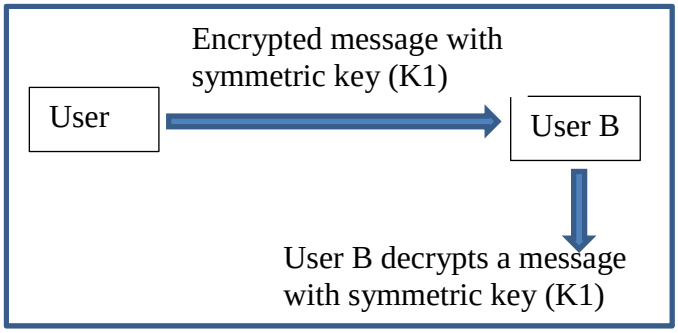

Figure 2 Symmetric key cryptography

Public key or asymmetric key encryption is also an alternative method for encrypting databases. The main features of this technique are:

- Uses two different but mathematically linked keys, one public and one private.
- The public key can be shared with everyone, whereas the private key must be kept secret.
- The RSA (Rivest Shamir Adleman) encryption algorithm is the most widely used public key algorithm.

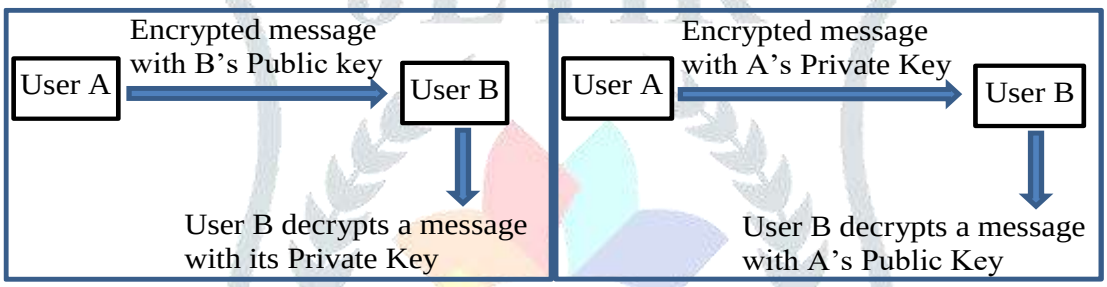

Figure 3 Two methods of Asymmetric key cryptography

V. CIPHERS FOR ENCRYPTION

The security of the encrypted data depends on the encryption algorithm, the encryption key size and its protection. Various ciphers could be chosen for encrypting data in databases; most popular ones are DES, AES, Blowfish and RSA. However REA

Data Encryption Standard (DES)

US Federal Government adopted IBM's Lucifier algorithm and gave it the new name DES in 1976. DES is a 6 Step process

1. 64 bit plain text is input into the initial permutation (IP) function.
2. Initial permutation will be on plain text.
3. Initial permutation splits the permuted block into two parts.
   - Left Plain Text (LPT) (32-bit)
   - Right Plain Text (RPT) (32-bit)
4. Each LPT and RPT passes through the 16 round encryption process. Each round has five steps.
   1. Key transformation
   2. Expansion permutation
   3. S-Box Substitution
   4. P-Box permutation
   5. XOR and swapping
5. Lastly, adding the LPT and RPT, it is send to the final permutation (FP) process.
6. The output of FP is 64 bit cipher text.

This DES algorithm work on 56-bit key and is a weak algorithm. So its variations double DES (with two keys) and triple DES (with two and three keys) is in use.

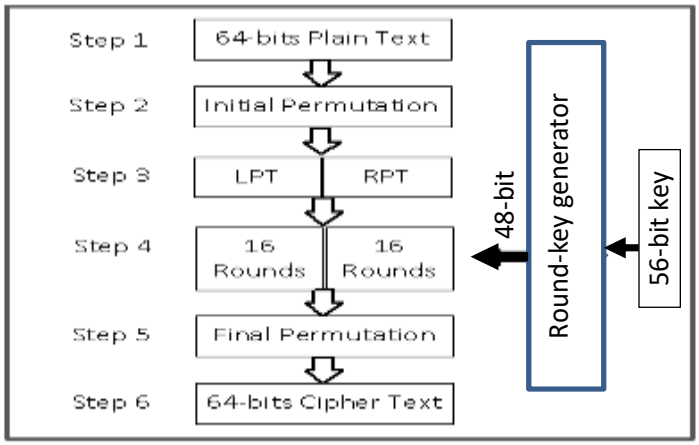

Figure 4 Six Step Process of DES

Advanced Encryption Standard (AES)

DES was working on 56-bit key and 64-bit block so it was not considered safe. In October 2000, Rijindeal was selected for AES. AES operation involves a complete byte while DES individual bit. The working process of AES is as follows:

1. **One-time initialization processes**
   - Expand the 16-byte key to get the actual key block to be used
   - Do one time initialization of the 16-byte plain text block called as state.
   - XOR the state with the key block
2. **Procedure of each round**
   - Apply S-box to each of the plain text bytes.
   - Rotate row K of the plain text block (i.e. state) by K bytes.
   - Perform a mix column operation
   - XOR the state with the key block.

Two versions of AES are used

- 128-bit plain text block with 128-bit key
- 128-bit plane text block with 256-bit key

Modes of operation

Block cipher encrypt fixed size blocks so they need some way to encrypt or decrypt arbitrary amounts of data in practise ANSI X3.106-1983 modes of use (now FIPS 81) defines 5 possible modes

- Electronic Codebook Book (ECB)
- Cipher Block Chaining (CBC)
- Cipher FeedBack (CFB)
- Output FeedBack (OFB)
- Counter (CTR)

Even having adopted strong algorithms, such as AES, the cipher text could still disclose plain text information if an inappropriate mode is chosen.

Blowfish

Blowfish is a symmetric-key block cipher, it is one of the fastest block ciphers in general use, except when changing keys. Blowfish has 16 rounds. Each round consists of a key dependent permutation, and a key- and data-dependent substitution. All operations are XORs and additions on 32-bit words [13]. The only additional operations are four indexed array data lookups per round.

Sub keys: Blowfish uses 18 32-bit sub-keys. These keys must be pre-computed before any data encryption or decryption.

Encryption and Decryption: Blowfish has 16 rounds. The input is a 64-bit data divided into two 32-bit halves: L, R. These L and R passes through a 16 round process to obtain cipher text.

Each new key requires pre-processing equivalent to encrypting about 4 kilobytes of text, which is very slow compared to other block ciphers.

Reverse Encryption Algorithm (REA)

Reverse Encryption Algorithm is a symmetric stream cipher. The keys are concatenated to the text in the encryption process and removed from the text in the decryption process. Mathematical Divide operation is performed on the text data by 4 in the encryption process and multiply operations on the text by 4 has been done in the decryption process. Divide by 4 operations is performed on the text to narrow the range domain of the ASCII code [14].

Algorithm [15]**:**

1. Input the text and the key.
2. Add the key to the text.
3. Convert the previous text to ASCII code.
4. Convert the previous ASCII code to binary data.
5. Find out One's complement of the previous binary data.
6. Gather each 8 bits from the previous binary data and obtain the Decimal value from it.
7. Divide the previous Decimal value by 4.
8. Obtain the ASCII code of the previous result divide and put it as one character.
9. Obtain the remainder of the previous divide and put it as a second character.
10. Return encrypted text.

This cipher is faster than other equivalents such as DES, AES and blowfish for encryption and decryption but is less secure compared to them [16].

RSA (Rivest Shamir Adleman)

The prime numbers are used in RSA algorithm. This algorithm is based on the mathematical fact that it is easy to multiply big prime numbers but the factorization of their product is very difficult. Private and public key in RSA is based on very large prime numbers. These prime numbers are of 100 or more digits. RSA's algorithm is very simple but the real challenge in RSA is to select and build public and private key.

Algorithm:

1. Choose two large prime numbers P and Q
2. Calculate N= P X Q
3. Select the public key E such that it is not a factor of (P-1) and (Q-1)
4. Select the private key D such heat the following equation is true
   (D X E) mod (P-1) X (Q-1) =1
   /* D- Decryption key, E- Encryption Key */
5. For encryption, calculate the cipher text CT from the plain text PT as follows:
   $CT = PT^E \bmod N$
6. Send CT as the cipher text to the receiver
7. For decryption, calculate the plain text PT from the cipher text CT as follows:
   $PT = CT^D \bmod N$

VI. KEY MANAGEMENT

Cipher chosen for securing data in a database plays an important role but encryption algorithms are as secure as the management of the keys. Therefore one needs to consider carefully several aspects regarding management of keys.

1. How many encryption keys are needed?
2. How access to the keys will be restricted to authorized users only?
3. How often should the keys change?

Keeping the number of the keys low makes implementation simpler but if one key is stolen more data is vulnerable. Moreover, if more systems can access the keys there is higher probability for a key to leak. A simple approach is to store the keys in a restricted database table or a file. This approach has the drawback that all database administrators can access the keys and decrypt the sensitive data without leaving any trace. Another approach is to use specialized tamper-resistant cryptographic chipsets called hardware security module (HSM). Generally the HSM stores a master key used to encrypt the keys which are stored in the database server [17]. Now database administrators cannot directly access the encryption keys.

A security server maybe may be used along with a HSM, which manages security related tasks such as users, roles, privileges, encryption policies and encryption keys. The database administrator and the security administrator must co-operate in order to decrypt the data.

VII. CONCLUSION

The security of sensitive data is a critical issue; if data are encrypted before storage in the database the risk from security leaks can be reduced and the whole database security issue can be reduced down to the protection of few cryptographic keys. The selection of cipher for encryption plays an important role. If an attacker somehow obtains the encrypted data he could not get any information out of it. The variation of DES i.e. triple DES and AES are largely used for encrypting databases but as we have disused the mode of operation should be chosen appropriately. Blowfish is another cipher in use it is although fast but for a new key require large pre-computations. REA is a new algorithm although it is fast but is considered less secure. For public key encryption model RSA is largely used.

Generally, encrypted databases are deploy on commodity DBMS. If designers of encrypted databases do not consider logs, caches, and data structures in their threat model the information may leak due to past queries [18]. Based on the workload there is a trend in database design towards adaptively changing the structure of the database [19, 20, 21]. The snapshots of such databases leak even more information about past queries. So, the designers have to look for ways in which information about past queries is stored in a DBMS so as to reduce the leakage.

REFERENCES

[1] https://www.thalesgroup.com/en/worldwide/security/press-release/2018-thales-data-threat-report-trends-encryption-and-data-security

[2] M. Rouse, "SearchSecurity," TechTarget, [Online]. Available: http://searchsecurity.techtarget.com/definition/encryption.

[3] Zongkai Yang, Samba Sesay, Jingwen Chen and Du Xu, "A Secure Database Encryption Scheme", American Journal of Applied Sciences 1 (4): 327-331, 2004 ISSN 1546-9239.

[4] Denning, D.E., Field Encryption and Authentication. Proc.of CRYPTO 8.9, Plenum Press, 1983.

[5] Denning, D. E., Cryptographic checksums for multilevel data security. Proc. Of Symp. on Security and Privacy. IEEE Computer Society, pp: 52-61, 1984.

[6] Davida GI, Wells DL, Kam JB, A Database Encryption System with subkeys. ACM Trans. Database Syst. 6, 312-328, 1981.

[7] Buehrer D, Chang C, A cryptographic mechanism for sharing databases. The International Conference on Information & Systems. Hangzhou, China, pp. 1039- 1045, 1991.

[8] Chang C, Chan CW, A Database Record Encryption Scheme Using RSA Public Key Cryptosystem and Its Master Keys. The international conference on Computer networks and mobile computing, 2003.

[9] Shmueli E, Waisenberg R, Elovici Y, Gudes E, Designing secure indexes for encrypted databases. Proceedings of Data and Applications Security, 19th Annual IFIP WG 11.3 Working Conference, USA, 2005.

[10] Hacigümüs H., Iyer B., Li C., Mehrotra S., Providing Database as a Service, International Conference on Data Engineering (ICDE), pp. 29-39, 2002.

[11] Ernesto Damiani, S. De Capitani Vimercati, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati, Balancing confidentiality and efficiency in untrusted relational dbms, Proceedings of the 10th ACM conference on Computer and communications security, ACM, pp. 93-102, 2003.
[12] Luc Bouganim and Philippe Pucheral, Chip-secured data access: confidential data on untrusted servers, Proceedings of the 28th international conference on Very Large Data Bases, pp. 131-142, 2002.
[13] B.Schneier, The Blowfish Encryption Algorithm. July 22, 2009
[14] Ayman Mousa, Osama S.Faragallah, EL-Rabaie, S., NigmE.M. (March 2013), "Security Analysis of Reverse Encryption Alogrithm for Databases", International Journal of Computer Applications(0975-8887),volume 66-No.14,pp.19-26.
[15] Priti V. Bhagat, Kaustubh S. Satpute, Vikas R. Palekar "Reverse Encryption Algorithm: A Technique for Encryption & Decryption", International Journal of Latest Trends in Engineering and Technology (IJLTET), Vol. 2 Issue 1 January 2013. ISSN: 2278-621X.
[16] Ayman Mousa, Osama S. Faragallah, S. EL-Rabaie, E. M. Nigm "Security Analysis of Reverse Encryption Algorithm for Databases", International Journal of Computer Applications (0975 – 8887), Volume 66– No.14, March 2013.
[17] Luc Bouganim, Yanli Guo, Database Encryption, 2009.
[18] Paul Grubbs, Thomas Ristenpart, and Vitaly Shmatikov, "Why Your Encrypted Database Is Not Secure", HotOS '17, May 08-10, 2017, Whistler, BC, Canada.
[19] Surajit Chaudhuri and Vivek Narasayya. Self-tuning database systems: A decade of progress. In VLDB, 2007.
[20] Stratos Idreos, Stefan Manegold, Harumi Kuno, and Goetz Graefe. Merging what's cracked, cracking what's merged: Adaptive indexing in main-memory column-stores. In VLDB, 2011.
[21] Alon Shalita, Brian Karrer, Igor Kabiljo, Arun Sharma, Alessandro Presta, Aaron Adcock, Herald Kllapi, and Michael Stumm. Social Hash: An assignment framework for optimizing distributed systems operations on social networks. In NSDI, 2016.