Providing Security for Database at Rest with Fine Grained Encryption
Kavitha N, Nisha K K
Associate Professor, RIT, Kottayam, Kerala, India, Assistant Professor, GEC, Idukki, Kerala, India

Abstract—Database is an important asset for any organization and there should be strong security measures to preserve the confidentiality of sensitive data. There are different levels of granularity for encryption. This paper focus on minimizing the overhead of encryption and decryption during query processing. These include encrypting the sensitive data in the attribute level at the time of insertion and update. Here we use the strong encryption algorithm, DES, which is the most widely used symmetric key block cipher in the world. The strength of the algorithm relies on its size of the key and its Feistel Network structure. In this paper we give the algorithms for insert, update and select query processing.

Index Terms—Database Encryption, Database Security, Encrypted Database, Security

I. INTRODUCTION
In most organizations, databases hold a critical concentration of sensitive information, and as a result, databases are vulnerable. Database system should be protected from any attacks. The Defense Information Systems Agency of the US Department of Defense (2004), in its Database Security Technical Implementation Guide, states that database security should provide controlled, protected access to the contents of a database as well as preserve the integrity, consistency, and overall quality of the data. Authentication and access control mechanisms are the only means used for the security of current relational database management systems (RDBMS). More sophisticated administration mechanisms can be revised such as joint administration, by which several subjects are jointly responsible for authorization administration [1]. A number of extensions have been provided to protect data at rest, in storage on database systems. Extra steps and precautions should be taken to carefully control access this data.

Today, enhancing the security of database is becoming one of the most urgent tasks in database research and industry. Generally, database security methods could be divided into four layers [1]: physical security, operating system security, DBMS security and database encryption. These layers protect database in different aspects. But only first three layers are inadequate to protect the confidential data in database satisfactorily, because data is still stored in readable form. So without database encryption, it makes no sense to guarantee that the sensitive information in plaintext will be protected against an untrustworthy user who has super-user power, such as a database administrator (DBA). In that way, database server will be compromised as long as the DBA or his credentials are compromised. Besides, data is not always static; it can also be in move. Therefore two main issues both need to be considered: secure data storage and secure data transmission. However, it is difficult to control the disclosure of confidential data while it's being transmitted on the Internet in plaintext. Our focus is on secure data storage.

Database encryption is a technology that introduces an additional security layer to traditional database management system [2]. It prevents exposure of sensitive information even if the database server is compromised. Furthermore, database encryption can be employed to maintain the data integrity, ensuring that even a little modification made on the data can be detected. Database encryption technology meets the data confidentiality requirements and has become an indispensable aspect of enterprise database security.

The security of encrypted data depends on several factors like what algorithm is used, what is the key size and how was the algorithm implemented in the product. Besides the quality of the encryption technology, any database protection effort is only as secure as the key management strategy that supports it. Encryption technology that used stored key is highly vulnerable. Sensitive data stored on networked servers are at risk from attackers who only need to find one way inside the network to access this confidential information. Additionally, perimeter defenses like firewalls cannot protect stored sensitive data from the internal threat. This paper is organized as follows. In section II, we discuss the related works done by predecessors, and analyze the strengths and weaknesses of existing solutions. In section III, we characterize the possible attacks that may be conducted against confidential data. In section IV, we present our database encryption scheme and show how it answers the data confidentiality and efficiency questions. And in section V, we discuss the algorithms for query processing section VI; we make the conclusion of our work.

II. RELATED WORK
G. I. David presented a database encryption scheme with sub keys based on the Chinese Reminder theorem [3], which enables encryption at the tuple level and decryption at the attribute level. In other existing proposals for database security the whole data is encrypted on the client side before being stored in the database [4, 5, 6]. Sohail Imran devised a system for the security of Object Oriented Database Systems (OODBMS)[7]. Unlike traditional RDBMSs, secure
OODBMSs have certain characteristics that make them unique. The object-oriented database model permits the classification of an object's sensitivity through the use of class and instance. When an instance of a class is created, the object can automatically inherit the level of languages in order to address a large variety of application requirements.

Controlling access to database tables or columns is frequently required and can be enacted by simply granting privileges to one of these objects. Restricting access to data contained in individual records (rows) requires additional steps. For instance, a student should only be able to view or modify the row or rows of data that correspond specifically to him or her. However, implementation of row level security cannot be done in the same manner as access control is applied to database objects such as tables. This is because the selection of a row is based on the evaluation of specific data values. Therefore, a common way to implement row level security is through the use of SQL Views. A View can be constructed that executes a select statement which returns specified rows of data evaluated against a specific value[8]

Yu Yonghong proposed an automatic attributes detection partition method and a new security model which partition data in unencrypted form to distributed secure database servers. The theoretical analysis and experimental results show that our new method is feasible and provide efficient privacy protection and efficient query execution, and support horizontal fragmentation and semantic attribute decomposition [9]. The problems we find with these proposals are that the whole or part of the database has to be encrypted and for all queries decryption have to be done, thereby reducing the efficiency.

III. POSSIBLE ATTACKS

Security breaches are an increasing phenomenon. As more and more databases are made accessible via the Internet and web-based applications, their exposure to security threats will rise. The objective is to reduce susceptibility to these threats. Considering the attackers, we categorize them into three types: intruder, insider, and administrator. Intruder is an external person who infiltrates a database server to steal or tamper with data information. Insider is an authorized user in database system, but he conducts some malicious work. And administrators include database administrator (DBA) and system administrator (SA), who have absolute right on database system. If DBA is not trustworthy, the security of database can get little guarantee. According to the attacks database faces, encryption must be done at a trusted environment, moreover encryption done by application itself also poses some challenges, if data is encrypted at the application, and then all applications that access the encrypted data must be changed to support the encryption/decryption model.

IV. DATABASE ENCRYPTION SCHEME

Figure 1. Shows the classification of encryption levels [10]. Storage-level encryption, Database-level encryption and Application Based on the granularity, the application level encryption can be classified into four categories.
A. Database-level encryption
B. Table-level encryption
C. Tuple-level encryption and
D. Attribute-level encryption. Attribute-level encryption is the fine grained encryption scheme. Our focus is on Attribute-level encryption in application level.

We group the attributes of each table in the database into two groups. 1. Sensitive attributes and insensitive attributes. In the database some tables not have sensitive attributes. We have to identify sensitive attributes and encrypt them only. In relation R(A1,A2,...,Ai,Ai+1......,An), it is presumed that A1,A2,...Ai are attributes without need of encryption (insensitive attributes), while Ai+1......,An need to be encrypted (sensitive attributes).

For each relation R, define Rs={ Ai+1......,An }, the set of sensitive attributes in relation R. Then the tuple stored is (A1,A2,...,e(Ai),e(Ai+1)......,e(An)) , where e(Ai) is the encrypted attribute value. The encryption algorithm is DES, which is a strong cipher. The strength relies on the size of the key and feistel network structure of the algorithm Figure 2 shows the round i of sixteen rounds in DES algorithm.

\[ L_{i} = R_{i-1} \]
\[ R_{i} = L_{i} \oplus F(R_{i-1}, K_{i}) \]

**Key Management**

Key management refers to the way cryptographic keys are generated and managed throughout their life. Because cryptography is based on keys that encrypt and decrypt data, the database protection solution is only as good as the protection of the keys. The location of encryption keys and their access restrictions are thus particularly important. In our system the user selects a keyword and is converted into an initial 64 bit key which will be again converted in to 54 bit key. Figure 3 gives the outline of database encryption and query operation, where encryption and decryption are done only on sensitive attributes.

**Database encryption process:**

1. The user submits a plain-text database to be encrypted and an encryption key to an encryption middleware.
2. The encryption middleware extracts database encryption parameters from encryption metadata.
3. The encryption middleware encrypts plain-text data according to the database encryption parameters and the user key to generate a cipher text data set.
4. The cipher text data are submitted to a remote database server.
5. The database server stores the cipher text in data files.

**Database query operation process:**

6. The user submits the key and database query condition to database query middleware.
(7) The query middleware extracts database encryption parameters from encryption metadata.

(8) According to the encryption metadata, the query middleware transforms the database query condition into an equivalent one corresponding to the cipher text database.

(9) The database server executes SQL statements to extract the cipher text data set on reception of the query condition.

(10) The cipher text data set is returned to the client.

(11) The query middleware obtains the cipher text data set and performs decryption according to the key and the decryption parameters provided by the user to generate a plain-text data set.

(12) The plain-text data set is returned to the end user.

---

**Fig 1. Three options of database encryption level**

**Fig 2. Single round in DES**
V. ALGORITHMS

Storage Queries:
1. Algorithm to insert
   Input : Attributes = (A1,A2,…,Ai,Ai+1……,An) attribute values (a1,a2,…,ai,ai+1……,an) Sensitive attribute names S=(Ai+1…….An) No of Sensitive attributes =scount
   For j=1 to scount
   { 
   bj=e(value(S[j])) ---encrypted value of attribute S[j] 
   }
   Output : The tuple to be stored in the database ((a1,a2,…,ai,b1,b2……,bscount)
   
   Query:
   Insert into R values ((a1,a2,…,ai,b1,b2……,bscount)

2. Algorithm to update
   Input : S=(Ai+1…….An)
   Update query : update R set Ai=val where Condition
   if any attributes in condition is in S and operator is ‘=’
   encrypt the corresponding values
   else decrypt the attribute values stored and compare.
   If Ai is in S
   newval=e(val)
   else
   newval=val
   output: update R set Ai=newval where Condition

Retrieval Query:
Algorithm for select query:
Input : S=(Ai+1…….An)
Select <attribute list/> from R where condition

1. If the condition contains any attribute in S
   Encrypt the attribute value and execute the query

2. If attribute list and S has common attributes
   Decrypt the common attribute values in the result set
   and return values

3. If attribute list contains aggregate function of an attribute in S
   Decrypt all attribute values in R and execute the query
   Output: the attribute list values

VI. CONCLUSION

To provide secure database service, we need to integrate many security techniques, such as data access control, network transformation control, database queries and privacy protection. This paper mainly focuses on privacy protection in secure database service architecture, and proposes a new security model in which only sensitive attributes are encrypted. Frequently executed queries are not expected to contain sensitive attributes and hence decryption is not needed. The efficiency of the DBMS is not much affected by encryption of sensitive attributes. One problem we find with our proposal is that, in the case of queries with aggregate functions on any one of the sensitive attributes all values of that attribute have to be decrypted. The future work may pay more attention to improving the query efficiency.

ACKNOWLEDGMENT

The authors gratefully acknowledge the insightful comments of our colleagues and peer reviewers which helped to improve the quality of the paper significantly.
REFERENCES


AUTHOR BIOGRAPHY

Kavitha N, M, Tech in Computer Science from NIT, Trichy, ISTE life time member.

Nisha K K, M, Tech in Computer Science from IIT, Madras, ISTE life time member.