Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

# Index seek technique for Querying Encrypted Databases

**Sahab Dheyaa Mohammed [a] , Abdul Monem S. Rahma [b]**

a University of Information Technology and Communications, Baghdad, Iraq , Email: sahab7dia@yahoo.com

b Department of Computer Science, University of Technology, Baghdad, Iraq, Email: Monem.rahma@yahoo.com

ARTICLE INFO

*Article history:*
Received: 16 /06/2019
Rrevised form: 30 /06/2019
Accepted : 09 /07/2019
Available online: 01 /09/2019

*Keywords:*

Encryption database, Encoding data, Indexing techniques, Index field generation, query processing,

ABSTRACT

Encryption is necessary to maintain data confidentiality and protect against outside attackers and internal administrators in the remote database (DB). However, the use of encryption makes it difficult to retrieve data from the remote database .where in traditional methods, is retrieved all data from the DB then, data are decrypted to obtain the specific data requested by the user. However, this process is time-consuming and does not maintain enough data security. In this paper, a technique was proposed to solve traditional method defects in encrypted DB systems. The proposed technique builds queries within the proxy server and retrieves only specified encrypted data from the remote DB without the requirement of retrieved all encryption data, then decrypted at the proxy site. Results showed that the proposed method took less time than conventional methods while maintaining data confidentiality.

## 1 . Introduction

Data, which must be kept confidential and stored securely, are a crucial factor in the effective operation of institutions. Data are usually stored in databases (DBs) that meet security requirements. However, administrators are faced with the challenge of meeting security requirements, especially in building strategies to protect data from being stolen by hackers. Encryption offers a large dimension in DB security, in which users are not allowed to access data illegally and steal DB contents when stored in various media, such as CDs, tapes and disks.

Encryption only assists in the implementation of the security system because SQL queries cannot be executed directly on an encrypted DB. Encrypted data requires decryption before an SQL query can be made. However, this process takes a long time. Many mechanisms have been applied to resolve this issue, but studies on overcoming this problem are limited [1].

The problem is magnified when additional information is stored and concentrated on data retrieval. Users can typically send an accurate query to the remote unencrypted DB easily and retrieve the exact information they need. However, when all information is encrypted, users cannot extract specific data from the server. Thus, the entire or part of the DB is recovered and then decrypted; only then can users send a query via their computer [2]. The

Corresponding author Sahab D. Mohammed

Email addresses: *sahab7dia@Gmail.com*

Communicated by Dr. Mustafa Jawad Radif

proposed technique exhibited better searching performance and faster data retrieval compared with conventional techniques. In addition, this technique used the index mechanism to distinguish different records.

## 2. Related works

DB security is a problem that must be considered. Thus, encryption techniques could be used in DB management systems (DBMSs). Although encryption provides high security, problems, such as reduced system efficiency caused by encryption, still exist.

In [3] execute the query on encrypted data at the server site, without decrypting the data. But the Decryption operation is performed at the client site. in this work, Different querying models are built and construct different indexes to execute computations for different data types, such as range queries of numeric by building a bucket index, and complete fuzzy queries of characters efficiently by using the bloom filter as the encrypted index.

In [4], a new indexing technique was proposed to search for domain queries of numeric data. This technique was only useful in this type of research and not for literal data. Reference [5] proposed to divide the client's attribute set into a set of time intervals. Original and temporal intervals were matched on the client side. In the DB, encrypted tables were stored with separator data. Therefore, the original domain and query values were specified with corresponding interval values to obtain data efficiently.

In[6 ] discussed how database search queries perform using a homomorphic encryption way based on the ideas of Gahi's method.

In this work uses two parts. In the first part, Homomorphic Query can be used with the ring-based fully homomorphic encryption.

In the second part, we use the Homomorphic Query building a keyword search way in the smart grid.

Reference [7] suggested the creation of a B-tree on normal text values, wherein each B node was encrypted and stored in the unauthenticated DBMS. The main B-tree in the unauthenticated DBMS was the executed as a table with two attributes, namely, the node ID (automatically assigned by the system at insertion) and content of the encrypted node. This technique has advantages and disadvantages. The advantage is that the B-tree content is invisible to an unreliable DB service provider. The disadvantage is that this technique involves the processing of large data in client machines.

## 3. Types of Indexing Techniques for Encrypted DB

Reducing the load on the client's machines is a main objective of the query process because client computers have limited storage space that constrains its accounting ability [8].

Two incompatible requirements exist in indexing and computation of data. Firstly, indexing data must be attached to the data in an appropriate form to perform an effective query. Secondly, no relationship must exist between indexes and data to help predict and allow attacks that could threaten the protection offered by encryption [7].

Several techniques are available in the implementation of different types of queries in the server.

### 3.1 Index by encryption

The query is converted into an encrypted form and executed by the encryption operation for every value in the authentic query. In this technique, a distinction is made between values. However, on the basis of frequency analysis, predicting between plaintext and encrypted values can be possible by comparing the distributions of plaintext relations with conformable ciphered values [8] [9].

### 3.2 Bucket-based index

The field of search values is in a set of nonoverlapping parts, and the index contains the equivalence class (section) [9].

This technique eliminates pseudorecords by using group-based indexing, receiving encryption data and decrypting of the authorised user. Domain queries are too complex to calculate because conversion generally does not (and should not) maintain the order relationship of plain text data [9]. Hence, this technique shows the effectiveness of reducing the number of pseudogroups in the range/equality query result [10].

### 3.3 Index by hashing

Bucketing preserves the relation between two adjacent values by using a secure one-way hash function. This function accepts clear input values and returns identical index values. We obtain the same result but without the proximity relationship [7].

### 3.4 Auxiliary B+-tree

Using an encrypted version of the B+-tree to store plaintext values and preserve the ordering can address range queries [11].
A B+-tree of order *m* is a tree wherein each internal node contains up to *m* branches (children nodes), and thus store up to (*m - 1)* search key values. m is a popular branching factor [12].

The B+-tree is created from a set of nodes that serve as indicators of records. All the nodes can reach the same level in the tree. Thus, the height of the tree is always balanced.
All internal nodes, except the root, are between m and ⌈m/2⌉children, which store search key values and divide them among the children in the node of a search tree.

## 4. The architecture of the proposed environment

The proposed technique operated on a remote DB, wherein the enterprise as a proxy server and the clients were represented as users. The proxy server is the contact centre between the client and the remote DB. Figure (1) shows an overview of the environment of the proposed method.

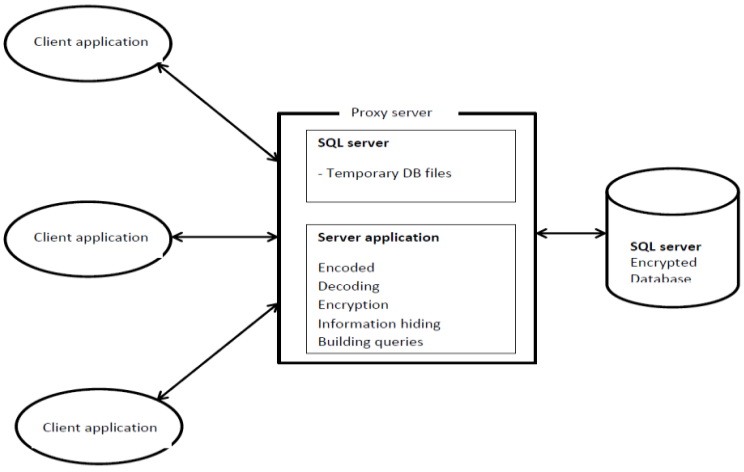


Figure (1) Scheme of the environment of the proposed method

Indirect communication is between the remote DB and clients. However, the proxy receives the records or queries from clients and performs a series of operations, such as encoding and encryption, then forwarded the records or queries to the DB at the remote server. Also, the proxy server performs the significant decryption and decoding operation on retrieved encryption records from the remote DB server and then returned the final result to the user. Encryption and decryption operations are only performed on the proxy server, thereby indicating that the proxy has an important role as the query parser. Hence, the proxy managed the communication between the DB applications and the encrypted remote DB.

### 4.1 Encoding and decoding operations

Before sending to the encryption algorithm, plaintext is encoded in binary code by using a encoding table, which is also known as the dictionary or substitution table. The encoding table contains words, symbols, letters, and numbers.
The table is constructed in two ways. The first way is to fill the table early with common words and symbols based on the analysis of real databases and their frequency. Each word, number, and symbol in the encoding table is called word set and is assigned into binary of two bytes (16 bits).
the second way, add the new words or symbols dynamically from the interface of the client application .when the user inserts a new record the application verifies from the remote database if the record is entered previously or not, also verifies if the keywords of the record entered previously in the encoding table or not, if keywords didn't enter previously, added at the end of list of the word set at encoding table.

When constructing the coding table, the frequency of the word within the databases was taken into account. The data that is more frequently in the databases are selected to be in the encoding table as a constant, the words that are rare inserts at the encoding table depend on when until their uses in the database.
Every byte represents a binary value (0…… 255) as an element polynomial in the finite field GF ($2^8$).
When the encoder obtains the required word, the encoder substitutes this word with a pair of bytes. Table (1) shows the table of encoding data.
The dictionary contains up to 65.536 pairs of bytes that same are stored dynamically or manually.
The purpose of creating a table coding is to use it in searches into an encrypted database. Where, the properties of the coding data can be used in the Building of queries into the proxy server to seeking on the required records [13].


Figure (2) Schematic of encryption technique

Then, a new technique in information hiding was suggested. This technique combined the characteristics of steganography and encryption methods. This technique enhanced the encryption method by hiding cipher records into the randomly matrix (256 × 256) [13].
The encrypted records are merged with matrix elements (cover file) before sending it to the remote DB.
This technique assists in the elimination of the repetitions between cipher text elements and the Detection of encrypted data by attackers is difficult when data were sent to the remote DB.
The embedding procedure started when encoded and encrypted records of data and block randomly matrix became available.
This procedure used a technique that embedded encrypted records in the rows of the random matrix that we selected randomly. To performing embedding operation, the encrypted records are substituted with elements of the row by excuses the function (1), the result was replaced with the same location of the element of the row into the cover file [13]. Figure (3) Flow chart illustrating the embedding process.

$$E(x) = (ax + b) \bmod m \qquad (1)$$

Where:
**x**: encrypted byte value of the record.
**m**: an irreducible polynomial of GF ($2^8$) , ($x^8 + x^4 + x^3 + x + 1$) .
**a**: element of cover-file (z) .
**b**: secret key.
**E(x)**: embedded element for sensitive data in the cover

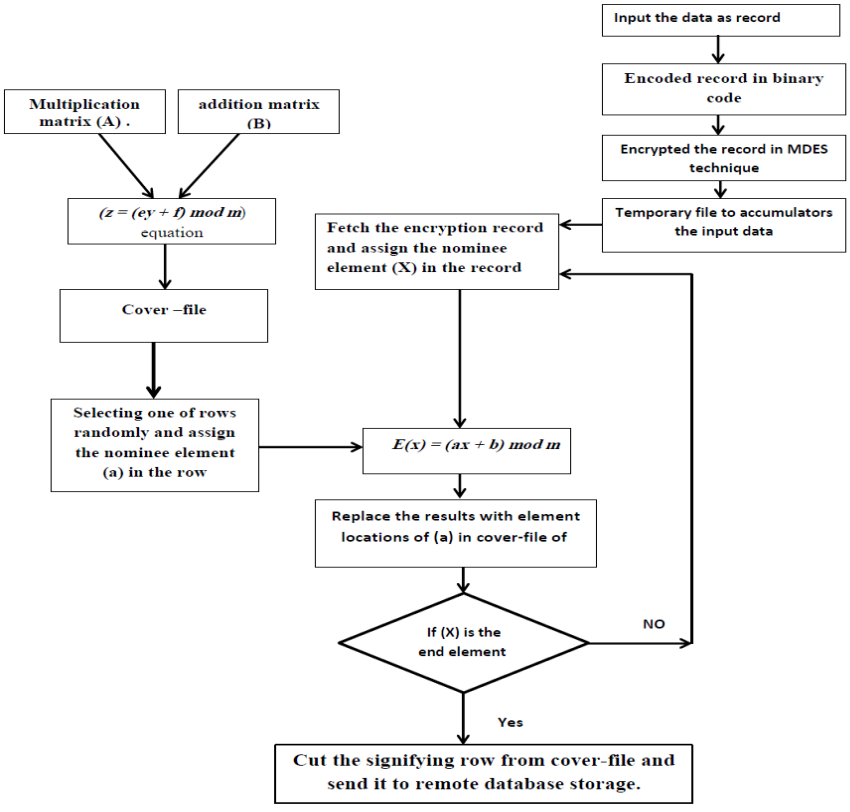


Figure (3) Flow chart illustrating the embedding process

## 5. Proposal work

This study suggested a method of retrieving data from an encrypted DB without extracting all or parts of the DB. However, certain records were retrieved, decrypted and sent to the used to preserve data confidentiality in an unreliable environment and improve search and update capabilities.
In this technique, data decryption was only performed at the proxy site. Therefore, a query was created to achieve consistency in the encryption data and assists in returning the exact set of requested data to the client. This rapid method of querying was used to retrieve the specific record from the encryption DB.

## 5.1 Querying by using index seek

This method used a rapid index-seeking mechanism to retrieve a specific record. The index method is usually difficult when the remote encryption DB when it contains an index field because the index values can be exposed to prediction by the external attacker when the index table has been detected. Hence, we used that technique when the security degree is low, as determined by the security policy. The index choice was dependent on attributes that were requested and amount of data retrieved from the DB.
Therefore, we used this new method instead of the previously suggested method because of its flexibility in selecting from the many attributes in all queries. In addition, this method was a rapid way to determine the record directly.
This method was based on the created index fields of the attribute (key) that was used in all queries. The index contents represented the equation results in each record. These index contents were unique values that make it impossible for attackers to predict the plain text. Hence, this new method was proposed to ensure that no repetition of index values exists.

### 5.1.2 Creation of indexing fields

Indexing fields of the attribute (key) field were created based on the combination of three element values by XOR operation in Equation (2). The first element represented the encoded value of the attribute (key) that was available in the dictionary table. The second element was the ID value of the record (ID value of the row in randomly matrix). The third element was the constant value, wherein each index had a constant value. Equation (2) is presented as follows:

$$y_i = x_i \oplus z_i \oplus c_i \quad \text{.....} \quad (2)$$

Where
$y_i$ : The new value of index field.
$x_i$ : The encoded value of the attribute (key).
$z_i$ : ID value of the record.
$c_i$ : Constant value
$i$: is a polynomial number of GF ( $2^8$) (0…….255)

The results represented the ending values of index fields in Figure (4) shows the index fields.

| Attribute Key | | | | | | | | | | | | ID | Indexes constant 120 | 70 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F0 | | F2 | F3 | | | | | | | | F255 | F256 | F257 | F258 | F259 |
| 3 | 78 | 98 | 140 | 170 | 2 | 66 | 93 | 68 | 42 | 103 | 105 | 106 | 16 | 31 | 20 |
| 56 | 6 | 211 | 83 | 1 | 50 | 229 | 41 | 21 | 43 | 22 | 24 | 101 | 204 | 8 | 250 |
| 29 | 34 | 67 | 16 | 29 | 7 | 19 | 17 | 244 | 69 | 28 | 26 | 89 | 71 | 38 | 76 |
| 238 | 123 | 221 | 19 | 39 | 48 | 31 | 82 | 54 | 40 | 84 | 25 | 151 | 134 | 27 | 32 |
| 121 | 54 | 21 | 18 | 55 | 59 | 222 | 9 | 89 | 33 | 215 | 43 | 289 | 98 | 35 | 11 |
| 11 | 21 | 117 | 30 | 11 | 57 | 33 | 59 | 15 | 61 | 109 | 57 | 104 | 19 | 276 | 90 |
| 251 | 242 | 15 | 10 | 12 | 20 | 4 | 71 | 68 | 61 | 29 | 106 | 65 | 277 | 2 | 100 |

Figure (4) show the elements of Index fields

Where F2 and F3 are the fields of attribute (key); F256 is a field of ID values of records that indicates the records' location in randomly matrix; and F257 and F258 are the index fields of attribute F2 and F3, respectively.

The algorithm of generate index fields

| **Input :** $x_i, z_i, c_i$ **fields**<br>**output: indexes fields** $y_i$ | |
|---|---|
| **Step1** | Fetch the encoded words symbols of attribute (key) $(x_1)$, $(x_2)$. |
| **Step2** | Calculate the index field (257) =$z_{12} \oplus x_1 \oplus c_{120}$ |
| **Step3** | Calculate the index field (258) =$z_{12} \oplus x_{12} \oplus c_{70}$ |

**Example.**
Compute the index fields F257 and F258 of the field F2, F3 (Mohamed) in the third row of the table in figure (4). Supposed, 120 is the constant of F257 and 70 for F258. ID value of the record is 89.
So, from the dictionary the encoded numbers of "Mohammed" are (102) and (57).
**Sol.**
To compute the index fields F257 and F258, uses following equation:

$$y_i = z_i \oplus x_i \oplus c_i$$

$$\mathbf{F257 = 89 \oplus 102 \oplus 120 = 71}$$

$$\boldsymbol{F258 = 89 \oplus 57 \oplus 70 = 38}$$

### 5.1.3 retrieving the records by using index seek

To retrieve a specific record from the remote DB, the user sends the request (for example, the first name of the person or card number) to the proxy server. The proxy server builds the SQL server query and sends it to the remote encrypted DB. The search mechanism is then applied in the indexes as follows:

SELECT #

FROM Encrypted Data Table

WHERE F257 = $F256_0 \oplus X_1 \oplus C_{F257}$ *and*
$F258 = F256_0 \oplus X_2 \oplus C_{F258}$ *OR*
$F257 = F256_1 \oplus X_1 \oplus C_{F257}$ *and*
$F258 = F256_1 \oplus X_2 \oplus C_{F258}$ *OR*
$F257 = F256_2 \oplus X_1 \oplus C_{F257}$ *and*
$F258 = F256_2 \oplus X_2 \oplus C_{F258}$ *OR*
……
……
……
$F257 = F256_{255} \oplus X_1 \oplus C_{F257}$ *and*
$F258 = = F256_{255} \oplus X_2 \oplus C_{F258}$

The index field values are matched with the SQL query by using index seek. The records that satisfy the condition of the required query are then returned to the proxy server and transferred to the user after decryption. Figure (5) shows the flowchart of the querying process via index seek method.

**The algorithm retrieve the required record by index seek**
**Step 1**. Enter the word of field "first name" for the required record in the application interface .
**Step 2**. User sends the request to the proxy server.
**Step 3**. The encoded word is fetched from the dictionary table.
**Step 4.** The proxy builds the query and sends to encryption table.
**Step 5**. The index seek operation is performed on indexes fields.

**Step6**. If the result not matching with a values of index fields
Go to step 9
Else Go to step 7

**Step 7** retrieved the specific records to proxy server and decrypts all encrypted records .
**Step 8.** The requested record are sent to the client.
**Step9**. Exit.

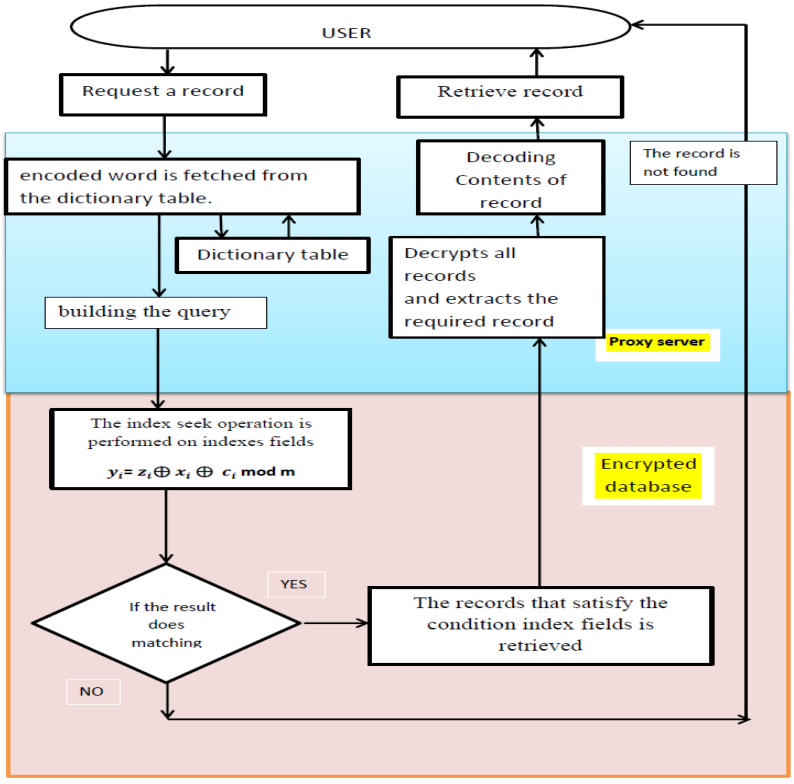


Figure (5) the flow chart of querying processing by index seek

## Example.

To retrieval of the required record in Figure (4), wherein the condition of query was the first name"Mohammed", is presented as follows:
- User sends a request for all records containing the name "Mohammad"via the interface application.
- The proxy server sets up the query, wherein the word 'Mohammed' is encoded as a pair of bytes (102), (57).

**SELECT #**
**FROM** Encrypted Data Table
**WHERE** $F257 = 0 \oplus 102 \oplus 120$ ***and***
$F258 = 0 \oplus 57 \oplus 70$ ***OR***
$F257 = 1 \oplus 102 \oplus 120$ ***and***
$F258 = 1 \oplus 57 \oplus 70$ ***OR***
$F257 = 2 \oplus 102 \oplus 120$ ***and***
$F258 = 2 \oplus 57 \oplus 70$ ***OR***
……
……
$F257 = 89 \oplus 102 \oplus 120$ ***and***
$F258 = 89 \oplus 57 \oplus 70$

- The search operation is executed via index seek. Firstly, the search starts in F257, and F258 is computed. If the results are consistent with the index fields, then the matching records are retrieved.

## 6. Results and Testing

This proposed technique is tested on the DB of an information system. The encryption table contains 1,000,000 records for testing purposes.
The proposed technique consumes less time during retrieval compared with conventional systems using the DES algorithm, which takes a long time to decryption and scanning operation on all records in a large encrypted table. Table (2) illustrates the time consuming to retrieve records of the proposed method compared to another common method such as DES for three samples of data.

Table (2) illustrates the time consuming to retrieve records

| Input File Size (KB) | Execution time of retrieve records (MS) | |
|---|---|---|
| | the Proposed technique | Conventional method (uses DES algorithm) |
| 1000 | 301.7648 | 333.8943 |
| 100000 | 544.2316 | 685.9431 |
| 1000000 | 951.9274 | 1.068.9231 |
| Average time | 599.308 | 696.254 |

Figure (6) shows two curves. The upper curve indicates the consuming time of the conventional method uses the DES algorithm in encryption database spends a lot of time to retrieve records from the encryption table and decrypts all records before sending the required record to the user. The second curve represents the consuming time in our proposed technique for retrieving the user's request by applying the query to the specific record.

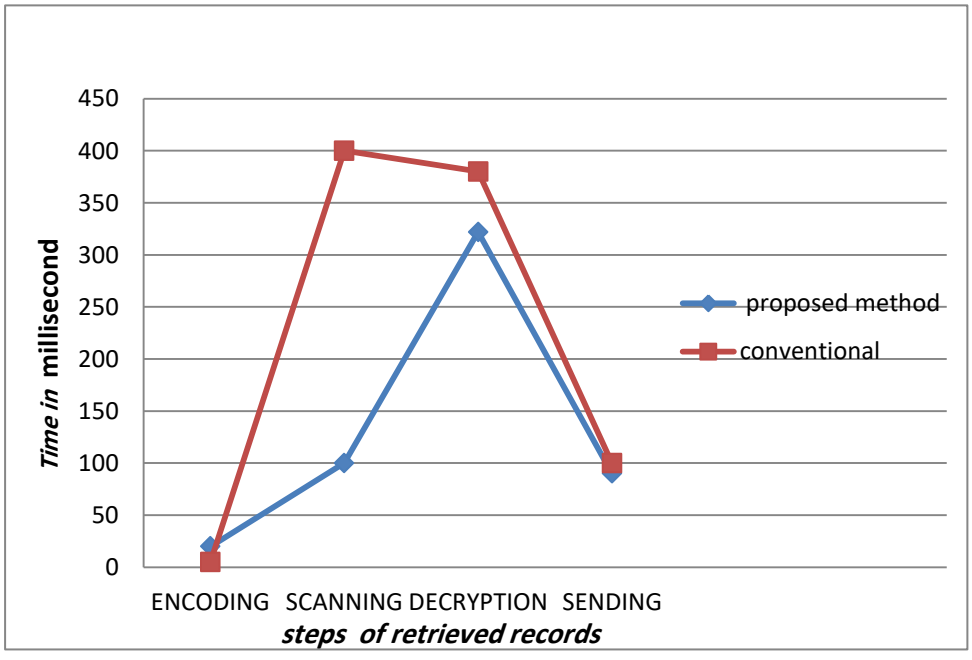


Figure (6) Comparison of queries in the encrypted DB

## 6. Conclusion

This study proposed an effective algorithm for querying encrypted data. The data retrieval process was considered the main objective in this paper rather than the encryption process. Thus, a simple encryption operation was performed to correspond with the retrieval method. In conventional systems, the query process in a large encrypted DB takes a long time because the DB must be retrieved entirely or partially before decrypted the results and sent to the client.

The proposed system improved the performance of the retrieval algorithm by decreasing the consumed time by retrieves only the requested records from the database without retrieving the set of records. The proposed system solved the time problem of seeking by building an indexing field on the encryption database.

## 7. References

[1] M. Sharma, A. Chaudhary, S. Kumar. "Query Processing Performance and Searching over Encrypted Data by using an Efficient Algorithm" International Journal of Computer Applications (0975 – 8887) Volume 62– No.10, January 2013.

[2] R. Brinkman. "searching in encrypted data" university of twente, to be publicly defended on friday, june 1, 2007 .

[3] J. Kumar. "Querying on Encrypted Data" CSE-303 Hall, Department of CSE, NIT Calicut. April 21, 2014.

[4] J. Li and E.R. Omiecinski, *"Efficiency and Security Trade-Off in Supporting Range Queries on Encrypted Databases"* Technical Report, pp. 69-83, 2005.

[5] H. Hacigümüs, B.R.I., C. Li and S. Mehrotra. "*Executing SQL over encrypted data in Database-Service-Provider Model*" ACM SIGMOD Madison, Wisconsin, USA, pp. 216-227, June 2002.

[6] P. Sudharaka. "*Homomorphic encryption and database query privacy*" Diss. Memorial University of Newfoundland, 2016.

[7] E. Damiani, et al. "*Balancing confidentiality and efficiency in untrusted relational DBMSs.*" Proceedings of the 10th ACM conference on Computer and communications security. ACM, 2003.

[8] E. Damiani, et al. "*Selective data encryption in outsourced dynamic environments."* Electronic Notes in Theoretical Computer Science 168 (2007): 127-142.

[9] E. Damiani, et al. "*A distributed approach to privacy on the cloud*." *arXiv preprint arXiv:1503.07994* (2015).

[10] Hore , B., Me hrotra, S., and Tsudik, G., *"A privacy-preserving index for range queries*", In Nascimento, M. et al., Eds., Proc. of the 30th International Conference on Very Large Data Bases , Toronto, Canada. Morgan Kaufmann, 2004, 720.

[11] Ceselli, Alberto, et al. *"Modeling and assessing inference exposure in encrypted databases."* ACM Transactions on Information and System Security (TISSEC) 8.1 (2005): 119-152.

[12] R. Raghu, G. Johannes*," Database Management Systems",* McGraw-Hill Higher Education (2000), 2nd edition (en) page 267.

[13] D .Sahab, AbdulMonem S.Rahma. *" Modifying Des Algorithm By Using Diagonal Matrix Based On Irreducible Polynomial."* Journal of Theoretical and Applied Information Technology Vol.97. No 5. 15th March 2019.