Table scan technique for querying over an encrypted database

Sahab Dheyaa Mohammed1
Prof. Dr. Abdul Monem S. Rahma2,
University of Information Technology
Department of Computer Science
and Communications, Baghdad, Iraq
University of Technology, Baghdad, Iraq

Abstract

The growth of data and the increase in the size of databases have led to difficulties for administrators in maintaining the confidentiality of data and protecting them from outside attackers. Encryption is necessary in protecting data in databases, but it has caused another problem in processing the required encrypted data and retrieving them from databases. In traditional encryption methods, the entire or a part of the database is retrieved and decrypted to detect user-requested data. This process is time consuming and does not prevent exposure by the administrator or external attackers. This work recommends a technique to address the drawbacks of traditional methods for encrypted database systems. The technique processes structural query language (SQL) queries over encrypted data in a remote database without decrypting all of the data, and data decryption is performed only at the proxy site. This technique is a new means of querying using the table scan mechanism by uses two steps, Distribution the encrypted tables and Generate indexes fields. This method is used when the number of the matching records is a large into the total number of records or when the table is small.

Keywords

encryption database, encoding data, indexing techniques, table scan, query processing

1. Introduction

Data are a critical resource and should be stored securely for the efficient operation of companies. Companies typically store data in secure databases (DBs), but this task poses a challenge to administrators in creating strategies to protect data from intruders. Cryptography is critical in DB security. Unlike conventional security techniques that cannot provide sufficient security for securing data, data encryption introduces considerable security, thereby preventing users from obtaining data illegally and stealing DB contents when these are saved on storage media, such as CD-ROM, tapes and disks.

Structural query language (SQL) queries cannot be performed immediately on an encrypted DB. All encryption data must be decrypted before an SQL query can be made, but this process is time consuming. Several mechanisms are applied to address this issue, but their applicability is limited [1]. The problem is exacerbated when additional information is stored. However, retrieval is the problem, not storage. When data are not encrypted, users can easily send an exact query to the remote DB and retrieve the exact information they need. However, users cannot acquire specific data from the server when all of the information is encrypted. Thus, every query involves retrieving the entire DB and decrypting its contents; only then can users send a query using their own computer [2]. The proposed technique exhibits enhanced searching performance

and faster data retrieval compared with conventional techniques. Moreover, the technique uses the scan mechanism to distinguish different records through the implementation of a hash function.

2. Related work

Providing security is an additional problem for DBs. The encryption techniques of the **database management system** (**DBMS**) can be used for this problem. However, despite the high security provided by encryption, several issues exist; for example, encryption reduces system efficiency.

In [3], an indexing technique for searching on range queries was proposed. However, this technique is helpful only for numeric data and not for character data.

[4] Suggested dividing the client's range of attributes into a set of intervals. The conformity between the intervals and the original values is preserved at the client's side, and encrypted tables with interval information are stored in the DB. Data are queried efficiently by mapping the original range and equivalent query values with the corresponding interval values.

[5] Proposed the creation of a B+-tree on plaintext values, and each B+-tree node is encrypted and stored in an unauthenticated DBMS. The main B+-tree is then applied in the unauthenticated DBMS as a table with two attributes, namely, the node ID, which is mechanically assigned by the system upon insertion, and the encrypted node content. The advantage of this technique is that the content of a B+-tree is invisible to an untrusted DB service provider, and the disadvantage is that it involves considerable data processing on client machines.

3. Indexing techniques for querying over an encrypted DB

Client machines may have a small storage and restricted computation capacity. Thus, one of the major goals of the query operation is to reduce the load at the client side whilst increasing the process performance at the server side [6]. The major question is how to compute and perform data indexing. Two incompatible requirements should be addressed. Firstly, the indexing data should be related to the data in a suitable form for efficient query execution. Secondly, the connection between indices and the data must not enable guesswork and attacks that can threaten the protection granted by encryption [5]. Several techniques that can be used to perform different types of queries at the server side are listed below.

3.1 Index by encryption

Every search key is encrypted with an invertible encryption function Ek(). A query in plaintext form must be converted into a query with a suitable encrypted form by executing the encryption operation on every value given in the authentic

query. This technique can maintain the distinction between values. However, intruders can guess between plaintext and encrypted values through frequency analysis by comparing the distributions of the plaintext relations with those of the ciphered values [6] [7].

3.2 Bucket-based index

This method differs from the previous approach. The domain D of search values is mapped into a set of non-overlapping parts P = {p1,..,pk} whose union covers all D. The index does not contain the single value of a row; rather, it contains its equivalence category (partition) [7]. Equality queries can be performed easily with the bucket-based index, although the resulting set must have an accuracy index <1 to prevent statistical data mining. Upon receiving the cipher data, an authorised user can decrypt them and eliminate spurious tuples. The range queries are too complex to compute because transmutation does not (and should not) generally preserve the arrangement relation of the plaintext data. This approach allows outsourced relational DBMS to respond to range queries that cannot be reduced to multiple equality conditions unless specific techniques are executed [7]. An efficient approach for dividing the domain of attributes to minimise the number of false tuples in the result of a range/equality query was demonstrated in [8].

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. The same result is obtained but without the proximity relationship [5].

3.4 Auxiliary B+-tree

A solution to address range queries is the use of an encrypted version of a B+-tree to store plaintext values and preserve the ordering [9].

Definition [10]: A B+-tree of order *m* is a tree where each internal node contains up to m branches (children nodes) and stores up to *m* -1 search key values. m is a popular branching factor.

The B+-tree is created from a set of nodes that serve as records or indicators of records. All 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, and they store the search key values and divide them amongst the children in the mode of a search tree.

4. Associated environment of the proposed method

The proposed technology operates on a remote DB, which is the enterprise proxy, and clients are represented as users. The proxy server is the contact centre between the client and remote DB. Figure (1) shows an overview of the query schema for the encrypted DB. No direct communication occurs between the remote DB and the clients. Instead, the proxy receives the records or queries from clients and performs a series of operations, such as encoding and encryption, then forwards the records or queries to the DB at a remote server. The proxy server performs the significant decryption and decoding operation on

the retrieved encryption records from the remote DB server and returns the final result to the user. The encryption and decryption operations are performed on the proxy server only, which means that the proxy plays the important role of a query parser. The proxy manages the communication between the DB applications and the encrypted remote DB.

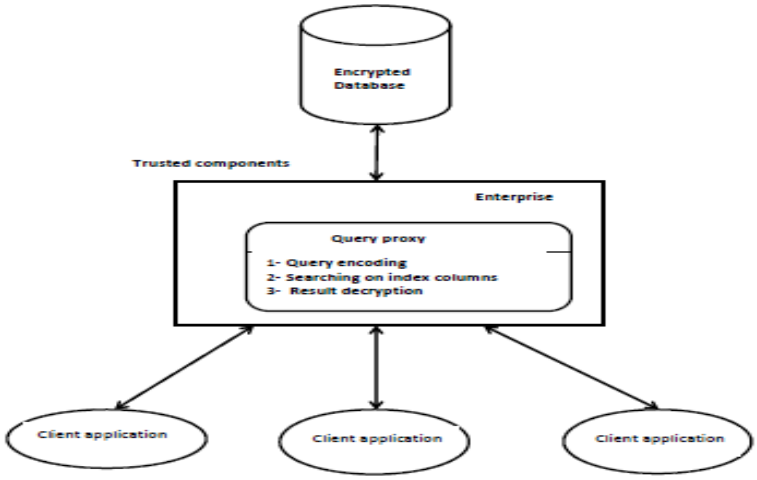


Figure (1) Scheme of querying encrypted database

4.1 Encoding and decoding operations

Before plaintext is sent to the encryption algorithm, it is encoded in a binary code by using a dictionary table, which is known as a substitution table. Each word or number (0…… 9) and symbol (#, $, /, (), *) of sensitive data is assigned into pairs of bytes and given a number in the secondary table used in the first method (to be explained in the subsequent sections). Every byte represents a numeric value (0…… 255) as an element polynomial of finite field GF (2^8).

When the encoder finds the required word, the word is substituted with a pair of bytes. Table (1) shows an example of encoding sensitive data. In the encoding operation, each pair of encoded bytes is matched with one word, one number and one symbol in the original DB. The dictionary contains up to 65,536 pairs of bytes that are stored dynamically or manually. All data in DB records are encoded in binary numbers in the form of polynomial numbers of GF (2^8). The dictionary includes words, numbers and symbols, and they are dynamically added by allowing the addition of new words or deletions of old words that are not used. Words that were previously entered manually are preserved.

Table (1) Encoding sensitive data

No.	Sensitive words	Numerical code		Binary code		Table NO.
		Byte 1	Byte 2	Byte 1	Byte 2	
1	Abbas	2	60	00000010	00111100	2
2	Ahmed	200	60	11001000	00111100	1
3	Ahlam	150	2	10010110	00000010	4
4	Bassam	15	223	00001111	11011111	10
5	Hamid	55	78	00110111	01001110	10
6	Hazim	1	98	00000001	01100010	11
7	Hakim	28	251	00011100	11111011	13
8	Male	254	1	11111110	00000001	-
9	female	254	2	11111110	00000010	-
10	Unmarried	33	28	00100001	00011100	-
⋮	⋮	⋮	⋮			
448	/	44	67	00101100	001000011	-
500	7	170	165	10101010	10100101	-
501	9	185	196	10111001	11000100	-
⋮	⋮	⋮	⋮			
65536	Adhamiyah	55	123	00110111	01111011	-

4.2 Encryption and steganography methods

The data are encrypted by a new algorithm used to modify the DES algorithm, thereby making this technique fast and safe. This technique performs all of its calculations on bytes rather than bits. Hence, the technique obtains the eight-byte plaintext block as a non-zero polynomial of finite field GF (2^8). The eight bytes are encrypted using 16 rounds. Each round comprises four keys (two external and two internal). The main key size is 64 byte (512) bit as a non-zero polynomial of finite field GF (2^8). The mathematical operations in this algorithm are based on GF (2^8). Hence, all operations use an irreducible polynomial ($x^8 + x^4 + x^3 + x + 1$). Therefore, decryption requires the inverse of the four key matrices in each round. Figure (2) shows a schematic of the proposed technique structure.

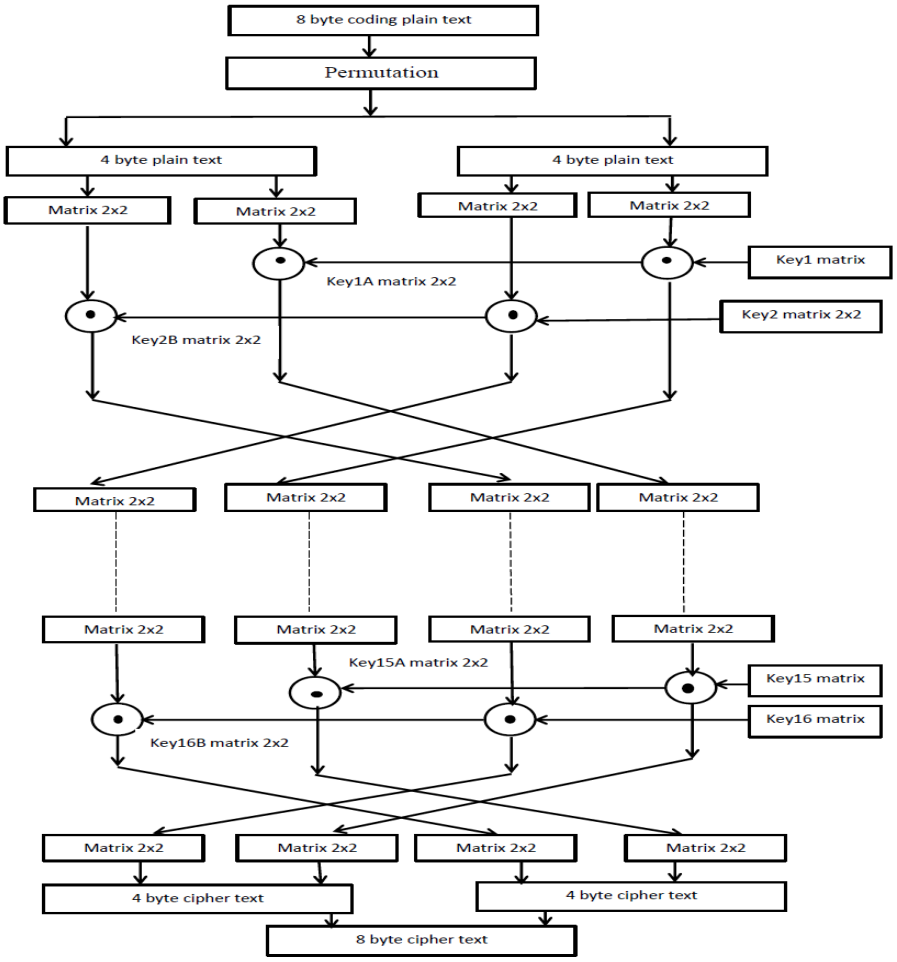


Figure (2) Schematic of MDES technique structure

The outputs of this encryption technique are gathered in a temporary table as DB records. A new hybrid technique for information hiding is then proposed. This hybrid technique combines the characteristics of steganography and classical encryption, so the encryption method is enhanced by hiding cipher records in a random matrix (256 × 256). The affine cipher function is used to merge encryption records with the elements of the matrix (cover file) before sending to the remote DB [11]. A flowchart of hiding cipher records in a random matrix is illustrated in Figure (3).

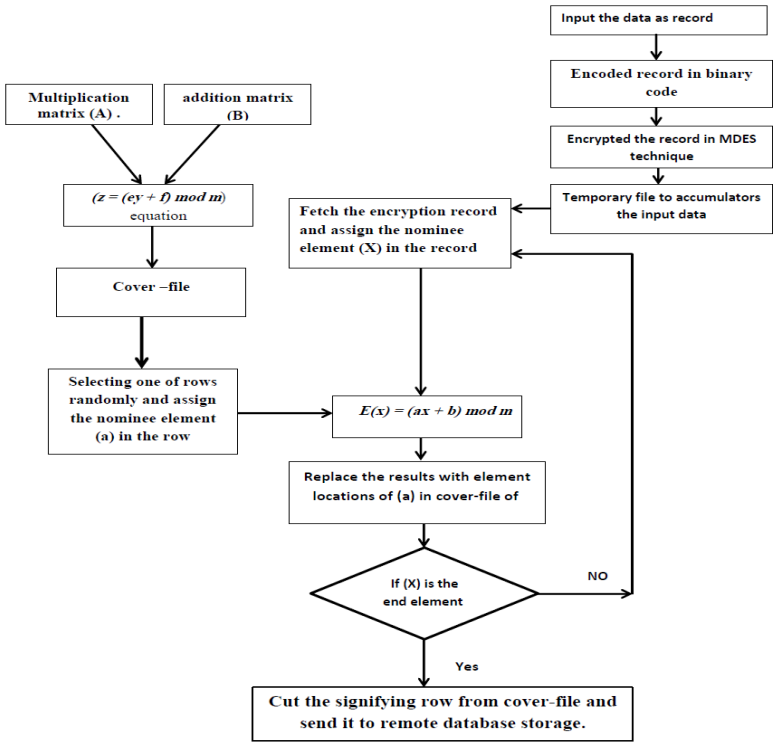


Figure (3) a flowchart of hiding cipher records in a random matrix

This technique prevents repetitions between cipher text elements and provides no chances for attackers to detect encrypted data being sent to the remote DB. The embedding procedure begins when encoded and encrypted records of data and block matrices (Z) are available. This procedure uses a technique of embedding the encrypted DB record with one of the rows randomly selected from a block matrix (Z). The results are replaced with the same location of the elements in a cover file and sent to the remote DB [11].

5. Proposed work

This work proposes a method to retrieve certain records directly from an encrypted DB without retrieving all or part of the data and then after that selected certain records. Certain records are retrieved, decrypted and sent to the user. This method preserves data confidentiality in an unreliable environment and improves search and update capabilities. This technique has been recommended to address drawbacks of traditional encryption methods for DB systems. It processes SQL queries over encrypted data on a remote DB without having to decrypt all of the data, and decryption is performed only at the proxy site. Therefore, it is efficient, and the exact set of requested data is returned to the client.

Two methods can be used to retrieve data in a DB. The first one uses a table scan, and the second uses an index. These techniques help retrieve required encryption data. The first technique is the proposed method of querying using a table scan by distributing encrypted tables and pointer records. The second is alternative method using indexes seek.

5.1 Proposed method of querying using a table scan

This method is an alternative for the index seek method when retrieving a large amount of data or when the table does not have an index. The index may present a vulnerability that helps the attacker guess the plaintext through the index information available in the dictionary table when exposed to attacks. This method involves two steps, namely, distributing the encrypted tables and generating record pointers.

5.1.1 Distribution of encrypted tables

Different techniques of storing data in DBs involve encrypting the records and distributing them into multiple partitions of tables to increase security and reduce search time. This technique executes queries on certain tables (sub-tables) as possible at the DB server site without needing to execute queries on a single large table and retrieved certain records. This process minimizes the computation of scanning time at the sub table of the DB server. To implement this technique, the attribute (key) that will be used in all queries needs to be determined, and the attribute values are classified based on the frequency of use and availability in the dictionary table to organise categories in each group. When the distribution is executed, each record is mapped to a specific table on the basis of the attribute (key) value. Figure (4) presents the original DB table with (2,000,000) records, where the field "first name" in the DB table is selected as an attribute (key).

	First Name	father	Grand father	surname	mother	Grand mother	gender	B.T.	Issuer	Effective date	Place of birth	Data of birth
1	Mohammed	Ali	Hakeem	AL-Bader	Noor	Akrm	Bkr	A	Rusafa	2/5/2020	Baghdad	4/7/1960
2	Huda	Bader	Ali	AL- Salam	Hind	Amjd	Bha'a	O+	Baiyaa	4/7/2019	Basra	4/3/1995
3	Sameer	Mustafa	Omar	Al-Hbyb	Mariam	Amyn	Thabt	AB	Adhamiyah	22/5/2018	Babil	8/9/1996
4	Mohammed	Mahmoud	Mohammed	Al-Ansary	Azhr	Baqr	Blal	O-	Al-Jadriya	25/8/2019	Najaf	5/7/1992
5	Kareem	Jabber	Hassan	Al-karam	Fatima	Bdry	Thamr	A	Karrada	11/1/2018	Diyala	2/8/1980
6	Mohammed	Hamada	Kader	Al-Rhmn	zainab	Bshyr	Thsyn	AB-	Adhamiyah	12/2/2017	Al Anbar	8/5/1970
----------	----------	----------	----------	----------	----------	----------	----------	------	----------	----------	----------	----------
1999999	Azhr	Blal	Bdry	Al-Fahd	Zahra	Ali	Ahmed	B	Rusafa	2/5/2017	Baghdad	2/8/1950
2000000	zainab	Mohammed	Amyn	Al-Kreem	Mariam	Jabber	sameer	A-	Karrada	5/8/2015	Babil	8/5/1940

Figure (4) represent the original database table

The frequency of attribute values usually exhibits a variation. Several values occur frequently in the table, whereas doing not; thus, they can be distributed

into groups of separated tables to accelerate the search and query process. Table (2) shows the frequency ratio of the attribute values in the DB table with (2,000,000) records. Figure (5) presents the distribution of attribute values on the sub-tables on the basis of the frequency ratio. The frequency ratio is computed using Equation (1).

$$\textbf{Frequency ratio} = \frac{\text{frequency number}}{\text{Number of records}} \times 100\% \qquad \textbf{(1)}$$

Table (2) shows the frequency ratio of the attribute values

	Values of fist name field	frequency number	frequency ratio	(table)NO.
1	Mohammed	100000	5 %	1
2	Sameer	10000	0.5 %	5
3	Kareem	20000	1 %	5
4	Huda	25000	1.25 %	4
5	Hassan	40000	2 %	4
6	Mahmoud	50000	2.5 %	3
7	Azhr	300	0.015 %	5
-------	----------	----------	----------	---------
-------	----------	----------	----------	----------
59999	Jabber	10000	0.5 %	5
60000	Fatima	45000	2.25 %	3

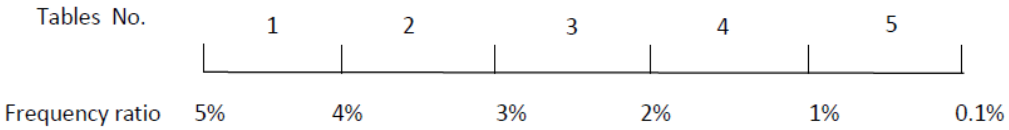


Figure (5) the distribution of attribute values on the sub-tables

In this method, the encrypted DB file is divided into five secondary files with the same structure. This method helps accelerate the search process during querying on a particular record by scanning only the specific secondary file instead of scanning all records in the large main file. If a new record is entered, the new name of the attribute (first name) is inserted into the dictionary table and assigns the number of the secondary table to it.

5.1.2 Construction the indexing

The index field is added to the encrypted DB table in the DB server. The values of this field represent pointers indicates to the encrypted records in the DB table. In this method, the pointers are calculated and added as a field with each record before sending it to the proxy server. The pointer values are calculated by applying a hash function on the elements of each record. The results of this function represent pointers to the specified records.

Searching for the plaintext value in the encrypted DB table is difficult. Thus, when the SQL query executes a table scan, it begins from the physical start of the table and continues with each record in the table. If a record matches the criterion, then it will be included in the results. The structure of the encrypted DB is based on the fact that each byte in the record is located under one field, so the pointers of each record may contain more than one field (byte) based on the number of bytes that represent the values of the field (key). Figure (6) shows the structure of the original DB and the encrypted DB.

	Name	father	Grand-f	surname	mother	Grand-m	gender	blood type	Issuer	Issuer Date	Effective Date	place of birth	date of Birth	Family number
1	Mohammed	Dlaa	Ali	Bader	Noor	Hmeed	Male	03	Adhamiya	2017/01/22	2027/01/21	Baghdad	1973/07/03	4578439
2	Huda	Ameer	Kreem	Al-karam	Hind	Salam	female	A+	Kadhimiya	2016/01/20	2027/01/21	Baghdad	1983/07/03	4578440

a . original database

Pointers

	F0	F1	F3	F4	F5	-	F23	-	F66	-	F79	-	F213	F235	-	F244	F254	F255	F256	F257
0	63	123	213	4	62	-	7	-	107	-	126	-	240	10	-	115	13	26	91	39
1	211	45	100	89	11	-	207	-	141	-	183	-	105	15	-	176	27	83	89	177
2	56	12	150	231	177	-	16	-	140	-	225	-	104	14	-	61	33	128	249	96
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
60001	51	122	102	164	99	-	136	-	246	-	126	-	226	206	-	238	72	8	32	33
6002	123	134	160	164	48	-	123	-	214	-	183	-	254	244	-	225	218	12	68	253
-	-	-	-	-	-	-		-	-	-		-	-	-	-		-	-	-	-
69678	34	129	188	190	81	-	297	-	109	-	259	-	169	19	-	235	94	13	55	75

b. encrypted database

Figure (6) shows the original database and encrypted database.

To generate the index records, each record computes pointers by using a hash function. Firstly, the parameters of the hash function, including the elements of records and the values of the attribute (key) used in all queries, are assigned. Secondly, the XOR operation is used to gather the parameters by applying the hash function (2) on each record as follows:

$$Y = W_i \oplus (f_i \oplus f_i \oplus f_i) \qquad (2)$$

Where

Y: value of pointer.
i: (0……..255)
f : the field that be selected
w: one byte of the encoded word that we wants to find it.

Thirdly, the results represent pointers that indicate the records. The properties of these pointers are not repeated for more than one record, and collision is rare. Figure (7) illustrates the algorithm for the computation of the pointers in the encrypted table.

The algorithm of generate the pointers records

Input: parameters of the coded words and the elements of record. **Output:** pointers	
Step1	Determined the attributes (keys) w_i that used in all queries.
Step2	Determined the attributes (f_i) that uses as parameters in hash function.
Step3	Applying the hash function $Y = w_i \oplus (f_i \oplus f_i \oplus f_i)$ on the parameters that are determined in steps above .
Step4	the results (Y_i) of hash function are a pointers located at the end record.

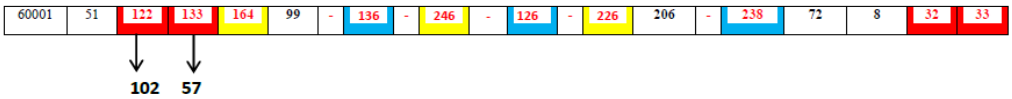
Example
In Figure (6), suppose that a proxy server encrypts and hides record number (60001) and wants to send this record to the encrypted DB. The first name field (key) is ‘Mohammad.’ The word ‘Mohammad’ is encoded as a pair of bytes [(102), (57)]. To add the pointers in the end record, the hash function is applied on the record as follows:

$$Y = W_i \oplus (f_i \oplus f_i \oplus f_i) \quad (2)$$

$$f257 = 102 \oplus (f4 \oplus f66 \oplus f213)$$
$$f258 = 57 \oplus (f23 \oplus f79 \oplus f244)$$
$$F257 = 102 \oplus (164 \oplus 246 \oplus 226)$$
$$F257 = 32$$
$$F258 = 57 \oplus (136 \oplus 126 \oplus 238)$$
$$F258 = 33$$

Key

	F0	F1	F3	F4	F5	-	F23	-	F66	-	F79	-	F213	F235	-	F244	F254	F255	F256	F257
0	63	123	213	4	62	-	7	-	107	-	126	-	240	10	-	115	13	26	91	39
1	211	45	100	89	11	-	207	-	141	-	183	-	105	15	-	176	27	83	89	177
2	56	12	150	231	177	-	16	-	140	-	225	-	104	14	-	61	33	128	249	96
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
60001	51	122	133	164	99	-	136	-	246	-	126	-	226	206	-	238	72	8	32	33
6002	123	134	160	164	48	-	123	-	214	-	183	-	254	244	-	225	218	12	68	253
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
69678	34	129	188	190	81	-	297	-	109	-	259	-	169	19	-	235	94	13	55	75



102 57

Figure (7) is illustrates the computation the pointers

When an authorised user wishes to ***retrieve the required record*** of his information from the remote encrypted DB table, he enters his first name in the application interface. The request is sent to the proxy server, which obtains the encoded word (of the first name) as a pair of bytes and the table number from the dictionary table. The proxy server sends the query of the SQL server to the particular table at a remote encrypted DB and uses the table scan mechanism by executing the hash function on each record. When the results of the equation match the values of the pointers, all records that satisfy the user query are returned to the proxy server, and the records are decrypted before they are sent to the user. Figure (8) shows the flow chart of querying processing.

Outline of the algorithm to retrieve the required record via table scan

The algorithm for retrieving the required record consists of the following steps.

Step 1. The word (first name field) for the required record is entered in the interface.

Step 2. The user sends the request to the proxy server.

Step 3. The encoded word and table number are fetched from the dictionary table.

Step 4. The proxy sends the query to the specific encryption table.

Step 5. The hash function is implemented on each record, and the results are compared with the

pointer columns.

If the result does not match all values of the pointer columns, then the system displays the message:

"*the record is not entered previously.*" *Go to step 9* Else

Step 6. The record that has a value identical to the pointer is retrieved.

Step 7. A proxy server decrypts encrypted rows and extracts the required record from the user.

Step 8. The requested record is sent to the client applications.

Step 9. Exit.

Example:

In Figure (6), suppose that a user wants to retrieve records from the encrypted DB, and the first name field is a key. For example, all records of the name 'Mohammad' in the encrypted DB are needed.

Solution

- The user sends a request for all records of the name (Mohammad) by using the interface application.
- The proxy server sets up the query as follows:

SELECT #
FROM Encrypted Data Table
WHERE Per_Name = Mohammad

- The proxy server performs the encoding mechanism where the word 'Mohammad' is encoded as a pair of bytes (102), (57) and table number (1) as follows:

SELECT #
FROM Table 1
WHERE f257= 102 ⊕f4⊕ f66 ⊕ f213 **And f258** =57 ⊕f23 ⊕f79⊕f244

- In the encrypted DB, the hash function is performed on each record as follows:

F257= 102 ⊕ 164 ⊕246 ⊕226 = 32
F258= 57 ⊕ 136⊕126 ⊕ 238 = 33

If the first byte of the value in F257 Column = 32 and the second byte value in F258 column=33, then the record that these values point to is selected, and a copy of that record is sent to the proxy server. Otherwise, the hash function is performed on the next record and so on.

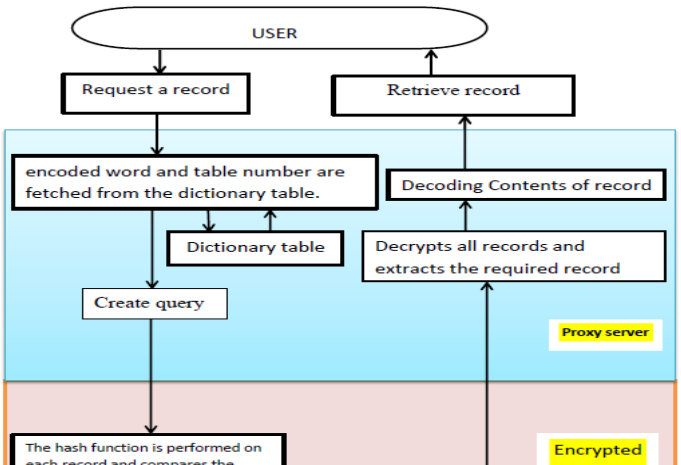


Figure (8) Flow chart of querying processing

6. Results and testing

This proposed algorithm is evaluated on the DB of an information system. The encryption DB table includes 2,000,000 records for testing purposes. In practice, retrieving these records is generally fast. Thus, the proposed algorithm is suitable for the ideal environment.

Our proposed technique consists of four steps for record retrieval. Each step takes different time in millisecond depending on the number of required records. Therefore, the complete time for record retrieval is equal to the total time required by these steps.

Our proposed technique consumes less time during retrieval compared with conventional systems using DES and 3DES algorithms, which require a long time to decrypt all records and a long time in scanning operation on all records in a large encrypted table. Table (3) shows the time it takes to decrypt the user-requested records for the proposed method compared to other common methods of DES and 3DES algorithms for three samples of data.

Table (3) shows the time it takes to decrypt on three methods

Input File Size (KB)	DET: Decryption Execution Time (MS).			
	Proposed encryption algorithm	DES 56	3DES	
			112	168
1000	331.9876	363.9520	370.3587	380.3472
100000	444.5721	485.5811	528.9593	550.6792
1000000	651.9862	716.3734	809.2216	812.4949
Average Time	**476.182**	**521.9688**	**569.5132**	**581.17376**

The table above shows that the decryption time in our proposed method Relatively less than the other two methods .also the time in scanning operation is less in our method because the main table is divided into five sub-tables, so the scan operation consuming in our method less 5 times than other conventional methods.

Figure (9) shows curves. The upper curve denotes the time that the conventional system that uses the DES algorithm in encryption database spends on retrieving records from one single large table of 2,000,000 records and decrypts all records before sending the required record to the user. The second curve represents the time spent by our proposed technique on retrieving the user's request from the encryption DB by applying the query to the specific sub-table, not to the main table.

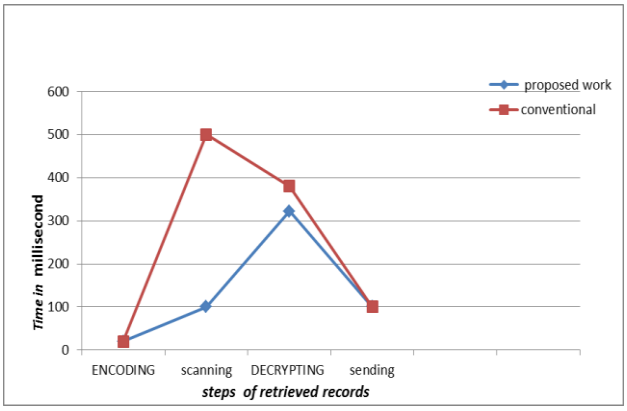


Figure (9) Comparison analysis of querying over an encrypted database

6. Conclusion

This work proposed an effective algorithm for querying encrypted data. Data retrieval, not encryption, was the main focus of this research. Thus, an encryption process was quickly reviewed to explain the performance of the data retrieval method (by matching the encoded query with the encrypted data). In traditional systems, the query process of a large encrypted DB requires a long

time because the DB needs to process all records and decrypt them before sending the results to the client. The proposed system improves the performance of the retrieval algorithm by reducing the time consumed. The DB file is divided into five files depending on the occurrence frequency of the elements of the attribute used in all queries. Thus, the amount of time spent to retrieve one record is reduced by five times compared with conventional methods.

7. References

[1] M. Sharma, A. Chaudhary, S. Kumar. "Query Processing Performance and Searching over Encrypted Data by using an Efficient Algorithm" ternational 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. Li and E.R. Omiecinski, "Efficiency and Security Trade-Off in Supporting Range Queries on Encrypted Databases," Technical Report, pp. 69-83, 2005.

[4] 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.

[5] Damiani, E., et al., "Balancing confidentiality and efficiency in untrusted relational dbmss", In Proceedings of CCS'03, pages 93{102, 2003.].

[6] E. Damiani, S. Vimercati, S.Foresti, S. Jajodia, S. Paraboschi, P. Samarati," Selective Data Encryption in Outsourced Dynamic Environments", Electr. Notes Theor.Comput. Sci. (ENTCS) 168:127-142 (2007) .

[7] Pagano, F., *A Distributed Approach to Privacy on the Cloud.* arXiv preprint arXiv:1503.08115, 2015.]

[8] 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.

[9] A.Ceselli, E. Damiani, . Vimercati, S. Jajodia, S.Paraboschi, P. Samarati," Modeling and assessing inference exposure in encrypted databases", ACM Trans. Inf. Syst. Secur. (TISSEC) 8(1):119-152 (2005).

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

[11] 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.