Light Weight and Secure Database Encryption Using TSFS Algorithm

D.Manivannan[1], R.Sujarani[2]
[1]*Asst. Professor, School of Computing*
[2]*Pursuing M.Tech (CSE), School of Computing, SASTRA University, Thanjavur*
dmv@cse.sastra.edu, rsujarani@yahoo.com

Abstract - **Database security has paramount importance in industrial, civilian and government domains. Organizations are storing huge amount of data in database for data mining and other types of analysis. Some of this data is considered sensitive and has to be protected from disclosure. Challenges for security in database are increased due to the enormous popularity of e-business. In recent years, insider attacks gathered more attention than periodic outbreaks of malware. Database systems are usually deployed deep inside the company network and thus insiders has the easiest opportunity to attack and compromise them, and then steal the data. So data must be protected from inside attackers also. Many conventional database security systems are proposed for providing security for database, but still the sensitive data in database are vulnerable to attack because the data are stored in the form of plaintext only. Database encryption is the only solution for avoid the risk posed by this threat.**

This paper focuses on a security solution for protecting of data-at-rest, specifically protecting the sensitive data that resides in databases by using TSFS algorithm with three keys thus it provide more security for database. This algorithm improves the efficiency for executing the queries in database by encrypting only the sensitive data.

Keywords - **Database Encryption, Key expansion, Transposition, Substitution, Folding, Shifting.**

I. INTRODUCTION

Many organizations increase their dependency on database systems for day-to-day operations and for making decision. With a networked database in the complex multi-tiered applications, multiple parties such as customers, partners, and internal and external users will share the information inside the database. The security of data managed by these systems becomes crucial. Damage and misuse of sensitive data stored in the database not only affect a single user also affect entire organization. The recent development of web based applications and information systems have further increased the risk exposure of databases. The available security policies cannot provide a secure support for the sensitive data, which reside in the database, as the illegal and unauthorized users may obtain the readable data.

There are four methods of enforcing database security: physical security, operating system security, DBMS security, and data encryption. Out of our survey, the first three methods however are not totally satisfactory solutions to the database security problem, for the following four reasons. First, it is difficult to control the attack on raw data because the raw data exist in readable form inside a database. Second, it is impossible for the operating system and DBMS security to the disclosure of sensitive data, because the sensitive data must be backed up in storage median in case of system failure. Third, it is hard to protect the confidential data in a distributed database system. Fourth, it is hard to verify that the origin of a data item is authentic, because an intruder may have modified the original data. Encryption of the data has the potential to solve all the previously mentioned problems, If the data are not in a readable form, obtaining the data will be of no advantage to a person without the proper key to decrypt it. Thus the problem of data disclosure can be eliminated and the data authenticity problem can also be largely solved by encryption.

In this paper, we propose an efficient light-weight database encryption techniques using TSFS (Transposition, Substitution, Folding, Shifting) algorithm, only the sensitive data in the database are encrypted by using this algorithm, so it will provide efficient execution of queries and give quick response to the users. TSFS is the symmetric - key block encipherment algorithm, for symmetric encryption, same key is used for encryption and decryption and security is dependent on the length of the key. Here we use three keys for the process of encryption and decryption. For providing effective and more security for the database these three keys are expanded in into 12 sub keys by using the key Expansion Technique.

The remainder of this paper is organized as follows: Section 2 discusses the existing system for securing the database and analyzes the strength and drawbacks. Section 3 presents how the three keys are expanded in to sub keys. Section 4 briefly discusses the design and implementation of the proposed approach. Section 5 presents some strength of the TSFS algorithm. Section 6 explain the security analysis of the algorithm. Section 7 draws the conclusion.

II. RELATED WORK

Many methods having more creativity and efficient implementation have been proposed for database security research field. Using efficient keys and sub keys a database encryption scheme based on the Chinese Reminder theorem [4] also implemented. In that theorem, a record oriented cryptosystem in the database, which enables encryption at all the level of rows and decryption at all the level of cells are implemented. Extension of the sub key in encryption by supporting multilayer access control is proposed by Hwang and yang to enhance the security level. They also introduced a two-phrase algorithm for database security [5]. Another scheme called chip secured data access [6] principles as a solution to data confidentiality problem. This solution is quite secure and effective, but it is still too complicated and the cost is too expensive.

These database encryption mechanisms provide an effective way to keep the sensitive data in security by store the data in encryption form. But once the database is encrypted, the efficiency of DBMS will fall, as the query cannot execute over the encrypted attribute directly. In order to get the query results, the users have to decrypted the whole data first and then conduct the query over the plaintext data. The process of decryption will not only affect cost of time but also leak the sensitive data to the attackers. There are several methods try to solve the problems on the efficiency and secure on the sensitive database. In privacy homomorphism technology [7], the queries can execute directly in the encrypted database. But this schema has weak encryption strengthen. The attacker could get some sensitive information through simple comparison of parts of cipher text and plaintext.

In the OPES approach [8] the queries are directly applied on encrypted data, without decrypting the operands. This scheme allows comparison operation to be directly applied on encrypted data without decrypting the operands. Thus equality queries as well as the max, min, and count queries can be directly processed over encrypted data. But it is not sufficient for executing the complex queries ant it is not inherently secure for straightforward attack. Encryption for indexing in a column-oriented DBMS [9] is another method for database encryption that encrypt only the specified columns so that only the sensitive data are selected for encryption thus the process of encryption and decryption time is reduced. Another scheme numeric to numeric database encryption for encrypting the numeric data [10], but this approach is not fit for the character data as the character data has its specific features which is different from the numeric data and this scheme work only for integers not for decimal point numeric data, Sometimes character data could be a very sensitive one so that we have to provide security for both numeric and character data.

Out of those surveys, we present TSFS algorithm to overcome some drawbacks in the existing methodologies and also ensure the confidentiality and integrity of the data in the database.

III. KEY EXPANSION

In this step, each key is expanded to many sub keys to be used in each round. In general the keys are expanded by shifting the rows [11] and by using Add round key technique in AES and many techniques are available for expanding the key. In the proposed scheme, three keys are used and each key is expanded to four sub-keys. The keys are in any format consist of numbers, alphabets, combination of both but not accepting the special characters and symbols. In order to increase the strength of the algorithm, the given keys are stored in the form of 4 x 4 matrix so the length of the key must be in 16 digits. If the user gave the keys below 16 digits, first the keys are converting in to 16 digits by using padding technique and then stored the keys in the matrix. After that shifting the rows for key expansion and it will be used in real time process to expand the keys.

The following example describes how the three keys are expanded into 12 sub keys.

For example when the

Key1 value is 66FF3CE3491C5EDA
Key 2 value is 95EFFBE191E22DB4
Key 3 value is 9CC98A29456677A6

Here, we get these keys by using a random key generator. It is not necessary that a random key generator must only be used for obtaining the key values. Key values are specified by the users as they wish.

First the keys are converted into numbers based on the position in the alphabets a-z (a-0, b-1------z-25). Then the keys are stored in 4*4 matrix form.

The keys are expanded based on shifting the rows.

Key1 is expanded into key_{10}, key_{11}, key_{12}, k_{13}.

For key_{10} - row 0 is not shifted, row 1 is shifted one time, row 2 is shifted two times and row 3 is shifted three times

For key_{11} - row 0 is shifted one time, row 1 is shifted 2 times, row 2 is shifted three times and row 3 is not shifted

For key_{12} - row 0 is shifted two times, row 1 is shifted three times, row 2 is not shifted and row 3 is shifted one time

For key_{13} - row 0 is shifted three times, row 1 is not shifted, row 2 is shifted one time and row 3 is shifted two times

Figure.1 shows the key expansion process

Key 1

6	8	5	5
3	2	4	3
4	9	1	2
5	4	3	0

Key 10

6	8	5	5
2	4	3	3
1	2	4	9
0	5	4	3

key 11

8	5	5	6
4	3	3	2
2	4	9	1
5	4	3	0

key 12

5	5	6	8
3	3	2	4
4	9	1	2
4	3	0	5

key 13

5	6	8	5
3	2	4	3
9	1	2	4
3	0	5	4

Key 2

9	5	4	5
5	1	4	1
9	1	4	2
2	3	1	4

Key 20

9	5	4	5
1	4	1	5
4	2	9	1
4	2	3	1

key 21

5	4	5	9
4	1	5	1
2	9	1	4
2	3	1	4

key 22

4	5	9	5
1	5	1	4
9	1	4	2
3	1	4	2

key 23

5	9	5	4
5	1	4	1
1	4	2	9
1	4	2	3

Key 3

9	2	2	9
8	0	2	9
4	5	6	6
7	7	0	6

Key 30

9	2	2	9
0	2	9	8
6	6	4	5
6	7	7	0

key 31

2	2	9	9
2	9	8	0
6	4	5	6
7	7	0	6

key 32

2	9	9	2
9	8	0	2
4	5	6	6
7	0	6	7

key 33

9	9	2	2
8	0	2	9
5	6	6	4
0	6	7	7

Fig .1 key expansion

IV. PROPOSED ENCRYPTION SCHEME

In this paper, we propose an efficient lightweight database encryption scheme (called TSFS). The main objective of this paper is to propose a secure database encryptions scheme that provides maximum security, whilst limiting the added time cost for encryption and decryption. Only the sensitive data are encrypted, so it is very effective for executing the queries. This algorithm is working for numeric data, decimal point numeric data and also for the alphanumeric data.

A .Overview of the algorithm

To provide security, TSFS algorithm uses four types of transformations: Transposition, Substitution, Folding and Shifting.

Transposition and substitution ciphers are still the most important kernel techniques in the construction of modern symmetric encryption algorithms. The benefit of these two ciphers is that they have two factors of cryptology and security, diffusion and confusion. In this algorithm, we mostly use transposition and substitution ciphers techniques.

Let's see how TSFS uses four types of transformations for encryption and decryption. In the standard, the encryption algorithm is referred to as the cipher and the decryption algorithm as the inverse cipher. This algorithm is a non-Feistel cipher, which means that each transformation or group of transformations must be invertible. In addition, the cipher and the inverse cipher must use these operations in such a way that cancel each other. The keys must also be used in the reverse order. The following figure shows the overall view of the algorithm.

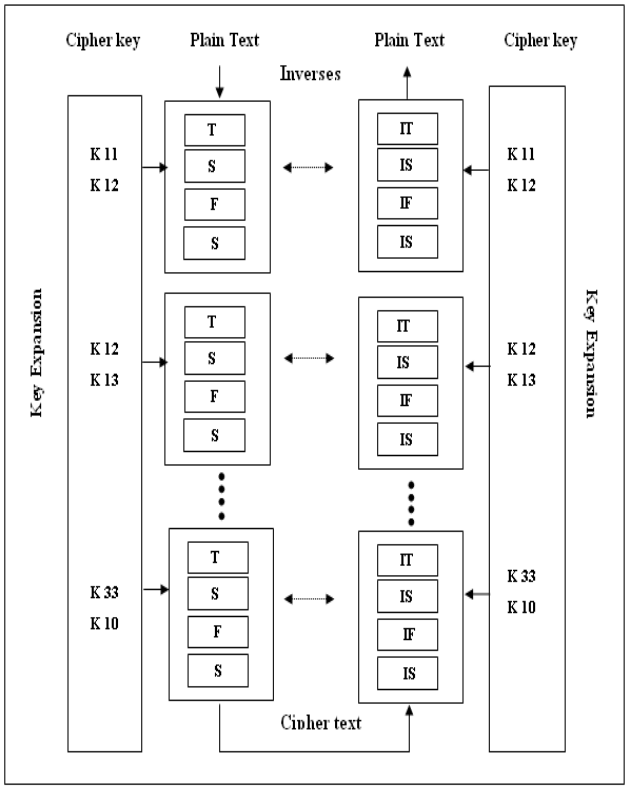

Fig .2 Overall view of the algorithm

T - Transposition, S - Substitution, F - Folding, S - Shifting

IT - Inverse Transposition , IS - Inverse Substitution

IF - Inverse Folding, IS - Inverse Shifting

V. DESIGN AND IMPLEMENTATION

If user wants to encrypt private data that can be seen and altered only by himself, a randomly generated working key will be used to encrypt the private data with a conventional encryption algorithm. After the data entered by the user immediately that data is encrypted by using the algorithm and the encrypted data is stored in the database. This algorithm is working for both numbers and characters and also it accept the alphanumeric data also (for example the account number is asdf40985490), if decimal point numbers are entered then that number is multiplied by 100 and then applied to the algorithm. Why we multiply by 100 means because for Indian Rupees the Pisa column have only two integers so that for covert the decimal point numbers to normal numbers, we use this technique. After the algorithm encrypted the data and the data is divide by 100 then it stored in the database. Thus the decimal point number is stored in the same data type in the form of cipher text.

A. *Transposition*

Transposition ciphers are and important family of classical ciphers. It does not substitute one symbol for another; instead it changes the location of the symbols. A symbol in the first position of the plaintext may appear in the tenth position of the cipher text. In other words, a transposition cipher reorders the symbols. In this algorithm we use diagonal transposition cipher the entered data are stored in 4 x 4 matrix form and then the data are taken diagonally and stored in the another matrix for the consequence. The following figure show how the data are taken diagonally. For example the entered data is the account number: asdf48723498. After getting the data, pad the input data with 0s and stored in the matrix form. Just like this (left matrix) and the data's in the matrix are read in the route of zigzag diagonal starting at the upper left corner. The following figure shows the result of the transposition.

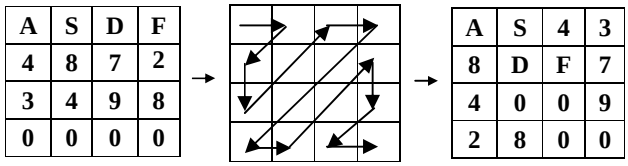

Fig.3 Transposition

B. Substitution

A substitution cipher replaces one symbol with another. If the symbols in the plaintext are alphabetic characters, we replace with one character with another. For example, we can replace letter A with letter D, and letter T with letter Z. If the symbols are digits (0 to 9), we can replace 3 with 7 and 2 with 6. Substitution ciphers can be categorized as either monoalphabetic ciphers or polyalphabetic ciphers. In monoalphabetic substitution, the relationship between a symbol in the plaintext to a symbol in the cipher text is always one-to-one. In polyalphabetic substitution, each occurrence of a character may have a different substitute. The relationship between a symbol in the plaintext to a symbol in the cipher text is one-to-many. Here we use a new modified affine cipher for encryption. It is one of the monoalphabetic ciphers available.

Normally affine cipher is a combination of additive and multiplicative cipher. For this we have to use two keys one for the additive cipher and another for multiplicative cipher.

By using this cipher the Encryption process is

$C = (P \times k1+k2) \bmod M$

Decryption process is

$P = ((C - k2) \times k1^{-1}) \bmod M.$

In this cipher the multiplicative inverse of k1 only exists if k1 and M are co prime. Hence without the restriction on k1 decryption might not be possible. What is the key domain for any multiplicative cipher? The key must be in the range from 0 to 26.

This set has only 12 members :1,3,5,7,9,11,15,17,19,21,23,25. Considering the specific case of encrypting messages in alphanumeric in English (i.e. M=26), So there are 12 x 26 or 312 possible keys. So it is easy for the cryptanalyst to find the key.

To overcome this draw back we slightly changes this affine cipher i.e. here we eliminate the process of multiplicative cipher and add one more additive cipher. In this cipher we give more importance for selecting the two keys for encryption. We expand the three keys into twelve keys and stored in the form of 4 x 4 matrix and also the entered data are also stored in the form of matrix. For encrypting the 0^{th} row and 0^{th} column data in the matrix we take the k1 from the same row and column of the expanded keys key_{10} and the k2 from key_{11} and the same format is used for encrypting the other data's in the matrix. Here for the first round we use the key $_{10}$ and key_{11} and for the second round we take the key k1 from key_{11} and k2 from key_{12} and the same process used up to the 11^{th} round, in the 12^{th} round we take k1 from key_{33} and k2 from key_{10}. Based on this method keys are selected for encryption process.

The encryption function E, for any given letter x is

$E(x) = (((k1+p) \bmod M) +k2) \bmod M$

Where modulus M is the size of the alphabet and k1 and k2 are the key of the cipher. In this cipher k1 is not need to be prime number.

The decryption function D is

$D(E(x)) = (((E(x) – k2) \bmod M) – k1) \bmod M$

The main strength of the encryption is in selecting the keys. Let's see how it works, the input of this cipher is the result of the transposition cipher, after getting the input, the modified affine cipher is applied to each data in the matrix and that data is covert in to another data. The following fig shows the result of the substitution.

A	**S**	**4**	**3**
8	**D**	**F**	**7**
4	**0**	**0**	**9**
2	**8**	**0**	**0**

→

Q	**F**	**14**	**14**
14	**K**	**L**	**12**
7	**6**	**13**	**19**
7	**17**	**7**	**3**

Fig.4 Substitution

C. Folding

After substitution the result is taken as input to the folding technique. Folding is one of the transposition cipher, just like the paper fold, the matrix is folded horizontally, vertically, and diagonally. This folding technique shuffles the data from one position to another position. The following figure shows the result of folding.

Q	**F**	**14**	**14**
14	**K**	**L**	**12**
7	**6**	**13**	**19**
7	**17**	**7**	**3**

→

3	**17**	**7**	**7**
12	**13**	**6**	**14**
19	**L**	**K**	**7**
14	**F**	**14**	**O**

Fig.5 Folding

D. Shifting

This is a simple shifting cipher which provides a simple way to encrypt and numbers by using 16- array element of numeric digits. Each element must contain the 26 numeric characters from 0 to 25. Each digit must appear only once in each element of the array. The digits can appear in any order you like.

The input of this cipher is the result of the folding cipher. In the shifting cipher the program replaces each digit of the number by its position within its array element. For decryption the position is given as an input based on the position the data is taken and that data is plaintext of the given cipher text.

This process is illustrated in the following figure.

I/p	Array element	O/p
3	**0 1 2 3 4 5 6 7 8 9 10…… 22 23 24 25**	**3**
17	**1 2 3 4 5 6 7 8 9 10 11 …….. 23 24 25 0**	**16**
7	**2 3 4 5 6 7 8 9 10 11 12…… 23 24 25 0 1**	**5**
•	•	•
•	•	•
•	•	•
F	**13 14 15 16 17 18 ……7 8 9 10 11 12**	**S**
14	**14 15 16 17 18 19 …… 9 10 11 12 13**	**0**
O	**15 16 17 18 19 20 …….. 10 11 12 13 14**	**Z**

Fig.6 Shifting

Thus the input data is encrypted by using TSFS algorithm, the above encryption process is the result of the first round of the algorithm. The output of the first round goes input to the second round and the output of the second round goes input to the third round. This process continue up to the 12^{th} round, the output of this round is the cipher text of the given plain text and that cipher text is stored in the database. The decryption algorithm is the reverse of the encryption. The details are omitted, as it is fairly easy to derive. Fig.6 shows the entire encryption process.

VI. STRENGTH OF TSFS ALGORITHM

The number of keys is more so that the key combination increases which makes guessing of keys harder. The main strength of the algorithm is in the substitution transformation because selecting the key for finding the cipher gave more security to the data. In this algorithm the numeric plaintext have numeric cipher text, character plaintext have character cipher text and if the input data is alphanumeric type then the output cipher text also in alphanumeric, so there is no need for change the data field type when the encrypted data are stored in the database. Only the sensitive data in the database are encrypted so that for getting one or two data from the database, it is not need to decrypt all the data's in the database. Thus the proposed technique increases the speed of executing the queries in the database.

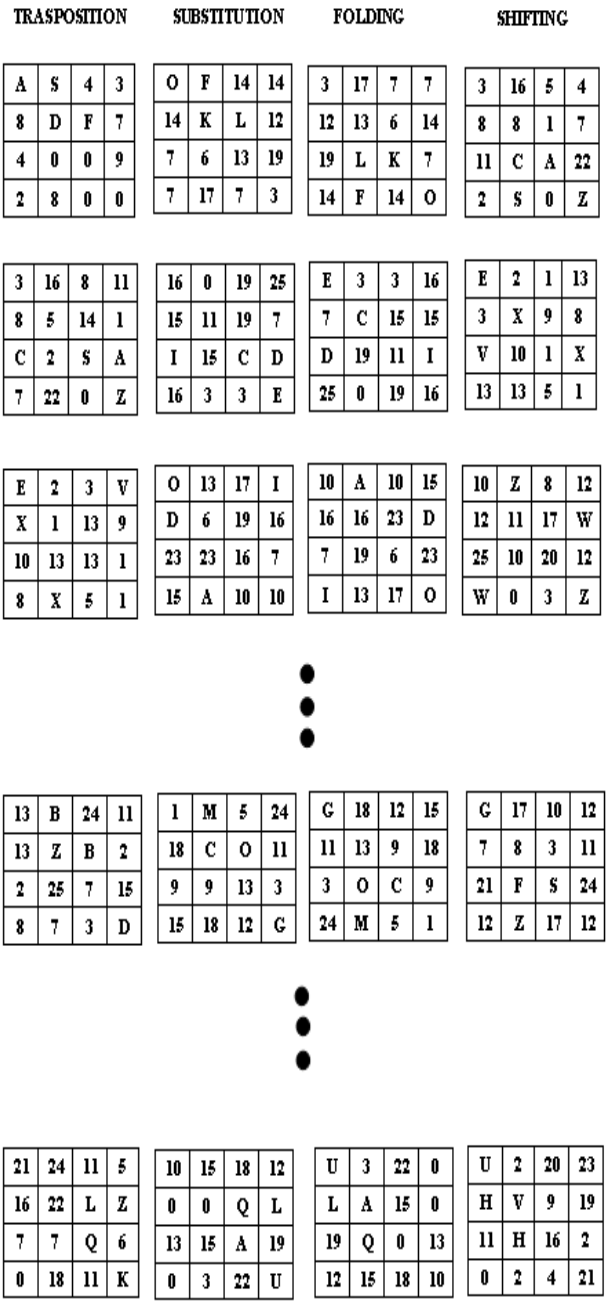

Fig. 6 sample output of the algorithm

VII. SECURITY ANALYSIS

Generally there are two kinds of attacks such as passive attack and active attack. Statistical attack comes under passive group. The cipher index value is computed with the help of TSFS algorithm and stored. It is too difficult to recover the data from the cipher index value by applying statistical attack. The benefit of the proposed scheme in terms of security is the data type of plain text and cipher text is same. It is difficult to analyze the recovered data by the attacker, it's encrypted or not. For example the credit card number of the customer in bank sector is encrypted by using TSFS algorithm is shown in the following figure.

Before Encryption	After Encryption
2546321756985412	8754123659874521

Fig.7 credit card encryption

Even though the adversary get the authorization to read the data in the database, he cannot identify whether the credit card number is encrypted or not. Another main strength of the proposed scheme is the number of keys are more and hence the key combination increases to 10^{64} which makes guessing of keys harder to the attacker. For the key expansion technique, the various keys and their values in the rows were plotted against randomness and key ranges are shown in the graph. The following graph depicts the variation of values in the seed key and their sub keys. Here we take a sample of key1 and their sub keys k_{10}, k_{11}, k_{12}, k_{13}. For each round two sub keys are used in the substitution transformation. k_{10} and k_{11} used in the first round, k_{11} and k_{12} used in the second round, k_{12} and k_{13} used in the third round and the continuous pair of keys are used for each round. Variations between values of rows in keys provide more security for the data.

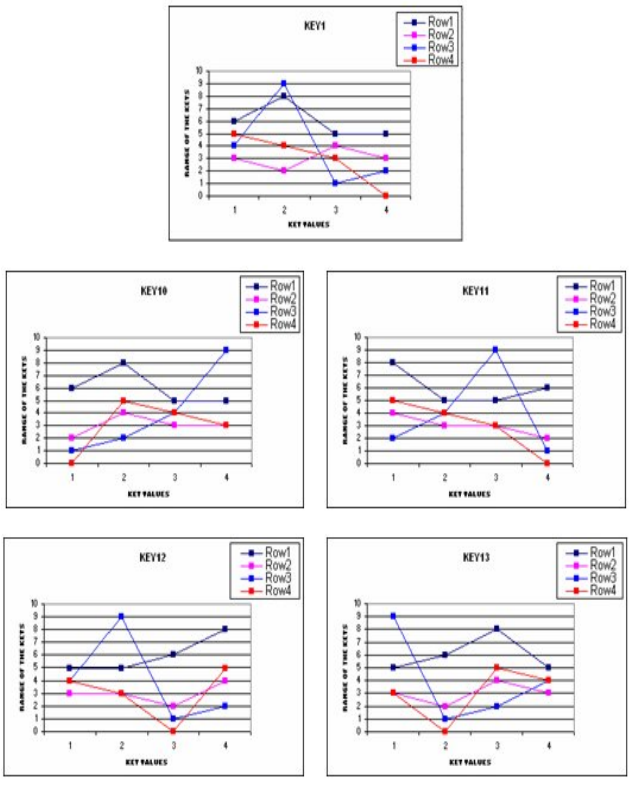

Fig.8 variation of keys

VIII. CONCLUSION

Database attacks are increases in the risks of data disclosure. Many organizations must deal with legislation and regulation on data privacy. In this environment, our security planning must include a strategy for protecting sensitive databases against attacker. In this paper we propose a new light-weight and effective database encryption algorithm for encrypting the sensitive data that reside in the database. If sensitive data are encrypted before storage in the database, risks from security leaks can be eliminated and the security issues of the database will reduce by using three cryptographic keys for protecting the data. Our proposed scheme is considered as efficient because it provides maximum security to the database and also increases the process of encryption and decryption.

The proposed algorithm can be implemented for securing any corporate related accounting information which contains numeric data, decimal point numeric data and alpha numeric data but it does not work with any symbols. Suppose if we want to encrypt the user email id, it is not possible to encrypt by using this algorithm. So it is possible to enhance this algorithm for encrypting the symbols and other special characters.

REFERENCES

[1] Behrouz A.Forowuzan. *Applied Cryptography and Network Security*, Tata McGraw Hill.

[2] "*Database Encryption in Oracle9i*" , An Oracle Technical White Paper, February 2001.

[3] SK Bhatnagar, "*Securing Data-At-Rest*", Literature by Tata Consultancy Services.

[4] Min-Shiang Hwang, and Wei-pang. Yang ., "*Multilevel Secure Database Encryption With Subkeys*", Data and Knowledge Engineering, 1997.

[5] M.S. Hwang and W.P. Yang, " *A Two-Phase Encryption Scheme for Enchancing Database Security*",J.Systems and Software, 1995.

[6] L.Bouganim and P.Puncheral, "*Chip-Secured data access*: Confidential data on untrusted servers", Proc. Of the 28th International Conference on Very Large Data Bases, Hong Kong, china, 2002.

[7] R.L.Rivest., L. Adleman. "*On Data Banks and Privacy Homomorphisms*" computer and security.1993.

[8] Rakesh Agrawal Jerry Kiernan Ramakrishnan Srikant Yirong Xu, "*Order Preserving Encryption for Numberic Data*",IBM Almanden Research Center

[9] Tingjian Ge, Stan Zdonik, " *Fast,Secure Encryption for Indexing in a Column-Oriented DBMS*", IEEE 2007.

[10] Kamaljit Kaur, K.S Dhindsa, Ghanaya Singh, "*Numeric To Numeric Encryption of Database: using 3Kdec Algorithm"*, IEEE International Advance Computing Conference, Partial, India, 2009.

[11] John H.Mathews, " *A Simple Encryption/Decryption Algorithm for Numbers*", 2004