Securing Numerical Data Using Image RGB Values

[1]Bhavesh V. Bangera, [2]Dr. Anupkumar Palsokar, [3]Prof. Pankaj Raibagkar

Abstract— **- In today's world, it has become necessary for all kinds of organisations, to store the entire data related to their business transactions in databases secured databases. Data, while being moved from one place to another on a network, can easily be intercepted by an Intruder. Also, there have been instances, wherein, an attacker gets hold of the database and holds the organisation to ransom. In such cases, the unencrypted data in the database is far from secure and especially if it is stored in plain text form. This increases the chances of data leaks and any intruder getting an access to the data may exploit the same causing loss to an organisation. This paper proposes a novel approach for encrypting numerical data stored in the database by using the RGB values of certain pixels from an image related to the data. This approach also overcomes the threat of an intruder exploiting the data in case he gets hold of the entire data.**

***Index Terms*—Secured Numerical Data, Image RGB, Encryption Algorithm using Image, Decryption Algorithm using Image.**

I. INTRODUCTION

Enterprise wide software applications are being extensively used by business houses for executing various business processes today. The use of such systems becomes more relevant when the business operations are not restricted to any one geographic location. The data generated out of the regular business transactions is humongous and cannot be stored in a single database and neither at a single location. These distributed applications and databases are connected together using public infrastructure i.e. internet. For an organisation, even the smallest data hack or leak can cause tangible or intangible losses. Security of the databases thus is of paramount concern to business organisations [1, 2, 3, 4]].

If we consider the Human Resource (HR) Database relating to the employees of an organisation, it contains a lot of sensitive and private data pertaining to the employees and the organisation in general. Most of the data in the HR database is numerical in nature such as Salary Details, Date of Joining, Date of Birth, Age, Height, Weight, Bank account number, Social security number/s etc. In today's networked world, the possibility of databases being hacked [5, 6, 7, 8] by Internal as well as External intruders is always high. If an intruder gets an access to this sensitive data, he can easily manipulate it or misuse it. This leads to growing demand for securing the databases. Apart from securing their networks, organisations today are actively engaged in securing the databases as well. Encrypting the data in the database is one such common measure. Many standard methodologies exist for encrypting the text and numbers stored in the database. Anyone with the knowledge of the standard encryption algorithms can decrypt the data. There is a need of innovative techniques for encrypting data elements in a database such that it becomes difficult for an intruder to decipher the values of data items. This paper aims to develop a simple yet robust approach for encrypting the numerical data items in a database by using the RGB values of select pixels from an image related to that dataset. For example, in an employee database the value of the salary earned by an employee can be encrypted by perform a mathematical transformation on the salary value by using the RGB values of a predefined pixel of the respective employee's image stored in the same database. Thus, making it difficult for an intruder, to decrypt the respective values. Moreover, since the secret key used in this encryption approach is drawn from some related image to that database entry it becomes difficult to identify a common thread from the encrypted values.

II. BACKGROUND

Many approaches have been implemented earlier to secure the data residing in the database. Samba Sesay, Zongkai Yang, Jingwen Chen and Du Xu in their paper have proposed an encryption method for encrypting sensitive data by dividing the data according to the access roles provided to the user [9]. It states that there are three types of data residing in the database namely Sensitive Data , Non – Sensitive Data and Private Data .Sensitive data like salary details of employees has restricted access i.e. it can be viewed by Managers and no one else. Non – Sensitive Data is not sensitive and can be accessed by all. Private Data includes details like card numbers which can only be

accessed by Users of that Card. This approach basically encrypts only the Sensitive data. Advantage of this approach is that only a single key is required to decrypt the entire column of Encrypted Sensitive Data and the only disadvantage of this approach is that queries like Sum, Average, Count and other statistical functions cannot be performed directly on the encrypted data. Ah Kioon, Mary Cindy, Zhao Shun Wang, and Shubra Deb Das in their paper have focused on securing the passwords in database using hashing functions [10]. MD5 hashing algorithm is preferred to encrypt the password. This paper also discusses about how MD5 algorithm are prone to different attacks like dictionary attacks and Rainbow tables. In dictionary attacks, an attacker tries to hash each password from the dictionary and then a binary search is performed on the hashed passwords. Rainbow table consist of hash chain and are used for reversing the hash functions. An Improved MD5 Algorithm was developed which includes :- Generation of random key which is stored in database, transformation of original password into complex password using columnar transposition cipher, creation of salt, generation of additional random string of 128 bits, hashing of password using key stretching, and finally storage of hashed password in the database. Advantage of this approach is that it is very easy to implement. The main disadvantage of this approach was that it can only be used for encryption, hence decryption is not possible. Moreover, more time is utilised for the computation. Xing-hui, Wu, and Ming Xiu-jun had proposed a hybrid approach which uses two encryption algorithms i.e. RSA and IDEA for securing the database [11]. The Encryption and Decryption in the data communication process was achieved using IDEA Algorithm. The public key of RSA algorithm is used for the encryption of key of the IDEA algorithm. This approach is very secure as it uses two algorithms for security, but at the same time it takes lot of time for computations. Min-Shiang Hwang and Wei-Pang Yang in their paper, have proposed a two-phase encryption algorithm for securing the database systems [12]. Two algorithms for cryptographic relational algebra were developed in the database systems. Two simple methods of solving the key management problem in the subkey scheme were also introduced. T. Ge and S. Zdonik in their paper , introduced a New Encryption Algorithm " Fast Comparison Encryption (FCE)" for data Warehouses. This approach was really efficient as it was used for encryption of large databases .With low decryption overheads the comparison of cipher texts was very fast [13]. Liu, Lianzhong, and Jingfen Gai proposed a new Encryption method for database alongwith Efficient Key Management . This approach stated that the keys were saved away from the encrypted data , so as to increase the level of security [14].

The review of literature reveals that most of the techniques proposed by researchers focus on encrypting the data without addressing the issue of decryption. In such a case separate transformations are required for decrypting the data while displaying it to the user through an appropriate front-end. In order to make the database more secure, the computing time was compromised with. Generation of a key for encryption was big concern and was addressed using complex transformations. There is a dearth of techniques which encrypts sensitive numeric data by utilising minimum computing resources and in a short span of time [15].

III. METHODOLOGY

Based on the above shortcomings, a new method was developed overcoming most of the issues that the previous approaches had. The Approach mentioned in this paper, particularly focuses on encrypting the numeric data in the database using selected RGB values taken from a set pixel of an image associated with the respective record. This approach consists of two processes (Encryption and Decryption). The algorithms for the two processes included in this approach are discussed herewith:

A. Algorithm for Encrypting the Numeric Data in Database

1) Fetch the Data (Numeric Data and Imagepath) from the User Interface or from the table in the database.
2) Store the Numeric Data and Imagepath in cell Array Form. Convert the following cell array data into ordinary array using cell2mat() function and store the result in a variable.
3) Fetch the image from the Imagepath using imread() function and store it in variable named 'f'.
4) Variable 'f' contains 3 matrices i.e. Red , Green and Blue Matrix of image. Values in these matrices are in Unsigned 8-bit Integer Form.
5) Convert all the three Unsigned 8-bit integer matrices to double using im2double() function and store it in variable 'f2'. The values of newly Generated (Red , Green , Blue) matrices are in double Form.
6) Store Newly Generated Red matrix in Red variable , Green matrix in Green variable , Blue matrix in Blue variable.
7) Calculate Average of 3 variables (Red , Green and Blue) and store the result in Matrix variable 'avg'. Thus 'avg' variable contains matrix of values (These values are a result of the average of Red , Green and Blue Matrix).
8) Select points from the Matrix variable 'avg' say :- avg(5,1) , avg(2,2) , avg(2,3) , avg(5,4). These points depend on number of numeric data available (Eg :- If 4 numeric data is to be encrypted, then select 4 points from the matrix variable) . The points thus selected are the same points which will be used for decryption.
9) Store the points selected in different variables . Lets say :- num1 = avg(5,1) and so on.
10) Divide the Variable (in which value of numeric data is stored) with the points selected from matrix

variable and store the result in the cell Array. Eg :- num(1,1) = {sal/num1}

11) Add the cell Array to datainsert() function . Eg :- datainsert(conn , tablename , colnames , num)
Here , conn = Database Connection Object
tablename = Name of the table where encrypted Records need to be inserted
colnames = Column names of the table mentioned in datainsert() function
num = name of the Cell Array

B. The Encryption process.

The encryption process transforms the data into an encrypted form and stores it in the database. For the purpose of encrypting the data two scenarios have been discussed:

1) Encryption of already Existing Records

If the Database already consists of a large number of records, then encrypting of one record at a time will be a time consuming job. This approach states that, even though there exist huge number of records in the database, all of them can be encrypted in one go.A sample database before encryption is shown in Table 1. MS-SQL Server was used to store the database. The "freshtest" table shown in Table 1 consists of 5 records.

Id	Salary	Age	Height	Weight	imgpath
1	10000	24	167	80	C:\1.bmp
2	14356	21	177	88	C:\2.bmp
3	32456	19	187	72	C:\3.bmp
4	67458	32	158	54	C:\4.bmp
5	34982	54	198	81	C:\5.bmp

Table 1. Original Data in the database

The above data passes through the proposed Encryption Process, and the encrypted values are stored back into the encrypted table "instest" as shown in Table 2.

Id	Salary	Age	Height	Weight	imgpath
1	127500	83	572	1073	C:\1.bmp
2	183039	73	607	1181	C:\2.bmp
3	413814	66	641	966	C:\3.bmp
4	860089	111	542	724	C:\4.bmp
5	446020	187	679	1087	C:\5.bmp

Table 2. The decrypted table "instest"

2) Encryption of New Records

The implementation of the proposed encryption also works seamlessly when the database has no previous records and new records are to be entered through the front-end. In this case the data is collected from by designing a user interface and the data is encrypted by the frontend before storing it in the database. Records are encrypted as and when they are entered. The embedded logic in the front-end decrypts the data while retrieving the data from the database at runtime, thus avoiding the time required for decrypting the entire database.

C. Algorithm for Decrypting Numeric Data from the Database

The general process of decryption for the entire database is presented below.

1) Get the ID from the User Interface.
2) Fetch the Encrypted Data (Numeric Data and Imagepath) for that particular ID from the table in the database
3) Numeric Data and Imagepath data, fetched , will be in cell Array Form. Convert the following cell array data into ordinary array using cell2mat() function and store the result in variable.
4) Fetch the image from the Imagepath using imread() function and store it in variable named 'f'.
5) Variable 'f' contains 3 matrices i.e. Red , Green and Blue Matrix of image. Values in this matrices are in Unsigned 8-bit Integer Form.
6) Convert all the three Unsigned 8-bit integer matrices to double using im2double() function and store it in variable 'f2'. The values of newly Generated (Red , Green , Blue) matrices are in double Form.
7) Store Newly Generated Red matrix in Red variable , Green matrix in Green variable , Blue matrix in Blue variable.
8) Calculate Average of 3 variables (Red , Green and Blue) and store the result in variable 'avg'.
9) Get same points (which were used for Encryption) from the Matrix variable 'avg' say :- avg(5,1) , avg(2,2) , avg(2,3) , avg(5,4). This points depend on number of numeric data available (Eg :- If 4 numeric data are available , then select 4 random points from matrix variable)
10) Store the points selected in different variables . Lets say :- num1 = avg(5,1)
11) Multiply the Variable (in which value of numeric data is stored) with the points selected from matrix variable.
12) Round off the result thus obtained and store it in the variable.
13) Display the variable values (Decrypted Data) in respective textboxes.

In a system where this proposed encryption method is used, decryption methodology needs to be embedded while retrieving the records form the database through a front-end, to be presented to the user. While retrieving the actual data from the database, the data has to pass through decryption process before being displayed to the user. The User has to provide some identifier of the record he wants to read. When user inputs the identifier, a code embedded in the front-end retrieves the encrypted data, decrypts it presents it to the user.

IV. IMPLEMENTATION

The approach used for implementing this RGB values based encryption method proposed here is elaborated in the following steps:

A. Encryption:

1. Fetching the Data (Numeric Data as well as Imagepath Data) from the User Interface and storing it in variables.

Assuming that there are four numeric records say Salary , Age , Height , Weight and one Imagepath (Image of Employee). Suppose the inputs are :- 10000(Salary) , 24(Age) , 167(Height) , 80(Weight) , 'C:\Users\Desktop\1.bmp'(Imagepath). We store this values in variables in following way :-

sal = 10000
age = 24
height = 167
weight = 80
e = C:\Users\Desktop\1.bmp

2. Reading the image from the Imagepath and storing the values in the variable.

f = imread(e)

Here imread() function reads the image from the Imagepath and stores the RGB matrices of the image in the variable 'f'.

For the sake of simplicity in identifying the pixel and using its RGB values, the image considered for implementing this approach is considered to be of size 5 X 4. The value of variable 'f' will be 5x4x3 where 3 denotes the three matrices i.e. Red, Green and Blue which contains values in Unsigned 8-bit Integer Form. But for the commercial implementation of this approach, the image size would be of a bigger size.

Variable 'f' contains 3 matrices, we now store Red matrix in variable 'r' , Green matrix in variable 'g' and blue matrix in variable 'b'.

r = f(: ,: , 1)
g = f(: ,: , 2)
b = f(: ,: , 3)

In the image under consideration here, the Variable 'r', 'g and 'b' contains matrices as shown in table 3, 4 and 5 respectively.

252	122	157	249
255	105	96	255
255	160	167	255
216	143	143	187
10	104	136	9

Table 3. Values of the matrix of variable 'r'

253	125	160	251
255	162	65	255
255	109	128	255
220	130	131	191
11	103	135	10

Table 4. Values of the matrix of variable 'g'

255	136	172	255
255	53	62	255
255	103	128	255
235	138	143	211
39	130	155	38

Table 5. Values of the matrix of variable 'b'

3. Converting all the three Unsigned 8-bit integer RGB matrices to double.

Variable 'f' contains 3 matrices (RGB) which are in Unsigned 8-bit integer form. Now , by using im2double() function , we convert the values of RGB matrices from Unsigned 8-bit integer form to the double form and store them in a variable 'f2'. From the variable 'f2' we further create matrices 'red', 'green' and 'blue' containing the respective RGB values in double form. The sample red, blue and green matrices are given in table 6, 7 and 8 respectively.

0.9882	0.4784	0.6157	0.9765
1.0000	0.4118	0.3765	1.0000
1.0000	0.6275	0.6549	1.0000
0.8471	0.5608	0.5608	0.7333
0.0392	0.4078	0.5333	0.0353

Table 6. Values of the matrix of variable 'red'

0.9922	0.4902	0.6275	0.9843
1.0000	0.2431	0.2549	1.0000
1.0000	0.4275	0.5020	1.0000
0.8627	0.5098	0.5137	0.7490
0.0431	0.4039	0.5294	0.0392

Table 7. Values of the matrix of variable 'green'

1.0000	0.5333	0.6745	1.0000
1.0000	0.2078	0.2431	1.0000
1.0000	0.4039	0.5020	1.0000
0.9216	0.5412	0.5608	0.8275
0.1529	0.5098	0.6078	0.1490

Table 8. Values of the matrix of variable 'blue'

4. Calculate Average matrix

For the sake of generating the key values average of 3 variables (red , green and blue) is calculated and stored in the matrix 'avg' as shown in table 9 .

0.9935	0.5007	0.6392	0.9869
1.0000	0.2876	0.2915	1.0000
1.0000	0.4863	0.5529	1.0000
0.8771	0.5373	0.5451	0.7699
0.0784	0.4405	0.5569	0.0745

Table 9. Values of the matrix of variable 'avg'

5. Selecting points from the matrix

Four values are selected from the 'avg' matrix and are stored in variables num1 , num2 , num3 , num4 respectively.

num1 = double(avg(5,1))
num2 = double(avg(2,2))
num3 = double(avg(2,3))
num4 = double(avg(5,4))

Thus num1 = 0.0784 , num2 = 0.2876 , num3 = 0.2915 , num4 = 0.0745

6. Division of Variables

Variables num1, num2, num3 and num4 are used as divisors to divide the actual value of the data to be encrypted. The result is stored in a1, b1, c1 and d1.

a1 = sal / num1
b1 = age / num2
c1 = height / num3
d1 = weight / num4

Thus:
a1 = 10000 / 0.0784 = 127500 (rounded off)
b1 = 24 / 0.2876 = 83 (rounded off)
c1 = 167 / 0.2915 = 572 (rounded off)
d1 = 80 / 0.0745 = 1073 (rounded off)

7. Creating the Cell Array

Further for the sake of entering these encrypted values a cell Array named "data" is created, which contains the values of variables a1 , b1 , c1 , d1 and e(Imagepath).

data = { a1 b1 c1 d1 e }

8. Insertion of Cell Array data into the database

Once the Variable values to be inserted to the database are stored in the Cell Array, the values are inserted into the database at appropriate positions. For inserting the data into the database, datainsert() function is used after creating appropriate connections as shown below :-

datainsert(conn , tablename , colnames , data)

Here conn is the database Connection object, tablename is the name of the table in which we are going to insert the record, colnames are the names of the columns of that table and
data is the name of the cell Array which contains variable values that need to be inserted into the table.

B. Decryption:

1. Getting the ID

The records are stored in the database in the Encrypted Form. Now in order to view the original record, the user needs to provide the ID of that encrypted record

2. Fetch the Encrypted Data

After the User has provided the ID, the encrypted data is fetched from the database and stored in a Cell Array. If ID provided by user is say 37, then the encrypted record of that particular ID is fetched.

x = curs.Data

'curs' is an object , which is fetched from the database, and values of that object are stored in cell Array named 'x'.

3. Conversion From Cell Array to ordinary Array

The data in the variable 'x' are in cell array format. In order to view the data in each cell , cell2mat() function is used . The data of the cell, thus fetched, are stored in respective variables.

```
sal = cell2mat(x(1,1))
age = cell2mat(x(1,2))
height = cell2mat(x(1,3))
weight = cell2mat(x(1,4))
img = cell2mat(x(1,5))
```

Suppose for ID valued 37 , the encrypted record is :- 127500 , 83 , 572 , 1073 and 'C:\Users\Desktop\1.bmp' , then variables will be assigned with following values :-

```
sal = 127500
age = 83
height = 572
weight = 1073
img = C:\Users\Desktop\1.bmp
```

4. Reading the image from the Imagepath and storing the values in the variable.

```
f=imread(img)
```

'imread()' function is used to read the image from the Imagepath and stores the RGB matrices of the image in the variable 'f'. The variable 'f' which contains 3 matrices are then splitted into three to store the RGB separately as shown in Table 10, 11 and 12 respectively.

```
r = f(: ,: , 1)    g = f(: ,: , 2)    b = f(: ,: , 3)
```

252	122	157	249
255	105	96	255
255	160	167	255
216	143	143	187
10	104	136	9

Table 10. Values of the matrix of variable 'r'

253	125	160	251
255	162	65	255
255	109	128	255
220	130	131	191
11	103	135	10

Table 11. Values of the matrix of variable 'g'

255	136	172	255
255	53	62	255
255	103	128	255
235	138	143	211
39	130	155	38

Table 12. Values of the matrix of variable 'b'

5. Converting all the three Unsigned 8-bit integer RGB matrices to double.

Variable 'f' which contains the 3 matrices (RGB) and contain unsigned 8-bit integer values are converted to double form and are further stored in red, green and blue matrices separately. The red, green and blue matrices in double form are shown in Table 13, 14 and 15 respectively.

0.9882	0.4784	0.6157	0.9765
1.0000	0.4118	0.3765	1.0000
1.0000	0.6275	0.6549	1.0000
0.8471	0.5608	0.5608	0.7333
0.0392	0.4078	0.5333	0.0353

Table 13. Variable 'red' with double values

0.9922	0.4902	0.6275	0.9843
1.0000	0.2431	0.2549	1.0000
1.0000	0.4275	0.5020	1.0000
0.8627	0.5098	0.5137	0.7490
0.0431	0.4039	0.5294	0.0392

Table 14. Variable 'green' with double values

1.0000	0.5333	0.6745	1.0000
1.0000	0.2078	0.2431	1.0000
1.0000	0.4039	0.5020	1.0000
0.9216	0.5412	0.5608	0.8275
0.1529	0.5098	0.6078	0.1490

Table 15. Variable 'blue' with double values

6. Calculating the Average of 3 variables (red , green and blue) .

The average of all the three matrices is done and is represented in table 16.

0.9935	0.5007	0.6392	0.9869
1.0000	0.2876	0.2915	1.0000
1.0000	0.4863	0.5529	1.0000
0.8771	0.5373	0.5451	0.7699
0.0784	0.4405	0.5569	0.0745

Table 16 . Average of RGB in double form

7. Selecting the pixels for identifying key values

Four predefined values from the 'avg' matrix as discussed earlier in step 5 of Encryption process, are selected and are stored in four variables.
num1 = double(avg(5,1)) = 0.0784
num2 = double(avg(2,2)) = 0.2876
num3 = double(avg(2,3)) = 0.2915
num4 = double(avg(5,4)) = 0.0745

8. Multiplication of Variables

The values thus received from step 7 which are in the encrypted form are used to decrypt the actual data values and are stored in the cell array. In order to view each value of cell array , cell2mat() function is used , and the result is rounded off and stored in new variables .
num(1,1) = { sal * num1 }
Thus num(1,1) = 127500 * 0.0784 = 10000

num(1,2) = { age * num2 }
Thus num(1,2) = 83 * 0.2876 = 23.8693

num(1,3) = { height * num3 }
Thus num(1,3) = 572 * 0.2915 = 166.7399

num(1,4) = { weight * num4 }
Thus num(1,4) = 1073 * 0.0745 = 79.9490

num(1,5) = x(1,5)
Thus num(1,5) = C:\Users\Desktop\1.bmp
Here 'num' is the cell array

a1 = round(cell2mat(num(1,1)))
Thus a1 = round(cell2mat(10000))) = 10000

b1 = round(cell2mat(num(1,2)))
Thus b1 = round(cell2mat(23.8693))) = 24

c1 = round(cell2mat(num(1,3)))
Thus c1 = round(cell2mat(166.7399))) = 167

d1 = round(cell2mat(num(1,4)))
Thus d1 = round(cell2mat(79.9490))) = 80

9. Displaying the Decrypted values

Once all the calculations have been performed, the decrypted values are displayed to the used through the front-end..

V. RESULTS AND DISCUSSIONS

Since the approach proposed in this paper is novel, it is evaluated on the basis of various parameters to find out its suitability for implementation on a larger scale.

A. Time Analysis

Average Time required for the entire process in databases containing 100, 500 and 1000 records is presented in Table 17. Table 18 presents the time required to encrypt and decrypt a single record in the database.

Iterations	Time taken For Encryption of 100 records (in seconds)	Time taken For Encryption of 500 records (in seconds)	Time taken For Encryption of 1000 records (in seconds)
1	0.674621	2.276870	4.131809
2	0.622516	2.056788	4.026600
3	0.620350	2.292837	4.297068
4	0.654092	2.135933	4.385021
5	0.607383	2.066223	4.330505
6	0.592070	2.261968	4.233992
7	0.615968	2.133649	3.858097
8	0.600832	2.227350	4.367666
9	0.613859	2.218652	4.214694
10	0.588499	2.271137	3.891946
Average Time (in seconds)	**0.619019**	**2.1941407**	**4.1737398**

Table 17. Time required to encrypt records in databases containing 100, 500 and 1000 records

Iterations	Time taken For Encryption of 1 record (in seconds)	Time taken For Decryption of 1 record (in seconds)
1	0.427874	0.350771
2	0.361635	0.336608
3	0.341541	0.335702
4	0.339781	0.340554
5	0.341828	0.346190
6	0.321059	0.317228
7	0.405827	0.338052
8	0.342195	0.351366
9	0.324881	0.350124
10	0.355182	0.316900
Average Time (in seconds)	**0.3561803**	**0.3383495**

Table 18. Encryption and Decryption of single record

B. Space Analysis

Memory is the most expensive resource at runtime. The memory requirement for storing the original and the encrypted database with 100, 500 and 1000 records is presented in Table 19.

	Records		
Size Occupied by Table	**100**	**500**	**1000**
Original Records	16kb	64kb	104kb
Encrypted Records	16kb	64kb	104kb

Table 19. Memory requirements

VI. CONCLUSION

The proposed encryption approach using RGB values from an image helps in providing an effective security mechanism to encrypt the numeric values in a database. This approach guarantees that the numeric records in the database are secured and in encrypted form and would not be visible to an intruder in its original form. The implementation of the proposed approach is carried out using Matlab as the primary tool and Microsoft SQL Server as the database. PHP was used to prepare the user interface for testing the approach. Although the approach discussed in this paper focused mainly on an employee database as an example, which contains numeric values like Salary , Height , Weight , age , date of birth , tax paid ,insurance etc , It can be applied in other databases maintained by business organisations in the banking, healthcare, government and the defence sector. Encryption of character data and randomisation of the pixel positions used for determining the key values are identified as avenues of further research.

REFERENCES

[1] Zixin Wang , Guoyuan Lin. “Computer network database of attack and defense.” In 2011 International Conference on Consumer Electronics , Communications and Networks (CECNet) , Pages :3986 – 3989 . IEEE , 2011.
[2] WebCohort. WebCohort`s application defense center reports results of vulnerability testing on Web applications [EB/OL].March 2004.
[3] Xiaolei Huang. SQL injection attacks and defense. Tsinghua University Press, 2010.
[4] Biao Meng, Junjing Liu. SQL injection attacks classified defense model [J]. Information Technology & Standardization, 2008.
[5] Nedhal A. Al-Sayid , Dana Aldlaeen. “Database security threats : A survey study” In 5th International Conference on Computer Science and Information Technology , Pages : 60 – 64 . IEEE , 2013.
[6] Baker,W. H., Hutton, A., Hylender, C. D., Novak, C., Porter, C., Sartin, B., Tippett, P., & Valentine, J. A. (2009). The 2009 data breach investigations report. Verizon Business. Retrieved January 31, 2010.
[7] Ernst and Young Global Limited, Data loss prevention: Keeping your sensitive data out of the public domain, Insights on governance, risk and compliance, EYGM Limited, Sep., 2011.
[8] Georgiev, M., Iyengar, S., Jana, S., Anubhai, R., Boneh, D., and Shmatikov, V., The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software, ACM, 2012.
[9] Sesay, Samba, Zongkai Yang, Jingwen Chen, and Du Xu. "A secure database encryption scheme." In Consumer Communications and Networking Conference, 2005. CCNC. 2005 Second IEEE, pp. 49-53. IEEE, 2005.
[10] Ah Kioon, Mary Cindy, Zhao Shun Wang, and Shubra Deb Das. "Security Analysis of MD5 algorithm in Password Storage." Applied Mechanics and Materials 347, Pages: 2706-2711, (2013).
[11] Xing-hui, Wu, and Ming Xiu-jun. "Research of the Database Encryption Technique Based on Hybrid Cryptography." In Computational Intelligence and Design (ISCID), 2010 International Symposium on, vol. 2, pp. 68-71. IEEE, 2010.
[12] Min-Shiang Hwang, Wei-Pang Yang, “A two-phase encryption scheme for enhancing database security”, Journal of Systems and Software, Volume 31, Issue 3 , Pages 257-265, December 1995.
[13] T. Ge and S. Zdonik, “Fast, secure encryption for indexing in a column oriented DBMS,” in International Conference on Data Engineering - ICDE 2007. pp. 676–685. IEEE, 2007.
[14] Liu, Lianzhong, and Jingfen Gai. "A new lightweight database encryption scheme transparent to applications." In Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on, pp. 135-140. IEEE, 2008.
[15] Kaur, Kamaljit, K. S. Dhindsa, and Ghanaya Singh. "Numeric to Numeric Encryption of Databases: Using 3Kdec Algorithm." In Advance Computing Conference, 2009. IACC 2009. IEEE International, pp. 1501-1505. IEEE, 2009.

***Bhavesh V. Bangera**, MCA Student Department of Computer Applications, SIES College of Management Studies, Nerul, Navi Mumbai, India.*

***Dr. Anupkumar Palsokar**, Associate Professor Department of Computer Applications, SIES College of Management Studies, Nerul, Navi Mumbai, India.*

***Prof. Pankaj Raibagkar**, Associate Professor Department of Computer Applications, SIES College of Management Studies, Nerul, Navi Mumbai, India.*