ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

PRESERVING DATABASE CONFIDENTIALITY USING MODIFIED USER SUPPLIED KEY BASED ENCRYPTION

Surya Pratap Singh*, Upendra Nath Tripathi

[1,2]Dept. of Computer science, Deen Dayal Upadhyay Gorakhpur University, Gorakhpur, India

DOI: 10.5281/zenodo.192615

ABSTRACT

The security issues in database are growing very fast as the attackers and hackers are trying everything to compromise the security of the database. The confidentiality of the database is one of the most important aspects that needed to be maintained. One of the most effective ways to maintain the database confidentiality is Encryption. One method to achieve the standard encryption is user supplied key based encryption because this encryption technique provides the cipher text whose length is different in comparison to plain text. But it have one major shortcoming one plain text character is converted into same cipher text for its entire occurrence.

In this paper we proposed the modified user supplied key based encryption in which we have all the benefits of user supplied key based encryption and at the same time it provides different characters in cipher text for different occurrence of the same plain text character.

KEYWORDS: Database Security, User Supplied Key based Encryption, Modified User Supplied Key based Encryption, Database Confidentiality, plain text, Cipher text.

INTRODUCTION

The term Cryptography came from the Greek word “kryptos,” meaning hidden, and “graphia,” meaning writing. Cryptography is a science that applies complex mathematics and logic to design strong encryption methods. Achieving strong encryption, the hiding of data’s meaning, also requires intuitive leaps that allow creative application of known or new methods. Cryptography involves developing, testing, and studying the science of encryption methods.

WHEN TO ENCRYPT?

Encrypt data when it moves

Data moving from one trust zone to another, whether within your organization or between you and an external network, is at high risk of interception. Encrypt it.

Data moving from trusted portions of the network to end-user devices over wireless LANs almost always at high risk. Encrypt it.

Encrypt when access control cannot be assured

For flat file storage, encrypting a spreadsheet file in a department share provides an additional layer of separation. Only employees with the right authorization have access.

Application access controls protecting databases often do not provide granular control to strictly enforce need-to-know or least privilege. Using database solutions with field- and row-level encryption capabilities can help strengthen weak application-level access controls.

Encrypt when it is required

And then there are local, state, and federal regulations. Couple regulatory constraints with auditor insistence, and you often find yourself encrypting because you have to. This type of encryption is often based on

generalizations instead of existing security context. For example, just because you encrypt protected health information does not mean it is secure enough… but it satisfies HIPAA requirements.

Encrypt when it is a reasonable and appropriate method of reducing risk in a given situation

This law is actually a summary of the previous three. After performing a risk assessment, if you believe risk is too high because existing controls do not adequately protect sensitive information, encrypt. This applies to risk from attacks or non-compliance.

ENCRYPTION ALGORITHMS

The standard encryption algorithm are described below-

Name	Description
RSA Encryption	The RSA encryption module uses the RSA Security, Inc. RC4 encryption algorithm. Oracle's optimized implementation provides a high degree of security for a minimal performance penalty. For the RC4 algorithm, Oracle provides encryption key lengths of 40-bits, 56-bits, 128-bits, and 256-bits.
DES Encryption	Oracle Advanced Security implements DES with a standard, optimized 56-bit key encryption algorithm, and also provides DES40, a 40-bit version, for backward compatibility.
Triple-DES Encryption	Oracle Advanced Security also supports Triple-DES encryption (3DES), which is available in two-key and three-key versions, with effective key lengths of 112-bits and 168-bits, respectively. Both versions operate in outer Cipher Block Chaining (CBC) mode.

Table 1: Encryption Methods

USER SUPPLIED KEY BASED ENCRYPTION

This approach works by taking 128 bit key provided by the user. We declare an integer variable which stores the key on the basis of which the database contents is encrypted the proposed method works in the following two steps-

Encryption

In this step first the user provide the authorization credentials to login to the database, after verifying the credentials user is allowed access to the database now user provides his chosen key to encrypt the desired content of the database. The encryption process works by the following algorithm.

Encryption Algorithm:

- In this algorithm following variables are assumed –
  int uk: to store the user supplied key
  char a, b : to store the plain and cipher text characters.
  int n: to store the positional equivalent of the character in plain text character variable a.
  int m: to store the positional equivalent of the character in cipher text variable b.
  int x,y and t to store the intermediate data in processing
- Now the Encryption algorithm works in following manner

Step 1 – read a character of the field in variable a.
Step 2 – now n = positional equivalent of a.
The positional equivalent are taken as follows
From a-z 1-26
From A-Z 27-52
From 0-9 and space 53-63
Step 3 – t = n + uk
Step 4 – x = t mod 63
y = t / 63
Step 5 – b = character equivalent of position in x.
Step 6 – store the char in variable b along with y in the field.
Step 7 – continue the step 1 to 6 till there is unprocessed character in the field.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

Decryption

In this step the valid user can decrypt the database content. The decryption algorithm works by the following algorithm.

Decryption Algorithm:

- In this algorithm following variables are assumed –

int uk: to store user supplied key
Char a : to store the character values in the field from 1,3,5,7……positions
int y: to store the converted character value in the field from 2,4,6,8,…… positions.
int m: to store positional equivalent of char in a
int n : to store the final positional equivalent
int t: to store the processing values.

- Now the decryption algorithm works in following manner

Step 1 – read first character of the field in variable a and second variable in y.
Step 2 – now m = positional equivalent of a
Step 3 – t = m + y * 63
Step 4 – n= t - uk
Step 5 – b = character equivalent of position in n.
Step 6 – store the char in variable b in the field.
Step 7 – continue the step 1 to 6 till there is unprocessed character in the field.

PROBLEMS WITH EXISTING USER SUPPLIED KEY BASED ENCRYPTION

The user supplied key based algorithm mentioned above have a serious problem when the plain text is passed for encryption it generates the same cipher text character. This problem is represented in the following fig 1.

Fig 1: User Supplied Key Based Encryption

In the above fig the character **'S'** is converted into **'g3'** and **'a'** is converted into **'o2'** for all occurrence. Hence by viewing the pattern of occurrence one can easily guess the repetition of the same character and can guess the character and break the cipher.

PROPOSED SOLUTION

To overcome from the above mentioned problem we proposed modified user supplied key based encryption, in this mechanism all occurrence of plane text is converted into different cipher text character. This mechanism also contains two steps Encryption and Decryption.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

MODIFIED USER SUPPLIED KEY BASED ENCRYPTION

Encryption Algorithm

To encrypt the plain text into cipher text we take the following variables –

- int val; to store user supplied key
  int sod; to store sum of digit of user supplied key
  char n; to store characters of plain text
  int sum ; to store the numeric equivalent of plain text character
  int val1,val2; to store numerator and denominator by 63 respectively.(note – we are dividing by 63 because 1-26 is taken for a-z , 27-52 is taken for A-Z, 53 is taken for space and 54-63 is taken for 0-9)
  int uk; to store a unique incremented value
  Now the encryption process works in following steps-

Step1: read the user supplied key in variable val.
Step 2: calculate some of digit and store it in variable sod.
sod=sod*5;
Step 3: initialize variable uk to one (i.e., uk=1)
Step 4: sum = sod + uk + getpos (n); {getpos() is used to find the position of character in the range of 1 - 63}
Step 5:val1=sum%63; val2=sum/63;
uk++;
Step 6: {append the value} getchar (val1), val2 in the final result set. (getchar() method is used to find the character by positional equivalent)
Example – let the value contend in some field of the database is **aaaa**. Then the algorithm works in the following manner –
Let val=1234567
Iteration 1-
sod = 28;
sod = sod*5;
= 28*5;
= 140;
uk=1;
sum=sod + uk + getpos(n);
sum=140+1+1; (because n='a' and positional equivalent of a=1)
sum=142;
val1=sum%63;
= 142 % 63;
= 16
val2= sum/63;
= 142/63
= 2
uk++ (i.e., uk =2)
Now the result would be getchar(16),2
i.e., p2
Iteration 2-
sum = sod +uk+getpos(n);
sum =140+2+1; (because n='a' and positional equivalent of a=1)
sum=143;
val1=sum%63;
= 143 % 63;
= 17
val2= sum/63;
= 143/63
= 2
uk++ (i.e., uk =3)
Now the result would be getchar(17),2
i.e., q2
Now in the similar way the algorithm encrypt the remaining characters. Now **"aaaa"** becomes **"p2q2r2s2"**

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

Decryption Algorithm

To decrypt the cipher text to find plain text we take the following variables –

- int val; to store user supplied key
  int sod; to store sum of digit of user supplied key
  sod=sod*5;
  char x; to store characters of odd position of Cipher text (i.e., 1,3,5,.....)
  int y; to store parsed result of even position of Cipher text (i.e., 2,4,6,.....)
  int sum ; to store the numeric equivalent of Cipher text character
  int int uk; to store a unique incremented value
  int i,j;
  Now the encryption process works in following steps

Step1: read the user supplied key in variable val. and initialize i=1;j=2;
Step 2: calculate some of digit and store it in variable sod.
Step 3: read ith character of cipher text in x.
Parse and store jth character of cipher text in y;
Step 4: sum=y*63+getpos(x); getpos() is used to find the position of character in the range of 1 - 63}
Step 5: sum=sum-sod-uk;
uk++;
i=i+2;
j=j+2;
Step 6: {now append the value} getchar (sum) in the final result set (getchar() method is used to find the character by positional equivalent)
Example – we explain it by the use of previous example let the cipher text be "**p2q2r2s2".** The decryption algorithm works in following manner
val=1234567
i=0,j=1;
Iteration 1-
sod = 28;
sod = sod*5;
= 28*5;
= 140;
uk=1;
x=p;
y=2;
sum = y * 63 + getpos(x);
= 2 * 63 + getpos('p')
= 126 + 16 (because positional equivalent of p is 16)
= 142
sum = sum – sod – uk
= 142 – 140 – 1
= 142 -141
= 1
uk++; (i.e., uk=2)
i = i + 2;
J = j + 2;
now the result is getchar(1), i.e., **a**
Iteration 2-
uk=1;
x=q;
y=2;
sum = y * 63 + getpos(x);
= 2 * 63 + getpos('q')
= 126 + 17 (because positional equivalent of q is 17)
= 143
sum = sum – sod – uk

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

= 143 – 140 – 2
= 143 -142
= 1
uk++; (i.e., uk=3)
i = i + 2;
j = j + 2;
now the result is getchar(1), i.e., **a**
Now in the similar way the algorithm decrypt the remaining characters. Now **"p2q2r2s2"** becomes **"aaaa"**

SIMULATION AND RESULT

We implemented our proposed encryption method by the use of JAVA Swing as front end and My SQL as back end. To encrypt the data the user has to select the table to be encrypted and then click on show button. To display the contents of the table. Now to encrypt the record user have to enter user supplied key in this case 1234567 and then press Encrypt and Show button. The contents of table get encrypted and then displayed. It is represented in following fig 2.

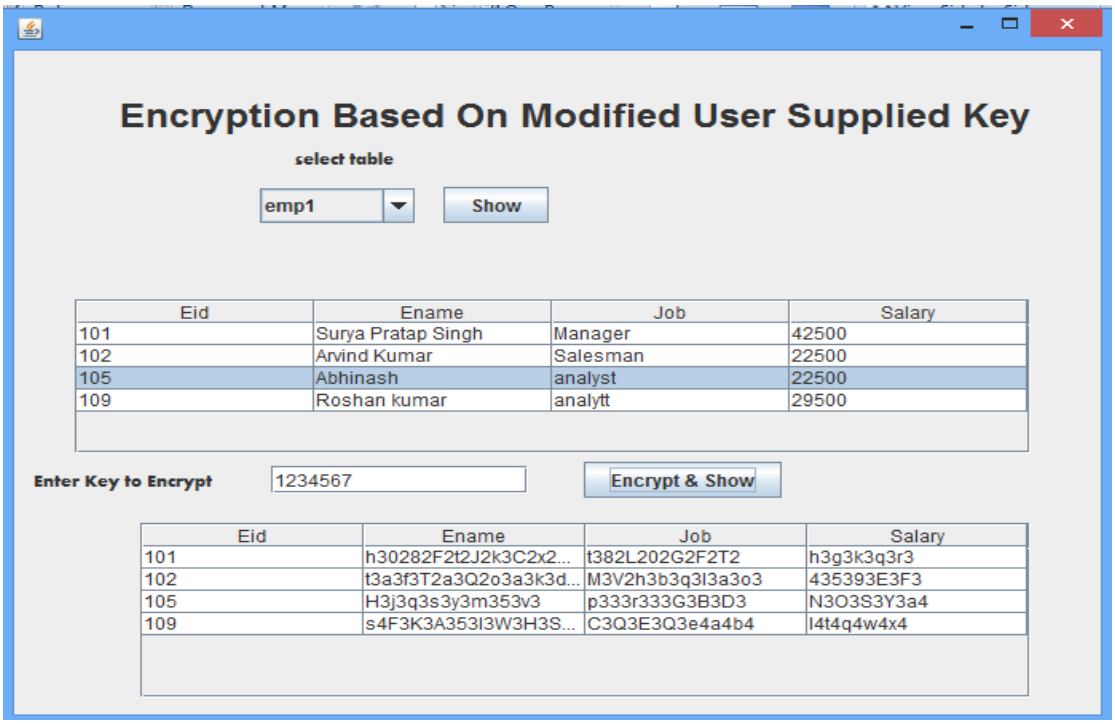

Fig 2: Encryption of database Content

From the above fig it is clear that same characters of the plain text are converted into different cipher text character and hence it is easy to guess.

For decryption the user have to supply the same user key. Here user can also view the content of database by selecting the table name and clicking on show button and then press Decrypt and show button to decrypt the content of table. This is explained in the following fig 3.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

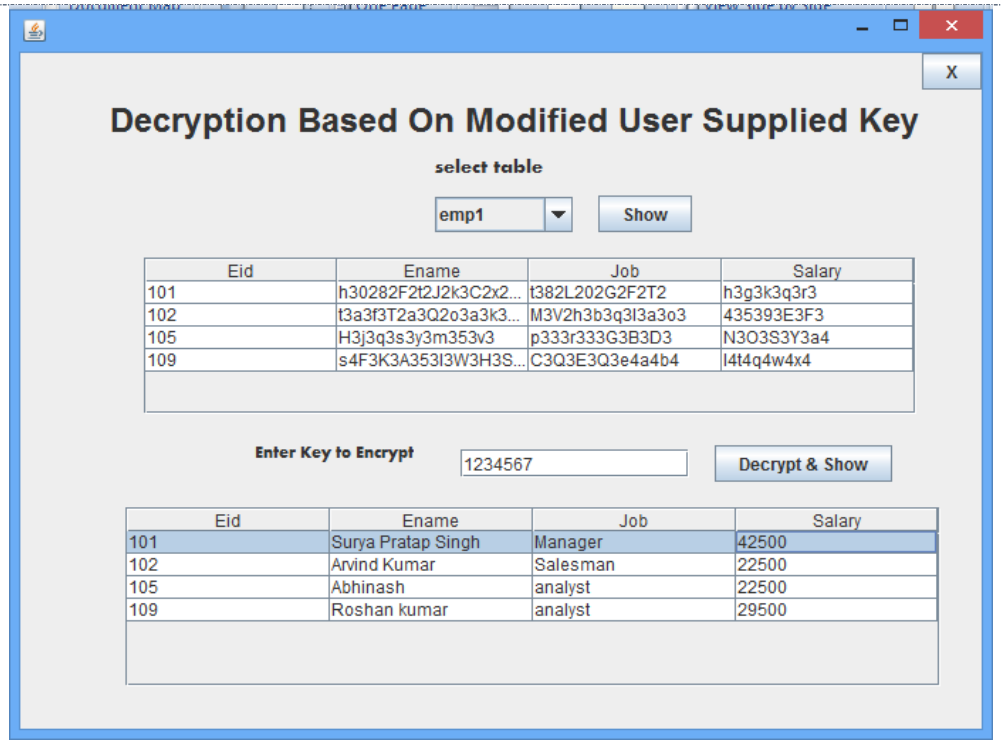

Fig 2: Decryption of database Content

CONCLUSION

The security of database is very important as the use of database is growing very fast. One of the best ways to secure the database is by the use of Cryptography. In earlier papers we presented the use of User supplied key base encryption which have some limitations like it convert the same plain text character to the same cipher text character for all occurrence.

In this paper we implemented the encryption and decryption based on modified user supplied key. In this mechanism we use sum of digit and an incremented key to overcome from the above mentioned problem. Here the same plain text character is converted into different cipher text character for different occurrence and hence the security of the database is enhanced.

REFERENCES

[1] Surya Pratap Singh, U. N. Tripathi, Manish mishra, "‘‘Preserving Database confidentiality using User key based encryption" International Journal of Engineering Sciences & Research Technology (IJESRTol. 5 Issue 10 October 2016"
[2] Surya Pratap sing, Arvind kumar Maurya, U. N. Tripahti, "Enforcing Database Security using Cryptography and Secure Database Catalog" International Journal of Engineering Research & Technology (IJERT) / Vol. 5 Issue 01, January-2015
[3] N. A. Ghani, Z. M, Sidek, "Owner-Controlled Towards Personal Information Stored in Hippocratic Database", Computer Technology and Development, 2009. ICCTD09. International Conference on (Vol: 2).
[4] A. Menezes, P. van Oorshot, and S. Vanstone, Handbook of applied cryptography. CRC Press, 1997, http://www.cacr.math.uwaterloo.ca/ hac/.
[5] Boneh D, Crescenzo GD, Ostrovsky R, Persiano G (2004) Public Key Encryption with Keyword Search. Encrypt 2004, LNCS 3027
[6] Al-Fedaghi, S. (2007). How sensitive is your personal information? . Paper presented at the ACM Symposium on Applied Computing, Seoul, Korea.
[7] K Hemanth, Srinivasulu Asadi, Dabbu Murali," High Secure Crypto Biometric Authentication Protocol", "International Journal of Computer Science and Information Technologies, Vol. 2 (6)"

[8] Stephane Jacob, "Cryptanalysis of a Fast Encryption Scheme for Databases and of its Variant" in Encyclopedia of Cryptography and Security. Springer, 2010, 2nd Edition.
[9] Hacigumus, H." Providing database as a Service" Proceedings. 18th International Conference on Data Engineering, 2002.
[10] Kadhem, H. Amagasa, T. ; Kitagawa, H. "A Novel Framework for Database Security Based on Mixed Cryptography" Fourth International Conference on Internet and Web Applications and Services, 2009. ICIW '09.