ISSN: 2321-7782 (Online)
Volume 3, Issue 7, July 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

A New Bit-level Columnar Transposition Encryption Algorithm

Sayantan Majumdar[1]
Department of Computer Science, St. Xavier's College, Kolkata, India

Abhisek Maiti[2]
Department of Computer Science, St. Xavier's College, Kolkata, India

Biswarup Bhattacharyya[3]
Department of Computer Science, St. Xavier's College, Kolkata, India

Asoke Nath[4]
Department of Computer Science, St. Xavier's College, Kolkata, India

Abstract: ***Confidentiality of information is the main aim of cryptography. But use of complex encryption techniques to secure less important information creates unnecessary technical overhead. Basic encryption techniques may be useful in such scenarios. In the presented paper,the authors have used a very simple but powerful encryption technique "Bit-level Columnar Transpose" to encrypt any file. In this method the entire bit string (obtained from the plain text) is mapped into a two dimensional matrix. The column and encryption numbers[1] are decided by a "secret key" given by the user.A random key[2] is then generated on the basis of the column size. Finally the encrypted text is obtained by column-wise scanning of the bit-string using the generated random key. To decipher it, the "secret key" has to be known so that the column and encryption numbers are generated correctly. Finally, the cipher is converted into a bit string and mapped into a 2D matrix on the basis of the random key and then re-ordered to get the plain text file.***

Keywords: Cryptography, Encryption, Decryption, Secret Key, 2D Matrix

I. INTRODUCTION

Columnar Transpose is a basic encryption method which may not be a strong encryption technique but it is useful when complexity of encryption method is an issue, i.e. it may be used to prevent unauthorised fast access to the information. In the presented method multiple-layer columnar transpose is applied at bit-level to strengthen the encryption.

Since the entire encryption and decryption phase works at bit-level, the presented method may take up a lot of time and computer resources when working with large files. However, due to bit-level encryption it is quite impossible to decrypt the cipher text using a brute force attack. The attacker has to know the secret key as well as the random key to correctly decipher and obtain the original file. In this method, the column number lies between 1 and 256 inclusive and the encryption number lies between 1 and 64 inclusive. So in the worst case scenario of a brute force attack, there will be 64×256! iterations to correctly decrypt the encrypted file.

Presented scheme typically consists of three algorithms:

» Generation of the column and encryption numbers on the basis of a secret key

» Encryption of the message using a random key

» Decryption of the cipher using the secret key and the random key

In the presented method the maximum secret key length is taken as 16 and the maximum column number and encryption number are taken as 256 and 64 respectively.

II. ALGORITHMS

a) Column and Encryption number

Generation Algorithm:

Secret Key Length(n)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Base (b)	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2

Table-I

STEP-1: Start

STEP-2: n:=length(secret_key)

STEP-3: Obtain the base value(b) using n from Table-I

STEP-4: Sum= $\sum_{m=1}^{n} ASCIICode \times b^m$

STEP-5: Store this sum as a String('s').

STEP-6: n:=length(s)

STEP-7: n1:= $\sum_{m=1}^{n}(ASCIICode - 48) \times m$

STEP-8: column_no:=Mod(Sum, n1)

STEP-9: If column_no=0 then column_no:=n1

STEP-10: Else if column_no>256 then column_no:=256;

STEP-11: s:=reverse_string(s)

STEP-12: n2:= $\sum_{m=1}^{n}(ASCIICode - 48) \times m$

STEP-13: encryption_no:= Mod(Sum, n2)

STEP-14: If encryption_no=0 then encryption_no:=n2

STEP-15: Else if encryption_no>64 then encryption_no:=64;

STEP-16: End

B. Encryption Algorithm:

STEP-1: Start

STEP-2: Inputsecret key, input file path and output directory path

STEP-3: Get column size(n1) and encryption number(n2) from 3.1

STEP-4: Generate akey(K) randomly(K will contain all integers from 0 to n1-1 in a random order)

STEP-5: Convert the entire plain text into a bit-string('s')

STEP-6: for i:= 1 to n2 do

STEP-7: Fill up an n×m character matrix(M) where m=n1 and n=length(s)/m in row major order.

STEP-8: Generate cipher text(C) using K and M.

STEP-9: done

STEP-10: Convert C to its ASCII form

STEP-11: Write C and K to files

STEP-12: End

C. Decryption Algorithm:

STEP-1: Start

STEP-2: Inputsecret key, cipher text, key and output directory paths

STEP-3: Get column size(n1) and encryption number(n2) from 3.1

STEP-4: Convert the cipher text into bit-string(s)

STEP-5: for i:= 1 to n2 do

STEP-6: Store the characters of 's' in a matrix(M) column-wise on the basis of the key.

STEP-7: Extract the chars from M row-wise and store the characters in a string(D)

STEP-8: done

STEP-9: Convert D to its ASCII form and write to the output directory.

STEP-10: End

III. CONCEPTUAL MODELS

a) Encryption:

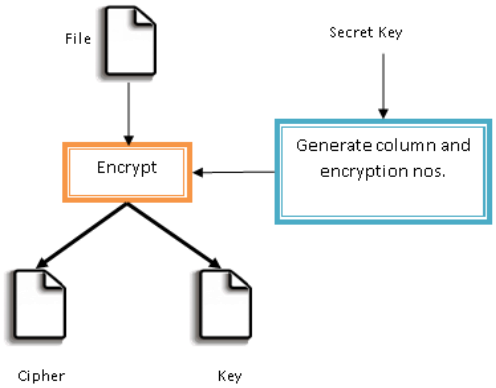


Fig.1 Encryption Module

b) Decryption:

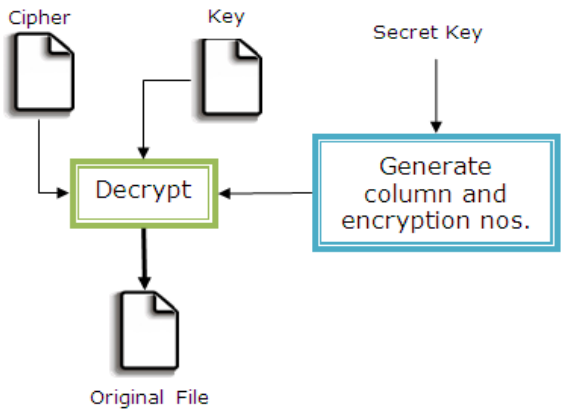
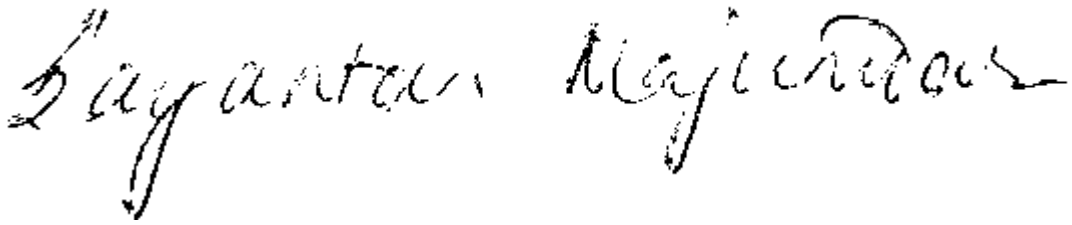


Fig.2 Decryption Module

IV. RESULTS AND DISCUSSION

a) With a pngimage(size= 1443 Bytes)



1) Encryption:

Secret Key="ABCD@#"

Column size=250

Encryption number=64

Random Key= 92 235 40 105 22 178 162 203 200 141 189 161 47 73 180 116 115 1 8 164 136 72 38 230 131 154 126 243 21 121 66 78 4 150 58 144 83 77 170 53 75 145 192 52 129 132 33 45 43 30 232 181 247 194 198 46 152 34 219 207 146 48 165 71 62 87 244 190 90 173 119 123 148 70 188 140 163 227 130 246 213 5 69 67 223 147 89 15 182 98 133 226 84 156 97 68 35 82 113 204 196 86 63 238 149 88 157 236 237 55 166 101 167 100 231 212 229 95 197 54 135 120 118 127 206 215 13 51 37 122 42 56 24 171 233 137 240 61 241 91 93 209 245 23 12 248 205 25 85 169 151 224 159 134 138 176 114 108 160 155 208 10 11 153 177 143 79 139 124 9 41 239 80 39 179 31 104 218 142 0 249 202 6 193 175 201 168 27 220 111 109 106 216 234 103 228 16 57 28 19 184 44 185 186 32 125 65 74 210 17 225 50 7 222 59 117 64 158 18 96 242 195 112 49 211 217 76 94 102 191 199 29 60 36 183 26 128 107 20 3 221 110 172 2 99 14 174 81 187 214

Cipher(size= 1443 Bytes):

_¶¬)éTJ_Tô³•ŒUR—Ÿ‰__Œ¡Ù*f*$@€nåÿ‹Àˆ÷äÉùq<éõµ¬û8ÔN • ò • ^«ÈÓ—;
{_ÊÍ»(._d_„_T[¨±óPÕPj%_VL^9Y_J_X_°PD_¢_'õÈz•b™€œ • :_EÄv ü?¸P J|µA¹ ÀIB9_()___{L‡_`„¥Ô_µJ1ó • PáR

.
.
.

mrÙ¤ãü;n • f6EÿEÔUp_Ã±¶&z$³___>—7ÛÖ_è®¥

¨é]Äåd

c_J,~ulÑpÑýN´xï ðæi__ü;OF¹~_±

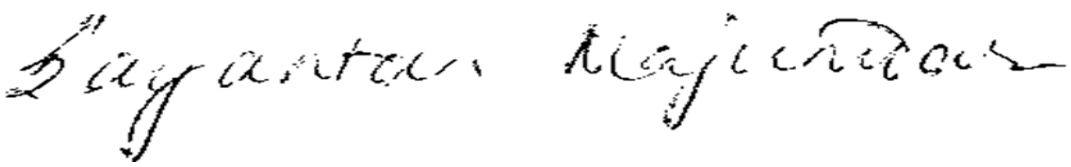
`V[_!`žj¥§□Æô'ç ð±÷®¬šø,ºJµ___3_èwú_¹u^{¶'i_/^˜M_Ä—¤_ã‰«

;Ì

Encryption Time= 5.042077482 s

2) Decryption:

Decrypted file:



Decryption time= 4.503205651 s

b) With a jpeg image (size= 2054 Bytes)

1) Encryption:

Secret Key="1234!"

Column size=91

Encryption number=55

Random Key= 58 9 43 17 10 79 89 38 33 2 53 31 8 32 3 15 65 86 29 68 13 27 59 85 47 41 78 44 36 48 51 76 75 54 56 14 30 88 5 40 18 52 12 90 69 22 19 71 66 21 49 81 24 34 60 46 77 62 67 28 20 39 35 64 45 80 72 83 4 1 6 57 16 84 26 23 61 63 87 0 37 74 42 70 73 55 82 7 50 25 11

Cipher (size=2054 Bytes):

Ž.(öxE_Ýý8ù,,Àqæø§]_Õå¦_€_ŸY_XÑüíLÙÔ±fc_õ_□_v$èž3~5@CÂr_a_&_ÝehR5—Î"d¦ÈçcQSš×j□
Êv‘.ÚÄ´3Ä'Òo_Ñ_Z&ÙaP_SÅ_à,xi¢_c_,□2_,$_º0_…t_Îã¾œþ_Ätö,£>ÒB+_-€ `#€_eMO! _‰Iíì§±O±«ÙkÎ–_,

.
.
.

É_ôžèK~J„0¹

”\¢å¢…£,Ú:ºò

_ B

´›C’_8@_ˆÜ'(f_IL*f*_Ý*å xkž·ŒÑ_Ÿ˜

Encryption Time= 8.036378337 s

2) Decryption:

Decrypted file:

Decryption time= 7.972700368 s

c) With a text file (size= 115 Bytes)

AbhisekMaiti

Roll- 3-15-12-0544

Department of Computer Science, St. Xavier's College Kolkata

email: xyz@gmail.com

1) Encryption:

Secret key="AB"

Column size=44

Encryptionnumber=8

Random Key= 20 11 26 23 25 2 35 19 0 22 28 31 9 21 32 38 13 17 43 41 16 10 33 30 39 27 4 14 34 1 24 15 6 5 8 18 40 42 7 36 29 37 12 3

Cipher (size=115 Bytes):

_Ð˜r+_$_|Ð¶Ø□ÍìÑ!+&êI¨A;Üfô'_/gÁã -WK__¨pc†_˜9uqü¼¶€.'ÉÝ‡_ùŠ_íÐ ªÀhh¢túµJ8u_Ö‹9ªý#@")ÀL&_'2IÛý¼»äùG%nx^@PS œ¡□Dâ:wJ[´_¿ëèˆsÀ_ü¤ ×_Aec.□ò7<¬É•Æ¢£ 8Œ_"ç&~¦E&èÇp0Ø®Ê1®_Qá‡EJ sxQ¤˜__£

Encryption Time= 59.849007 ms

2) Decryption:

Decrypted file:

Abhisek Maiti

Roll- 3-15-12-0544

Department of Computer Science, St. Xavier's College Kolkata

email: xyz@gmail.com

Decryption time= 32.945815 ms

d) With a bit pattern(size= 65 Bytes)

11111111111111111111111111111111111111111111111111111111111111111

1) Encryption:

Secret Key=www.sxccal.edu

Column size=130

Encryption number=64

Random Key= 76 120 31 42 40 125 95 90 58 115 97 73 28 113 29 108 67 107 26 14 99 128 59 78 74 111 24 61 36 117 22 75 46 64 16 35 25 57 13 93 17 129 12 83 4 39 88 56 70 122 110 9 3 82 100 44 51 53 19 68 60 48 33 52 50 15 85 81 8 2 55 37 23 1 62 124 96 27 30 71 105 38 43 69 121 86 41 6 112 5 10 66 77 72 34 45 114 92 20 98 102 126 89 101 123 65 21 119 7 103 63 104 87 0 118 80 18 109 94 79 106 47 32 91 84 49 127 116 54 11

Cipher (size=65 Bytes):

CMzB³U (˜Éo nŠ_ £___fÖc_ØH¼¾

Š_ _@@X

_'À`Òd□□_'(Y_táxY2__~7"d˜_&Å

Encryption Time= 125.686046 ms

2) Decryption:

Decrypted file: 11111111111111111111111111111111111111111111111111111111111111111

Decryption time= 110.528257 ms

e) With a docx file (size= 4680Bytes)

Name- Biswarup Bhattacharyya

SXC CMSA

Roll- 3-15-12-0568

email: wxyz@gmail.com

1) Encryption:

Secret Key="5688"

Column size=91

Encryption number=61

Random Key= 80 50 44 57 85 62 6 83 89 8 61 27 46 70 26 68 5 38 33 17 21 47 30 74 76 37 58 31 16 84 3 52 14 72 56 9 64 35 78 82 41 90 15 42 32 63 66 65 7 55 28 73 12 0 24 88 20 36 4 40 22 71 43 10 69 51 67 18 48 54 39 13 60 79 86 59 53 49 23 2 1 25 87 11 77 19 45 29 75 34 81

Cipher (size=4680 Bytes):

Rq0° • ì$5#È

8_};__è…ô5_4°lü¡ÿ>Â¶Ž_ò_LD‚_‘ÚÃOœKt€B_2_□˜Û_µ7

thn

.
.
.

l¦£^JÐØTw‚ø£µêÀ_Œ_MRV¶4_(àÁiE*ƒ*- ŒBŠæc“@Ä__CoÐ´_Ö_…%é»æE8„ÅC_ó¬‚5ÒÇ4ÎëŠ¾t®ÍÂþ’Œép

Encryption Time= 50.12955906 s

2) Decryption:

Decrypted File:

Name- Biswarup Bhattacharyya

SXC CMSA

Roll- 3-15-12-0568

email: wxyz@gmail.com

Decryption time= 51.208750505999994 s

f) With a java file (size= 180 Bytes)

```
class bits
{
public static void main(String args[])
{
    int n=65,mask=1<<7;
    while(mask!=0)
    {
        int k=(n & mask)!=0?1:0;
        System.out.print(k);
        mask=mask>>1;
    }
```



```
}

}
```

1) Encryption:

Secret Key="ABCD@#"

Column size=250

Encryption number=64

Random Key= 222 180 3 108 223 13 38 52 208 159 66 93 192 205 248 68 46 132 135 241 163 247 207 120 113 177 149 183 187 225 41 134 232 24 91 7 227 153 191 226 83 158 230 188 28 45 143 184 71 6 175 97 229 234 203 228 164 209 118 212 105 129 172 101 89 30 141 179 17 240 131 87 64 62 136 124 48 169 1 171 19 15 145 73 133 29 173 121 197 220 231 233 74 244 111 152 165 154 237 20 65 215 182 57 213 42 23 123 8 224 59 119 186 116 80 127 178 102 77 122 95 0 10 144 166 168 54 151 49 200 106 125 156 214 22 147 26 88 161 115 189 36 157 138 139 96 12 137 112 40 126 201 204 198 185 37 90 103 99 58 72 69 16 35 216 196 33 174 114 92 238 56 31 146 142 202 160 55 242 2 155 32 235 85 70 109 51 50 221 11 176 5 150 75 100 63 110 140 44 86 167 25 94 236 170 76 193 98 246 18 206 43 128 107 194 162 190 21 81 4 130 249 60 243 245 39 9 217 61 210 219 239 14 148 53 79 82 218 117 211 104 47 84 78 27 199 67 195 181 34

Cipher (size=180 Bytes):

_Ð˜r+_$_|Ð¶Ø□ÍìÑ!+&êI¨A;Üfô'_/gÁã -WK_

_¨pc†_˜9uqü¼¶€.'ÉÝ‡_ùŠ_íÐ ªÀhh¢túµJ8u_Ö‹9ªý#@")ÀL&_'2IÛý¼»äùG%nx^@PS œ¡□Dâ:wJ[´_¿ëèˆsÀ_ü¤

×_Aec. • ò7<¬É•Æ¢£ 8Œ_"ç&~¦E&èÇp0Ø®Ê1®_Qá‡EJ sxQ¤˜__£

Encryption Time= 300.469514 ms

2) Decryption:

Decrypted file:

```
class bits
{
public static void main(String args[])
{
    int n=65,mask=1<<7;
    while(mask!=0)
    {
        int k=(n & mask)!=0?1:0;
        System.out.print(k);
        mask=mask>>1;
    }
}
}
```

Decryption time= 153.635495 ms

N.B: For test cases ***A***and ***F***, the secret keys are the same but since the key is randomly generated the two keys are different.

V. Conclusion and Future Scope

During the end of 19th century and the beginning of 20th century columnar transpose encryption was considered as a strong encryption technique. At the time of second world war French mathematicians broke these type of ciphers. Breaking columnar transpose is not very difficult. Most easy attack is guessing the key length using brute force attack then reordering the columns to obtain a meaningful text. More scientific approach is applying frequency distribution and anagramming the cipher text. Columnar Transpose is also vulnerable to genetic algorithm and dictionary attack. One major flaw of this type of encryption is that the keys close to the correct key will reveal long portion of the cipher text.

Another drawback of columnar transposition cipher is its huge computing complexity. It takes hours to encrypt ~100 KB file in a standard home computer. But it works fine for very small input files. The application of presented method is not recommended for the files greater than 15 KB.

Although it has many drawbacks, the columnar transposition is anextremely secure encryption algorithm when multiple layer of encryption is used at bit-level along with sufficiently long key. Another advantage of using this method is that the cipher text size is same as the plain text size. Besides its mechanism and implementation is quite easy.

References

1. Symmetric Key Cryptography Using Random KeyGenerator, AsokeNath, SaimaGhosh, MeheboobAlamMallik, Published in Worldcomp 2010 Proceedings , Las Vegas, USA in July 2015.: http://www.researchgate.net/publication/221199745
2. http://docs.oracle.com/javase/7/docs/api/java/security/SecureRandom.html
3. Modern Ecryption Standard (MES) : Version-II, Rahul Deep Sircar, GunjanSekhon, AsokeNath, Prooceedings of IEEE International Conference CSNT-2013 held at Gwalior April 6-8,2013, Page-506-511.
4. Bit Level Generalized Modified Vernam Cipher Method with Feedback, International Journal of Advanced Computer Research(ISSN(print):2249-7277 ISSN(online): 2277-7970), Volume-2, Number-4 Issue-6, Page-24-30, Dec(2012).
5. Modern Encryption Standard(MES) version-I : An Advanced Cryptographic Method, SomdipDey, AsokeNath, Proceedings of IEEE International Conference WICT- 2012 held at IIITM-K, Trivandrum Oct 30 to Nov 1, 2012, Page No. 242-247(2012).
6. Bit Level Encryption Standard(BLES) : Versiob-II, GauravBhadra, Tanya Bala, SamaikBanik, JoyshreeNath and AsokeNath, Proceedings of IEEE International Conference WICT-2012 held at IIITM-K, Trivandrum Oct 30 to Nov 1, 2012, Page No. 121-127(2012).
7. Ultra Encryption Algorithm(UEA): Bit level Symmetric key Cryptosystem with Randomized Bits and Feedback Mechanism, International Journal of Computer Applications(IJCA) USA, Vol-49, No.5, Page-34-40(2012).
8. Ultra Encryption Standard(UES) Version-IV: New Symmetric Key Cryptosystem with bit-level columnar Transposition and Reshuffling of Bits, Satyaki Roy, NavajitMaitra, JoyshreeNath, ShalabhAgarwal and AsokeNath, International Journal of Computer Applications(IJCA)(0975-8887) USA Volume 51-No.1.,Aug, Page. 28-35(2012).

