Symmetric key Cryptography using two-way updated -Generalized Vernam Cipher method: TTSJA algorithm

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

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

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

Joyshree Nath
A.K.Chaudhuri School of IT
Raja Bazar Science College,
Calcutta University
Kolkata, India

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

ABSTRACT
In the present paper the authors have introduced a new updated two-way generalized vernam cipher method called TTSJA. Chatterjee et.al developed a method [1] where they used three independent methods such as MSA [2], NJJSAA [3] and modified generalized vernam cipher method. Nath et al already developed some symmetric key methods [2,3,4,5] where they have used bit manipulation method and some randomized key matrix for encryption and decryption purpose. In the present work the authors have used updated generalized vernam cipher method in two directions. One from first character to last character and then we perform vernam method with XOR operation from last to first We found the results are quite satisfactory even for short message and repeated characters also. The advantage of the present method is that the overhead is minimum but the encryption is very hard. This method may be applied to encrypt short message such as SMS, password, ATM code etc. In the present work the authors have introduced updated Vernam Cipher method. The authors modified the standard Vernam Cipher method for all characters (ASCII code 0-255) with randomized keypad and also introduced feedback. After first phase encryption the modified vernam cipher method applied from last character to the first using random keypad and feedback. In the second phase instead of adding the keypad ASCII the authors performed the XOR with keypad and the encrypted text (after first phase). This method closely monitored on different known plain text and it was found that this method is almost unbreakable. The present method allows the multiple encryption and multiple decryption. To initiate the encryption process a user has to enter a text-key which may be maximum of 16 characters long. From the text- key the randomization number and the encryption number is calculated using a method proposed by Nath et al [2]. A minor change in the text-key will change the randomization number and the encryption number quite a lot. The present method is a block cipher method and it can be applied to encrypt confidential data in Defense system, Banking sector, mobile network, Short message Service, Password, ATM key etc. The advantage of the present method is that one can apply this method on top of any other standard algorithm such as MSA, DJSA, NJJSAA, TTJSA, DJMNA etc [2,4,3,1,5]. The method is suitable to encrypt any type of file.

Keywords
MSA Algorithm, NJJSAA, TTJSA, DJMNA, Vernam Cipher method.

1. INTRODUCTION
The data security and data encryption are two important terms in Computer network. Few years back it was not a serious issue but now it is the most important issue. It is a big challenge for a sender to send confidential data from one computer to another computer or from one computer to a remote computer through a server. Imagine a situation when the entire banking information is hacked and what will happen after few seconds. Probably the entire Banking Industry will collapse. It means the security of originality of data has now become a very important issue or challenge in data communication network. The confidential data cannot be sent from one computer to another computer in original form as the intruder can intercept the data and can do any kind of danger for sender or for the receiver. Sometimes we send bank transaction report, question paper, suggestions over the mail. The hackers can intercept those mails and can do any kind of damage. All these things are happening because of free network access. If someone applies some common sense can access any data from any machine. Imagine that the hacker somehow break secured key of e-banking and intercept all data. The data must be protected from any unwanted intruder otherwise any massive disaster may happen all on a sudden. The disaster may happen in any business house. This may be further worse if the confidential data of one business house is stolen by some intruder and pass it to some rival company then what will happen. That implies now in every places there must be some security of data. There must be various levels of security in data management. Because of this hacking problem network security and cryptography is an emerging research area where the people are trying to develop some good encryption algorithm so that no intruder can intercept the encrypted message. These cryptographic algorithms can be classified into two

categories: (i) symmetric key cryptography where one key is used for both encryption and decryption purpose. (ii) Asymmetric key cryptography: where two different keys are used one for encryption and the other for decryption purpose. The advantage of symmetric key cryptography is that the key management is very simple as one key is used for both encryption as well as for decryption purpose and this key must be secret and it should be known to sender and the receiver only and no one else. On the other hand in public key cryptography there are two keys one key is called public key which may be available to anyone who wants to encrypt the message and the other one which is called secret key or the private key that must be kept only with the receiver. Because of factorization problem from encryption key no one can construct the decryption key. The problem of Public key cryptosystem is that one has to do massive computation for encrypting any plain text. Due to massive computation the public key crypto system may not be suitable to encrypt short message, in sensor networks, mobile networks etc. In the present work we are proposing a symmetric key method called TTSJA which is a combination of 2 updated vernam cipher methods. In forward vernam cipher method we add the ASCII code of the plain text and the keypad and we take modulo with 256 and take the result as feed back in the next column. After finishing forward pass we apply slightly different way the vernam cipher method from last character of the encrypted text. Here instead of adding the ASCII code of the two characters we apply the XOR operation and the result we make modulo with 256 and take the same result as feedback to the next column. In the second pass we apply XOR operation instead of addition of ASCII codes. First we choose a block of 256 characters in a block. The last few characters which will be <256 we apply same method but now the size of key will be changed according to length of the residual characters. We have tested this method on various types of known text files and we found that the results were satisfactory. The present method may be applied in Defense network, mobile network, ATM network, Short message service etc.

2. TTSJA ALGORITHM

Now we will describe TTSJA algorithm:

A. ENCRYPTION ALGORITHM:

Step 1 : Start
Step 2 : mat[][] is initialized (values from 0 to 255) in row major manner
Step 3 : keygen() and randomization() function called
Step 4 : times2=times
Step 5 : file f1 is copied into outf2
Step 6 : k=1
Step 7 : if k>secure go to Step 12
Step 8 : p=k%2
Step 9 : if p==1
vernamenc(outf2,outf1);
times=times2;
file_rev(outf1,file1);
vernamencxor(file1,outf1);
times=times2;
Step 10 : if p==0
vernamencxor(outf2,outf1);
times=times2;
file_rev(outf1,file1);
vernamenc(file1,outf1);
times=times2;
Step 11 : k=k+1, Goto step 7
Step 12 : file_rev(outf1,outf2)
Step 13 : The contents of file 'outf2' is copied into file 'file2'
Step 14 : k=k+1 Goto Step 7
Step 15 : Stop

vernamenc(f1,f2)

Step 1 : Start
Step 2 : mat[][] is initialized (values from 0 to 255) in row major manner
Step 3 : randomization() function called
Step 4 : value of mat[][] after randomization is copied into key[] (row major)
Step 5 : pass=1 , times3=1 , ch1=0
Step 6 : A block from the input file f1 is taken(≤256 characters)
Step 7 : If block size < 256 , goto Step 14
Step 8 : Each character of the block is copied into a character array str[]
Step 9 : encryption() function called and str[] is passed as parameter along with the size of the block
Step 10 : if(pass==1)
times=(times+times3*11)%64
pass++
else if(pass==2)
times=(times+times3*3)%64
pass++
else if(pass==3)
times=(times+times3*7)%64
pass++
else if(pass==4)
times=(times+times3*13)%64
pass++
else if(pass==5)
times=(times+times3*times3)%64
pass++
else if(pass==6)
times=(times+times3*times3*times3)%64
pass=1
Step 11 : randomization() called using current value of times
Step 12 : value of mat[] copied into key[]
Step 13 : next block extracted , goto Step 7
Step 14 : Each character of the last block (residual characters , if any) is copied into str[]
Step 15 : encryption() called using str[] and the no. of residual characters
Step 16 : Return

encryption(str[],n)

Step 1 : Start
Step 2 : ch=(str[0]+key[0]+ch1)%256
Step 3 : ch is wrote into output file
Step 4 : ch1=ch
Step 5 : i=1
Step 6 : if i≥n , goto Step 11
Step 7 : ch=(str[i]+key[i]+ch1)%256
Step 8 : ch is wrote into the output file
Step 9 : ch1=ch
Step 10 : i=i+1 , goto Step 6
Step 11 : Return

vernamencxor(f1,f2)

Step 17 : Start
Step 18 : mat[][] is initialized (values from 0 to 255) in row major manner
Step 19 : randomization() function called
Step 20 : value of mat[][] after randomization is copied into key[] (row major)
Step 21 : pass=1 , times3=1 , ch1=0
Step 22 : A block from the input file f1 is taken(≤256 characters)
Step 23 : If block size < 256 , goto Step 14
Step 24 : Each character of the block is copied into a character array str[]
Step 25 : encryptionxor() function called and str[] is passed as parameter along with the size of the block
Step 26 : if(pass==1)
times=(times+times3*11)%64
pass++
else if(pass==2)
times=(times+times3*3)%64
pass=pass+1
else if(pass==3)
times=(times+times3*7)%64
pass=pass+1
else if(pass==4)
times=(times+times3*13)%64
pass=pass+1
else if(pass==5)
times=(times+times3*times3)%64
pass=pass+1
else if(pass==6)
times=(times+times3*times3*times3)%64
pass=1
Step 27 : randomization() called using current value of times
Step 28 : value of mat[] copied into key[]
Step 29 : next block extracted , goto Step 7
Step 30 : Each character of the last block (residual characters , if any) is copied into str[]
Step 31 : encryption() called using str[] and the no. of residual characters
Step 32 : Return

encryptionxor(str[],n)

Step 12 : Start
Step 13 : ch=str[0] XOR key[0] XOR ch1
Step 14 : ch is wrote into output file
Step 15 : ch1=ch
Step 16 : i=1
Step 17 : if i≥n , goto Step 11
Step 18 : ch=str[i] XOR key[i] XOR ch1
Step 19 : ch is wrote into the output file
Step 20 : ch1=ch
Step 21 : i=i+1 , goto Step 6
Step 22 : Return

B. DECRYPTION ALGORITHM:

Step 1 : Start
Step 2 : mat[][] is initialized (values from 0 to 255) in row major manner
Step 3 : keygen() and randomization() function called
Step 4 : times2=times
Step 5 : file_rev(f1,outf1)
Step 6 : k=secure
Step 7 : if k<1 go to Step 12
Step 8 : file_rev(outf1,outf2)
Step 9 : p=k%2
Step 10 : if p==1
vernamdecxor(outf2,outf1);
times=times2;
file_rev(outf1,file1);
vernamdec(file1,outf1);
times=times2;
else if p==0
vernamdec(outf2,outf1);
times=times2;
file_rev(outf1,file1);
vernamdecxor(file1,outf1);
times=times2;
Step 11 : k=k-1 Goto Step 7
Step 12 : copy the contents of file ‘outf1’ into file ‘file2’
Step 13 : Stop

vernamdec(f1,f2)

Same as vernamenc() function. The only difference is that decryption() function is called instead of encryption() function.

decryption(str[],n)

Step 1 : Start
Step 2 : ch=(256+str[0]-key[0]-ch1)%256
Step 3 : ch is wrote into the output file
Step 4 : i=1
Step 5 : if i≥n , goto Step 9
Step 6 : ch=(256+str[i]-key[i]-str[i-1])%256
Step 7 : ch is wrote into the output file
Step 8 : i=i+1 , goto Step 5
Step 9 : ch1=str[n-1]
Step 10 : Return

vernamdecxor(f1,f2)

Same as vernamencxor() function. The only difference is that decryptionxor() function is called instead of encryptionxor() function.

decryptionxor(str[],n)

Step 11 : Start
Step 12 : ch=str[0] XOR key[0] XOR ch1
Step 13 : ch is wrote into the output file
Step 14 : i=1
Step 15 : if i≥n , goto Step 9
Step 16 : ch=str[i] XOR key[i] XOR str[i-1]
Step 17 : ch is wrote into the output file
Step 18 : i=i+1 , goto Step 5
Step 19 : ch1=str[n-1]
Step 20 : Return

Now we will describe how we calculate randomization number(=times) and encryption number(=secure). The present method is fully dependent on the text-key which is any string of maximum length 16 characters long. From the text-key we calculate two important parameters (i) Randomization number and (ii) Encryption number. To calculate these two parameters we use the method developed by Nath et al (2). We are giving below how we calculate the above two parameters:

Suppose key=AB
Choose the following table for calculating the place value and the power of characters of the incoming key:

TABLE I. LENGTH OF KEY() VS. BASE VALUE(B)

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

(i) Calculate $Sum = \sum_{m=1}^{n} \text{ASCII Code} * b^{m}$ ----(1)

where n=number of characters in the input text-key.

Now we calculate the sum for key="AB" using equation (1). Here n=2, b=16

Sum=65*161 + 66 * 162
=17936

Now we will show how we calculate 2 parameters from this sum:

(i)Randomization number(n1):

num1=1*1+7*2+9*3+3*4+6*5=84
n1=sum mod num1=17936 mod 84=44
Note: if n1=0 then n1=num1 and if n1>32 then n1=Mod(n1 , 32)

(ii)Encryption number(n2):

num2=6*1+3*2+9*3+7*4+1*5=72
n2=sum mod num2 =17936 mod 72 =8
Note: if n2=0 then n2=num2
if n2>32 then n2=Mod(n2 , 32)

The randomization of key matrix is done using the following function calls:

Step-1: call Function cycling()
Step-2: call Function upshift()
Step-3: call Function downshift()
Step-4: call Function leftshift()
Step-5: call Function rightshift()

For detail randomization methods we refer to the work done by Nath et al [2] and Chatterjee et. al [1]...

3. RESULTS AND DISCUSSIONS

We apply our encryption and decryption algorithm on various types of files and we found in every case it was working perfectly ok. In Table-2 the different type of files are given below:

TABLE II. ENCRYPTION TEST CASES

ORIGINAL TEXT	**ENCRYPTED TEXT**
1. AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA (64-A's)	1. _@ÑÔ ‰šÝQ□š|Œ_þ7ÿ`6_Å9yãï>¹ 1ùÜáøg+,e¹_È…'ÅæÝÄæ"IjÈ;_ëåiOQß†×Ú¤
2.BAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA (B + 63-A's)	2. v[Èa__Ôž¼Ø_O_3NH× šÅQý□m_ } árÝS2^ß>‰±êÙ]iäOîte™$_¹,0ÕhJ¡½…™îrEŸ_Î_b
3. AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB (63-A's + B)	3. ÷r!èè,□RRd‹9J½Ú)sš6ØX´□ò"dQ&¹©·¨ÙJ í_jŠêe_I dC¾æMQzÄÒ'HÄFr□Ê¾_NV
4. 1111111111111111111111111111111111111111111111111111111111111111 (64-1's)	4. zÙ_·Žj_p`sæ„´□'~_èWN_'¡_ÓƒU ß=ÓŠ:¿_Æ– 1*v¹Î¯ÝHÓ"îÍ³@'_Y§9xj¹Ý□□zÉÛúõ…œÖÒðkT!åÇ_=#‰&ˆ .=Ãm*, ƒà}
5.	5.

<table>
<tr><td>22222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222 (64-2’s)</td><td>U ¬Šì§
"ªs_
JŸ©Bx´k6OÁéU!__B°ÛÙ
!Øk+Mcèf ƒ…‘œ_õx÷,mWÿ__…T{9Ä~!',Qmx«S€x_¡
ŽÀˆÄP×s¿_xå·F_(Ç»</td></tr>
<tr><td>6.
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 (64-0’s)</td><td>6.
Ãž_É_Âª'Óš3h[›@¬¢yº…Á]   t¬_&7ƒ‘->-
W?¯¨ÒÓKq®
_ _ü {_·Ãa@_E
¸Ý_º5óþ›
_°ĒÍ÷I(þ,þB¿'Ñˆ šM/¦¾Î+}Z</td></tr>
<tr><td>7. 000</td><td>7. F î</td></tr>
<tr><td>8. 001</td><td>8. çce</td></tr>
<tr><td>9. 010</td><td>9. ñsx</td></tr>
<tr><td>10. 011</td><td>10. š__</td></tr>
<tr><td>11. 100</td><td>11. ò|q</td></tr>
<tr><td>12. 101</td><td>12. çbf</td></tr>
<tr><td>13. 110</td><td>13. çbg</td></tr>
<tr><td>14. 111</td><td>14. @_î</td></tr>
<tr><td>15.
St Xavier's college with the pioneering legacy was the first educational institute of the city to think in terms of introducing Computer Studies. St. Xavier's Computer Centre (SXCC), since its inception, on the 4th November 1985, has grown from strength to strength and at present is a renowned institution providing computer education to people from all walks of life.</td><td>15. ?   6   }Ý • Dzyâ   ÔD}ÄæEIÀ¦ïi   ¦Z [,PÑ5YïQ   Ë
  æSãù E•C þÃ> (NÚ„n½-¦¢a’ËT¥‘°I   ÔD
û5V • NöÁ¨IÒ)»)ºÁ4Ù ©õn «€ šS` óU>°<br> Ä´ ”'ù ÿ>÷*éÔ˜ïŠ4iØ   8]@‰½ª· –»<br>n;? µÜ{ È,<đ<br>ž·ÙþZ €+Òo^W³wiþ¿»}Íf©¼f |Ý! ¯¬   ai! ÷Ïdi_<br>¼'[,Ûåd Ý<žÐØÄ @S4Moê+›Ý* ã ð®…ŸÊ<br>  ?`.=ˆ Êºyð=æJë›ƒ¢öû7d ê{+>Ô @ø5m·UxÏ²`8²ƒì
Í ÙÔz ‘tÖÙ   ^Z½, Âåk^< _({4.>z~ãÈ” -
ê8F!9Ö ›pÑ
*1 • Ó $¹¼)Å œŸóÙ§žùµw“ ÎâÅŽðài@IT</td></tr>
<tr><td>16.
St Xbvier's college with the pioneering legacy, was the first educational institute of the city to think in terms of introducing Computer Studies. St. Xavier's Computer Centre (SXCC), since its inception, on the 4th November 1985, has grown from strength to strength and at present is a renowned institution providing computer education to people from all walks of life.

<u>(The 5th character of the previous paragraph is changed from ‘a’ to ‘b’)</u></td><td>16. _

I$Y˜SDÙ_[¸qñÛô²O\ƒýÑ· o•´_ac›½aú°_¬!-
_|ÇŽ_ L_>É¾_Šš_Žÿ9n

__=ë»¾"¦‡C,5Z_ú¬{œ_X*m_ò Ã]¦__¯ŠÈc´‘x6ôi„Ò^_×ñ í¡
UNkÌ/¡äj_ sÄÀçë}Ë_¦ëŽpsàÁãÉÀ¢LÎÒL__|_Ì^5`
úNóÁ_@´’_SÛª¬£V8·ÛŽà âÀ¢ÅOb,Ibmýc_2wØ3_ [e øÃ _

Æ_* -_H$º úÈŽ__vt

Ã ÷õU€ç_Fg¼__·O$ =è¥¿–²zÌ‡fiØ€äf£â¾% Ùðùvüþ9þ
ßˆã]A__Oð´Ýy¶á@®_”‰ü^È_ÞT^ù*_Á,nÎ úý\«|<Ù_ÓÐ¯Õì
"K“Çu•¡ùb“_ßf<zk+__’ôÓfÌ___§è _;„s–
_?Âß_Ž$_i gŸÈâ« › É</td></tr>
</table>

4. CONCLUSION

In the present work we use two different algorithms to make the encryption process too hard. From the results also it is very clear that our encryption method is very hard. We apply our method on some known text where the same character repeats number of times and we found that after encryption in the output pattern there is no repetition of pattern in the output string. We have tested this feature closely and we found in almost all cases we have got satisfactory result. This is possible as we have used modified vernam cipher method with feedback character. The key matrix of size 16x16. This key may be generated in 256! ways. The present method uses two distinct methods i.e. modified vernam cipher method and vernam cipher method using XOR operation. Our method is block cipher method. The overhead of the present method is very less and hence this method may be applied specially in encryption of bio-informatics data where the same pattern is repeated or to encrypt short message, password etc. There is lot of scope to modify the present method. The merit of this method is that it is almost impossible to break the encryption algorithm without knowing the exact key matrix. We propose that this encryption method can be applied for data encryption and decryption in banks, in defense, in government sectors for

sending confidential data. The present algorithm may be used for database encryption also.

5. ACKNOWLEDGMENTS

We are very much grateful to Department of Computer Science to give us opportunity to work on symmetric key Cryptography. One of the authors (AN) sincerely expresses his gratitude to Fr. Dr. Felix Raj and Fr. Jimmy Keepuram for giving constant inspiration to carry out research work. AN is grateful to University Grants Commission for giving financial assistance through Minor Research Project. JN expresses her gratitude to A.K.Chaudhuri School of IT for allowing to work in research project at St. Xavier's College. TC, TD, SD and AN are also thankful to all 3rd year Computer Science Hons. Students (2011-2012 batch) for their inspiration and support to finish this work.

6. REFERENCES

[1] Symmetric key cryptosystem using combined cryptographic algorithms- generalized modified vernam cipher method, MSA method and NJJSAA method: TTJSA algorithm – Trisha Chatterjee, Tamodeep Das, Joyshree Nath, Shayan Dey and Asoke Nath, Proceedings of IEEE International conference: World Congress WICT-2011 held at Mumbai University 11-14 Dec, 2011, Page No. 1179-1184(2011)

[2] Symmetric key cryptography using random key generator, A.Nath, S.Ghosh, M.A.Mallik, Proceedings of International conference on SAM-2010 held at Las Vegas(USA) 12-15 July,2010, Vol-2,P-239-244

[3] Advanced Symmetric key Cryptography using MSA method: DJSSA symmetric key algorithm, Dripto Chatterjee, Joyshree Nath, Soumitra Mondal, Suvadeep Dasgupta and Asoke Nath, Journal of Computing, Vol 3, issue 2, Page 66-71(Feb 2011).

[4] An Integrated symmetric key cryptography algorithm using generalized vernam cipher method and DJSA method: DJMNA symmetric key algorithm : Debanjan Das, Joyshree Nath, Megholova Mukherjee, Neha Chaudhury and Asoke Nath: Proceedings of IEEE International conference : World Congress WICT-2011 to be held at Mumbai University 11-14 Dec, 2011, Page No.1203-1208(2011)

[5] Data Hiding and Retrieval, A.Nath, S.Das, A.Chakrabarti, Proceedings of IEEE International conference on Computer Intelligence and Computer Network held at Bhopal from 26-28 Nov, 2010.

[6] A new Symmetric key Cryptography Algorithm using extended MSA method :DJSA symmetric key algorithm, Dripto Chatterjee, Joyshree Nath, Suvodeep Dasgupta and Asoke Nath, IEEE conference CSNT-2011 held at SMVDU 03/06/2011 to 05/06/2011.

[7] Symmetric key Cryptography using modified DJSSA symmetric key algorithm, Dripto Chatterjee, Joyshree Nath, Sankar Das, Shalabh Agarwal and Asoke Nath, WORLDCOMP-2011 held at Las Vegas,USA 18-21 Jul 2011

[8] Cryptography and Network, William Stallings, Prectice Hall of India.

[9] Modified Version of Playfair Cipher using Linear Feedback Shift Register, P. Murali and Gandhidoss Senthilkumar, UCSNS International journal of Computer Science and Network Security, Vol-8 No.12, Dec 2008.