Ultra Encryption Algorithm (UEA): Bit level Symmetric key Cryptosystem with Randomized Bits and Feedback Mechanism

Satyaki Roy
Department of Computer Science (Autonomous),
St. Xavier's College, Kolkata, India

Shalabh Agarwal
Department of Computer Science (Autonomous),
St. Xavier's College, Kolkata, India

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

Navajit Maitra
Department of Computer Science (Autonomous),
St. Xavier's College, Kolkata, India

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

ABSTRACT

The present paper proposes a new cryptographic algorithm called Ultra Encryption Algorithm (UEA). Nath et al recently developed few efficient encryption methods such as UES version-I, Modified UES-I, UES version-II, TTJSA, DJMNA Nath et. al showed that TTJSA and DJMNA is most suitable methods to encrypt password or any small message. The name of the present method is Ultra Encryption Algorithm (UEA) as it is a Symmetric key Cryptosystem which includes multiple encryption, advanced bit-wise randomization, new serial feedback generation and bit-wise encryption technique with feedback. Evidenly, in the result section the authors have shown the spectral analysis of encrypted text as well as plain text to show the effectiveness of the bit-level algorithm. The spectral analysis reveal that the present module offers encryption that is free from repetitive text patterns and strong enough against the standard cryptographic attacks.

General Terms

Data Security, Cryptography

Keywords

UES, multiple encryption, TTJSA, bit extraction, randomization, feedback, password, shift

1. INTRODUCTION

Due to massive growth in communication technology and the tremendous growth in internet technology in the last decade, it has become a real challenge for a sender to send confidential data from one computer to another. It has become very important to protect the integrity of date at every cost. Public softwares are available to decode password of some unknown e-mail. The data must be protected from any unwanted intruder otherwise a massive disaster may take place. Any intrusion into protected information could cause damage to any organization. Cryptography algorithms are of two types (i) Symmetric key cryptography where we use single key for encryption and decryption purpose. And (ii) Public key cryptography where we use one key for encryption purpose and one key for decryption purpose.. Both the methods have their advantages as well as disadvantages.

The current algorithm is a symmetric key cryptosystem however its uniqueness lies in the fact that it works at the bit-level and our tests on various types of known text files reveal that even if there is repetition in the input file, the encrypted file contains no repetition of patterns. The test results for the UEA algorithm shall prove that the rarest of text inputs and texts with subtle alteration when encrypted, yields vast variations in the corresponding cipher text files.

2. UEA ALGORITHM

The UEA algorithm includes a number of modules (i) Bit-level Randomization and Integration Module (ii) Advanced Bit-wise Encryption Technique with Feedback (ABETF). (iii) A simple serial encryption module. The integration module combines the randomization module with the two feedback modules.

(i). Bit-wise Randomization and Integration

ENCRYPTION:

Step 1: Enter the names of the plain text file, cipher file and the password that may have a maximum length of 64-bytes.

Step 2: Calculate le=length (secret key).

Step 3: Calculate cod= ∑ [key[i]*(i+1)], where i is the index value of array 'key' (0 < i < le).

Step 4: Calculate enc=mod (cod, 60) and rand=mod (cod, 20) where enc=encryption number and rand = randomization number. If enc<10 then enc=10. If rand < 10 then rand=10.

Step 5: Invoke bit-wise encryption on the plain file with feedback with the ABETF algorithm.

Step 6: Compute l=sizeof (plain text file).

Step 7: Compute count = (siz*siz)/8 where siz=64. It signifies the number of bytes encrypted at once. By default count=512 bytes.

Step 8: Compute z=l/count. It signifies the number of iterations that are needed to encrypt a file.

Step 9: Create key matrix 'mat' of dimension siz X siz. It holds the values 1 to [(siz*siz) -1] row-wise.

Step 10: Define s=0

Step 11: If s > = enc then GOTO 26

Step 12: Randomize the key matrix ‘mat’ by MSA algorithm as many as ‘rand’ times.

Step 13: Extract count bytes of plain file and split them into respective bits and store in 1d array ‘parr’.

Step 14: Define auxiliary array ‘parr2’. Define i=0.

Step 15: If i> = (siz*siz) then GOTO 18

Step 16: Define ro=i/siz, co=mod (i, siz) and n= mat [ro][co] where n is the position corresponding to position i of the plain bits array.

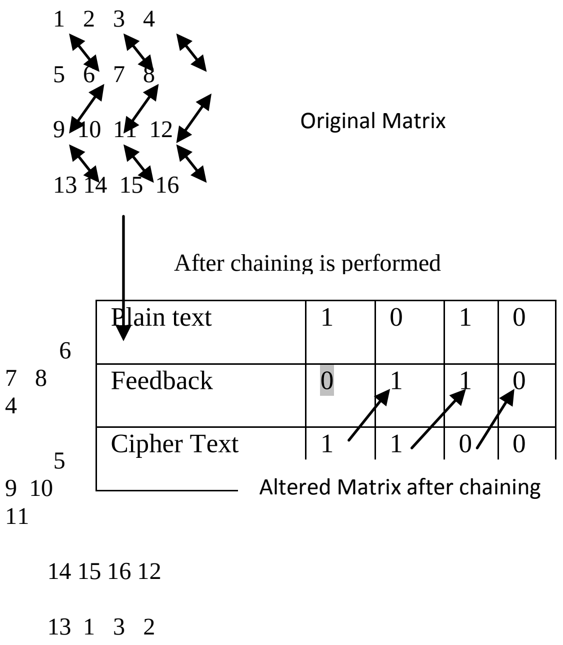
Step 17: Perform parr2 [i] = parr[n] in order to randomize the bits of the plain file by exchanging the bits according to the randomized key matrix. Increment i. GOTO 15

Step 18: Convert the bits in the array parr2 into corresponding bytes. GOTO 13

Step 19: Now the algorithm processes the residual bytes. The residual bytes are again split into bits and stored in 1d array ‘parr’.

Step 20: Randomize the residual key matrix ‘matt’ using the module rant which takes the number of residual bytes ‘rem’ as parameter. We perform *leftshift, cycling, downshift, chaining operations* on the key matrix ‘matt’. The chaining operation is shown below.

Fig 1: Chaining operation



Step 21: The size of the array 1d array ‘parr’= (rem*8) where rem = number of residual bytes. Define i=0.

Step 22: If i >= (rem*8) then GOTO 24. Define ro=i/8, co=i%8 and n=matt [ro][co].

Step 23: Perform parr2 [i] = parr[n] in order to randomize the bits of the plain file where parr2 is again the auxiliary array of plain bits. Increment i. GOTO 22.

Step 24: The cipher bits are now serially encrypted using the serial encryption module.

Step 25: Convert the bits into the respective bytes to yield residual cipher bytes. Increment s. GOTO 11.
Step 26: END

DECRYPTION

The decryption process applies the randomization of bits, the bit-wise feedback generation and the feedback generation in the exact opposite order to yield the plain text.

(ii). Serial feedback generation module

ENCRYPTION

1. Enter the name of the file containing the plain bits.
2. Define character ch1 = Starting value of feedback=ASCII 48 (ASCII for character 0)
3. Define character ch2=1 extracted bit of plain text.
4. Perform ch1=ch1+ch2-96 to generate the serial bit feedback. Character ch1can have values 0 or 1 i.e. ASCII 48 or 49.
5. Write the encrypted bit ch1 into the cipher file.
6. Goto 3 until the entire plain text bits is processed.
7. End

Table I: Serial feedback generation.

Initial feedback=0
Plain bits: 1010
Cipher bits: 1100

Plain text	1	0	1	0
Feedback	0	1	1	0
Cipher Text	1	1	0	0

DECRYPTION

1. Enter the name of the file containing the cipher bits.
2. Define character ch1=1 extracted bit of cipher file.

3. Define character ch2=another extracted bit of cipher file.
4. Perform ch1=ch1+ch2-96 where ch1=decrypted bit. The variable ch1 may have values 0 or 1 i.e. ASCII 48 or 49.
5. Write the character ch1 into the cipher file.
6. Perform ch2=ch1. Goto 3 until the entire cipher file is processed.
7. End

(iii). The Bitwise encryption module using bit-level feedback generation.

ENCRYPTION

Step-1: Compute cod=∑key[i]*(i+1) from the password 'key' provided by the user.

Step-2: Compute k=modulus (cod, 256). Define i=0.

Step-3: Write the character with ASCII value k in the file containing the feedback keypad. Increment k and i. If i<l GOTO 3.

Step-4: Randomize the feedback keypad using simple character randomization.

Step-5: Split the plain and feedback keypad into respective bits and store them in two files.

Step-6: Extract one bit from the plain file and the feedback file each and store them in characters 'ch' and 'chb' respectively. Define c=0.

Step-7: Compute m= (ch + chb + c)-96

Step-8: If m>=2 then perform m=m-2.

Table-II: ABETF ENCYPTION (FEEDBACK GENERATION): The algorithm takes into consideration that the ASCII value of '0' is 48. Hence, the subtraction of (2*48) is performed during the computation of m.

After the bitwise OR Cipher bit becomes the feedback and the carry is ignored

Step-9: Perform c=m

Step-10: Write the integer m in the output file.

Step-11: Goto 6 until the end of the file is reached.

Step-12: Convert the bits in the output file into respective bits to produce the cipher file.

Step-13: END

DECRYPTION

Step-1: Compute cod=∑key[i]*(i+1) from the password 'key' provided by the user.

Step-2: Compute k=modulus (cod, 256). Define i=0.

Step-3: Write the character with ASCII value k in the file containing the feedback keypad. Increment k and i. If i<l GOTO 3.

Step-4: Randomize the feedback keypad with bit-wise randomization.

Step-5: Split the plain and feedback keypad into respective bits and store them in two files.

Step-6: Extract one bit from the plain file and the feedback file each and store them in characters 'ch' and 'chb' respectively. Define c=0.

Step-7: Compute character chb= (ch + chb + c)-96

Step-8: If chb > = 2 then perform chb=chb-2.

Step-9: Perform c=ch-48

Table-III: ABETF DECYPTION (the cipher bit is feedback)

FEEDBACK: **c**	0	1	1	0
PLAIN BITS: **ch**	0	0	1	1
KEY BITS: **chb**	1	0	0	1
CIPHER BITS:	1	1	0	0

FEEDBACK: **c**	0	1	1	0
CIPHER: **ch**	1	1	0	0
KEY BITS: **chb**	1	0	0	1
PLAIN BITS:	0	0	1	1

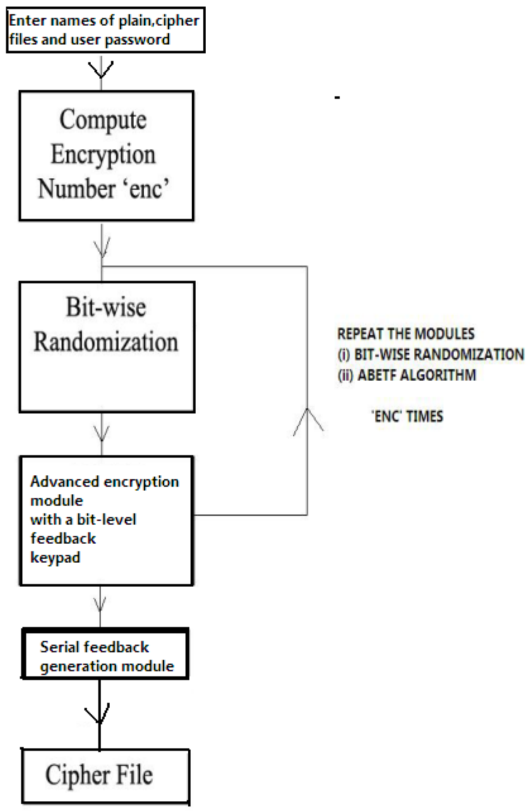
Step-10: Write the ASCII of character 'chb' in the output file.

Step-11: Goto 6 until the end of the file is reached.

Step-12: Convert the bits in the output file into respective bits to produce the cipher file.

Step-13: END

3. WORKING OF UEA MODULE


Enter names of plain,cipher files and user password
Compute Encryption Number 'enc'
Bit-wise Randomization
Advanced encryption module with a bit-level feedback keypad
Serial feedback generation module
Cipher File
REPEAT THE MODULES
(i) BIT-WISE RANDOMIZATION
(ii) ABETF ALGORITHM
'ENC' TIMES

4. TEST RESULTS

A number of expansive experiments have been performed on the plain texts in order to ascertain whether the UEA algorithms is effective in all possible varieties of plain files irrespective of their sizes and formats. The authors have looked closely in order to be able to detect repetitions in patterns that may make the programs susceptible to cryptographic attacks like Brute Force. Secondly the dependence of the encryption on the user password has also been tested thoroughly to make sure that the algorithm makes good use of the feature of multiple encryption.

The combination of the three modules have been done in a manner that the process of encryption is accelerated. The algorithm works at the bit-level and the test results show that the quality and strength of encryption obtained is significantly higher than the techniques that work with bytes. The test results include **(i) Some General plain text inputs (ii) the change in the cipher text when applied on the same plain text but with different passwords and (iii) Frequency analysis of some rare test cases. (iv) Time Performance. (v)**

Byte wise comparitive analysis of the cipher files of three similar inputs.

(i) Some general plain text inputs

Table IV: The miscellaneous text inputs and corresponding cipher files.

Password: people

Number of times encryption is performed=5 (minimum encryption number)

Plain Text	Cipher Text
Dddddddddddddddddddddd dddddddd	†_}7S^_¥‹0^Ñð¼þÆ½:Ôa Ð·óI]-Ùá
He is great	631Iµ_àø_}□
Ce is great	8U?_Dˆ_gô"s_
A Christian Minority Higher Educational Institution, St. Xavier's was founded in 1860 by a Catholic Minority Religious body, the Society of Jesus, and was affiliated to Calcutta University in 1862. While preference is shown to the educational and cultural needs of the Minority community, admission is open to all irrespective of caste, creed and nationality!	)¼9Ú×ÏŽ□¨Z_œL«õùm3ÏE Tig„‡,□o__6DÉÒv+_□!_Ñ _!pÕËok$H□DSð˜t?)¢Ô/5 ÄO}J2Ãt›®:G»□Æ{¹êkôÑ 0__¬¦È¥š.±¼¤_*ßv□ÍL`Óø Ež]MÊn•Ã=$__2ÔrÞÇ□… Å8[Í_ÿ|•ø:{@Ç•□ÈH²ûöA; ±d\Á___Ù‡ó$\Ø_q__R'E¤ "g¤g‰íö\ß_õ´ÄVÅ¬Ô$È3 Âø□o_@¦2¨.Csé¼®ª_I†mÃ TÈ_ßhÈ.`ùïp8_P¦RŽÓ Ù`Ï_ z_Æo__µä'"j~Eñ_¯_<[À2á ôO_»_ã_Gµ«Š782'ÏÚ___" b«‰ýU_ëU□¨y/g□__9=_M išR0Üò¬FlmV_¥□aÂ•¶$Þö YÆs□&&6['×__U«ØLæK D@:=€ä¢‡‰¢þï_°0û
aaaaabaaaaa	ùæý_¥Ÿ¶ü,n

(ii) The change in the cipher text when applied to the same plain text but with different passwords

Table-V: The following table will help us better understand the importance of the user password and how the cipher file changes drastically with changing passwords.

Plain text	Password	Cipher Text
AAAAAAAAAAAAAA AAAAAAAAAAAAAA AAAAAAAAAAAA	10	´A·õ_êú£&!wÎƒŠ:ÿnY(=4šbì>èÈ¶jZr™¹ÈøÃíz·
AAAAAAAAAAAAAA AAAAAAAAAAAAAA	11	3|_yÔ_Cˆ'□G.ŽÖ™÷Y ÕÅ<_iyµy'd‹„Âô_ºÜA¨ Õø¶

AAAAAAAAAAAA		
AAAAAAAAAAAAAA AAAAAAAAAAAAAA AAAAAAAAAAAA	12	÷J8¡É□_|]^Bn¶DÚ(zã9ñ ZðÅf_5±HdŸ□©ýu³Kn À„i
AAAAAAAAAAAAAA AAAAAAAAAAAAAA AAAAAAAAAAAA	13	____Ù·*_[ˆŸY_N_Œ_™ +_ñž>ì™_‡„¦B*ò á¯_J ‰NIë

(iii)Frequency Analysis of Rare text inputs

The diagram below represent the frequency spectrum of two rare text inputs. **1000 characters of ASCII 1 have been shown in Result-I. 1000 characters of ASCII 2 have been shown in Result-II and ASCII 255 in Result-III**. The objective of this experiment is to understand the distribution of characters in the cipher file. The dominance of a few characters in the cipher file is a negative. However in the frequency distribution in the following figures, it will be evident that the distribution of characters in the cipher file is rather balanced.

The Y-AXIS represents the frequency and the X-AXIS signifies the characters 0-255. The graphs shown in the diagram represent the distribution of characters in the cipher file. The spectrum is expected to be balanced with regard to the distribution of characters.

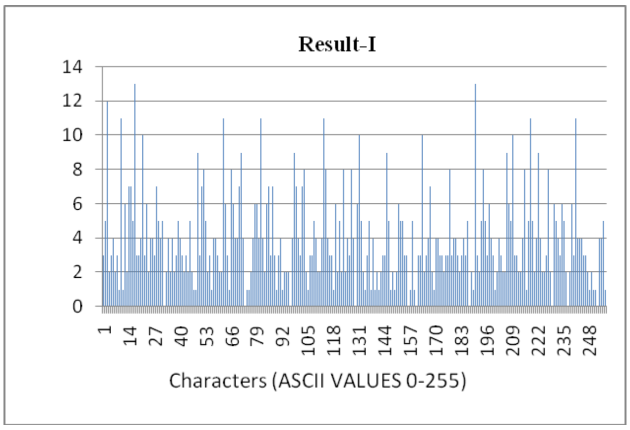


Fig-II: The result-I – 1000 characters of ASCII 1.

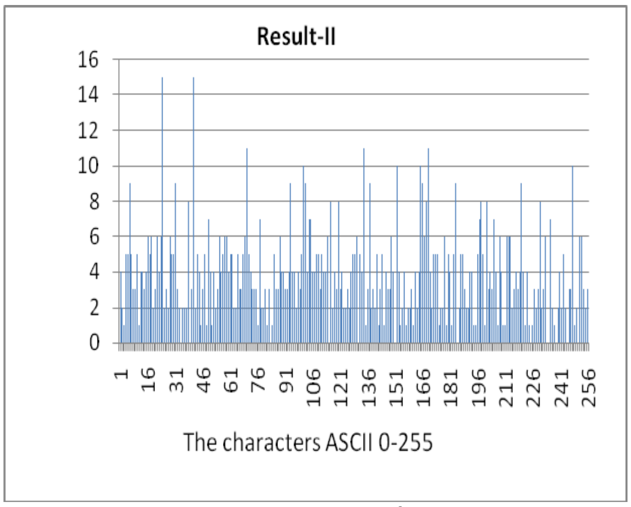


Fig-III: The result-II – 1000 characters of ASCII 3.

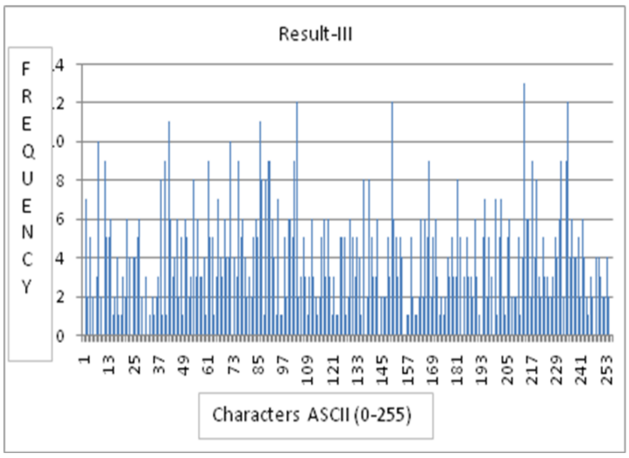


Fig-IV: The result-III – 1000 characters of ASCII 3.

(iv) Performance Analysis w.r. t time

Table-VI: The UEA algorithm has been tested under various conditions in order to ensure that the time taken to perform the encryption and decryption procedures can be noted down and studied. The results have been catalogued below.

Plain Text	Time to Encrypt (in seconds)	Time to Decrypt (in seconds)
64 characters of ‘A’	1	1
256 characters of ‘A’	2	2
300 characters of ‘A’	2	2
512 characters of ‘A’	3	3

(v) Byte wise comparitive analysis of the cipher files of three similar inputs.

Table-VII: The table below shows the cipher files for 30 characters of ASCII 252, 253 and 254 and their byte wise comparison. We look for any similarity in the corresponding bytes of cipher files. From the table below, we see that there are almost no such occurrences.

Byte Number	Cipher for ASCII 252	Cipher for ASCII 253	Cipher for ASCII 254
1	Ü	Û	Þ
2	Q	q	?
3	Ê	_	□
4	÷	_	» {_
5	_	6	Õ
6	§	M	Ó
7	e	Í	³
8	Ñ	Ö	□
9	Þ	þ	@
10	*	À	_
11	ð	X	ë
12	_	_	_
13	™	¹	,
14	G	Ô	¦
15	|	£	ü
16	¤	®	O
17	Ž	à	¥
18	Õ	R	s
19	Q	‰	«
20	Ù	□	"
21	G	ß	(
22	X	®	T
23	V	!	ƒ
24	Ñ	_	¯
25	Ê	E	²
26	Â	,,	□
27	Ƒ	‰	Õ
28	¦	Û	Þ
29	Ü	q	?
30	Q	_	□

5. DISCUSSION ON FUTURE SCOPE AND CONCLUSIONS

The idea behind this UEA algorithm was to ensure that we devised a method that could handle all possible files. The plain text files have been split into respective bits before applying the aforementioned algorithms. The exhaustive test results reveal a satisfactory range of cipher files. Even when the same characters are provided as input, the cipher files have almost no occurrence of repetitive patterns. The combination of the three methods of randomization, serial feedback and the bit-level feedback pad has proved very effective. The use of multiple encryption and the role of the password provided by the user have also been demonstrated in the test result

The stress was largely on rare text inputs which are likely to suffer from bad encryption. As mentioned before have applied our method on some known text where the single character repeats itself for a number of times and we have found that after encryption there is no repetition of pattern in the output file.

It was also understood that any tampering of the cipher file could make the retrieval of plain text impossible.

6. ACKNOWLEDGEMENT

We are grateful to the Department of Computer Science for giving us the unique 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 allowing us to carry out research work. AN is thankful to the University Grant Commission for their support and financial assistance. JN is grateful to A.K. Chaudhuri School of IT and SR, NM, SA and AN are thankful to St. Xavier’s College. The authors extend their thanks to the Computer Science Hons. (2011-2012)for their encouragement and help.

7. REFERENCES

[1] Symmetric Key Cryptography using Random Key generator: Asoke Nath, Saima Ghosh, Meheboob Alam Mallik: ŖProceedings of International conference on security and management(SAMř10ŗ held at Las Vegas, USA Jull 12-15, 2010), P-Vol-2, 239-244 (2010).

[2] A new Symmetric key Cryptography Algorithm using extended MSA method :DJSA symmetric key algorithm, Dripto Chatterjee, Joyshree Nath, Suvadeep Dasgupta and Asoke Nath : Proceedings of IEEE CSNT-2011 held at SMVDU(Jammu) 3-5 June,2011, Page-89-94.

[3] New Symmetric key Cryptographic algorithm using combined bit manipulation and MSA encryption algorithm: NJJSAA symmetric key algorithm: Neeraj Khanna,Joel James,Joyshree Nath, Sayantan Chakraborty, Amlan Chakrabarti and Asoke Nath : Proceedings of IEEE CSNT-2011 held at SMVDU(Jammu) 03-06 June 2011, Page 125-130.

[4] Advanced Symmetric key Cryptography using extended MSA method: DJSSA symmetric key algorithm: Dripto Chatterjee, Joyshree Nath, Soumitra Mondal, Suvadeep Dasgupta and Asoke Nath, Jounal of Computing, Vol3, issue-2, Page 66-71,Feb(2011).

[5] Advanced Steganography Algorithm using encrypted secret message : Joyshree Nath and Asoke Nath, International Journal of Advanced Computer Science and Applications, Vol-2, No-3, Page-19-24, March(2011).

[6] Symmetric key Cryptography using modified DJSSA symmetric key algorithm, Dripto Chatterjee, Joyshree Nath, Sankar Das, Shalabh Agarwal and Asoke Nath, Proceedings of International conference Worldcomp 2011 held at Las Vegas, USA, July 18-21, Page 312-318, Vol-I(2011).

[7] Cryptography and Network, Willian Stallings, Prentice Hall of India.

[8] Cryptography & Network Security, B.A.Forouzan, Tata Mcgraw Hill Book Company.

[9] 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 conference WICT-2011 held at Mumbai University Dec 11-14,2011

[10] 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, Shyan Dey and asoke Nath, Proceedings of IEEE conference WICT-2011 held at Mumbai University Dec 11-14,2011.

[11] Ultra Encryption Standard(UES) Version-I: Symmetric Key Cryptosystem using generalized modified Vernam Cipher method, Permutation method and Columnar Transposition method, Satyaki Roy, Navajit Maitra, Joyshree Nath,Shalabh Agarwal and Asoke Nath, Proceedings of IEEE sponsored National Conference on Recent Advances in Communication, Control and Computing Technology-RACCCT 2012, 29-30 March held at Surat, Page 81-88(2012).

[12] Symmetric key Cryptography using two-way updated – Generalized Vernam Cipher method: TTSJA algorithm, International Journal of Computer Applications(IJCA, USA), Vol 42, No.1, March, Pg: 34 -39(2012).