iJET EXPLORE International Journal of Engineering & Technology
CENTRE OF PROFESSIONAL RESEARCH PUBLICATIONS
ISSN 2049-3444

International Journal of Engineering and Technology Volume 4 No. 6, June, 2014

Random Key Permutation Stream Algorithm Based on Modified Functions in AES Algorithm

[1]Nada Hussein Mohammad Ali Al-Khafaji, [2]Abdul Monem S. Rahma, [3]Abdul Mohsen Jaber, [4]Sufian Yousef

[1]Computer Science Department, University of Baghdad, Baghdad, Iraq
[2,3]Computer Science Department, University of Technology, Baghdad, Iraq
[4]Engineering & Built environment Department, Anglia Ruskin University

ABSTRACT

The present research deals with the implementation of stream cipher algorithm based on modified AES block cipher concept to achieve high complexity in encryption and decryption processes. Two functions have been modified in the original AES (Modified SubByte and Modified MixColumn) to increase the complexity and keeping the same executed time for both original and modified AES algorithms. The proposed algorithm implemented in stream cipher uses three functions, two of them are Modified SubByte and Modified MixColumn transformation of AES algorithm plus the Block Permutation. Tiny blocks of sizes (2*2, 4*4, 6*6 etc.) have been implemented on the proposed algorithm. The study concluded that the proposed algorithm applied on stream cipher gives successful results with a considerable time for encryption and decryption process.

Keywords: *stream cipher algorithm, cryptographic algorithms, AES, block cipher*

1. INTRODUCTION

Depending upon the number of keys used, cryptographic algorithms are classified as asymmetric algorithms (public key) and symmetric algorithms (secret key). Symmetric-key system uses a single key that both the sender and recipient have. Asymmetric-key system uses two keys, a public key known to everyone and a private key that only the recipient of messages uses. The symmetric key algorithms are further classified as block cipher and stream cipher. Block ciphers operate with a fixed transformation on large blocks of plain text data while stream ciphers operate with the time varying transformation on individual plain text bits. Stream ciphers do not have a standard model and varieties of structures are followed in their design [1].

The implementation of a wide, complex and varied networks, let the security development has great importance to the communication innovation. The advance encryption standard (AES) algorithm considered to be the leader of this innovation. A large variety of approaches for modifying AES have been appeared so as to satisfy the varying criteria of different applications [2].

The present article applies stream cipher algorithm based on modified AES block cipher concept with high complexity in encryption and decryption processes.

Stream and Block Cipher Concepts

Stream cipher comprises of two main components: a mixing function and a key stream. Mixing function is usually exactly an XOR function, whereas key stream generator is the main unit in stream cipher encryption [3].

Stream ciphers are typically faster than block ciphers but the latter typically require more memory. Because block ciphers encrypt a whole block at a time they are more susceptible to noise in transmission whereas with stream ciphers bytes are individually encrypted with no connection to other chunks of data. Also, stream ciphers do not provide integrity protection or authentication, whereas some block ciphers (depending on mode) can provide integrity protection, in addition to confidentiality. Because of all the above, stream ciphers are usually best for cases where the amount of data is either unknown, or continuous - such as network streams. On the other hand, block ciphers are more useful when the amount of data is pre-known such as a file, data fields or request/response protocols such as HTTP where the length of the total message is known already at the beginning [4].

2. THE ADVANCED ENCRYPTION STANDARD (AES) ALGORITHM

The Advanced Encryption Standard (AES) was published by NIST (National Institute of Standards and Technology) in 2001. AES is a symmetric block cipher that is intended to replace DES as the approved standard for a wide range of applications. It has a variable key length of 128,192 or 256 bits. It encrypts data blocks of 128 bits in 10, 12, 14 rounds depending on key size. AES encryption is fast and flexible in block ciphers. It can be implemented on various platforms. AES can be operated in different modes of operation like ECB, CBC, CFB OFB, and

ISSN: 2049-3444 © 2014 – IJET Publications UK. All rights reserved.

CTR. In certain modes of operation they work as stream cipher [1].

The AES is a private key block cipher that processes data blocks of 128 bits with key length of 128, 192, or 256 bits. The AES algorithm's operations are performed on a 2-D array of 4 *times* 4 bytes called the State. The initial State is the plaintext and the final State is the ciphertext.

Rijndael round function operates on a state Nr times, where Nr is equal to the number of rounds that can be 10, 12 or 14 rounds, depending on Nb and Nk, where Nb is equal to the block size divided by 32. Rijndael round is composed of 4 transformations [2]:

1. SubByte: S-box substitution provides nonlinearity and confusion.
2. Shiftrow: rotations, provides inter-column diffusion.
3. MixColumn: linear combination provides inter-Byte diffusion.
4. AddRoundKey: round key bytes XOR into each byte provide confusion.

The present work will deal with SubByte and MixColumn transformation functions only.

Substitution Byte Transformation Function

A number of known block ciphers are of substitution-permutation (SP) type. S-boxes are used in such cipher systems as the important nonlinear component. A strong block cipher should be resistant to various attacks, such as linear and differential cryptanalysis [5]. Nonlinear transformations are implemented as lookup tables (S-boxes). In the AES, the S-box generate two transformations in the Galois fields GF(2) and GF(2^8). S-box is a nonlinear transformation where each byte of the State is replaced by another byte using the substitution table [6].

The first transformation: S-box finds the multiplication inverse of the byte in the field GF (2^8). Since it is a algebraic expression, it is possible to mount algebraic attacks. Hence, it is followed by an affine transformation. The affine transformation is chosen in order to make the SubBytes a complex algebraic expression while preserving the nonlinearity property. The both S-box transformations can be expressed in a matrix form as:

$$S\tilde{} = M \bullet S^{-1} + C, \qquad (1)$$

Where the 8 × 1 vector S˜ denotes the bits of the output byte after the S-box transformations; the sign • is multiplication and the sign + is addition in the field GF(2^8). The inverse S-box transformation can be obtained by multiplying both sides of equation (1) by M^{-1} and it performs the inverse affine transformation followed by the multiplicative inverse in GF(2^8) [7]:

$$S^{-1} = M^{-1} \bullet S\tilde{} + M^{-1} \bullet C, \qquad (2)$$

MixColumn Transformation

The mix column transformation operates on the state matrix column-by-column, treating each column as a four-term polynomial. In this transformation, each column of the state matrix is multiplied by a constant fixed matrix where each member is represented in GF(2^8) [8].

The forward mix column transformation, called MixColumns, operates on each column individually. Each byte is mapped into a new value that is a function of all four bytes in the column. The transformation can be defined as the following matrix multiplication on State (Figure 1):

02	03	01	01	*	S00	S01	S02	S03	=	S00’	S01’	S02’	S03’
01	02	03	01		S10	S11	S12	S13		S10’	S11’	S12’	S13’
01	01	02	03		S20	S21	S22	S23		S20’	S21’	S22’	S23’
03	01	01	02		S30	S31	S32	S33		S30’	S31’	S32’	S33’

Figure (1) Multiplication process in MixColumn Transformation

Each element in the product matrix is the sum of products of elements of one row and one column. In this case, multiplications and additions are performed in GF(2^8). The inverse mix column transformation, called InvMixColumns, is defined by the following matrix multiplication:

0E	0B	0D	09	*	S00	S01	S02	S03	=	S00’	S01’	S02’	S03’
09	0E	0B	0D		S10	S11	S12	S13		S10’	S11’	S12’	S13’
0D	09	0E	0B		S20	S21	S22	S23		S20’	S21’	S22’	S23’
0B	0D	09	0E		S30	S31	S32	S33		S30’	S31’	S32’	S33’

Figure (2) Multiplication process in InvMixColumn Transformation

To show that matrix of Figure 2 is inverse to the of matrix of Figure 1, one may check that their product in GF(2^8) is a unity matrix [9].

Modified Substitution Byte transformation in AES

In general, S-Box is nonlinear substitution table that gets number and returns another number. The first step is to generate multi random S-boxes depending on using multi keys that lead to generate S- boxes provided that each one has its inverse associated with it and that represents the first key. The second step is to create a random index distribution of the S-boxes that created in the first step to get more complexity and the same delay time and this represents the second key as shown in Algorithm 1. The second key, if eight different S-Boxes are used then the key space available is 16! different keys. Algorithms 1 and 2 demonstrate this process.

***Algorithm 1**: Modified S-box generation*

***Input**:: Different Eight values as { Rnd_Key[k], Con_c[k] , k=1,2,……..,8 }*

***Output**:: Different* Eight *S-Boxes { $(S\text{-}box[i][j])_k$, $(S^{-1}\text{-}box[i][j])_k$, k=1,2,.....,8 }*

Step1: Select 8 keys *Rnd_Key[k]* that each one generate a unique S-box in condition every key has its inverse to rebuild Inverse S-box

Step 2: Select 8 random values *Con_c[k]* for constant C

Step3: For every key *Rnd_Key[k]* and corresponds constant *Con_c[k]* create its own *S-box[i][j]* using affine transformation as:

$$(S\text{-}box[i][j])_k = Rnd\_Key[k] * mulp[r][c] + Con\_c[k]$$

Where *mulp[r][c]* represent the multiplicative inverse in $GF(2^8)$ as shown in table 1.

Step4: Use the above($S\text{-}box[i][j])_k$ in encryption process .

***Algorithm 2**: Modified SubByte Transformation Function*

***Input** :plaintext Block message { State[Row][Column],Row, Colum=1,2,3,4}*

Different Eight *S-Boxes { $(S\text{-}box[i][j])_k$, $(S^{-1}\text{-}box[i][j])_k$, k=1,2,.....,8*

Key distribution matrix {Key_Enc[4][4]}

***Output:** ciphertext Block message {State[Row][Column],Row, Colum=1,2,3,4}*

For every block to be ciphered in AES algorithm using the new created *S- box[i][j]* as follows:

Step1: For Every Row in State matrix Do

Step2: For every Column in State matrix Do

Step 3: Y=(State[Row][Column])&0x0f;
X=(State[Row][Column]>>4)&0x0f;
Where X, Y represent the index of row and column in each S-Box respectively

Step 4: The state matrix State[Row][Column] can be encrypted using the index of each S-Box *Key_Enc[4][4]* as:
State[Row][Column]=$(S\text{-}box[x][y])_{Row, Column}$

Modified MixColumn four keys transformation in AES

The proposed algorithm to improve mix column transformation uses four keys with dimension 2*2 for each, split the state matrix into four parts each of them is 2*2 dimension as shown in Algorithm 3. The multiplication process is demonstrated in figure (4).

ISSN: 2049-3444 © 2014 – IJET Publications UK. All rights reserved.

***Algorithm 3**: Modified MixColumn Transformation*

Input: Plaintext Block Message P_Block[r][c], r,c =1,…….,4

The mix column key encryption Mix_Key[r][c], r,c = 1,……,4

Output: Ciphertext Block Message C_Blok[r][c], r,c =1,…..,4 .

Step1: Split P_Block[r][c] and Mix_Key[r][c] into four parts each of which of size 2*2, and for each part of Mix_Key[2][2] find its inverse matrix.

Step2: Multiply each part of P_Block[2][2]* Mix_Key[2][2] to produce C_Matrix[2][2]. The multiplication and addition is executed in GF(2^8)

Step3: Reconstruct the C_Block[r][c] r,c=1,2,….,N, from each part of C_matrix[2][2]

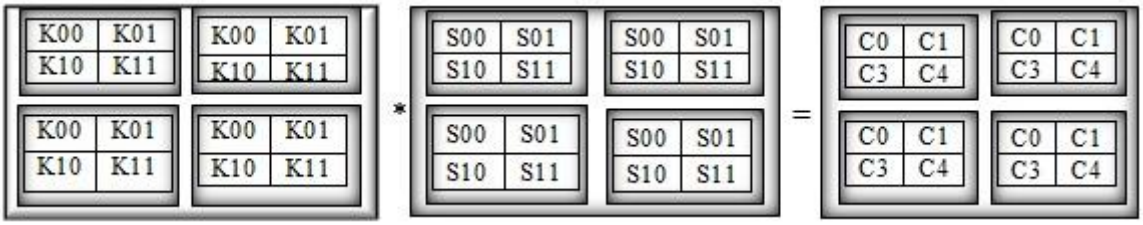

Figure (4) Multiplication process in Modified MixColumn Transformation

Where:

C0= (K00*S00+K01*S10) mod (irreducible polynomial)
C1= (K00*S01+K01*S11) mod (irreducible polynomial)
C3= (K10*S00+K11*S10) mod (irreducible polynomial)
C4= (K10*S01+ K11*S11) mod (irreducible polynomial)

In AES, Mix Column Transformation is the most expensive operation where input matrix is multiplied (Over GF^n) with MDS Matrix (An **MDS "matrix Maximum Distance Separable"** is a matrix representing a function with certain diffusion properties that have useful applications in cryptography). This transformation plays an important role with respect to the wide trail strategy of the cipher. It is a vital component of diffuser part of the cipher. It also guarantees a high number of active S-boxes over a round. The quantitative measure of this transformation is the branch number of the MDS Matrix used. The branch number of 4 by 4 MDS Matrix is 5 (optimal), so as the branch number of AES MDS Matrix. This property ensures that a linear approximation or a differential of one round always involves at least five active S-boxes using any 4x4 MDS matrix in Mix Column transformation. To achieve unchanged throughput with purposed modification, we get a high complexity with less CPU time consuming.

Random Key Permutation Encrypt and Decrypt Process in Stream Cipher

Stream ciphers have several advantages which make them suitable for some applications. Most notably, they are usually faster and have a lower hardware complexity than block ciphers. They are also appropriate when buffering is limited, since the digits are individually encrypted and decrypted. Moreover, synchronous stream ciphers are not affected by error-propagation [10].

Both modified SubByte, InvSubByte transformation functions and modified MixColumn, InvMixColumn transformation functions that use in AES algorithm could be used to implement encryption and decryption process in stream cipher in the proposed algorithm.
Three Functions is used for encryption process in proposed algorithm as follows:

1- Modified SubByte Transformation Function.
2- Block permutation function
3- Modified MixColumn Transformation Function.

The decryption process is in reverse order as follows:

1- Modified InvMixColumn Transformation Function.
2- Block InvPermutation Function
3- Modified InvSubByte Transformation Function
4- The propose algorithm is applied on stream cipher with tiny block (N*N) e.g. (4*4). Another key of size (N*N) is used in Block permutation function. The random key is used to re permutated between pair of tiny blocks.

ISSN: 2049-3444 © 2014 – IJET Publications UK. All rights reserved.

Figure (5) demonstrate the re permutated function while Algorithm 4 demonstrates the encryption process in Block Permutation function.

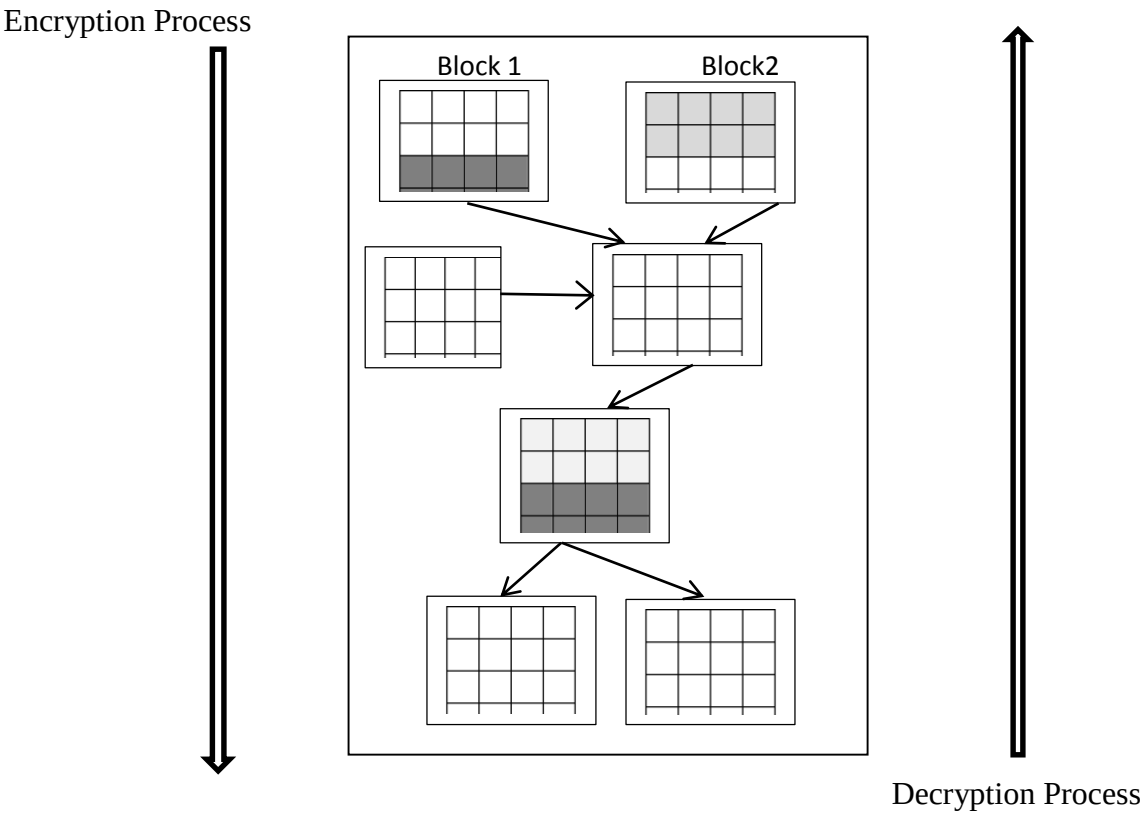

Figure (5) Permutation function details

***Algorithm 4**: Block Permutation*

Input: Two plaintext tiny blocks of size N*N, Block1[r][c], Block2[r][c], r,c=1,2…..,N

Output: New_Block1[r][c],New_Block2[r][c], r,c=1,2……,N.

Step1: For Per_Key[r][c], generate a unique index in each item in Per_Key, where the index range is 1,2,…….,N*N.

Step2: Create Perm_Matrix[r][c], r,c=1,2,…….,N, where

First half of Perm_Matrix= first half of Block2

Second half Perm_Matrix=Second half of block1

Step3: The New_Per_Matrix[r][c], r,c=1,2,…….,N, is constructed from Per_Matrix depending on the Per_Key[r][c].

Step4: The New_Block1[r][c], New_Block2[r][c] is reconstructed again as follow:

First half of New_Block =First half of New_Per_Matrix

3. RESULTS AND DISCUSSION

The present work is aimed to achieve higher complexity compared to original AES keeping the required time for encryption and decryption processes the same. Such complexity was justified by applying the Modified SubByte and MixColumn functions in AES algorithm. The obtained results will be discussed in two steps.

Step1: The results presented in figure (5) have been obtained after apply modified SubByte and MixColumn transformation in AES algorithm. Using the following keys (the values is written in hexadecimal) :

a. Sub-byte key creation (different 8 S-boxes creation) Rnd_Key[16]={0xf1,0x67,0x25,0x85,0xb5,0xA4,0x19,0x4c}
b. *Key distribution matrix which use for random selection of the s-boxes created in previous step.* Key_Enc[16]={5,3,7,6,2,4,1,7,8,2,6,4,1,3,5,8};
c. Four keys of size 2*2 use in MixColumn transformation
d. Add round key (32 bytes)

63	72	79	70
74	6F	67	72
61	70	68	79
69	73	69	6D

Block plaintext

5b	A9	60	6e
00	E0	58	36
C7	42	E7	C7
CE	8F	8A	3B

Block Ciphertext

Figure (5) encryption Result in Modified AES

Step 2: The results presented in Figure (6) are obtained using the proposed algorithm implemented in stream cipher which use both modified SubByte and MixColumn transformation of AES algorithm with addition to the permutation function mentioned in Algorithm 4. The permutated key = [P_Key[4][4]={8,0,14,4,1,3,10,7,6,11,2,15,13,12,5,9}], Rnd_Key and Key_Enc as in (a,b) in step 1

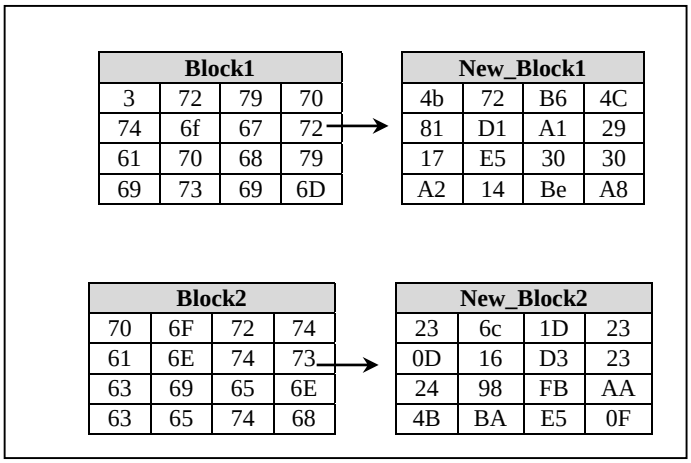

Figure (6) Output results in Key Permutation algorithm

The proposed algorithm has been applied to the variable length of tiny blocks s(2*2, 4*4, 6*6,….), depending on user desire. The modified algorithm has been verified on text data, the results reflect a successful encryption decryption results with considerable time. Table (1) gives a summary for comparison between standard AES, the modified AES and the Random Key Permutation Stream Algorithm with respect to number of S-boxes, complexity, mapping using S-Box etc.

Table(1): Comparison between Standard AES ,modified AES and Random Key Permutation Stream Algorithm

	Standard AES	Modified AES	Random Key Permutation Stream Algorithm
1- Block length	16 bytes	Same	(2*2, 4*4, 6*6,….)
2- Numbers of rounds	10//12//14	Same	No round
3-Key length	16//24//32	Same	Depend on block length

4-MixColum keys	Single	Four	1,2,4 and more depend on key length
5- S-box	Single	Multi (2-16)	Multi, depend on block length
6-Mapping using S-Box	Depend on State matrix	Depend on the index of every byte in state matrix	Depend on the index of every byte in state matrix
7-Single round details	AddRoundKey, SubByte,MixColum, ShiftRow	Same + Create the New S-Boxes + Using four keys with dimension 2*2 for each in ,MixColum	Modified SubByte , Block permutation, Modified MixColumn
8-Complexity	256!	8!*4!*256!	8!*4!*256!* (key length)!

4. CONCLUSIONS

The following conclusion remarks may be highlighted as follows:

1- Two functions have been modified in original AES (SubByte and MixColumn*)* to achieve higher complexity keeping the required time for encryption and decryption processes the same.
2- The proposed algorithm implemented in stream cipher use three functions, two of them are the Modified SubByte and Modified MixColumn transformation of AES algorithm with addition to the permutation function mentioned in Algorithm 4.
3- Tiny blocks of sizes (2*2, 4*4, 6*6,….) have been implemented on the proposed algorithm.
4- The proposed algorithm applied on stream cipher reflects successful results with a reasonable time for encryption and decryption process.

REFERENCES

[1]. N. Singhal1, J.P.S.Raina, "Comparative Analysis of AES and RC4 Algorithms for Better Utilization", *International Journal of Computer Trends and Technology"* PP 177-181, *July to Aug Issue 2011*.

[2]. F. Yousif, A. E. Rohiem, A. Elbayoumy, "A Novel S-box of AES Algorithm Using Variable Mapping Technique" , 13th International Conference on Aerospace Sciences & Aviation Technology, ASAT-13, May 26 – 28, 2009.

[3]. S. O. Sharif, S.P. Mansoor," Performance analysis of Stream and Block cipher algorithms",3rd International Conference on Advanced Computer Theory and Engineering(ICACTE), 2010.

[4]. Ming-Haw Jing, Jian-Hong Chen, Zih-Heng Chen, "Diversified Mixcolumn Transformation of AES", 6th International Conference on Information, Communications & Signal Processing, 10-13 Dec. 2007.

[5]. D. Lambić, M. Živković, "Comparison of Random S-Box Generation Methods, "PUBLICATIONS DE L'INSTITUT MATHÉMATIQUE Nouvelle série, tome 93 (107), 109–115. 2013.

[6]. K. Kazlauskas, J. Kazlauskas, "Key-Dependent S-Box Generation in AES Block Cipher System ",Informatica, Vol. 20, No. 1, 23–34, 2009.

[7]. S.F. Hsiao, M.C. Chen, M.-Y. Tsai, C.C. Lin, "System on chip implementation of the whole advanced encryption standard processor using reduced XOR-based sum-of-product operations", IEEE Proc. Inf. Security., 152(1), 21–30, 2005.

[8]. S. M. Farhan , S. A. Khan, H. Jamal, "An 8-bit systolic AES architecture for moderate data rate applications", Microprocessors and Microsystems 33) 221–231, 2009.

[9]. W. Stallings. ,"Cryptography and Network Security Principles and Practic", Fifth Edition, Prentice Hall. 2012.

[10]. G.S. Vernam, "Cipher printing telegraph systems for secret wire and radio telegraphic Communications", Journal of the American Institute of Electrical Engineers, Vol. 55, 109–115, 1926.

ISSN: 2049-3444 © 2014 – IJET Publications UK. All rights reserved.