A Variable Length Data Encryption Technique using Square Grid Transposition with Key Wrapping

Dr Allam Appa Rao[†], *Dr. M. Shashi*[††], *P. Pandarinath*[†††] *and Sai Pradeep Chakka*[††††]

[†]*Vice Chancellor, JNTU Kakinada, Kakinada, Andhra Pradesh*
[††]*Head of the Department, Dept. Of CS&SE, Andhra University, Visakhapatnam- 530 003, Andhra Pradesh*
[†††] *Associate Professor, CSE, Sir C R Reddy College of Engineering Eluru-534001, Andhra Pradesh*
[††††] *B.Tech, CSE, Sir C R Reddy College of Engineering Eluru-534001, Andhra Pradesh*

Summary
This paper has a new approach for encryption of various file formats like text, image, audio and video with key wrapping is proposed. The binary data of file is divided into equal sized blocks called as grids. The bit stream of each grid is taken and Square grid transposition is applied. A variable length key based on the grid size considered, say 160 bits for 32-sized grid, 384 bits for 64-sized grid is generated. The key is wrapped up with some public key algorithm say RSA for secret key transposition so that intruder cannot identify. For decryption the reverse grid transposition and private key is required. The included session key is obtained by decrypting the wrapped key with receiver's private key.

Key words: Cryptography, Cipher Methodology, Grid Transposition, Circle generation, key wrapping.

1. Introduction

Today the prominence of internet in day to day life has increased a lot. Various transactions in defense, file transfers in an organization internally requires network [1] security. With the availability of internet, many intruders across the world can access our data. In order to rescue our data from intruders we need CRYPTOGRAPHIC [2] techniques. Cryptography [2,6,7,8] is conversion of original data into some modified form of data called cipher. Various algorithms [6,7,8,9,10] have been proposed till now and each has their merits and demerits. As a result researchers are working in the field of cryptography to enhance the security further. In this paper a new approach is proposed where the file is considered as a stream of bits and constructed as various grids. The technique transforms each grid into encrypted grid by applying bit permutations [3,4,5]. The key generated is also wrapped to get encrypted key. The efficiency of this method is that it supports a variable length grid, secure variable length key. The file can be recovered using the reverse algorithm.

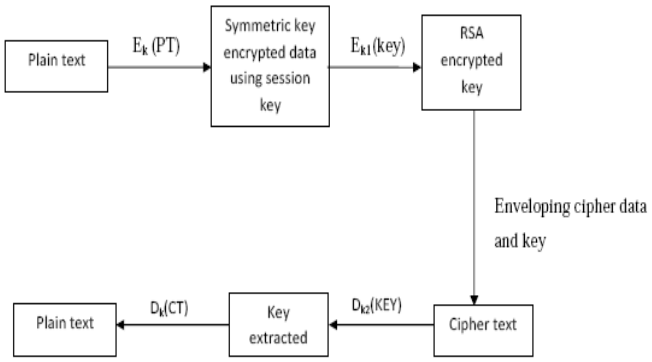
2. Methodology

Step 1: A square grid of required size is constructed by taking the binary data from source file.
Step 2: Now grid transposition is applied by reading data as various circles starting from the center of the grid to the bottom left at various levels and writing it down on columnar basis from top left to bottom right.
Step 3: A secret key that varies with each session as combination of 0's and 1's is generated based on the grid size, say 160 – bit for 32 – sized, 384 – bit for 64 – sized etc., to produce decimal sequence. Accordingly columnar transposition is done on the grid.
Step 4: A new grid is generated after transposition.
Step 5: The new grid is converted into ASCII sequence and written to another file called encrypted file.
Step 6: Steps 1 to 5 are repeated until the total file is formed into grids and encrypted. Padding with 0's is done in grid formation deficiency.
Step 7: Key generated for each file is encrypted with the public key of sender using RSA Algorithm. This technique of hiding the key is called key wrapping.
Step 8: The encrypted key is then divided into various blocks and appended to file.

Hence the new encrypted file with wrapped key is generated.

Detailed Explanation

The operational structure of the technique is as follows.



This technique forms the raw data from the file into various grids of sizes say 32, 64, 128… by reading the binary data as bytes. A grid is nothing but a 2-dimensional array of equal size. The size of the grid is fixed for a file for each session. If the grid is not filled with the data from the file can be pad the grid by adding 0's at the end. The grid thus formed is encrypted starting from the center of the grid to the bottom left. This process is done for all the grids formed. Now a sequence of bits that varies with grid size as a combination of 0's and 1's is generated. The key generated at each session for a given file is different and size of key varies with the grid size. The security of the data is increased and also the complexity of key generated. Now the session key is encrypted using the RSA algorithm. This makes the key generated secure from intruders. Longer the size of the grid, longer is the size of the key.

2.1 Encryption:

Broadly our technique can be divided into 3 phases

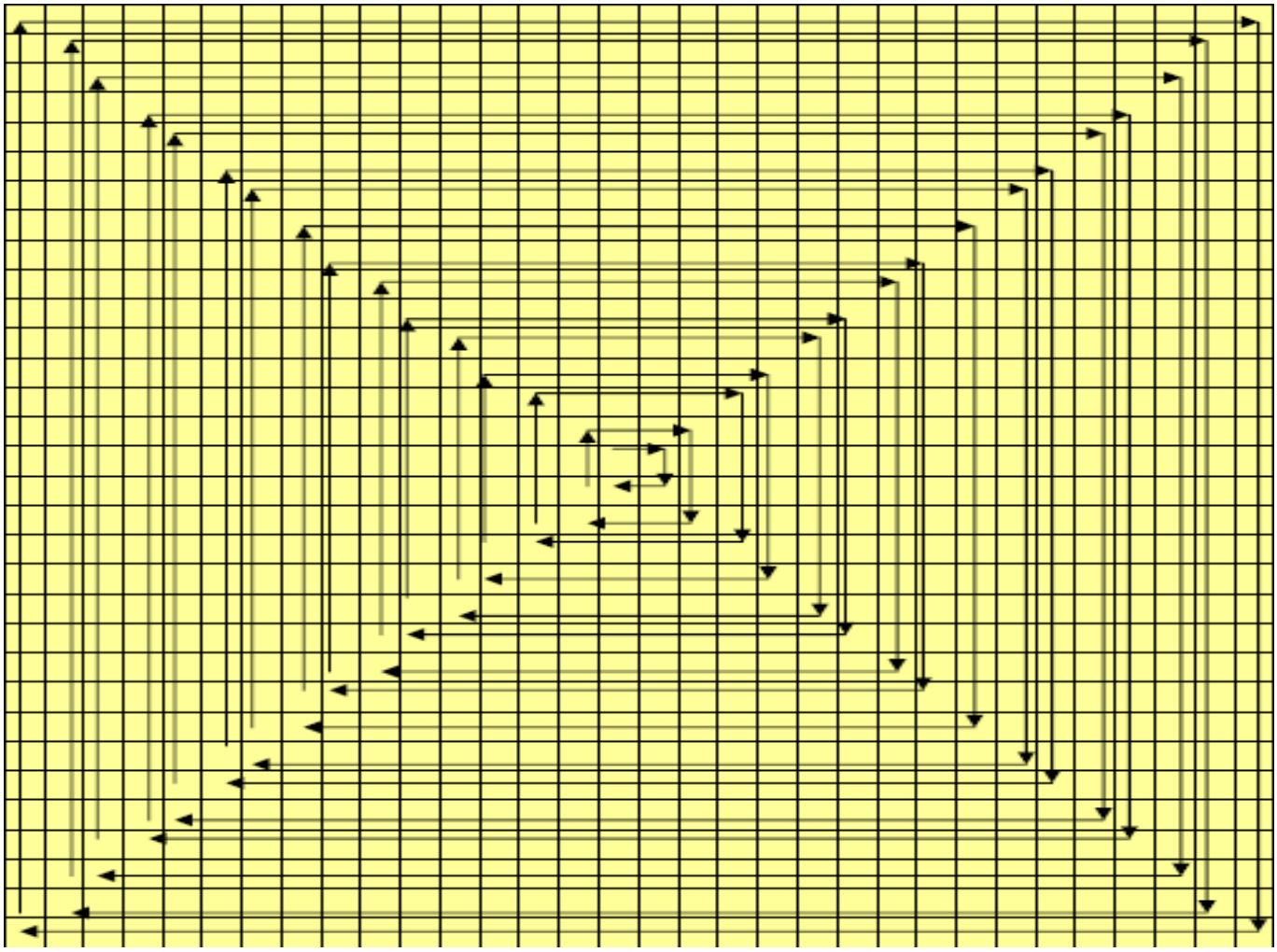
1) Grid transposition of data.
2) Columnar transposition based on session key.
3) RSA encryption of key.

2.1.1 Grid transposition:

Let us suppose the size of grid is say 32. The grid transposition is as follows:

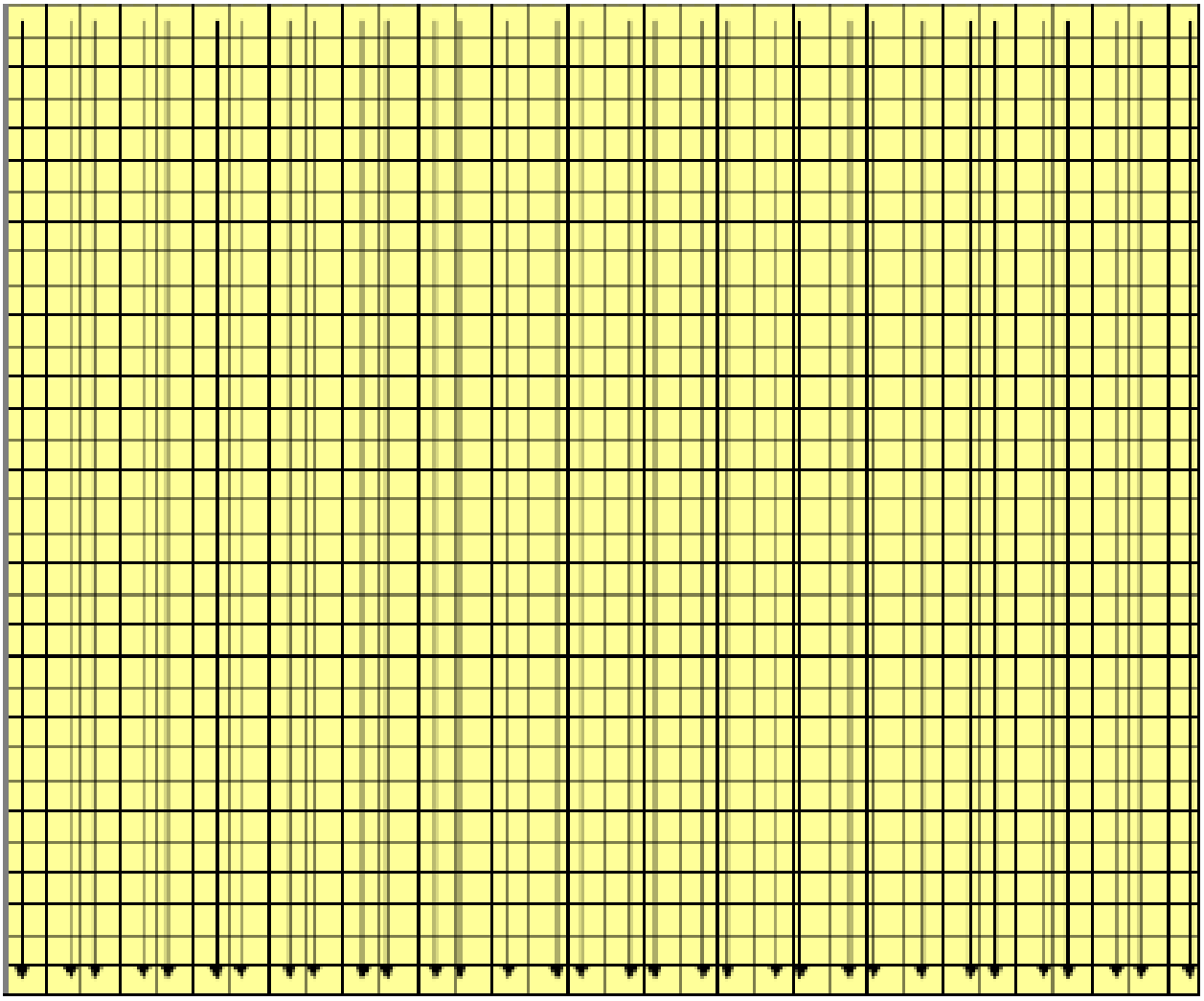
Now 1024 bits (32X32) are taken from the file, i.e. 128 bytes are read from the file and is converted to binary string by taking the ASCII values of each byte. This is arranged as a grid starting from top left row by row and ending at bottom right. Now the data is read from the center as various circles generated at each level. The reading can be shown as belo

32X32 Grid Reading



The data read as above format is transposed by writing the data into a new equal sized grid starting from the top left to bottom right column by column. This is as follows.

32X32 Grid Writing



Now the raw data is applied a form of transposition where the intruder cannot identify the hidden data.

2.1.2 Columnar transposition:
This includes the generation of a sequence of 0's and 1's whose size varies with the grid size. The generated sequence is then converted to a decimal sequence by grouping the bits together so that all the non negative indexing for the grid is generated.

Discussion of the key generation in detail:

Suppose if the grid size is equal to 32, then all the non negative sequencing of grid size 32 has to be genrated. Each index requires atmost 5 bits to represent. Now the size of the key should be 32X5= 160 bits such that by grouping any 5 bits we get the all the indexing and no indexing once occurred shouldno repeat in the key. The no of possibilities of keys with grid size=32 is 2.6313083693369355e+35.

Now suppose the grid size is equal to 64, then all the non negative sequencing of grid size 64 has to be genrated. Each index requires atmost 6 bits to represent. Now the size of the key should be 32X6= 384 bits such that by grouping any 6 bits we get the all the indexing and no indexing once occurred shouldno repeat in the key. The

no of possibilities of keys with grid size=64 is 1.2688693218588414e+89.

Suppose if the grid size is equal to 128, then all the non negative sequencing of grid size 128 has to be genrated. Each index requires atmost 7 bits to represent. Now the size of the key should be 128X7=896 bits such that by grouping any 7 bits the algorithm derive the indexing and no indexing once occurred shouldnot repeat in the key. The no of possibilities of keys with grid size=128 is 3.856204823625808e+215.

Accordingly columnar transposition is done based on the session key generated by reordering the elements of each row of grid as per the new key. Now that the grid is converted back to a binary string and is divided into 8 bits each and is written to a new file. The same process is applied for all the grids. Hence the generation of cipher i.e. encryption of data of file is completed.

RSA encryption of key:

The key generated is encrypted based on the public key given to the sender. Instead of encrypting the entire key at a time, it is first divided into N/2 decimal parts where N is grid size. Now each part is considered and is encrypted by using the RSA algorithm. Each part produces 4 bytes of data. Hence a 32 sized grid has an encrypted key written to the file of size 512. Similarly for 64-sized grid it is 1024 and so on. This process of encrypting the key is called encapsulation.

2.1.3 RSA algorithm:

1. Assume any two prime numbers P , Q
2. Calculate N = P * Q
3. Calculate Z= Φ(N) = Φ (P * Q)
   = Φ (P) * Φ (Q) (According to modular arithmetic)
   = (P-1) * (Q-1)
4. Assume a value ‘e’ i.e. relatively prime to Z and e < Z and gcd (e , Z) = 1
5. Calculate d, such that e * d ≅ 1 (mod Z) ≅ 1 (mod Φ(N))
6. Cipher (C) = (m ^ e) mod N
   Plaintext (m) = (C ^d) mod N
   Thus our key generated is both complex and secure.

2.2 Decryption:

For decryption the reverse process is applied. The individual bytes from the file are combined and the combined result is decrypted using the private key at receiver’s side. Hence the session key is obtained. The reverse process is done i.e. columnar re-transposition and anti grid transposition to get the plain data.

3. Example

Now consider an example to illustrate our technique easily. Let us suppose the plain text in a .txt file as “COMPUTER SCIENCE”. Now the ASCII value representation of each character is as follows:

Character	ASCII	Bit Representation
C	67	01000011
O	79	01001111
M	77	01001101
P	80	01010000
U	85	01010101
T	84	01010100
E	69	01000101
R	82	01010010
space	32	00100000
S	83	01010011
C	67	01000011
I	73	01001001
E	69	01000101
N	78	01001110
C	67	01000011
E	69	01000101

The grid can be formed as follows.

0	1	0	0	0	0	1	1	0	1	0	0	1	1	1	1	0	1	0	0	1	1	0	1	0	1	0	1	0	0	0	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	1	0	0	1	0
0	0	1	0	0	0	0	0	0	1	0	1	0	0	1	1	0	1	0	0	0	0	1	1	0	1	0	0	1	0	0	1
0	1	0	0	a	1	0	1	0	1	0	0	1	1	1	0	0	1	0	0	0	0	1	1	0	1	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PADDING

Now the transposition is applied as follows:

Actual Position	Transposed position
(15,15)	(0,0)
(15,16)	(1,0)
(16,16)	(2,0)
(16,15)	(3,0)
(16,14)	(4,0)
(15,14)	(5,0)
(14,14)	(6,0)
(14,15)	(7,0)
(14,16)	(8,0)
(14,17)	(9,0)
(15,17)	(10,0)
(16,17)	(11,0)
(17,17)	(12,0)
(17,16)	(13,0)
(17,15)	(14,0)
(17,14)	(15,0)
.	.

CIRCLE 1

CIRCLE 2

.	.
.	.
(18,15)	(0,1)
(18,14)	(1,1)
.	.
.	.
.	.

(31,2)	(29,31)
(31,1)	(30,31)
(31,0)	(31,31)

Hence the data is transposed as above. All the bits are shifted to some new location.

The data obtained after transposition is as follows:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

If the length of the key is 160 bits, the algorithm generates a session key such that all the indexing for 32-sized grid is generated. The complete generation process is as follows.

00000	11101	01110	00110	01100	01111
0	29	14	6	12	15
10101	01000	10011	01011	11010	10111
21	8	19	11	26	23
01010	10010	00111	10100	00010	00011
10	18	7	20	2	3

00001	00101	10001	01101	10000	11100
5	17	13	9	16	28
11110	00001	10110	11000	00100	11011
30	1	22	24	4	27
11111	11001				
31	25				

Now a columnar transposition based on the key is done in each row. That is the 0^{th} row is placed in 0^{th} position, 29^{th} row in 1^{st} position, 14^{th} row in 2^{nd} position and so on. The result is

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Now the bit sequence that is generated is as follows.

Bit Representation	ASCII	Character
00000000	0	null
00000000	0	null
00000000	0	null
00100000	32	space
00000000	0	null
00100000	32	space
00000000	0	null
00000000	0	null
00000000	0	null
00000000	0	null
00000000	0	null
00000000	0	null
01000000	64	@
00100000	32	space
00000000	0	null
10000000	128	NPC
00000000	0	null
.	.	.
.	.	.
.	.	.
00000000	0	null
00000000	0	null
00000000	0	null

This is written back to the file.

4. Result

The desired technique has been implemented successfully using JAVA programming language and various files have been experimented with varying file sizes and grid size.

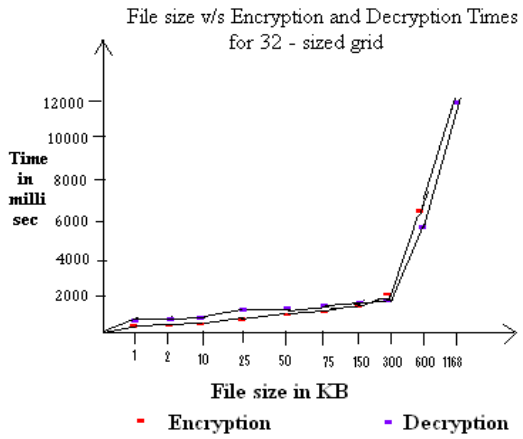
A. With grid size=32

The technique is implemented on .txt files. The same encryption and decryption times continue though it may be applied for other types of file as the algorithm is reading the binary data from file. The results are tabulated as follows

File size v/s Encryption time for .txt files with 32 – sized grid

Source File name	File size before encryption (in KB)	File size after encryption (in KB)	Encryption time (in milli sec.)	Decryption time (in milli sec.)
test01.txt	1	1	13	15
test02.txt	2	2.08	31	47
test03.txt	10	10	80	138
test04.txt	25	25	199	278
test05.txt	50	50	496	576
test06.txt	75	75.1	732	824
test07.txt	150	150	1512	1523
test08.txt	300	300	2944	2839
test09.txt	600	600	6183	6001
Test10.txt	1168	1169	12142	11230

The graph that compares the encryption decryption time is as follows for 32 sized grid



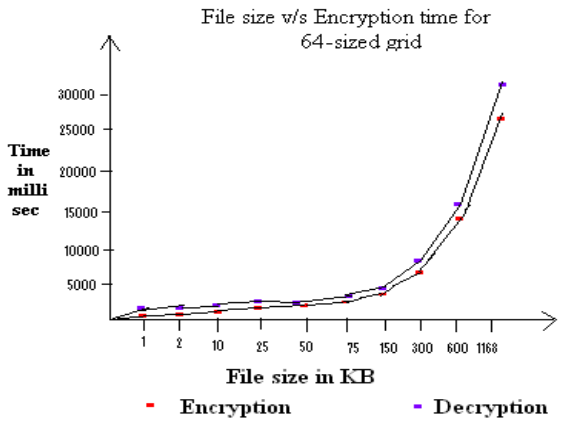
B. *With grid size= 64*

With the change of grid size the complexity in key changes and the time taken to form the grid changes hence the encryption and decryption time also increases.

File size v/s Encryption time for .txt files with 64 – sized grid

Source File name	File size before encryption (in KB)	File size after encryption (in KB)	Encryption time (in milli sec.)	Decryption time (in milli sec.)
test01.txt	1	1	48	108
test02.txt	2	2.08	63	139
test03.txt	10	10	234	310
test04.txt	25	25	545	718
test05.txt	50	50	1234	1195
test06.txt	75	75.1	1652	1794
test07.txt	150	150	3545	4003
test08.txt	300	300	6993	7654
test09.txt	600	600	14238	15511
Test10.txt	1168	1169	26936	30025

The graph that compares the encryption decryption time is as follows for 64 sized grid



Let us compare the efficiency of our algorithm with DES algorithm with a block size of 64. the various encryption and decryption times are as follows:

File size v/s Encryption time for .txt files with DES algorithm for block size=64

Source File name	File size before encryption (in KB)	File size after Encryption (in KB)	Encryption time (in milli Sec.)	Encryption time (in milli Sec.)
test01.txt	1	1	151	148
test02.txt	2	2.02	333	352
test03.txt	10	10	621	608
test04.txt	25	25	2333	2123
test05.txt	50	50	3865	3489
test06.txt	75	75.05	4889	4695
test07.txt	150	150	8666	8652
test08.txt	300	300	16385	16357
test09.txt	600	600	36333	36158
test10.txt	1168	1168	57333	55373

Thus we can explain that our algorithm is efficient when compared to DES both in case of time taken and key size.

5. Conclusion

The proposed technique has been implemented for text files, image files, audio and video files with variable grid sizes of length 32, 64, and 128. It is being executed efficiently for 32 and 64 but there is a delay with grid size=128 as more time is required to form a big grid of size 128. Due to some padding (with 0) and enveloping of key the cipher file size varies from the original but not to a greater extent. Since the key is generated randomly with varying size based on the grid size differently for different sessions the complexity of key is enhanced. Due to key wrapping its security is further enhanced. Thus the proposed technique is implemented effectively.

Further enhancement of this technique can be the implementation of other asymmetric key cryptographic algorithms in place of RSA.

6. References

[1.] I. Tanenbaum, AS.: "Computer Networks" 2nd edition, Prentice Hall, London. 1989

[2.] Stinson, D.R. : "Cryptography Theory and Practice", CRC Press, London, 1995

[3.] Z. Shi and R. B. Lee. Bit permutation instructions

for accelerating software cryptography. In Proceedings of the 11th International Conference on Application-Specific Systems, Architectures and Processors, pages 138-148, July 2000.
[4.] Z. Shi and R. B. Lee. Sub word sorting with versatile permutation instructions. In Proceedings of the International Conference on Computer Design(ICCD 2002),pages 234-24 1,September 2002.
[5.] Z. Shi and R. B. Lee. Implementation complexity of bit permutation instructions. In proceedings of the Asilomar Conference on Signals, Systems and Computers, November 2003.
[6.] Network security Essentials Applications and Standards, William Stallings, Pearson Education, New Delhi.
[7.] Cryptography and Network security, 2nd Edition by Atul Kahate. Tata Mc- Graw-Hill Publications, New Delhi.
[8.] Security Requirements for Cryptographic Modules, FIPS PUB 140-1, 1994 January 11
[9.] National Institute of Standards and Technology. *Data Encryption Standard*. FIPS PUB 46-2. December 30, 1993.
[10.] Special Publication 800-12: An Introduction to Computer Security - The NIST Handbook.

Dr. Allam Appa Rao received his B.Sc.(M.P.C.) in 1967 from Andhra University. He completed M.A. in Economics: Mathematical Economics and Econometrics, from Andhra University. He received the Ph.D. in Computer Engineering from Andhra University. in 1984. He is serving as Vice Chancellor for JNTU Kakinada. He has presented 150 research articles. Dr Allam holds two patents as a co inventor for Method(s) of stabilizing and potentiating the actions and administration of brain-derived neurotrophic factor (BDNF) patent Number 20080234197 dated 25th September 2008 (Refer http://www.faqs.org/patents/inv/83318) and Method(s) of preventing, arresting, reversing and treatment of atherosclerosis Patent Number 20080279925 dated 13th November2008(Referhttp://www.faqs.org/patents/inv/171637). He achieved the best Researcher in Engineering in recognition of commendable record of research in Engineering, Andhra University, Visakhapatnam, India, 2003.

Dr. M. Shashi completed her B.E. in EEE, and M. E. in Computers Engineering. She received Doctorate in Engineering. She is working as Head of the Department in Computer Science and Systems Engineering, Andhra University, Visakhapatnam. She has a total experience of 24 years. She had presented 17 international journals and 3 national journals. She had presented papers in 10 international conference and 8 national conferences. She had attended 7 international and 25 national conferences. She guided 85 M.Tech projects. She had received AICTE Career Award for Young Teachers in 1996-1998 at National Level and Best Ph.D. Thesis Prize for 1994 & 1995 at University Level. Her areas of specializations include Data Warehousing & Mining, AI, Data Structures, Soft computing and Machine Learning.

P. Pandarinath is a B.Tech in CSE in 1994 and an M.Tech in CSE in 2001. He is presently pursuing his doctorate. He has a total experience of 15 years. He is working as an Associate Professor in Sir C R Reddy college of Engineering. He has successfully completed 80 projects and guided 100 projects. His areas of interest include Network Security and Cryptography, Advanced Computer Architecture, DataBase Management System, Computer Organization, Computer Networks.

Sai Pradeep Chakka, completed his B.Tech in Computer Science and Engineering at Sir C R Reddy College of Engineering, Eluru, Andhra Pradesh, India. He achieved Gold Medal for being 1st in the college for 2 times. He had presented 5 Papers in Various National Level Technical Symposium held at Engineering Colleges in Andhra Pradesh and got many prizes. He is very good technically and won prizes in this field. He is Interested in Programming, Cryptographic Algorithms for Cryptography, Computer and Network Security.