International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 25081-25093

http://www.ripublication.com

Implementation of Parallelism on Block Cipher Modes Using Des In Lab View

Dr. K. Raja Sekhar[1], V. Sankeerthanreddy[2], D. Priyanka[2], S.GopiKrishna[2], B.Vijaya Sree[2]

[1] Professor, Department of CSE, K L University, Vaddeswaram, Guntur, AP, India
[2] Students, Dept. of ECE, K L University, Vaddeswaram, Guntur, AP, India

Abstract

The Computational Performance of implementing cryptographic block ciphers can be improved by exploiting parallel execution in Lab View (Laboratory Virtual Instrument Engineering Workbench). In this paper, we present a parallel processing method for block cipher algorithm: Data Encryption Standard (DES) based on software implementation in lab view. Test of common Lab view Program show on average, a 20 to 35 percent improvement in execution time in most Lab view code.
Optimizing a Lab view program for a multicore computing is the best way to ensure the maximum speed up when running code on the multicore computing systems. The proposed approach can be implemented by using String To Byte Array Function, U8 To TF Array, Split 1D Array Function, Index Array Function, Array Subset Function, Build Array Function. In this we will give a file input which contains large amount of plane text which is divided each 8byte in table like manner with same key, when this file is given as input several 8byte of plaintext will be executed parallel.

Keywords: Block ciphers, Cryptographic, DES, Labview, Parallelism.

Introduction

There are two main types of cryptography in use today they are symmetric or secret key cryptography and asymmetric or public key cryptography. In our paper we have used data encryption standard algorithm (DES). It is a symmetric key cryptography. DES algorithm uses the same key for both encryption of the plaintext and decryption of the cipher text. The keys may be similar or there may be a little transformation in between the two keys. The block size of DES is 64 bits and it uses a 56-bit key during execution (the remaining 8 bits are used for parity checking and are neglected).It uses a 16 round Feistel cipher When used for communication, both sender and receiver must know the same secret key. The DES is also be used for unique user encryption, such as storing files on a hard disk etc. In Des the plain text to be encrypted is divided into 64 bit blocks. Every single block is encrypted using the secret key into a 64-bit cipher text by means of permutation and substitution. This process involves 16 rounds of operation and can be implemented in four different modes, either encrypting blocks individually or making each cipher block dependent on all the previous blocks.

Initially the plain text to be encrypted is sent into the initial permutation block. After finishing the initial permutation the block is broken into two halves right and left each 32 bits long. Then there are 16 rounds of similar operations called function **f** in which the data is combined with the key in each round. After the execution of the sixteenth round, the right and left halves are combined and a final permutation which is the inverse of the initial permutation which finishes off the algorithm.

The initial permutation takes place before the round one; it transforms the input block. For example the initial permutation moves the 58^{th} bit of the plain text to the first bit position, 50th bit to 2^{nd} bit position, 42^{nd} bit to 3^{rd} position and so on. The initial and final permutations do not effect DES security. First, the 64 bit DES key is shortened to 56 bits by neglecting every eighth bit by using a compression permutation, these bits are used for parity checking to make sure that the key is error free.

After the generation of 56 bit key a different 48 bit sub key is extracted for each of the 16 rounds of the DES. Initially, the 56 bit key is divided into two 28 bit halves. Then these halves are left shifted circularly either one or two bits depending upon the round. Later Right half of the data is sent to expansion permutation, it expands the right half of the data from 32 bits to 48 bits. After compression the key is xored with the expanded block and the 48 bit result is moved to substitution operation. The substitution is performed by 8 substitution boxes. Each s box has a 6 bit input and a 4 bit output.

The 32 bit output of the s- box substitution is permuted according p-box. This permutation gives each input bit to an output position, no bits are used twice no bits are ignored. The final result of the p box permutation is xored with the left half of the initial 64 bit block. Then the left and right halves are exchanged and another round begins. The final permutation is the inverse of the initial permutation, left and right halves are not switched after the last round of DES, they are concatenated and are used as input to the final permutation.

The same algorithm can be used for both encryption and decryption. It is possible to use the same function to encrypt or decrypt a block except the difference that the keys must be used in reverse order. That is if the keys used in encryption for each round are K1, K2, K3…K16 then the decryption keys are K16….K2, K1. The algorithm that generates the key used for each round is circular as well.

Parallelism in DES block cipher for encryption and decryption can be implemented only in Electronic Code Book (ECB) and in Counter Mode (CTR) for which output is not depended on next block input. For other modes like Cipher Block Chaining (CBC) Encryption cannot parallizable but decryption can be parallizable in which output of one block is not depended on other block. In Cipher Feedback (CFB) encryption cannot parallizable but decryption can. In Output Feedback mode (OFB) both encryption and decryption cannot parallizable.

Lab view (Laboratory Virtual Instrument Engineering Workbench) is a system design platform and development environment for a visual programming language from national instruments.

LabVIEW is the mostly used tool which is highly productive for developing environment in creating custom applications that deal with real-world data or signals in fields such as science and engineering. The final result of using this LabVIEW tool is that more complex projects can be completed within a short span of time with less number of people involved and with increased quality.

LabVIEW is distinctive because a wide variety of tools can be used in a single environment, assuring the compatibility as it is simple to draw wires between functions. LabVIEW itself is a software development environment which contains plenty of components, many of which are required for any type of test, measurement, or control application.

Lab View Components

String Length Function:
It returns in length the number of characters (bytes) in string as shown in fig1.

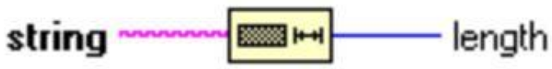


Figure 1: String length function

String to Byte Array Function:
It converts a string into an array of unsigned bytes. Every byte in the array has the ASCII estimation of the corresponding character in the string as shown in fig 2.

Figure 2: String byte function

U8 to TF Array:
The input string is sent to U8 Array to TF Array which Converts unsigned byte array to Boolean array as shown in fig 3. It is not the inbuilt block, when you double click on it you will get the inner block diagram.

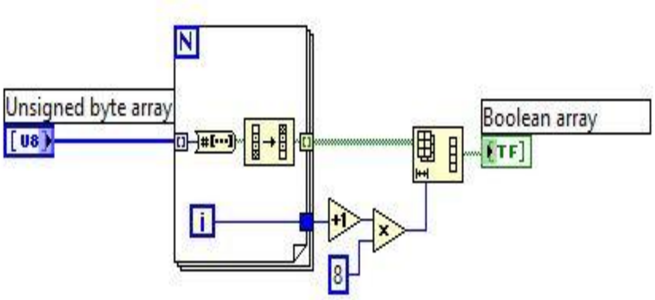


Figure 3: U8 to TF Array

Unsigned byte array is the corresponding ASCII values of each byte when it is passed to number to Boolean array function it converts integer to a Boolean array. Later it is passed to Reverse 1D array it reverses the corresponding index elements. Then corresponding array elements are passed to reshape array with dimension size as other input it changes for every byte array for first byte dimensional size will be 8 as i=0,for second byte array dimensional size will be 16 as i=2 and so on. Output will be corresponding Boolean array as DES excepts.

Permute bits:

The initial permutation of data input which takes place before round one; it transforms the input block. For example the initial permutation moves the 58th bit of the plain text to the first bit position, 50th bit to 2nd bit position,42nd bit to 3rd position and so on. it is not the inbuilt block we have to write block diagram to change the bit positions, when you double click on it you will get inner block diagram.

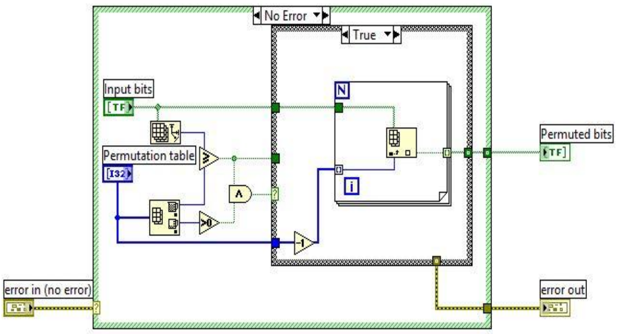


Figure 4: Permute Bits

Split 1D Array Function:

It partitions array at index and returns the two segments with the element of index at the beginning of the second array as shown in fig 5.

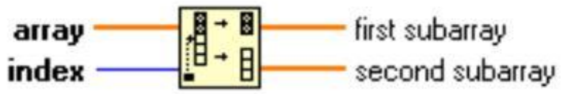


Figure 5: Split 1D Array Function

Select function:
Returns the value wired to the t input or f input, depending on the value of s. If it is true, this function returns value wired to t. If s is false, this function returns value wired to f as shown in fig 6.

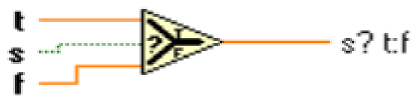


Figure 6: Select Function

Build Array Function:
Concatenates multiple arrays or appends elements to an n-dimensional array as shown in fig 7. Array can be any n-dimensional array or scalar component.

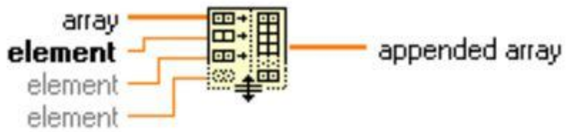


Figure 7: Bulid Array Function

TF to U8 array:
Converts Boolean array to unsigned byte array, U8 is unsigned 8byte array as shown in fig 8. It is not the inbuilt block, it is the sub module which contains inner block diagram.

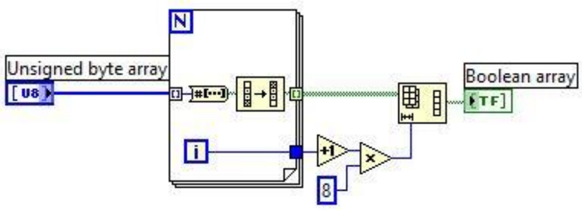


Figure 8: TF to U8 array

Byte Array to String Function:
Changes over an array of unsigned bytes representing ASCII characters into a string as shown in fig 9.

Figure 9: Byte Array to String Function

Number to Boolean Array Function:
Converts an integer or fixed-point number to a Boolean array as shown in fig 10.

Figure 10: Number to Boolean Array

Reverse 1D Array Function:
Reverses the order of the elements in the array as shown in fig 11. If array has n elements, array [0] becomes reversed array [n-1], array [1] becomes reversed array [n-2], and so on.

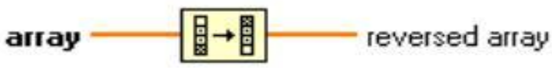


Figure 11: Reverse 1D Array

Reshape Array Function:
Changes the dimensions of an array according to the values of dimension size 0...m-1 as shown in fig 12. This function reads the array data in memory from left to right, row by row and populates the reshaped array the same way.

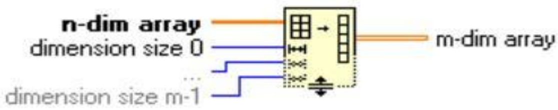


Figure 12: Reshape Array

Array Size Function:
Returns the number of elements in each dimension of array as shown in fig 13.

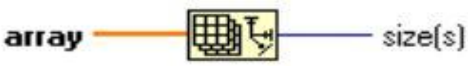


Figure 13: Array Size Function

Array Max & Min Function:
Returns the maximum and minimum values found in array, along with the indexes for each value as shown in fig 14.

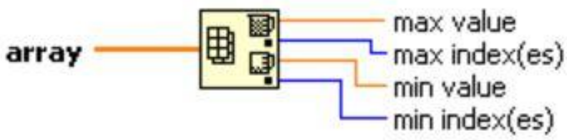


Figure 14: Array Max &Min

Index Array Function:
It returns the component or subarray of n-dimension array at index as shown in fig 15**.**

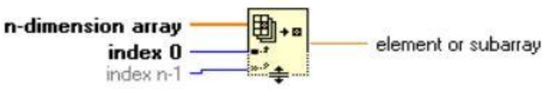


Figure 15: Index Array

Array Subset Function:
Starting with index it returns a portion of array and containing length elements as shown in fig 16.

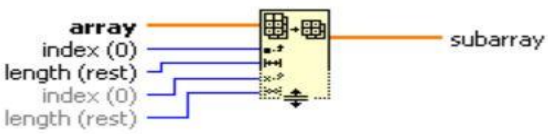


Figure 16: Array Subset

Cipher function:
Cipher function which consists of expansion permutation,s-box substitution, p-box permutation is explained from fig 17.

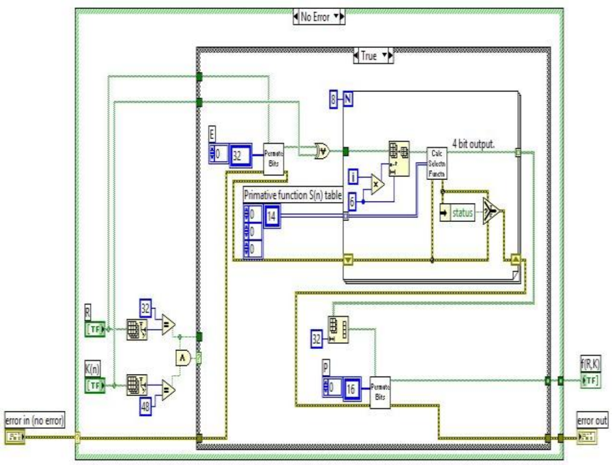


Figure 17: Cipher Function

First right half of the data and key bits are passed into Array Size Function if right half of the data is equal to 32 bit and key size is equal to 48 bit then case selector will be true and inner block will be executed else it shows error message. Later the right half is passed into expansion permutation this operation expands the right half of the data, from 32 bit to 48 bit based on the index input.

This operation changes the order of the bits as well as repeating certain bits, it is known as an expansion permutation. This operation has two purposes: it makes the right half the same size as the key for the xor operation and it provides a longer result that can be compressed during the substitution operation. After xor operation with the key then bits are passed into array subset function which sends 6 bits at a time to the selection function based on index. Initially i=0 after multiplication index will be 0 and length will be 6 so it sends six bits at a time for i=1 it sends six bits from 6 to 12 indexed bits and so on to selection function which are performed by eight substitution boxes.

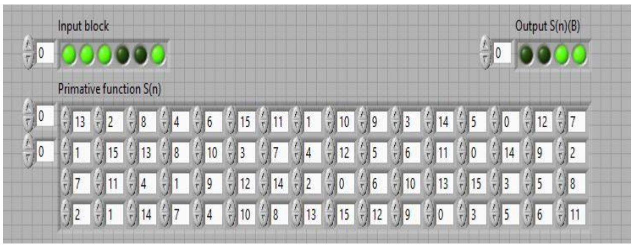
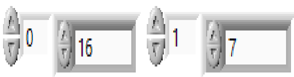


Figure 18: Selection Box

Each S-box has a 6-bit input and a 4-bit output so a for loop with N=8 is taken from fig 18. For example after 8 rounds if the input 6 bits are 111001 first and last bits are 11 which selects row(i.e. last row) middle four bits are 1100 which is 12th column after matching last row and 12th column from below table output will be 3(0011)of 4 bits in this way 8 S-box operations are performed.

After the selection function the input bits are passed to reshape array function with dimension size of 32 it reorder bits into one 32 bit 1D array. Later 32 bit output of the S-box is permutes according to P-box.DES has the P-Box permutation table shown above shows the position to which each bit moves. Bit positions are in the index input.



and so on.

DES Implementation In Lab View

Implementing the DES algorithm in lab view involves several steps that are described below:

1. The first step is giving the information data, 64 bit block of data to be encrypted is passed in as an 8 byte character. Then a string to length converter is used to find the length of input string, if it is 8byte then case selector will be true and processed further else case selector will be false inner block will not be executed and shows an error as input string is not 64bit length.
2. Then input string is sent to U8 Array to TF Array which Converts unsigned byte array to Boolean array as shown in fig 3.
3. The next step is initial permutation of data input which takes place before round one; it transforms the input block .it is explained by permute bits as shown in fig4.
4. After the initial permutation the entire 64 bit must be divided into left half and right half for these input array is passed into Split 1D array function it partitions array at index and returns the two segments with the element of index at the beginning of the second array. Array should be divided equally so index is 32 bit.it divides into first subarray and second sub array each 32 bit as shown in fig5.
5. The next step is the key generation, in this step the user will provide a secret key which is in text form. 64 bit Key (password) used to encipher or decipher data is passed in as an 8-character string. Then it is passed to String to Byte Array Function which converts string into an array of unsigned bytes and then to U8 array to TF array to produce a Boolean array. In each byte of the key, the initial seven bits are data bits and the eighth bit is a parity bit, these bits can be used as parity check to ensure the key is error-free. The same Key must be used for both enciphering and deciphering data. After the 56-bit key is extracted, a different 48 bit sub key is generated for each of the 16 rounds of DES. First, the 56-bit key is divided into two 28-bit halves. Then, the halves are circularly shifted left by either one or two bits, depending on the round.
6. Later right half array is passed into cipher function with key as other input the function of these block is explained from fig 17.
7. Later cipher function output is xored with the left half of the input then the result becomes right half and the initial right half is brought to left side, this procedure is repeated for 16 rounds then case selector will be true after 16 rounds and passed through loop through shift registers.
8. In this step, after 16 rounds of execution both left and right are passed through Build array function which concatenates both arrays.
9. The final permutation is the inverse of the initial permutation .the concatenated block is used as input to the final permutation. The shift of bit positions of default inverse permutation is given as index input.
10. The output of the inverse permutation is Boolean array it should be converted into unsigned 8 byte array so it is passed through TF to U8 array. In the last step the output of UB array is given to data output which returns Enciphered or Deciphered data as an 8 character string.
11. Decryption also follows the same algorithm, except that the input to decryption is a cipher text. Des implementation in labview is shown in fig 19.

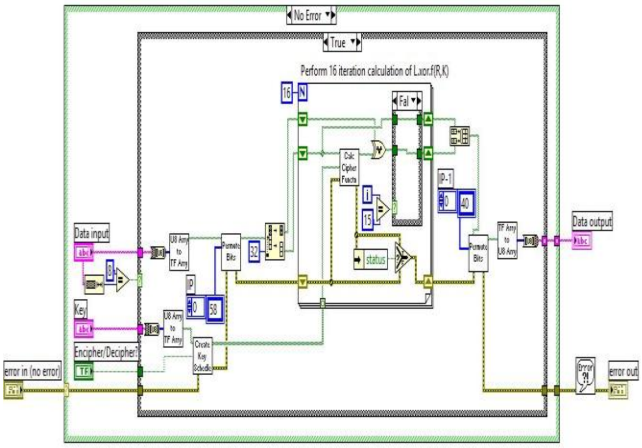


Figure 19: DES Block Diagram

Simulation Results:

ECB Mode 8byte Input:

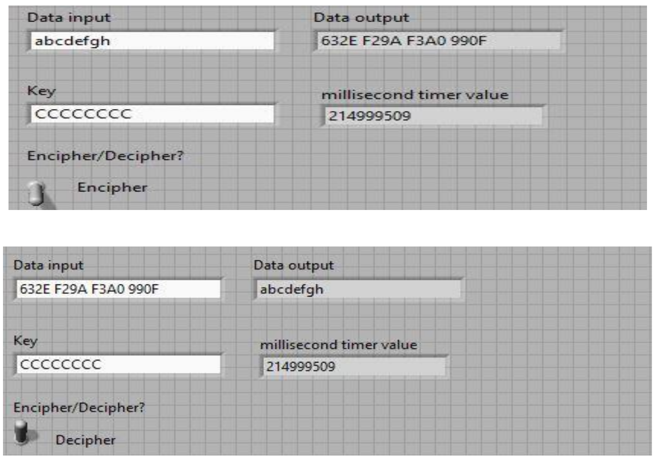


Figure 20:

The screen shot of 8 byte ECB mode encryption and decryption is shown in figure 20. In which 8 byte input is given to single DES processor and time taken for both encryption and decryption is 214999509ms.

ECB Mode 16 Byte Parellel Input:

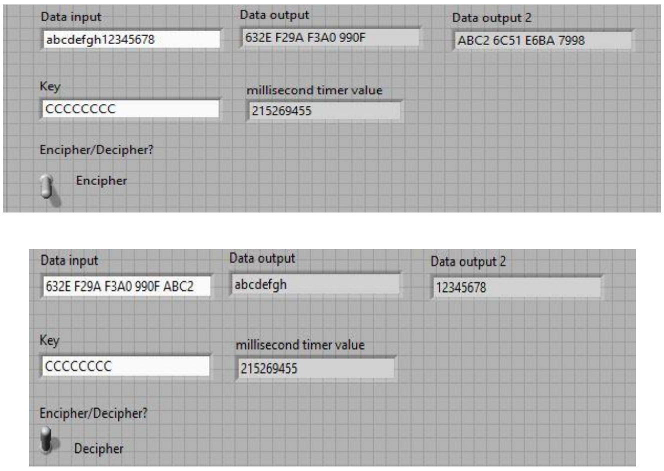


Figure 21:

The outputs in figure 21 are for 16 byte input in which we create multiple DES block ciphers with one input and multiple outputs, in which each block takes 8 byte as input and gives output simultaneously of both blocks parallel. so time taken for 16 byte input is nearer as 8 byte and it is 215269455ms.

CTR Mode 8 Byte Input

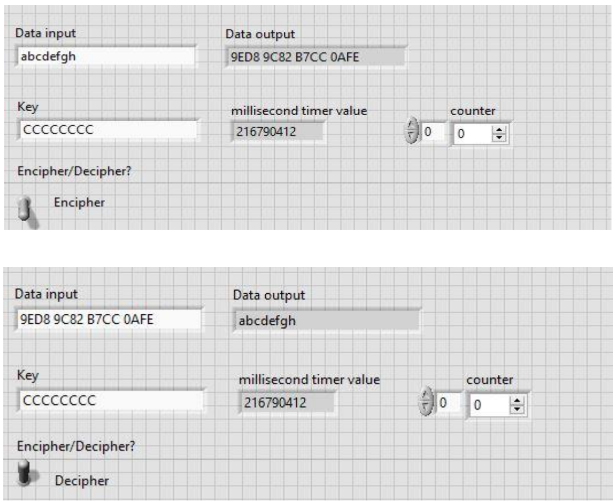


Figure 22:

The screen shots of 8 byte CTR mode encryption and decryption is shown in figure 22. In which 8 byte input is given to single DES processor and time taken for both encryption and decryption is 216790412ms.

CTR Mode 16 Byte Parallel Input:

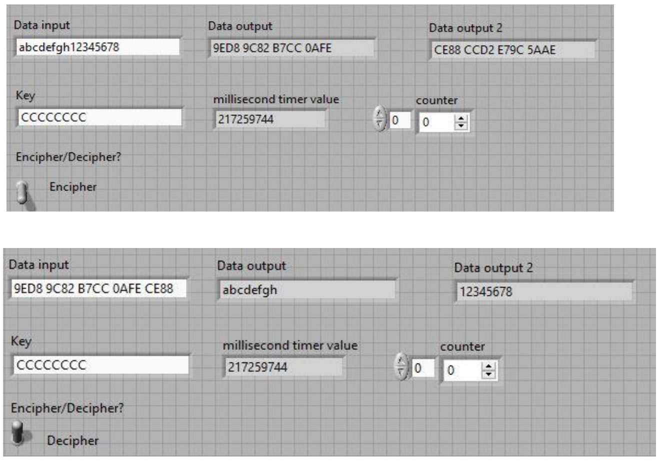


Figure 23:

The outputs in figure 23 are for 16 byte input in which we create multiple DES block ciphers with one input and multiple outputs, in which each block takes 8byte as input and gives output simultaneously for both blocks in parallel. so the time taken for 16 byte input is same as 8 byte and it is 217259744ms.

Conclusion

In this paper we presented our implementation of parallelism on DES block cipher modes using lab view software. We choose this algorithm because it is standard encryption algorithm in cryptographic library. Using lab view as a support for parallel implementation makes this cipher available to software with very little effort.

The objective of the paper is to show the execution time of both encryption and decryption of ECB and CTR block cipher modes by giving 8 byte as input and comparing by giving parallel input either 16, 32, 64 bytes…. where the input will be divided into several 8 byte multiple blocks.

References

[1] William Stallings, “Cryptography and Network Security Principles and Practice”, 5th Edition, Pearson Education 2011.
[2] Programming Strategies for Multicore Processing: Data Parallelism Publish Date:Dec 06,2011.
[3] Multicore Programming with Lab View Publish Date:Aug 12,2014.
[4] Virtual Instrumentation using Lab View by sanjay Gupta,Joseph John second Edition.
[5] Computer applications, ISSN 0975 – 8887, Volume 98-No.8, pp.11-14

[6] Parallel Programming for Everyone Take Advantage of Multicore CPUs With Lab View Publish Date:Feb 12,2010.

[7] J. John Raybin Jose, Dr. E. George Dharma Prakash Raj, “PACTS – A Communication Intensive Symmetric Block Cipher for Parallel Computing Environments” in the proceedings of IEEE International Conference on Research and Development Prospects on Engineering and Technology, Nagapattinam, India, April 2013.

[8] Data Encryption Standard announced by Federal Information Processing Standards Publications, October7 25th1999.

[9] National Institute of Standard Technology, NIST Special Edition 800-38A 2001 Edition.