CBC And Interleaved CBC Implementations Of PACMA Cryptographic Algorithm

J. John Raybin Jose[#1], E. George Dharma Prakash Raj[*2]

[#1]*Bishop Heber College (Autonomous), Tiruchirappalli, India.*
[*2]*Bharathidasan University, Tiruchirappalli, India.*

***Abstract*— PACMA (Parallelized Adaptive Cipher with Modular Arithmetic) is a Symmetric Cryptographic Algorithm designed with traditional techniques to efficiently utilize the parallel processing capabilities of modern computing systems. It overcomes the performance inconsistencies prevalent in conventional cryptographic algorithms when they are implemented in different computing systems with different processing capabilities. The size of the key and the plain text blocks are each 1024-bits. The adaptive nature of this algorithm is achieved by incorporating flexibility in the size of the key and plain text sub-blocks and the number of rounds. Level of Intra-packet parallelization, variety in grain size and the required security strength are achieved by suitably deciding the sub-block size. Flow of the algorithm is made dynamic by determining the execution steps through each key value at runtime. In spite of these advantages the ECB mode implementation of PACMA always produces the same cipher text block for a particular plain text block when the same key is used. CBC and Interleaved CBC modes with 2-way and 4-way interleaving are employed to alleviate this problem. The performance of the PACMA in ECB, CBC and Interleaved CBC modes are analyzed with implementations in shared memory parallel programming environment using OpenMP, Java Threads and MPI.**

***Keywords* : Symmetric Block Cipher, Parallel Adaptive Cryptography, CBC, Interleaved CBC, Modular Arithmetic.**

I. INTRODUCTION

Traditional cryptographic algorithms focus only on the complexity of the algorithm and the strength and the secrecy of the key [1]. We face variety of challenges in efficiently implementing these crypto systems as new trends and technologies have crept into modern communication and computing systems. Conventional symmetric cryptographic algorithms such as DES, IDEA, RC6, Blowfish and AES are developed before the year 2000 when computers were built around single 32, 16 or even 8 bits processors. But now, Cryptographic algorithms can be executed much faster on modern computers. The present day computing systems and that of future are not that of single core 32-bits desktops, but of multi-cored chips and multiprocessor machines whose processing capacities are 64 or 128 or more bits. Parallelizing the cryptographic algorithms is the only means to utilize these systems productively [2].

Further, there is a sharp increase in the rate of encryptions and decryptions carried out per unit time as the amount of information passing through communication networks have increased exponentially. This imposes additional overhead in the information exchange and may cause congestion. A way out of this trouble is to develop a new class of parallel cryptographic algorithms that can reduce the time required for encryption and decryption without diminishing the security strength using parallel execution techniques.

Addition and multiplication modulo operations are reversible and they are efficiently involved in cryptography. The binary coded value of the plain text is added or multiplied with the key, which is a member of the set of residues 'Z_m' to yield a sum or product value. When the value obtained is divided with 'm' yields a remainder, which is the resultant cipher text. Similarly the plain text can be retrieved by performing the same operation with the cipher text and the additive or multiplicative inverse value of the key used[3],[4].

Adaptive Cryptography is a trend, which deals with attaining flexibility in the cryptographic algorithms by dynamically varying the algorithmic flow and the choice of the key and the plain-text sub-blocks. Adaptive nature of these Cryptographic Algorithms can be classified as (i) Inter-Algorithmic Adaptation and (ii) Intra-Algorithmic Adaptation. Inter-Algorithmic Adaptation is achieved by employing different algorithms [5], whereas Intra-Algorithmic Adaptation instills dynamism within the same algorithm. Intra-Algorithmic Adaptation is employed in this work.

Parallel Cryptography is a recent development, which deals with implementation of cryptographic algorithms in modern parallel computing environments. Parallelism accelerates processing by simultaneous execution of multiple tasks. Implicit parallelism is achieved by the inherent resources and techniques in the processing hardware. Explicit parallelism is extracted by the external arrangements and codes by utilizing the available parallel hardware resources efficiently. Techniques used for explicit parallelism can be categorized as (i) Per-Connection Parallelism (ii) Per-Packet Parallelism and (iii) Intra-Packet Parallelism.

Per-connection parallelism is a method in which each connection is given its own thread or process that runs exclusively on one processor. This is the most common method of parallelization, and requires no modification to the existing algorithm or the software implementation. The per-connection parallelization method makes no attempt to fully utilize modern architectures. In Per-packet parallelism connections disperse their packet processing load over multiple processors, wherein each packet is treated individually. Many current algorithms lend themselves well to this kind of parallelization, but, no cryptographic software implementing this per-packet parallelism is available. Intra-packet parallelism is the most difficult type of parallelism, as it depends on algorithm design. It also requires changes in the implementation of the cryptographic algorithm, depending no longer on the flexibility of the hardware or operating system upon which it is run [2], [6]. Intra-packet parallelism is employed in PACMA.

This paper is designed as follows. section II gives the related works, section III depicts the PACMA Cryptographic Algorithm and its ECB implementation, section IV deals with the CBC implementations of PACMA, section V deals with the Interleaved CBC implementation of PACMA, and Section VI concludes the paper.

II. RELATED WORKS

Efforts to parallelize existing cryptographic algorithms have been pursued by several researchers from year 2000 onward. The prominent of these efforts can be classified broadly as Hardware or Software Parallel Cryptographic Implementations involving several technical approaches beneath them as depicted in Fig 1.

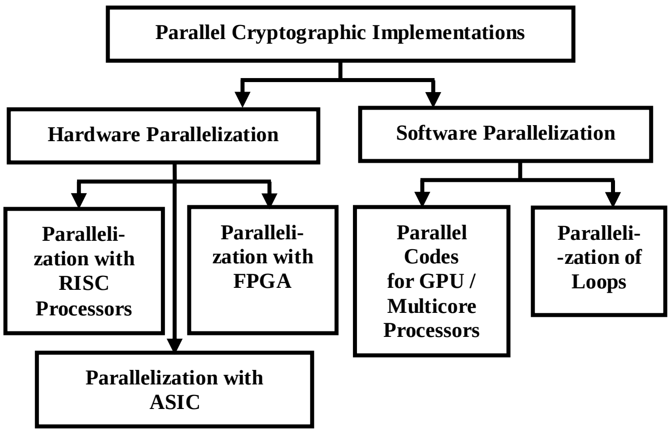
Parallel Cryptographic Implementations
Hardware Parallelization
Software Parallelization
Paralleli-zation with RISC Processors
Paralleli-zation with FPGA
Parallelization with ASIC
Parallel Codes for GPU / Multicore Processors
Paralleli--zation of Loops

Fig 1: Classification of Parallel Implementation of Cryptographic Algorithms

Design and implementation of a crypto processor by HoWon Kim et al, introduced a special-purpose microprocessor optimized for the execution of cryptography algorithms. The crypto processor consists of a 32-bit RISC processor block and a coprocessor block dedicated to the SEED and Triple-DES algorithms [7]. Pionteck et al, in their work presented a hardware design of AES with reconfigurable encryption/decryption engines which supports all key lengths. The reconfigurable crypt-engine is integrated in a 32 bit RISC processor as a functional unit and can operate in parallel with the standard ALU. The Reconfigurable Cryptographic Unit (RCU) is integrated into a 32 bit five stage pipeline RISC Processor and the area which is used for the RCU is less than 27% of the total area [8].

An Application-Specific Integrated Circuit (ASIC) is an Integrated Circuit (IC) customized for a particular use, rather than intended for general-purpose use. When ASICs are used to implement cryptographic algorithms it provides robust operation and much of the overhead involved in hardware implementation is reduced. The works carried out on ASIC implementation of DES, 3DES, IDEA and all the round 2 candidates of AES by S.Mukherjee et al., T.Ichikawa et al., and B.Weeks et al., are prominent in this category[9]-[11].

Field Programmable Gate Array (FPGA) logic cells are reconfigurable platforms that provide relatively a low cost, high performance method of implementing cryptographic primitives. Several standard algorithms such as DES, Triple DES, and AES are parallelized using FPGAs by Swankoski et al. [12]. The implementation environment is composed of Virtex-II Pro, FPGA Platform with Verilog HDL and Block RAM resources. Another hardware design of AES in chip proposed by Kotturi, et al., uses hierarchical simultaneous key generation, is implemented with ten separate units in XC2VP70 device with speed grade -7 with Virtex II Pro FPGA. Each unit can execute one round of the algorithm and ten rounds of the algorithm can be executed in parallel in a chip using external pipelined design. The throughput rate achieved in this method is higher than most other implementations [13]. In yet another implementation by Chi-Wu, et al, 128-bit AES was decomposed into four 32-bit AES to be executed in parallel. This outperformed all other recent works by requiring less than 20% reconfigurable area and operated four times faster than 32-bit AES by providing double the throughput [14].

Microprocessors with multiple cores and Graphical Processing Units (GPUs) are widely available at affordable prices. Considering the computational demands of the cryptographic algorithms, these parallel platforms are relevant to parallelize the existing algorithms to enhance the performance. CUDA programming is used to parallelize the algorithms in GPU [15], [16]. OpenMP is used to extract parallelism from Multi-Core Processors [17].

Praveen Dongara et al. parallelized several prominent private-key cryptographic algorithms such as 3DES, IDEA, Twofish, RC4, Rijndael, Blowfish, RC6, MARS and RC4. They yielded better performance in ECB and mode through multithreading on symmetric multiprocessors. As ECB implementations produced same cipher text for a given plain text always, CBC and Interleaved CBC implementations were verified [18]. Similar works with CBC and Interleaved CBC modes were carried out on AES by Zadia Codabux-Rossan et al. [19] and Ashokkumar et al. [20]

Considering the fact that the most time-consuming elements of source code of cryptographic algorithms without including the I/O functions are loops, they are parallelized for all the popular cryptographic algorithms such as DES, Triple DES, IDEA, AES, RC5, Blowfish, GOST and LOK191 by Bielecki et al. The standard modes of operations selected for Ciphers are ECB, CBC, CFB, OFB and CTR. The Data Dependences are resolved before parallelizing the Loops. OpenMP was used to parallelize the loops in these algorithms and Petit was used to resolve dependences in the loops. Speedup measurements were presented for all these implementations [21], [22].

Even though all the efforts to parallelize the existing conventional cryptographic algorithms with hardware and software techniques had given better results, they cannot be fully parallelized or implemented efficiently in present day computing systems. The dependency problems and the inability to efficiently modularize the sections of the algorithms hover around and haunt the parallelization. Thus a path for the new class of cryptographic algorithms that is devoid of these problems is set in.

III. PARALLEL ADAPTIVE CIPHER WITH MODULAR ARITHMETIC (PACMA)

In our previous work we have developed the Parallel Adaptive Cipher with Modular Arithmetic (PACMA) and implemented in ECB mode [23]. PACMA is a symmetric block cipher with the block length and the key size each of 1024 bits. The sub-block size of the key and the plain text is made adaptive by varying them suitably based on the processing capacity of the computing system used. The behavior of the algorithm is decided dynamically by deriving the control information from the key. The granularity of the algorithm is decided by forming sub-blocks of various sizes in the range 2^n where n=3 to 8. The processing resources available and the security strength required are used to decide the number of rounds, size of the key and the plaintext sub-blocks. This is depicted in Fig 2.

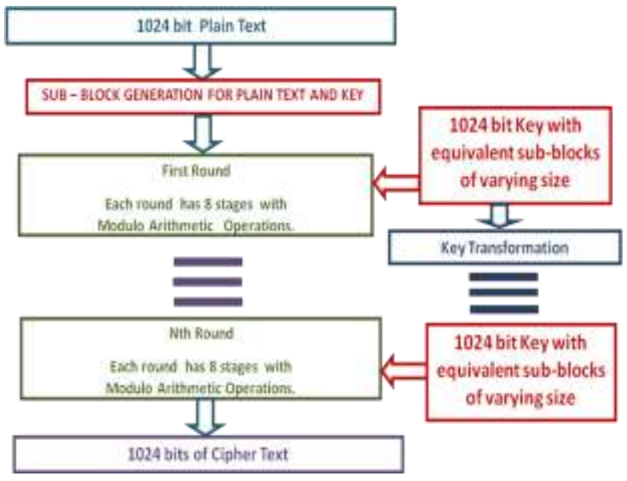
1024 bit Plain Text
SUB – BLOCK GENERATION FOR PLAIN TEXT AND KEY
First Round
Each round has 8 stages with Modulo Arithmetic Operations.
1024 bit Key with equivalent sub-blocks of varying size
Key Transformation
Nth Round
Each round has 8 stages with Modulo Arithmetic Operations.
1024 bit Key with equivalent sub-blocks of varying size
1024 bits of Cipher Text

Fig 2. General Block Diagram of PACMA

Each round in PACMA has eight stages as depicted in fig 3. The 1024 bit key is directly used in the first stage but it undergoes different transformations in the remaining stages. The sub-block generation routine is run to generate the key and the plain text sub-blocks; before they are involved in operations at each stage. In the first stage modulo addition operation is performed with the key sub-blocks and the plain text sub-blocks in a pattern decided by the initial and the final bits of the key. If both these bits are of same value then modulo addition operation is performed directly, otherwise the plain text bits are reversed before the operation. The key bits are then rotated to right or left by *b/2* positions within the key sub-blocks so that it can be used in the next stage. Here '*b*' refers to the number of bits in each key sub-block. The direction of rotation is determined by the parity of the key. If the parity is odd the bits are rotated to the right, otherwise they are rotated to the left.

In the second stage addition modulo 2^8 operation is carried out with each key and plain text sub-blocks. If the sub-block size is greater than 8 bits, the key and the plain text sub-blocks are further divided into chunks of 8 bits in this stage to facilitate the addition modulo 2^8 operation. Following this exchange manipulation is carried out between the key sub-blocks. This is achieved by swapping the odd numbered sub-blocks with the next higher order even numbered sub-blocks. In the third stage, multiplication modulo 2^8+1 operation is carried out on the 8 bit chunks of the key and the plain text that are available after the second stage. Intra sub-block rotation is then carried out for each 8 bit chunk of key sub-blocks as it was done before the second stage.

In the fourth stage addition modulo 2^{16} operation is performed with the key sub-blocks and the plain text sub-blocks after dividing or grouping them into chunks of 16 bits. An inter sub-block rotation is performed for the key as it is done before the fifth stage. The position for rotation is derived from the value given by the first ten bits of the key block. The direction of rotation is specified by the eleventh bit of the key block. If the value of the eleventh bit is '1'the key bits are rotated to the right and if it is '0' then it is rotated to the left. Multiplication modulo $2^{16}+1$ operation is carried out after further dividing or grouping the key and plain text sub-blocks into chunks of 16 bits in the fifth stage. The key manipulation is then performed with the 16 bit chunks of the key as it is done before the fourth stage for 8 bits chunks.

The operation in the second state is repeated for the sixth stage. The key sub-blocks then undergo an exchange key manipulation by swapping the sub-blocks at the extreme ends. The inner sub-blocks at the next level are then swapped and this is carried towards the sub-blocks in the central position of the key. The operation performed in the third stage is repeated in the seventh stage. The key bits then experience the Intra sub-block rotation similar to the first key manipulation operation and then they are utilized for modulo addition operation in the eighth stage. Brief algorithmic depiction of PACMA with single round can be given as follows:

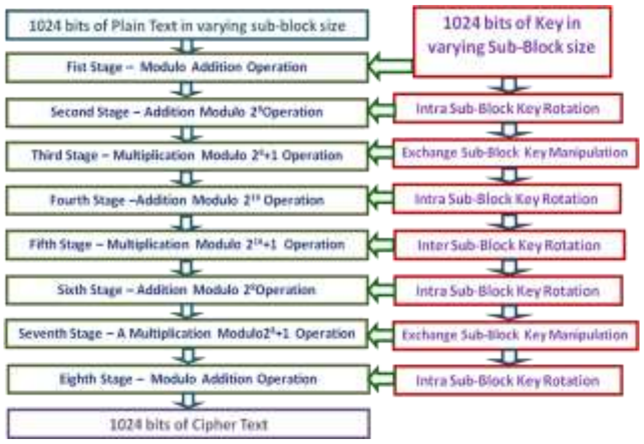
1024 bits of Plain Text in varying sub-block size
1024 bits of Key in varying Sub-Block size
Fist Stage – Modulo Addition Operation
Second Stage – Addition Modulo 2⁸Operation
Intra Sub-Block Key Rotation
Third Stage – Multiplication Modulo 2⁸+1 Operation
Exchange Sub-Block Key Manipulation
Fourth Stage –Addition Modulo 2¹⁶ Operation
Intra Sub-Block Key Rotation
Fifth Stage – Multiplication Modulo 2¹⁶+1 Operation
Inter Sub-Block Key Rotation
Sixth Stage – Addition Modulo 2⁸Operation
Intra Sub-Block Key Rotation
Seventh Stage – A Multiplication Modulo2⁸+1 Operation
Exchange Sub-Block Key Manipulation
Eighth Stage – Modulo Addition Operation
Intra Sub-Block Key Rotation
1024 bits of Cipher Text

Fig 3. Stages in each round of PACMA

Input : 1024 bit plain text block

Output : 1024 bit cipher text block

Sub-Block Generation:

1. Run environment identification routine to identify the number of processors/cores 'p', their data handling capacities 'c' and clock speed 's' to divide the 1024 bits key and the 1024 bits plain text into sub-blocks of 'b' bits.
2. if p==1 && c < 16 bits && s ≤ 10 MHz then b
3. else if p==1 && c ≥ 16 bits && c < 32 bits && s > 10 MHz && s ≤ 100 MHz then b = 16 bits.
4. else if p==1 && c ≥ 32 bits && c < 64 bits && s > 100MHz && s ≤ 1000 MHz then b = 32 bits.
5. else if p≤4 && c≥64 bits && s > 1GHz then b=64 bits.
6. else if p>4 c ≥ 64 bits && s > 3GHz then b=256 bits
7. else display "resources unsuitable to implement PACMA".

Steps in Single Round of PACMA :

1. Modulo addition with key and plain text sub-blocks.
2. Intra sub-block rotation on key sub-blocks.
3. Addition modulo 2^8 with key and plaintext sub-blocks.
4. Exchange sub-block operation on key sub-blocks.
5. Multiplication modulo 2^8+1 with key and plaintext sub-blocks.
6. Intra sub-block rotation on chunks of key sub-blocks.
7. Addition modulo 2^{16} with key and plaintext sub-blocks.
8. Inter sub-block rotation on key sub-blocks.
9. Multiplication modulo $2^{16}+1$ with key and plaintext sub-blocks.
10. Intra sub-block rotation on chunks of key sub-blocks.
11. Addition modulo 2^8 with key and plaintext sub-blocks.
12. Exchange sub-block operation on key sub-blocks.
13. Multiplication modulo 2^8+1 with key and plaintext sub-blocks.
14. Intra sub-block rotation on key sub-blocks.
15. Modulo addition with key and plaintext sub-blocks.

PACMA is implemented in shared memory computing system with multi-core architecture with four cores using MPI, OpenMP and Java Thread programming with different sub-block sizes and compared with the sequential results. The speedup of various combinations of executions in ECB mode are analyzed and compared and the results are given in Table I.

TABLE I

ECB MODE IMPLEMENTATION OF PACMA (ONE ROUND)

SUB-BLOCK SIZE	PACMA - SPEEDUP IN ECB MODE					
	MPI		OpenMP		JAVA Threads	
	ENC	DEC	ENC	DEC	ENC	DEC
8 bits	**2.85**	**2.89**	**2.33**	**2.38**	**2.18**	**2.22**
16 bits	**3.42**	**3.47**	**2.57**	**2.63**	**2.41**	**2.46**
32 bits	**3.54**	**3.62**	**2.96**	**3.02**	**2.75**	**2.81**
64 bits	**3.61**	**3.68**	**3.37**	**3.43**	**3.17**	**3.22**
128 bits	**3.83**	**3.89**	**3.72**	**3.68**	**3.56**	**3.61**
256 bits	**3.86**	**3.94**	**3.81**	**3.88**	**3.74**	**3.79**

ECB Mode: Electronic Code Book Mode
ENC : Encryption **DEC : Decryption**

The performance of cryptographic algorithms in parallel computing environment can be indicated using speedup. Speedup is the ratio of the time taken by the serial implementation of the algorithm to that of its parallel implementation and is denoted by $S_P = T_s/T_P$. Where '*p*' denotes the parallel implementation with *p* number of processors or cores and 's' denotes its sequential implementation.

All the parallel implementations provided similar variation in their output. When the sub-block size is kept small the speedup is low, but it gradually increased linearly when the sub-block size is increased. The decryption process provided better speedup than the encryption process because most of the values and the decisions computed for the encryption stages are made available to the decryption stages. A comparative representation of the performance of encryption algorithm using MPI, OpenMP and Java threads are shown in Fig. 4 and that of the decryption algorithm is shown in Fig 5.

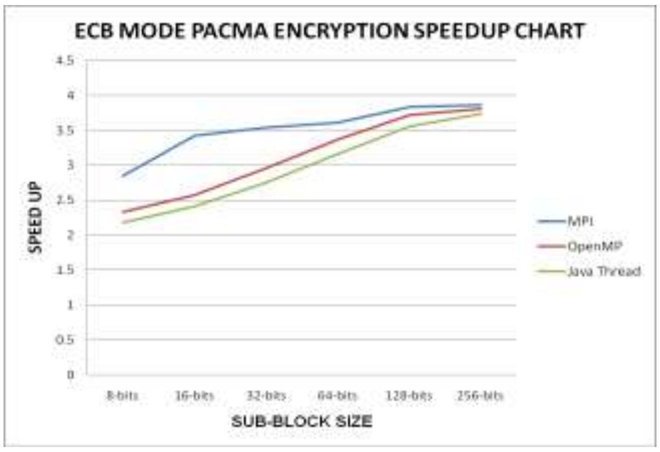

Fig. 4: Performance of ECB mode Encryption of PACMA

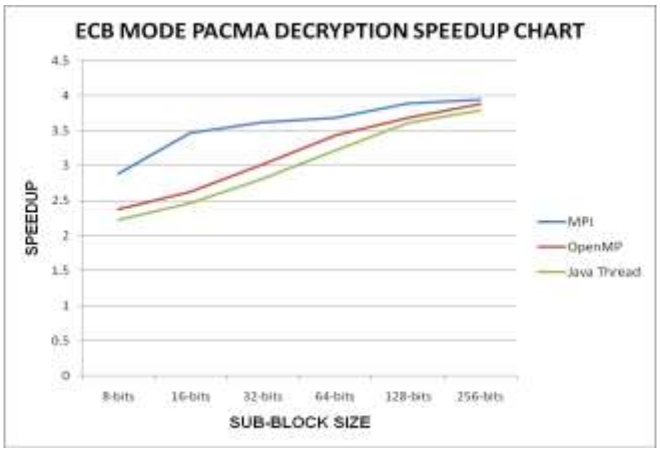

Fig. 5: Performance of ECB mode Decryption of PACMA

IV. CBC MODE IMPLEMENTATION OF PACMA

In ECB mode of PACMA, a plain text block always produces the same cipher text block, when the same key is used. Cipher Block Chaining (CBC) mode is used to overcome this problem. CBC mode ensures that even if the same plain text block is repeated again and again it yield totally different cipher text blocks in the output. In CBC mode result of the encryption of the previous block are fed back into the encryption of the current block. As there is no feedback available for the first block of the plaintext a random block of text known as Initialization Vector (IV) is used in the first step of encryption. The encryption process in CBC mode is shown in Fig. 6 and the decryption process in Fig. 7.

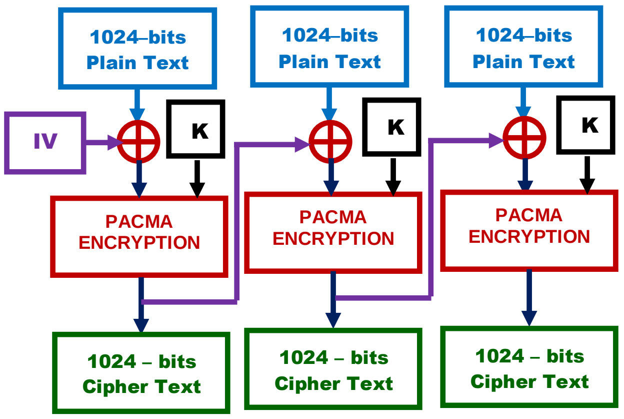

Fig. 6: CBC mode Encryption of PACMA

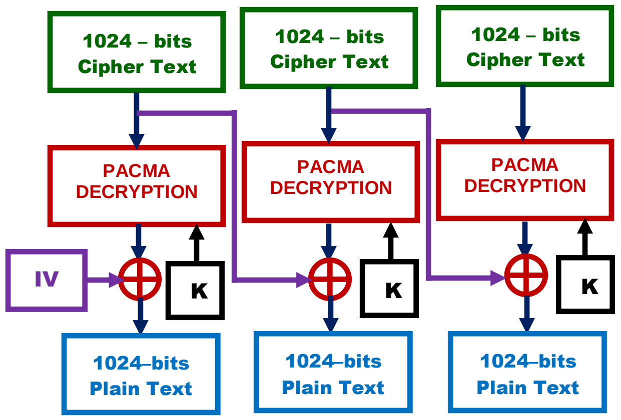

Fig. 7: CBC mode Decryption of PACMA

The decryption is just the reverse of the encryption, except the feedback of the previous level is readily available in decryption, whereas in encryption it is not. The speedup results of CBC mode implementation of PACMA in MPI, OpenMP and Java Threads are given in Table II.

TABLE II

CBC MODE IMPLEMENTATION OF PACMA (ONE ROUND)

SUB-BLOCK SIZE	PACMA - SPEEDUP IN CBC MODE					
	MPI		OpenMP		JAVA Threads	
	ENC	DEC	ENC	DEC	ENC	DEC
8 bits	1.28	2.71	1.14	2.21	1.12	2.04
16 bits	1.49	3.31	1.31	2.46	1.25	2.27
32 bits	1.62	3.48	1.52	2.87	1.44	2.70
64 bits	1.66	3.52	1.61	3.31	1.60	3.09
128 bits	1.85	3.72	1.82	3.57	1.74	3.48
256 bits	1.97	3.78	1.94	3.76	1.89	3.64

CBC Mode : Cipher Block Chaining Mode
ENC : Encryption **DEC : Decryption**

The performance of CBC mode encryption of PACMA is drastically reduced because of the dependencies caused by feedback of the ciphertext to the next level. The decryption is not affected as the feedback to the next level is readily available. The encryption and decryption performance graphs of CBC mode PACMA is given in Fig. 8 and Fig. 9.

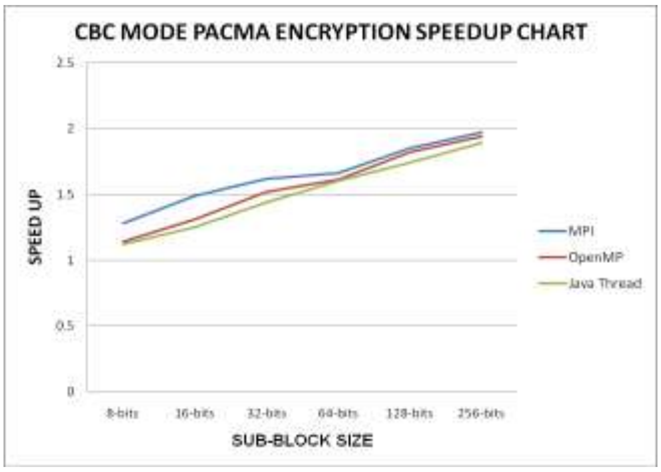

Fig. 8: Performance of CBC mode Encryption of PACMA

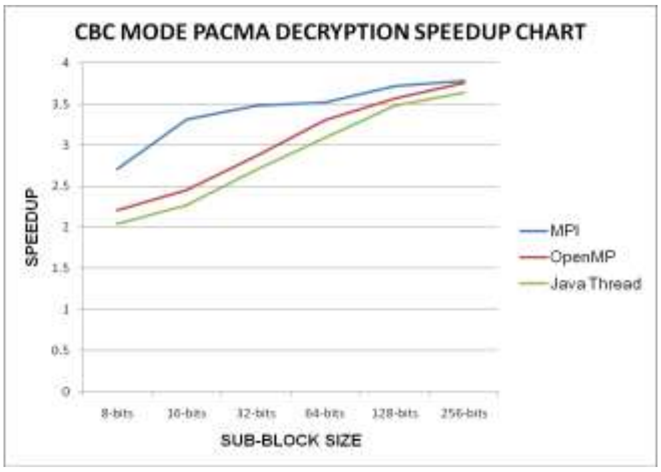

Fig. 9: Performance of CBC mode Decryption of PACMA

V. INTERLEAVED CBC MODE IMPLEMENTATION OF PACMA

The encryption in CBC mode depends on the encryption of the previous sub-blocks. This makes it difficult to parallelize encryption. The only solution to this problem is to interleave multiple encryption blocks [24]. Interleaving can be done in n-ways, wherein the two-way and four-way interleaving are adopted in this work.

A. *Two-Way Interleaved CBC mode of PACMA*

Two-Way interleaving is the next immediate improvement to the CBC implementation. In two-way interleaving the output of the first encryption sub-block is feedback to the third and that of second to fourth and so on. In this case two Initialization Vectors are required to start the encryption and the decryption processes. The structure of two-way interleaving for encryption is shown in Fig. 10 and that of decryption in Fig. 11.

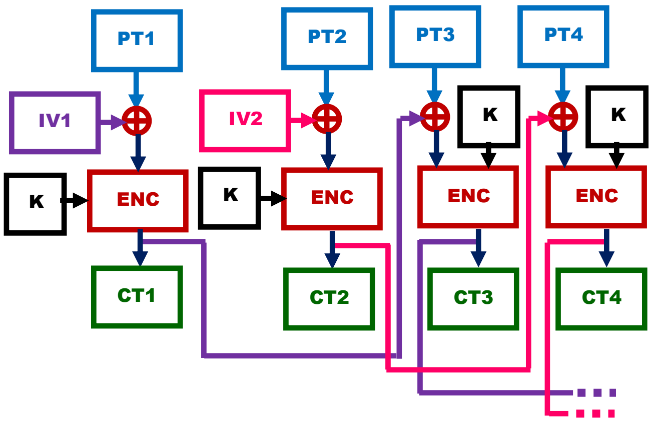

Fig. 10: Two-Way Interleaved CBC mode Encryption of PACMA

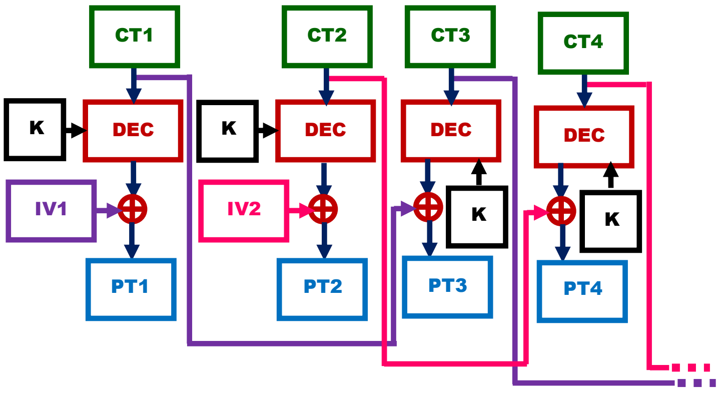

Fig. 10: Two-Way Interleaved CBC mode Decryption of PACMA

The decryption process in this case too is just the reverse of the encryption process. The feedback ciphertext of the previous level is also readily available in each block of decryption, whereas in encryption it is not so. The speedup results of two-way interleaved CBC mode implementation of PACMA in MPI, OpenMP and Java Threads are given in Table III.

TABLE III

2-WAY IMPLEMENTATION OF ICBC MODE PACMA (ONE ROUND)

SUB-BLOCK SIZE	PACMA - SPEEDUP IN 2-WAY ICBC MODE					
	MPI		OpenMP		JAVA Threads	
	ENC	DEC	ENC	DEC	ENC	DEC
8 bits	1.73	2.73	1.44	2.24	1.32	2.07
16 bits	2.18	3.32	1.59	2.49	1.49	2.30
32 bits	2.33	3.50	1.89	2.89	1.76	2.72
64 bits	2.45	3.54	2.22	3.34	2.06	3.12
128 bits	2.64	3.74	2.49	3.59	2.34	3.50
256 bits	2.70	3.80	2.58	3.78	2.53	3.66

ICBC Mode : Interleaved Cipher Block Chaining Mode
ENC : Encryption **DEC : Decryption**

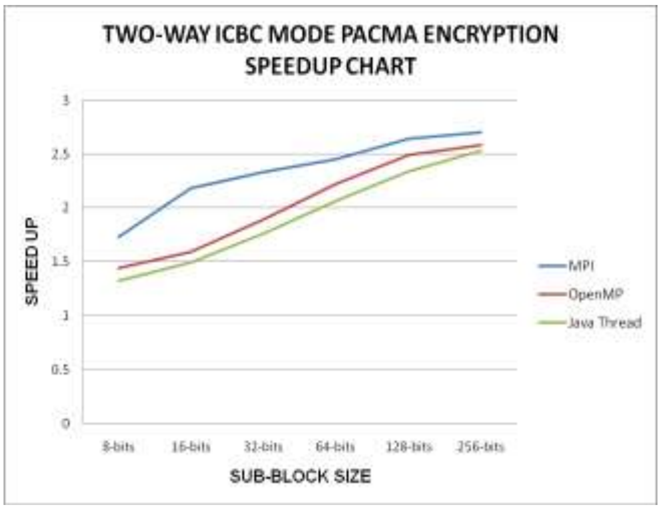

Fig. 12: Performance of 2-Way ICBC mode Encryption of PACMA

The performance of two-way interleaved CBC implementation is found to be better than the CBC implementation and it is illustrated in Fig. 12 and Fig. 13. The additional processes or threads that handle the two-ways of encryption and decryption separately are responsible for this enhancement.

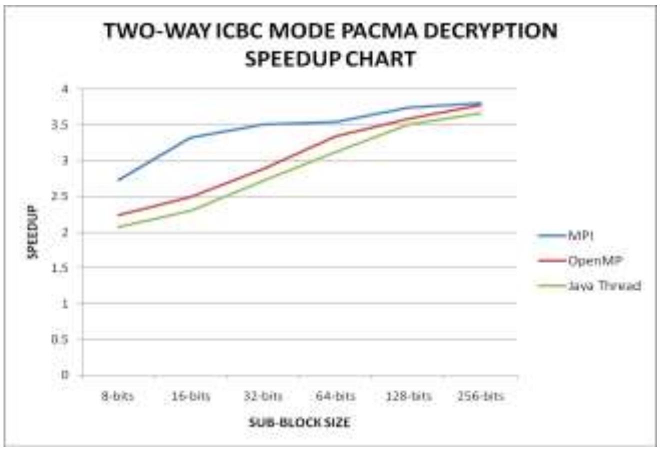

Fig. 13: Performance of 2-Way ICBC mode Decryption of PACMA

B. *Four-Way Interleaved CBC mode of PACMA*

In four-way interleaving the output of the first encryption sub-block is feedback to the fifth and that of second to sixth, third to seventh, fourth to eighth and so on. In this case four Initialization Vectors are required to start the encryption and the decryption processes. The structure of four-way interleaving for encryption is shown in Fig. 13 and that of decryption in Fig. 14.

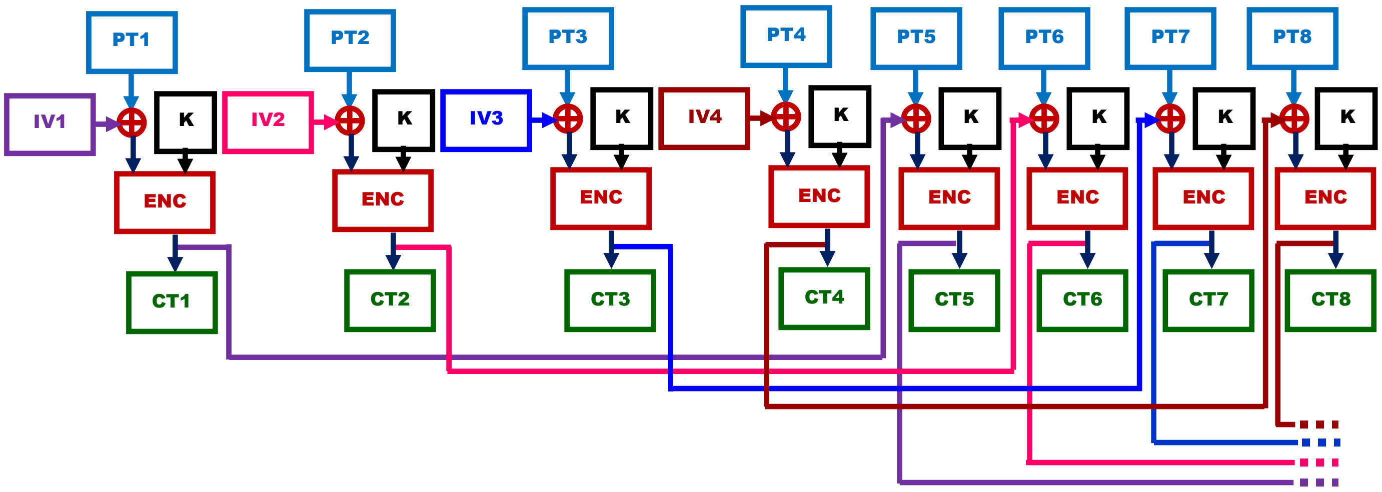

Fig. 13: Four-Way Interleaved CBC mode Encryption of PACMA

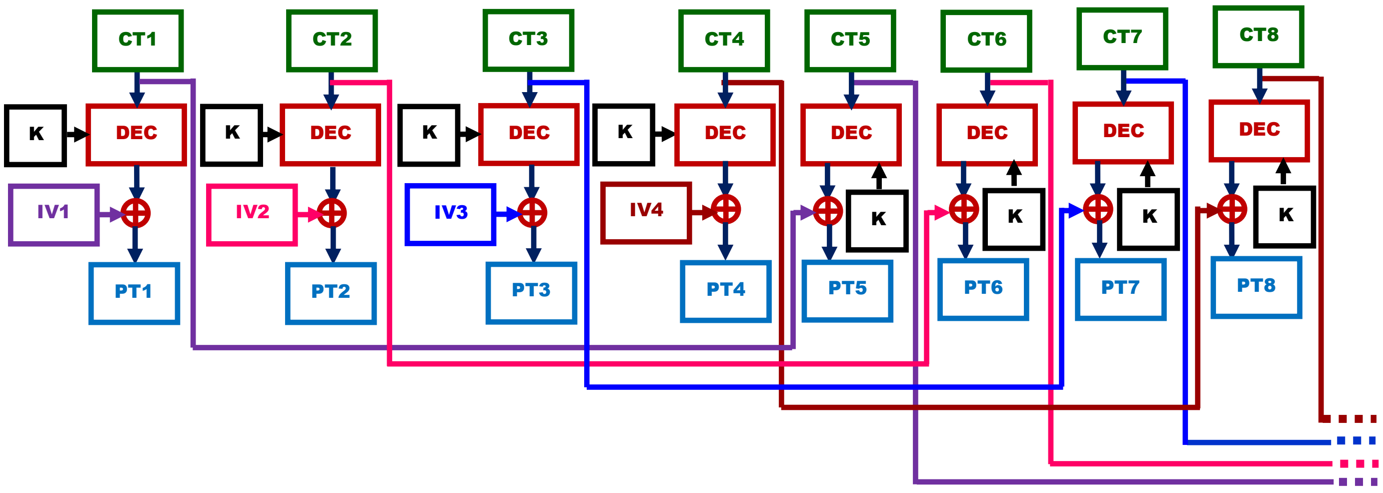

Fig. 14: Four-Way Interleaved CBC mode Decryption of PACMA

A considerable improvement is seen in the performance of four-way interleaved CBC mode implementation of PACMA when compared with the simple CBC and two-way Interleaved CBC modes. This is shown in Table IV and in Fig. 15 and Fig. 16.

TABLE IV

4-WAY IMPLEMENTATION OF ICBC MODE PACMA (ONE ROUND)

SUB-BLOCK SIZE	PACMA - SPEEDUP IN 4-WAY ICBC MODE					
	MPI		OpenMP		JAVA Threads	
	ENC	DEC	ENC	DEC	ENC	DEC
8 bits	1.98	2.77	1.63	2.26	1.54	2.10
16 bits	2.46	3.36	1.86	2.52	1.76	2.34
32 bits	2.58	3.54	2.16	2.92	2.05	2.72
64 bits	2.67	3.56	2.49	3.36	2.36	3.15
128 bits	2.91	3.77	2.78	3.62	2.67	3.53
256 bits	2.97	3.83	2.89	3.80	2.78	3.69

ICBC Mode : Cipher Block Chaining Mode
ENC : Encryption **DEC : Decryption**

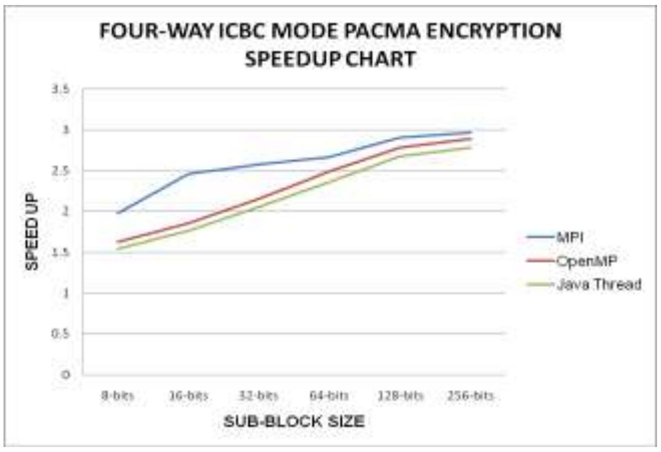

Fig. 15: Performance of 4-Way ICBC mode Encryption of PACMA

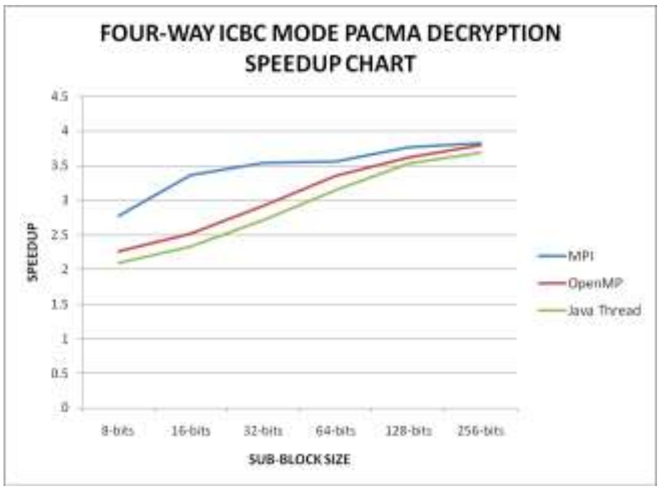

Fig. 16: Performance of 4-Way ICBC mode Decryption of PACMA.

Increasing the level of the interleaved CBC mode enhances the parallel performance, but it also increases the number of Initialization Vectors required and the complexity of implementations. Even though the encryptions are made to perform better, it cannot be enhanced like ECB mode implementations due to the dependency issues involved with the feedback of the ciphertext from the previous stage. The decryption processes does not suffer such drawbacks and they perform well in parallel executions, as the ciphertext of the previous stage is available well in advance before the beginning of the process in the current level.

VI. CONCLUSION

The advantages of PACMA are its adaptive nature, its ability to run on different parallel computing architectures efficiently, its flexibility in deciding the size of the key and plain text sub-blocks and its ability to expand by suitably deciding the number of rounds. The security level of PACMA is very strong as key size and plaintext block size are each 1024 bits. When executed in parallel computing environments the performance of ECB mode is found to be better. But it always produces the same ciphertext for a particular plaintext when the same key is used. Although CBC mode is employed to alleviate this problem, its decryptions support parallelization, whereas its encryptions do not. The issue faced in parallelization of CBC mode encryptions is solved to some extent with two-way and four way Interleaved CBC implementations.

REFERENCES

[1] William Stallings, “Cryptography and Network Security-Principles and Practice”, 5th Edition, Dorling Kindersley (India) Pvt. Ltd., licensees of Pearson Education, 2011.

[2] Eric C. Seidel, Joseph N. Gregg, “Preparing Tomorrow’s Cryptography : Parallel Computation via Multiple Processors, Vector Processing, and Multi-Cored Chips”, Research Paper, May 13, 2003.

[3] Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman, “An Introduction to Mathematical Cryptography”, Springer International Edition, Springer (India) Pvt. Ltd., New Delhi, 2008.

[4] Menezes A. J., Van Oorschot P. C., Vastone S. A., “Handbook of Applied Cryptography”, CRC Press, 1996.

[5] Suman Khakurel, Prabhat Kumar Tiwary, Niwas Maskey, Gitanjali Sachdeva, “Security Vulnerabilities in IEEE 802.11 and Adaptive Encryption Technique for Better Performance”, IEEE Symposium on Industrial Electronics and Applications, Penang, Malaysia, 2010.

[6] Thomas Rauber, Gudula Runger, “Parallel Programming –for Multicore and Cluster Systems”, International Edition, Springer (India) Pvt. Ltd. New Delhi, 2010.

[7] HoWon Kim, YongJe Choi, Kyoil Chung, and HeuiSu Ryu, "Design and Implementation of a Private and Public Key Crypto Processor and Its Application to Security System," proceedings of the 3rd International Workshop on Information Security Applications, pp. 515 – 531, Jeju, Korea, 2002,

[8] Pionteck, T., Staake T., Stiefmeier T., Kabulepa L. D., Glesner M., “Design of reconfigurable AES encryption/decryption engine for mobile terminals”, Paper presented at the proceedings of the International Symposium on Circuits and Systems ISCAS, 2004.

[9] Sourav Mukherjee, Bidhudatta Sahoo, “A survey on hardware implementation of IDEA Cryptosystems” Information Security Journal : A Global Perspective, Vol. 20, Nr. 4-5, pp 210-218, 2011.

[10] Tetsuya Ichikawa, Tomomi Kasuya, and Mitsuru. Matsui. “Hardware evaluation of the AES finalists.” In Proc. Third Advanced Encryption Standard Candidate Conference (AES3), pages 279–285, New York, USA, 2000.

[11] Bryan Weeks, Mark Bean, Tom Rozylowicz, and Chris Ficke. “Hardware performance simulations of Round 2 Advanced Encryption Standard algorithms”. In Proc. Third Advanced Encryption Standard Candidate Conference (AES3), New York, USA, 2000.

[12] Swankoski E. J., Brooks R. R., Narayanan V., Kandemir M., and Irwin M. J., “A Parallel Architecture for Secure FPGA Symmetric Encryption”, Paper Presented at the 18th International Parallel and Distributed Processing Symposium, Santa Fe, New Mexico, 2004.

[13] Kotturi D., Seong-Moo Y., Blizzard J., “AES crypto chip utilizing high-speed parallel pipelined architecture” Paper presented at the IEEE International Symposium on Circuits & Systems ISCAS, 2005.

[14] Chi-Wu H., Chi-Jeng C., Mao-Yuan L., Hung-Yun T., “The FPGA Implementation of 128-bits AES Algorithm Based on Four 32-bits Parallel Operation”, Paper presented at the First International Symposium on Data, Privacy, and E-Commerce, ISDPE, 2007.

[15] Chonglei, M., J. Hai and J. Jennes, “CUDA-based AES Parallelization with fine-tuned GPU memory utilization”, Paper Presented at the IEEE International Symposium on Parallel and Distributed Processing, Workshops and Ph. D. Forum (IPDPSW), pp19-23, 2010.

[16] Julian Ortega, Helmuth Tefeffiz, Christian Treffiz, “Parallelizing AES on Multicores and GPUs”, Proceedings of the IEEE International Conference on Electro/Information Technology (EIT), 15-17 May 2011, Mankato, US, pp. 1-5.

[17] Li, H. and J. Z. Li, “A new compact dual-core architecture for AES encryption and decryption”, Canadian Journal of Electrical and Computer Engineering, pp 209-213, 2008.

[18] Praveen Dongara, T. N. Vijaykumar, Accelerating Private-key cryptography via Multithreading on Symmetric Multiprocessors. In Conference Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software, pp 58-69, 2003.

[19] Zadia Codabux-Rossan, M. Razvi Doomum, “AES CCMP Algorithm with N-Way Interleaved Cipher Block Chaining”, University of Mauritius Research Journal, Volume – 15, pp 527-544, 2009.

[20] S. Ashokkumar, K. Karuppasamy, Balaji Srinivasan, V.Balasubramanian “Parallel Key Encryption for CBC and Interleaved CBC” International Journal of Computer Applications(0975-8887), Volume 2–No. 1, 2010.

[21] Bielecki W., Burak D., “Parallelization of Standard Modes of Operation for Symmetric Key Block Ciphers”, Image Analysis, Computer Graphics, Security Systems and Artificial Intelligence Applications Vol 1 (ACS-CSIM 2005), Bialystok 2005.

[22] Bielecki W., Burak D., “Parallelization of Symmetric Block Ciphers”, Computing, Multimedia and Intelligent Techniques special issue on Live Biometrics and Security, Volume 1 (2005), Czestochowa University of Technology, June 2005.

[23] J. John Raybin Jose, Dr. E. George Dharma Prakash Raj, “PACMA – An Adaptive Symmetric Cryptographic Algorithm for Parallel Computing Environments” in the proceedings of the Fifth International Conference on Advances in Recent Technologies in Communication and Computing, Bangalore, India, September 2013.

[24] Schneier B., “Applied Cryptography : Protocols, Algorithms, and Source Code in C”, Second Edition, Wiley & Sons, 1995.