Implementation of PACMATS Cryptographic Algorithm in CBC and ICBC Modes

J. John Raybin Jose
Department of Information Technology,
Bishop Heber College (Autonomous),
Tiruchirappalli-620 017, India

E. George Dharma Prakash Raj
School of Computer Science, Engineering & Applications,
Bharathidasan University,
Tiruchirappalli-620 023, India

Abstract — **Parallelized Adaptive Cipher with Modular Arithmetic Transposition and Substitution (PACMATS) is a Symmetric Cryptographic Algorithm designed with conventional techniques to efficiently utilize parallel computing capabilities available today. It is designed to overcome 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. This algorithm derives the adaptive nature from its flexible design in fixing the size of the key and plain text sub-blocks and the number of rounds. Flow of the algorithm is made dynamic by determining the execution steps through each key value at runtime. In spite of these advantages PACMATS always produces the same cipher text block for a particular plain text block when the same key is used. CBC mode along with 2-way and 4-way Interleaved CBC modes are employed to overcome this problem. The performance of the PACMATS in ECB, CBC and ICBC modes are analyzed with implementations in shared memory parallel computing environment using OpenMP, Java Threads and MPI.**

Keywords : Symmetric Block Cipher, Parallel Adaptive Cryptography, Cryptograhic Modes, Modular Arithmetic, Transposition, Substitution.

I. INTRODUCTION

Conventional cryptographic algorithms focus only on the complexity of the algorithm and the strength and the secrecy of the key [1]. Wide varieties of challenges are faced today 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].

Nowadays there is a sharp increase in the rate of encryptions and decryptions carried out per unit time as the amount of information exchanged between networks have increased exponentially. This imposes overhead in the information exchange and causes congestion. A way out of this trouble is to develop a new class of parallel algorithms that reduce the time required for encryption and decryption without diminishing the security strength.

Cryptographic algorithms are classified as Symmetric and Asymmetric Key Algorithms. Symmetric key algorithms use only one key and they rely on the secure distribution and management of the session key. Symmetric key algorithms are divided into stream ciphers and block ciphers. Stream ciphers encrypt the bytes of the message one at a time but block ciphers take a number of bytes and encrypt them as a single unit. In Asymmetric Key Algorithms a public key and a private key are used. The public key is used to encrypt the information at the sending end; whereas the private key is known only to the receiver and it is used to decrypt the information [3]. Symmetric Block Ciphers are involved in this work.

Substitution, transposition and modular arithmetic techniques are employed in PACMATS. Substitution replaces a character by another. Transposition permutes the characters in a block of data. Substitution causes confusion in a cipher and adds more complexity to the algorithm in finding a relationship between the key and the cipher text from one side and the key and the plaintext in another side. Transposition causes diffusion and makes sure that there is no local relationship between the statistics of characters in plaintext and ciphertext [4].

Modular Arithmetic involves the remainder of division operation. Addition and multiplication modulo operations are reversible and they are 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 [5].

Adaptive Cryptography is a trend, which focuses on attaining flexibility in the cryptographic implementations by dynamically varying the algorithmic flow and the choice of the key and the plain-text sub-blocks. Adaptive Cryptographic techniques can be classified as (i) Inter-Algorithmic Adaptive Techniques and (ii) Intra-Algorithmic Adaptive Techniques. Inter-Algorithmic Adaptation is achieved by employing different algorithms [6], 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. Implicit parallelism uses the inherent resources and techniques whereas, explicit parallelism is extracted by the external arrangements and codes. 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. 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], [7]. Intra-packet parallelism is employed in PACMATS.

This paper is planned as follows. Section 2 gives the related works, Section 3 depicts the ECB implementation of PACMATS, Section 4 gives the CBC implementation of PACMATS, Section 5 deals with its Interleaved CBC implementations, Section 6 gives the Future Scope and Section 7 concludes the paper.

II. RELATED WORKS

Efforts to parallelize existing cryptographic algorithms have been pursued by many 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 under them as shown in Figure 1.

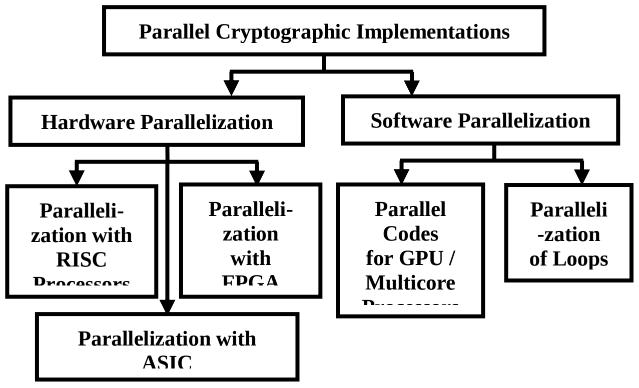

Parallel Cryptographic Implementations
Hardware Parallelization
Software Parallelization
Paralleli-zation with RISC Processors
Paralleli-zation with FPGA
Parallel Codes for GPU / Multicore Processors
Paralleli-zation of Loops
Parallelization with ASIC

Figure 1. Parallel Implementation of Existing Block Ciphers

Parallelization with RISC Processors by HoWon Kim et al., introduced a special-purpose microprocessor known as crypto processor. It had a 32-bit RISC processor block and a coprocessor block dedicated to SEED and TDES [8]. Pionteck et al., presented a hardware design of AES with reconfigurable encryption/ decryption engines which supports all key lengths [9].

An Application-Specific Integrated Circuit (ASIC) is an Integrated Circuit customized for a particular use. ASICs provide robust operation and much of the overhead involved is reduced. The works carried out on ASIC implementation of DES, 3DES, IDEA and AES by S. Mukherjee et al., T. Ichikawa et al., and B. Weeks et al., are prominent in this category [10]-[12].

Field Programmable Gate Array (FPGA) logic cells are reconfigurable platforms that provide low cost, high performance implementations of Block Ciphers. Several standard algorithms such as DES, TDES, and AES are parallelized using FPGAs by Swankoski et al., Kotturi, et al., and Chi-Wu, et al. [13]-[15].

Multi-core Processors and Graphical Processing Units (GPUs) are used to parallelize the existing Cryptographic algorithms to enhance the performance. CUDA programming is used to parallelize algorithms in GPU and OpenMP is used to extract parallelism from Multi-core Processors [16]-[19]. Praveen Dongara et al., implemented several symmetric cryptographic algorithms in ECB, CBC and interleaved CBC modes [20]. Similar works with CBC and Interleaved CBC modes were also carried out on AES by Zadia Codabux-Rossan et al. [21] and Ashokkumar et al. [22]

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 Burak et al. in standard modes of operations such as ECB, CBC, CFB, OFB and CTR modes [23] [24].

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. ECB IMPLEMENTATION OF PACMATS

In our previous work we have developed the Parallel Adaptive Cipher with Modular Arithmetic, Transposition and Substitution Techniques (PACMATS) and implemented it in ECB mode [25]. It is a symmetric block cipher with block length and key size each of 1024 bits. The sub-block size of key and the plain text is varied based on the computing environment used. The flow of the algorithm is decided dynamically by the control information in 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 and size of the sub-blocks. This is depicted in Figure 2.

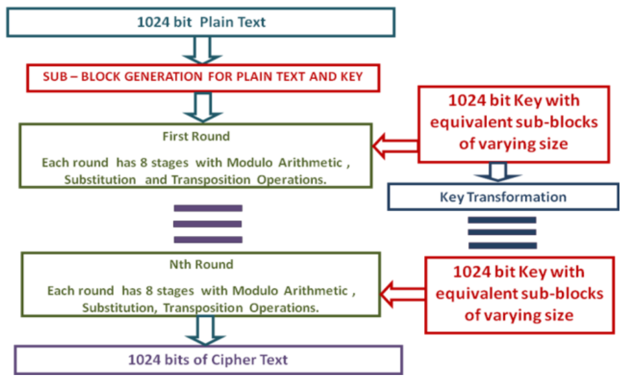

1024 bit Plain Text
SUB – BLOCK GENERATION FOR PLAIN TEXT AND KEY
1024 bit Key with equivalent sub-blocks of varying size
First Round
Each round has 8 stages with Modulo Arithmetic , Substitution and Transposition Operations.
Key Transformation
Nth Round
Each round has 8 stages with Modulo Arithmetic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

Figure 2. General Block Diagram of PACMATS

Each round has eight stages and the 1024 bits key is transformed for use in each stage as depicted in Figure 3. In the first stage Addition Modulo 2^8 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 sub-blocks. If both these bits are of same value then the operation is performed directly, otherwise the plain text bits are reversed before the operation. The key bits are rotated right or left by the position decided by the value got from the initial '*l*' bits of each sub-block. Here '*l*' takes the value 3 to 8 based on the sub-block size 2^l.

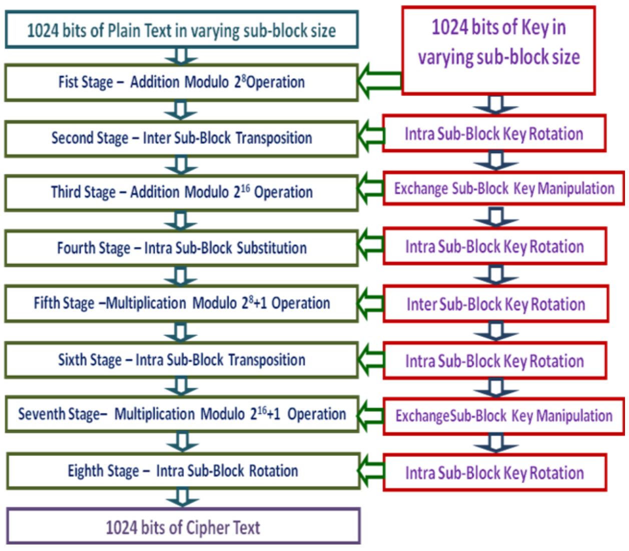

1024 bits of Plain Text in varying sub-block size
1024 bits of Key in varying sub-block size
Fist Stage – Addition Modulo 2⁸Operation
Intra Sub-Block Key Rotation
Second Stage – Inter Sub-Block Transposition
Exchange Sub-Block Key Manipulation
Third Stage – Addition Modulo 2¹⁶ Operation
Intra Sub-Block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

Figure 3. Stages in each round of PACMATS

In the second stage Inter Sub-block transposition is done by swapping the odd and even numbered sub-blocks or by swapping them in the reverse order. 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 or by swapping them in the reverse order. In the third stage, addition modulo 2^{16}operation is performed with key and plain text sub-blocks after dividing the sub-blocks into chunks of 16 bits. The Intra sub-block rotation is carried for each key sub-block based on the key value as it is done before the second stage. In the fourth stage intra sub-block substitution is carried out with plaintexts and key based on the key sub-block values. An inter sub-block rotation is performed for the key before the fifth stage and multiplication modulo 2^8+1 operation is performed in the fifth stage with plain text sub-blocks further divided into chunks of 8-bits based on control information derived from the key. Then key manipulation is performed with the key sub-blocks as it is done before for the second and the fourth stages. In the sixth stage Intra Sub-block transposition operation is carried out on the plain text sub-blocks. The key undergoes exchange sub-block key manipulation once again, as it is done just before the third stage. In the seventh stage multiplication modulo 2^{16}+1 operation is carried with key and plain text sub-blocks by dividing the plaintext sub-blocks into chunks of 16 bits. The key bits are then involved in the Intra sub-block rotation before they are utilized in the eighth stage. Intra sub-block rotation based on the initial few bits of each key sub-block is done in the eighth and the final stage. Brief algorithmic depiction of PACMATS with single round is given below.

Input: 1024 bit plain text block, 1024 bit key block

Output: 'n' blocks of 'b' bits

Sub-Block Generation:

1. Run environment identification routine to identify the number of processors/cores 'p' data handling capacity 'd' and clock speed 's' in order to divide the key and plaintext into 'n' sub-blocks of 'b' bits.
2. if (p=1&&d<16 bits&&s<=100 MHz) then b=8 bits.
3. else if(p==1 && d> =16 bits && d<32 bits && s>100 MHz && s<=1000 MHz) then b=16 bits.
4. else if(p==1 && d>=32 bits && d<64 bits && s>1000 MHz) then b=32 bits.
5. else if(p>1 && p<=4 && d>=64 bits && d< 128 bits && s>2000 MHz) then b=64 bits.
6. else if(p>4 && p<= 12 && d>=128 bits && s>3000 MHz) then b=128 bits.
7. else if(p>12 && d>=128 bits && s>3000 MHz) then b=256 bits.
8. else display "resources unsuitable to implement PACMATS".

Steps in Single Round execution of PACMATS:

1. Addition modulo 2^8 operation.
2. Intra sub-block key rotation.
3. Inter sub-block transposition.
4. Exchange sub-block key manipulation.

5. Addition modulo 2^{16} operation.
6. Intra sub-block key rotation.
7. Intra sub-block substitution.
8. Inter sub-block key rotation.
9. Multiplication modulo 2^8+1 operation.
10. Intra sub-block key rotation.
11. Intra sub-block Transposition.
12. Exchange sub-block key operation.
13. Multiplication modulo $2^{16}+1$ operation.
14. Intra sub-block key rotation.
15. Intra sub-block rotation.

For all normal ECB mode implementation of PACMATS in Personal Computers, single round execution is sufficient as it provides the required security strength. Utilization of inter sub-block and intra sub-block transpositions, substitutions and modular arithmetic operations makes PACMATS both communication and computation intensive for execution in parallel computing environments. PACMATS is implemented in shared memory architecture using MPI, OpenMP and Java Threads programming with different sub-block sizes and compared with sequential results. The speedup of various combinations of executions are analyzed and compared and the results are shown in Table 1.

TABLE I. ECB MODE IMPLEMENTATION OF PACMATS

SUB-BLOCK SIZE	SPEEDUP IN ECB MODE OF PACMATS					
	MPI		OpenMP		JAVA Threads	
	ENC	DEC	ENC	DEC	ENC	DEC
8 bits	2.32	2.36	2.79	2.82	2.63	2.68
16 bits	2.81	2.85	2.98	3.03	2.88	2.92
32 bits	3.43	3.47	3.56	3.6	3.49	3.53
64 bits	3.63	3.67	3.76	3.79	3.66	3.69
128 bits	3.7	3.73	3.82	3.86	3.75	3.78
256 bits	3.81	3.84	3.94	3.98	3.86	3.89

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

Speedup, throughput, efficiency and cost are the metrics popularly used to measure the performance of parallel computing systems. This work focuses on single encryptions and decryptions to determine the performance. Hence, speedup alone is considered to determine the performance of PACMATS algorithm. Speedup is the ratio of the time taken by the serial implementation of the algorithm to that of its parallel implementation. It is denoted by $S_p = T_s/T_p$. Where, 'T_p' is the parallel execution time and 'T_s' is the sequential execution time.

All the parallel implementations provided similar variations 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 on the performance of encryption using MPI, OpenMP and Java threads are shown in Figure. 4 and the decryption is shown in Figure 5.

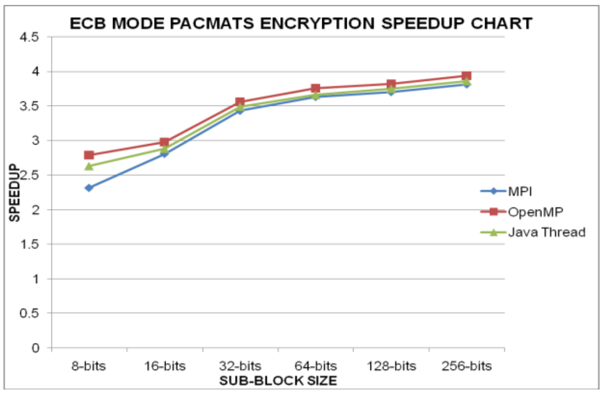


Figure 4. Performance of ECB mode Encryption of PACMATS

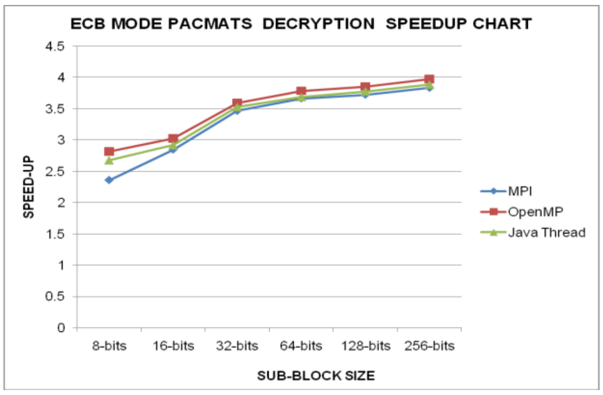


Figure 5: Performance of ECB mode Decryption of PACMATS

IV. CBC IMPLEMENTATION OF PACMATS

In ECB mode of PACMATS, 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 Figure 6 and the decryption process in Figure 7.

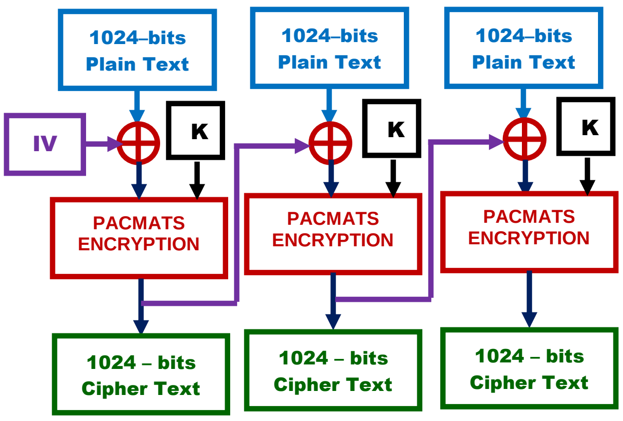


Figure 6: CBC mode Encryption of PACMATS

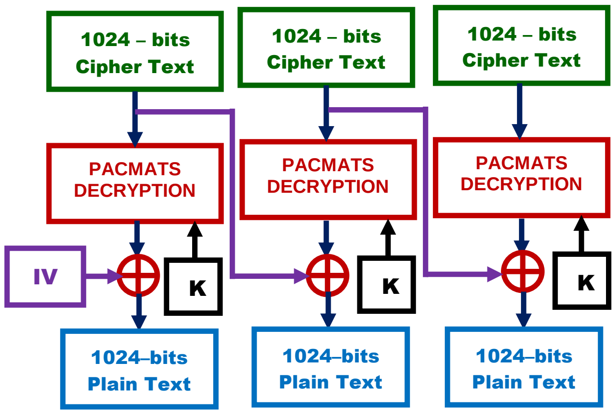


Figure 7. CBC mode Decryption of PACMATS

The decryption is just the reverse of the encryption, except that the feedback of the previous level is readily available for decryption, whereas in encryption it is not. The speedup results of CBC mode implementation of PACMATS in MPI, OpenMP and Java Threads are given in Table 2

TABLE II. CBC MODE IMPLEMENTATION OF PACMATS

SUB-BLOCK SIZE	SPEEDUP IN CBC MODE OF PACMATS					
	MPI		OpenMP		JAVA Threads	
	ENC	DEC	ENC	DEC	ENC	DEC
8 bits	1.26	2.25	1.51	2.69	1.42	2.54
16 bits	1.59	2.71	1.66	2.89	1.58	2.78
32 bits	1.83	3.34	1.96	3.44	1.91	3.4
64 bits	2.03	3.51	2.08	3.58	2.05	3.54
128 bits	2.1	3.56	2.18	3.69	2.15	3.62
256 bits	2.19	3.68	2.26	3.81	2.22	3.73

CBC Mode: Cyber Block Chaining Mode
ENC : Encryption DEC : Decryption

The performance of CBC mode encryption of PACMATS is reduced considerably 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 performance graphs of encryption and decryption of PACMATS in CBC mode is given in Figure 8 and Figure 9.

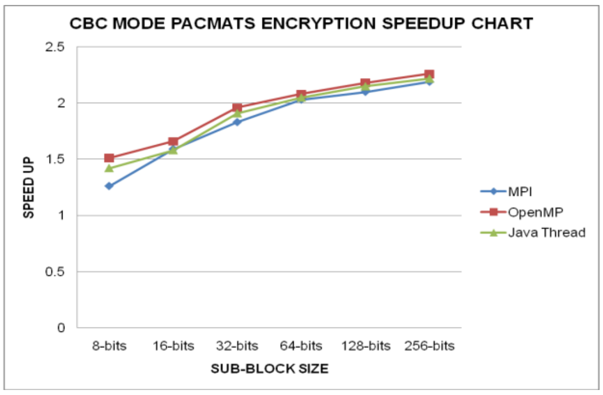


Figure 8. Performance of CBC mode Encryption of PACMATS

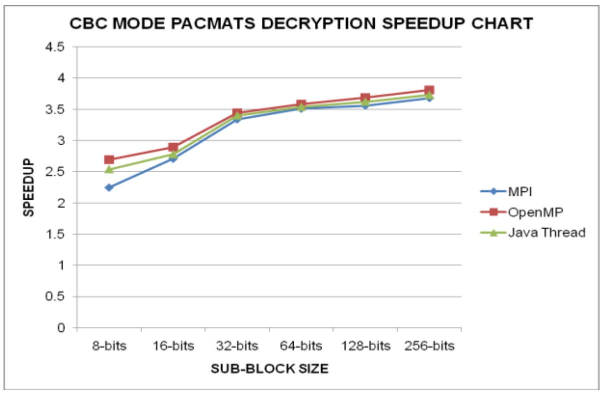


Figure 9. Performance of CBC mode Decryption of PACMATS

V. INTERLEAVED CBC IMPLEMENTATION OF PACMATS

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. Interleaving can be done in n-ways, wherein the 2-way and 4-way interleaving are adopted in this work.

A. Two-Way Interleaved CBC mode of PACMATS

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 Figure 10 and that of decryption in Figure 11.

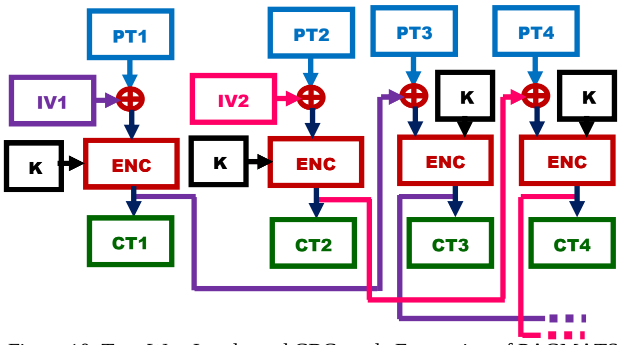


Figure 10. Two-Way Interleaved CBC mode Encryption of PACMATS

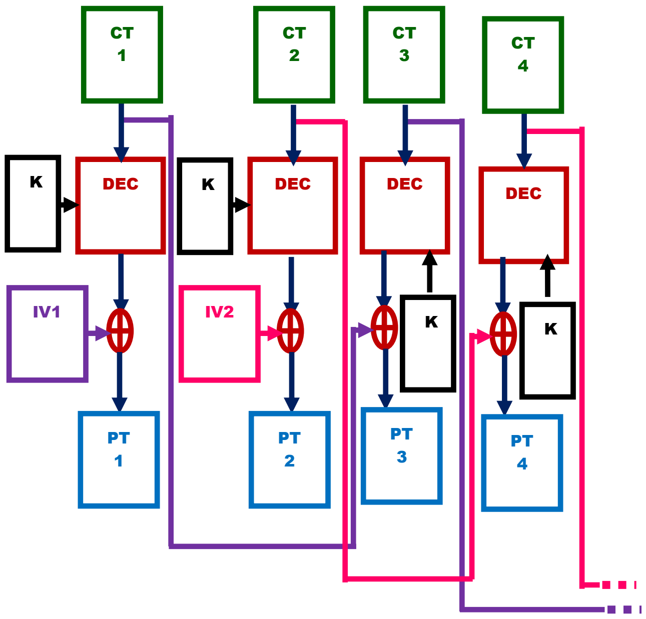


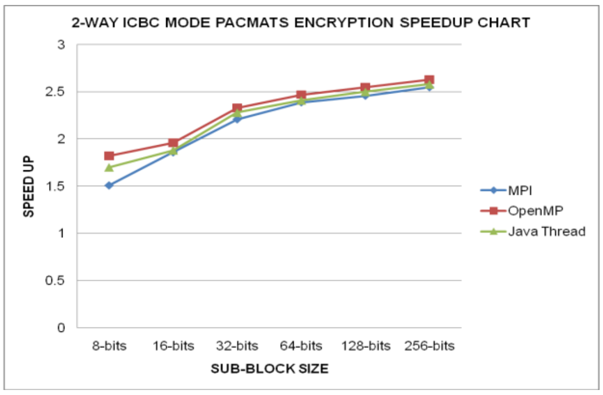
Figure 11. Two-Way Interleaved CBC mode Decryption of PACMATS

The decryption process in two-way interleaved CBC mode of PACMATS is just the reverse of the encryption process. The ciphertext feedback 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 PACMATS in MPI, Open MP and Java Threads are given in Table 3.

TABLE III. 2-WAY ICBC MODE IMPLEMENTATION OF PACMATS

SUB-BLOCK SIZE	SPEEDUP IN 2-WAY ICBC MODE OF PACMATS					
	MPI		OpenMP		JAVA Threads	
	ENC	DEC	ENC	DEC	ENC	DEC
8 bits	1.51	2.29	1.82	2.73	1.7	2.6
16 bits	1.86	2.76	1.96	2.94	1.88	2.83
32 bits	2.21	3.36	2.33	3.48	2.28	3.43
64 bits	2.39	3.55	2.47	3.67	2.41	3.58
128 bits	2.46	3.62	2.55	3.76	2.5	3.67
256 bits	2.55	3.73	2.63	3.87	2.58	3.78

ICBC Mode: Interleaved Cyber Block Chaining Mode
ENC : Encryption DEC : Decryption



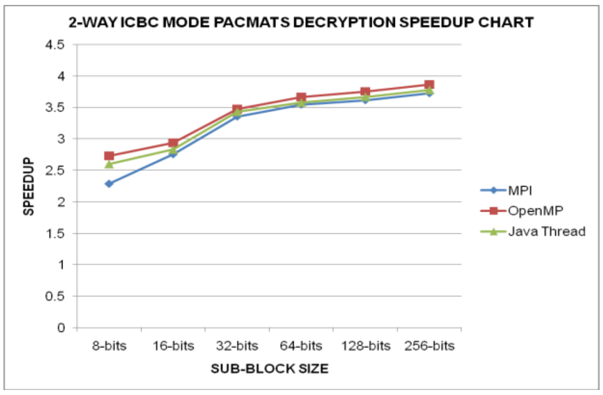


Figure 13. Performance of 2-Way ICBC mode Decryption of PACMATS

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

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

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 four-way interleaving, four Initialization Vectors are required to start the encryption and the decryption processes. In order to enhance the efficiency of execution of 4-way ICBC mode implementation of PACMATS in parallel computing environments, the number of processes or threads used for implementing the encryption or the decryption algorithms is increased by four times. The complexity of implementing 4-way interleaved CBC technique is considerably increased because of the additional operations that has to be performed for every level of interleaving. The structure of four-way interleaving of PACMATS for encryption is shown in Figure 14 and that of decryption in Figure 15.

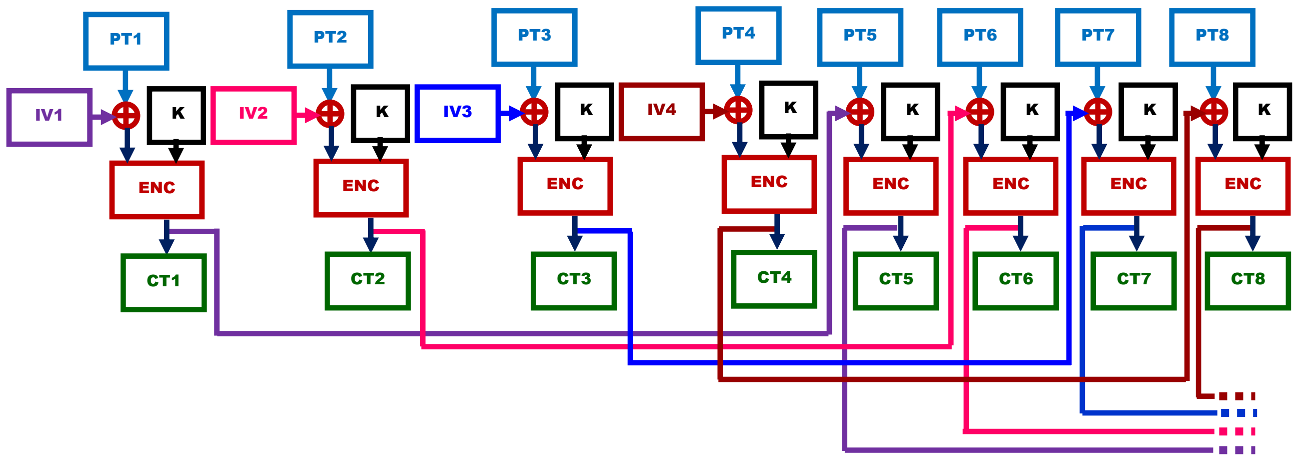


Figure 14: Four-Way Interleaved CBC mode Encryption of PACMATS

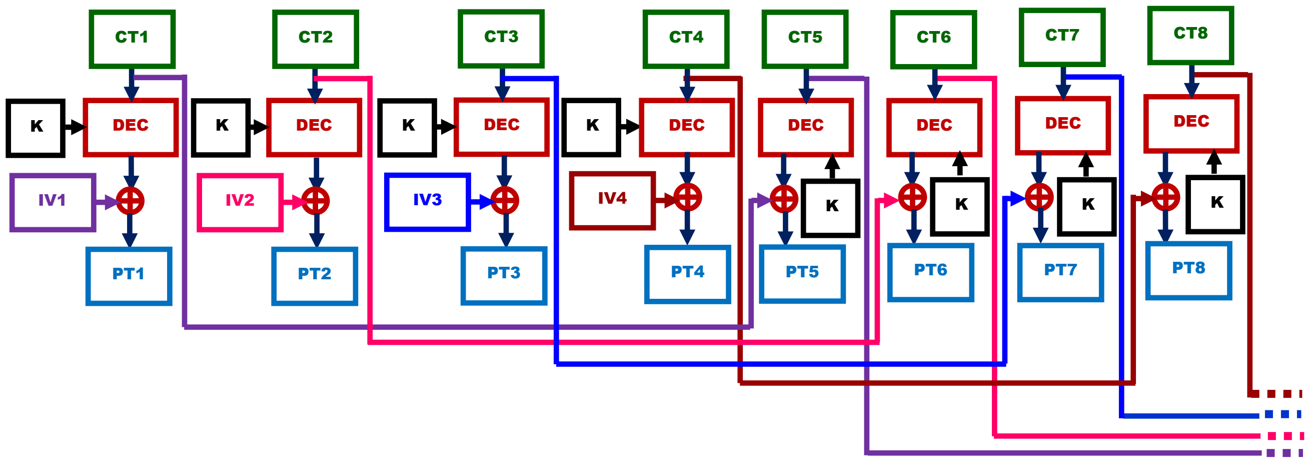


Figure 15: Four-Way Interleaved CBC mode Decryption of PACMATS

TABLE IV. 4-WAY ICBC MODE IMPLEMENTATION OF PACMATS

SUB-BLOCK SIZE	SPEEDUP IN 4-WAY ICBC MODE OF PACMATS					
	MPI		OpenMP		JAVA Threads	
	ENC	DEC	ENC	DEC	ENC	DEC
8 bits	1.77	2.33	2.13	2.78	1.99	2.65
16 bits	2.18	2.81	2.3	2.99	2.22	2.87
32 bits	2.59	3.42	2.73	3.56	2.67	3.48
64 bits	2.81	3.63	2.89	3.74	2.83	3.64
128 bits	2.88	3.68	2.99	3.81	2.94	3.73
256 bits	2.99	3.79	3.09	3.92	3.03	3.83

ICBC Mode: Interleaved Cyber Block Chaining Mode
ENC : Encryption DEC : Decryption

A considerable improvement is seen in the performance of four-way interleaved CBC mode implementation of PACMATS when compared with the simple CBC and two-way Interleaved CBC modes. This is shown in Table 4. The 4-Way ICBC mode encryption performance of PACMATS is shown in Figure 16 and that of decryption is shown in Figure 17.

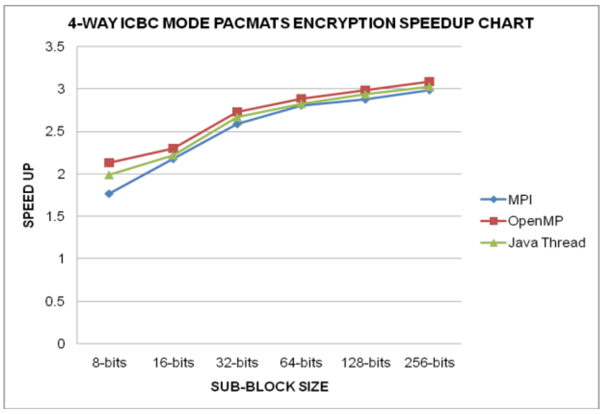


Figure 16. Performance of 4-Way ICBC mode Encryption of PACMATS

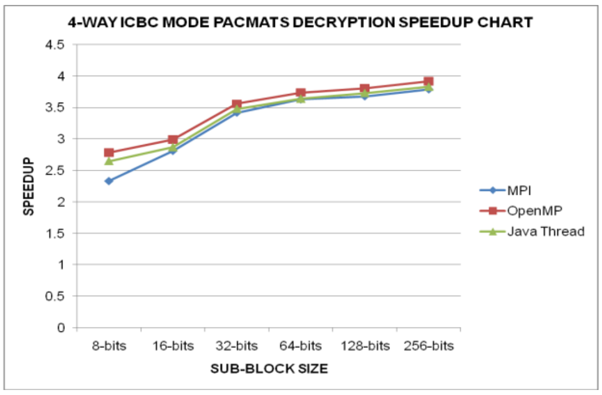


Figure 17. Performance of 4-Way ICBC mode Decryption of PACMATS

Increasing the level of the interleaving in CBC mode of PACMATS 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 processes in the current level.

VI. FUTURE SCOPE

In this work, PACMATS is implemented in multi-core machines. Its behavior in other computing environments can be tested and suitable improvements can be incorporated. The issue of interoperability with different architectures can be studied and suitable modifications can be made to the basic design and structure of the algorithm. Further, the possibility of a developing a unique algorithm with single adaptive key for all parallel computing environments can be researched.

VII. CONCLUSION

VIII.

PACMATS is an adaptive cryptographic algorithm that provides better security strength and performance in parallel computing environments. It requires 5.7 X 10^{288} years to break this cipher with brute force attack. PACMATS is a dynamic algorithm as its granularity and execution stages are decided during runtime using the bit patterns in the key. As the general reversible techniques are used, this algorithm is scalable. The algorithm is exclusively designed for software implementations and to avoid dependency problems in the parallel processing environments. PACMATS is both computation and communication intensive block cipher with Inter-block operations incurring more communication cost than Intra-block operations.

When executed in parallel computing environments the performance of PACMATS in 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] Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman, “An Introduction to Mathematical Cryptography”, Springer International Edition, Springer (India) Pvt. Ltd., New Delhi, 2008.
[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 2003.
[3] William Stallings, “Cryptography and Network Security-Principles and Practice”, 5th Edition, Dorling Kindersley (India) Pvt. Ltd., licensees of Pearson Education, 2011.
[4] Menezes A.J., Van Oorschot P.C., Vastone S.A., “Handbook of Applied Cryptography”, CRC Press, 1996.
[5] Schneier B., “Applied Cryptography : Protocols, Algorithms, and Source Code in C”, Second Edition, Wiley & Sons, 1995.
[6] 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.
[7] Thomas Rauber, Gudula Runger, “Parallel Programming – for Multicore and Cluster Systems”, International Edition, Springer (India) Pvt. Ltd. New Delhi, 2010.
[8] 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,
[9] Pionteck, T., Staake T., Stiefmeier T., Kabulepa L. D., Glesner M., “Design of reconfigurable AES encryption/decryption engine for mobile terminals”, in the proceedings of the International Symposium on Circuits and Systems, 2004.
[10] 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.
[11] Tetsuya Ichikawa, Tomomi Kasuya, and Mitsuru. Matsui. “Hardware evaluation of the AES finalists.” In Proc. Third Advanced Encryption Standard Candidate Conference, pages 279–285, USA, 2000.
[12] 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, USA, 2000.
[13] Swankoski E. J., Brooks R. R., Narayanan V., Kandemir M., and Irwin M. J., “A Parallel Architecture for Secure FPGA Symmetric Encryption”, proceedings of the 18th International Parallel and Distributed Processing Symposium, Santa Fe, New Mexico, 2004.
[14] Kotturi D., Seong-Moo Y., Blizzard J., “AES crypto chip utilizing high-speed parallel pipelined architecture” proceedings of the IEEE International Symposium on Circuits & Systems, 2005.

[15] 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", First International Symposium on Data, Privacy, and E-Commerce, 2007.

[16] Chonglei, M., J. Hai and J. Jennes, "CUDA-based AES Parallelization with fine-tuned GPU memory utilization", proceedings of the IEEE International Symposium on Parallel and Distributed Processing, Workshops and Ph. D. Forum, pp19-23, 2010.

[17] Julian Ortega, Helmuth Tefeffiz, Christian Treffiz, "Parallelizing AES on Multicores and GPUs", IEEE International Conference on Electro/Information Technology, Mankato, US, pp. 1-5, 2011.

[18] 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.

[19] Hua Li., "A parallel S-box architecture for AES byte substitution", paper presented at IEEE sponsored International Conference on Communication, Circuits and Systems, Chengdu, China, 2004.

[20] 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.

[21] 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.

[22] S. Ashokkumar, K. Karuppasamy, Balaji Srinivasan, V.Balasubramanian "Parallel Key Encryption for CBC and Interleaved CBC" International Journal of Computer Applications, Volume 2–No. 1, 2010.

[23] 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, Bialystok 2005.

[24] Bielecki W., Burak D., "Parallelization of Symmetric Block Ciphers", Computing, Multimedia and Intelligent Techniques special issue on Live Biometrics and Security, Vol. 1, Czestochowa University of Technology, June 2005.

[25] J. John Raybin Jose, E. George Dharma Prakash Raj, "PACMATS – An Adaptive Symmetric Block Cipher for Parallel Computing Environments" in the proceedings of the International Conference on Wireless Sensor Networks & Information Security, SASTRA University, Thanjavur, India, December 2013.