Indian Journal of Science and Technology, *Vol 7(S4), 99–105, April 2014*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

PACMATS: An Adaptive Symmetric Block Cipher for Parallel Computing Environments

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

[1]Department of Information Technology, Bishop Heber College (Autonomous), Tiruchirappalli, Tamilnadu, India; raybinjose@yahoo.com

[2]School of Computer Science, Engineering and Technology, Bharathidasan University, Tiruchirappalli, Tamilnadu, India

Abstract

Conventional symmetric cryptographic algorithms are designed considering the cryptographic technicalities alone rather than the computing capabilities of the modern processing devices. Hence the performances of these algorithms are inconsistent and vary enormously when implemented in different computing systems with different processing capabilities. Parallelized adaptive cryptographic algorithms that can be related with the characteristics of the present day computing devices are in great demand. "Parallelized Adaptive Cipher with Modular Arithmetic, Transposition and Substitution" (PACMATS) is a different class of cryptographic algorithm involving traditional techniques, which addresses these issues effectively. The size of the key and the plain text blocks are each 1024-bits. This symmetric block cipher is made adaptive by incorporating flexibility in deciding the size of the key and plain text sub-blocks and the number of rounds. Degree 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 the runtime. The performance of the PACMATS is analyzed with implementations in shared memory parallel programming environment using OpenMP, Java Threads and MPI.

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

1. Introduction

Cryptography deals with the art and science of hiding Information. Cryptographic techniques enable to preserve the confidentiality, integrity, authenticity and validity of our information in move or in store. Cryptographic algorithms are broadly classified into Symmetric or Private Key Algorithms and Asymmetric or Public Key Algorithms. The Private Key algorithms make use of a single key for encryption and decryption and they depend on the safe exchange and handling of the key. Private key algorithms can be further classified into stream ciphers which deal with single byte at a time and block ciphers which handle a group of bytes at a time. Asymmetric key algorithms make use of two keys. A public key is used to encrypt the message and it is available to all. The secret key is known only to the receiver and it is used to decrypt the information. Thus the issue of key distribution is overcome in asymmetric cryptography[1–3].

PACMATS is a Symmetric Block Cipher, which utilize reversible operations involving Modular Arithmetic, Substitution and Transposition techniques. Modular Arithmetic operations revolve around the remainder of division operation. Modular arithmetic operations are denoted using the relation: x mod m = r. Here 'x' is an integer from the set of integers 'Z'. A positive integer 'm' known as modulus is used to divide the value 'x' to produce a positive integer known as residue 'r' from the set

**Author for correspondence*

of residues 'Z_m'. The set of residues comprises all the elements between 0 and m-1. Modular arithmetic allows three binary operations such as addition, subtraction and multiplication to be applied on the elements of Z_m. After applying each operation, the result obtained is mapped to Z_m with the help of the modulus operator. There is always a many-to-one relationship between the elements of 'Z'and 'Z_m' and this relationship is referred to as congruence.

Addition and multiplication modulo operations can be used in cryptography, selecting a value from the set of residues 'Z_m' as key. In order to perform decryption the same operation is carried out with the inverse of the element as the key. Therefore it is essential to find the additive or multiplicative inverse of the key for decryption. Each element in modular arithmetic has a unique additive inverse and at times, the additive inverse of an element is the element itself. If 'x' and 'y' are two elements of the set 'Z_m', 'x' is said to be the additive inverse of 'y' and vice versa if: x+y = 0 (mod m).

An element may or may not have a multiplicative inverse. If 'x' and 'y' are the two elements of the set 'Z_m', then 'x' is said to be the multiplicative inverse of 'y' and vice versa if x * m = 1 (mod m). The simple method to determine whether or not a number 'x' in 'Z_m' has a multiplicative inverse is to compute the GCD of 'x' and 'm'. If gcd (x, m) = 1 then 'x' and 'm' are said to be relatively prime and 'x' has a multiplicative inverse; otherwise, the multiplicative inverse for x in Zm does not exist and these values of 'Z_m' cannot be used for the multiplicative modulo operations[1–4].

Every symbol is replaced with another one in Substitution. But, transposition jumbles the symbols in a block of data. Substitution causes confusion by making it more difficult in to find a connection between the key and the cipher text from one end and the key and the plaintext from the other end. Transposition produces diffusion and makes sure that there is no connection between the statistics of the symbols in the plaintext and the ciphertext[2,3].

Adaptive Cryptography is a trend in Cryptography, which makes the cryptographic algorithm and its key adaptive by dynamically scaling them, according to the requirements of the application or the processing device. Adaptive Cryptographic Techniques can be classified as (i) Inter-Algorithmic Adaptive Techniques and (ii) Intra-Algorithmic Adaptive Techniques. Inter-Algorithmic Adaptive Techniques involve more than one algorithm for different applications or processing environments5. Intra-Algorithmic Adaptive Techniques are used within the same algorithm by dynamically changing the sub-block size of the key or the plaintext based on the nature of the application or the processing environment in which it is executed. Intra-Algorithmic Adaptive Techniques are utilized in this research work.

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 utilizing the available parallel resources efficiently. Techniques used for explicit parallelism are (i) Per-Connection Parallelism (ii) Per-Packet Parallelism and (iii) Intra-Packet Parallelism.

Per-connection parallelism is a method in which each connection is provided with individual thread or process that executes separately on a single processor. This type of parallelization requires no changes to be effected in the existing algorithm or the software and. it does not make any effort to use sophisticated computing facilities. Per-packet parallelism is a method in which the connections distribute their packet load to several processors, and each packet is handled separately. Several Cryptographic algorithms available today support this type of parallelization. But cryptographic software that implements this type of parallelism is not yet available. Intra-packet parallelism is the most difficult concept to implement as it heavily depends on the algorithm design. It demands different approach in the implementation of the cryptographic algorithm, no longer relying on the flexibility of the hardware or operating system over which it is executed6–9. Intra-packet parallelism is employed in PACMATS with the ability to adjust the operations according to parallel environment in which it is executed.

The design of this paper is given as follows. Section II gives the Related Works. Section III highlights the drawbacks in Traditional Cryptography. Description of Parallel Adaptive Cipher with Modular Arithmetic, Transposition and Substitution (PACMATS) is given in section IV. Implementation and Discussion is found in section V and Section VI concludes the paper.

2. Related Works

Efforts to parallelize existing cryptographic algorithms have been pursued by several researchers from year 2000 onward. The prominent of these efforts are classified

broadly as Hardware or Software Parallel Cryptographic implementations involving several approaches under them as shown in figure 1.

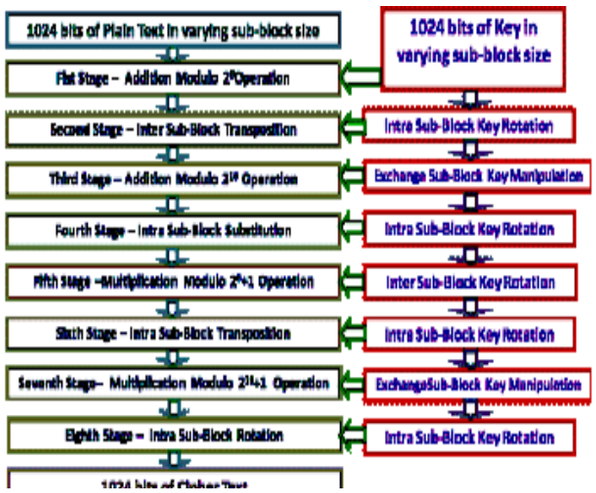

Figure 1. Classification of parallel implementations of cryptographic algorithms.

HoWon Kim introduced a special-purpose microprocessor optimized for the execution of the ciphers. This crypto processor is made up of a 32-bit RISC processor block and a coprocessor block dedicated to SEED and Triple-DES algorithms[10]. Pionteck in their work has given a hardware design of AES with reconfigurable encryption/decryption engines which supports all key sizes. 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 32 bit five stage pipeline RISC Processor and the area which is used for the RCU is less than 27% of the total area[11].

Application-Specific Integrated Circuits (ASIC) are developed for specific use. When ASICs are used to implement cryptographic algorithms they provide robust operation and much of the overhead involved in hardware implementation is reduced. The work carried out on ASIC implementation of DES, 3DES, IDEA and the candidates of AES by S. Mukherjee,, T. Ichikawa and B. Weeks et al are prominent in this category[12–14].

Field Programmable Gate Array (FPGA) logic cells are reconfigurable platforms that provide cost and performance effective methods for implementing cryptography. Ciphers such as DES, Triple DES, and AES are parallelized using FPGAs by Swankoski et al[15]. Virtex-II Pro, FPGA Platform with Verilog HDL and Block RAM resources are used for implementation. A hardware design of AES in chip is proposed by Kotturi Hierarchical simultaneous key generation is used for implementation in ten separate units in XC2VP70 device with speed grade 7 in Virtex II Pro FPGA. Every unit can run a single round of the algorithm and ten rounds of the algorithm can be executed in parallel in a chip with external pipelined design. The throughput rate achieved in this method is higher than most other implementations[16]. In yet another implementation by Chi-Wu, 128-bit AES was decomposed into four 32-bits 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[17].

Multi-core Processors and Graphical Processing Units (GPUs) are 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[18,19]. OpenMP is used to extract parallelism from Multi-Core Processors[20].

The most time-consuming portions of the cryptographic algorithms without involving the I/O functions are the loops. Loop parallelization is done for cryptographic algorithms such as DES, Triple DES, IDEA, AES, RC5, Blowfish, GOST and LOK191 by Bielecki. 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.

3. Drawbacks of Traditional Cryptography

Traditional view on the efficiency of a cryptographic algorithm focuses only on the complexity of the algorithm and the strength and the secrecy of the key. All conventional

symmetric cryptographic algorithms are developed before the year 2000 when computers were built around 8, 16 or 32 bit processors. But now, Cryptographic algorithms can be executed much faster on modern computers. The Computers of today, and that of future, is not that of 32-bit desktops, but of multi-cored chips and multiprocessor machines whose processing capacity is 64 or 128 or more bits. Parallelizing the cryptographic algorithms is a means to utilize these systems effectively. Moreover, the rate of encryptions and decryptions carried out per unit time in communication systems has increased exponentially. This imposes additional overhead in the information exchange and causes congestion. A way out of this trouble is to develop a new class of adaptive parallel algorithms that can reduce the time for ciphering, without compromising the security strength[6].

The computing capacity of the devices used for different applications varies from low end hand held devices to the high end multi-processor systems with several hundreds of processors. A number of Cryptographic Algorithms are in existence today and different Cryptographic algorithms are implemented for different devices, applications and environments. When information has to move through these differences it is inevitable that the enciphering and deciphering has to be preformed repeatedly several times to switch it over from one cipher to another[5,6]. This makes it evident the need for using the same cipher which is adaptive in all situations. The adaptive nature of such a cipher can be achieved by flexibly changing the sub-block size of the key and the plain-text.

4. Parallel Adaptive Cipher With Modular Arithmetic, Transposition and Substitution (PACMATS)

PACMATS is a symmetric block cipher with block length and key size each of 1024 bits. 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 2n 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. The sub-block generation routine is run to generate key and plain text sub-blocks; before they are involved in operations at each stage.

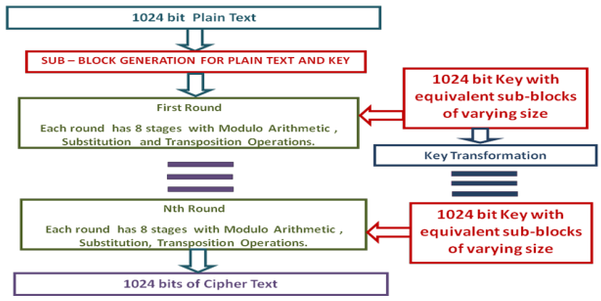

Figure 2. General block diagram of PACMATS.

Each round in PACMATS has eight stages as depicted in figure 3. In the first stage sub-blocks are divided into chunks of 8 bits and addition modulo 2^8 operations is performed with the key sub-blocks and the plain text sub-blocks in the pattern decided by the initial and the final bits of the key. If both these bits are of same value then addition modulo 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, transposition is carried between the sub-blocks. Following this an exchange manipulation is carried out between different key sub-blocks. 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 once again based on the key value as it is done before the second stage.

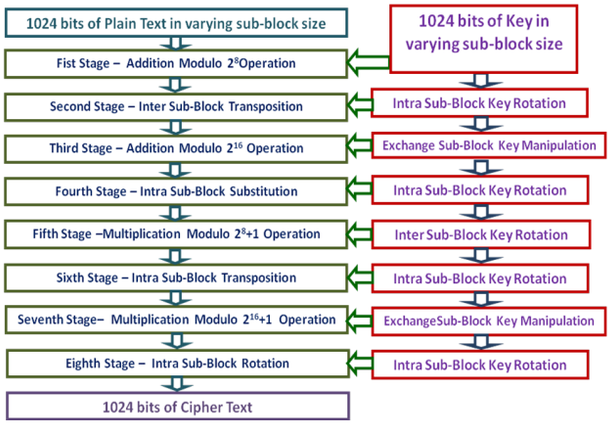

Figure 3. Stages in each round of PACMATS.

In the fourth stage intra plain text substitution is carried out based on the key sub-block values. An inter sub-block rotation is performed for the key before the fifth stage and multiplication modulo 28+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 216+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 then experience Intra sub-block rotation before they are utilized for intra sub-block rotation to the left or right based on the initial few bits of each key sub-block in the eighth and the final stage of each round. Brief algorithmic depiction of PACMATS with single round is given as follows:

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=8 bits.
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≥32bits && c<64bits && s>100MHz && s≤1000 MHz then b=32bits.
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 for implementing PACMATS".

Steps in Single Round of PACMATS:

1. Addition modulo 28 operation with key and plaintext sub-blocks.
2. Intra sub-block rotation on key sub-blocks.
3. Inter sub-block Transposition on plaintext sub-blocks.
4. Exchange sub-block operation on key sub-blocks.
5. Addition modulo 216 operation with key and plaintext sub-blocks.
6. Intra sub-block rotation on key sub-blocks.
7. Intra sub-block substitution on plaintext sub-blocks.
8. Inter sub-block rotation on key sub-blocks.
9. Multiplication modulo 28+1 operation with key and plaintext sub-blocks.
10. Intra sub-block rotation on key sub-blocks.
11. Intra sub-block Transposition on plaintext.
12. Exchange sub-block operation on key sub-blocks.
13. Multiplication modulo 216+1 operation with key and plaintext sub-blocks.
14. Intra sub-block rotation on key sub-blocks.
15. Intra sub-block rotation on plaintext sub-blocks.

For all normal implementation of PACMATS in Personal Computers, single round execution is sufficient as it provides the required security level. Utilization of straight and reverse modulo arithmetic operations and inter sub-block and intra sub-block transpositions, substitutions and shift operations makes PACMATS both computation and communication intensive in between the different processing elements available for execution in parallel environments.

5. Implementation and Discussion

PACMATS is implemented in shared memory architecture using MPI, OpenMP and Java Threads programming with different plain text sub-block sizes and compared with sequential results. The speedup of various combinations of executions are analyzed and compared and the results are given in Table 1.

Table 1. Implementation of PACMATS (Single Round)

SUB-BLOCK SIZE	SPEEDUP IN ECB MODE					
	MPI		OpenMP		JAVA Threads	
	ENC	DEC	ENC	DEC	ENC	DEC
8 bits	3.32	3.36	2.78	2.82	2.17	2.22
16 bits	3.65	3.71	2.95	3.02	2.54	2.61
32 bits	3.91	3.97	3.42	3.47	2.89	2.95
64 bits	3.93	3.98	3.69	3.75	3.35	3.41
128 bits	3.88	3.96	3.51	3.55	3.21	3.26
256 bits	3.62	3.67	3.16	3.20	2.86	2.90

ECB Mode: Electronic Code Book Mode

ENC : Encryption DEC : Decryption

When PACMATS is implemented in a machine with multi-core processor with 4 cores yielded a better speedup in MPI than OpenMP and Java threads, as thread implementations behave better only for a communication intensive operation. For all implementations when the sub-block size is small the speedup is low but it gradually increases and reaches maximum as the sub-block size in increased to 64 bits for MPI, OpenMP and Java Threads. If the sub-block size is increased further the speedup decreased. A comparative representation of encryption using MPI, OpenMP and Java threads is shown in Figure 4 and the decryption is shown in Figure 5.

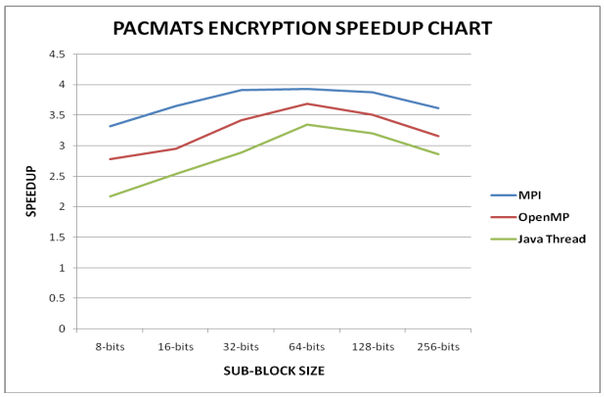

Figure 4. Performance of PACMATS encryption.

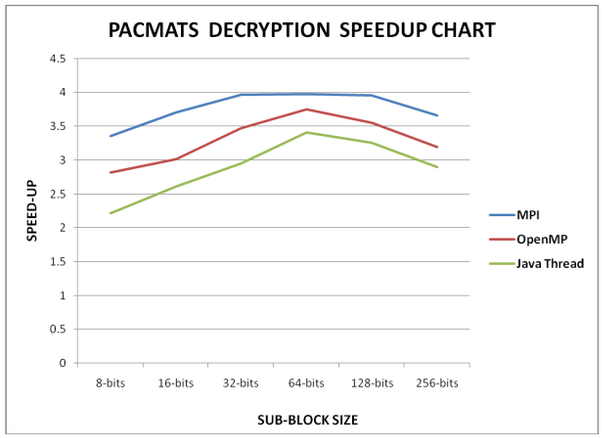

Figure 5. Performance of PACMATS decryption.

The advantages of PACMATS are its adaptive nature, ability to run on all parallel architectures, flexibility in deciding the size of the key and plain text sub-blocks and the number of rounds. The level of Intra-packet parallelization, variety in grain size and the required security strength are obtained by suitably deciding the sub-block size. Flow of the algorithm is made dynamic by determining the execution steps through each key value at the runtime.

The performance of parallelized traditional symmetric block ciphers cannot be directly compared with that of PACMATS' as those algorithms are static and cannot adapt themselves to the nature of the computing environments in which they are executed. Any how the results o obtained by the research of Bielecki et al[21,22] in similar environments are given in Table 2 for reference. soft Word document.

Table 2. IPerformance of parallel implementations of traditional symmetric block ciphers.

Cryptographic Algorithms	Speedup in ECB Mode	
	Encryption	Decryption
DES	1.65	1.65
Triple DES	1.70	1.70
IDEA	1.65	1.70
AES – 128 bits	3.10	3.30

6. Conclusion

PACMATS is an adaptive cryptographic algorithm that provides better security strength and performance in parallel computing environments. It requires 5.7 X 10288 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 a block cipher in which, computation and communication intensive operations are equally distributed over the different stages in each round of operations and hence the performance of MPI implementations are better than implementations with OpenMP and Java Threads.

6. References

1. Stallings W. Cryptography and Network Security-Principles and Practice. 5th ed. Pearson Education Licensees from Dorling Kindersley (India) Pvt. Ltd. 2011.
2. Schneier B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. 2nd ed. Wiley and Sons. 1995.

3. Menezes AJ, Van Oorschot PC, Vastone SA. Handbook of Applied Cryptography. CRC Press. 1996.
4. Hoffstein J, Pipher J, Silverman JH. An Introduction to Mathematical Cryptography. New Delhi: Springer International Edition. Springer (India) Pvt. Ltd; 2008.
5. Khakurel S, Tiwary PK, Maskey N, Sachdeva G. Security vulnerabilities in IEEE 802.11 and adaptive encryption technique for better performance. The proceedings of IEEE Symposium on Industrial Electronics and Applications (ISIEA 2010); 2010 Oct 3-5; Penang, Malaysia.
6. Seidel EC, Gregg JN. Preparing tomorrow's cryptography: parallel computation via multiple processors, vector processing, and multi-cored chips.2003 May 13.
7. Rauber T, Runger G. Parallel programming – for multicore and cluster systems. New Delhi: International Edition. Springer (India) Pvt. Ltd. 2010.
8. Wilkinson B, Allen M. Parallel programming – techniques and applications using networked workstations and parallel computers. 2nd ed. New Delhi: Pearson Education. 2005.
9. Grama A, Gupta A, Karypis G, Kumar V. Introduction to Parallel Computing. 2nd ed. New Delhi: Pearson Education. 2003.
10. Kim HW, Choi YJ, Chung K, Ryu HS. Design and implementation of a private and public key crypto processor and its application to security system. The 3rd International Workshop on Information Security Applications (WISA 2002); 2002 Aug 28-30; Jeju, Korea. p. 515–31.
11. Pionteck T, Staake T, Stiefmeier T, Kabulepa LD, 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.
12. Mukherjee S, Sahoo B. A survey on hardware implementation of IDEA Cryptosystems. Information Security Journal: A Global Perspective. 2011; 20(4-5):210–8.
13. Ichikawa T, Kasuya T and Matsui M. Hardware evaluation of the AES finalists. Proceedings of Third Advanced Encryption Standard Candidate Conference (AES3); 2000 Apr 13–14; New York, USA. p. 279–85.
14. Weeks B, Bean M, Rozylowicz T, Ficke C. Hardware performance simulations of round 2 advanced encryption standard algorithms. Proceedings of Third Advanced Encryption Standard Candidate Conference (AES3); 2000 Apr 13–14; York, USA.
15. Swankoski EJ, Brooks RR, Narayanan V, Kandemir M, Irwin MJ. A parallel architecture for secure FPGA symmetric encryption. 18th International Parallel and Distributed Processing Symposium, (IPDP'04); 2004; Santa Fe, New Mexico.
16. Kotturi D, Seong-Moo Y, Blizzard J. AES crypto chip utilizing high-speed parallel pipelined architecture. IEEE International Symposium on Circuits and Systems ISCAS; 2005.
17. 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; 2007.
18. Chonglei M, Hai J and Jennes J. CUDA-based AES Parallelization with fine-tuned GPU memory utilization. IEEE International Symposium on Parallel, Distributed Processing, Workshops and Ph.D. Forum (IPDPSW); 2010. p.19–23.
19. Ortega J, Tefeffiz H, Treffiz C. Parallelizing AES on Multicores and GPUs. Proceedings of the IEEE International Conference on Electro/Information Technology (EIT); 2011 May 15–17; Mankato, US. p. 1–5.
20. Li H, Li JZ. A new compact dual-core architecture for AES encryption and decryption. Can J Electr Comput Eng. 2008; 33(3-4):209–13.
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. Bialystok; 2005.
22. Bielecki W, Burak D. Parallelization of symmetric block ciphers. Computing, Multimedia and Intelligent Techniques special issue on Live Biometrics and Security. Czestochowa University of Technology; 2005.