International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 02 Issue: 06 | Sep-2015 www.irjet.net p-ISSN: 2395-0072

PARALLELIZATION OF ENCRYPTION AND HASHING ALGORITHM USING GPU

Mr. Vijay J.Bodake[1], Prof.D.B.Bangal[2] ,Mr.K.B.Dhatrak[3]

[1] *HOD, Computer Engg Department, KVN Naik polytechnic Nashik, Maharashtra, India*
[2] *Principal, KVN Naik Polytechnic Nashik, Maharashtra, India*
[3] *HOD, Electrical Engg Department, KVN Naik Polytechnic Nashik, Maharashtra, India*

Abstract - With the development of the GPGPU (General-purpose computing on graphics processing units), more and more computing problems are solved by using the parallel property of GPU (Graphics Processing Unit). CUDA (Compute Unified Device Architecture) is a framework which makes the GPGPU more accessible and easier to learn for the general population of programmers. This is because it builds on C and hides many of the complicated details of how the GPU works from a CUDA developer. Using the unique properties of the GPU through CUDA has greatly increased the efficiency of many computational problems. The target in this project is to study and analyses the majority of algorithms related to the cryptography and then to design and make an implementation of algorithm in CUDA. Finally, this reach will compare the performance between the GPU implementation and the CPU implementation in order to look into the possibility of improving the performance of algorithms. The survey done in three cryptography algorithm AES, SHA, MD5.

Key Words: Advanced Encryption Standard (AES), Graphics Processing Unit (GPU), Parallel Computing, CUDA and cryptography.

1. INTRODUCTION

This section introduces to the entire background of the proposed work. It highlights the overall description of entire work. It also highlights the concept of cryptography, hash algorithm and GPU.

1.1 Cryptography

Cryptography is the study of mathematical techniques focused on information security, including confidentiality, data integrity, and authentication. An implementation of cryptography is typically comprised of computationally intensive algorithms which are used by applications when encrypting, decrypting, and hashing data. Some implementations also include authentication and verification techniques. Information security is the hot topic of research in the field of computer science and technology, and the data encryption is one of the most important methods for information security. Since a new kind of encryption algorithm, i.e. Advanced Encryption Standard (AES), has been proposed for replacing the previous encryption of Data Encryption Standard (DES) in 2001, more and more applications are starting to use AES instead of DES to protect their information security in the past ten years. Currently, the implementations of AES are based on CPU because CPU is regarded as the computing component in the computer system from the traditional point of view. With the rapid growth of information data, more and more applications require encrypting data with the performance of more and more high speed. The traditional CPU based AES implementation shows the poor performance and cannot meet the demands of fast data encryption. Therefore, how to develop a new method for high performance is a challenging topic of research, which are interesting more and more researchers in developing new approaches for fast AES encryption.

1.2 Literature Survey

There have been several attempts utilizing GPU, especially in NVIDIA CUDA framework for AES cryptography, and then showed the performance improvement over a traditional CPU. Most of them have applied a traditional AES CUDA-GPU implementation, i.e. dividing plaintext into an equal-block size, and then each individual block will be encrypted in each AES GPU block with multiple threads with/without operational modes [10-13]. Notice that all of them investigated on different GPU models with the comparative performance illustration. Some proposals have also considered other performance factors for example, S. A. Manavski et al. [17] proposed AES implementation on CUDA-GPU utilizing an off chip constant (slow) memory without operational mode consideration[16]. In 2009, A. D. Biagio et al. [18] evaluated the performance of NVIDIA 8800 GT GPU of parallel AES implementation using on-chip shared memory with CTR mode. Similarly, P. Maistri et al. [16] also investigated on AES parallelism in CTR mode but with NVIDIA 9600 GT and 260 GTX comparatively. In addition, C. Mei et al. [19] evaluated parallel AES design implementation of NVIDIA GeForce 9200M given the

discussion of various memory usages. Note that there is a difficulty to acquire the actual CUDA implementation. In addition, they do not practically discuss the actual parallel algorithm for performance evaluation among various parallel AES implementations.

2. AES ALGORITHM

AES, i.e. Rijndael algorithm is a symmetric key cryptography. The AES standard comprises three block ciphers, i.e. AES-128, AES-192 and AES-256. The encryption of AES is carried out in blocks with a fixed block size of 128 bits each. The AES cipher calculation is specified as a number of repetitions of transformation rounds that convert the input plaintext into the final output of cipher text. Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to transform the cipher text back into the original plaintext using the same encryption key.

The AES encryption perform on the CPU and GPU.If User select the GPU module for execution then encryption perform parallel. The well-formed 64 byte data blocks are transferred to GPU for encryption. The number of data blocks decides the number of threads to be created. Each thread in kernel then performs operations on its corresponding data block logically at same time. The Key Expansion is performed on CPU, as its computations are inherently serial. The sub-functions of AES are implemented as simple device functions. After implementation of kernel, the data blocks are copied back to CPU.

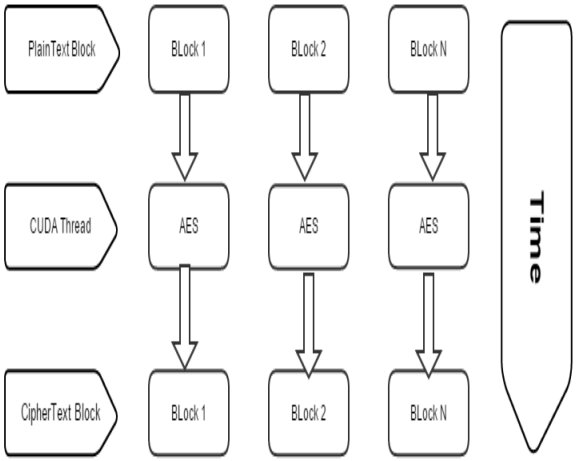

Fig -1. AES on GPU Logical Execution

The AES round key and lookup tables are stored within constant memory rather than slow global memory to optimize the execution of AES on GPU. A data structure is maintained to reconstruct the records from these data blocks after encryption. Below fig 1 shows the logical execution of the AES on GPU with the help of CUDA. Then each thread is divided into its subpart and then each subpart is executed parallel for the high speed to decrease the process time of encryption and decryption. The AES perform the four functions SubBytes, ShiftRows, MixColumns and AddRoundKey. These functions are executed parallel. NVIDIA's new Kepler GK110 GPU's has introduced dynamic parallelism technology that enables CUDA kernel to launch other kernels using CUDA runtime API. Till now, AES sub functions were implemented as device functions on GPU due to lack of dynamic parallelism support. Now on this new dynamic parallelism supported GPU's, System propose to implement these sub-functions as CUDA kernel and launch these sub-function kernels from parent AES kernel. The fig 2 shows the thread parallel processing of the kernel.

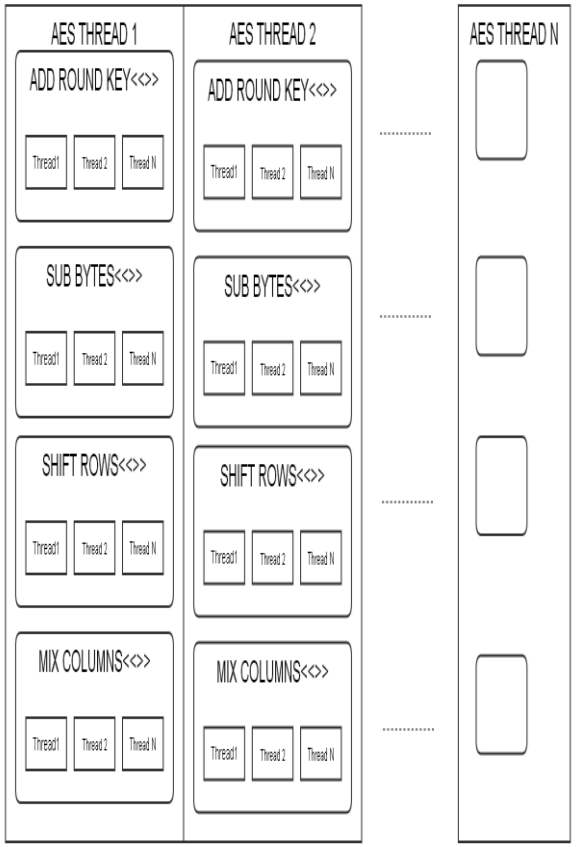

Fig-2 AES parallel Thread Process

3. MD5 ALGORITHM

In MD5, the input message is broken up into chunks of 512-bit blocks (each with sixteen 32-bit sub-blocks). After a series of operations, MD5 produces a 128-bit message digest with four concatenated 32-bit blocks for the integrity of a _le. To compute the digest of a message, padding bits are appended first to make the messages length congruent to 448, modulo 512, and then the length bits. A 64-bit portion is appended to indicate the length of the actual message. MD5 algorithm operates on a 128-bit state which is divided into four 32-bit words (denoted as A, B, C and D) and initialized. Each 512-bit message block is applied in turn to modify the state. The processing of a message block consists of four similar rounds, each of which is composed of 16 similar operations based on a non-linear function F, modular, addition, and left rotation. At last, MD5s output is produced by cascading A, B, C and D after the final round.

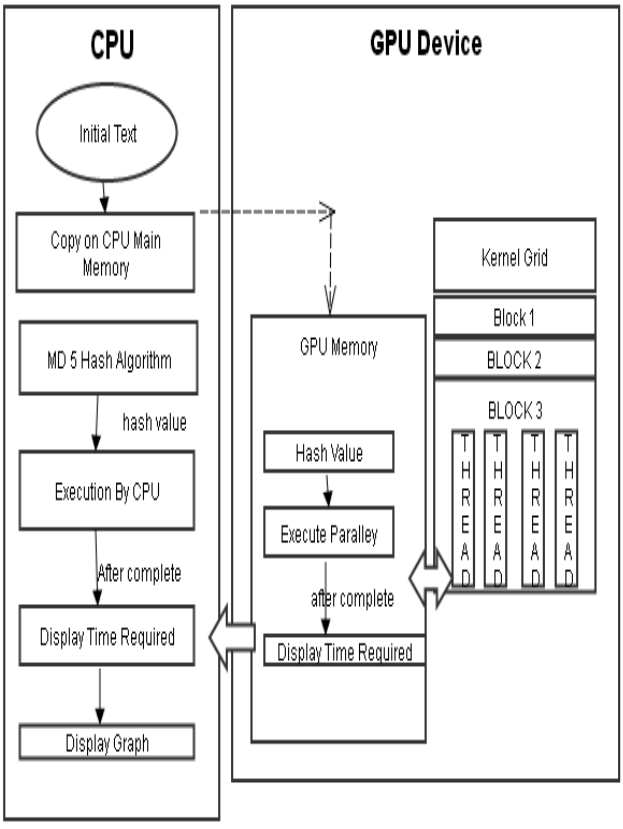

Fig-3 MD5 System Architecture

There are four steps in parallelized hybrid MD5 encryption algorithm. First, key initialization and expansion are accomplished by MD5 key encryption module and key expansion module in Host (CPU). Thus the hashed and expanded new key is prepared and ready for the initialization procedure. Second, initialization procedure generates Sub-keys, which is also operated by the host (CPU). Third, Sub-keys and input data objects (a number of input data blocks) are copied into GPU Memory. Encryption Module is coded as CUDA kernel functions to encrypt the input message and generate outputs (encrypted message) on GPU. Finally, the encrypted message is copied back to the host and delivered to users as shown in fig 3.

4. SHA-1 ALGORITHM

The Secure Hash Algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS PUB 180) in 1993; a revised version was issued as FIPS PUB 180-1 in 1995 and is generally referred to as SHA-1. The algorithm takes as input a message with a maximum length of less than 264 bits and produces as output a 160-bit message digest. The input is processed in 512- bit blocks. The heart of the algorithm is a module that consists of four rounds of processing 20 steps each. The logic is illustrated in Figure 4.9.In that _g user takes the input then calculate the hash value from the hash function. After calculating the hash value then system execute the SHA-1 algorithm to crack that password with the help of hash value. Then system finds multiple combinations of the password and try to compare with hash value. If it get then it display the password and time required for the total process. For this process it used the above algorithm for message digest.

There are four steps in parallelized hybrid SHA-1 encryption algorithm. First, key initialization and expansion are accomplished by SHA-1 key encryption module and key expansion module in Host (CPU). Thus the hashed and expanded new key is prepared and ready for the initialization procedure. Second, initialization procedure generates Sub keys, which is also operated by the host (CPU). Third, Sub-keys and input data objects (a number of input data blocks) are copied into GPU Memory. Encryption Module is coded as CUDA kernel functions to encrypt the input message and generate outputs (encrypted message) on GPU. Finally, the encrypted message is copied back to the host and delivered to users as shown in Fig 4.

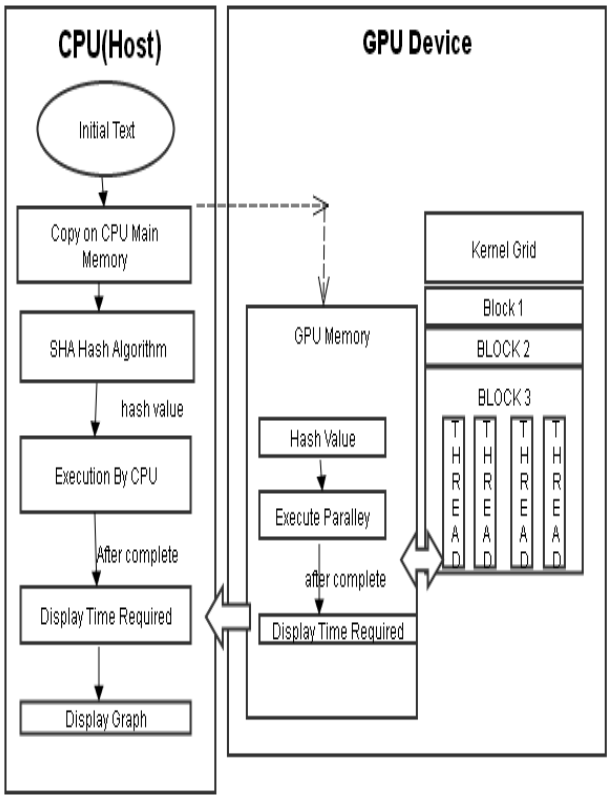

Fig-4 SHA-1 algorithm Architecture

5. RESULTS AND ANALYSIS

In this section, the actual results obtained from the comparison of CPU, GPU and optimized GPU implementations are presented. The CPU used for the experiments was a 2.8 GHz Intel CORE I3 processor with 4 cores, 4 GB total memory. The GPU used for the implementation was a GTX 680 device with 1536 CUDA Cores.

5.1 AES Algorithm

However, the presented computational system composed of CPU and GPU devices can be used to any massively parallel and intensive-data computations we used it to develop efficient cryptanalysis and cryptography. The results of evaluations of selected encryption and decryption AES algorithm is described in section 2.The aim of the first series was to test the efficiency of CPU-based implementation of the symmetric block ciphers AES and various modes of this algorithms operations. The table 1 shows the time required by CPU encryption and Decryption. The results shows that decryption takes much time than encryption.

Table -1: AES Encryption and Decryption on CPU and GPU

Datasize	Encryption by CPU(ms)	Decryption by CPU(ms)	Encryption by GPU(ms)	Decryption by GPU(ms)
10 MB	6.705	13.748	0.986	0.987
20 MB	12.868	26.076	1.916	1.92
100 MB	68.937	144.106	10.257	10.21
200 MB	123.632	251.978	17.734	17.765
300 MB	189.748	364.066	26.00	27.106

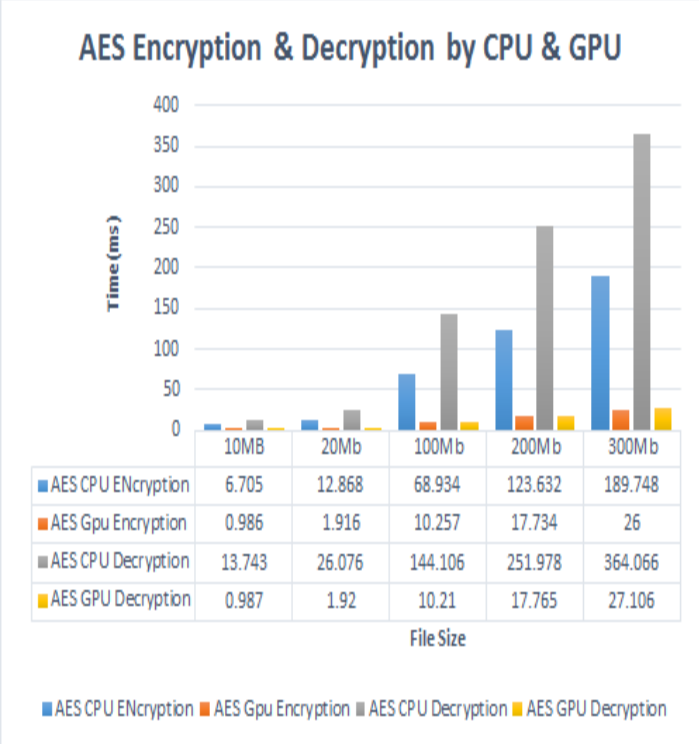

Chart -1: AES algorithm result

5.2 Hashing Algorithm

In this report discuss the efficiency of parallel implementations of selected password recovery algorithms. The only reasonable technique for recovering a password from hash is to scan all potential password, compute their hash, and test the coincidence. In general, cryptographic hash functions include integer and binary operations such as: addition modulo power of two, bit shift and rotation, bitwise xor, bitwise or, bit negation and words permutation. All those operations are natively supported by GPU processors. Multiple tests were performed for parallel implementations of password recovery from MD5, SHA-1.The aim of the first series of tests was to compare the performance achieved by Intel core i3 central processing units (CPU).Then multiple tests were performed for password recovery algorithms. The

performance of CPU-based and GPU-based versions of MD5, SHA-1, and hash techniques for hashes generation was evaluated. The multi-threaded implementations of the MD5 and SHA-1 were executed on GPU, each composed on Nvidia GPU. Then compare the performance of CPU-based and GPU-based algorithms for password recovery. Two techniques for hash generation were considered: MD5 and SHA-1. The number of hashes generated per second running MD5 and SHA-1 algorithms by CPU and NVIDIA Gforce740m are presented in chart 2 and chart 7 respectively

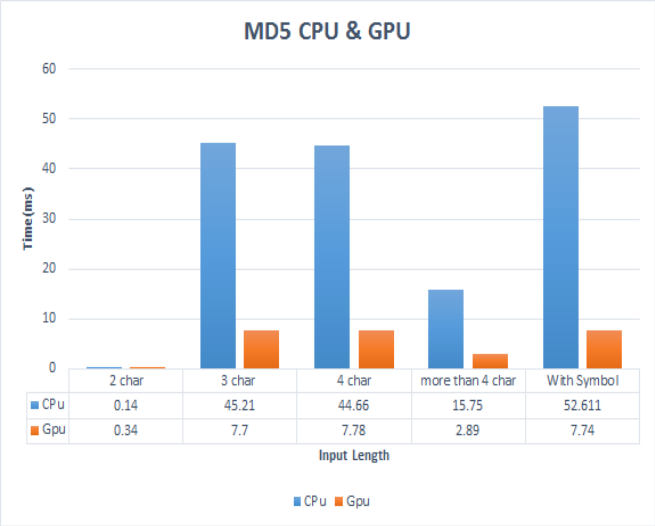

Chart -2:MD5 Algorithm Results

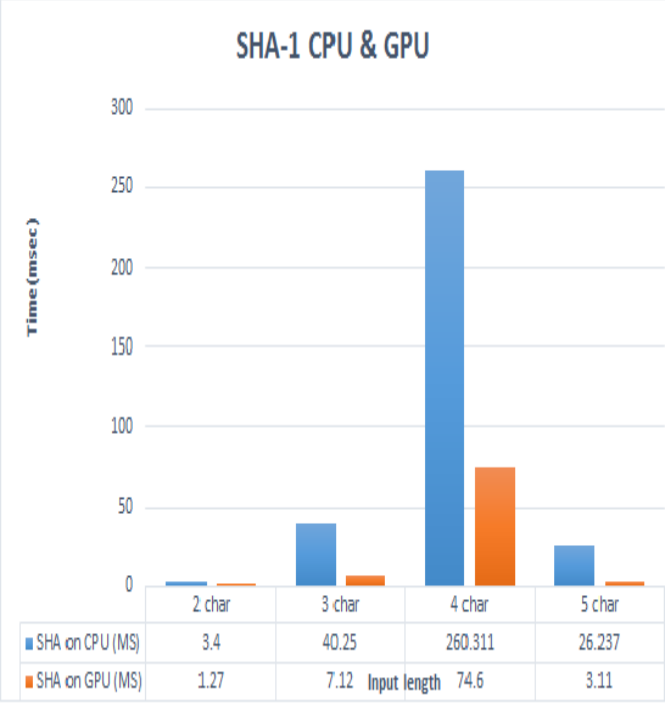

Chart -3: SHA-1 Results

6. CONCLUSIONS

The main goal of the project was to present the wide applicability of the GPU technology to cryptography and cryptanalysis. With the results compared the efficiency of the CPU based with the GPU devices, for AES algorithm, MD5 algorithm and SHA-1 algorithm. The results confirmed that the modern unified GPU architecture can perform as an efficient cryptographic acceleration board. Moreover, system showed that hybrid computer offer a new opportunity to increase the performance of parallel implementations, by combining traditional CPU and efficient GPU devices.

REFERENCES

[1] Bin Liu, Student Member, IEEE, and Bevan M. Baas, Senior Member, IEEE, Parallel AES Encryption Engines for Many-Core Processor Arrays, IEEE TRANSACTIONS ON COMPUTERS, VOL. 62, NO. 3, MARCH 2013

[2] NIST, Advanced Encryption Standard (AES), http://csrc.nist.gov/publications/_ps/_ps197/_ps-197.pdf, Nov. 2001

[3] NIST, Data Encryption Standard (DES), http://csrc.nist.gov/ publications/_ps/_ps463/_ps46-3.pdf, Oct. 1999.

[4] J D. Owens, D. Luebke, N. Govindaraju et al. A Survey of General- Purpose Computation on Graphics Hardware. Computer Graphics Forum, Vol. 26, No. 1, pp. 80-113,2007.

[5] L. Marziale, G. G. Richard III, and V. Roussev. Massive Threading: Using GPUs to Increase The Performance of Digital Forensics Tools. Digital Investigation, pp. S73-S81,2007.

[6] J. Daemen, V. Rijmen. The Design of Rijndael: AES The Advanced Encryption Standard. New York, USA: Springer-Verlag, 2002.

[7] S. Arul, M. Dash, M. Tue and N. Wilson. Hierarchical Agglomerative Clustering Using Graphics Processor with Compute Unified Device Architecture, In Proceedings of International Conference on Computer Design and Applications (ICCDA 2009), Singapore,May 2009, pp. 556-561.

[8] J. L. D. Comba, C. A. Dietrich, C. A. Pagot and C. E. Scheidegger. Computation on GPUs: From a Programmable Pipeline to an Efficient Stream Processor. Revista de Informtica Terica e Aplicada, vol. X, no. 1, 2003, pp. 41-70.

[9] D. Luebke. CUDA: Scalable Parallel Programming for High- Performance Scientific Computing. In Proceedings of 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI 2008), Paris France, May 2008, pp. 836-838.

[10] M. Garland, S. Le Grand and J. Nickolls et al. Parallel Computing Experiences with CUDA. IEEE Micro, vol. 28, no. 4, pp. 13-27, 2008.

[11] NVIDIA. CUDA [EB/OL]. (2010-01-09). http://www.nvidia.cn/object/cudahomecn.html.

[12] C. J. Thompson, S. Hahn, and M. Oskin. Using Modern Graphics Architectures for General-Purpose Computing: a Framework and Analysis. In Proceedings of the 35th Annual ACM/IEEE international Symposium on Microarchitecture (MICRO-35). Is-tanbul, Turkey. November 2002, pp.306-317.

[13] Brandon P. Luken, Ming Ouyang, and Ahmed H. Desoky"AES and DES Encryption with GPU",Computer Engineering and Computer Science Department University of Louisville Louisville, KY 40292

[14] Robert Szerwinski and Tim Guneysu"Exploiting the Power of GPUs for Asymmetric Cryptography" Horst Gortz Institute for IT Security, Ruhr University Bochum,Germany szerwinski,gueneysu@crypto.rub.de.

[15] Svetlin A. Manavski "CUDA COMPATIBLE GPU AS AN EFFICIENT HARDWARE ACCELERATOR FOR AES CRYPTOGRAPHY"2007 IEEE International Conference on Signal Processing and Communications (ICSPC 2007), 24-27 November 2007, Dubai, United Arab Emirates.

[16] Chakchai So-In, Sarayut Poolsanguan, Chartchai Poonriboon, Kanokmon Rujirakul, Comdet Phudphut "Performance Evaluation of Parallel AES Implementations over CUDAGPU Framework" Department of Computer Science, Faculty of Science, Khon Kaen University Maung, Khon Kaen, Thailand, 40002.

[17] P. Maistri, F. Masson, and R. Leveugle, Implementation of the Advanced Encryption Standard on GPUs with the NVIDIA CUDA framework, In Proceeding(s) of the IEEE Symposium On Industrial Electronics and Applications, pp. 213-217, 2011.

[18] A. D. Biagio, A. Barenghi, G. Agosta, and G. Pelosi, "Design of a Parallel AES for Graphics Hardware using the CUDA framework," In Proceeding(s) of the IEEE International Conference on Parallel and Distributed Processing, pp. 1-8, 2009.

[19] C. Mei, H. Jiang, and J. Jenness, " CUDA-based AES parallelization with fine-tuned GPU memory utilization ", In Proceeding(s) of the IEEE International Symposium on Parallel and Distributed Processing, Workshops and Ph.D. Forum, pp. 1-7, 2010.

[20] D. Le, J. Chang, X. Gou, A. Zhang, and C. Lu, "Parallel AES Algorithm for Fast Data Encryption on GPU ", In Proceeding(s) of the International Conference on Computer Engineering and Technology, pp. V6-1-6-6, 2010.

[21] R. Rivest, (1992) The MD5 Message-Digest Algorithm, RFC 1321.

[22] www.wikipedia.com/SHA-1.

[23] N Wilt. The CUDA Handbook: A Comprehensive Guide to GPU Programming. Addison-Wesley Professional, 2013.