Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 3, March 2015, pg.247 – 252

RESEARCH ARTICLE

Parallel Implementation of AES Algorithm on GPU

Sanket Wagh[1], Pawan Phad[2], Amit Surwade[3]

[1](Computer, University of Pune, India)
[2](Computer, University of Pune, India)
[3](Computer, University of Pune, India)
[1] Er.sanketwagh@gmail.com; [2] pawanphad12@gmail.com; [3] surwadeamit@gmail.com

Abstract*— With the recent improvements in cryptanalysis, more applications have started to use Advanced Encryption Standard (AES) instead of Data Encryption Standard (DES) to protect their information and security. However current implementations of AES algorithm suffer from huge CPU resource consumption and low throughput. We studied the technologies of GPU parallel computing and its optimized design for cryptography. Then, using CUDA functions for AES parallel encryption; we designed and implemented a fast data encryption system based on GPU. The tests proved that our approach accelerated the speed of AES encryption significantly.

***Keywords*— Encryption, Decryption, Cryptography, CUDA, GPU**

I. INTRODUCTION

In this paper, we explained the need of parallelism and also described the importance of using graphics processor. It introduced the basic concept of the role of GPU in performing highly complex calculation.
Network technology in recent years, has been developing faster and faster, the demand for information security has become increasingly high. Advanced Encryption Standard (AES) as a representative of one of today's cryptography has very broad prospects for development. However, affected by the CPU serial system, the CPU serial AES encryption algorithm to achieve the speed is not satisfactory; accelerated AES encryption is achieved if the use of programmable logic device such as FPGA, accompanied by. It is difficult to develop; it is difficult to increase the functionality, hardware upgrades need to modify the code and a series of cost increase in human and material resources, and limitations of the field and scope of application. Therefore, the method has become a top priority to find how to improve the speed of AES encryption algorithm. As an emerging computing resources GPU(graphics processor) has a lot of parallel processing architecture.CPU computing power compared to the development of hysteresis, GPU's floating point computing power and data bandwidth in order to achieve market expectation for the real-time requirement of computer simulation and computer games since 2003, has been far more than the CPU, ideal for large scale parallel computing has attracted scholars at home and abroad for the GPU encryption algorithm the vote for concern, in order to achieve the purpose of effective use of around surplus resources to full use. Visible using the GPU as a means to improve the AES algorithm encryption speed has a very real sense. There are many ways to choose GPU programming, CUDA (Compute Unified Device Architecture) applications is one of the most widely used, This selection of CUDA GPU programming platform in order to achieve the parallelization of the AES encryption algorithm, the main job of the following aspects: knowledge of the AES encryption algorithm mathematical knowledge a brief introduction, the encryption process is described in detail, and summarizes the operating mode of the AES encryption algorithm AES parallel encryption the model of the algorithm; The purpose of this project is to study the possibility of using alternative computing solution in cryptography, the use of a graphic processing unit in non graphical calculations.[4]

II. LITERATURE SURVEY

Cook et al. achieved AES encryption at a rate of about 1.53 Mbps throughput using OpenGL on an NVIDIA Geforce3 Ti200 (Nvidia Corp.)[3]. Although, its performance potential is only 2.3 of the Pentium IV 1.8 GHz; Intel Corporation performance because the earlier GPU architecture had an insufficient instruction set for general purpose computing. Using Direct X9 (Microsoft Corp.) Harrison et al. achieved AES encryption at 870.8 Mbps throughput on a GeForce 7900GT (Nvidia Corp.) [1]. Manavski implemented CUDA-AES achieved an astounding 8.28 Gbps throughput rate with an input size of 8MB using an NVIDIA GeForce 8800GTX (Nvidia Corp.) .Manavski reports that the thread block size engenders the fastest implementation in CUDA-AES. Performance improvement can be observed as the number of thread blocks increases. Furthermore, some effort to reduce the usage of the shared memory is reportedly necessary for better performance in CUDA-AES because the shared memory is divided into equally sized memory modules at the time of AES encryption. Although, if bank conflict occurs the access must be serialized, which causes slow performance.

Di Biagio et al. also implemented counter mode AES (AES-CTR) with CUDA, Using an NVIDIA GeForce 8800GT (Nvidia Corp.) .They achieved a 12.5 Gbps throughput rate with input size of 128 MB considering processing granularity. A solution exposing the internal parallelism of each AES round was defined by them as a fine-grained design. They proposed that four 32-bit word blocks with each thread as a fine-grain processing were dispatched to four SPs. Moreover, the coarse-grained design was defined as exploiting higher-level parallelism, which exists between independent plain text blocks. In this granularity, each thread processes each 128-bit plain text block. Nishikawa et al. also discussed granularity. They defined 16 bytes/thread when one thread processes one plain text block (whichcontains16bytes), as was true also for coarse-grained processing suggested by Di Biagioet al. [2]. Other granularities such as 4 bytes/thread and 1 byte/thread were defined in the same manner. They implemented AES-ECB encoding according to standard AES algorithm without T-box, with a GeForce GTX285 (Nvidia Corp.) Which achieved 6.25 Gbps. Later on, they proposed DES implementation on a GTX285 (Nvidia Corp.), which was intended as a brute force attack. They present a discussion of a more detailed implementation of AES referred from these respective previous works, with an attempt to improve AES performance using a GTX285 (Nvidia Corp.).

III. AES ALGORITHM

Advanced Encryption Standard (AES) is a variant of Rijndael cipher algorithm, a symmetric block cipher which translates the plain text into cipher text in blocks. This algorithm has the fixed input block size of 128 bits and the key size of 128, 192, 256 bits. The input - the array of bytes A0, A1 A15 is copied into the state array. Advanced Encryption Standard (AES) is a variant of Rijndael cipher algorithm, a symmetric block cipher which translates the plain text into cipher text in blocks. This algorithm has the fixed input block size of 128 bits and the key size of 128, 192, 256 bits. The input – the array of bytes A0, A1 A15 is copied into the state array.

The Encryption process consists of 4 phases. They are: 1. Key Expansion 2. Initial Round a. AddRoundKey 3. Middle Rounds a. Substitute Bytes b. Shift Rows c. Mix Columns d. Add Round Key 4. Final Round. a. Substitute Bytes. b. Shift Rows The phases of Decryption are (for the particular cipher text): 1. Key expansion 2. Initial round a. Add Round Key 3. Middle Round a. Inverse Shift Rows b. Inverse Substitute Bytes c. Add Round Key d. Inverse Mix Columns 4. Final Round a. Inverse Shift Rows b. Inverse Substitute Bytes c. Add Round Key [12].

1) Key Expansion:

Key expansion takes the input key of 128,192 or 256 bits and produces an expanded key for use in the subsequent stages. The expanded key's size is related to the number of rounds to be performed. For 128-bit keys, the expanded key size is 352 bits. For 192 and 256 bit keys, the expanded key size is 624 and 960 bits. It is the expanded key that is used in subsequent phases of the algorithm. During each round, a different portion of the expanded key is used in the AddRoundKey step [12].

***2**) AddRoundKey*:

During this stage, the message is combined with the state using the appropriate portion of the expanded key [12].

3) Sub Bytes:

During this stage, an 8-bit substitution, or SBox is used to modify the block. This is a non-linear transformation used to help avoid attacks based on algebraic manipulation [12].

4) Shift Rows:

This stage of the algorithm shifts cyclically shifts the bytes of the block by certain offsets. Blocks of 128 and 192 bits leave the first 32-bits alone, but shift the subsequent 32-bit rows of data by 1, 2 and 3 bytes accordingly [12].

5) Mix Columns:

This stage takes the four bytes of each column and applies a linear transformation to the data. The coefficient polynomial c(x) = 3x3+x2+x+2 (modulo x4+1) is multiplied by the column. This step, in conjunction with the Shift Rows step, provides diffusion in the original message, spreading out any non-uniform patterns. At the end of the algorithm, the original message is encrypted. To decrypt the cipher text, the algorithm is essentially run in reverse, however, if the key used for Key Expansion is not known, a brute-force attack on the cipher could take thousands of years. The decryption implementation results and encryption implementation results are similar to each other. The key expansion module is modified in the reverse order. In which first round key is treated as the last round [12].

IV. IMPLEMENTATION ON GPU

Since the objective was to fairly compare implementations of AES on GPU and CPU, we ported to GPU the open source CPU implementation. Implementing a cryptographic algorithm to run on a graphic processor is justified because, theoretically, it is cheaper to use a GPU as a co-processor to relieve the CPU from intense computing tasks, than purchasing a dedicated cryptographic co-processor that is more expensive. Adapting an algorithm, that has a computational complexity that consists in simple byte operations (AND, OR, XOR, Shifting, 32 bit adding etc.), on a video card which is designed to run complex operations and floating point operations, is the starting point of our scientific research. The source has two main entries: Encrypt () and Decrypt (). These functions take in a plain text / cipher text 128-bit source block, a 1408-bit expanded key and output an encrypted/decrypted 128-bit block. Encrypt () and Decrypt () functions take in a 128-bit key and a variable [5].

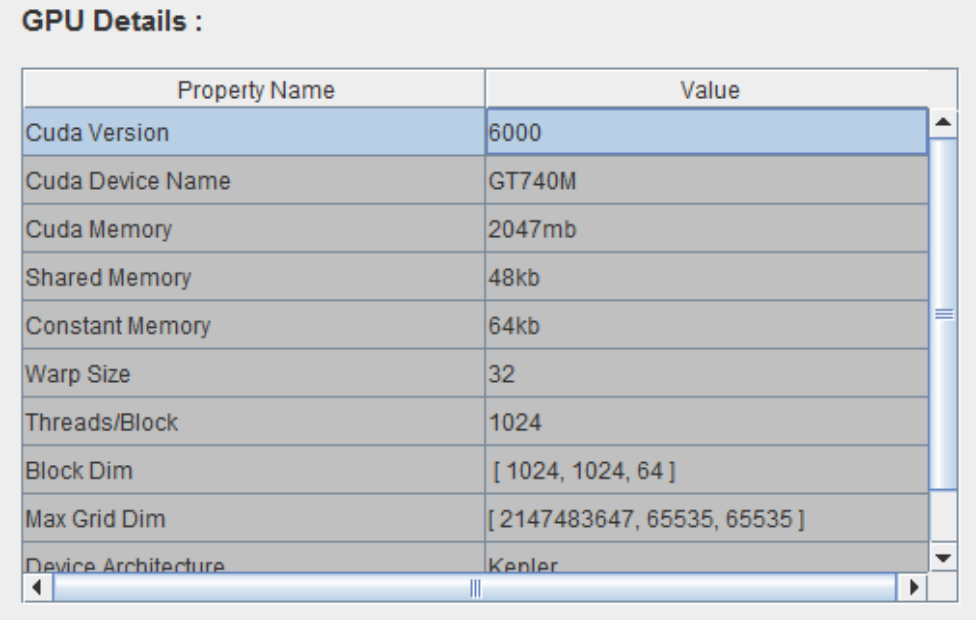

Fig.1 GPU specifications

V. PROPOSED SCHEME

The proposed scheme represents a study of the efficiency in applying modern graphics processing units in symmetric key cryptographic solutions. The scheme describes both traditional style approaches and new ones based on the recent technology trends of major hardware vendors. Also it presents an efficient implementation of the advanced encryption standard (AES) algorithm in the novel CUDA platform by Nvidia. Currently AES is the most widely adopted modern symmetric key encryption standard. Also the performance of the new fastest

GPU solution is compared with those of the sequential implementations running on an Intel Pentium processor CPU [8].

Unlike previous research in this field, the results show that GPU can perform as an efficient cryptographic accelerator. Also the developed solutions run up to 10 times faster than OpenSSL and in the same range of performance of existing hardware based implementations.

Advantages:

The proposed scheme will provide flexibility to the vendor, as it hides latency and helps maximize the GPU utilization. It will provide transparency for the programmer. Automatic thread management will be possible. Limited synchronization between threads is provided. Also dead-locks are avoided.

Limitation:

The GPU operation is unable to produce result if data is dependent on each other.

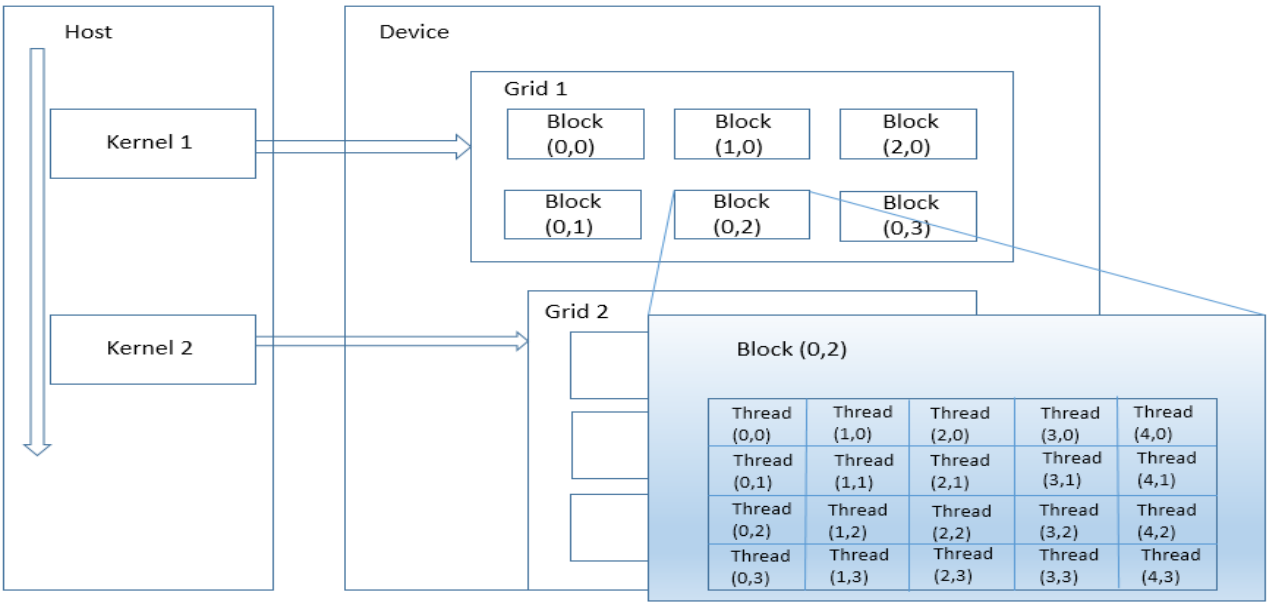

Fig.2 GPU threads [7]

VI. EXPERIMENTAL SETUP AND SPECIFICATIONS

The minimum processor speed required is 2.53GHz with Intel core 2 PC of 1 GB RAM installed on the system. Also a NVIDIA Graphics card with ubuntu operating system version 14 installed. The softwares required are Nsight Eclipse Edition with NVCC and NVIDIA visual profiler. Also java must be installed in the system with CUDA version 5.5 or above.

CPU Details

Property Name	Value
Operating System Name	Windows 7
Number of Processor	4
Ram Size	2
Total JVM Memory	16
Maximum Memory in Bytes	259522560
CPU Time for Encryption	15949ms
CPU Time for Decryption	20782ms

Fig.3 CPU specifications

Test Result of Various Input File

Dataset file Name	Size of File	Encryption On CPU(MS)	Encryption On GPU(MS)	Decryption On CPU(MS)	Decryption On GPU(MS)
AesDataset10Mb.txt	10MB	6705	983.353	13743	987.341
AesDataset20Mb.txt	20MB	12868	1916	26076	1920
AesDataset100Mb.txt	100MB	68934	10257	144102	10210
AesDataset200Mb.txt	200MB	123632	17734	251978	17765
AesDataset300Mb.txt	300MB	189748	26000	364066	27103

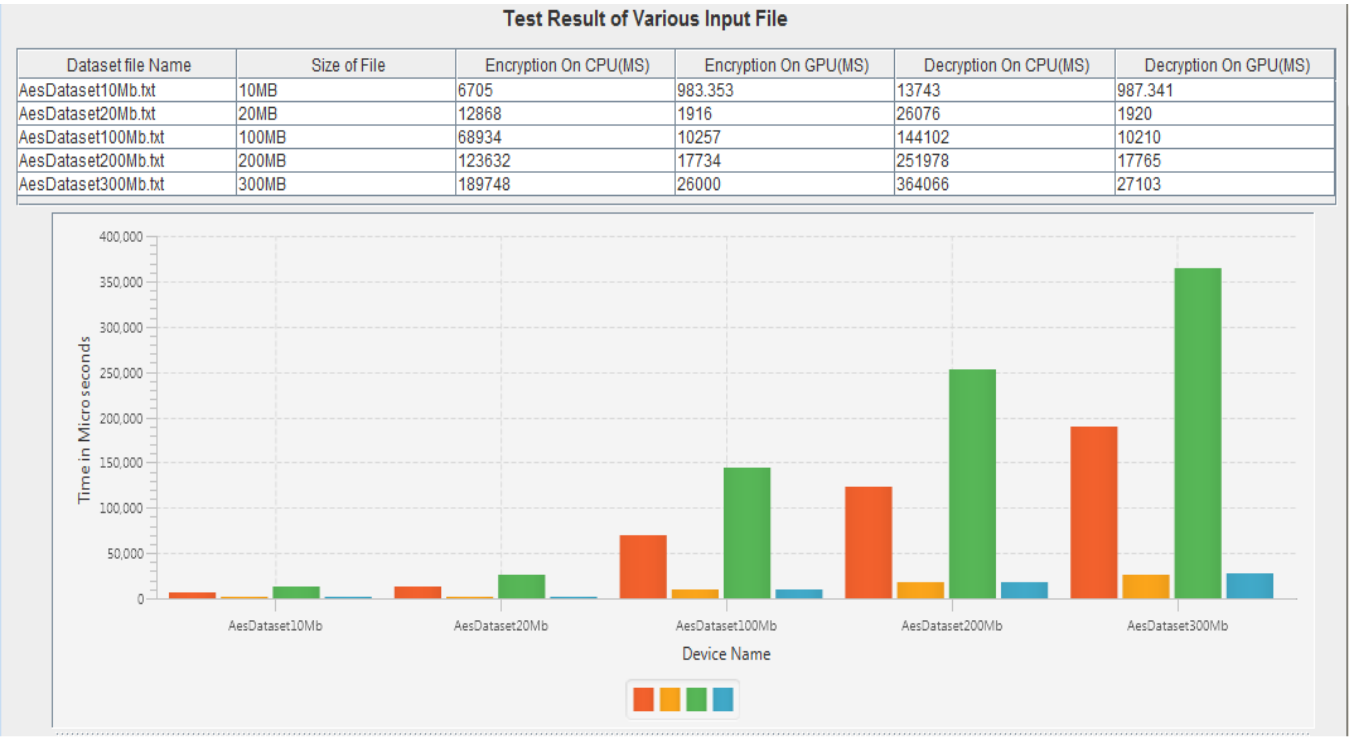

Fig.4 Graph of CPU vs. GPU Time Required

VII. CONCLUSION

Here we presented a new approach for parallel implementation of AES algorithm using modern architecture of GPU. The proposed algorithm works faster than CPU implementation of AES algorithm. In this proposed scheme we tried to use all cores of GPU to get maximum speed up.

Acknowledgements

We wish to thank the Department of computer of MET BKC IOE College Nasik for providing the materials and their support during the course of this work.

References

Journal Papers:

[1] Acceleration of AES encryption on CUDA GPU. International Journal of Networking and computing wwwijnc.org ISSN 2185-2839(print) ISSN 2185-2847 (online ion Volume 2, Number 1, pages 131-145, January 2012).

[2] Parallel AES Encryption Engines for Many-core Processor Arrays. IEEE Transactions On Computers, Vol. 62, No. 3, March 2013.

[3] 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.

[4] Mei C., Jiang H., Jenness J.: Cuda-based aes parallelization with fine-tuned gpu memory utilization. [in:] Parallel & Distributed Processing, Workshops and PhD Forum (IPDPSW), 2010 IEEE International Symposium on, pp. 1–7. IEEE, 2010.

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

[6] Cook, D., Ioannidis, J., Keromytis, A., and Luck, J., CryptoGraphics: Secret Key Cryptography Using Graphics Cards, 2005.

[7] K. Iwai, T. Kurokawa, and N. Nisikawa, "AES encryption implementation on CUDA GPU and its analysis," in *Networking and Computing (ICNC), 2010 First International Conference on*, nov. 2010, pp. 209 –214.

[8] A. Di Biagio, A. Barenghi, G. Agosta, and G. Pelosi, "Design of a parallel AES for graphics hardware using the CUDA framework", in *Proc. 23rd IEEE Int. Parallel Distrib. Proces. Symp. IPDPS 2009*, Rome, Italy, 2009, pp. 1–8.

[9] Le D., Chang J., Gou X., Zhang A., Lu C.: Parallel aes algorithm for fast data encryption on gpu. [in:] Computer Engineering and Technology (ICCET), 2010 2nd International Conference on, vol. 6, pp. V6–1. IEEE, 2010.

[10] Li C., Wu H., Chen S., Li X., Guo D.: *Efficient implementation for md5-rc4 encryption using gpu with cuda.* [in:] *Anti-counterfeiting, Security, and Identification in Communication, 2009. ASID 2009. 3rd International Conference on*, pages 167–170. IEEE, 2009.

[11] Harrison O., Waldron J.: *Aes encryption implementation and analysis on commodity graphics processing units. Cryptographic Hardware and Embedded Systems-CHES 2007*, pages 209–226, 2007.

[12] National Institute of Standards and Technology, Federal Information Processing Standard 197, The Advanced Encryption Standard (AES), 2001.

[13] D. Luebke, M. Harris, J. Kruger, T. Purcell, N. Govindaraju, I. Buck, C. Woolley, and A. Lefohn, "GPGPU: general purpose computation on graphics hardware," in *ACM SIGGRAPH 2004 Course Notes*, ser. SIGGRAPH '04. New York, NY, USA: ACM, 2004.

[14] G. Barlas, A. Hassan, and Y. Al Jundi, "An analytical approach to the design of parallel block cipher encryption/decryption: A CPU/GPU case study," in *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on*, feb.2011, pp. 247 –251.

[15] R. Detomini, R. Lobato, R. Spolon, and M. Cavenaghi, "Using GPU to exploit parallelism on cryptography," in *Information Systems and Technologies (CISTI), 2011 6th Iberian Conference on*, june 2011, pp. 1 –6.

