ANALYSIS AND IMPLEMENTATION OF PARALLEL AES ALGORITHM BASED ON T-TABLE USING CUDA ON THE MULTICORE GPU

Manjit Jaiswal[1] Rani Kumari[2] Itti Singh[3]
[1]Assistant Professor [2] B.tech Student [3] B.tech Student

[1,2,3]Dept. of C.S.E,
Guru Ghasidas Vishwavidyalaya,
Bilaspur (C.G.), Chhattisgarh, India.

Abstract: The Advanced Encryption Standard (AES) is used in security areas widely now. However, there is still a large room for further improvement of its execution efficiency. GPUs are very efficient at manipulating computer graphics and image processing, and their highly parallel structure makes them more efficient than general-purpose CPUs for algorithms where the processing of large blocks of data is done in parallel. In this paper, a design of parallel AES on the multiprocessor platform is presented. Our design is based on combining the steps of MixColumn, SubstituteBytes and AddRoundKey using the concept of T-Table. In order to enhance the throughput of AES algorithm, it is implemented parallel on multicore GPU which uses SIMD architecture using CUDA which is the framework for writing program using C++.

***Keywords*- AES, Encryption techniques, Block encryption, SIMD, GPGPU, GPU, and CUDA.**

I. INTRODUCTION

Now a days, Information Security has become very important for protecting the confidential data from unauthorized access.
Encryption techniques have paved the way for protecting the data from unauthorized users and maintaining the confidentiality of data thereby increasing data security. One of the best encryption techniques is the Advanced Encryption Standard. AES algorithm has many sources of parallelism. In this paper, a design of parallel AES on the multiprocessor platform is presented. While most of the previous designs either use pipelined parallelization or take advantage of the Mix_ Column parallelization.

Recently, Graphics Processing Units (GPUs) with multithreaded, multi-core processor and tremendous computational power, are widely used for general purpose applications such as physics, mathematics, and biology and so on [3]. Cryptography is one of the applications [12][15]. There are many implementations of cryptographic algorithms in GPU [3]. So, in order to enhance the throughput of AES algorithm, it is implemented parallel on multicore GPU which uses SIMD architecture using CUDA, a framework developed by NVIDIA which is friendly to use [1]. Parallel input of plain text is used for encrypting terabytes of data because it reduces encryption time a lot. In order to evaluate algorithm performance comparison with CUDA, we test the performance in multi-core CPUs and GPU, and compare with traditional serial programming [2] and CUDA [3].

1.1 INTRODUCTION TO AES

The Advanced Encryption Standard (AES) was published by the National Institute of Standards and Technology (NIST) in 2001.
AES is a symmetric block cipher which was introduced to replace DES based encryption from applications providing better prospects of data confidentiality and integrity compared to public-key ciphers such as RSA, the structure of AES and most symmetric ciphers is quite complex and cannot be explained as easily as many other Cryptographic algorithms [14]. The input is given as a single 128 bit block and this block is depicted as 4x4 matrix of bytes. This block is copied into the State array, which is modified at each stage of encryption or decryption. After the final stage, State is copied to an output matrix.

AES consists of 10 rounds of same operations which are:

- **SubBytes ():** replace one byte to another byte by an S-box.
- **ShiftRows ():** a simple substitute.
- **MixColumns ():** an arithmetic substitute using GF (2^8).

- AddRoundKey(): a Round Key is added to the State by a simple bitwise XOR operation.

The structure is quite simple to decipher. The cipher begins with AddRoundkey stage which are then followed by nine rounds including four stages each. The last round, i.e. the tenth round has only three stages, excluding the MixColumn stage.

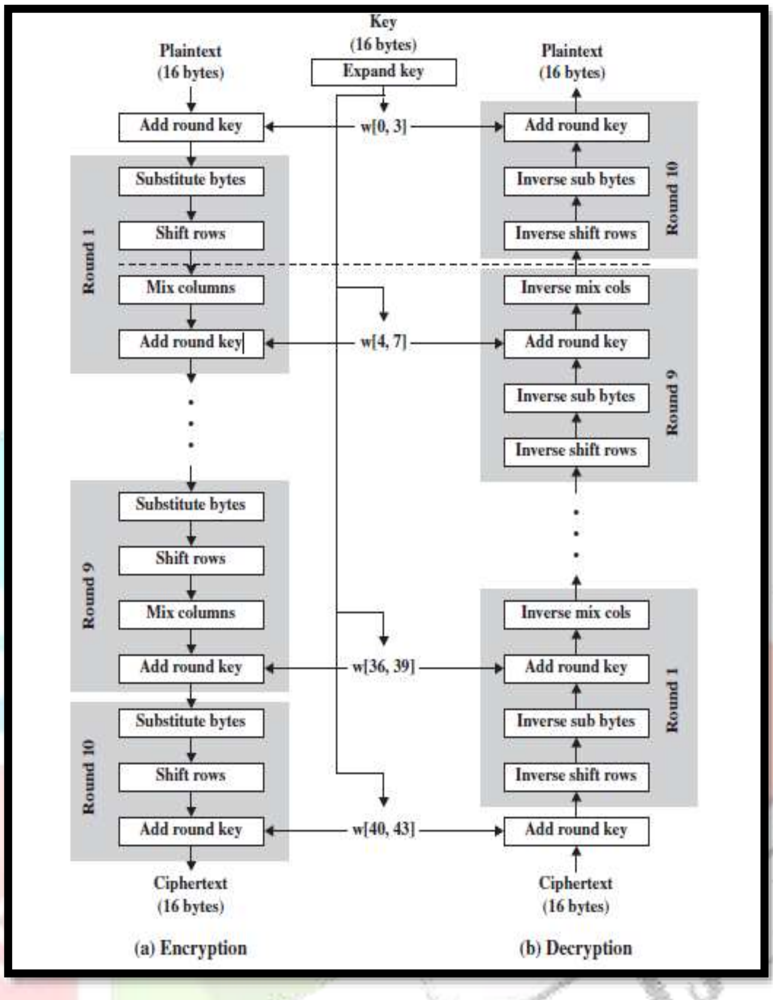

Figure1: AES Encryption and Decryption

1.2 GRAPHICS PROCESSING UNIT

A graphics processing unit is able to perform mathematical calculations more rapidly, mainly when images are rendered .It is sometimes referred to as Visual Processing Unit (VPU) specifically designed to manipulate memory for accelerating the creation of images intended for displaying as the output. GPU's are rapidly gaining the attention for fast paced and efficient implementation of algorithms from a wide range of areas. The highly parallel structure of GPU are making them indispensable and are hence quickly displacing CPU's when algorithms require the processing of large blocks of data in parallel.

GPGPU are being used for many types of parallel tasks which include ray tracing. They are often used for high throughput type combinations that showcases data-parallelism, thereby exploiting the wide vector width SIMD architecture of the GPU. One reason is that GPU programming becomes easy, because it is supported by new frameworks, such as CUDA (Computing Unified Device Architecture) offered by NVidia. The other reason is that GPU parallelism can produce powerful computing, because GPU usually has hundreds (currently thousands on high-end GPU) of processing cores and very high data bandwidth.

1.3 GPU ARCHITECTURE

The architecture of the GPU progresses in a highly different way than the architecture of CPU. To execute such a pipeline, a CPU would take a single element (or group of elements) and process the first stage in the pipeline, then the next stage, and so on. CPU divides the pipeline in accordance with time and all resources are applied in turns to each stage. GPU divides resources in accordance with the different stages therefore dividing it in space and not time. The programming over GPU follows a single instruction multiple-data (SIMD) Programming model.

Many elements are processed in parallel by the GPU using the same program for gaining more efficiency. The elements are unique, i.e. they are independent from each other and cannot communicate, in the base programming model.

1.3.1 OVERVIEW OF CUDA

NVIDIA invented Compute Unified Device Architecture (CUDA) as a parallel computing platform and programming model. CUDA takes advantage of NVIDIA GPU's parallel architecture and computing engine. CUDA API extends C programming language. Thousands of threads run in CUDA.CUDA provides a much needed simpler platform for programmers to program on GPU.CUDA routine consists of host part and device part. Thus, CUDA is much more efficient at solving complex computational problems than CPU's. The host part runs on CPU while the device part runs on GPU. Firstly, it allocates GPU memory for data. Secondly it copies the data from host memory to device memory. Thirdly, it invokes the function running on the GPU called kernel function. Kernel function will be operated by thousands of GPU threads concurrently.

1.3.2 CUDA ARCHITECTURE

CUDA architecture is a parallel computing architecture which has revolutionized the way programmers approach the programming on GPU. It delivers NVIDIA's graphic technology to general purpose GPU computation. Applications running on CUDA architecture get the additional advantage of an installed base of over one hundred million CUDA-enabled GPU's in personal computers and other bases such as notebook computers, professional workstations etc. The CUDA Architecture consists of several components, in the green boxes

- Parallel compute engines inside NVIDIA GPUs.
- OS kernel-level support for hardware initialization, configuration, etc.
- User-mode driver, which provides a device-level API for developers.

PTX instruction set architecture (ISA) for parallel computing kernels and functions

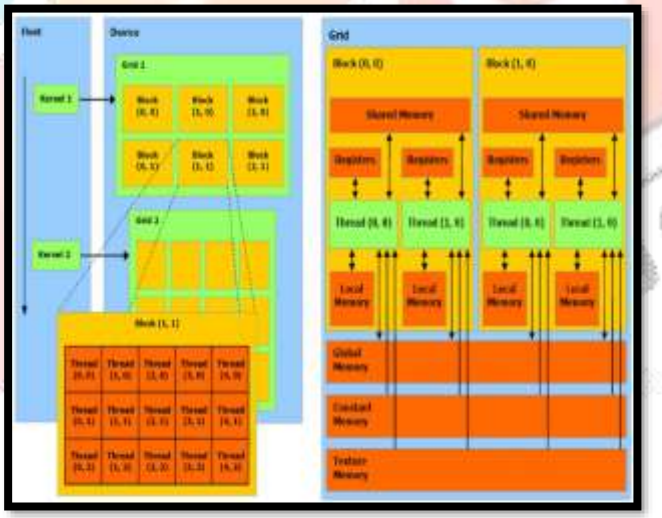

Figure2: Schematic representation of CUDA threads and memory hierarchy

II. RELATED WORK

Iwai et al. [10] implemented AES on GeForce GTX285 using CUDA. By implementing tests on different memory allocation strategies, computation granularity and structures of T-box and even combining the pipeline latency hiding technology, they found out that the best schemes for maximization the performance is to store T-boxes and round keys in local memory and plaintexts in registers and use granularity of 16 bytes/thread. Under this scheme, they achieved a 35.2 Gbps throughput rate.

In [16] it has been described that standard AES algorithm has many aspects of parallelization, it describes a design of parallel AES on the multiprocessor platform .This paper was based on combining pipelining of rounds and parallelization of MixColumn and AddRoundKey transformations .

In [11] the experiment achieves coalesced global memory access and high speed grouping on low latency registers. The highest throughput reached is 60GB/s by implementing AES on NVIDIA Tesla C2050.

In [1] Jianwei Ma et al. used the concept of T-Table with the help of CUDA. They calculated the performance related to threads per block with both small and large data. The result also presents that implementation on GPU is suitable for large data to encrypt or decrypt. The advantage of the GPU doesn't present completely, if the size is small.

III. PROPOSED WORK

The proposed algorithm consists of two levels of implementation. Generally in the sequential algorithm we have four steps in each round namely, Add Round key, Mix Column, Substitute bytes and Shift Rows. In our proposed algorithm, in the first level i.e., for round 1 to 9 we are combining the step SubstituteBytes, MixColumns and AddRoundKey and in the second level i.e., round 10 SubstituteBytes, ShiftRows and Add Round Key are performed sequentially. The level in which we are merging the steps, the concept of T-Table is used. For getting the T-table, S-box and the constant matrix (normally used in mix column) are used.

The operations in each round can be described as table1, *ai, j* represents the *state* matrix, *ki, j* represents round key matrix. Then we combine the four steps to one expression, which is illustrated as formula 2; thus, we can represent matrix's multiplication by vector's linear combination. As is showed in formula 3, we define the four tables in the front of formula 2 as T-Box which contains 256 words. Then the expression can be presented as formula 4. When one column of the *state* is operated, each byte of the column looks up one T-box and is combined into one column of a new state.

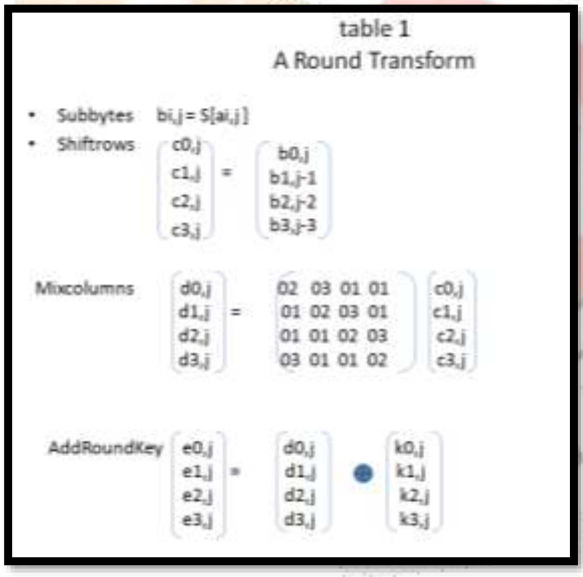
table 1

A Round Transform

- Subbytes $b_{i,j} = S[a_{i,j}]$
- Shiftrows $\begin{pmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{pmatrix} = \begin{pmatrix} b_{0,j} \\ b_{1,j-1} \\ b_{2,j-2} \\ b_{3,j-3} \end{pmatrix}$

Mixcolumns $\begin{pmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{pmatrix}$

AddRoundKey $\begin{pmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{pmatrix} = \begin{pmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{pmatrix} \oplus \begin{pmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{pmatrix}$

As a result, the implementation of AES only needs 4 T-Box lookups, 4 XORs per round per column and 4KB storage Tables; this is very suitable for paralleling on GPU which avoids a large of logical computing. This is about encryption and decryption is similar to it which we do not present here. In the next sections, we implement AES on Nvidia GPU based on fast implementation.

$$\begin{bmatrix} e_{0,j}\\ e_{1,j}\\ e_{2,j}\\ e_{3,j}\end{bmatrix} = \begin{bmatrix}02&03&01&01\\01&02&03&01\\01&01&02&03\\03&01&01&02\end{bmatrix}\begin{bmatrix}S[a_{0,j}]\\S[a_{1,j-1}]\\S[a_{2,j-2}]\\S[a_{3,j-3}]\end{bmatrix}\oplus\begin{bmatrix}k_{0,j}\\k_{1,j}\\k_{2,j}\\k_{3,j}\end{bmatrix} =$$
$$\left(\begin{bmatrix}02\\01\\01\\03\end{bmatrix}\cdot S[a_{0,j}]\right)\oplus\left(\begin{bmatrix}03\\02\\01\\01\end{bmatrix}\cdot S[a_{1,j-1}]\right)\oplus$$
$$\left(\begin{bmatrix}01\\03\\02\\01\end{bmatrix}\cdot S[a_{0,j-2}]\right)\oplus\left(\begin{bmatrix}01\\01\\03\\02\end{bmatrix}\cdot S[a_{0,j-3}]\right)$$
$$\oplus\begin{bmatrix}k_{0,j}\\k_{1,j}\\k_{2,j}\\k_{3,j}\end{bmatrix} \qquad (2)$$

$$T_0[x]=\left(\begin{bmatrix}02\\01\\01\\03\end{bmatrix}\cdot S[x]\right)\ T_1[x]=\left(\begin{bmatrix}03\\02\\01\\01\end{bmatrix}\cdot S[x]\right)$$
$$T_2[x]=\left(\begin{bmatrix}01\\03\\02\\01\end{bmatrix}\cdot S[x]\right)\ T_3[x]=\left(\begin{bmatrix}01\\01\\03\\02\end{bmatrix}\cdot S[x]\right) \qquad (3)$$

$$\begin{bmatrix}s'_{0,j}\\s'_{1,j}\\s'_{2,j}\\s'_{3,j}\end{bmatrix}=T_0[s_{0,j}]\oplus T_1[s_{1,j-1}]\oplus$$
$$T_2[s_{2,j-2}]\oplus T_3[s_{3,j-3}]\oplus\begin{bmatrix}k_{0,j}\\k_{1,j}\\k_{2,j}\\k_{3,j}\end{bmatrix} \qquad (4)$$

FIGURE3: Combined Operations in one instruction

In [1] they used the concept of T-Table for 9 Rounds and in the 10th round all steps except mix column are working sequentially. We are also using the concept of T-table but we are taking only 9 rounds containing T-Table and these are run on GPU.

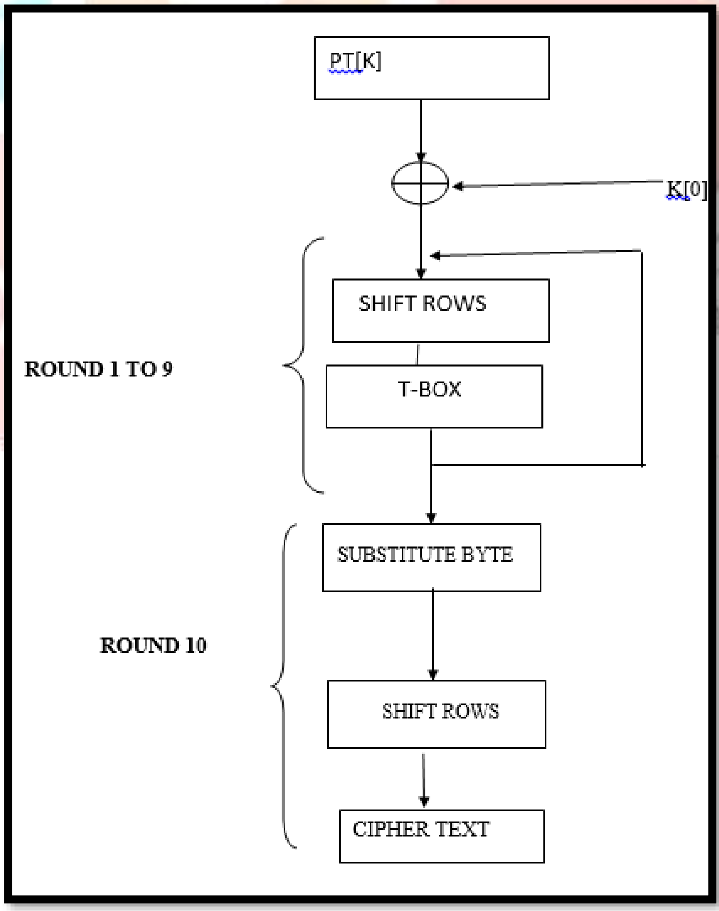

Figure 4: Flowchart of enhanced algorithm using T-Table

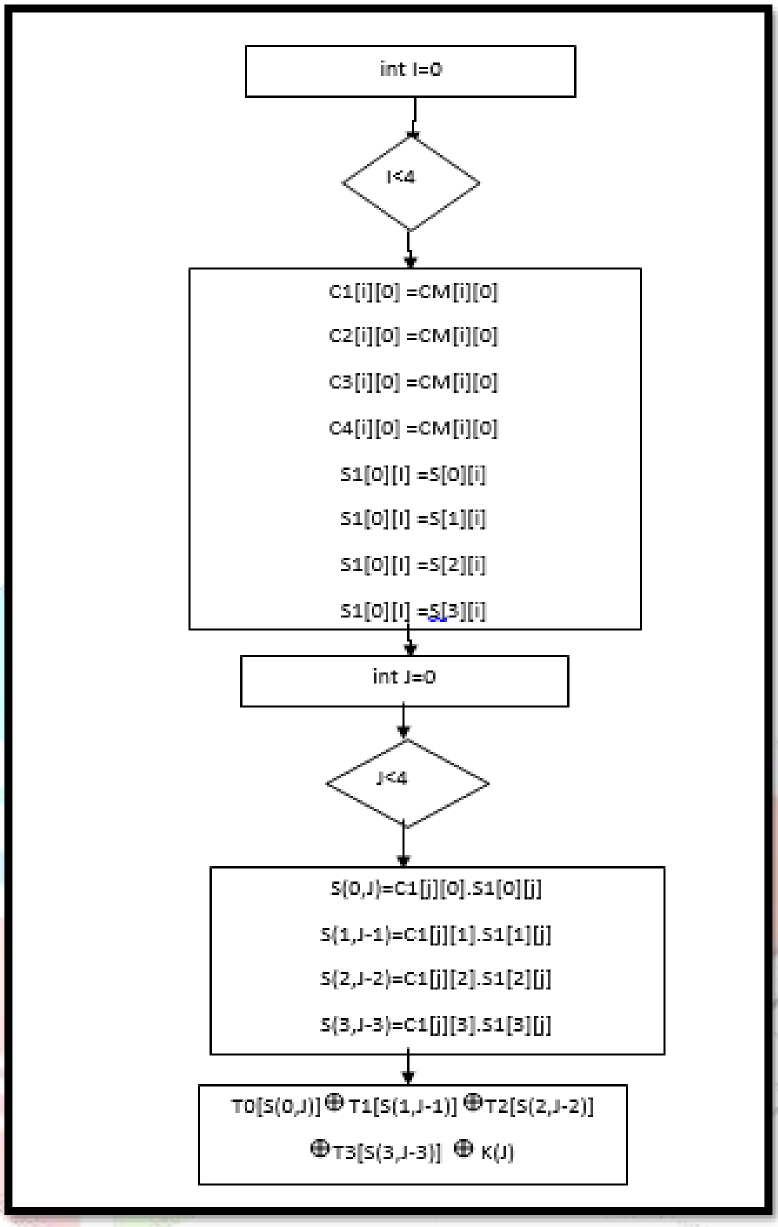

FIGURE 5: Flowchart for T-Table

3.1 EFFICIENT ALGORITHM USING T-TABLE

```
Step 1: Input plaintext of n bit in an array PT[k]
Step 2: int nB = Ceil Of(n/16), k=0, m=0;
Step 3: int i, j
Step 4: char PE[][], PT[]
Step 5: for(i=0; i<nB; i++)
Step 6: for(j=0; j<3; j++)
Step 7: for(k=0; k<3; k++)
Step 8: Initialize a $PE^{nB}$ [4][4] Matrix
Step 9: while(m<nB)
Step 10: for (i=0; i<3; i++)
Step 11: for (j=0; j<3; j++)
Step 12: $PE^{m}$[i][j] = PT[k]
```

Step 13: k++
Step 14: m++
Step 15: Allocate memories to various device arrays.
Step 16: All the Arrays are inserted as input into cores of GPU including T-boxes through cudaMemCopy in-built function with direction HostToDevice.
Step 17: Write a kernel program for AES that will be replicated to all the cores.
Step 18: Call the kernel program as AESROUND <<<8, 16, 16 >>>(pl_size);
Step 19: Through this kernel program AES is applied on all cores in parallel.
Step 20: Synchronize the data using cuda Device Synchronize function.
Step 21: The data is retrieved using CudaMemCopy function but this time direction parameter changes to DeviceToHost.
Step 22: Then display the output using main function on Host.

IV. CONCLUSION

This Algorithm will be implemented on multicore Nvidia GPU using CUDA, implementing AES algorithm on GPU (Graphics Processor Unit) for parallelization makes AES algorithm more fast and reliable. This will allow us to encrypt more data in less time. The proposed AES Algorithm improves the efficiency of AES Algorithm by using T-box and highly parallel CUDA architecture. The proposed Encryption Algorithm can produce best results in terms of Encryption time and Security when compared to the Original AES Algorithm and Enhanced AES Algorithm. It helps us to ensure that the data encrypted can withstand brute force attacks. In terms of enhancing the security it will prove to be more reliable and efficient as the complexity is increased by increasing the key size

REFERENCES

[1]Jianwei Ma, Xiaojun Chen, Rui Xu, Jinqiao Shi “Implementation and Evaluation of Different Parallel Designs of AES Using CUDA”, 2017 IEEE Second International Conference on Data Science in Cyberspace.

[2]Shivedra Shekhar, Pushkar Singh, Manjit Jaiswal “An Enhanced AES Algorithm Based on Variable S-box And 200 Bit Data Block”International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization) Vol. 4, Issue 4, April 2016.

[3]Xingliang Wang, Xiaochao Li*, Mei Zou and Jun Zhou, “AES Finalists Implementation for GPU and Multicore CPU based on OpenCL” Department of Electronic Engineering, Xiamen University, Xiamen, China, 361005.

[4]Ankit Khedlekar, Tejas Shelke, Shraddha Walhekar, Nikhita Nerkar," Smart Secure System using Parallel AES”, Department of Computer Engineering, RMD Sinhgad School of Engineering, India

[5]Vikas Kaula, Bhushan Nemade, Dr. Vinayak Bharadi, Dr. S. K. Narayan khedkar, "Next Generation Encryption using Security Enhancement Algorithms for End to End Data Transmission in 3G/4G Networks", 7th International Conference on Communication, Computing and Virtualization 2016.

[6]Ahmed Tariq Sadiq, Faisal Hadi Faisal” Modification AES algorithm based on Extended Key and Plain Text” Design for scientific renaissance ISSN: 2231-8852

[7]K. Rahimunnisa, P.Karthigaikumar.NAnitha Christy, S. Suresh Kumar, J. Jayakumar, “Parallel Sub-Pipelined Architecture for high Throughput”, Central European Journal of Computer Science, December 2013, Volume 3, Issue 4, pp 173–186

[8]James A. Muir,” A Tutorial on White-box AES”, Irdeto Canada http://www.irdeto.com

[9]Osvaldo Gervasi, Diego Russo, Flavio Vella, “The AES Implantation Based on OpenCL for Multi/many Core Architecture”, *International Conference of Computational Science and Its Applications, Fukuoka, Japan*, pp.129-134, March 2010.

[10]Keisuke Iwai, Naoki Nishikawa, Takakazu Kurokawa,” Acceleration of AES encryption on CUDA GPU”, International Journal Of Networking and Computing – www.ijnc.org, ISSN 2185-2839 (print) ISSN 2185-2847 (online), Volume 2, Number 1, pages 131–145, January.

[11]Qinjian Li, Chengwen Zhong, Kaiyong Zhao, Xinxin Mei, Xiaowen Chu,” Implementation and Analysis of AES Encryption on GPU”, Inspur-HKBU Joint Lab of Heterogeneous Computing Department of Computer Science Hong Kong Baptist University Hong Kong, CHINA.

[12]Owens, John D, Houston, Mike, Luebke, David, Green, Simon, Stone, John E, Phillips, James C, “GPU Computing”, *Proceedings of the IEEE*, vol.96, No.5, pp.879-899, May 2008.

[13]Sabbir Mahmud.” A Study on Parallel Implementation of Advanced Encryption Standard (AES), INDEPENDENT UNIVERSITY, BANGLADESH

[14]Stallings W, Cryptography and Network Security, Pearson Prentice Hall, New Delhi, 6th Impression.

[15]Debra L. Cook, John Ioannidis, Angelos D. Keromytis, Jake Luck, “CryptoGraphics: Secret Key Cryptography Using Graphics Cards”, The Cryptographers' Track at the RSA Conference, San Francisco, CA, United states, vol.3376, pp.334-350, February 2005.

[16]Ghada F.Elkabbany, Heba K.Aslan and Mohamed N.Rasslan “A Design of A Fast Parallel-Pipelined Implementation of AES: Advanced Encryption Standard” International Journal of Computer Science & Information Technology (IJCSIT) Vol 6, No 6, December 2014.