Performance Evaluation of Parallel AES Algorithm Implementing GPU

Suman Goyat, Shri Kant

Abstract: ***The Graphics Processing Unit (GPU) was initially intended for the purpose of allowing video GPUs are competent for taking extensive measure of information and playing out a similar task again and again rapidly, in contrast to CPU, which will in general skip activities everywhere. In recent, GPU are used in many multimedia tasks, such as accelerating Adobe Flash video, translating video between different formats, national security, emergency services, image recognition, virus pattern matching, sector like medicine, natural resources, financial modelling, cutting-edge scientific research and oil and gas exploration, automobiles and data mining in data centers. The very parallel structure of GPUs makes handling more powerful than universally useful CPUs for calculations where processing of large blocks of data is done in parallel. This paper sought to continue the work of other researchers in applying these general-purpose computing capabilities of GPUs to cryptographic systems as well Encryption Standard (AES) cryptosystem, which is the current standard. As we know AES on large blocks is computationally escalated and to a great extent byte-parallel. The main objective of this paper is to study and analyze the performance of GPU accelerating AES cryptosystem. The investigation demonstrates that our methodology can quicken the speed of AES encryption essentially. At last, this achieve will analyze the execution between the GPU usage and the CPU usage with the end goal to investigate the likelihood of enhancing the execution of calculations***

Index Terms: SIMD; Advanced Encryption Standard (AES); Data Encryption standard (DES); CUDA and Cryptography; Graphics Processing Unit (GPU).

I. INTRODUCTION

In this section we will introduce basics of cryptography and an overview of GPU. Further, the primary concern is on the advances of GPU parallel processing and its improved arrangement for cryptography. Usage is finished utilizing the CUDA stage. CUDA is a parallel enlisting stage and programming model made by NVIDIA and executed by the delineations getting ready units (GPUs) that they make. Brief of CUDA is referenced in the following segment. We have tested our AES algorithm encryption/decryption using CUDA framework and identified comparative performance over existing CPU. The test demonstrates that our methodology can quicken the speed of AES encryption essentially. An illustration of whole work has been concluded later.

1.1. Cryptography

In the ancient period people keep their mystery data disguised or generally shielded from outsiders. There might be one of the answers for the security issues are to encode the data so that an assailant would not have the capacity to recoup it regardless of whether he figured out how to get it. The technique which is utilized for that reason now days called cryptography. Cryptography is the investigation of scientific strategies concentrated on data security, including classification, non-revocation, information trustworthiness, and verification. It is the study of composing insider facts, which has been utilized for quite a long time to disguise data from meddlers and spies. For a considerable length of time cryptography was utilized exclusively in strategic and military circles. These days cryptography turn out to be more mainstream and utilized at the national security level. The detail execution of cryptography is usually included computationally complex calculations which are used by applications while encoding, unscrambling, and hashing data. Encryption is the technique for changing over plain content into a mystery message with the goal that the busybodies are not ready to alter the first message. Unscrambling is the turnaround of the encryption and done at the recipient end to secure unique message by changing over mystery message into plain content. Despite the presence of security conventions and usage, numerous online administrations abstain to utilize cryptographic calculations because of their poor execution, notwithstanding when utilizing cryptography would be an unmistakable acceptable position. A few usages likewise incorporate validation and check procedures. The primary prerequisite of cryptography is to have these properties.

Confidentiality: Keep information secret to anyone but the intended recipient(s).

Integrity: Ensure the information has not been corrupted or tampered with.

Validation: Corroborate the data or potentially its sender's inception.

Non-repudiation: Prevent a party from denying previous actions or agreements.

Data security is the hotly debated issue of research in the field of software engineering and innovation, and information encryption is a standout amongst the most imperative strategies for PC security. Cryptography plays a vital role in keeping information secure and it is being used from a long decade. The complete overview of the methods for cryptography has been summarized in [1]. Cryptography is basically divided into two major types symmetric key cryptography and asymmetric key cryptography with basic difference of type of key used for encryption and decryption. Symmetric key cryptography utilizes a similar key for encryption and decoding. Asymmetric key

Revised Manuscript Received on December 22, 2018.

Suman Goyat Department of Computer Science and Engineering, Sharda University, Greater Noida, U.P, India.

Dr Shri Kant, Department of Computer Science and Engineering, Sharda University, Greater Noida, U.P, India.

cryptography uses different key that may be public or private key for the encryption and decryption.

With the changes in the technology there is improvement in the methods of encryption/decryption for the better results. As per Cook, 2005 passed the pioneer symmetric encryption structure, Data Encryption Standard (DES), Advanced Encryption Standard (AES), formally called Rijndael, was chosen as a crude symmetric key cryptographic calculation, broadly utilized in different applications by US National Institute of Standards and Technology (NIST). Since another kind of encryption computation, i.e. Impelled Encryption Standard (AES), has been proposed for replacing the past encryption of Data Encryption Standard (DES) in 2001, a consistently expanding number of usages are starting to use AES as opposed to DES to guarantee their information security in the past ten years. Right now, the executions of AES depend on CPU since CPU is viewed as the registering segment in the PC framework from the customary perspective. With the quick development of data information, an ever-increasing number of utilizations require encoding information with the execution of more rapid. Bodake et al. 2015 referenced that the conventional CPU based AES usage demonstrates the poor execution and can't meet the requests of quick information encryption [2]. Consequently, how to build up another technique for superior is a testing point of research, which is intriguing an ever-increasing number of scientists in growing new methodologies for quick AES encryption.

1.2. Overview of GPU

GPU computing is the utilization of a designs preparing unit together with a CPU to perform profound learning, information gain, investigation, and building applications. This registering gives basic figure serious parts of the application to the GPU, and the remaining code keeps running on the CPU. GPU acceleration makes use of combined scheme in which GPU is included with general PC's and performs computing. From a user's perspective, it can be easily obtained that applications simply run much faster. The basic architecture of GPU is shown in the figure. The figure 1.2.1 shows CPU with few cores and GPU with many processors advanced for estimations ordinarily and over and again required for Computer Graphics like in lattice duplication, especially SIMD tasks. One can do designs preparing on a CPU - yet it likely won't create the outcome anyplace as quick as a legitimately modified GPU.

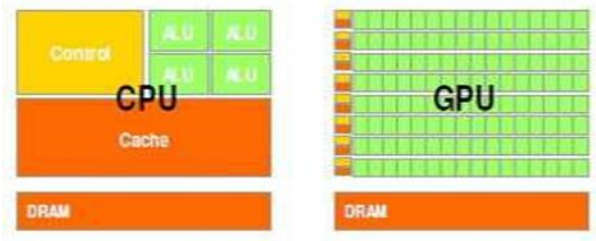

***Fig** 1.2.1 Architecture of CPU & GPU*

It is composed into 16 exceptionaly threaded Streaming Multiprocessors (SMs). Several SMs frames a building block. Each SM has 8 streaming processors (SPs), for an aggregate of 128 (16*8). Each SP has a multiply-add (MAD) unit, and an extra multiply (MUL) unit. GPU create high computational power as well as with low expenses. More transistors can be given to information calculation as opposed to information reserving and stream control as on account of CPU. On the off chance that the memory chip is customized appropriately, the high transmission capacity compensates for the substantial idleness. With numerous threads that control by high memory data transfer capacity, these days GPU present with fantastic assets for both non-illustrations handling and designs preparing in the meantime. Each GPU at present accompanies 1.5 megabytes of DRAM. These DRAMs differentiate from the structure memory DIMM DRAMs on the motherboard in that they are basically the casing support memory that is utilized for designs [3]. For designs applications, they hold superior quality video pictures, and surface data for 3D rendering as in recreations. Be that as it may, for processing, they work high data transmission off-chip store, however with more dormancy normal reserve or framework memory [3].

II. BACKGROUND

In this section we will discuss about brief of CUDA and AES algorithm.

2.1. CUDA

At November 2006, NVIDIA present CUDA, an extensively helpful figuring stage and programming model that utilization the parallel procedure engine in NVIDIA GPUs to handle various complex computational issues in a more powerful way than on a CPU (CUDA C programming guide version 6.5, 2014). CUDA (Compute Unified Device Architecture) is a structure which makes the GPGPU increasingly open and less requesting to learn for the overall public of software engineers. This is on the grounds that it expands on C and shrouds a significant number of the entangled points of interest of how the GPU functions from a CUDA [2]. With CUDA, the GPU is seen as a profoundly multithreaded processor, working a similar program (in each thread) autonomously on different information. CUDA offers various utilitarian libraries obscuring a load and complex low-level gear get to programming including a course of action of hardware bearings to help SIMD (Single Instruction Multiple Data) for compelling parallelism Thread structure of CUDA is shown in the figure 2.1.1 below:

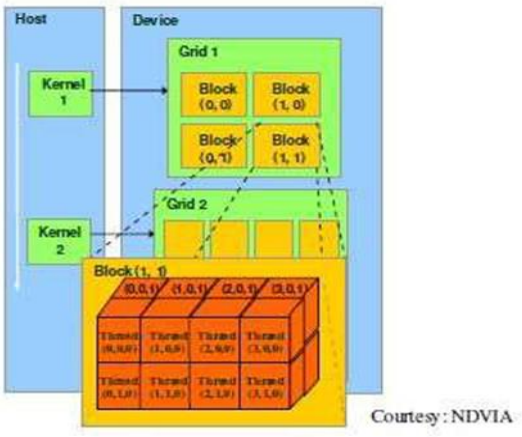

***Fig** 2.1.1 CUDA Thread Structure*

A program that runs on the GPU is called a kernel. When a kernel function is called using CUDA, it is necessary to specify how many threads will run the function. The programming model is C-like compiler upgraded to use equipment asset of GPU. CUDA gives adaptability to designers to access the execution of many threads in parallel on GPU on the off chance that a piece of a program that executes a few times autonomously can be detached, at that point, this can be modified to be executed on GPU as a free thread. When issuing a computational capacity, a kernel can run over different threads and in various thread-blocks. Threads are organized in thread blocks: groups of threads that have a common shared memory. It empowers dramatic increases in computing performance that make use of the parallel compute engine in NVIDIA GPUs to solve many complex problems in a more proficient manner than on a CPU. The framework of CUDA appears in figure 2.1.2

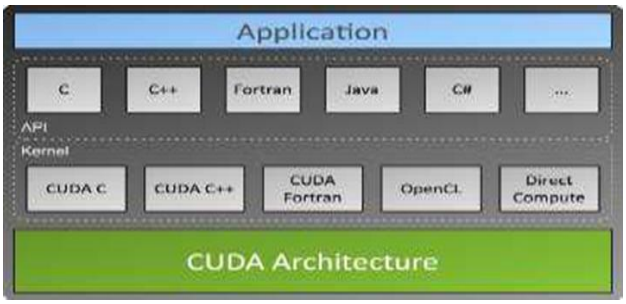

***Fig** 2.1.2 CUDA Framework*

Threads within the same block can share information through a quick constrained on-chip shared memory. Each block can communicate with another using various methodologies, e.g., constant, local, and global memory, depending on required speed, data size, etc. The architecture includes:

- Enable Parallel computing architecture and programming model
- Include a CUDA C compiler, support for OpenCL and Direct Compute
- Designed to natively support multiple computational interfaces (standard languages and APIs)

2.2. AES

Advanced Encryption Standard (AES) is a variation of Rijndael cipher algorithm, a symmetric block cipher is applied which makes an interpretation of the plaintext 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 and performs 10, 12, or 14 rounds of the cipher depending on key size.

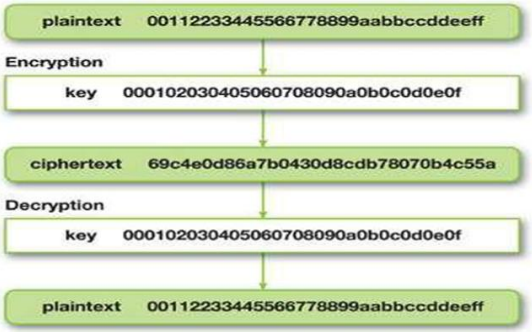

***Fig** 2.2.1 AES Cipher Operation*

In cryptography, algorithms such as AES are called product ciphers. Each round consists of several processing steps, which each consists of four main states: SubByte, ShiftRow, MixColumn, and AddRoundKey (except for the final round omitting the MixColumn step) including one that depends on the encryption key as shown in figure 2.2.1.

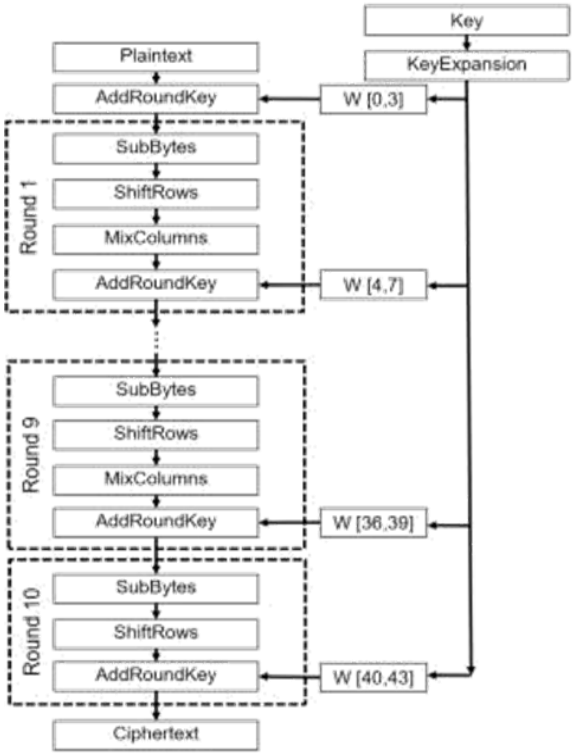

***Fig** 2.2.2 AES Encryption stage*

Each step operates on the state, at each round *r*, as follows (Figure 2.2.2 shows the overview of various AES states).

- SubByte – Every byte in 16-bytes AES block will be substituted by the byte-entry expressed in the S-Box table.

- ShiftRow – Each row in the AES block will be left-pivoted by I ($0 \leq I \leq 3$) in each line, separately.

- MixColumn – Given as a polynomial of degree below 4 with coefficient in GF(2^8), each column is processed through vector multiplication in Galois binary fields.

- AddRoundKey – The post-processing AES block will perform an arithmetic XOR operation to the r^{th} round key.

A lot of invert rounds are applied to transform the cipher text back into the original plaintext utilizing same encryption key. This type of key is a symmetric key. Other algorithms require a different key for encryption and decryption in asymmetric key cryptography. Moreover, many of these product ciphers, including AES, change the cipher key at each round. Overall algorithm steps can be sum up in the following figure 2.2.3.

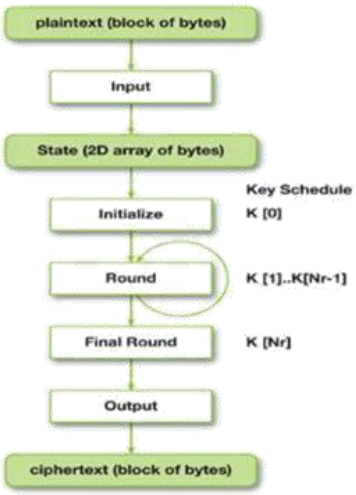

***Fig** 2.2.3 AES Algorithm*

III. LITERATURE SURVEY

Many researchers have worked in this area and one of them presented GPU features by implementing Advanced Encryption Standard (AES) (NIST 2001) encryption and decryption on the GPU [4]. Unlike previous attempts and precisely thanks to these new features, our implementation shows slight performance gains over CPU implementations. Using the remarkable properties of the GPU through CUDA has unbelievably extended the capability of various computational issues [4]. A couple of works prior done considered other execution factors for instance, [5] proposed AES usage on CUDA-GPU using an off-chip consistent (moderate) memory without operational mode thought [6]. There is a trouble to gain the genuine CUDA usage. Furthermore, some don't for all intents and purposes examine the genuine parallel calculation for execution assessment among different parallel AES usage. Some evaluated the execution of NVIDIA 8800 GT GPU of parallel AES utilization using on-chip bestowed memory to CTR [7]. Assessment of parallel AES structure usage of NVIDIA GeForce 9200M gives the talk of various memory employments [8]. [5] likewise explored on AES parallelism in CTR mode however with NVIDIA 9600 GT and 260 GTX nearly. This is the issue distinguished in AES quickening utilizing GPU system.

Some of the researcher provide solution to such problems. Notice that although several proposals have expressed numerous points of interest of mapping the full round calculation (AES obstruct) into each GPU core, these are only a fine-grained approach of parallelism optimization. On the off chance that we Consider the firmly related work, [10] analysed on parallel AES execution without operational mode giving the pseudo-code inciting open code [9] with the exception of not considering each individual AES state for advancement reason. Along these lines, with no enhancement, there is certifiably not a very much wanted point of view for AES on parallel designing. Likewise, [10] discussed the AES execution detail and give the genuine code; and while analysing into its detail, the enhancement occurred amid SubByte and MixColumn states. These are some related work we summarized here. The primary worry of our methodology is that we researched on the likelihood to accelerate the AES increasing speed over GPU with the true objective to upgrade the profitability of AES parallelism by advancing each inside stage to evade the confinement of full-parallel capacity. There is comparative performance of latest GPU technology over existing CPU environment. The combination of GPU and CPU provide heterogeneous system for the upcoming challenges over information security and other issues related to computer systems.

IV. PARALLEL AES OVER CUDA

Since the objective was reasonably think about usage of AES on GPU and CPU, we ported to GPU the open source CPU execution. This implementation is byte-oriented which is suitable for a byte-level parallelism on GPU. The source has two fundamental passages: Encrypt() and Decrypt(). These capacities take in a plaintext/ciphertext 128-piece source hinder, a 1408-piece extended key and yield a scrambled/decoded 128-piece block. At a more elevated amount, the Encrypt() and Decrypt() capacities take in a 128-piece key and a variable length message. They split the message into 128-bits blocks, suitably cushion the block when the message estimate is numerous of 128 bits, and pass the blocks to the Encrypt() and Decrypt() capacities. The AES play out the four capacities SubBytes, ShiftRows, MixColumns, and AddRoundKey. These capacities are executed parallel. We play out the Key Expansion on the CPU and pass the extended key into the GPU. Presently on this new special parallelism reinforced GPU's, System proposes to execute these sub functions as CUDA piece and dispatch these sub-work parts from parent AES bit. This typically prompts an exceptionally parallel GPU execution. The GPU threads play out the Encrypt() and Decrypt() works in parallel. Each thread tackles a subset of the data, so there are no conditions between threads. Execution showed as pursues table 4.1.

4.1 Modes of encryption

AES is a block-based encryption standard and makes an interpretation of the plaintext into ciphertext in blocks. There are distinctive modes under which the encryption can take where a few modes are inalienably more secure and some loan themselves more to parallelism.

Electronic Codebook (ECB): In this mode, each block is encoded with an indistinguishable key and there is no sequential reliance between the blocks. While this prompts broad parallelism, the extensive scale structures in the plaintext are safeguarded [9].

Figure Feedback (CFB): In this mode, the contribution to the block cipher encryption is the ciphertext from the past block. The yield of the block cipher encryption is XORed with the plaintext to create the ciphertext for the block. Similarly, as with CBC and PCB, this mode is unsatisfactory for enormous parallelism [9].

Table 4.1 AES CUDA-GPU and CPU Implementation Performance (ms)

Data (MB)	CPU	Traditional Parallel AES [9]	SubByte-MixColumn [10]	SubByte-ShiftRow	ShiftRow-MixColumn
1	779.000	38.905	39.104	39.030	**38.509**
2	1537.000	73.175	73.488	73.067	**72.514**
4	2885.333	141.498	142.098	141.285	**140.789**
8	5723.333	281.337	281.579	281.214	**279.074**
16	11370.000	556.231	558.444	556.300	**554.569**
32	22943.333	1110.919	1114.174	1110.913	**1106.311**
64	46410.000	2215.511	2222.213	2215.524	**2199.202**
128	105434.666	4434.192	4438.005	4434.401	**4396.180**

Cipher block Chaining (CBC): In this mode, the plaintext block is XOR'ed with the ciphertext created from the past block preceding entering the block cipher encryption. This dispenses with the huge scale structures in the ciphertext however the sequential reliance is unacceptable for parallelization [9].

Spreading Cipher-block Chaining (PBC): In this mode, the plaintext and the ciphertext from the past block are XOR'ed with the plaintext from the current block. Likewise, with CBC, the vast scale structures of the plaintext are dispensed with from the ciphertext, however again the sequential idea of the calculation does not fit parallelization [9].

Counter (CTR): This mode is fundamentally the same as CFB in that the yield of the block cipher encryption is XORed with the plaintext to deliver the ciphertext. It varies in that the contributions to the block cipher encryption are not reliant on the past block. Rather than affixing the outcome from the past block, a counter esteem is utilized as a contribution alongside the key [9]. This mode takes into account an abnormal state of parallelization as the contributions to a block are autonomous of alternate blocks. By utilizing a counter, the substantial scale structure that may have been available in the first plaintext is decreased. It is this mode that we utilized in our parallelization endeavors on the GPU.

V. TESTING AND EVALUATION

The main prerequisite is correctness and guaranteeing no adjustment of the usefulness of the AES calculation, both the GPU and the CPU implementation were dissected and results are gotten accurately. It should be checked whether encryption decryption did well or not. The plain text after converted into cipher text should give original plain text when decryption is applied on the cipher text. After the verification of correction, we evaluated the performance. As the motivation behind any cipher is to rapidly scramble approaching information, the execution metric we took is the general run-time of the calculation. This includes two aspects, key generation and moving data between GPU and CPU memories. The results of this comparison have been shown in the following table. First of all, the AES parameters which are being taken is shown in table 5.1

Table 5.1 AES parameters for GPU

Key Size (Bytes/bits)	16/128	24/192	32/256
Plaintext Block size (Bytes/bits)	16/128	16/128	16/128
Number of Rounds	10	12	14
Round key Size (Bytes/bits)	16/128	16/128	16/128
Expanded key Size (Bytes/bits)	176/1408	208/1664	240/1920

Results of running a random data set of AES algorithm through the encryption and decryption modules 5 times shown below. The following table 5.2 shows the total time taken for completing both encryption/decryption on CPU and GPU simultaneously.

Table 5.2 Comparison of runtime of AES on CPU and GPU

S. No	Input File size(KB)	CPU Encryption time (ms)	CPU Decrypt ion time (ms)	GPU Encryption time (ms)	GPU Decryption time (ms)
1.	23	110.000	32720.000	30.6846	34.628
2.	58	420.00	20500	30.786	34.551
3.	115	770.000	440128	32.1342	36.0249
4.	457	4010.000	6347.000	39.4578	44.113
5.	914	8870.00	10347.000	78.4082	88.131

To think about GPU and CPU executions, we partitioned the GPU run-time by the CPU run-time. Even

though this proportion is great marker of what benefits the GPU can bring, it ought to be utilized with alert when looking at results from changed productions [11].

Since the CPU usage runs sequentially, we hope to see a direct increment in runtime as the message measure increments. The parallel GPU runtime should increment directly also, be that as it may, there might be a stage like conduct as the message estimate crosses the limits of the most extreme work unit.

The message estimate starts at 16-bytes, the smallest block unit, and duplicates in size until 64MB. This enables us to see a profile of the calculation as the issue measure increments. Varieties in equipment, relative freshness of GPU and CPU chips, CPU executions, and CPU advances will all prompt changes in the GPU/CPU proportion. The outcomes were accomplished by running an irregular informational index through the encryption and decryption modules multiple times in table 5.3.

***Table** 5.3 Results*

Message Size (KB)	GPU Time (s)	CPU Time (s)	Speedup
16	0.05	n/a	n/a
64	0.06	n/a	n/a
256	0.11	n/a	n/a
512	0.14	0.01	0.1
1024	0.14	0.01	0.1
4096	0.30	0.05	0.2
8192	0.31	0.08	0.3
16384	0.32	0.17	0.5
65536	0.37	0.74	2.0
131072	0.56	1.50	2.7
262144	0.74	2.95	4.0
524288	1.14	5.86	5.1
2097152	3.23	23.50	7.3
4194304	4.88	47.09	9.6
8388608	8.29	94.07	11.3
33554432	27.35	376.54	13.8
67108864	52.86	753.32	14.3

VI. CONCLUSION

This paper gives legitimate execution of the AES acceleration over NVIDIA GPU. We researched how to execute parallel AES, particularly over CUDA-GPU, and furthermore raised a few issues amid the genuine usage. Our usage accomplishes up to approx. multiple times speedup over fundamentally the same as usage of AES on a similar CPU. This is tantamount to what different creators accomplished in their work. We additionally assessed the likelihood to use each AES organize enhancement to use AES parallelism, thus the outcomes demonstrated the execution enhancement over GPU, and particularly, a conventional CPU. Despite the fact that the execution could be altogether enhanced by considering the AES organize advancement, more examination could be performed, i.e., high accuracy of parallelism, memory the board, colossal information encryption execution, and multi-mode transmission, e.g., CBC and CTR (like ECB), and these are for our future work. We chose not to actualize AES animal power wafer. Since just a single request of extent speedup was accomplished, animal power breaking AES on GPU is as yet infeasible. In coming era cryptography will take a new environment for processing and throughput maximization.

VII. ACKNOWLEDGEMENT

I would like to say thank you to my supervisor for the assistance provided by him in my work. We affirm that we have NO alliance with or inclusion in any association or substance with the budgetary intrigue, for example, honoraria, instructive gifts, investment in speakers' dressers; participation, business, consultancies, stock possession, or other value premium; and master declaration or patent-permitting plans), or non-money related premium, (for example, individual or expert connections, affiliations, learning or convictions) in the topic or materials examined in this composition.

VIII. REFERENCES

1. Kahn, D. (1996). *The Codebreakers: The Story of Secret Writing.* New York: Scribner, NY, USA, revised edition.
2. Bodake, v. J., Bangal, D. B., & Dhatrak, K. B. (2015). PARALLELIZATION OF ENCRYPTION AND HASHING ALGORITHM USING GPU. *International Research Journal of Engineering and Technology (IRJET).*

3. P, S. (2013). ACCELERATING ENCRYPTION/DECRYPTION USING GPU'S FOR AES ALGORITHM. *International Journal of Scientific & Engineering Research*, ISSN 2229-5518.
4. Cook, D. L. (2005). CryptoGraphics: Secret Key Cryptography Using Graphics Cards. *RSA Conference, Cryptographer's Track (CT-RSA)*, (pp. 334–350).
5. P. Maistri, F. M. (2011). Implementation of the Advanced Encryption Standard on GPUs with the NVIDIA CUDA framework. *In Proceeding(s) of the IEEE Symposium On Industrial Electronics and Application*, (pp. 213-217).
6. Chakchai So-In, S. P. (n.d.). Performance Evaluation of Parallel AES Implementations over CUDAGPU Framework. *Department of Computer Science, Faculty of Science, Khon Kaen University Maung, Khon Kaen.* Thailand.
7. D. Biagio, A. B. (2009). 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).
8. C. Mei, H. J. (2010). CUDA-based AES parallelization with fine-tuned GPU memory utilization. *In Proceeding(s) of the IEEE International Symposium on Parallel & Distributed Processing, Workshops and Ph.D. Forum*, (pp. 1-7).
9. D. Le, J. C. (2010). 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).
10. M. Kipper, J. S. (2009). *Implementing AES on GPU.* Toronto: University of Toronto. Retrieved from [http://www.eecg.toronto.edu/~moshovos/CUDA08/arx/AES_ON_GPU_report.pdf
11. *CUDA C programming guide version 6.5.* NVDIA Corporation (2014).

AUTHORS PROFILE

Suman Goyat pursuing PhD published papers in the field of GPU-CPU Scheduling and explored the new trends and application of High-performance computing like in medical, GIS and Remote Sensing methods with the help of new era technology of GPU with the improvement of performance basis for higher throughput, increased parallelism and advancement with the utilization of CUDA and OpenCL platform.

Dr. Shri Kant has received his Ph. D. in applied mathematics from applied mathematics departments of institute of technology, Banaras Hindu University (BHU), Varanasi in 1981. He started his career as Technical Assistant with Joint Cipher Bureau (JCB) of Armed Forces Head Quarter (AFHQ), M/O Defense in 1981 and thereafter he moved to Scientific Analysis Group (SAG) of DRDO, M/O Defense in 1985 and served there in various capacities and again he was sent to Joint Cipher Bureau (JCB) as coordinator in 2008. He brought JCB into DRDO as its one of the laboratories in 2013 and became its first Director. He got superannuated from SAG in Dec. 2015. During his about thirty-five (35) years of service in AFHQ and DRDO he has completed many projects and was on the board of several defense programs. He is recipient of several commendation certificates from SA to RM and best scientist of the year Lab award for exhibiting the excellence in pattern recognition application to cryptology.

Currently, he is working as a Professor at Research and Technology Development Centre (RTDC), Dept. of Computer Science and Engineering of Sharda University, India and involved actively in teaching and research mainly in the area of cyber security and Machine learning.