© 2018 IJSRSET | Volume 4 | Issue 4 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section : Engineering and Technology

A Survey on AES (Advanced Encryption Standard) and RSA Encryption-Decryption in CUDA

Manisha N. Kella*[1] and Sohil Gadhiya[2]
*[1]Computer Engineering, C.U. Shah University, Wadhwan, Gujarat, India
manishakella111@gmail.com
[2]Computer Engineering, C.U. Shah University, Wadhwan, Gujarat, India
sohilgadhiya@gmail.com

ABSTRACT

CPUs and GPUs have fundamentally different design philosophies, but combining their characteristics could avail better performance and throughput. In this paper, we will study the technologies of GPU parallel computing and its optimized design for cryptography. The advent of the Compute Unified Device Architecture (CUDA) from NVIDIA Technology has shifted Graphics Processing Units (GPUs) from primarily graphics enabling devices to general-purpose stream processing devices. Cryptography is the study of techniques focused on security. Two algorithms are selected for investigation AES and RSA in CUDA. The designs of most cryptographic algorithms are such that they can benefit considerably from parallel computing, which consumer GPUs can provide inexpensively and economically. Parallelization of these security algorithms in order to distribute the complex computational part among the various cores available with the processors today, will achieve higher performance and also be more energy efficient. Hence, combination of both algorithms will help to provide better security and efficiency.

Keywords: GPU, CPU, CUDA, AES, RSA, Cryptography

I. INTRODUCTION

Parallel computing involves simultaneous use of multiple compute resources to solve a large computational problem. The performance and energy benefits of parallelism are key drivers for the growth in parallel computing. Parallelization of security algorithms in order to distribute the complex computational part among the various cores available with the processors today, will achieve higher performance and also be more energy efficient. Cryptography is the study of techniques focused on security. Typically, an implementation of cryptography is computationally heavy, leading to performance issues on general purpose system. General-purpose computing on graphics processing units (GPGPU) is the use of a graphics processing unit (GPU), which typically handles computation only for computer graphics, to perform computation in applications traditionally handled by the central processing unit (CPU).

Encryption and decryption are increasingly important. In order to protect the security of individuals, corporations and even governments, information needs to be secured against potential threats. Since AES on large blocks is computationally intensive and largely byte-parallel. Certain modes of AES are more easily parallelizable and these are ideal candidates for parallelization on GPUs. Also, using GPU resources as co-processors allows better utilization of the central

processing unit. Analysis of RSA algorithm is also discussed.

II. PARALLEL PROGRAMMING WITH CUDA

The Graphics Processing Unit (GPU) has been an integral part of today's mainstream computing systems. The modern GPU is not only a powerful graphics engine but also a highly parallel programmable processer featuring peak arithmetic. And NVIDIA provides CUDA framework which is a general purpose parallel computing platform and programming model. With CUDA, programmers can parallel many computations easily.

III. PERFORMANCE OF GPU OVER CPU

As a kind of computing device, GPU is featured of parallel computing compared with traditional CPU that is serial computing. The reason behind the discrepancy in computing mode between CPU and GPU is that GPU is specialized for compute-intensive, highly parallel computation exactly what graphics rendering is about and therefore designed such that more transistors are devoted to data processing rather than data caching and flow control.

For parallel computing, the user can define threads which run on GPU in parallel using standard instructions we are familiar with within the field of general purpose programming. The user declares the number of threads which must be run on a single SM by specifying a block size. The user also defines multiple blocks of threads by declaring a grid size. A grid of threads makes up a single kernel of work which can be sent to GPU and when finished, in its entirety, is sent back to the host and made available to the application.

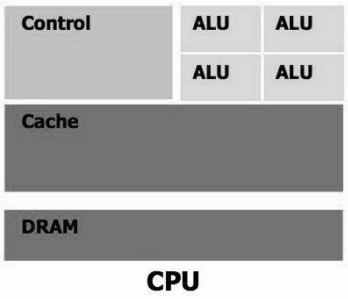

Figure 1. CPU Architecture

The processor architecture differs considerably from that of the GPU because of the CPU main scope. The CPU has a purpose of multiple tasks and multiple types of computations. The GPU has a single purpose of mathematical computation in order to display graphics.

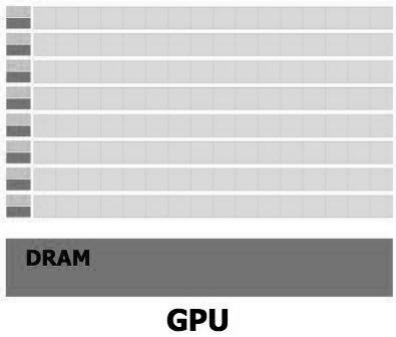

Figure 2. GPU architecture

IV. TRADITIONAL AES ALGORITHM

AES 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. From this figure, we can see that the AES-algorithm is iterative and several rounds. The input is a block of data and the initial key. Each

round operates on the intermediate result of the previous round and is a sequence of the four transformations, namely Sub Bytes, Shift Rows, Mix Columns and Add Round-Key. The intermediate result of any step is called the state. The final round is slightly different and the output after several rounds is the block of encrypted data.

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 Add Round Key step.

2. Add Round Key:

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

3. Sub Bytes:

During this stage, the block is modified by using an 8-bit substitution, or S-Box. This is a non-linear transformation used to help avoid attacks based on algebraic manipulation.

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 respectively.

5. Mix Columns:

This stage takes the four bytes of each column and applies a linear transformation to the data. The column is multiplied by the coefficient polynomial c(x) = 3x3+x2+x+2 (modulo x4+1). 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.

The decryption implementation results are similar to the encryption implementation. The key expansion module is modified in the reverse order. In which last round key is treated as the first round and decreasing order follows.

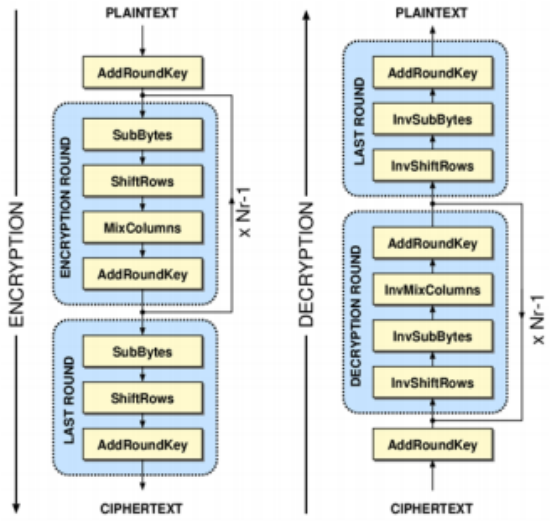

Figure 3: AES algorithm

V. PRINCIPLE OF AES PARALLELISM

In the traditional implementation of AES, the computation of data blocks is performed serially in GPU. Therefore, the efficiency and speed is poor. Figure illustrates the principle of AES parallelism.

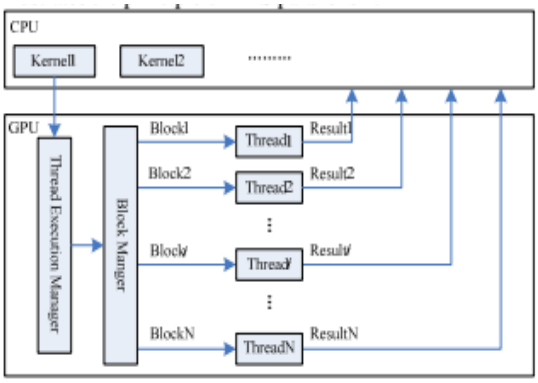

Figure 4. Principle of AES parallelism

From this figure, we can see that in GPU, it includes three components: the thread execution manager, the block manager and multiple threads. After CPU calls the kernel function executed in GPU, GPU will enable the block manger active through the thread execution manager. The block manager will then divide plaintext into multiple blocks. Then, each block will be computed in individual thread. Finally, the encrypted block will be outputted to CPU, which will be assembled into the cipher text. Since GPU allows the number of thread in parallel to be the magnitude of one hundred thousand. Therefore, the AES encryption on GPU has high efficiency of parallel computing.

VI. EXISTING RSA ALGORITHM

RSA is a algorithm for public-key cryptography developed by Ronald Rivest, Adi Shamir, and Leonard Adleman in 1977. It is suitable for both signing and encryption. Sufficiently long keys and up-to-date implementation of RSA is considered more secure to use.

RSA is an asymmetric key encryption scheme which makes use of two different keys for encryption and decryption. The public key that is known to everyone is used for encryption. The messages encrypted using the public key can only be decrypted by using private key. The key generation process of RSA algorithm is as follows:

The public key is comprised of a modulus n of specified length (the product of primes p and q), and an exponent e. The length of n is given in terms of bits, thus the term “8-bit RSA key" refers to the number of bits which make up this value. The associated private key uses the same n, and another value d such that d*e = 1 mod φ (n) where φ (n) = (p - 1)*(q - 1) [3]. For a plaintext M and cipher text C, the encryption and decryption is done as follows:

C= M^e mod n

M = C^d mod n.

VII. PARALLELIZATION OF RSA ALGORITHM

This novel parallelized implement of RSA algorithm can run parallel in two levels, thread level and the computer level, as we can see in below figure.

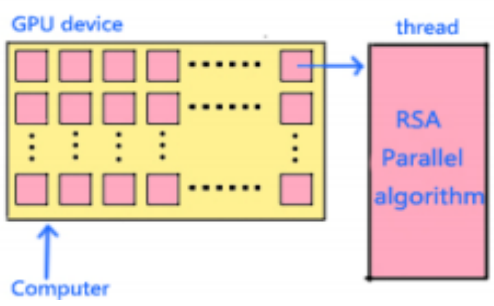

Figure 5. Parallelization of RSA

In the thread level, the parallel RSA algorithm is realized with CUDA framework. RSA algorithm divides the plaintext or the cipher text into several packets with the same length, the same encrypt or decrypt operation will be done for each packet. Here we suppose each packet is an array with the same element number, then the encrypt and the decrypt process can be done with multiple threads, each thread only need to gain the elements which are assigned to it, and run the same encrypt or decrypt function for these elements. CUDA user can get the thread and block index of the thread call it in the function running on device, so we can use the index to control the offset of the element and assign the right elements to the certain threads. In this level, the CUDA Multi-threaded programming model will dramatically enhanced the speed of RSA algorithm. CUDA user can get the thread and block index of the thread call it in the function running on device, so we can use the index to control the offset of the element and assign the right elements to the certain threads. In this level, the CUDA Multi-threaded programming model will dramatically enhance the speed of RSA algorithm. And if the plaintext needs to be encrypted or the cipher text needs to be decrypted is too large to

be process on single computer, we can use distributed file system to distribute the text to a cluster system. Each node of the cluster system will
run this parallel RSA algorithm. This is the computer level parallel. In this cluster environment, each node get a part of the plaintext or the cipher text, and then divides the part into several strings, run the RSA algorithm function in multiple threads just like the thread level does.

VIII. CONCLUSION AND FUTURE WORK

In this paper, an efficient way to enhance the encryption/decryption of AES and RSA using GPU's is proposed. This report presents the most efficient, currently known approaches in encryption and decryption of text with AES and RSA on programmable graphics processing units, achieving up to a great speed on a comparable CPU. If the amount of data is large, the encryption/decryption time required is greatly reduced, if it runs on a graphics processing environment. Future work will explore this concept and a combination of algorithms will be applied to setup a more secure environment for data storage and retrieval.

IX. REFERENCES

[1]. Jason Sanders, Edward Kandrot, "Cuda by Example" Nvidia.

[2]. J. Nickolls, I. Buck, M. Garland, and K. Skadron, Mar.2008.-Scalable Parallel Programming with CUDA.

[3]. Mahajan, Sonam, and Maninder Singh. "Analysis of RSA algorithm using GPU programming.", 2014.

[4]. National Institute of Standards and Technology(NIST), "FIPS 197: Advanced Encryption Standard(AES)," 2001.

[5]. NVIDIA Corp. NVIDIA CUDA Programming Guide 2.3, 2009.

[6]. J. Daemen, V. Rijmen, "AES Proposal: Rijndael". Original AES Submission to NIST, 1999.

[7]. GPU Computing By John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips.

[8]. Best Practice Guide – GPGPU Momme Allalen, Leibniz Supercomputing Centre Vali Codreanu, SURFsara Nevena Ilieva-Litova, NCSA Alan Gray, EPCC, The University of Edinburgh Anders Sjöström, LUNARC.

[9]. http://www.tomshardware.com/reviews/nvidia-cuda-gpu,1954-7.html

[10]. The RSA Algorithm R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and publickey cryptosystems. Communications of the ACM, 21(2):120{126, 1978.

[11]. International Journal of Innovative Research in Computer and Communication Engineering Vol. 4, Issue 4, April 2016