IDEA Algorithm on GPU using CUDA

Aditya Supugade[1] Jignesh Patil [2] Pratik Suryawanshi[3] Sanket Dige[4] Mohseen Mukaddam[5]

[1,2,3,4,5]B.Tech Student

[1,2,3,4,5]Department of Information Technology

[1,2,3,4,5]College Of Engineering Pune, India

Abstract— To provide network security, various encryption algorithms can be used. These algorithms can work along with key provided for the purpose of encryption and decryption. Longer the key size, better the security. But to encrypt and decrypt data with longer keys, more time is required for computation. The block ciphers, when subjected to parallelism offered by CUDA, shows high speedups compared to CPUs because of the absence of data dependency in the following data blocks. For optimizing block cipher the advantage of threaded execution of parallelizable code is taken. This method has advantage in the upgradation on CUDA, because a GPU has relatively large amount of memories for storing data.

Key words: GPU, Network Security

I. INTRODUCTION

Early days people used paper to record the data. With the evolution of computer this has been changed, we use computers to store data. When we started using email, need of securing information has increased. Where cryptography has become mandatory. GPU is used to accelerate encryption and decryption of the algorithm instead of multicore CPUs which is costs high. A graphics processing unit is a electronic device used to change memory such that it speeds up image processing and displaying that output on screen. IN 1980, first Graphics Processing Unit was created. Having GPU means that all graphics related work is done by GPU and there in less load on CPU for better performance. GPUs can be used for processing other than visual; their capacity to handle multiple threads simultaneously makes them more useful than CPU in algorithm having large data to process in parallel. GPU act as a co-processor with the CPU reducing the overload of CPU by enhancing the processing time.

A. Cryptography:

Cryptography is study of secure communication in presence of third parties. If hacker wants to steal data from computer while transmitting that data over network, with help of various cryptographic algorithms one can stop such act. Modern cryptography is based on mathematical theories and computer science applications. Cryptography has various domains for its applications

B. Encryption:

In cryptography, the data that has to be sent over network is encrypted i.e. plaintext is converted into cipher text and then sent over network. This process is known as encryption. It is used to protect data from cryptographic attack. Recently there are many cases of theft of data from laptop, encrypting that data make that data useless to read.

C. Decryption:

When data is reached at destination node, it is decrypted i.e. to get plaintext from cipher text, decryption is done. It is done manually or automatically, using keys. Decryption is reverse of encryption. At end of decryption, original data is obtained

D. Performance of GPU over CPU:

Graphics Processing Units are powerful, programmable and designed for highly parallel operation while a CPU executes programs serially which is where the performance issue comes. GPUs are faster and they have more advanced memory interfaces which shift around a lot more data than CPUs. The GPU computes some part of running code on CPU hence decreases load by parallel execution. The remaining part of code or application still runs on CPU like normal execution. The application becomes faster because it uses massively parallel execution ability of GPU. Since this involves use of both GPU and CPU it is also called as hybrid computing. The application runs its parallel parts on GPU, via kernels. But GPU has instant switching between CPU and GPU. Many threads execute same kernel. GPU threads are extremely lightweight when compared to CPU. GPU uses 1000s of threads for efficiency.

E. CUDA Program Structure:

CUDA program can be executed with help of CPU (host) and GPU (device) both. The NVCC i.e. Nvidia C compiler separates the two: Host code contains sequential part of code and device code contains paralleled part of code. Kernel functions are written in extended CUDA C. At beginning of code execution, host code is stated to get executed and when some kernel function is appeared during execution then device executes that part with many threads. Grid of thread gets terminated as soon as kernel part of code is finished, flow of execution is directed towards host as long as new kernel is invoked.

II. INTERNATIONAL DATA ENCRYPTION ALGORITHM

In 1991 the block cipher IDEA (International Data Encryption Algorithm) was published. Concept behind the algorithm was a replacement for the old Data Encryption Standards.

A. Explanation of Encryption Algorithm:

IDEA encrypts a 64-bit plaintext block to a 64-bit cipher text block by using its 128 bit key. It consists of Eight identical rounds with last round is half transformation round.

The different incompatible mathematical operations are mixed to operate on 16-bit blocks. The mathematical operations are:

1) Bit wise XOR
2) Addition modulo 2^{16}.

3) Multiplication modulo 2^{16} +1.

First 8 rounds does following Operations:

1) Multiplication-modulo 2^{16} +1.
2) Addition modulo 2^{16}.
3) Bit wise XOR.

Operations required in the last half output transformation round are:

1) Multiplication module 2^{16} +1.
2) Addition modulo 2^{16}.

B. Key Scheduling:

The number of keys required for IDEA algorithm:

1) First eight rounds: 6 sub-keys.
2) Last round of half transformation : 4 sub-keys.

Overall process needs 52 (8 * 6 + 4) sub-keys.

The subsequent sub-keys are produced by left shifting of 25-bits on the original key, which are used in different rounds. The process is done as follows:

1) The given 128 bit-key is divided in eight 16-bit sub-keys.
2) Left-shift key by 25 bits.
3) The resulting 128 bit-key is divided in eight 16-bit sub keys.
4) The 52 sub-keys are produced using similar process of shifting and splitting.

For ensuring that repetition does not occur in sub keys, it is shifted by 25 bits.

C. Encryption Algorithm:

1) First four 16-bit input plaintext blocks are operated with the four 16-bit keys. The 2 operations used are multiplication modulo 2^{16} +1 on two plaintext blocks and addition-modulo 2^{16} on other two input plaintext blocks.
2) In later stage, combination of all three operations i.e. Bit wise XOR, Multiplication module 2^{16} +1, Addition modulo 2^{16} is used.
3) The output of the first round of encryption is used as input to the next round of encryption.
4) Similar process is repeated to calculate the values of next round.
5) At the last encryption round four blocks are operated with remaining four keys. The operations used are multiplication-modulo 2^{16} +1 and addition-modulo $2^{\wedge 16}$ to obtain the resulting cipher text.

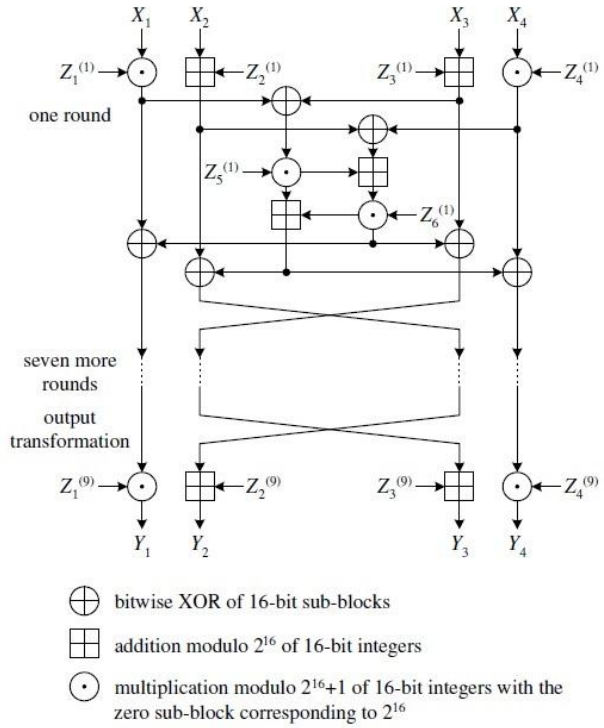

Fig. 1: Encryption Process

D. Decryption Algorithm*:*

1) Both encryption and decryption processes are similar.
2) Only change is that we use here inverse of blocks of 52 keys used in encryption.
3) Also these blocks must be used in reverse order of encryption rounds.

III. EXPLANATION OF GPU ALGORITHM AND IMPLEMENTATION

The main purpose of this paper is to implement IDEA algorithm on CPU and also on GPU for encryption and decryption of file and compare the time required for encryption and decryption for both the implementation. The algorithm is implemented serially on CPU and by using parallelism on GPU. As IDEA is block oriented symmetric cryptographic algorithm, it is implemented using bit level parallelism on GPU. It contains four important parts in algorithm Encryption-key-scheduling (), Encryption (), Decryption-key-scheduling (), Decryption (), where Encryption () and Decryption are same functions only keys provided and inverse and in reverse order. At higher level of implementation, these functions take 128-bit key and any size of data which is to be encrypted. At first the input file data is read and stored in array of unsigned short which has size in multiple of variable thread count. Array generated is fatherly divided into 64-bit blocks. The unsigned short is used because its size is 16-bit which is exactly same as size required for the processing of encryption and decryption algorithm. Also four 16-bit blocks are processed at a time, so array is divided into 64-bit blocks . Pladding is used appropriately on the blocks. It is done when block size is below 64-bit, all remaining data is copied directly without encryption. Then it is forwarded to encryption function.

The GPU thread computes Encryption-key-schedule () and Decryption-key-schedule () in parallel. Both functions mentioned above are executed using instruction level parallelism. In Encryption-key-schedule () all the six keys required for one round are calculated in parallel using six threads. And for cyclic left shift each bit is shifted by individual thread, so total of 128 threads are required for calculation of that operation. The implementation details are mentioned below:

1) CPU Implementation:

```
for (j = 25; j < 128; j++)
        bits2[i++] = bits1[j];
for (j = 0; j < 25; j++)
        bits2[i++] = bits1[j];
```

1) GPU Implementation:

```
if( tid < 103)
        bits2[tid] = bits1[tid + 25];
else if(tid >= 103 && tid <128)
        bits2[tid] = bits1[tid - 103];
```

And call to this function is given as:
IDEA_encryption_key_schedule<<<1,128>>>(dev_key, dev_K);
Here, dev_key – Input key which is allocated in device memory as follows:
cudaMemcpy(dev_key, key, 8*(sizeof(short)), cudaMemcpyHostToDevice);
dev_K – Array of separate 52 keys required for encryption.

In Decryption-key-schedule () inverse of all the keys are calculated and stored in inverse order. For this purpose, each individual thread is used to process one key, so total of 52 threads are used.

GPU threads compute Encryption () function in parallel using Data parallelism and thread parallelism. The parallel programming model used are SIMD (Single Instruction Multiple Data) and SIMT (Single Instruction Multiple Threads). The threads are independent of each other as they are operating on different blocks. Because of it high level of parallelism implemented on GPU.

Initially file data is read and stored into unsigned short values depending upon value on each byte. Big array of unsigned short values is produced which is used as a plaintext for this algorithm. This array is passed to encryption function. This large array is divided into 64-bit array which is encrypted using one thread. And large number of such threads are used simultaneously. The call to encryption function is done using variable thread count.

1) IDEA_encryption<<<1,tc>>>(dev_X, dev_Y, dev_K);
2) Here, tc - thread count.
3) dev_X – plain text.
4) dev_Y – cipher text.
5) dev_K – Key.

Another optimization is loading 64-bit array into constant GPU memory. Because these arrays are common part in all threads. Thread memory uses cache-back mechanism. Which has capacity of accommodating all data. It provides penalty for accessing data again which loaded in cache memory. As GPU runs many threads concurrently which is not possible in CPU. Because of this reason CPU provide benefit for low message sizes, as CPU not transfers data from one memory to main memory. As we are not implementing CPUs threads in parallel fashion.

IV. RESULT

As project objective is to compare CPU and GPU timings for encryption and decryption of different file,it is checked by running fixed size files like images, audio and videos as a input to the program for encryption and decryption. The output performance is not affected by data. The correctness of system is checked by comparing output file with input file. Time required for whole process does not include time required for verification.

At the start phase, the files of small sizes are checked and observed and then file size is increased. Because of this the behavior of program for increasing file sizes became clear. In these observations It has been seen that CPU execution time increases linearly as file size increases. Also GPU time increases linearly.
The performance analysis was run on a Ubuntu 64 bit machine using two configurations as follows:

1) Intel core i5 processor at the speed of 2.5 GHz with 4GB of RAM available. The GPU used was NVIDIA GE force 630M with 1GB graphics card memory.
2) Intel XEON processor at the speed of 3.0 GHz with 8GB of RAM available. The GPU used was NVIDIA GTX 870M with 8GB graphics card memory.

All implementations were compiled in Ubuntu 12.04. The CUDA version used was release 6.5. The computation time was taken using built in timer utility. Only the key scheduling and encryption processes were considered to be part of the execution, as well as any memory operations between the device and the host.

The results for file encryption using IDEA are located in Table. All time values are in seconds. The CPU is faster for the smaller file sizes. However, around the file size of 1MB, for 1st configuration and 500KB, for 2nd configuration the GPU implementations catch and surpass the CPU, where both CPU and GPU implementation takes same time. The GPU completes the process for the 100MB file size at around 6.83 times faster for 1st configuration and 3.9 times faster for 2nd configuration than the CPU timings.

File size	Timings		Speed up
	CPU	GPU	
1 KB	0.003	0.262	0.1145
10 KB	0.007	0.221	0.0316
100 KB	0.032	0.214	0.1495
1 MB	0.241	0.244	0.9877
10 MB	2.195	0.526	4.173
100 MB	21.975	3.215	6.835

Table 1: CPU and GPU Timings (In Seconds) For Intel Core I5 Processor and NVIDIA GE Force 630M Graphics Card And Respective Speedup

Fig.2: shows CPU and GPU timings for different file sizes for above configuration.

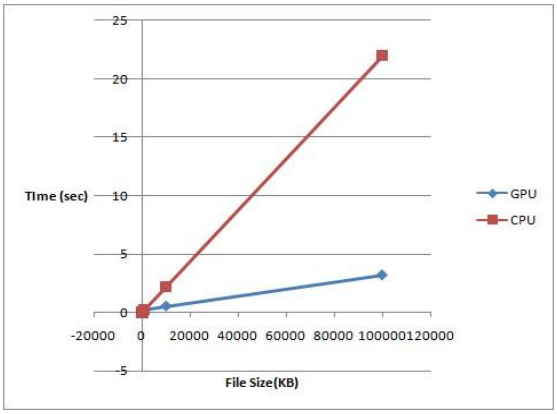

Fig. 2: CPU and GPU Timings

File size	Timings		Speed up
	CPU	GPU	
1 KB	0.002	0.060	0.033333
10 KB	0.006	0.063	0.095238
100 KB	0.032	0.066	0.484848
500 KB	0.098	0.083	1.180723
1 MB	0.190	0.104	1.826923
5 MB	0.893	0.264	3.382576
10 MB	1.754	0.476	3.684874
100 MB	17.741	4.542	3.905989

Table 2: CPU and GPU Timings (In Seconds) For Intel XEON Processor and NVIDIA GTX 870M Graphics Card And Respective Speedup

Fig.3: shows CPU and GPU timings for different file sizes for above configuration

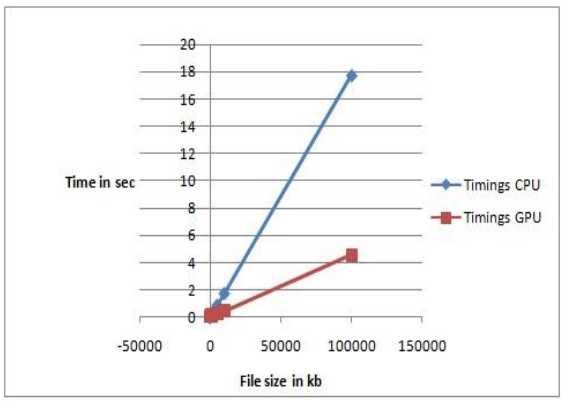

Fig. 3: CPU and GPU Timings

V. Conclusion

We made study of different encryption algorithm and used IDEA algorithm because till 2014 it is not possible to attack it with any attacks like Brute Force attack having taken care of weak keys. Also compared to other algorithms this is very effective and efficient. Parallel implementation is possible for this algorithm in efficient way.

We made full use of computing power of GPU. For processing image or video files GPUs are most powerful tool. Now a days GPUs have reduced computation time by much margin that it is used in almost all the fields.

VI. Acknowledgment

We are thankful to our guide Dr. Vandana Inamdar, professor, Department Of Computer Engineering and Information Technology, for her constant help and support from starting of paper work till the end. Because of her great guidance, We were able to make this work successful. We express our gratitude towards our Head Of Department Dr. Jibi Abraham for all necessary co-ordination in the accomplishment of project. We would like thank to our faculty and friends who constantly guiding us throughout the project work.

It would not have been possible to successfully complete the paper work without support of all above people.

References

[1] O.Y.H. cheung, K.H. Tsoi, P.H.W. Leong and M.P. Leong, "Trade of in serial and parallel implementation of International data encryption algorithm (IDEA)".
[2] Brandon P. Luken, Ming Ouyang, and Ahmed H. Desoky,., "AES and DES Encryption with GPU Computer".
[3] Sonam Mahajan and Manindar Singh, "Analysis of RSA algorithm using GPU programming".
[4] Vladimir Beletskyy, Dariusz Burak, "Parallelization of the IDEA Algorithm".
[5] Lukasz Swierczewski, "3DES ECB Optimized for massively parallel CUDA GPU architecture".
[6] Justin Naoki Nishikawa, Keisuke Iwai, and Takakazu Kurokawa, "High-Performance Symmetric Block Ciphers on Multicore CPU and GPUs" International Journal of Networking and Computing ISSN 2185-2839 (print) ISSN 2185-2847 (online) Volume 2, Number 2, pages 251–268, July 2012.
[7] Sandipan Basu, "International Data Encryption Algorithm (IDEA) – A typical illustartion", volume 2, number 7, July 2011, Journal of Global Research in Computer Science.

