IJAER/ Aug-Sep 2012 /Volume-1/Issue-2 ISSN: 2278-9677

International Journal of Arts & Education Research

ANALYTICAL PRACTICE FOR MANAGING AND SECURING DATA IN SIMULTANEOUS PARALLEL COMPUTATION ENVIRONMENT

Kapil Dev

Research Scholar

CMJ University Shillong

Dr. Amal Kumar Deka

Professor

Guwahati University Assam

Abstract

A large number of constraint-based devices are connected with the internet in developing computing environments such as the internet of things (IoT) or smart device networking. The devices automatically communicate with one another by way of the linked network, which in turn provides us with new experiences. It is extremely necessary to secure the security of connected end nodes in order to make optimal use of the capabilities offered by the Internet of Things (IoT). If the security of even one of the nodes is breached, the operation as a whole will suffer significantly. Due to the limited resources available, however, implementing suitable cryptographic functions on the device is a very challenging task. The purpose of this research is to offer a way for injecting the high performance security algorithm into data analytics that are carried out using IOT-based devices. With parallel computing devices and algorithms, the efficiency of data analysis security mechanisms will be significantly improved. AES is a symmetric encryption method that works efficiently for both hardware and software applications. The performance of data analytics in IoT-based systems can be increased by running the AES algorithm in parallel. The performance of several different Intel-based multi-core processing architectures is evaluated with this technology, and the results show a significant increase.

Keywords: *security, parallel computation*

Introduction

The analysis of data stored in networked databases has become more important in the modern day. During the process of transferring and analysing the data, these network databases must to be protected. Consequently, the provision of an efficient security method to the sensitive data should be the key focus. To address the scenario effectively, you should go with the option of using the AES algorithm. Since this study makes use of three main technologies, let's begin by determining the reasons why these technologies are necessary.[1]

Why AES algorithm?

The advanced encryption standard, often known as the AES algorithm, is now the symmetric encryption method that is most likely to be encountered in the modern day. It is also the most widely used method. It is discovered at a rate that is at least six times quicker than the triple data encryption norm (DES). It was necessary to find an alternative to DES since the key size it used was insufficient. The increasing processing capability led many people to believe that it was defenceless against a thorough key search assault. Triple DES was developed with the intention of addressing this drawback; nonetheless, it was found to be moderate.

The components of AES are created as per the following points:

- 128-piece information, 128/192/256-piece keys
- More grounded and quicker than triple-DES
- give full determination and plan points of interest
- Programming implementable in C and Java is easy.

The operation of AES involves an iterative, and its success is contingent on a "substitution-permutation network." It is composed of a series of interconnected processes, some of which entail switching out inputs for certain outputs (known as replacements), and others of which include moving bits about in different configurations (stages). AES performs each and every one of its calculations on bytes rather than bits, which is an odd choice. As a consequence of this, AES considers the plaintext block's 128 bits as if they were 16 bytes. For the purpose of working with it as a framework, these 16 bytes are organised into four segments and four columns. In contrast to DES, the number of rounds in AES is not fixed; rather, it varies depending on the length of the key being used. Ten rounds are utilised by AES for 128-piece keys, twelve rounds are utilised for 192-piece keys, and fourteen rounds are utilised for 256-piece keys. Due to the fact that it is secured using many rounds and keys, the initial AES key is used to calculate each of these round keys, and each of these round keys uses an alternative 128-piece round key. As a result, there is an absolute requirement to make the AES algorithm scalable. Because it uses many rounds to encrypt the data, it takes a longer amount of time to encrypt huge amounts of data. It is possible to improve the performance of the AES algorithm by decreasing the amount of time it takes for the algorithm to respond. In addition to providing effective data analysis, MATLAB also includes a toolbox for parallel computing, which may be used at any time.[2]

The implementation of the parallel AES algorithm will make use of the toolbox's fundamental building blocks. Therefore, using a combination of MATLAB's parallel AES algorithm and its data analytics algorithm will enable safe data processing that is also highly efficient. The ability to respond in a timely manner is the most important factor in the success of the internet of things (IoT), and the addition of data security to the processing of sensitive data will lead to a positive societal effect. This strategy will result in an improvement in the performance of the AES algorithm, which will allow for faster processing times. Performance may vary depending on the type of multi-core processing that is utilised in the constructions (such as GPU or FPGA, for example). Consequently, the scalability of the data analytics algorithm will not suffer as a result of the combination of the security algorithm and the data analytics algorithm. Due to the sensitive nature of the data analytics algorithm, it is essential that it be protected. During this process of analysing this data, many fascinating patterns will emerge, and important choices will need to be made. Therefore, the combination of the security algorithm and the data analytics algorithm will give protection for a wide variety of patterns and judgments. When doing an analysis of medical data, such as determining the association between blood cells and diseases such as cancer and HIV, for instance, we need to exercise extreme caution when transmitting sensitive data across a connected media. The Advanced Encryption Standard (AES) technique will be used to encrypt sensitive data used in transactions.

Why GPU?

The graphical processing unit, or GPU, is initially developed for the purpose of managing and processing multimedia programmes in addition to other applications that are data heavy. However, due to the

effectiveness of the method, it is currently utilised for the solution of standard numerical computing issues. It is equipped with tens of thousands of concurrent threads and hundreds of processors at the same time. It uses data level parallelism, which implies that data is partitioned and distributed among numerous processors, and those processors will handle multiple data at the same time. Due to the fact that it will have 100s of ALU to handle numerical computations, it will do numerical computations extremely well, making it incredibly efficient.[3]

Parallel computing toolbox

It is a collection of tools for resolving data-intensive problems with the help of multi-core CPUs and other processing architectures such as GPUs and FPGAs. This toolkit can accelerate GPU to address massive data issues even without utilising the CUDA framework, which is CUDA's primary strength in the data-intensive computing space. With the help of this toolbox, we are able to generate workers on the local level to make use of the various cores included within the computer. Libraries have the ability to speed up the several cores that are included in a parallel pool. If the amount of data is very large, it should be saved in a data store so that it may be retrieved quickly and used in parallel programming. For the GPU to accelerate calculations, the data must first be saved in the gpuarray (local memory of GPU). The calculations will be done on the GPU, and the results will be transferred from the GPU to the local memory of the CPU using library functions (gather, Parfeval, Mapreduce, Parcluster, Parpool and Gcp). The remaining sections of the study are organised as follows: creation of IOT, development of algorithm, experimental analysis, and related studies done in this field.

Constructing IoT computational model

The Internet of Things (IoT) is the inter-networking of physical nodes, vehicles (also referred to as 'associated nodes' and'smart devices'), buildings, and other things that are equipped with hardware, programming, sensors, actuators, and system networks that enable these items to collect and exchange information.[4] There are still various misconceptions and facts regarding the Internet of Things, such as power concerns, performance challenges, and problems associated with decision-making systems. Figure 1 contains a summary of the most important contributing elements to these problems. In most cases, the amount of delay that occurs is used as the performance metric. However, connecting a device to the internet does not automatically qualify as Internet of Things (IoT), because there are three essential components that are included in virtually all applications that use IoT.

1. 1 Computing part: Computing is done at one point and moved to connectivity part and our proposed AES algorithm works in this part. In this part, repeatability and speed of task is considered as prime factor.
2. Connectivity part: creates a network to pass information between the connected nodes.
3. Decision-making systems: When values and information come from various nodes and it should be reconstructed or recommendations should be made on the incoming values. This power can be achieved through the appropriate use of reasoning and recommendations.

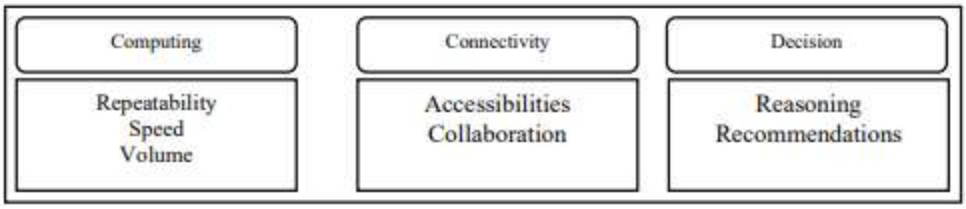

Figure 1 Requirements for IoT-based applications

Security mechanism through IOT

The objective of the time constraint-based secured key generation approach is to provide an efficient means of managing the keys that are necessary for a secure connection. The objective of the process known as "local key generation" is to produce a key simultaneously on both the transmitting device and the receiving device, which are known as "communicating devices." Figure 2 provides an overview of the procedure, which states that, in order to increase the overall degree of data security during transmission, only a selected key will be transmitted back and forth. In addition, a key change will be scheduled either at the time of transmission or in response to a user's request. As shown in Figure 1.2, the production of the key is a task that is individually carried out by each communication party. The sole limitation imposed by this scenario is that both parties must generate keys utilising the same time structure. Because reply assaults that are based on legitimate messages that were delivered using keys created in previous time periods are ignored, the strength of the validity of the keys is restricted to a time interval. This helps to keep the validity of the keys as strong as possible. By capitalising on such aspects, we have demonstrated that one of the primary benefits of the time-based approach to secure key generation is that it eliminates the requirement for a server in the management of secure keys and the production of security constructs in the manner described below.[5]

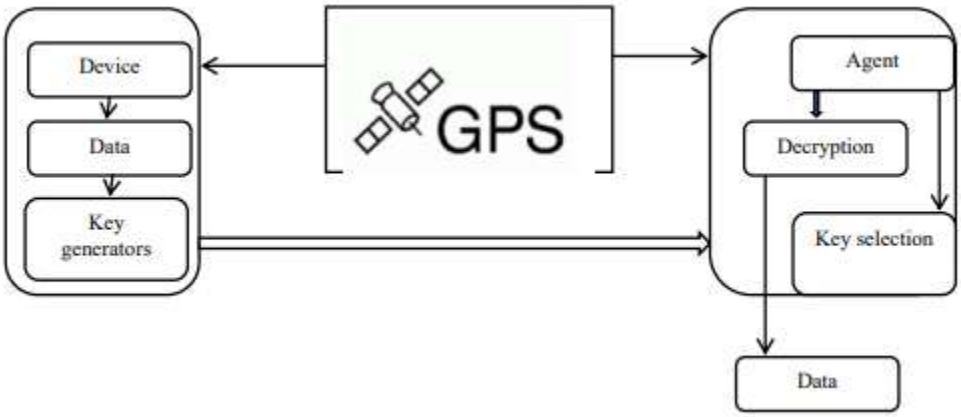

Figure 2 Constraint-based encryption for IoT

Plain text

1. Generate a time stamp using the GPS data that you have received. 2 Generate the ID using the plain text (PID). Calculations for hash 3 should be carried out using the parallel AES technique, which is covered in section 4.[6]

Encrypted text

2. Create an encryption ID (EID) to improve the authentication Payload is used convey information to associated application running on remote servers
3. When the agent receives the ciphered message, it will de-cipher it by generating the appropriate decryption key using the attached timestamp. If the time constraint difference between the current and received is more than the threshold level the message is discarded.

Security protocols to be used with IoT network

We are concentrating on improving efficiency by utilising parallel algorithms. If secrecy is required, a block cypher such as AES can be employed, and SHA can be used if integrity is required. The AES strategy has been utilised in this method.

Parallel AES algorithm

Plaintext for the AES method has traditionally been said to be 128 bits. And the key's size might range anywhere from 128 bits to 256 bits or even 192 bits. If only one core of a multi-core machine is executing this encryption, the other cores will be idle during this time. Plaintext is given into each core for processing in order to prevent the cores from being properly utilised to find a solution to the problem. The scalability of the technique will be improved if encryption is done on each core individually. The T text file that is produced by Internet of Things devices is segmented into blocks of 128 bits. Each block of 128 bits is sent into its own independent core, which then carries out the encryption. Following encryption, a cypher text will be produced from each core. If we combine those cipher text, we can have the cipher text for whole plaintext file.

Data level parallelism

In the data level parallelism approach, the incoming data is initially chunked into 128 bit blocks before being distributed among parallel computing nodes. Every node will do the identical operation, but on a distinct set of data. Data taken as input this data is obtained from a text file that is composed of data generated by the device and varies in terms of both its size and its content. The data is divided into blocks of 128 bits each.

Table 1 Sample input data

Sample input text data
Key, Na, K, WBC, RBC, Hgb, Hct
Patient1, 143, 4, 10.7, 6, 13.6, 45.7
Patient2, 150, 6.5, 8.7, 5.2, 20, 57
Patient3, 157, 4.3, 9.2, 4.4, 14, 42.6
Patient4, 141, 3.9, 9.9, 4.3, 17.1, 39.6
Patient5, 142, 3.7, 2.8, 6.5, 15.4, 44.6

• Comparisons – We compared

1. 1 Execution time of Intel multi-core processors
2. 2 GPU-based computing with CPU-based computing against speedup.

Algorithm design

With respect to Figure 4

Algorithm 1 Serial algorithm

BEGIN
Setup segment mechanism, variables and constant
Read the Plain text
Initialise variables and arrays

FOR every round DO

Step 1 Calculate the set of round keys from the time key.

Step 2 Initiate the state array with the blocks of data from plain text.

Step 3 Add the first round key to the state array.

Step 4 Iterate nine rounds of state manipulation.

Step 5 Finish the tenth and final round of manipulation.

Step 6 Copy the content of final state as encrypted data

The following instructions will show you how to parallelize a serial process. Let input segment A contains m × n data blocks.

Where

m no of rows

n no of columns.

$$\sum_{i=1}^{m}\sum_{j=1}^{n} A_i(j+n) = A_{ij} \quad (1)$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij} = A_{(i+1)j} \quad (2)$$

$$\sum_{i=1}^{m} A_{i1} = A_{i+1} + i \quad (3)$$

Steps 1, 2, 3 should be iterated in a parallel computing tool with the following algorithm

Algorithm 2 Parallel algorithm

```
BEGIN
    Setup segment mechanism, variables and constant
    Read the Plain text
    Initialise variables and arrays
MASTER processor seeds the SLAVE processors
MASTER processor sends the required parameters to SLAVE processors
FORALL processors simultaneously DO
FOR every round DO
    Step 1:  Substitute bytes in array
    Step 2:  Shift rows
    Step 3:  Mix columns
    Step 4:  Add round key
END
END
```

Process description

1 Splitting the plaintext:

The input will be in the form of plaintext, which will be delivered as a string that consists of 0s and 1s. The process consists of converting this text into an array that consists of zeros and ones. String indexing and type casting will be utilised in order to complete this conversion.[7]

Output: array which contains 0s and 1s.

2 Moving the plain text from CPU to GPU:

Input: Array which contains 0s and 1s

Output: The array will be copied to GPU's local memory.

3 Processing AES algorithm in parallel:

Input: Array resides in GPU's local memory

The process consists of feeding each 128-bit block into its respective individual core. A concurrent implementation of the AES algorithm will be carried out on each and every 128-bit block. The activity will be simultaneously processed by each core. Therefore, the 128-bit encrypted block will be produced by each and every core in the system.[8]

Output: Several 128 bit blocks of encrypted text.

4 Combining as a whole cipher text:

Input: Several 128 bit blocks of encrypted text.

Process: The each and every 128 bit blocks will be copied to separate array. This array contains the whole cipher text.

Output: The whole cipher text array

5 Moving the whole cipher text array from GPU to CPU:

Input: The whole cipher text

Process: The cipher text will be moved from GPU to CPU using parallel computing toolbox gather function. This function will move data from GPU to CPU.

Output: The cipher text will be residing in CPU memory.

Parallel implementation

This section discusses the CPU and GPU Cores Accelerating Technique that may be achieved with the help of the MATLAB Parallel Computing Toolbox. MathWorks has developed a toolkit they term the "parallel computing toolbox" in order to speed up both the CPU and the GPU. The CPU and GPU may be sped up with the help of this toolbox's preset routines. In the same way that parallel computing can be done efficiently in the MATLAB environment, so too can data analytics. Therefore, the security algorithm is also developed in MATLAB environment in order to link this security algorithm with the data analytics algorithm for decision making in IoT. The heterogeneous environment accelerates both the CPU and the GPU. Take a look at Figure 3 for a more in-depth portrayal.[9]

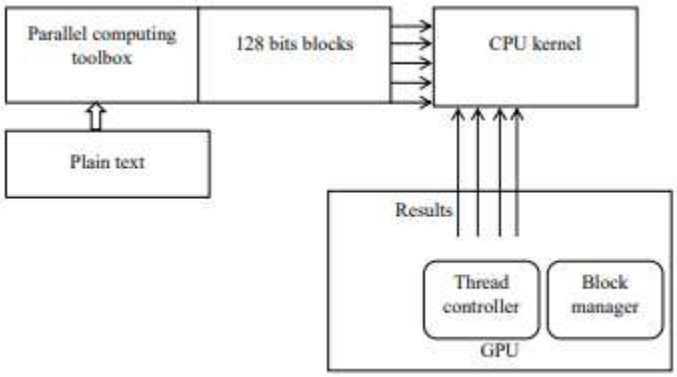

Figure 3 Parallel AES execution in GPU

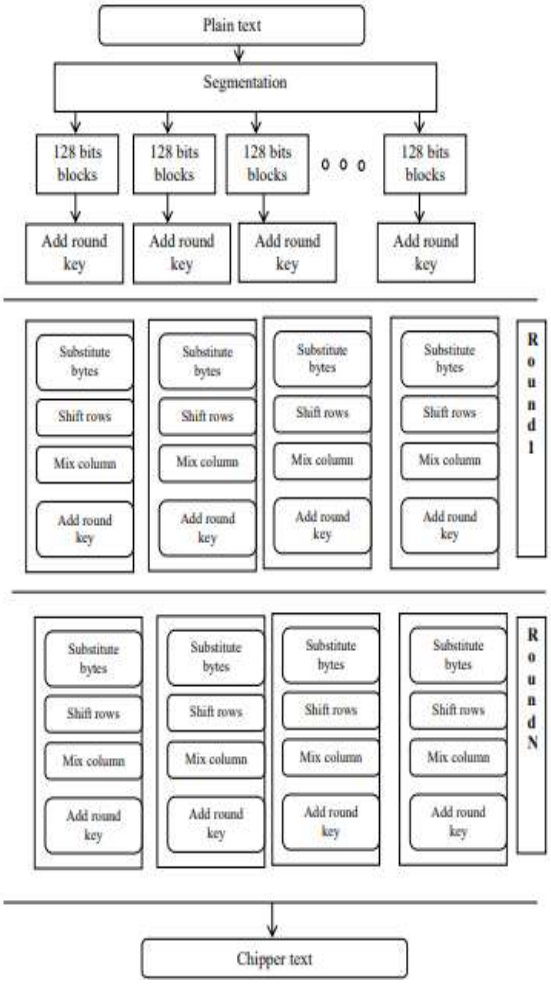

Figure 4 Parallellising AES algorithm

Experimental results

Number of simultaneous threads is equals to number of physical cores residing in the processing system. But we can create any number of threads in repeated core. So performance improvement can be seen only when creating the number of simultaneous threads equals to the number of physical cores. Some of the operations are not supported by GPU, in that situation, we have transferred the data from GPU local memory to CPU local memory and vice versa. Parallel AES algorithm implementation will give much performance improvement on large amount of data. The transformation of data from CPU to GPU or vice versa may decrease the efficiency of GPU. So the number of transfer between CPU and GPU needs to be minimised as much as possible. The basic level optimisation only done using GPU and the performance improvement will be more or less equals to i5 processing system performance improvement. The performance improvement rate will be proportional to number of cores residing in CPU. If the improvement rate needs to be increased, then use processing system which has more number of cores. By analysing various situations in executing parallel AES algorithm, we found the following feature selection process listed in Table 2.

Table 2 Feature selection in Intel processors

Implementation system	*Processing unit*	*Multi threaded*	*Basic optimisation*	*Vectorisation enabled*
I2	CPU	✓	✓	
I2	CPU	✓	✓	✓
I3	CPU	✓	✓	
I3	CPU	✓	✓	✓
I5	CPU	✓	✓	
I5	CPU	✓	✓	✓
I5	GPU	✓	✓	

Table 3 Serial code execution times

Processing system	*100 block parts*	*500 block parts*	*1,000 block parts*
I2	216.90 ms	1,092.20 ms	2,164.70 ms
I3	188.54 ms	896.34 ms	1,585.53 ms
I5	148.45 ms	746.10 ms	1,282.35 ms

Parallelization of the AES method is accomplished with both the CPU and the GPU. The processing system that we have utilised can increase the efficiency of the algorithm, which in turn leads to better overall performance. The effectiveness of the algorithm would increase by a factor of two if we employed an i2 processing system. The efficiency of the method would increase by a factor of four if we employed a processing system with i5 architecture. If we employ a processing machine with n cores, then the performance will improve at a rate that is n times greater. However, while speeding GPU, all we have done is use the GPU's most fundamental optimization settings for calculation. In addition, the GPU does not support all of the operations that are part of the AES algorithm. Therefore, these computations have been completed by the CPU. Therefore, the process of transforming data between the CPU and GPU has developed into an additional source of overhead. The calculations using the GPU required much more time than the concurrent processing on the i5. However, the fundamental optimization was completed in each and every processing system by default. When we ran the serial code in i2 and i3, i3 took less time than i2 did, and i3 and i5 produced the same results in both cases.

Table 4 Parallel code execution times

Processing system	*100 split texts*	*500 split texts*	*1,000 split texts*
I2	124.4174	657.0675	1,270.4
I3	102.7802	478.75	885.02
I5	82.2087	428.5335	735.2
GPU	95.8027	550.8035	847.3421

Both Le et al. (2010) and Di Biagio et al. (2009) worked on parallelizing the AES algorithm by utilising the CUDA framework and GPU-based implementation. This straightforward framework leads to an increase in speedup by making better use of the resources that are already available. The success of this solution is being

evaluated based on how well it scales in terms of processing power. Take a look at Figure 1 which illustrates the difference between serial and parallel processing in terms of execution.

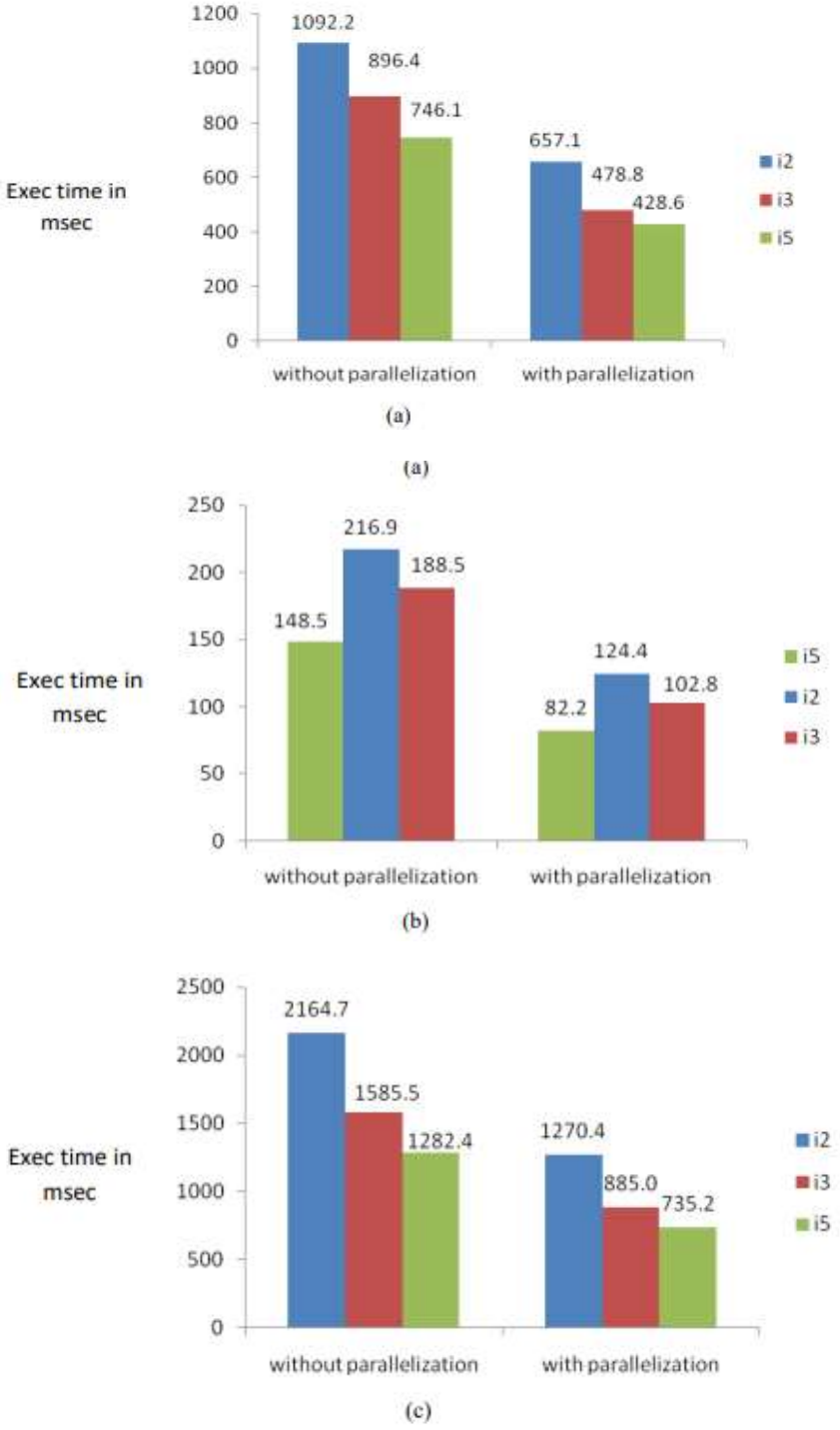

Figure 1 (a) 500 pieces of divided text chart depicting performance (b) Graph showing the performance of 100 partitioned blocks (c) Graph showing the performance of partitioned blocks (d) Graph showing the comparison between the i5 and GPU with some minimal optimization (see online version for colours)

Conclusions

When it comes to performance, doing data analytics in applications based on the Internet of Things presents a number of challenges. Incorporating security measures into data analytics systems will thus result in an increase in load. [10]The speed of the AES algorithm may be enhanced by employing parallelization and time stamp-based key generation. This can be done using either the CPU or the GPU. The injection of security algorithms is carried out effectively in such a way that it does not have an impact on the scalability of data analytics. Some of the operations cannot be performed within the GPU since MATLAB's environment prevents it from doing so. We are able to complete all of the procedures that are necessary for AES encryption if we make use of CUDA and several other types of approaches that accelerate computation. With this strategy, the graphics processing unit (GPU) was only employed for very basic level optimization, and only few of the GPU's available cores were put to good use. In the not-too-distant future, AES encryption and decryption operations will be carried out using the GPU to their fullest potential. The use of heterogeneous multi-core computing, such as CPU-GPU, coprocessors, and specialised processors like Intel Xeon, can boost the performance of applications that are based on the Internet of Things (IoT).

References

[1] Altinigneli, M.C., Plant, C. and Böhm, C. (2010) 'Massively parallel expectation maximization using graphics processing units', in Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, August, pp.838–846.

[2] Babar, S., Stango, A., Prasad, N., Sen, J. and Prasad, R. (2011) 'Proposed embedded security framework for internet of things (IoT)', in 2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronics Systems Technology (Wireless VITAE), IEEE, February, pp.1–5.

[3] Bae, G.C. and Shin, K.W. (2011) 'An efficient hardware implementation of lightweight block cipher algorithm CLEFIA for IoT security applications', Journal of the Korea Institute of Information and Communication Engineering, Vol. 20, No. 2, pp.351–358.

[4] Cao, H. and Chen, J. (2012) 'Multicore computing for SIFT algorithm in MATLAB® parallel environment', in 2012 IEEE 18th International Conference on Parallel and Distributed Systems (ICPADS), IEEE, December, pp.924–929.

[5] Irani, S., Picone, M., Gonizzi, P., Veltri, L. and Ferrari, G. (2011) 'IoT-OAS: an oauth-based authorization service architecture for secure services in IoT scenarios', IEEE Sensors Journal, Vol. 15, No. 2, pp.1224–1234.

[6] Kepner, J., Gadepally, V., Hancock, B., Michaleas, P., Michel, E. and Varia, M. (2012) 'Parallel vectorized algebraic AES in Matlab for rapid prototyping of encrypted sensor processing algorithms and database analytics', in High Performance Extreme Computing Conference (HPEC), IEEE, September, pp.1–8.

[7] Kumar, U., Borgohain, T. and Sanyal, S. (2011) Comparative Analysis of Cryptography Library in IoT, arXiv preprint arXiv: 1504.04306.

[8] Liu, B. and Baas, B.M. (2013) 'Parallel AES encryption engines for many-core processor arrays', IEEE Transactions on Computers, Vol. 62, No. 3, pp.536–547

[9] Mao, Y., Li, J., Chen, M.R., Liu, J., Xie, C. and Zhan, Y. (2012) 'Fully secure fuzzy identity-based encryption for secure IoT communications', Computer Standards & Interfaces, Vol. 44, pp.117–121.

[10] Wang, M., Wang, B., He, Q., Liu, X. and Zhu, K. (2012) Analysis of GPU Parallel Computing based on Matlab, arXiv preprint arXiv: 1505.06561.