A project report on

IMPROVED PARALLEL HILL CIPHER ALGORITHM USING MAPREDUCE

Submitted in partial fulfillment of the requirement for the award of degree of

Masters of Technology

In

Information Systems

Submitted by:

Akshay Mool

(2K15/ISY/01)

Under the guidance of

Mr. Vinod Kumar

(Associate Professor, Department of Computer Science and Engineering, DTU)

2015-2017

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

Bawana Road, Delhi-110042

CERTIFICATE

This is to certify that **Akshay Mool (2K15/ISY/01)** has carried out the major project entitled "**Improved Parallel Hill cipher algorithm using MapReduce"** in partial fulfillment of the requirements for the award of Masters of Technology Degree in Information Systems during session 2015-2017 at Delhi Technological University.

The major report is bonafide piece of work carried out and completed under my supervision and guidance. To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any degree or diploma.

Mr. Vinod Kumar

Associate Professor

Department of Computer Science and Engineering

Delhi Technological University

Delhi-110042

ACKNOWLEDGEMENT

I express my sincere gratitude to my project mentor Mr. Vinod Kumar, Associate Professor, Department of Computer Science and Engineering, Delhi Technological University, Delhi, for providing valuable guidance and constant encouragement throughout the project. It is my pleasure to record my sincere thanks to him for his constructive criticism and insight without which the project would not have been shaped as it has.

I extend my gratitude to Dr. Kapil Sharma, Head of Computer Science and Engineering Department for permitting me access to the department facilities and giving me opportunity to work on this project.

I thank God for making all of this possible, my parents and friends for their constant support and encouragement throughout the project work.

Akshay Mool

2K15/ISY/01

M.Tech (Information Systems)

Department of Computer Science and Engineering

Delhi Technological University

ABSTRACT

In this thesis, a hill cipher algorithm using parallel block matrix multiplication on MapReduce is put forward to reduce the time for encryption process and provide additional security against internal and external attacks. The data being stored in the cloud is usually very large, the demand to decrease the time for encryption, as well as to increase the security of the overall cloud, becomes necessary. To overcome these drawbacks, a parallel algorithm on the modified hill cipher is implemented on a MapReduce framework, with parallelism implemented for easier simplification of the crux of hill cipher. The key matrix generated for the algorithm is intended to be self-invertible, to decrease the decryption time of the encrypted text. The experimental analysis on the input data demonstrates the effectiveness of the proposed technique.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1: INTRODUCTION

This thesis focuses on applying the Hill Cipher algorithm in a parallel manner on MapReduce model. A parallel algorithm for modified Hill Cipher is designed with respect to the parallelization of the block matrix multiplication. This approach has observed to improve the performance and speed of encryption cycle Hill Cipher by converting the plaintext as blocks, and then converting it to matrix, with applying parallel computations on large scale. This technique takes complete advantage of computational capabilities of high-performance machines involved in parallel computation, therefore achieving faster Hill Cipher encryption for large sets of data. [11]

The science of cryptography deals with hiding of messages in such a way that only the authorized people are allowed to access them. This is a two-process method – encryption and decryption. Encryption is the process of converting the message from it's understandable form to a form such that it's meaning is hidden and is not very obvious upon direct investigation. It is usually done using a "key", which is used to encrypt the message, and is known only to the sender and the receiver of the message. The process of decryption is the opposite of that of encryption, i.e. it is the process of converting the encrypted message back to its original form that is supposed to have a meaning for the concerned recipient. [2]

Cipher is the encoded form of the message that is the result of the encryption process that is applied on the original message.

There are many types of ciphers, that are the results of different types of encryption processes being applied to the message. One such type is a Poly-alphabetic cipher. A Polyalphabetic cipher is a cipher in which separate occurrences of the same letter can have different substitutions in the corresponding cipher text. One such type of polyalphabetic block ciphers is Hill Cipher. [4]

In this work, Hill Cipher is being implemented for the security of large databases. For efficiency and speed of the performance, parallel block matrix multiplication is implemented in the process for applying Hill Cipher. The parallelism is helpful in utilizing maximum machine capabilities for computations and generating the overall result. [3]

MapReduce is a type of programming framework that is used to process huge data collections with a distributed and parallel algorithm. It usually breaks down the input data into separate blocks, each of which is processed by each algorithm involved in the framework in a parallel way. MapReduce comes into play in such situations where there is a large amount of data that has to be processed in a speedy and efficient manner, where the data can be broken down into separate independent parts and each part can be dealt with individually. [11]

Formal definitions:

The Hill Cipher was proposed by Lester S. Hill in the year of 1929. It acts on a group of letters instead of acting on individual letters separately. The whole plaintext (message) is divided into a number of blocks of equal sizes. All the blocks are encrypted one at a time – and each character in the block contributes to the encryption of other characters present in the same block. The key is of the size of $m \times m$, where m is the size of each block. Hill Cipher ensures that repeated occurrences of same letters or group of letters in the message have different substitutions in the corresponding cipher, so that breaking of the cipher to obtain the original plaintext is not obvious and easy by any malicious attacker trying to decipher the message who does not have the proper authority to do so.

MapReduce framework is used for efficient processing of large data sets that generates intermediary key/value pairs and later combines the results to generate an overall final outcome. [11]

1.1 OBJECTIVE

To enhance and improve the performance of base hill cipher implementation, using Parallel Block Matrix Multiplication and MapReduce framework.

1.2 MOTIVATION

The increase in the amount of information every day these days leads to a requirement of a lot of storage space and the added security that comes with it. The processing and computations required for such huge amounts of data becomes quite cumbersome. And these days when tonnes of data is being added every moment, efficient and speedy

processing of the same becomes important as well as necessary to cope up with the incoming amount.

Also, increase in the amount of data leads to an obvious increase in the amount of sensitive data that needs to be protected from elements with malicious intents, whether it be from a hacker, cracker or a software. The sensitivity of the data leads to the requirement of security for preventing such malicious elements from accessing and interfering with the integrity of such data. Thus, the added requirement of speedy and efficient processing and security of the large data sets leads to research in the field of MapReduce framework implementation with respect to a strong cryptography technique.

In this thesis, a Hill Cipher algorithm with parallel computations is suggested to lower the time taken for the encryption cycle. Because the cloud data is becoming quite large these days, demands to reduce the time taken for encryption of the data has increased significantly, with the storage security in the concerned cloud. For catering to this need, a Modified Hill Cipher using Parallel algorithm is utilized for working on the MapReduce model. It is a symmetric-transposition encipherment technique, and the parallel block matrix multiplication purely achieves the parallelism here. The self-invertible key matrix is generated to reduce the decryption time. Experimental analysis on the input data demonstrates how effective the proposed technique has proven to be.

Due to the advanced developments in cloud computation in recent years, information security has become a very essential problem these days. As large volumes of unprotected data are communicated over shared cloud, the process of encryption for the large data is recommended to shield the data protection, and reducing the time of encryption simultaneously.

Parallel computing is a very essential property for cloud computing. It has driven many architecture models designs of parallel computing, like the MapReduce framework, the parallel RAM (PRAM) and BSP model. In this thesis, owing to the practical considerations, the MapReduce framework has been chosen. The MapReduce framework has the features of being very simple and general. The model that executes the algorithm of MapReduce manages the distribution of the work between the of number of machines effectively.

In 1929 Lester Hill published an article in the American Mathematical Monthly called "Cryptography in an Algebraic Alphabet." In it, he describes the Hill cipher, in which a plaintext message is encrypted by matrix multiplication. This cipher is especially noticeable due to the fact that the letters are enciphered in groups because of the way in

which matrices are multiplied, which means it bypasses frequency analysis and other of traditional cryptanalysis techniques. The Hill cipher was also responsible to combine mathematics and cryptography. Using mathematics to encode messages is noted to be a very important turning point for development in the cryptographic techniques. Addition of mathematics to the cryptographic process unlocked various new possibilities. The vast use of public key cryptography these days is all due to the help of Hill cipher. [3]

There is a very huge demand for large scale data processing these days. Owing to this increase in the requirement, research work for decreasing processing time and increasing the speed for large data sets began in various sectors, and is still progressing at a fast rate.

1.3 GOAL

This thesis introduces an improved and secured methodology for parallel implementation of hill cipher on a MapReduce framework, taking into account the complexity and large sizes of the input data sets, as well as the probability of malicious attacks on the concerned data. Hill Cipher is used here because due to the size of the data sets, they are divided into smaller chunks of data, and the encryption and transmission of these individual chunks have no effect on other blocks of data. Moreover, it can be easier to implement than other cryptographic schemes because it avoids time consuming bit manipulation and they operate on computer sized blocks of data. MapReduce provides an easy to use, clean abstraction for large scale data processing. It is very robust in fault tolerance and error handling, and can be used for multiple scenarios for huge computations. Restricting the programming model to the Map and Reduce paradigms makes it easy to parallelize computations and make them fault-tolerant.

1.4 THESIS ORGANISATION

Chapter 2 includes literature review for Hill Cipher, matrix multiplication and MapReduce framework. The formulation of the problem and MapReduce approach based on data size and speed of processing with the help of parallelism in matrix multiplication are briefly given in this chapter.

Various approaches to generate a cipher, and architectural models for parallel computing, are discussed in Chapter 3. This chapter is organized in a step-wise manner.

We will start with a simple cryptographic model for cipher generation and approach to the use of Hill cipher for security in the thesis as we move along. We will cover some basic features of some of the architectural models that have evolved over time due to the need of faster processing of large data, and some of the mathematical theory behind proposed method here.

In Chapter 4, we discuss the results and evaluate them with the implementation. We elaborate the experimental results we obtained with implementations. Security of the proposed algorithm is analysed and experimental results are also presented. Discussion and analysis of the results will be given in this Chapter. Conclusion and future directions are presented in Chapter 6.

Chapter 2: LITERATURE SURVEY

The Hill cipher was introduced by L.S. Hill in 1929. It is a famous polygram and a symmetric-key cipher which is based on matrix multiplication but it is vulnerable to the known-plaintext attack. Although its weakness to cryptanalysis makes it barely possible to use in practice, it still provides a good insight in both cryptology and linear algebra. The Hill cipher is a polyalphabetic block cipher that has many advantages like concealing letter frequencies of the plaintext, its simple approach because of matrix multiplication, similarity of processes of encryption and decryption, and high speed and throughput [2]. In 2009 and 2011, Toorani and Falahati introduced two variations of the classical Hill Cipher, with guidelines and protocols for the communication of encrypted messages. They claim that the new variants get rid of the weaknesses of the original Hill Cipher, and do not succumb to any ciphertext-only, known-plaintext, chosen-plaintext, or chosen-ciphertext attack [3].

Public-key encryption systems like RSA, El Gamal, are more secure cryptosystems than any other private key encryptions. But due to their relative high computational complexity, and therefore slow processing, cannot be used for transferring large datasets for practical purposes. So, they are better suited for authentication purposes and key exchange scenarios that use private keys for transfer of data. Therefore, necessity of an efficient and highly secure private key cryptosystems become an important need. One of such methods is a Block Cipher, called the Hill Cipher [4].

MapReduce is a type of programming framework, with an associated application to process and generate huge data collections. A map function is specified by the users, that processes a key/value pair for generating a set of transitional key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real-world tasks are expressible in this model [5].

Advanced architecture computers of recent disposition have such hierarchical memories in which accesses to data in the upper levels of the memory hierarchy (registers, cache, and/or local memory) are faster than those in lower levels (shared or off-processor memory). One of the techniques to effectively exploit the power of such machines more is to evolve algorithms that boost reuse of data held in the higher levels of the hierarchy, thereby reducing the need for more expensive accesses to lower levels. For dense linear algebra

computations, this can be done by using block-partitioned algorithms, that is by recasting algorithms in forms that involve operations on submatrices, rather than individual matrix elements. The Level 3 Basic Linear Algebra Subprograms (BLAS) executes various commonly-used matrix operations, and are accessible in better form on almost every computing platforms.

Different advancements suggested for matrix multiplication comprises 1D-systolic, 2D-systolic, broadcast-multiply-roll, the transpose algorithm, and Cannon's algorithm. Two latest efforts advance the work by Fox et al. to general meshes of nodes: the paper by Choi et al.[6] uses a two-dimensional block-wrapped (block-cyclic) data decomposition, while the papers by Huss-Lederman et al.[7,8] use a 'virtual' 2-D torus wrap data layout. Both these efforts report very good performance attained on the Intel Touchstone Delta, achieving a sizeable percentage of peak performance.

Chapter 3: VARIOUS APPROACHES FOR CIPHER GENERATION AND ARCHITECTURAL MODELS OF PARALLEL COMPUTATION

In this chapter, we will discuss about the different types of simple cryptography and cryptanalysis systems, as well as several architectural models of parallel computations, because of the parallel computing in cloud computation.

3.1 CRYPTOGRAPHY

The fundamental objective of cryptography is to enable two people, usually referred to as Alice and Bob, to communicate privately over an insecure channel, in such a way that an opponent, Oscar, does not understand what is being said. The information that Alice wants to send to Bob is generally called "plaintext", and could be anything, ranging from text in any language to numerical symbols or digital media. Alice encrypts the plaintext using a predetermined key between Alice and Bob, and sends the resulting ciphertext over the insecure channel. Oscar, who tries to eavesdrop on the conversation, sees the ciphertext, but is unable to generate any meaning out of it; but Bob can, because he has the predetermined key which is used to decipher the ciphertext back into the plaintext.

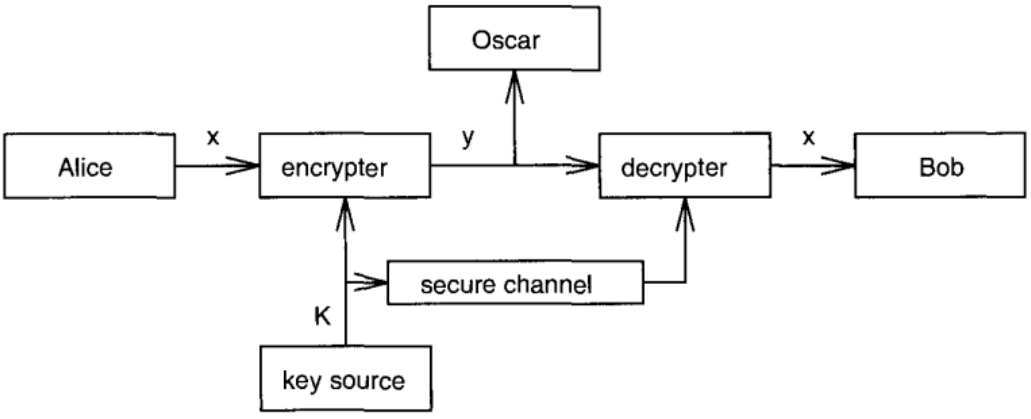

Figure 1: Communication channel for a cryptographic system

Although in the past, cryptography was concerned with only two processes: encryption and decryption, of the messages using secret keys, in the present day, is described using 3 unique techniques: Hashing, Asymmetric-key encryption, and Symmetric-key encryption.

3.1.1 Symmetric-Key Encipherment

In Symmetric-key encipherment, sender sends a message to receiver over an insecure channel, assuming that any adversary is not able to understand the meaning of the information simply by eavesdropping. Symmetric-key encryption uses a sole secret key for encrypting and decrypting algorithms. Encrypting and decrypting in this technique can be visualized as locking electronically. Sender inserts the information in a box, then bolts it with the secret key that is shared with the receiver, then the receiver unbolts the box using the secret key that is shared, and gets the information.

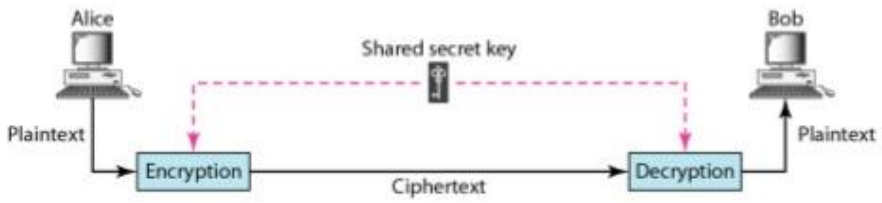

Figure 2: Symmetric-key encipherment

3.1.2 Asymmetric-Key Encipherment

In Asymmetric-key encipherment, we have the same situation as the symmetric-key encipherment, but with a few exceptions. Firstly, there are two keys instead of just one common key: a public key and a private key. For Alice to send a secured message to Bob, she encrypts the message using public key of Bob. Bob then decrypts the message using his own private key.

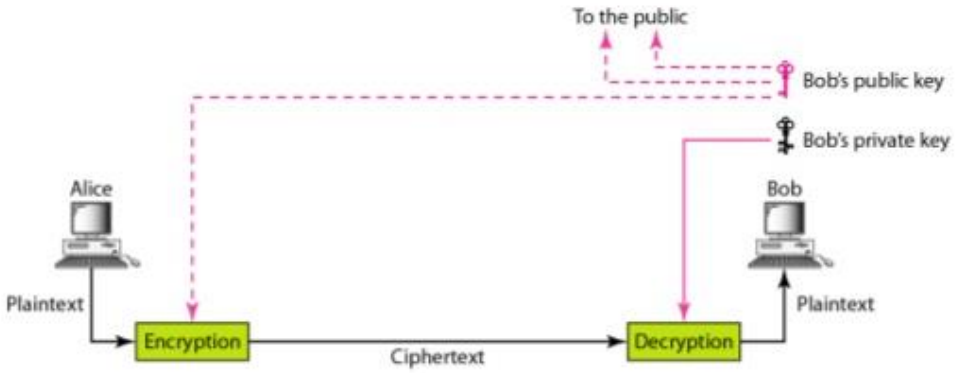

Figure 3: Asymmetric-key encipherment

3.1.3 Hashing

Hashing uses a message-digest of fixed length that is generated out of a information of variable length. The message-digest is usually very small when compared to the information itself. For the effective encipherment, the information and its digest are both sent to receiver. This technique is especially useful in relation with checkvalues, like checksums, which are used to provide integrity of the message.

3.2 TRADITIONAL CIPHERS

Traditional ciphers are divided into two types of ciphers: Substitution ciphers and Transposition ciphers.

3.2.1 Substitution Ciphers

A substitution cipher replaces one symbol in the plaintext with a different symbol in the ciphertext. If the symbols in the plaintext are alphabets, we simply just replace one alphabet with another. Substitution ciphers are again divided into two types: Mono-alphabetic ciphers and Poly-alphabetic ciphers.

3.2.1.1 Mono-alphabetic ciphers

In monoalphabetic ciphers, a symbol in the plaintext is always substituted with the same symbol in the ciphertext, regardless of their position in the message. This means that the relationship between the symbol and its substitute in the ciphertext is always one-to-one. Examples of monoalphabetic ciphers are: Additive cipher, Shift cipher, Caesar cipher.

3.2.1.2 Poly-alphabetic ciphers

In polyalphabetic ciphers, different locations of a same character in the same plaintext may have different substitutes in the ciphertext. This means that the relationship between the symbol and its substitute in the ciphertext is many-to-one. Examples of polyalphabetic ciphers are: Autokey cipher, Playfair cipher, Vigenere cipher, **Hill cipher.**

3.2.2 Transposition ciphers

A transposition cipher does not replace the symbols in the plaintext, instead it changes the location of the symbols in the plaintext to form the ciphertext. In other words, it reorders (transposes) the position of symbols. Examples of transposition ciphers are: Keyless Rail-fence cipher, Keyed permutation cipher.

3.3 PARALLEL COMPUTATION ARCHITECTURAL MODELS

Due to advances in the sizes of databases and data sets, and increase in the requirement of fast processing of that large data, various architectural models have been proposed and developed to aid parallel computations. Some of those models are as follows:

3.3.1 BSP MODEL

BSP stands for Bulk Synchronous Parallelism. It is a parallel programming model based on Synchronizer Automata, that is a methodology in Distributed Algorithms. [5]

The model consists of several memory-processor pairs, a communication network that delivers messages to the devices in the network in a point-to-point manner. It also comprises of a mechanism for the efficient barrier synchronization for all or a subset of processes. The BSP model lacks any special combining, replicating or broadcasting facilities.

This model considers communication and computation capabilities with respect to the whole program and the computer that is executing the program, instead of taking care of individual processes and separate communications.

The main features of BSP model are:

- Ease of writing programs
- Independent of the target architecture
- Model performance is predictable.

3.3.2 PARALLEL RAM (PRAM)

PRAM stands for Parallel Random-Access Machine. It is a parallel computing model that is just a natural extension of a RAM in the way that each processor acts as a RAM for the program. [5] It is an abstract machine that is helpful in designing algorithms suitable for parallel computations. All the processors operate synchronously, and is one of the earliest and best-known model for parallel computation. It consists of a global access memory, a set of processors that run usually the same program, with the help of a private stack. The

processors have the freedom of accessing all memory cells in a unit time, and all communications are carried out using the shared memory.

The main features of PRAM are:

- The number of operations executed in one cycle on p processors does not exceed p.
- A processor can perform any read/write operation on any shared memory cell in a unit time.
- It is simple due to the fact that it abstracts from any synchronization overhead or communication.
- It can be established as a benchmark, as if a problem has no feasible solution on PRAM, no other parallel machine can produce any feasible or efficient solution.

3.3.3 MAPREDUCE MODEL

The researchers at Google came up with the common processes that are essential in processing large scale data inputs, provided the processing could be done at multiple machines at the same time: Map and Reduce. Utilizing these two functions, Google came up with a framework called MapReduce. [11] The problems solved by the MapReduce framework are:

1. Fault-tolerance- handling component failures
2. Distribution- distributing data to the various machines
3. Parallelization- parallelizing the computations required

The main advantage of using MapReduce is the convenience of not being forced to move the data to different locations, instead a program is sent to the data centers to process the chunks of data on their places, which is unlike other traditional data warehouses and relational databases.

The data and computing division distributions among various processors in the MapReduce scenario gives multiple advantages:

- Using MPI (Message Passing Interface), low effort is required with respect to the data handling timing.
- MPI gives data exchange flexibility.

Chapter 4: THE PROPOSED WORK

4.1 OVERVIEW

The need of security with databases change with different organizations, depending on the information type and the priority of importance it holds for the organization. The flexible and unreliable nature of the cloud makes it vulnerable to inside and outside attacks, and the virtual nature of the model makes security of cloud environments a complex process, as per the demand.

Parallel computing is one of very important components of cloud computation. One of the architectural model used for implementation of parallel computing is the MapReduce Framework. The MapReduce model is chosen for this paper due to practical purposes and ease of implementation. MapReduce is a type of programming framework and an associated application to process and generate huge data collections with a distributed, parallel algorithm on a set of machines. Properties of this model include generality and simplicity. MapReduce allows the effective parallelization of processing data stored in a file system. [7]

In this thesis, the application of a Hill cipher algorithm using parallelism on MapReduce model has been focused. A parallel Hill Cipher algorithm with some modifications in accordance with the parallelization of the blocks for block matrix multiplication has been implemented. This approach improves the speed of processing of Hill Cipher by converting the plaintext into computable blocks and then computing them parallelly. The algorithm tries to make full use of high performance capability of the machine cluster, and therefore can encrypt large data files using the Hill cipher. [3]

4.2 IMPLEMENTED VERSION OF HILL CIPHER

The basic hill cipher was introduced by Lester Hill. It is a Polyalphabetic block cipher, meaning each occurrence of a character may have a different substitute in the encrypted text. When the plaintext is to be encrypted, it is converted to a corresponding numerical value

matrix **P**, and a private key matrix **K** is generated for encryption. [1] If **C** is the cipher matrix for the plaintext matrix **P**, then

$$\textit{Encryption\_process: } \mathbf{C} = \textit{Encrypt}(\mathbf{K},\mathbf{P}) = \mathbf{KP} \bmod m$$

$$\textit{Decryption\_process: } \mathbf{P} = \textit{Decrypt}(\mathbf{K^{-1}},\mathbf{C}) = \mathbf{K^{-1}C} \bmod m$$

The inverse of the key **K, K^{-1}**, leads to consumption of memory resources and time, because it is a very tedious task. Also, in some cases, matrix inverse is non-existent, so it creates problems in decrypting the cipher text. The modified hill cipher implemented for this thesis makes use of the invertibility of key matrix for the process of encryption. Let matrix **K** be a M X M matrix. [1]

$$\mathbf{K}=\begin{bmatrix} k_{11} & \dots & k_{1M} \\ \dots & & \dots \\ k_{M1} & \dots & k_{MM} \end{bmatrix}$$

The partition of **K** is assumed as

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}$$

$\mathbf{K_{12}}$ is assumed to be one of the factors, therefore $\mathbf{K_{12}} = (\mathbf{I}-\mathbf{K_{11}}) \bmod m$, and $\mathbf{K_{21}} = \mathbf{(I+K_{11})} \bmod m$, and $\mathbf{K_{11}+K_{22}} = \mathbf{0} \bmod m$. [1]

Now, using this, the algorithm for generating Self-invertible matrix is as follows: [1]

ALGORITHM FOR GENERATING SELF-INVERTIBLE KEY MATRIX

Input: K_{22}, scalar constant a, modulus m

Begin

1. Obtain an arbitrary M/2 x M/2 Matrix K_{22}
2. $K_{11} = -K_{22} \bmod m$
3. $K_{12} = (I+K_{22}) * a \bmod m$
4. $K_{21} = (I+K_{11}) * 1/a \bmod m$
5. Therefore, key matrix K is obtained by (K_{11} K_{12} K_{21} K_{22}).

End

Table 1: Algorithm for generating Self-Invertible Key Matrix

4.3 MAPREDUCE VERSION OF HILL CIPHER

MapReduce is a type of programming framework and an associated application to process and generate huge data collections with a distributed, parallel algorithm on a set of machines. [11]

- **"Map" step:** Every worker node implements the "map()" method to the local data, and copies the output on a temporary storage device. A dictating node ensures that only one copy of redundant input information is managed.
- **"Shuffle" step:** All the worker nodes redistribute the data depended on the output keys that were produced by the "map()" function, so that all data belonging to one key is located on the same worker node.
- **"Reduce" step:** Worker nodes process each collection of output data according to each key parallelly.

Now, an algorithm for MR-parallel hill cipher scheme, PHC = (Generate, Split, Map, Part, Reduce), is as follows: [1]

ALGORITHM FOR HILL CIPHER ON MAPREDUCE

1. **Generate** (K_{22}, m, a): generate the key matrix using previous algorithm.
   K <- Gen(K_{22}, m, a).
2. **Split**(Plaintext): with respect to the processor used by the machine, the size of block is computed, after that the block ID is generated as key for map process. Therefore the output pair for map process is **Output** (block ID (i,j,p), plaintext).
3. **Map**(block ID (i,j,p), plaintext): calculate the allocation of block according to the block ID. Commence the block matrix multiplication next. Output is partial sum of the cipher matrix:
   $psum = psum + K^{+}_{i,p}(x,y)\ X\ P^{+}_{p,j}(y,z)$
   Output (block ID (i,j,p), partial sum $C^{+}_{i,j}$)
4. **Part** (block ID(i,j,p), partial sum $C^{+}_{i,j}$): divide the block ID parameter (i,j,p), the partial sum having the same parameters of (i,j) will be assigned to the same reduce.
5. **Reduce** (block ID(i,j,p), Set(partial sum C+i,j)): atomic add the partial sum to compute the final cipher matrix.

Table 2: Algorithm for MapReduce Hill Cipher

This algorithm describes the common encryption technique. The process of decryption follows the same routine as the process of encryption with the identical key matrix.

4.4 PARALLEL BLOCK MATRIX MULTIPLICATION

The block matrix multiplication gives rise to the parallelism in the algorithm of hill cipher.

The block matrix multiplication in parallel manner is carried out as follows:

ALGORITHM FOR PARALLEL BLOCK MATRIX MULTIPLICATION

Number of processors in parallel machines are p. Two square matrices A, B of size n have to be multiplied:

1. Both matrices are divided in p square blocks.

2. A processes matrix of size $p^{1/2}$ x $p^{1/2}$ is generated, in order that each process is able to maintain a block of both input matrices.

3. Each process is given one block, and those sub blocks are multiplied together, and the results are appended to the partial results of C sub-blocks.

4. The sub-blocks of A are shifted left one step and the sub-blocks of B are shifted up one step.

5. Steps 3 and 4 are repeated sqrt(p) times.

Restrictions:

- The available number of processors must be an exact square root.
- The accurate distribution of the all data to the available processors must be possible.

Table 3: Algorithm for Parallel Block Matrix Multiplication

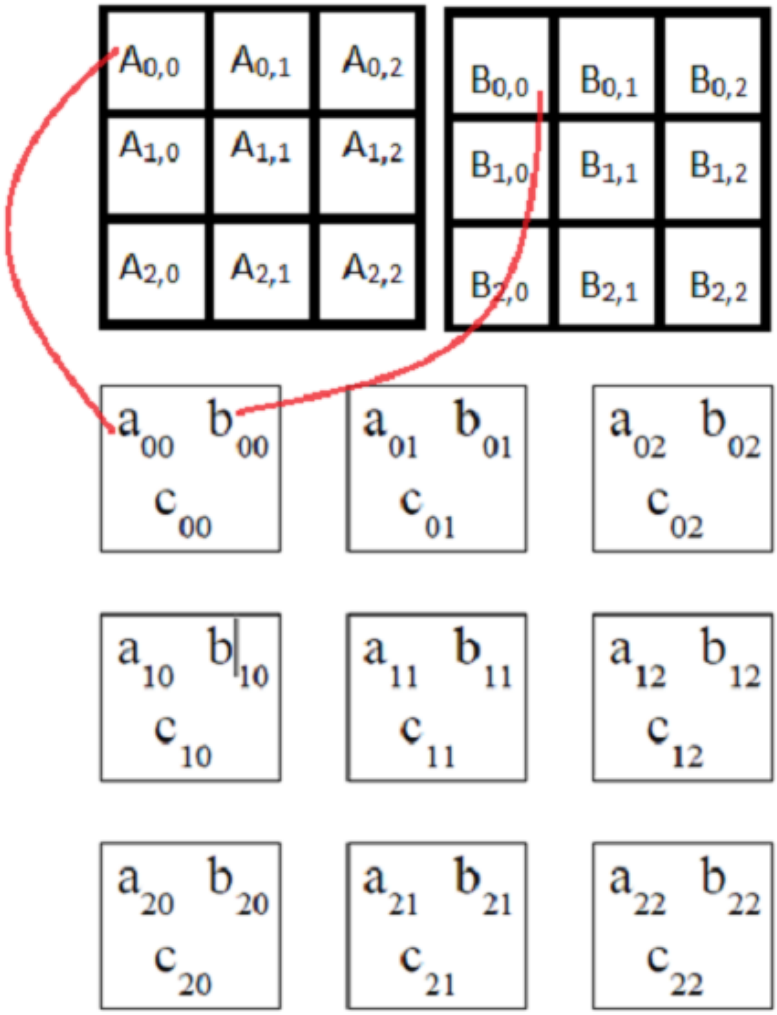

Figure 4: Demonstration of the Parallel multiplication of two matrices

The number of reduce process would be just one, because merging the result in a single file ultimately is needed. All the map results are received by a single reducer.

4.5 COMPLEXITY OF THE PARALLEL HILL CIPHER ALGORITHM

To calculate the complexity of the method, we take into consideration each major step in the whole process. The complexity of the matrix multiplication is found out to be $O(n^3)$. Let the dimension of plaintext matrix be P, key matrix be K, cipher matrix be C. [1] Furthermore, to carry out the parallel block matrix multiplication, we have

$$P=K=C=N$$

Let the mapper count be M. Therefore,

$$M=p \times k \times c$$

Sequential time, T_{seq}, can be represented as time for key matrix generation with the sequential matrix multiplication. So,

$$T_{seq} = T_{key} + O(n^3)$$

Parallel time, T_{par}, can be written as the time for generating key matrix with the parallelizing of the matrix multiplication. So,

$$T_{par} = T_{key} + T_{m\_par}$$

The speed boost for the parallel matrix multiplication can be found out by dividing the time into two parts: the time for computing the matrix multiplications, and the time that the reducer has to wait to read all the mapper results. Therefore,

$$T_{m\_par} = T_{comp} + T_{wait}$$

And T_{comp} can be found by

$$T_{comp} = N^3/M$$

The number of output key and value pairs for the mapper are

$$K \times (p \times C + c \times P)$$

And the total results from the Result process are PxC.

Because we assumed P=K=C=N, therefore,

$$p=k=c=\sqrt[3]{M}$$

Let t_w denote one pair of records written by mapper. Therefore,

$$T_{wait} = K \times (p \times C + c \times P)t_w$$

$$= (cN^2+pN^2)t_w$$

$$= 2\sqrt[3]{M}N^2t_w$$

$$T_{par} = (N^3/M + 2\sqrt[3]{M}N^2t_w)$$

Therefore, the increase in the performance is

$$\eta=\frac{T_{seq}}{T_{par}}=\frac{N^3}{\left(\frac{N^3}{M}+2\sqrt[3]{M}N^2\,t_w\right)}=\frac{M}{1+\frac{2M^{2/4}\,t_w}{N}}$$

When $N->\infty$, $T_{seq}\approx MT_{par}$

4.5 PROPOSED METHOD FOR INCREASING SECURITY

The major drawback of this parallel hill cipher implementation is the security of key and consequently, its generation by the attacker.

The plain text for the Hill cipher is converted to several 4X4 matrices and those are encrypted individually. It would be too irrelevant to have a separate key for every 4X4 portion of the plaintext; therefore, we take a common key for all the matrices. The key, which is the self-invertible matrix, is generated by using 4 elements. These elements are user (client) dependent, and hence these elements are asked to enter by the user at the encryption phase. The security of these elements is the issue of concern in this method, because if the attacker can figure out these elements, the whole key can be generated using these 4 elements. These elements can be made secure using several standard or user-defined methods. The method implemented in this report is a simple one. Each element is generated using 4 different integer inputs by simple mathematical operations performed on all of them for each of the 4 elements of the self-invertible matrix. We can secure of these 4 elements of using random number generator:

$$a1 = ((x1 * p1) + q1) \bmod m1$$

$$a2 = ((x2 * p2) + q2) \bmod m2$$

$$a3 = ((x3 * p3) + q3) \bmod m3$$

$$a4 = ((x4 * p4) + q4) \bmod m4$$

where,

x1, x2, x3 & x4 are user entered prime number integer

p1, p2, p3, p4, q1, q2, q3, q4, m1, m2, m3 & m4 are prime number.

Chapter 5: RESULTS AND EVALUATIONS

The performance of the proposed system is calculated on the basis of milliseconds, as the machine cannot generate the results in such small units as nanoseconds.

5.1 RESULTS ON SEPARATE PROCESSORS

Firstly, the performance is measured on an i3 processor computer:

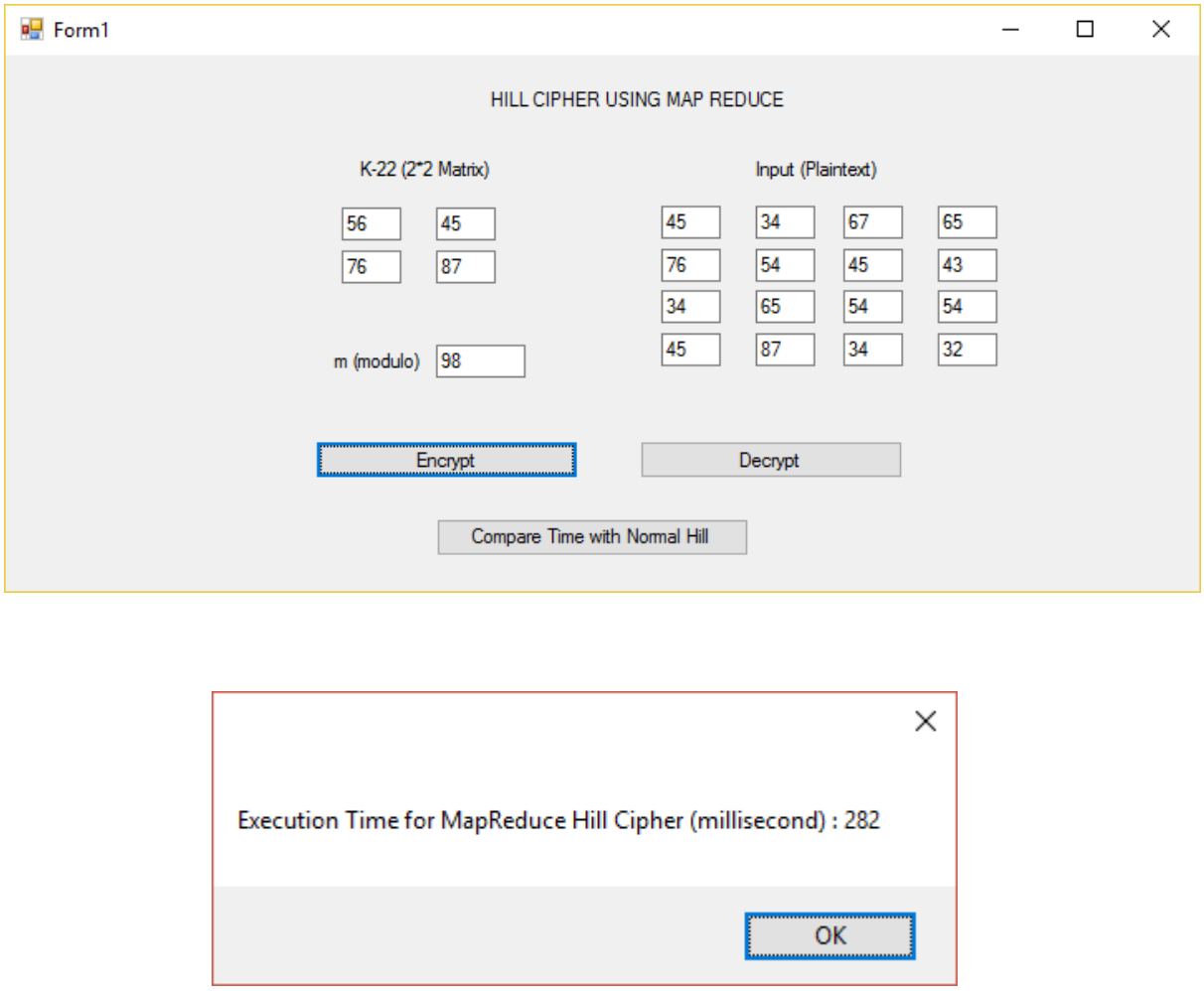

Figure 5: Executing the encryption cycle of Hill Cipher using the Parallel Multiplication on i3 processor

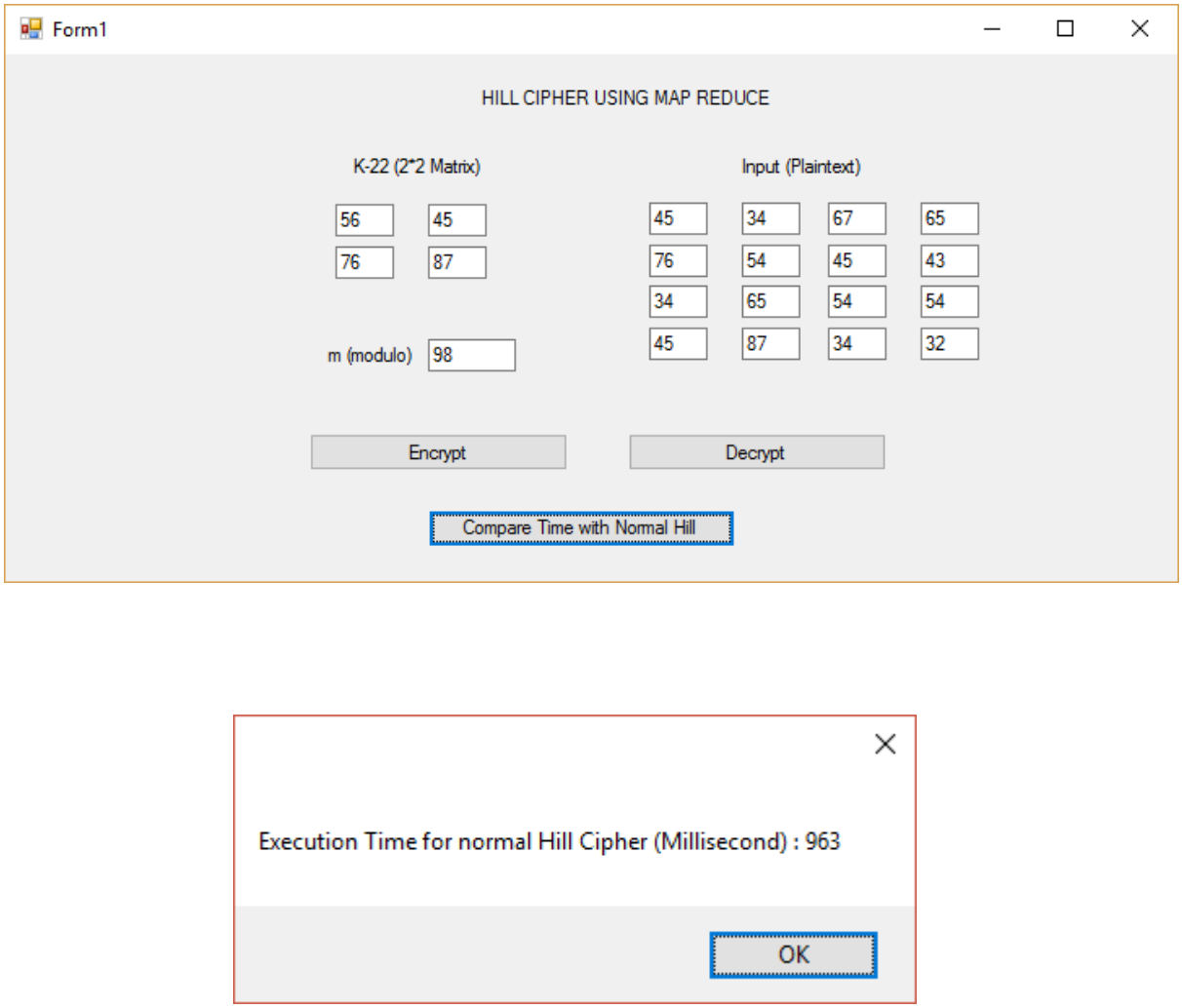

Figure 6: Executing the encryption cycle of Hill Cipher without Parallel Multiplication on an i3 processor

As we can see from the execution time of the Parallel Multiplicative version of Hill cipher versus the normal Hill cipher, it is observed that the performance of the parallel version is better than the performance of Hill cipher without applying the matrix multiplication.

Performance is also compared on other processors to check the accuracy and validity of the system.

Now, comparing the performances of the classical Hill Cipher vs the Parallel Hill Cipher on an i5 processor:

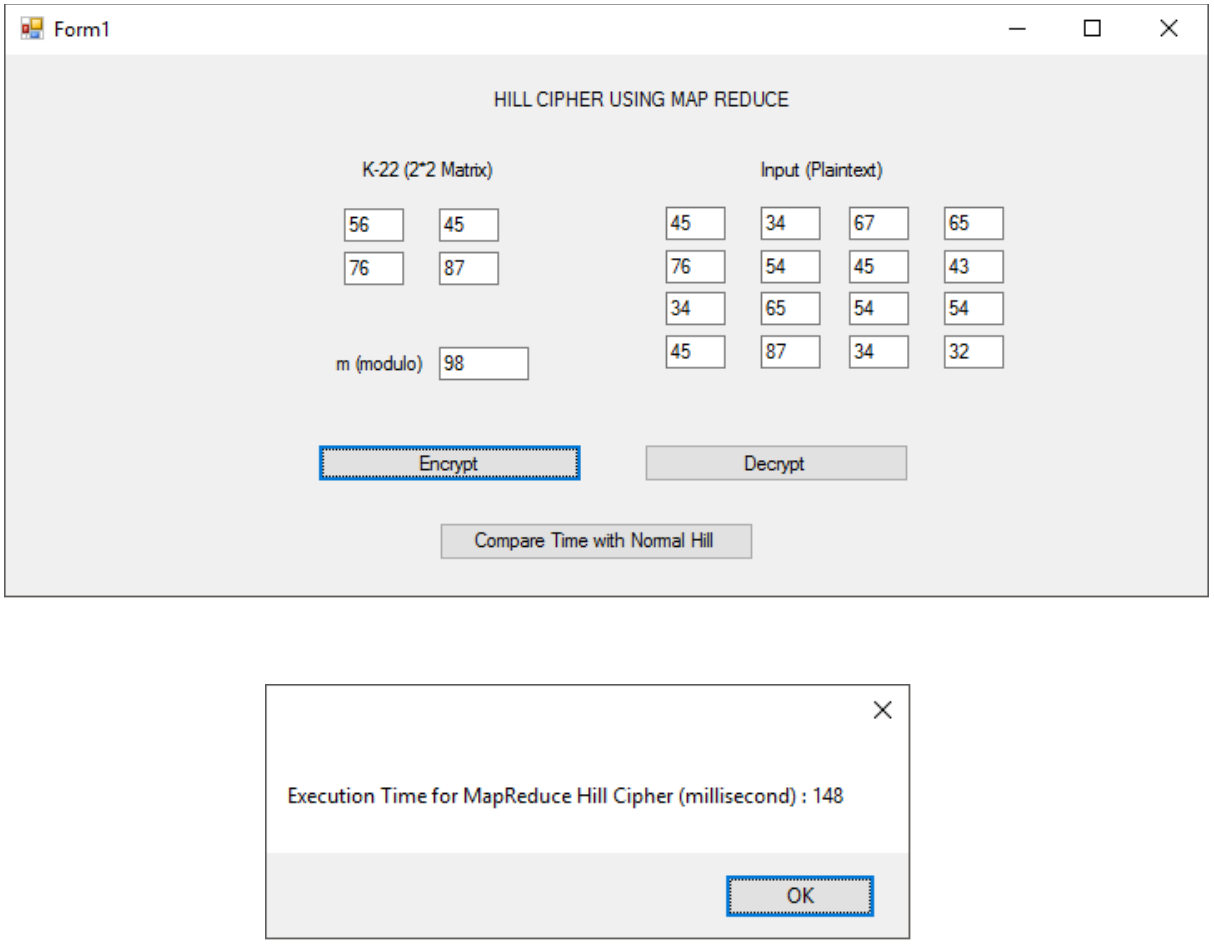

Figure 7: Executing the encryption cycle of Hill Cipher using the Parallel matrix multiplication on an i5 processor

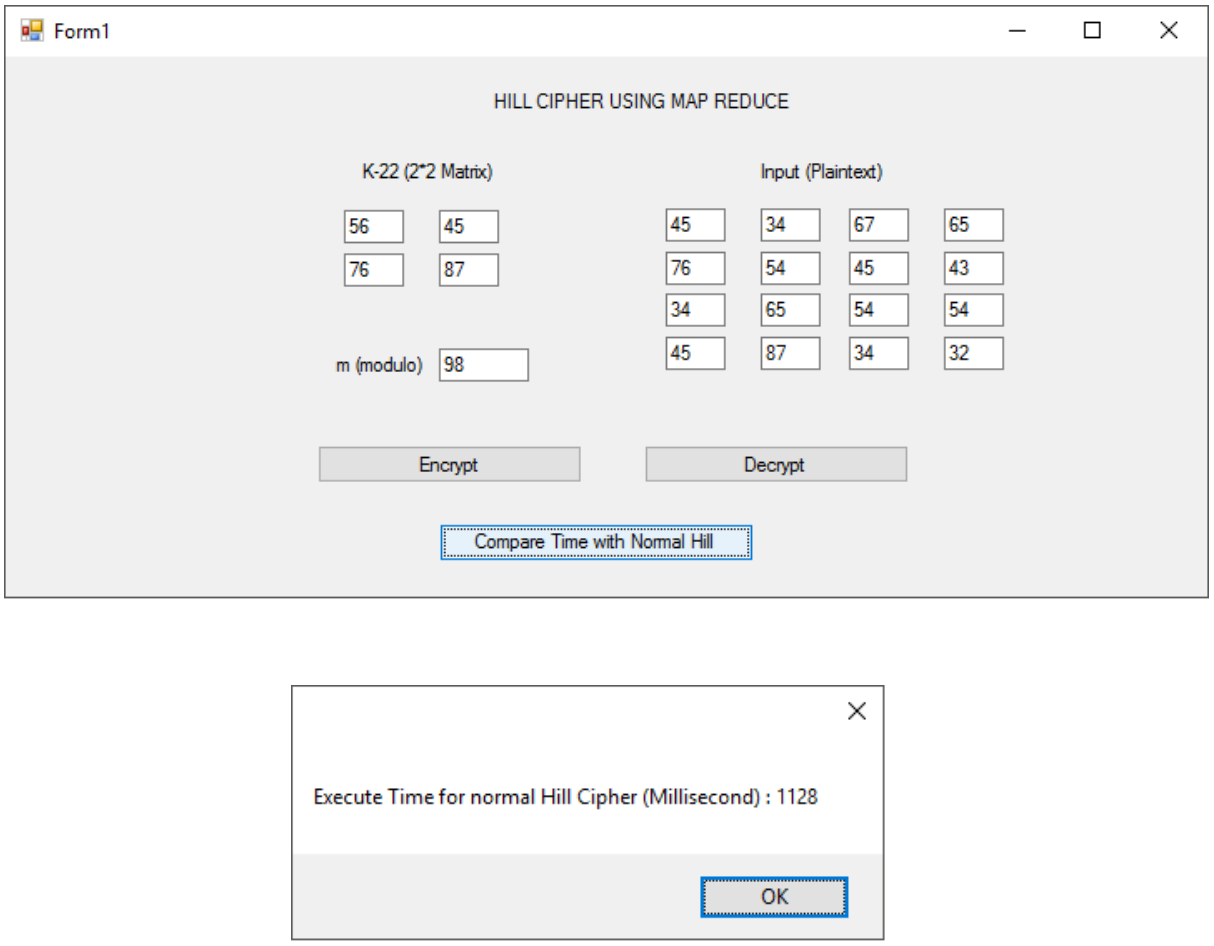

Figure 8: Executing the encryption cycle of Hill Cipher without Parallel Multiplication on an i5 processor

As we can see, the performance of the Parallel version of the Hill Cipher is still better than the classical version, even after executing it on a different processor.

Now, comparing the performances of the classical Hill Cipher vs the Parallel Hill Cipher on an i5 iMac processor:

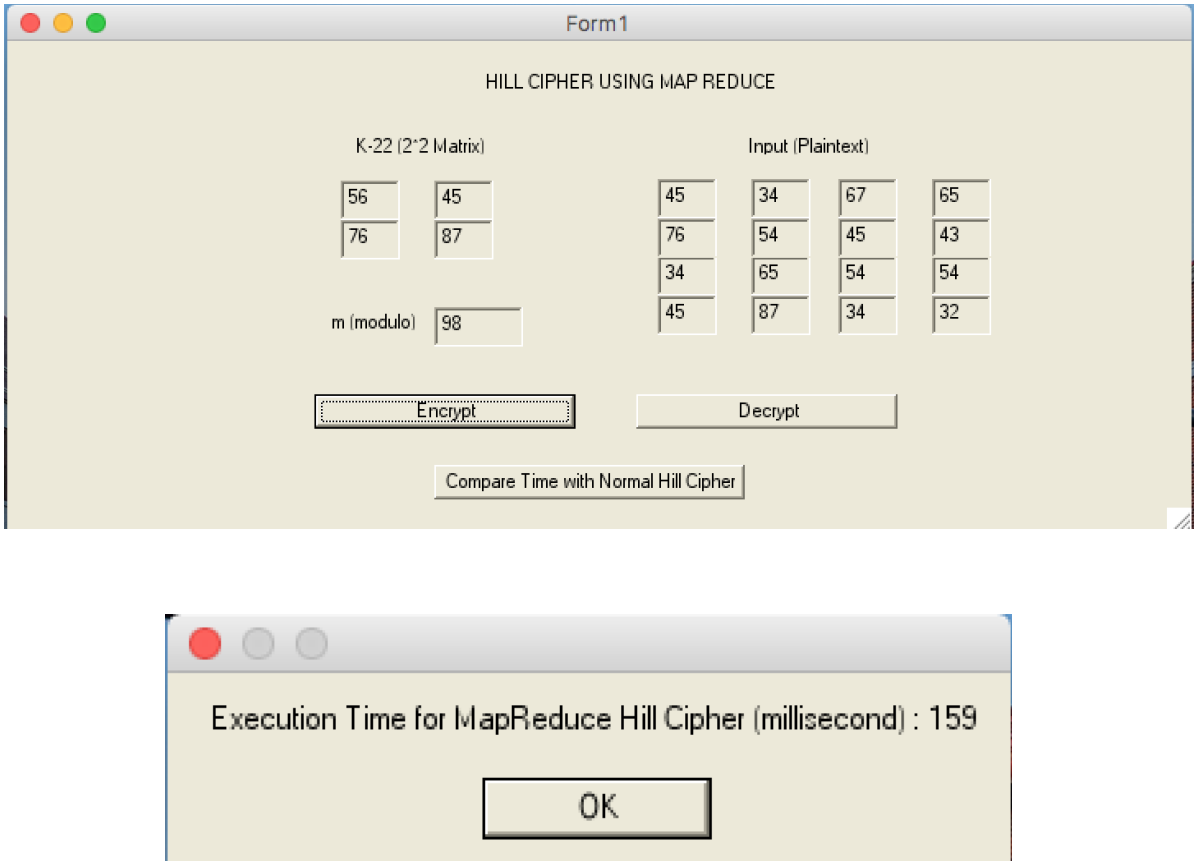

Figure 9: Executing the encryption cycle of Hill Cipher using the Parallel matrix multiplication on an i5 iMac processor

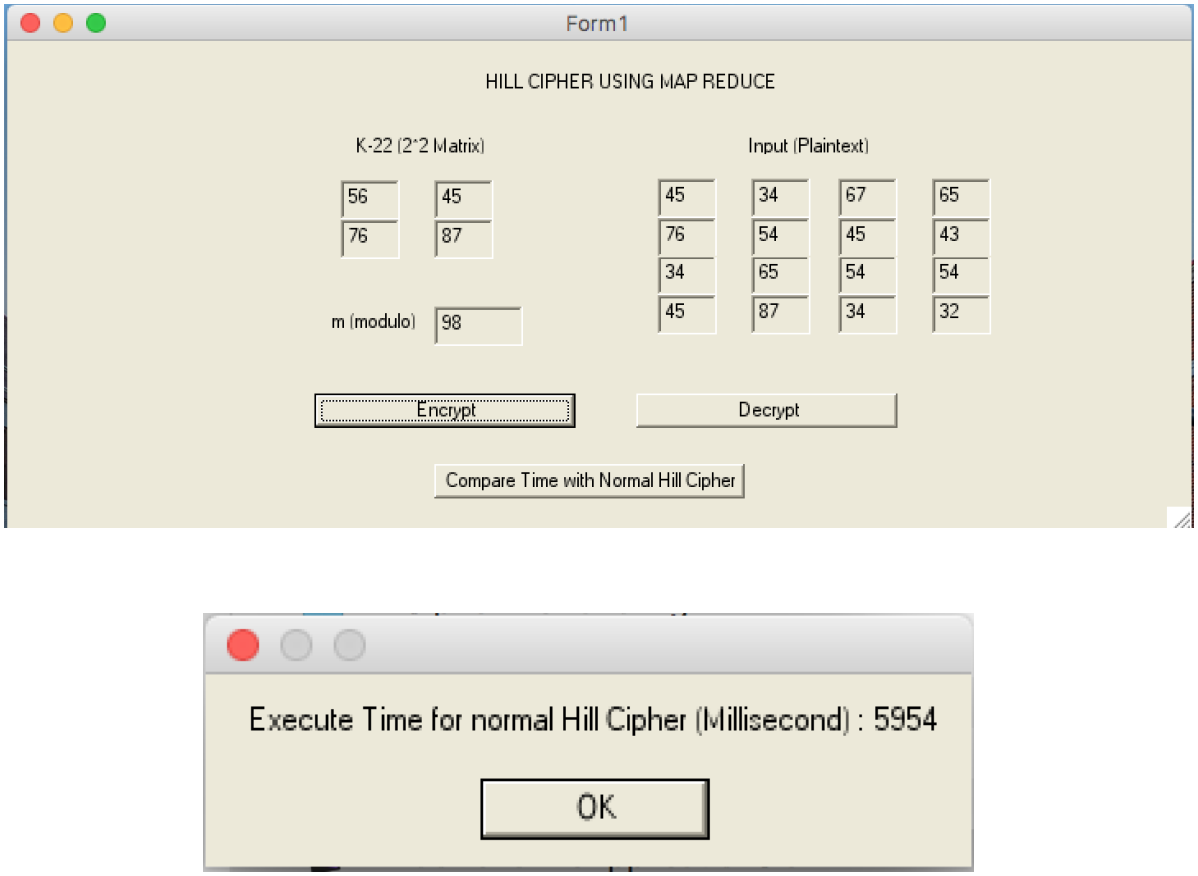

Figure 10: Executing the encryption cycle of Hill Cipher without Parallel Multiplication on an i5 iMac processor

As we can observe here too, the performance of the Parallel version of the Hill Cipher is still better than the classical version, even after executing it on a different processor.

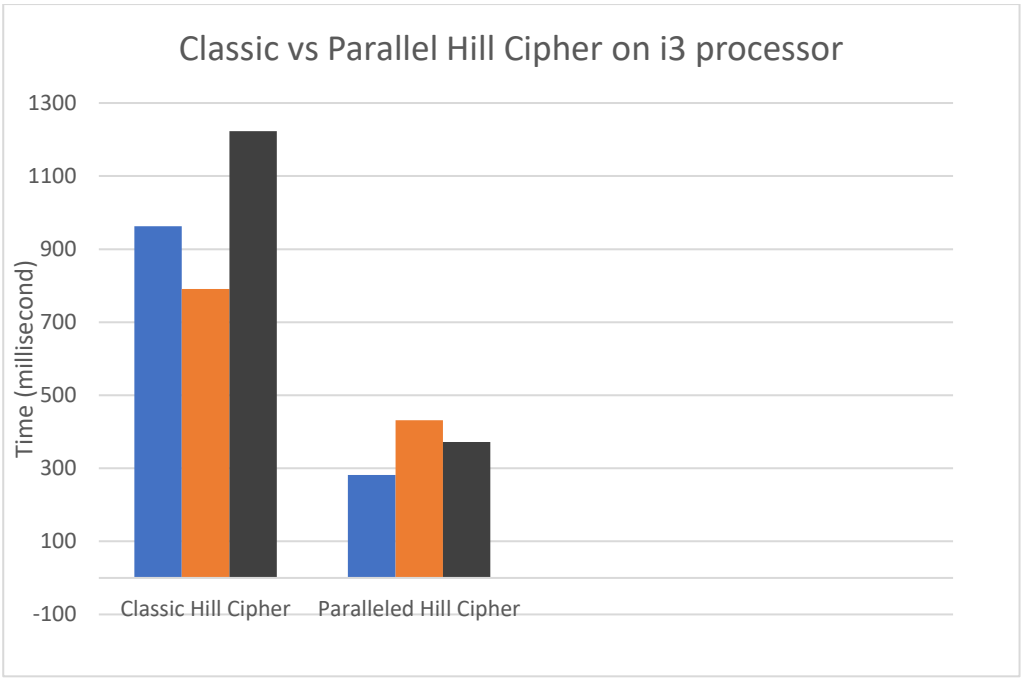

Figure 11: Comparing the performances of the Classic Hill Cipher vs its Parallel version on an i3 processor on a bar graph

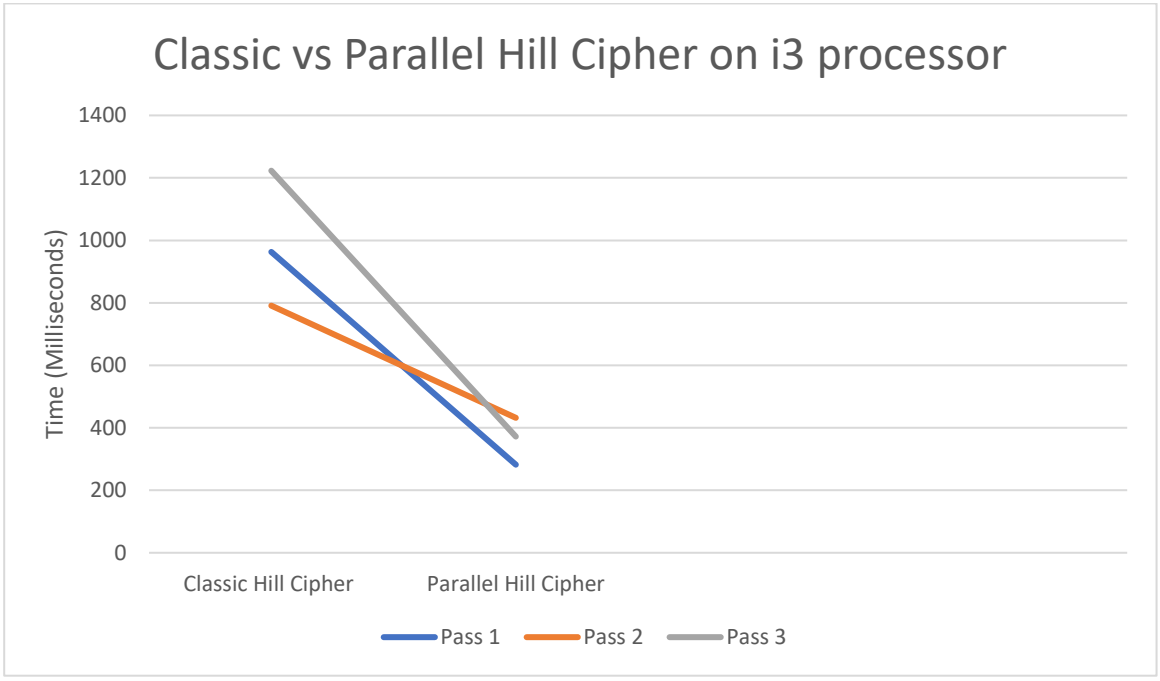

Figure 12: Comparing the performances of the Classic Hill Cipher vs its Parallel version on an i3 processor on a line graph

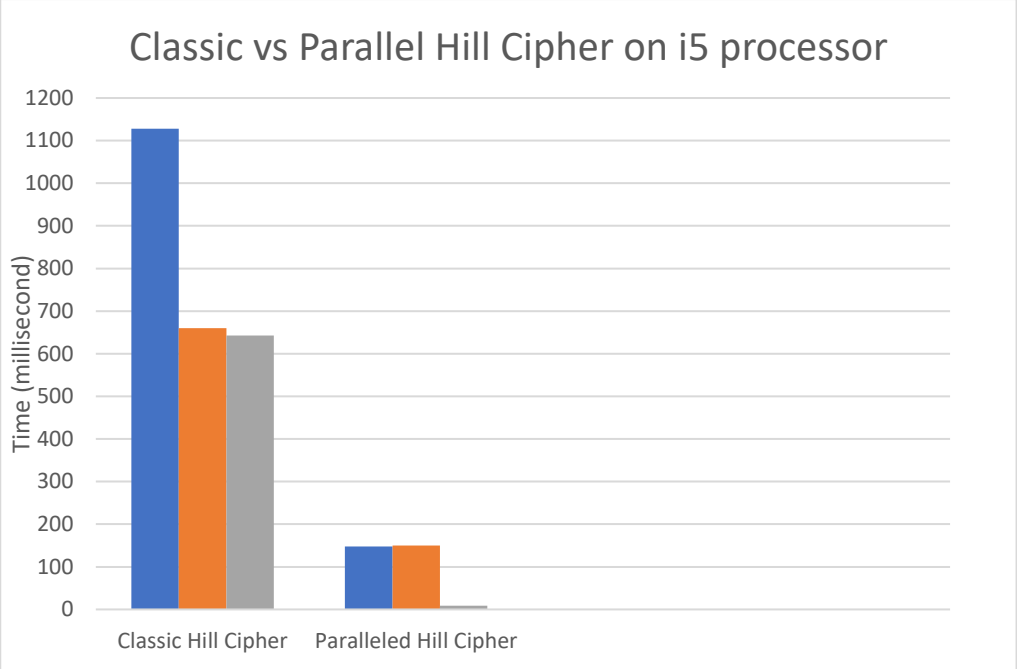

Figure 13: Comparing the performances of the Classic Hill Cipher vs its Parallel version on an i5 processor on a bar graph

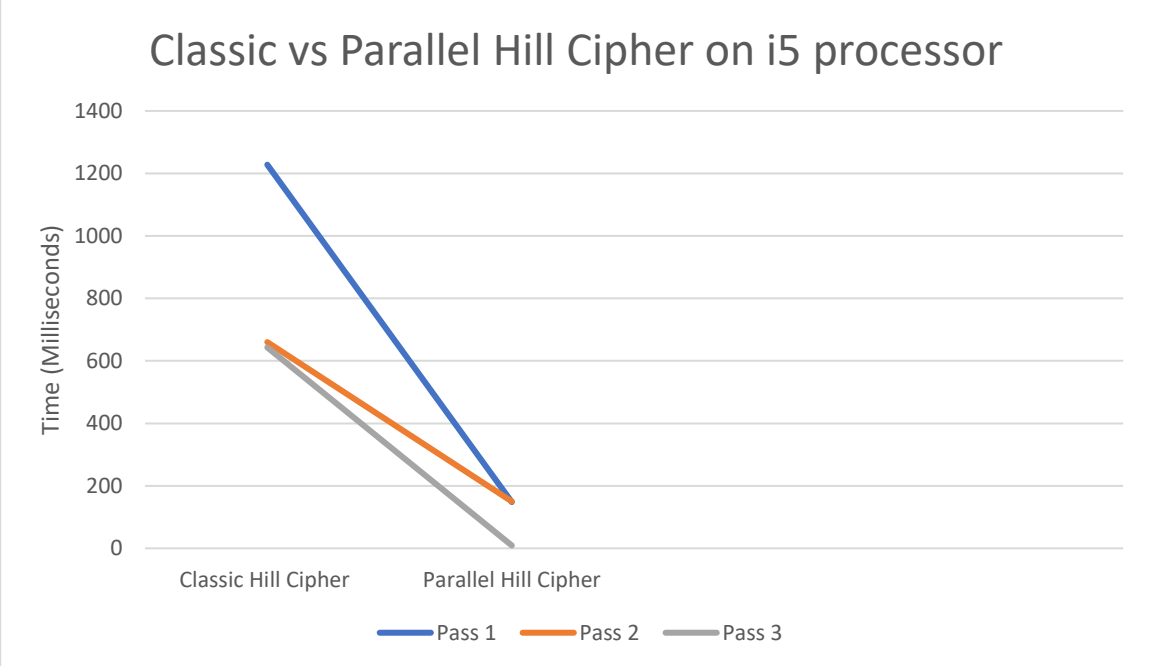

Figure 14: Comparing the performances of the Classic Hill Cipher vs its Parallel version on an i5 processor on a line graph

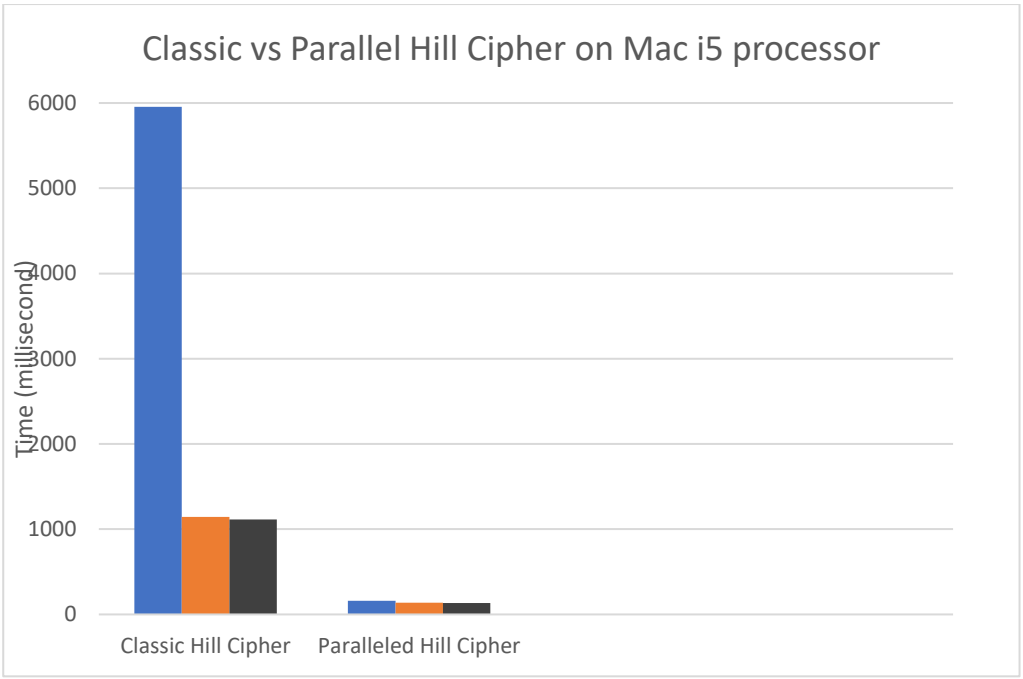

Figure 15: Comparing the performances of the Classic Hill Cipher vs its Parallel version on an i5 iMac processor on a bar graph

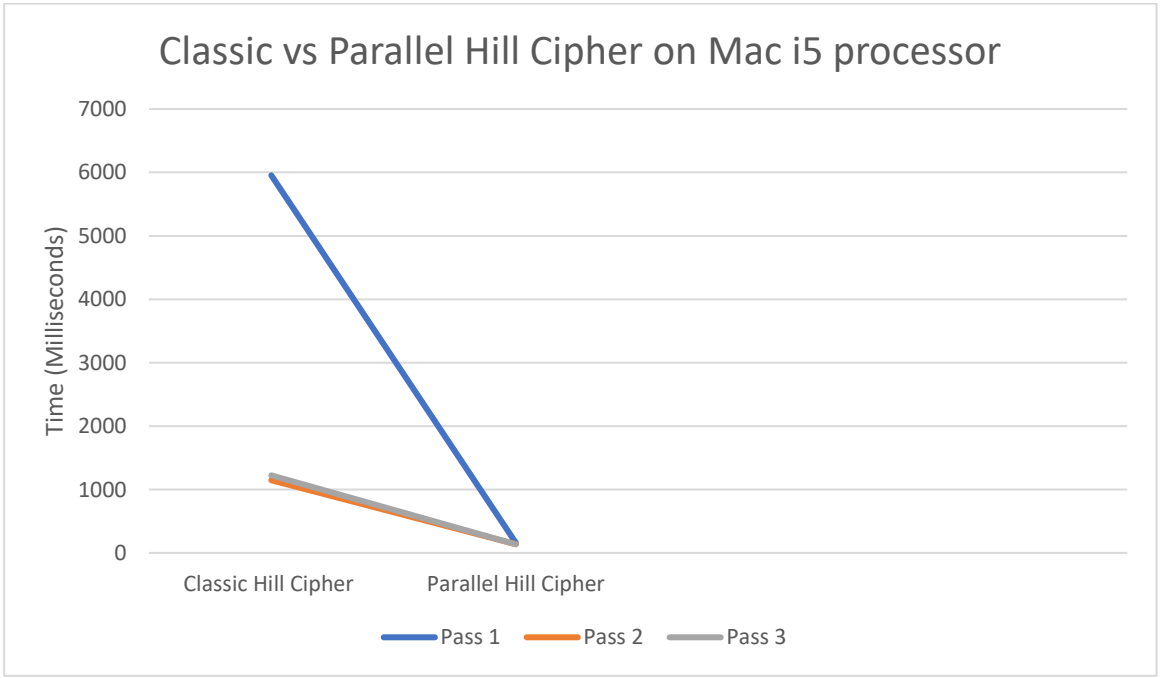

Figure 16: Comparing the performances of the Classic Hill Cipher vs its Parallel version on an i3 iMac processor on a line graph

5.2 REDUCING TIME COMPLEXITY

Processor	Time (milliseconds) for Normal Hill Cipher	Time (milliseconds) for Modified Hill Cipher
i3	963	282
i5	1128	148
iMac i5	5954	159

Table 4: Time-complexity comparison on different processors

As we can see from this table, the Modified Hill Cipher algorithm reduces the computational time quite significantly, when compared to the Normal Hill Cipher algorithm implementation. Therefore, it is proved that our implemented method for modification of the Hill Cipher algorithm is effective than the Normal Hill Cipher algorithm. Here, the clock ticks are used as a method of comparison of time in these methods because the time taken for computing the 4X4 matrix doesn't take any appreciable amount of time (not even in milliseconds, which is the shortest element of time that can be measured by the machine).

Chapter 6: CONCLUSION AND FUTURE WORK

This thesis shows the implementation of an improved version of the hill cipher using the MapReduce framework. The security of the implementation is further increased by using prime numbers to determine the key matrix using its four elements. This added security prevents any malicious characters and softwares to get hold of the key matrix, which is the crux of the whole cipher. The increased performance of the process depends upon the block matrix multiplication taking place in parallel, where each block is processed with a key matrix to generate a partial sum. The results from all the mappers is combined by a single reducer to generate the final cipher matrix. This process is effective in encrypting large databases.

An abundant amount of future work is expected for this technique. The current process of encryption is classic Hill Cipher scheme. The big security drawback of Hill Cipher cryptosystem is considered to be its endangerment to the attack of known-plaintext type, conditional to linear algebra. Exploring an advancement to remedy its security faults using parallel approach is essential stride for the future work. [1]

REFERENCES

[1] Wang, Xinyu, and Min, Zhaoe. “Parallel Algorithm for Hill cipher on MapReduce”: IEEE International Conference on Progress in Informatics and Computing (2014), pp. 493-497

[2] Toorani M, Falahati A. “A secure variant of the Hill cipher.” Computers and Communications, 2009. ISCC 2009. IEEE Symposium on. IEEE, 2009: pp.313-316.

[3] Keliher L, Delaney A Z. “Cryptanalysis of the Toorani-Falahati Hill Ciphers”22nd IEEE Symposium on Computers and Communications. IEEE Press, New York.

[4] Obimbo, Charlie, and Behzad Salami. “A Parallel Algorithm for determining the inverse of a matrix for use in block cipher encryption/decryption.” The Journal of Supercomputing 39.2 (2007): pp.113-130.

[5] Valiant, Leslie G. “A bridging model for parallel computation.” Communications of the ACM 33.8 (1990): pp.103-111.

[6] J. Choi, J. J. Dongarra and D. W. Walker, ‘PUMMA: Parallel universal matrix multiplication algorithms on distributed memory concurrent computers’, Concurrency, Pract. Exp., 6, (7),543–570 (1994).

[7] Jaeyoung Choi and Soongsil University, “A New Parallel Matrix Multiplication Algorithm on Distributed-Memory Concurrent Computers”, High Performance Computing on the Information Superhighway, 1997. HPC Asia '97, 224-229 (1997)

[8] R. C. Agarwal, F.G. Gustavson, and M. Zubair. “A High-Performance Matrix-Multiplication Algorithm on a Distributed-memory Parallel Computer Using Overlapped Communication.” IBM Journal of Research and Development, 38(6):673-68 1,1994.

[9] R. van de Geijn and J. Watts. “SUMMA Scalable Universal Matrix Multiplication Algorithm.” LAPACK Working Note 99, technical report, University of Tennessee, 1995.

[10] Anshul Gupta and Vipin Kumar, “Scalability of Parallel Algorithms for Matrix

Multiplication”, 1993 International Conference on Parallel Processing - ICPP'93, 115-123

[11] Dean, Jeffrey, and Sanjay Ghemawat. “MapReduce: simplified data processing on large clusters.” Communications of the ACM 51.1 (2008): 107-113. J.Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[12] Van De Geijn, Robert A., and Jerrell Watts. “SUMMA: Scalable universal matrix multiplication algorithm.”Concurrency-Practice and Experience 9.4 (1997): pp.255-274.

[13] Krishnan, Manojkumar, and Jarek Nieplocha. “SRUMMA: a matrix multiplication algorithm suitable for clusters and scalable shared memory systems.”Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International. IEEE, 2004