# PARALLEL COMPUTING OF AES ENCRYPTION ALGORITHM

**A PROJECT REPORT**

*Submitted by*

Tanwi Agrawal -19BCE2489

Shaina Agarwal - 19BCE2214

Neel Choksi - 19BCE0990

Course Code: CSE4001

Course Title: PARALLEL AND DISTRIBUTED COMPUTING

Under the guidance of

**Dr. S. Anto**

**Associate Professor, SCOPE,**

**VIT , Vellore.**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**April, 2022**

## ABSTRACT

The Advanced Encryption Standard is a block cipher cryptographic algorithm used in symmetric encryption systems. It contains 11 rounds of encryption of blocks of data . We will be parallelizing the internal functions performed during a round inspired by already available approaches and the 9 rounds itself using our own approach which includes utilizing the data and control dependency during most parts of the round and moving ahead in rounds based on that. We will be using the irregular parallelism functionality in OpenMP to implement our method. Evaluation metrics involve speedup times and execution times for individual parts of code parallelized and comparison with the serial time.

# INTRODUCTION

## 1. Introduction

### 1.1. AES Algorithm

The Advanced Encryption Standard (AES), also known as the Rijndael algorithm, is a symmetric block cipher approved by the NIST institute and used for the encryption of electronic data. AES is one of the most popular and widely adopted algorithms likely to be encountered nowadays in cryptographic systems. This level of security is reached by the application of a multitude of complex mathematical transformations on the input data.Thus, it requires a large amount of execution time which may not be feasible for some real time applications.

### 1.2. Existing solutions for this problem

There have been many approaches made to reduce the execution time by decreasing the number of rounds , by using double round key features[6]. Other ways of improving performance of AES includes using AES in Counter mode and Increasing the block size,fine grain approach where the operations performed on the block for encryption are parallelized, coarse grained approach where the process of input of huge data is converted into blocks and merged back after encryption is parallelized [2], both operations inside AES and distribution and aggregation of data can be parallelized, only one operation in AES can also be parallelized [3,4]. The fine grain approach to parallelize individual processes of AES involves multiple threads accessing the same cache block leading to false sharing where the data accessed by the thread is modified in between by another thread accessing the same block of data.

### 1.3. Our Approach to parallelize AES Algorithm

We will be attempting to parallelize the 9 Rounds of AES Encryption by identifying the data dependencies and the control level dependencies in the operations including Subbytes, Shiftrows, MixColumns, AddRoundKey. We will present a detailed analysis of the obtained results in terms of execution time and speed-up measurements.

# LITERATURE SURVEY

## 2. Literature Survey

### 2.1. Accelerating Encryption Algorithms Using Parallelism

The AES algorithm was used in CBC mode . Unlike CTR mode, in CBC mode the previous block's cipher text is used in the encryption of the next block as a result encryption of blocks is not independent.The authors proposed an interleaved solution of CBC mode by increasing the number of initialization vectors to run it in parallel. Two approaches were used to implement the ICBC mode in parallel including a thread reading the data and another threads encrypting the data. Another technique used was to set locks to all blocks before reading and unlock when reading of the block is complete. The free threads could be allotted for encryption when the block is unlocked. Issues faced were that Encryption threads were experiencing busy waiting to implement synchronization.[1]

### 2.2. Parallel Execution of AES-CTR Algorithm Using Extended Block Size

Solution in this paper was to increase the block size of plain text on which the encryption is applied.AES algorithm is used in the CTR mode since it is parallelizable inherently. A fine grained and a coarse grained approach are applied to compute the AES algorithm in parallel . In the fine grain approach , each of the step in the encryption function is parallelized. In the coarse grain approach multiple blocks are encrypted at the same time using threads. To distribute the work , overheads may occur as a result the author tried to increase the block size from 16 bytes to 1024 bytes. The new approach was significantly faster but the standard of 128 bit blocks was not maintained which may lead to a weaker encryption of the data. [2]

### 2.3. Detailed analysis of the AES CTR mode parallel execution using OpenMP

AES is used in Counter mode since it encrypts the blocks of data independently. AES includes Key Expansion , SubBytes, Shift Rows, Mix Columns, add round key steps. In this paper the authors have firstly taken a finest grain approach and parallelized each procedure in the encryption procedure. Another approach adopted was to only parallelize outer loops of the procedures.Running outer loop parallelism showed that only MixColumns procedure is the most resource intensive and can benefit from parallelism . A final approach taken by the authors involved parallelising only the mix columns procedure as a result they could achieve faster execution speeds.In the finest grain approach , nested loop parallelism has a negative effect and parallelising overheads are more in this approach. The authors did not take into account the other modes of AES algorithm ,neither did they focus on parallelising the blocks of data as a whole. [3]

## 2.4. Optimized AES algorithm using Galois field multiplication and parallel key scheduling

In this paper, sequential and parallel methods of key generation are analysed. Analysis reveals that sequential key generation is easier to implement, but utilizes larger resources. Another resource hungry area is mix column transformation. By using galois field multiplication compared to shift and add method in mix column transformation, it has been observed that area utilization on FPGA is reduced due to less gate count. This analysis suggests there is a larger scope for the improvement exists for add round key transform and overall AES implementation. [4]

## 2.5. An enhanced AES algorithm using cascading method on 400 bits key size used in enhancing the safety of next generation internet of things (IOT)

The proposed algorithm consists of diffusion [2] of AES algorithm with cascading technique which uses 200 bit plain text and 400 bit key which is divided into two equal parts in order to provide different keys at different cascading levels[4] which will Increase the security. AES-AES cascading is done which consists of 5 rounds instead of 10, thus it will omit the mix column twice from the original AES Encryption technique. Omitting mix column [3] single time reduces the time complexity. Hence, it takes half the time than a simple AES algorithm to encrypt the block data. The proposed algorithm saves encryption time upto 60% more than original as a result of removal of mixed column and increase security as a result of cascading and increased key size.[5]

## 2.6. Improving the throughput of the AES algorithm with multicore processors

In this paper, a sequential program is taken that implements the AES algorithm and convert the same to run on multicore architectures with minimum effort. They implemented two different parallel programmes, one with the fork system call in Linux and the other with the pthreads, the POSIX standard for threads. They analysed the throughputs between the implementations and among different architectures. The pthreads implementation outperformed in all the experiments conducted and the best throughput obtained was around 7Gbps on a 32-core processor (the largest number of cores) with the pthreads implementation.Performance has only slightly increased . [6]

## 2.7. Modification AES Algorithm based on Extended key and Plain text

This paper is referred by authors of paper [5].In this paper, a modification AES algorithm is presented within five proposals, the first modification is an extended plain text 4x4 for AES from 16 bytes to 64 bytes (8x8 array) this modification increases speed encryption, security, and complexity, the second modification is an extended keys by changing the key length used for encryption and decryption process from 176 bytes to 704 bytes because change input state involves key length, and change key length is a significant change. The third alteration is to raise the number of shifts in each row and the number of shifts based on part of the key, because the number of rows and columns in the state input has increased. Fourth, in the mix column stage, alter the static matrix to four different matrices used for multiple input states to boost data diffusion, and fifth, in the add round key stage, make it based on part of the key to reduce the number of sub keys used with input state in the XOR operation. The modifications procedure centred on how to increase the random sequence process within the algorithm by utilising several important components in the development process. [7]

## 2.8. Implementation Approaches for the Advanced Encryption Standard Algorithm

The AES algorithm is discussed in this study in terms of efficient hardware implementation methods. Hardware implementations give greater physical security and speed when compared to software implementations. Different uses of the AES algorithm may necessitate different trade-offs in terms of performance and space. Smart cards and cellular phones, for example, demand a modest amount of space. Other applications, such as WWW servers and ATMs, require a high level of performance. Other applications, such as digital video recorders, necessitate speed/area ratio optimization. [8]

### 2.9. Parallel AES algorithm for performance improvement in data analytics security for IoT

This paper proposes a method of injecting the high performance security algorithm in data analytics done with IOT-based devices. Parallel algorithms will improve the efficiency of security mechanisms in data analysis with parallel computing devices. AES algorithm is a symmetric encryption algorithm that works efficiently for hardware and software. Through parallel processing of the AES algorithm, data analytics in IoT-based systems performance can be improved. This method is tested with varieties of Intel-based multi-core processing architecture and considerable performance improvement is achieved. [9]

### 2.10. A Parallel AES Encryption Algorithms and Its Application

This paper presents a parallel AES encryption algorithm based on MapReduce architecture, which can be applied in a large-scale cluster environment. It can improve the efficiency of massive data encryption and decryption by parallelization. And the paper designs a parallel cipher block chaining mode to apply the AES algorithm. Experiments show that the proposed algorithm has good scalability and efficient performance, and can be applied to the security of massive data in a cloud computing environment. [10]

### 2.11. Parallelization of AES algorithm using OpenMP

This paper proposes an optimised parallel architecture of AES algorithm at both data and control level, suitable to be implemented in a multicore environment. The AES algorithm has been implemented in C language and is parallelised using OpenMP standard. The performance analysis is done using Intel VTune TM Amplifier XE 2013. The proposed parallel design exhibits improved performance over the sequential approach which addresses several applications that have a time constraint. [11]

### 2.12. Efficient Parallel Secure Outsourcing of Modular Exponentiation to Cloud for IoT Applications

In this paper a parallel secure outsourcing technique is used to provide the ability for modular exponentiation operations, which are used in IoT devices. After that, decompose the task of modular exponentiation and go over the strategy in further depth. In addition,  developing an RSA extension method based on this approach to provide increased security for IoT devices. Th e parallel modular exponentiation algorithm suggested by the authors involves computing the power in parallel based on the binary representation of the exponent due to non interdependency of bits of the exponents  . Finally, a comparison of experimental findings based on 512–4096 b of data shows that the current technique is preferable in terms of scalability and time consumption.[12]

## 2.13. Problem Definition

In this project we propose the parallelization of the AES algorithm in Electronic Codebook mode and show why it is parallelizable.We will compute the resource intensity of internal operations of AES including subbytes, shiftrows, mix columns and adding round key .Further we will try to parallelize the process of 9 rounds in AES-128 using OpenMP. To parallelise the process we will be trying to use the partial result of the block in the processing of the next round.

# OVERVIEW OF THE WORK

## 3. Overview of the Work

### 3.1. Objectives of the Project

The objective of the project is to encrypt the input data to cipher text using AES Algorithm in ECB mode , executing the algorithm in parallel using openMP, since the papers in literature review have suggested approaches to parallelize the functions Subbytes, Shiftrows, MixColumns and AddRoundKey, our main objective is to parallelize the 9 Rounds of AES performed on the block state of data by identifying the data and control dependencies and utilizing the Single Processor Multiple Data architecture of OpenMP.

3.2. Software Requirements

Operating system including Linux , Windows and Macintosh with g++ compiler . On the Windows system ,a subsystem of linux could be installed for the g++ compiler or mingw could be used.

3.3. Hardware Requirements

An intel or AMD processor computer or laptop with at least 2 cores and a hyperthreading of 4 is required for the project.

# SYSTEM DESIGN

## 4. System Design

### 4.1. AES Encryption Modes

Electronic Code Book (ECB) ECB describes the use of a symmetric cipher in its raw form, where each block is encrypted independently . The main drawback of ECB mode is that identical plaintext blocks are encrypted into the identical ciphertext blocks . Thus, data patterns in ECB mode are not hidden enough, which may lead to some vulnerabilities . ECB is the easiest and fastest mode to implement. It is the most common mode of AES algorithm used in commercial applications . The two formulae used by the ECB mode for encryption and decryption are as follows:

$C_i = E_k(P_i)$ [Encryption]

$P_i = D_k(C_i)$ [Decryption]

[14]

Cipher Block Chaining (CBC) The CBC mode solves the problem in ECB. It reduces the likelihood of appearing repeated patterns in the ciphertext . In CBC mode, before encrypting a block of plaintext, it is XORed with the previous ciphertext block. The first plaintext block is XORed with an initialization vector . The encryption and decryption in CBC are done as follows:

$C_i = E_k(P_i \oplus C_{i-1})$, with $C_0 = IV$ [Encryption]

$P_i = D_k(C_i) \oplus C_{i-1}$, with $C_0 = IV$ [Decryption

Large plaintext with possibly repeated patterns can be handled more securely in this mode . Thus, if the same plaintext is encrypted multiple times, the resulting ciphertexts are distinct due to the use of CBC mode requires more processing time than ECB mode due to its chaining mechanism . CBC can be synchronized to avoid channel noise error propagation. Unlike ECB, the CBC mode does not support parallelism, and therefore, it is not recommended for disk encryption .[14]

Cipher Feedback (CFB) In CFB mode, a plaintext block of length s bits is XORed with a portion of the output of the encryption module fed with some shift-register R. The encryption and decryption in CFB are done as follows:

$Ci = Pi \oplus Ms[Ek(Ri-1)]$, with $R0 = IV$ [Encryption]

$Pi = Ci \oplus Ms[Ek(Ri-1)]$, with $R0 = IV$ [Decryption]

Where Ms[Ek(Ri)] is the most significant s bits of the encrypted value of Ri . The process is repeated with the next input blocks until the ciphertext is created. Each consecutive input block is encrypted to generate an output block. The shiftregister is updated by: $Ri = Shift(Ri-1, Ci)$. Like CBC, in the CFB mode, the block of plaintext is based on the result of the previous ciphertext blocks. Therefore, multiple cipher operations cannot be carried out in parallel. The CFB mode does not require any padding to handle the plaintext that has a variable-length (not necessarily a multiple of the block size) . The CFB mode can be synchronized to avoid channel noise error propagation .[14]

Output Feedback (OFB) This mode is similar to CFB, but instead of using Ci to update the shift-register, we use the selected portion of the output of the encryption module to feedback the shift-register. First, the shift-register is initialized by a random initialization vector, and then it is updated by shifting the Ms[Ek(Ri)] bits for each block. Thus, the shift-register in OFB is updated as follows:

$Ri = Shift(Ri-1, Ms[Ek(Ri-1)])$

The encryption and decryption formulae for the OFB mode are similar to the ones for the CFB mode. Moreover, the OFB mode does not require any padding to handle the plaintext that has a variable-length. In the implementation of CFB and OFB's, only the encryption module is needed for their cryptographic applications . Unlike CFB where the ciphertext is the feedback, the feedback in OFB is taken directly from the encryption module . Thus, the XOR operation of each block of the plaintext in OFB is performed independently of both ciphertext and plaintext. Therefore, bit errors do not propagate in OFB mode during transmission.[14]

Counter Mode (CTR)

In the CTR mode, a counter R that starts in an initial value together with a nonce is used instead of a shift-register to generate a key stream. The key stream is XORed with the plaintext to generate the ciphertext. The counter is updated simply by: $R_i = R_{i-1} + 1$. The encryption and decryption in the CRT mode are as follows:

$C_i = P_i \oplus E_k(R_{i-1})$, with $R_0 = IV$ [Encryption]

$P_i = C_i \oplus E_k(R_{i-1})$, with $R_0 = IV$ [Decryption]

Like OFB and CFB modes, the CTR mode does not require any padding to match the block size of the cipher. Moreover, the encryption of a plaintext block does not depend on the result from previous blocks. Thus, CTR has become the mode of choice for high-speed applications due to its highly parallelizable architecture [1]. However, the CTR mode does not provide data integrity. [14]

## 4.2. AES Encryption ECB Mode

AES(Advanced Encryption Standard ) Algorithm is a block cipher. We have implemented the 128bit Electronic Codebook Mode of AES Algorithm .It divides the input plain text data into 16 byte blocks which are encrypted by application of the AES Algorithm functions including SubBytes, ShiftRows, MixColumns and AddRoundKey. AES is a symmetric key cryptosystem which means it uses the same key to encryption and decryption.The key used to encrypt a block of 16 byte is also of size 16 bytes. The 16 byte block is operated on for 11 rounds. 11 sets of keys are used to encrypt the block for 11 rounds, the initial key is expanded to 11 keys using KeyExpansion. One key is used in each round to encrypt the block state. In the first round , the initial key is added to the block state using XOR operation. In the subsequent 9 rounds, the block state is operated by SubBytes, ShiftRows, MixColumns, AddRoundKey functions. In the final round the MixColumns function is not applied on the block .

AddRoundKey is the operation used to add randomness to the data based on the key , each byte of the 16 byte key is added to the data block by using the xor operator since the operations done are inside the galois field.

SubBytes operation is used to replace each byte by another byte according to the key provided. AddRoundKey operation occurs before every SubBytes operation to randomize the substitution of bytes of the plain text. In the implementation , the bytes of the block state are replaced using a lookup table with 256 entries.

ShiftRows operation is used to shift the subblocks to the right . The rows are shifted by performing left shift operation , the first row is left shifted 0 times, the second row is left shifted 1 time, third row is left shifted 2 times and the fourth row is left shifted 3 times. The bytes that are shifted to the left reappear on the right.

MixColumns operation is used to mix the column elements among themselves, it is achieved by performing Galois field multiplication [4]. The implementation of galois field multiplication used is to multiply the state by a 4 by 4 matrix consisting the elements [2,3,1,1] in the first row , second row elements consists of one time right shifted first row elements , third row has elements of two times right shifted elements of row1 and the fourth row elements include three times right shifted elements of row1. The matrix multiplication is performed between columns of block state with rows of the multiplier. The multiplication is implemented using lookup tables for multiplication of block state elements with 2 and with 3. Since the matrix multiplication is in the galois field , the XOR operation is done instead of addition .

KeyExpansion operation is used to generate 11 pairs of keys from the key initially provided to the algorithm . The generation of next key is dependent on the previous key ,to generate the next key , the previous key is read , RCore operation is performed on it and the result of RCore operation is added to the previous using the XOR operation to generate the next key.The RCore operation involves left rotation of the 4 byte subblock , substitution of bytes of the blocks using the lookup table and raising the 2 to the power of iteration number -1 and adding to results using XOR operation. The 2 is raised and normalized using the galois field.

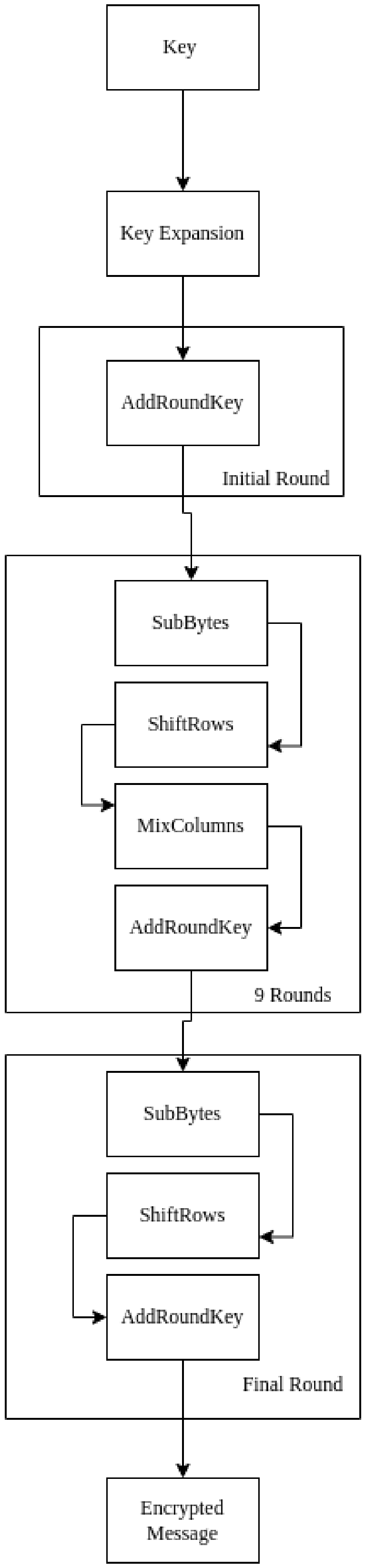


Figure 1. AES Encryption ECB Mode [2]

## 4.3. Highly Coarse grained approach (thread execution)

The input data can be of any size , to enable the algorithm to perform its operations, the data is padded to the nearest multiple of 16 since the algorithm inputs 16 byte blocks. The input data is divided into blocks of 16 bytes , in this coarse grained approach , 4 threads are tasked to encrypt each of the blocks by performing the 11 iterations of the algorithm and providing the cipher text. On completion of operation by the thread, it is assigned with another block to encrypt.

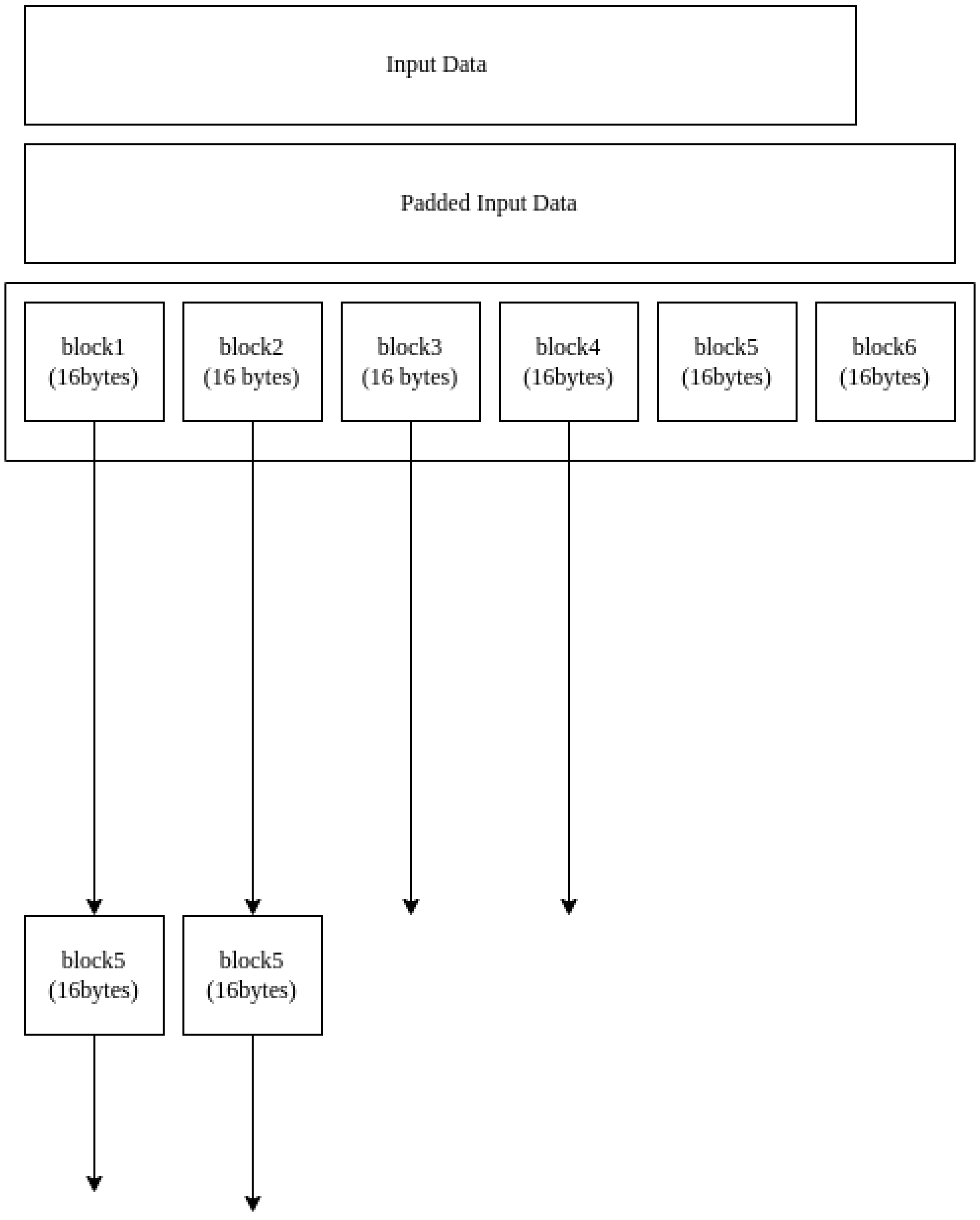


Figure 2. Parallel AES using Coarse Grained approach [2]

## 4.4. Fine grained approach (thread execution)

The 4 by 4 matrix of blocks represents the 16 bytes of input data to be encrypted . Figure 3. represents a single iteration of the 9 Rounds of AES algorithm. 4 threads are created to perform SubBytes, ShiftRows, MixColumns and AddRoundKey . Since the input of each operation is dependent on the output of the previous operation , barriers are applied at the end of each iteration.

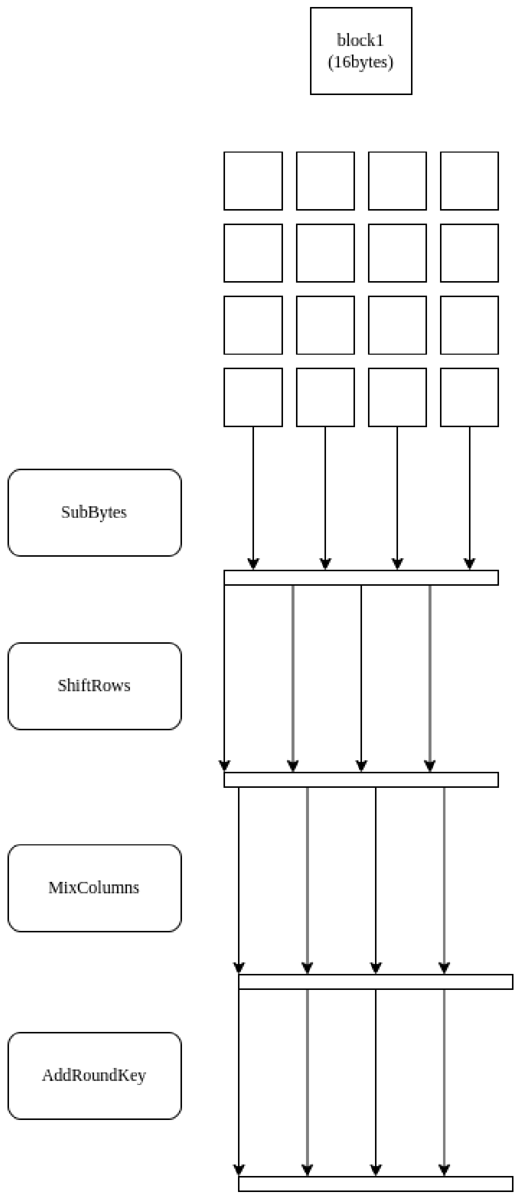


Figure 3. Parallel AES using Fine Grained approach [3]

## 4.5. Better fine grained approach (thread execution)

The 4 by 4 matrix of blocks represents the 16 bytes of input data to be encrypted . Figure 3. represents a single iteration of the 9 Rounds of AES algorithm. Since the operations include table lookups and bitwise XOR operations , the threads can complete their tasks at a faster rate, therefore the cost of parallelization was dominating the time taken by the threads to execute in parallel . In this approach , two threads are given 8 subblocks at a time to perform operations which increases the time of execution of threads , hence balancing the time to parallelise and cost of parallelization tradeoff.

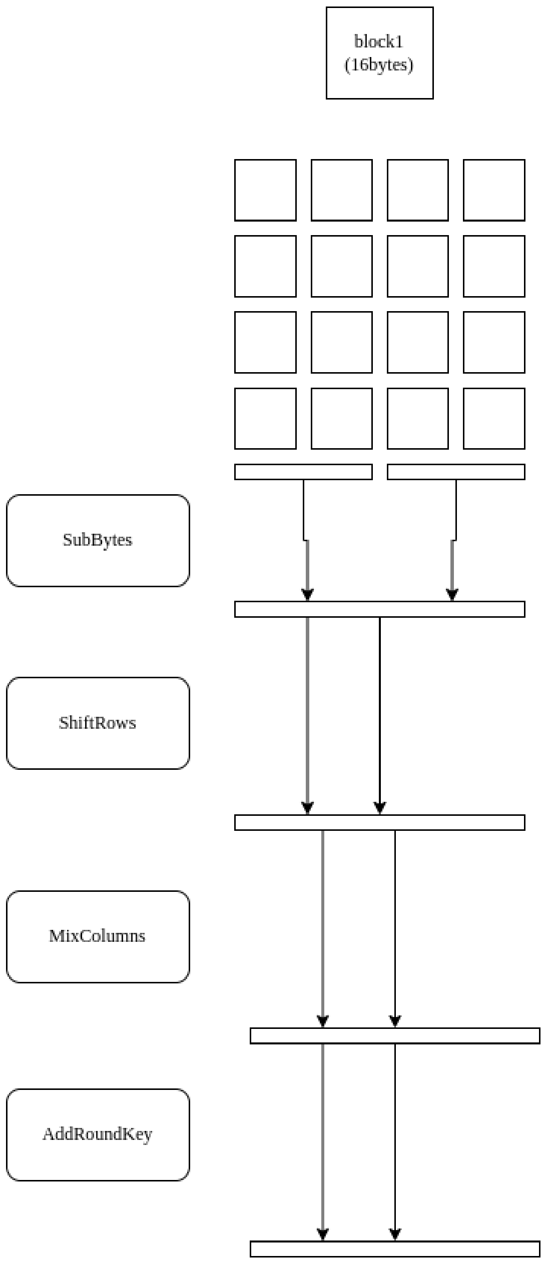


Figure 4. Parallel AES using Better Fine Grained approach [3]

## 4.6. Rounds parallelized , moving ahead for iterations

In this approach , the 4 threads are allotted 4 subblocks each to encrypt. The threads apply operations to encrypt the 4 blocks and create subthreads to divide the work within the operations, for 9 iterations each. Each of the 4 threads is independent of the other and can be at any iteration at any point of time. On evaluating the results, we realized the flaw in this approach ,which is that the ShiftRows operation requires the block state data from other threads being at the same iteration , this control dependency lead us to the next approach of irregular parallelism .

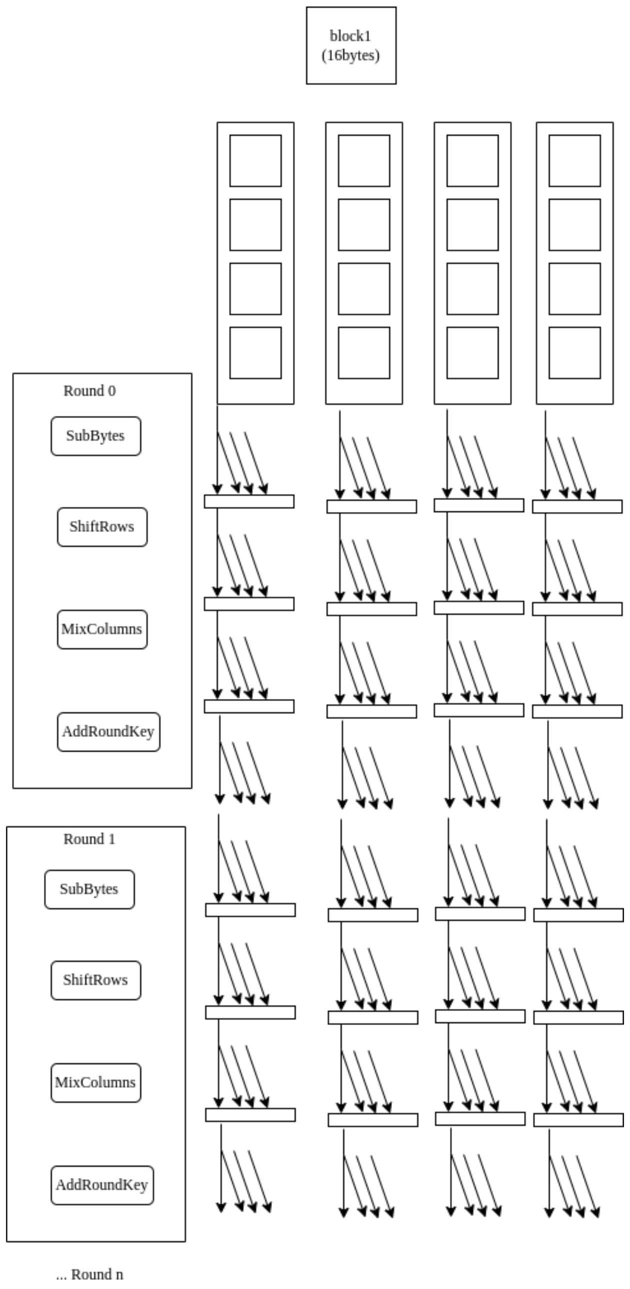


Figure 5. 9 Rounds in AES Parallelized.

## 4.7. 9 Rounds parallelized , moving ahead for iterations without losing correctness of the algorithm

This approach uses irregular parallelism which allows threads to execute specific parts of code as tasks , independent from the other threads. The thread starts to perform the SubBytes operation on the input data for round0 . The thread stops at the barrier before starting ShiftRows since the ShiftRows operation requires the data from other columns of blocks and the 4 threads are performing operations on 4 columns of block , simultaneously. ShiftRows operation is performed by a single thread at a time , no other threads are executed while that thread is executing since the ShiftRows operation uses the entire previous block state and the data that it accesses is not independent. After completion of the ShiftRows, MixColumns , AddRoundKey and SubBytes from the next iteration are performed by 4 threads in parallel , each thread is assigned with a column of data consisting of 4 sub blocks which are not dependent on each other for execution of the three operations. The 4 threads are stopped by the barrier again before the ShiftRows operation of the next iteration begins. This way the data dependency and the control dependency identified in the previous approach is utilized to execute the maximum part of the round in parallel.

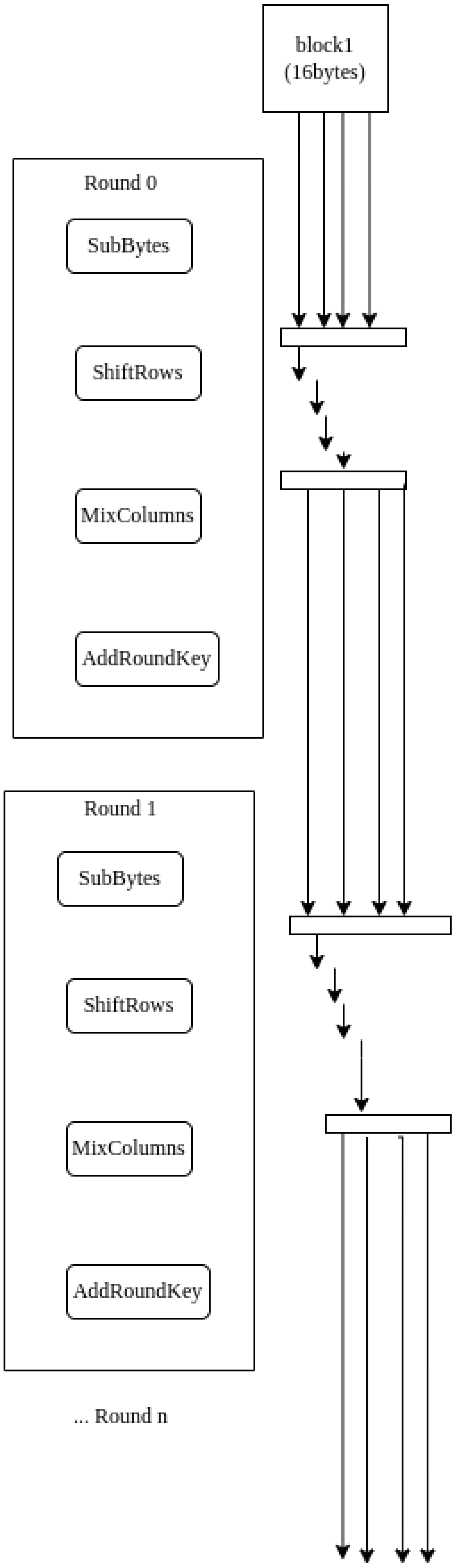

block1
(16bytes)
Round 0
SubBytes
ShiftRows
MixColumns
AddRoundKey
Round 1
SubBytes
ShiftRows
MixColumns
AddRoundKey
... Round n

Figure 6. 9 Rounds in AES Parallelized maintaining correctness.

# IMPLEMENTATION

## 5. Implementation

### 5.1. Description of Modules/Programs

#### 5.1.1. 9 rounds moving ahead in rounds

The AESEncrypt() function is called to encrypt the 16 byte message , and an expanded key is provided to it . The block state of 16 bytes is defined and initialized with the values of the 16 byte message. The round key is added to the block and 4 threads are set. The parallel block of code is started, each thread is given its independent execution of 9 iterations with its indices range for 4 subblocks column wise. The set of 4 subblocks is updated by each thread for 9 iterations by performing Subbytes , ShiftRows, MixColumns and AddRoundKey operations. SubBytes, ShiftRows, AddRoundKey are also performed in parallel by spawning child threads and allocating subblocks to them for operation. This approach is incorrect due to the data dependency of the ShiftRows operation for the data from other columns in the same iteration, since the threads are independent, they are in different iterations when the ShiftRow is performed. Although the approach was incorrect, we could discover that the operations AddRoundKey , MixColumnns and SubBytes can be performed independently from each other, MixColumn needs all the 4 blocks of the column under it to execute in parallel while maintaining the correctness.

#### 5.1.2. 9 rounds moving ahead in rounds , maintaining correctness

In this approach , instead of moving ahead in rounds by execution of threads independently , the threads are collapsed during the ShiftRows operation since it was found from the earlier approach that the shiftrows operation needs the data from other columns in the same iteration . Here there are 4 threads spawned in the beginning of the parallel region , each thread is given 4 subblocks of data for performing the Subbytes operation in the form of tasks since irregular parallelism is utilized.These threads collapse at the barrier before ShiftRows operation begins, this barrier is achieved using the taskwait . Only one thread at a time performs the ShiftRows operation, the taskwait barrier ensures that the output of shiftrows is completed before the next operation is started. The subsequent operations are done by 4 threads in parallel , each thread is allotted 4 subblocks, utilizing the independence of 4 subblocks from each other while performing MixColumns, AddRoundsKey and SubBytes of the next iteration. The three operations are independent of each other but the MixColumns operation is dependent on contiguous data of 4 sub blocks in the same column therefore 4 subblocks are allotted to a single thread.

#### 5.1.3. Highly Coarse grained approach

The data on input from the user is padded to a multiple of 16 bytes for even splitting of input data into blocks. Four threads are spawned and one block is provided to each thread at a time using the static allocation. Once the thread completes the execution of the block , it is allotted another block . The thread performs the entire 11 rounds of encryption on the block allocated to it. In this manner , multiple blocks are encrypted at the same time to speed up the encryption process of the entire input data.

#### 5.1.4. Fine grained approach

The SubBytes operation involves substitution of the subblock of data with the corresponding value in the lookup table which is prebuilt. Four threads are spawned and each thread is allotted 4 subblocks, the thread substitutes the bytes in the allotted blocks and waits at the barrier for the other threads to complete.

The ShiftRows operation is implemented by copying the state indexes to the corresponding shifted indexes to a temporary state, this is done by providing 4 threads each 4 subblocks of the state and it updates the temporary state, after the completion of the threads, the temporary state is copied to the state in serial manner.

MixColumns involves the state columns to be multiplied by a 4 by 4 matrix . 4 threads are allotted with 1 column of state each and the thread calculates the final output state element by multiplying the column elements with the corresponding element in the 4 by 4 matrix . The output is stored in a temporary state since a column is multiplied 4 times to generate 4 elements in the output state. On completion of execution of all 4 threads, the temporary state matrix is copied to the state in a serial manner.

AddRoundKey operation involves adding a key of 16 bytes to 16 bytes of block .Since the data is divided into sub blocks of 1byte each , four threads are allotted a set of 4 blocks and tasked to perform XOR addition with the key which is also divided into 16 subblocks . The thread adds all its subblocks with the corresponding subkeys and waits for other threads to complete their execution .

### 5.1.5. Better fine grained approach

The SubBytes operation involves substitution of the subblock of data with the corresponding value in the lookup table which is prebuilt. Two threads are spawned and each thread is allotted 8 subblocks, the thread substitutes the bytes in the allotted blocks and waits at the barrier for the other threads to complete.

The ShiftRows operation is implemented by copying the state indexes to the corresponding shifted indexes to a temporary state, this is done by providing 2 threads each 8 subblocks of the state and it updates the temporary state, after the completion of the threads, the temporary state is copied to the state in serial manner.

MixColumns involves the state columns to be multiplied by a 4 by 4 matrix . 2 threads are allotted with 2 columns of state each and the thread calculates the final output state element by multiplying the column elements with the corresponding element in the 4 by 4 matrix . The output is stored in a temporary state since a column is multiplied 4 times to generate 4 elements in the output state. On completion of execution of all 2 threads, the temporary state matrix is copied to the state in a serial manner.

AddRoundKey operation involves adding a key of 16 bytes to 16 bytes of block .Since the data is divided into sub blocks of 1byte each , two threads are allotted a set of 8 blocks and tasked to perform XOR addition with the key which is also divided into 16 subblocks . The thread adds all its subblocks with the corresponding subkeys and waits for other threads to complete their execution .

## 5.2. Source Code

Only the part of the source code involving the parallelism is shown here; the entire source code is deployed on the github repository mentioned in Appendix - A.

### 5.2.1. 9 rounds moving ahead in rounds

```
void PartialRound(int start,int end,unsigned char* state, unsigned char* tmp, unsigned char* temp, unsigned char* key){

        int temp_start = start;
        int o_temp_start = start;
        #pragma omp for schedule(static,4)
        for(int i=start;i<end;i++){
                state[i] = s[state[i]];
        }

        int j = o_temp_start;
        #pragma omp for schedule(static,4)
        for(int i = temp_start;i<end;i++){
                tmp[i] = state[j];
                j=j+5;
        }

        #pragma omp for schedule(static,4)
        for (int i = start; i < end; i++) {
                state[i] = tmp[i];
        }


        temp[temp_start++] = (unsigned char) mul2[state[o_temp_start++]] ^ mul3[state[o_temp_start++]] ^ state[o_temp_start++] ^ state[o_temp_start];
        o_temp_start = start;
```

```
	temp[temp_start++] = (unsigned char) state[o_temp_start++] ^ mul2[state[o_temp_start++]] ^ mul3[state[o_temp_start++]] ^ state[o_temp_start];
	o_temp_start = start;
	temp[temp_start++] = (unsigned char) state[o_temp_start++] ^ state[o_temp_start++] ^ mul2[state[o_temp_start++]] ^ mul3[state[o_temp_start]];
	o_temp_start = start;
	temp[temp_start] = (unsigned char) mul3[state[o_temp_start++]] ^ state[o_temp_start++] ^ state[o_temp_start++] ^ mul2[state[o_temp_start]];

	#pragma omp for schedule(static,4)
	for (int i = start; i < end; i++) {
		state[i] = temp[i];
	}

	#pragma omp for schedule(static,4)
	for (int i = start; i < end; i++) {
		state[i] = state[i] ^ key[i];
	}
}

void AESEncrypt(unsigned char *message, unsigned char *expandedKey, unsigned char *encryptedMessage) {
	unsigned char state[16]; // Stores the first 16 bytes of original message

	for (int i = 0; i < 16; i++) {
		state[i] = message[i];
	}

	int numberOfRounds = 9;

	AddRoundKey(state, expandedKey); // Initial round
```

```
          // Round(state, expandedKey + (16 * (i+1)));
// 9 rounds :
omp_set_num_threads(4);

double tdata = omp_get_wtime();
unsigned char* key ;
#pragma omp parallel
{

          int id;
          unsigned char tmp[16];
          unsigned char temp[16];
          id = omp_get_thread_num();

          #pragma omp sections
          {
                    #pragma omp section
                    for(int i=0;i<numberOfRounds; i++){
                              // state 0 --> 3
                              key = expandedKey + (16 * (i+1));
                              PartialRound(0,4,state,tmp,temp,key);

                    }

                    #pragma omp section
                    for(int i=0;i<numberOfRounds; i++){
                              // state 4 --> 7
                              key = expandedKey + (16 * (i+1));

                              PartialRound(4,8,state,tmp,temp,key);
```

```
        }

        #pragma omp section
        for(int i=0;i<numberOfRounds; i++){
                // state 8 --> 11
                key = expandedKey + (16 * (i+1));

                PartialRound(8,12,state,tmp,temp,key);

        }

        #pragma omp section
        for(int i=0;i<numberOfRounds; i++){
                // state 12 --> 15
                key = expandedKey + (16 * (i+1));

                PartialRound(12,16,state,tmp,temp,key);

        }
}
}

tdata = omp_get_wtime()-tdata;
nineRounds +=tdata;

FinalRound(state, expandedKey + 160);

// Copy encrypted state to buffer
for (int i = 0; i < 16; i++) {
        encryptedMessage[i] = state[i];
```

```
	}
}
```

5.2.2. 9 rounds moving ahead in rounds , maintaining correctness

```
void AESEncrypt(unsigned char *message, unsigned char *expandedKey, unsigned char
*encryptedMessage) {
	unsigned char state[16]; // Stores the first 16 bytes of original message

	for (int i = 0; i < 16; i++) {
		state[i] = message[i];
	}

	int numberOfRounds = 9;

	AddRoundKey(state, expandedKey); // Initial round

// 9 rounds :
omp_set_num_threads(4);




double tdata = omp_get_wtime();
  #pragma omp parallel
  {
  int id;
  unsigned char tmp[16];
  unsigned char temp[16];
	// #pragma omp parallel for nowait
	for (int i = 0; i < numberOfRounds; i++) {

		// Round(state, expandedKey + (16 * (i+1)));
```

```
                unsigned char* key = expandedKey + (16 * (i+1));
                        id = omp_get_thread_num();
                        // SubBytes(state);
                        // for (int i = 0; i < 16; i++) {
                        //          state[i] = s[state[i]];
                        // }

    #pragma omp single nowait
    {
      #pragma omp task shared(state,key)
      {
        //subbytes:

        if(id==0){
          state[0] = s[state[0]];
          state[1] = s[state[1]];
          state[2] = s[state[2]];
          state[3] = s[state[3]];



        }else if(id==1){
          state[4] = s[state[4]];
          state[5] = s[state[5]];
          state[6] = s[state[6]];
          state[7] = s[state[7]];

        }else if(id==2){
          state[8] = s[state[8]];
          state[9] = s[state[9]];
          state[10] = s[state[10]];
          state[11] = s[state[11]];
```

```
 }else if(id==3){
   state[12] = s[state[12]];
   state[13] = s[state[13]];
   state[14] = s[state[14]];
   state[15] = s[state[15]];

 }
}

#pragma omp taskwait

#pragma omp task shared(state,tmp)
{


 // shift rows :

   tmp[0] = state[0];
   tmp[13] = state[1];
   tmp[10] = state[2];
   tmp[7] = state[3];


 /* Column 2 */

   tmp[4] = state[4];
   tmp[1] = state[5];
   tmp[14] = state[6];
   tmp[11] = state[7];
```

```
  /* Column 3 */

    tmp[8] = state[8];
    tmp[5] = state[9];
    tmp[2] = state[10];
    tmp[15] = state[11];

  /* Column 4 */

    tmp[12] = state[12];
    tmp[9] = state[13];
    tmp[6] = state[14];
    tmp[3] = state[15];

    for (int i = 0; i < 16; i++) {
                                    state[i] = tmp[i];
                        }



}

#pragma omp taskwait

#pragma omp task shared(state,temp)
{
  //mix columns
  if(id==0){

    temp[0] = (unsigned char) mul2[state[0]] ^ mul3[state[1]] ^ state[2] ^ state[3];
    temp[1] = (unsigned char) state[0] ^ mul2[state[1]] ^ mul3[state[2]] ^ state[3];
    temp[2] = (unsigned char) state[0] ^ state[1] ^ mul2[state[2]] ^ mul3[state[3]];
    temp[3] = (unsigned char) mul3[state[0]] ^ state[1] ^ state[2] ^ mul2[state[3]];

    state[0] = temp[0];
    state[1] = temp[1];
    state[2] = temp[2];
    state[3] = temp[3];

  }else if(id==1){

    temp[4] = (unsigned char)mul2[state[4]] ^ mul3[state[5]] ^ state[6] ^ state[7];
    temp[5] = (unsigned char)state[4] ^ mul2[state[5]] ^ mul3[state[6]] ^ state[7];
    temp[6] = (unsigned char)state[4] ^ state[5] ^ mul2[state[6]] ^ mul3[state[7]];
    temp[7] = (unsigned char)mul3[state[4]] ^ state[5] ^ state[6] ^ mul2[state[7]];
    state[4] = temp[4];
    state[5] = temp[5];
    state[6] = temp[6];
    state[7] = temp[7];

  }else if(id==2){
    temp[8] = (unsigned char)mul2[state[8]] ^ mul3[state[9]] ^ state[10] ^ state[11];
    temp[9] = (unsigned char)state[8] ^ mul2[state[9]] ^ mul3[state[10]] ^ state[11];
    temp[10] = (unsigned char)state[8] ^ state[9] ^ mul2[state[10]] ^ mul3[state[11]];
    temp[11] = (unsigned char)mul3[state[8]] ^ state[9] ^ state[10] ^ mul2[state[11]];
    state[8] = temp[8];
    state[9] = temp[9];
    state[10] = temp[10];
    state[11] = temp[11];

  }else if(id==3){
    temp[12] = (unsigned char)mul2[state[12]] ^ mul3[state[13]] ^ state[14] ^ state[15];
    temp[13] = (unsigned char)state[12] ^ mul2[state[13]] ^ mul3[state[14]] ^ state[15];
    temp[14] = (unsigned char)state[12] ^ state[13] ^ mul2[state[14]] ^ mul3[state[15]];
    temp[15] = (unsigned char)mul3[state[12]] ^ state[13] ^ state[14] ^ mul2[state[15]];
    state[12] = temp[12];
    state[13] = temp[13];
    state[14] = temp[14];
    state[15] = temp[15];

  }



  // AddRoundKey(state, key);
    if(id==0){
      state[0] = state[0] ^ key[0];
      state[1] = state[1] ^ key[1];
      state[2] = state[2] ^ key[2];
      state[3] = state[3] ^ key[3];



    }else if(id==1){
      state[4] = state[4] ^ key[4];
      state[5] = state[5] ^ key[5];
      state[6] = state[6] ^ key[6];
      state[7] = state[7] ^ key[7];

    }else if(id ==2){
      state[8] = state[8] ^ key[8];
      state[9] = state[9] ^ key[9];
      state[10] = state[10] ^ key[10];
```

```
      state[11] = state[11] ^ key[11];

    }else if(id ==3){
      state[12] = state[12] ^ key[12];
      state[13] = state[13] ^ key[13];
      state[14] = state[14] ^ key[14];
      state[15] = state[15] ^ key[15];

    }
  }
  #pragma omp taskwait

 }
          }

    }
1
    tdata = omp_get_wtime()-tdata;
    nineRounds +=tdata;

    FinalRound(state, expandedKey + 160);

    // Copy encrypted state to buffer
    for (int i = 0; i < 16; i++) {
          encryptedMessage[i] = state[i];
    }
}
```

5.3. Test cases

Input text for the encryption process is the following which is padded to 1952 bytes.

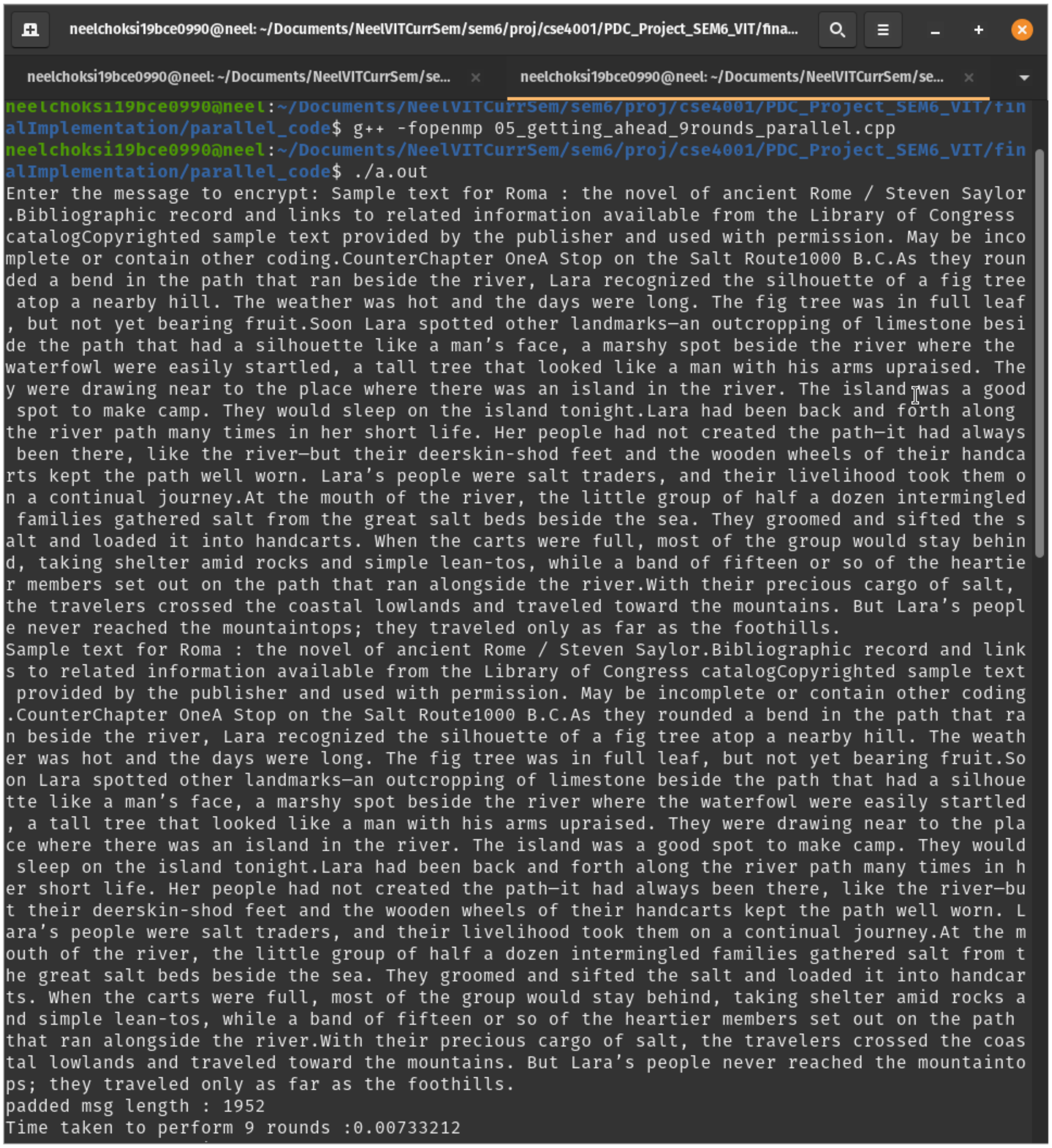
Sample text for Roma : the novel of ancient Rome / Steven Saylor.Bibliographic record and links to related information available from the Library of Congress catalogCopyrighted sample text provided by the publisher and used with permission. May be incomplete or contain other coding.CounterChapter OneA Stop on the Salt Route1000 B.C.As they rounded a bend in the path that ran beside the river, Lara recognized the silhouette of a fig tree atop a nearby hill. The weather was hot and the days were long. The fig tree was in full leaf, but not yet bearing fruit.Soon Lara spotted other landmarks—an outcropping of limestone beside the path that had a silhouette like a man's face, a marshy spot beside the river where the waterfowl were easily startled, a tall tree that looked like a man with his arms upraised. They were drawing near to the place where there was an island in the river. The island was a good spot to make camp. They would sleep on the island tonight.Lara had been back and forth along the river path many times in her short life. Her people had not created the path—it had always been there, like the river—but their deerskin-shod feet and the wooden wheels of their handcarts kept the path well worn. Lara's people were salt traders, and their livelihood took them on a continual journey.At the mouth of the river, the little group of half a dozen intermingled families gathered salt from the great salt beds beside the sea. They groomed and sifted the salt and loaded it into handcarts. When the carts were full, most of the group would stay behind, taking shelter amid rocks and simple lean-tos, while a band of fifteen or so of the heartier members set out on the path that ran alongside the river.With their precious cargo of salt, the travelers crossed the coastal lowlands and traveled toward the mountains. But Lara's people never reached the mountaintops; they traveled only as far as the foothills.

# OUTPUT AND PERFORMANCE ANALYSIS

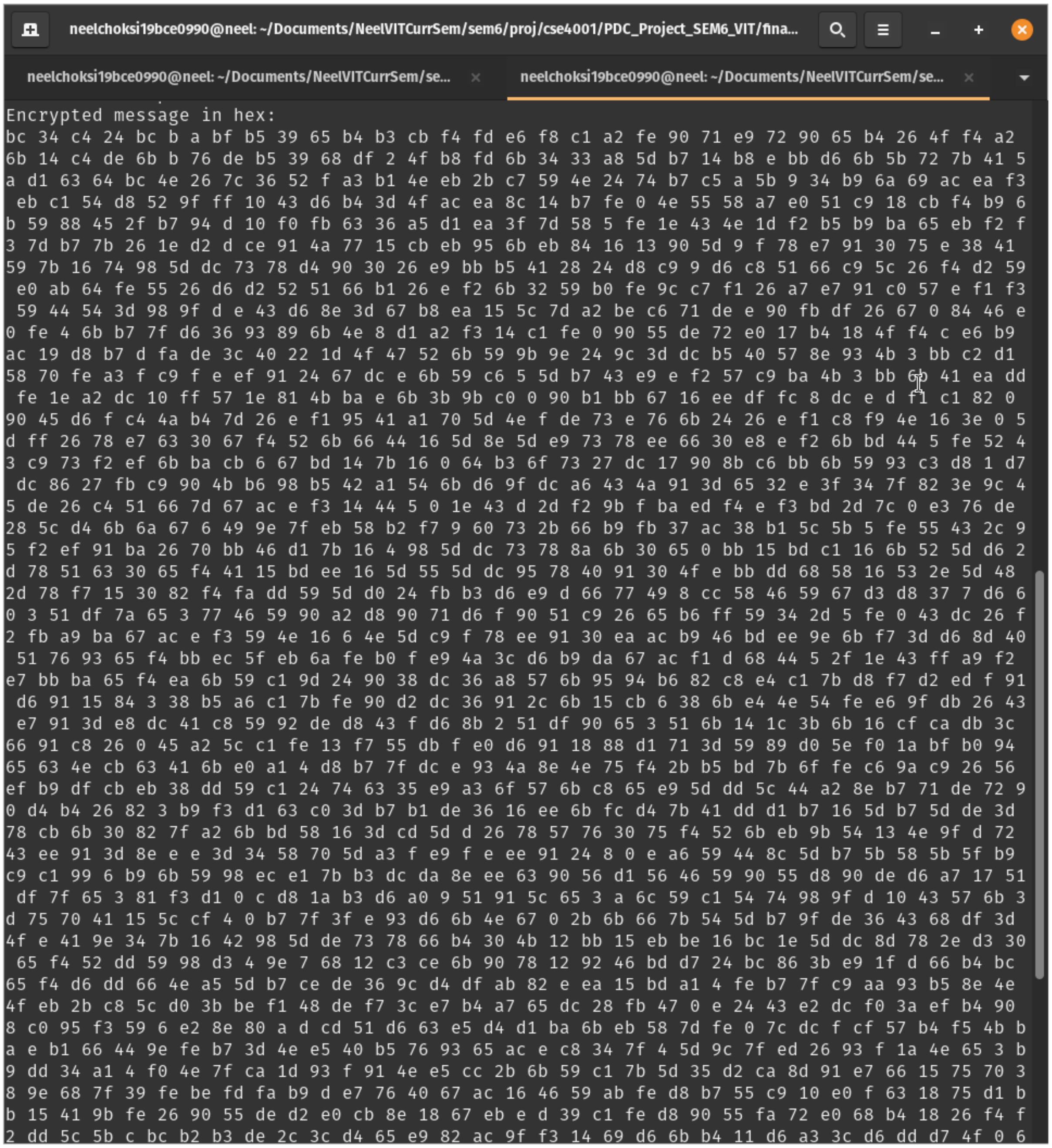
## 6. Output and Performance Analysis

6.1. Execution snapshots

6.1.1. 9 rounds moving ahead in rounds
Parallel :

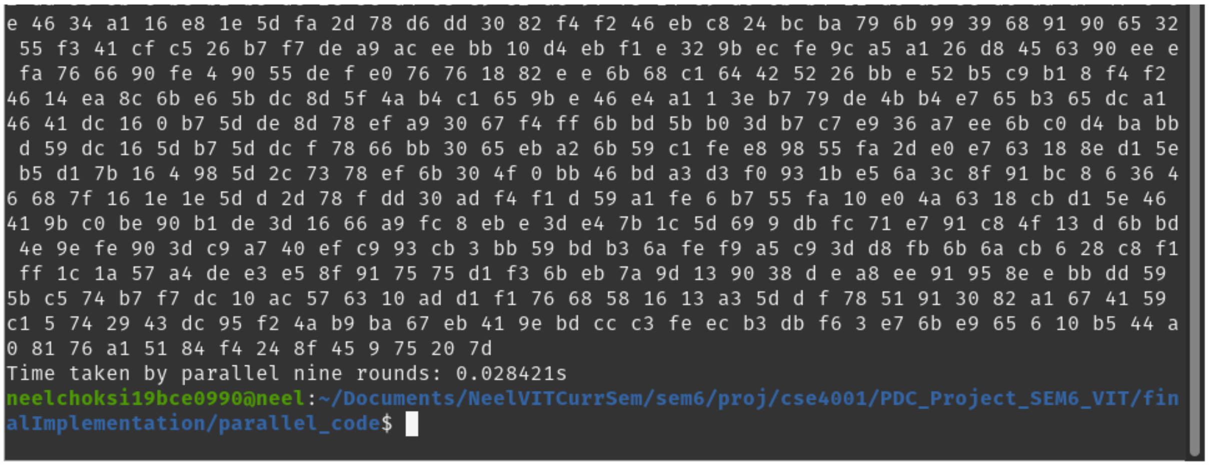


```
neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fina...
neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/se...    neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/se...
neelchoksi19bce0990@neel:~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fin
alImplementation/parallel_code$ g++ -fopenmp 05_getting_ahead_9rounds_parallel.cpp
neelchoksi19bce0990@neel:~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fin
alImplementation/parallel_code$ ./a.out
Enter the message to encrypt: Sample text for Roma : the novel of ancient Rome / Steven Saylor
.Bibliographic record and links to related information available from the Library of Congress
catalogCopyrighted sample text provided by the publisher and used with permission. May be inco
mplete or contain other coding.CounterChapter OneA Stop on the Salt Route1000 B.C.As they roun
ded a bend in the path that ran beside the river, Lara recognized the silhouette of a fig tree
 atop a nearby hill. The weather was hot and the days were long. The fig tree was in full leaf
, but not yet bearing fruit.Soon Lara spotted other landmarks—an outcropping of limestone besi
de the path that had a silhouette like a man’s face, a marshy spot beside the river where the
waterfowl were easily startled, a tall tree that looked like a man with his arms upraised. The
y were drawing near to the place where there was an island in the river. The island was a good
 spot to make camp. They would sleep on the island tonight.Lara had been back and forth along
the river path many times in her short life. Her people had not created the path—it had always
 been there, like the river—but their deerskin-shod feet and the wooden wheels of their handca
rts kept the path well worn. Lara’s people were salt traders, and their livelihood took them o
n a continual journey.At the mouth of the river, the little group of half a dozen intermingled
 families gathered salt from the great salt beds beside the sea. They groomed and sifted the s
alt and loaded it into handcarts. When the carts were full, most of the group would stay behin
d, taking shelter amid rocks and simple lean-tos, while a band of fifteen or so of the heartie
r members set out on the path that ran alongside the river.With their precious cargo of salt,
the travelers crossed the coastal lowlands and traveled toward the mountains. But Lara’s peopl
e never reached the mountaintops; they traveled only as far as the foothills.
Sample text for Roma : the novel of ancient Rome / Steven Saylor.Bibliographic record and link
s to related information available from the Library of Congress catalogCopyrighted sample text
 provided by the publisher and used with permission. May be incomplete or contain other coding
.CounterChapter OneA Stop on the Salt Route1000 B.C.As they rounded a bend in the path that ra
n beside the river, Lara recognized the silhouette of a fig tree atop a nearby hill. The weath
er was hot and the days were long. The fig tree was in full leaf, but not yet bearing fruit.So
on Lara spotted other landmarks—an outcropping of limestone beside the path that had a silhoue
tte like a man’s face, a marshy spot beside the river where the waterfowl were easily startled
, a tall tree that looked like a man with his arms upraised. They were drawing near to the pla
ce where there was an island in the river. The island was a good spot to make camp. They would
 sleep on the island tonight.Lara had been back and forth along the river path many times in h
er short life. Her people had not created the path—it had always been there, like the river—bu
t their deerskin-shod feet and the wooden wheels of their handcarts kept the path well worn. L
ara’s people were salt traders, and their livelihood took them on a continual journey.At the m
outh of the river, the little group of half a dozen intermingled families gathered salt from t
he great salt beds beside the sea. They groomed and sifted the salt and loaded it into handcar
ts. When the carts were full, most of the group would stay behind, taking shelter amid rocks a
nd simple lean-tos, while a band of fifteen or so of the heartier members set out on the path
that ran alongside the river.With their precious cargo of salt, the travelers crossed the coas
tal lowlands and traveled toward the mountains. But Lara’s people never reached the mountainto
ps; they traveled only as far as the foothills.
padded msg length : 1952
Time taken to perform 9 rounds :0.00733212
```



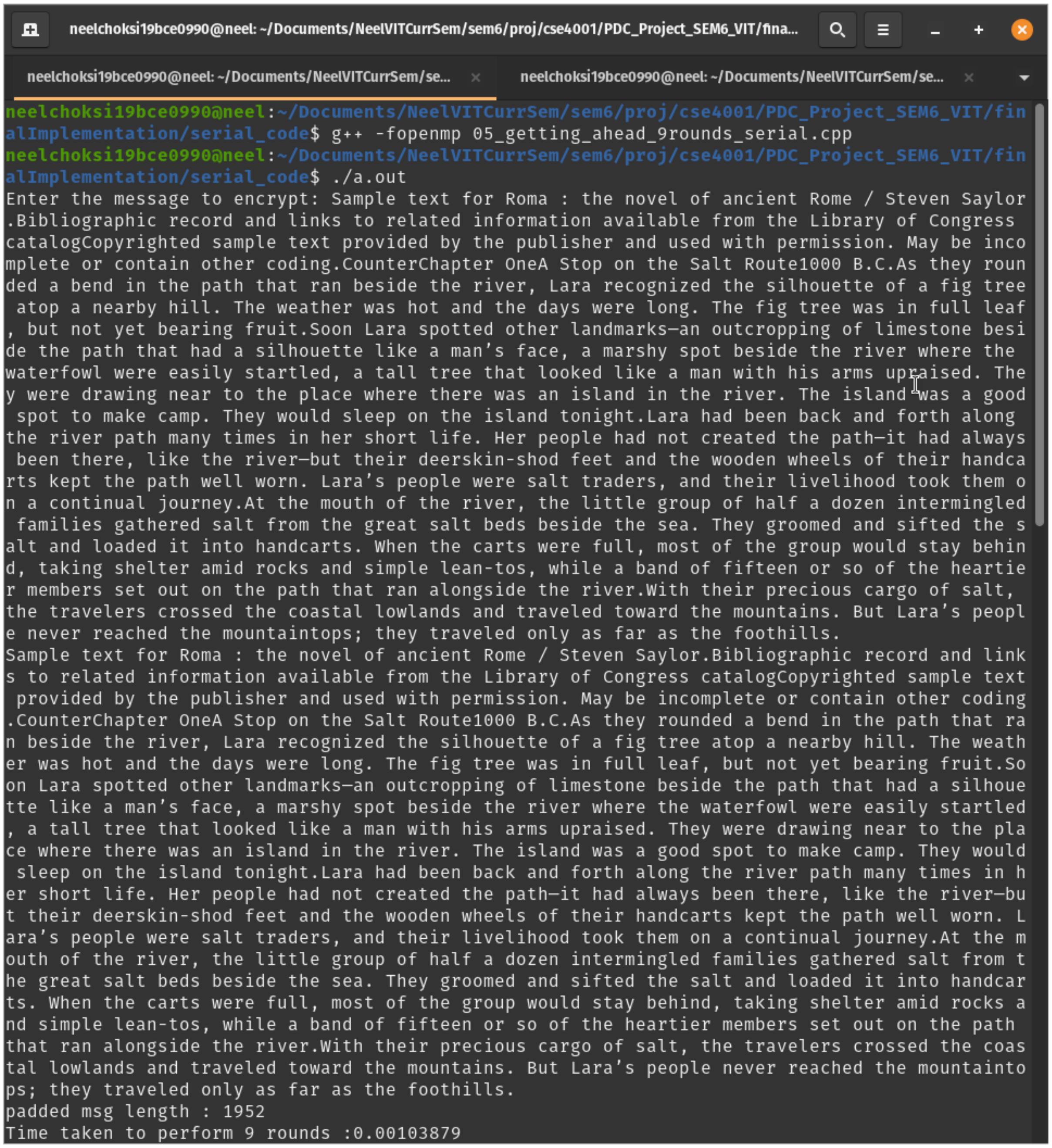
neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fina...

```
Encrypted message in hex:
bc 34 c4 24 bc b a bf b5 39 65 b4 b3 cb f4 fd e6 f8 c1 a2 fe 90 71 e9 72 90 65 b4 26 4f f4 a2
6b 14 c4 de 6b b 76 de b5 39 68 df 2 4f b8 fd 6b 34 33 a8 5d b7 14 b8 e bb d6 6b 5b 72 7b 41 5
a d1 63 64 bc 4e 26 7c 36 52 f a3 b1 4e eb 2b c7 59 4e 24 74 b7 c5 a 5b 9 34 b9 6a 69 ac ea f3
 eb c1 54 d8 52 9f ff 10 43 d6 b4 3d 4f ac ea 8c 14 b7 fe 0 4e 55 58 a7 e0 51 c9 18 cb f4 b9 6
b 59 88 45 2f b7 94 d 10 f0 fb 63 36 a5 d1 ea 3f 7d 58 5 fe 1e 43 4e 1d f2 b5 b9 ba 65 eb f2 f
3 7d b7 7b 26 1e d2 d ce 91 4a 77 15 cb eb 95 6b eb 84 16 13 90 5d 9 f 78 e7 91 30 75 e 38 41
59 7b 16 74 98 5d dc 73 78 d4 90 30 26 e9 bb b5 41 28 24 d8 c9 9 d6 c8 51 66 c9 5c 26 f4 d2 59
 e0 ab 64 fe 55 26 d6 d2 52 51 66 b1 26 e f2 6b 32 59 b0 fe 9c c7 f1 26 a7 e7 91 c0 57 e f1 f3
 59 44 54 3d 98 9f d e 43 d6 8e 3d 67 b8 ea 15 5c 7d a2 be c6 71 de e 90 fb df 26 67 0 84 46 e
0 fe 4 6b b7 7f d6 36 93 89 6b 4e 8 d1 a2 f3 14 c1 fe 0 90 55 de 72 e0 17 b4 18 4f f4 c e6 b9
ac 19 d8 b7 d fa de 3c 40 22 1d 4f 47 52 6b 59 9b 9e 24 9c 3d dc b5 40 57 8e 93 4b 3 bb c2 d1
58 70 fe a3 f c9 f e ef 91 24 67 dc e 6b 59 c6 5 5d b7 43 e9 e f2 57 c9 ba 4b 3 bb 6b 41 ea dd
 fe 1e a2 dc 10 ff 57 1e 81 4b ba e 6b 3b 9b c0 0 90 b1 bb 67 16 ee df fc 8 dc e d f1 c1 82 0
90 45 d6 f c4 4a b4 7d 26 e f1 95 41 a1 70 5d 4e f de 73 e 76 6b 24 26 e f1 c8 f9 4e 16 3e 0 5
d ff 26 78 e7 63 30 67 f4 52 6b 66 44 16 5d 8e 5d e9 73 78 ee 66 30 e8 e f2 6b bd 44 5 fe 52 4
3 c9 73 f2 ef 6b ba cb 6 67 bd 14 7b 16 0 64 b3 6f 73 27 dc 17 90 8b c6 bb 6b 59 93 c3 d8 1 d7
 dc 86 27 fb c9 90 4b b6 98 b5 42 a1 54 6b d6 9f dc a6 43 4a 91 3d 65 32 e 3f 34 7f 82 3e 9c 4
5 de 26 c4 51 66 7d 67 ac e f3 14 44 5 0 1e 43 d 2d f2 9b f ba ed f4 e f3 bd 2d 7c 0 e3 76 de
28 5c d4 6b 6a 67 6 49 9e 7f eb 58 b2 f7 9 60 73 2b 66 b9 fb 37 ac 38 b1 5c 5b 5 fe 55 43 2c 9
5 f2 ef 91 ba 26 70 bb 46 d1 7b 16 4 98 5d dc 73 78 8a 6b 30 65 0 bb 15 bd c1 16 6b 52 5d d6 2
d 78 51 63 30 65 f4 41 15 bd ee 16 5d 55 5d dc 95 78 40 91 30 4f e bb dd 68 58 16 53 2e 5d 48
2d 78 f7 15 30 82 f4 fa dd 59 5d d0 24 fb b3 d6 e9 d 66 77 49 8 cc 58 46 59 67 d3 d8 37 7 d6 6
0 3 51 df 7a 65 3 77 46 59 90 a2 d8 90 71 d6 f 90 51 c9 26 65 b6 ff 59 34 2d 5 fe 0 43 dc 26 f
2 fb a9 ba 67 ac e f3 59 4e 16 6 4e 5d c9 f 78 ee 91 30 ea ac b9 46 bd ee 9e 6b f7 3d d6 8d 40
 51 76 93 65 f4 bb ec 5f eb 6a fe b0 f e9 4a 3c d6 b9 da 67 ac f1 d 68 44 5 2f 1e 43 ff a9 f2
e7 bb ba 65 f4 ea 6b 59 c1 9d 24 90 38 dc 36 a8 57 6b 95 94 b6 82 c8 e4 c1 7b d8 f7 d2 ed f 91
 d6 91 15 84 3 38 b5 a6 c1 7b fe 90 d2 dc 36 91 2c 6b 15 cb 6 38 6b e4 4e 54 fe e6 9f db 26 43
 e7 91 3d e8 dc 41 c8 59 92 de d8 43 f d6 8b 2 51 df 90 65 3 51 6b 14 1c 3b 6b 16 cf ca db 3c
66 91 c8 26 0 45 a2 5c c1 fe 13 f7 55 db f e0 d6 91 18 88 d1 71 3d 59 89 d0 5e f0 1a bf b0 94
65 63 4e cb 63 41 6b e0 a1 4 d8 b7 7f dc e 93 4a 8e 4e 75 f4 2b b5 bd 7b 6f fe c6 9a c9 26 56
ef b9 df cb eb 38 dd 59 c1 24 74 63 35 e9 a3 6f 57 6b c8 65 e9 5d dd 5c 44 a2 8e b7 71 de 72 9
0 d4 b4 26 82 3 b9 f3 d1 63 c0 3d b7 b1 de 36 16 ee 6b fc d4 7b 41 dd d1 b7 16 5d b7 5d de 3d
78 cb 6b 30 82 7f a2 6b bd 58 16 3d cd 5d d 26 78 57 76 30 75 f4 52 6b eb 9b 54 13 4e 9f d 72
43 ee 91 3d 8e e e 3d 34 58 70 5d a3 f e9 f e ee 91 24 8 0 e a6 59 44 8c 5d b7 5b 58 5b 5f b9
c9 c1 99 6 b9 6b 59 98 ec e1 7b b3 dc da 8e ee 63 90 56 d1 56 46 59 90 55 d8 90 de d6 a7 17 51
 df 7f 65 3 81 f3 d1 0 c d8 1a b3 d6 a0 9 51 91 5c 65 3 a 6c 59 c1 54 74 98 9f d 10 43 57 6b 3
d 75 70 41 15 5c cf 4 0 b7 7f 3f e 93 d6 6b 4e 67 0 2b 6b 66 7b 54 5d b7 9f de 36 43 68 df 3d
4f e 41 9e 34 7b 16 42 98 5d de 73 78 66 b4 30 4b 12 bb 15 eb be 16 bc 1e 5d dc 8d 78 2e d3 30
 65 f4 52 dd 59 98 d3 4 9e 7 68 12 c3 ce 6b 90 78 12 92 46 bd d7 24 bc 86 3b e9 1f d 66 b4 bc
65 f4 d6 dd 66 4e a5 5d b7 ce de 36 9c d4 df ab 82 e ea 15 bd a1 4 fe b7 7f c9 aa 93 b5 8e 4e
4f eb 2b c8 5c d0 3b be f1 48 de f7 3c e7 b4 a7 65 dc 28 fb 47 0 e 24 43 e2 dc f0 3a ef b4 90
8 c0 95 f3 59 6 e2 8e 80 a d cd 51 d6 63 e5 d4 d1 ba 6b eb 58 7d fe 0 7c dc f cf 57 b4 f5 4b b
a e b1 66 44 9e fe b7 3d 4e e5 40 b5 76 93 65 ac e c8 34 7f 4 5d 9c 7f ed 26 93 f 1a 4e 65 3 b
9 dd 34 a1 4 f0 4e 7f ca 1d 93 f 91 4e e5 cc 2b 6b 59 c1 7b 5d 35 d2 ca 8d 91 e7 66 15 75 70 3
8 9e 68 7f 39 fe be fd fa b9 d e7 76 40 67 ac 16 46 59 ab fe d8 b7 55 c9 10 e0 f 63 18 75 d1 b
b 15 41 9b fe 26 90 55 de d2 e0 cb 8e 18 67 eb e d 39 c1 fe d8 90 55 fa 72 e0 68 b4 18 26 f4 f
2 dd 5c 5b c bc b2 b3 de 2c 3c d4 65 e9 82 ac 9f f3 14 69 d6 6b b4 11 d6 a3 3c d6 dd d7 4f 0 6
```

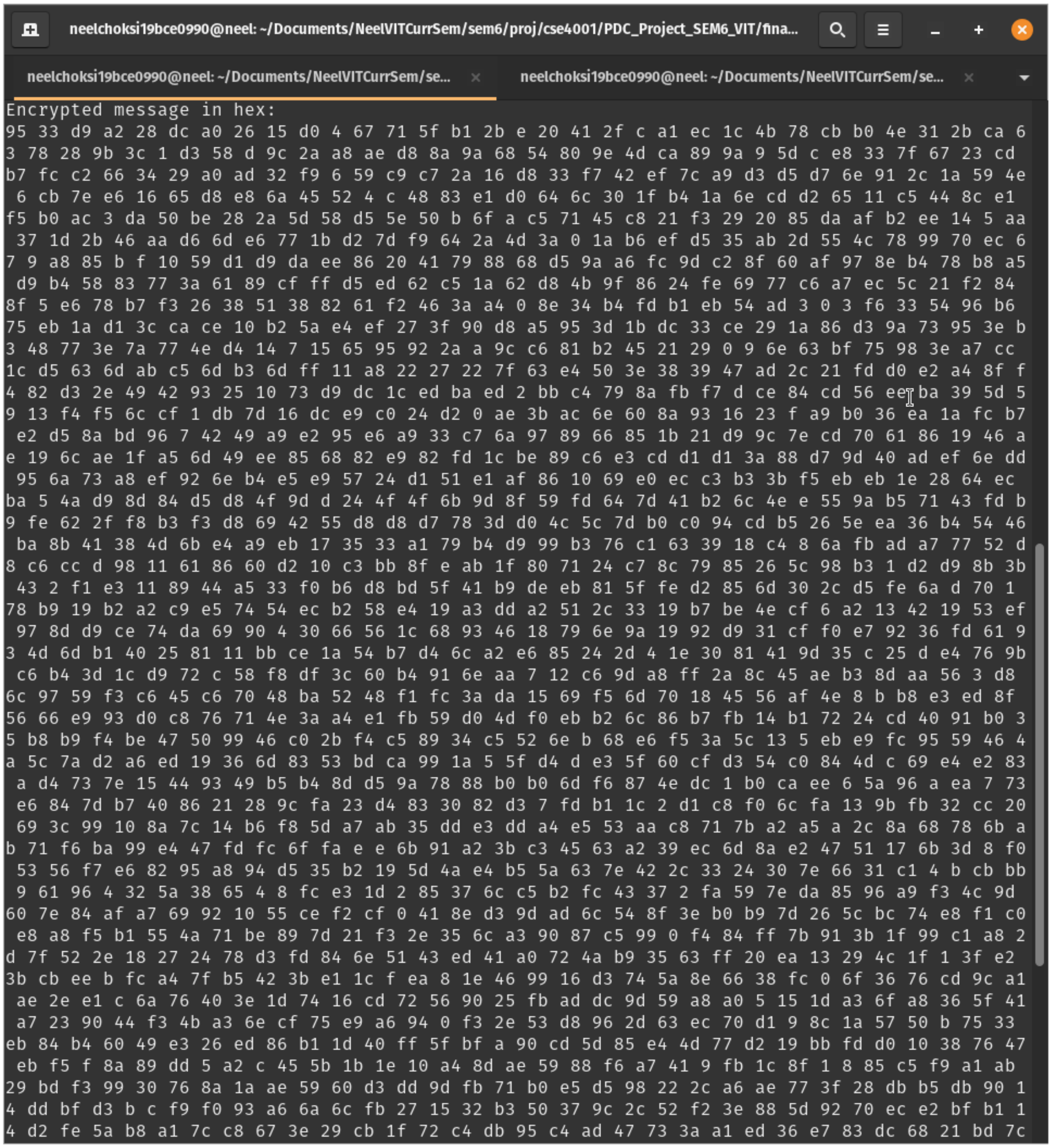


```
e 46 34 a1 16 e8 1e 5d fa 2d 78 d6 dd 30 82 f4 f2 46 eb c8 24 bc ba 79 6b 99 39 68 91 90 65 32
 55 f3 41 cf c5 26 b7 f7 de a9 ac ee bb 10 d4 eb f1 e 32 9b ec fe 9c a5 a1 26 d8 45 63 90 ee e
 fa 76 66 90 fe 4 90 55 de f e0 76 76 18 82 e e 6b 68 c1 64 42 52 26 bb e 52 b5 c9 b1 8 f4 f2
46 14 ea 8c 6b e6 5b dc 8d 5f 4a b4 c1 65 9b e 46 e4 a1 1 3e b7 79 de 4b b4 e7 65 b3 65 dc a1
46 41 dc 16 0 b7 5d de 8d 78 ef a9 30 67 f4 ff 6b bd 5b b0 3d b7 c7 e9 36 a7 ee 6b c0 d4 ba bb
 d 59 dc 16 5d b7 5d dc f 78 66 bb 30 65 eb a2 6b 59 c1 fe e8 98 55 fa 2d e0 e7 63 18 8e d1 5e
 b5 d1 7b 16 4 98 5d 2c 73 78 ef 6b 30 4f 0 bb 46 bd a3 d3 f0 93 1b e5 6a 3c 8f 91 bc 8 6 36 4
6 68 7f 16 1e 1e 5d d 2d 78 f dd 30 ad f4 f1 d 59 a1 fe 6 b7 55 fa 10 e0 4a 63 18 cb d1 5e 46
41 9b c0 be 90 b1 de 3d 16 66 a9 fc 8 eb e 3d e4 7b 1c 5d 69 9 db fc 71 e7 91 c8 4f 13 d 6b bd
 4e 9e fe 90 3d c9 a7 40 ef c9 93 cb 3 bb 59 bd b3 6a fe f9 a5 c9 3d d8 fb 6b 6a cb 6 28 c8 f1
 ff 1c 1a 57 a4 de e3 e5 8f 91 75 75 d1 f3 6b eb 7a 9d 13 90 38 d e a8 ee 91 95 8e e bb dd 59
5b c5 74 b7 f7 dc 10 ac 57 63 10 ad d1 f1 76 68 58 16 13 a3 5d d f 78 51 91 30 82 a1 67 41 59
c1 5 74 29 43 dc 95 f2 4a b9 ba 67 eb 41 9e bd cc c3 fe ec b3 db f6 3 e7 6b e9 65 6 10 b5 44 a
0 81 76 a1 51 84 f4 24 8f 45 9 75 20 7d
Time taken by parallel nine rounds: 0.028421s
neelchoksi19bce0990@neel:~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fin
alImplementation/parallel_code$
```

Serial :

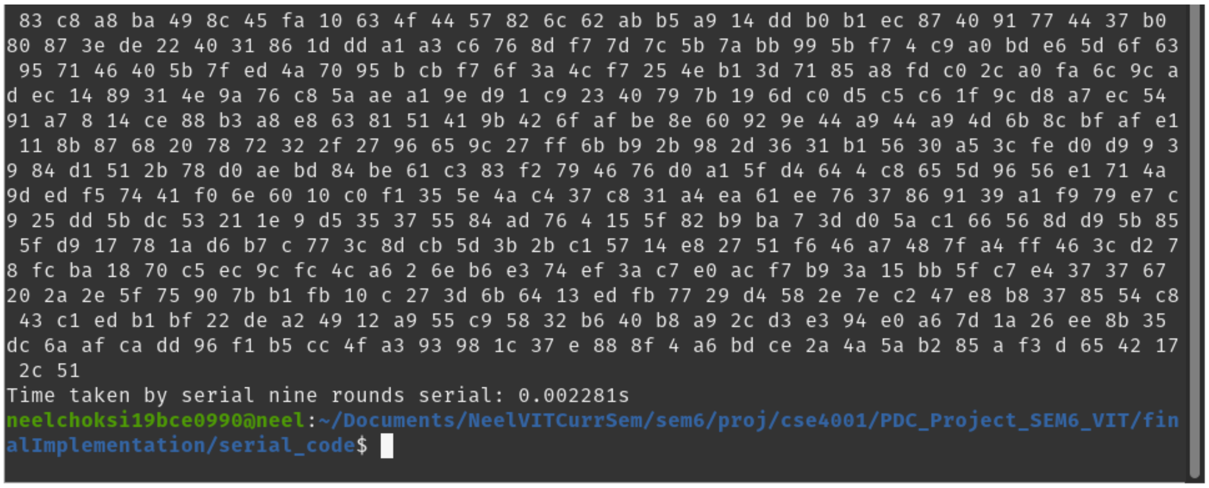


```
neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fina...
neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/se...    neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/se...
neelchoksi19bce0990@neel:~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fin
alImplementation/serial_code$ g++ -fopenmp 05_getting_ahead_9rounds_serial.cpp
neelchoksi19bce0990@neel:~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fin
alImplementation/serial_code$ ./a.out
Enter the message to encrypt: Sample text for Roma : the novel of ancient Rome / Steven Saylor
.Bibliographic record and links to related information available from the Library of Congress
catalogCopyrighted sample text provided by the publisher and used with permission. May be inco
mplete or contain other coding.CounterChapter OneA Stop on the Salt Route1000 B.C.As they roun
ded a bend in the path that ran beside the river, Lara recognized the silhouette of a fig tree
 atop a nearby hill. The weather was hot and the days were long. The fig tree was in full leaf
, but not yet bearing fruit.Soon Lara spotted other landmarks—an outcropping of limestone besi
de the path that had a silhouette like a man's face, a marshy spot beside the river where the
waterfowl were easily startled, a tall tree that looked like a man with his arms upraised. The
y were drawing near to the place where there was an island in the river. The island was a good
 spot to make camp. They would sleep on the island tonight.Lara had been back and forth along
the river path many times in her short life. Her people had not created the path—it had always
 been there, like the river—but their deerskin-shod feet and the wooden wheels of their handca
rts kept the path well worn. Lara's people were salt traders, and their livelihood took them o
n a continual journey.At the mouth of the river, the little group of half a dozen intermingled
 families gathered salt from the great salt beds beside the sea. They groomed and sifted the s
alt and loaded it into handcarts. When the carts were full, most of the group would stay behin
d, taking shelter amid rocks and simple lean-tos, while a band of fifteen or so of the heartie
r members set out on the path that ran alongside the river.With their precious cargo of salt,
the travelers crossed the coastal lowlands and traveled toward the mountains. But Lara's peopl
e never reached the mountaintops; they traveled only as far as the foothills.
Sample text for Roma : the novel of ancient Rome / Steven Saylor.Bibliographic record and link
s to related information available from the Library of Congress catalogCopyrighted sample text
 provided by the publisher and used with permission. May be incomplete or contain other coding
.CounterChapter OneA Stop on the Salt Route1000 B.C.As they rounded a bend in the path that ra
n beside the river, Lara recognized the silhouette of a fig tree atop a nearby hill. The weath
er was hot and the days were long. The fig tree was in full leaf, but not yet bearing fruit.So
on Lara spotted other landmarks—an outcropping of limestone beside the path that had a silhoue
tte like a man's face, a marshy spot beside the river where the waterfowl were easily startled
, a tall tree that looked like a man with his arms upraised. They were drawing near to the pla
ce where there was an island in the river. The island was a good spot to make camp. They would
 sleep on the island tonight.Lara had been back and forth along the river path many times in h
er short life. Her people had not created the path—it had always been there, like the river—bu
t their deerskin-shod feet and the wooden wheels of their handcarts kept the path well worn. L
ara's people were salt traders, and their livelihood took them on a continual journey.At the m
outh of the river, the little group of half a dozen intermingled families gathered salt from t
he great salt beds beside the sea. They groomed and sifted the salt and loaded it into handcar
ts. When the carts were full, most of the group would stay behind, taking shelter amid rocks a
nd simple lean-tos, while a band of fifteen or so of the heartier members set out on the path
that ran alongside the river.With their precious cargo of salt, the travelers crossed the coas
tal lowlands and traveled toward the mountains. But Lara's people never reached the mountainto
ps; they traveled only as far as the foothills.
padded msg length : 1952
Time taken to perform 9 rounds :0.00103879
```



neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fina...
neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/se...
neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/se...

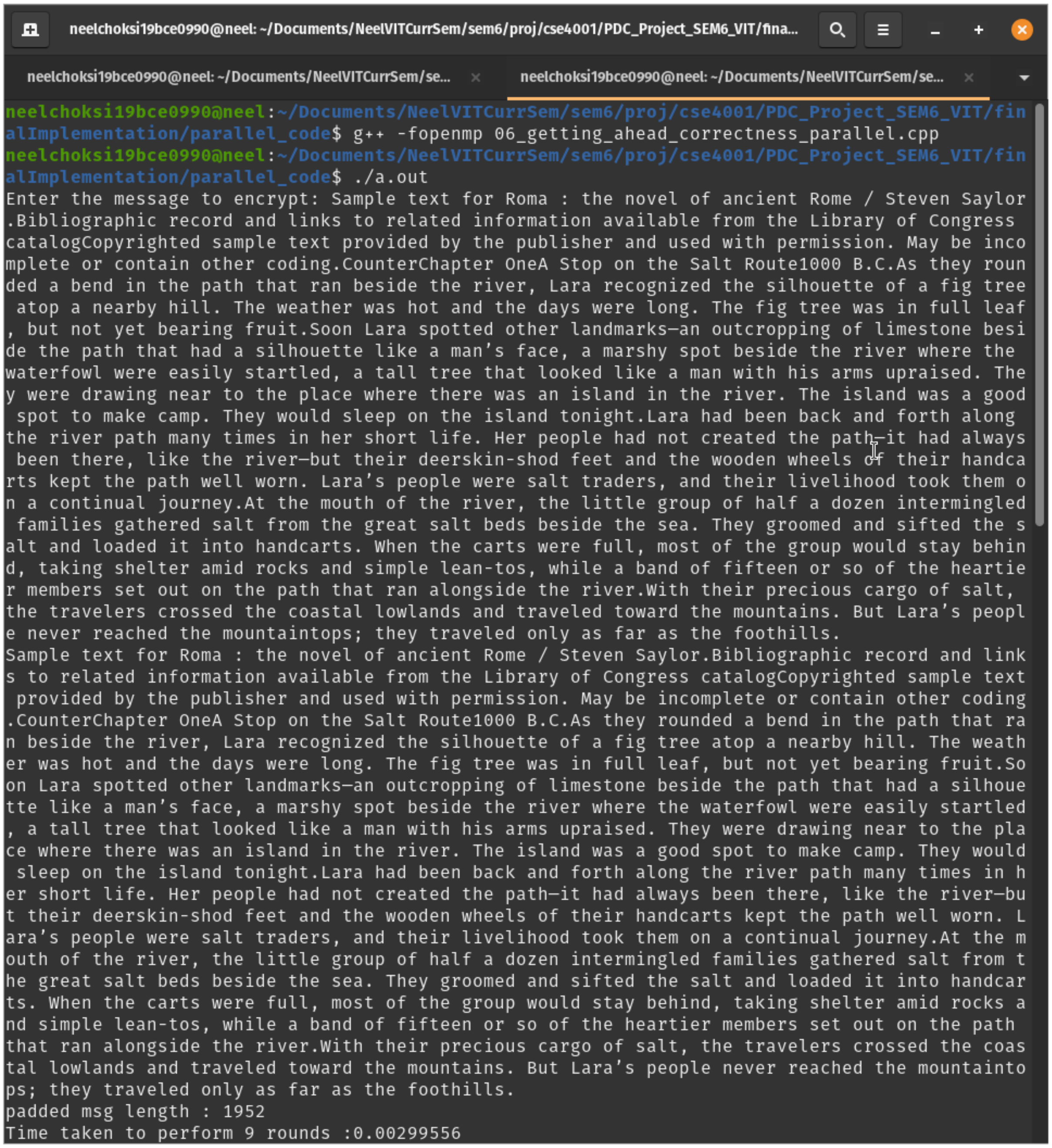
```
Encrypted message in hex:
95 33 d9 a2 28 dc a0 26 15 d0 4 67 71 5f b1 2b e 20 41 2f c a1 ec 1c 4b 78 cb b0 4e 31 2b ca 6
3 78 28 9b 3c 1 d3 58 d 9c 2a a8 ae d8 8a 9a 68 54 80 9e 4d ca 89 9a 9 5d c e8 33 7f 67 23 cd
b7 fc c2 66 34 29 a0 ad 32 f9 6 59 c9 c7 2a 16 d8 33 f7 42 ef 7c a9 d3 d5 d7 6e 91 2c 1a 59 4e
 6 cb 7e e6 16 65 d8 e8 6a 45 52 4 c 48 83 e1 d0 64 6c 30 1f b4 1a 6e cd d2 65 11 c5 44 8c e1
f5 b0 ac 3 da 50 be 28 2a 5d 58 d5 5e 50 b 6f a c5 71 45 c8 21 f3 29 20 85 da af b2 ee 14 5 aa
 37 1d 2b 46 aa d6 6d e6 77 1b d2 7d f9 64 2a 4d 3a 0 1a b6 ef d5 35 ab 2d 55 4c 78 99 70 ec 6
7 9 a8 85 b f 10 59 d1 d9 da ee 86 20 41 79 88 68 d5 9a a6 fc 9d c2 8f 60 af 97 8e b4 78 b8 a5
 d9 b4 58 83 77 3a 61 89 cf ff d5 ed 62 c5 1a 62 d8 4b 9f 86 24 fe 69 77 c6 a7 ec 5c 21 f2 84
8f 5 e6 78 b7 f3 26 38 51 38 82 61 f2 46 3a a4 0 8e 34 b4 fd b1 eb 54 ad 3 0 3 f6 33 54 96 b6
75 eb 1a d1 3c ca ce 10 b2 5a e4 ef 27 3f 90 d8 a5 95 3d 1b dc 33 ce 29 1a 86 d3 9a 73 95 3e b
3 48 77 3e 7a 77 4e d4 14 7 15 65 95 92 2a a 9c c6 81 b2 45 21 29 0 9 6e 63 bf 75 98 3e a7 cc
1c d5 63 6d ab c5 6d b3 6d ff 11 a8 22 27 22 7f 63 e4 50 3e 38 39 47 ad 2c 21 fd d0 e2 a4 8f f
4 82 d3 2e 49 42 93 25 10 73 d9 dc 1c ed ba ed 2 bb c4 79 8a fb f7 d ce 84 cd 56 ee ba 39 5d 5
9 13 f4 f5 6c cf 1 db 7d 16 dc e9 c0 24 d2 0 ae 3b ac 6e 60 8a 93 16 23 f a9 b0 36 ea 1a fc b7
 e2 d5 8a bd 96 7 42 49 a9 e2 95 e6 a9 33 c7 6a 97 89 66 85 1b 21 d9 9c 7e cd 70 61 86 19 46 a
e 19 6c ae 1f a5 6d 49 ee 85 68 82 e9 82 fd 1c be 89 c6 e3 cd d1 d1 3a 88 d7 9d 40 ad ef 6e dd
 95 6a 73 a8 ef 92 6e b4 e5 e9 57 24 d1 51 e1 af 86 10 69 e0 ec c3 b3 3b f5 eb eb 1e 28 64 ec
ba 5 4a d9 8d 84 d5 d8 4f 9d d 24 4f 4f 6b 9d 8f 59 fd 64 7d 41 b2 6c 4e e 55 9a b5 71 43 fd b
9 fe 62 2f f8 b3 f3 d8 69 42 55 d8 d8 d7 78 3d d0 4c 5c 7d b0 c0 94 cd b5 26 5e ea 36 b4 54 46
 ba 8b 41 38 4d 6b e4 a9 eb 17 35 33 a1 79 b4 d9 99 b3 76 c1 63 39 18 c4 8 6a fb ad a7 77 52 d
8 c6 cc d 98 11 61 86 60 d2 10 c3 bb 8f e ab 1f 80 71 24 c7 8c 79 85 26 5c 98 b3 1 d2 d9 8b 3b
 43 2 f1 e3 11 89 44 a5 33 f0 b6 d8 bd 5f 41 b9 de eb 81 5f fe d2 85 6d 30 2c d5 fe 6a d 70 1
78 b9 19 b2 a2 c9 e5 74 54 ec b2 58 e4 19 a3 dd a2 51 2c 33 19 b7 be 4e cf 6 a2 13 42 19 53 ef
 97 8d d9 ce 74 da 69 90 4 30 66 56 1c 68 93 46 18 79 6e 9a 19 92 d9 31 cf f0 e7 92 36 fd 61 9
3 4d 6d b1 40 25 81 11 bb ce 1a 54 b7 d4 6c a2 e6 85 24 2d 4 1e 30 81 41 9d 35 c 25 d e4 76 9b
 c6 b4 3d 1c d9 72 c 58 f8 df 3c 60 b4 91 6e aa 7 12 c6 9d a8 ff 2a 8c 45 ae b3 8d aa 56 3 d8
6c 97 59 f3 c6 45 c6 70 48 ba 52 48 f1 fc 3a da 15 69 f5 6d 70 18 45 56 af 4e 8 b b8 e3 ed 8f
56 66 e9 93 d0 c8 76 71 4e 3a a4 e1 fb 59 d0 4d f0 eb b2 6c 86 b7 fb 14 b1 72 24 cd 40 91 b0 3
5 b8 b9 f4 be 47 50 99 46 c0 2b f4 c5 89 34 c5 52 6e b 68 e6 f5 3a 5c 13 5 eb e9 fc 95 59 46 4
a 5c 7a d2 a6 ed 19 36 6d 83 53 bd ca 99 1a 5 5f d4 d e3 5f 60 cf d3 54 c0 84 4d c 69 e4 e2 83
 a d4 73 7e 15 44 93 49 b5 b4 8d d5 9a 78 88 b0 b0 6d f6 87 4e dc 1 b0 ca ee 6 5a 96 a ea 7 73
 e6 84 7d b7 40 86 21 28 9c fa 23 d4 83 30 82 d3 7 fd b1 1c 2 d1 c8 f0 6c fa 13 9b fb 32 cc 20
 69 3c 99 10 8a 7c 14 b6 f8 5d a7 ab 35 dd e3 dd a4 e5 53 aa c8 71 7b a2 a5 a 2c 8a 68 78 6b a
b 71 f6 ba 99 e4 47 fd fc 6f fa e e 6b 91 a2 3b c3 45 63 a2 39 ec 6d 8a e2 47 51 17 6b 3d 8 f0
 53 56 f7 e6 82 95 a8 94 d5 35 b2 19 5d 4a e4 b5 5a 63 7e 42 2c 33 24 30 7e 66 31 c1 4 b cb bb
 9 61 96 4 32 5a 38 65 4 8 fc e3 1d 2 85 37 6c c5 b2 fc 43 37 2 fa 59 7e da 85 96 a9 f3 4c 9d
60 7e 84 af a7 69 92 10 55 ce f2 cf 0 41 8e d3 9d ad 6c 54 8f 3e b0 b9 7d 26 5c bc 74 e8 f1 c0
 e8 a8 f5 b1 55 4a 71 be 89 7d 21 f3 2e 35 6c a3 90 87 c5 99 0 f4 84 ff 7b 91 3b 1f 99 c1 a8 2
d 7f 52 2e 18 27 24 78 d3 fd 84 6e 51 43 ed 41 a0 72 4a b9 35 63 ff 20 ea 13 29 4c 1f 1 3f e2
3b cb ee b fc a4 7f b5 42 3b e1 1c f ea 8 1e 46 99 16 d3 74 5a 8e 66 38 fc 0 6f 36 76 cd 9c a1
 ae 2e e1 c 6a 76 40 3e 1d 74 16 cd 72 56 90 25 fb ad dc 9d 59 a8 a0 5 15 1d a3 6f a8 36 5f 41
 a7 23 90 44 f3 4b a3 6e cf 75 e9 a6 94 0 f3 2e 53 d8 96 2d 63 ec 70 d1 9 8c 1a 57 50 b 75 33
eb 84 b4 60 49 e3 26 ed 86 b1 1d 40 ff 5f bf a 90 cd 5d 85 e4 4d 77 d2 19 bb fd d0 10 38 76 47
 eb f5 f 8a 89 dd 5 a2 c 45 5b 1b 1e 10 a4 8d ae 59 88 f6 a7 41 9 fb 1c 8f 1 8 85 c5 f9 a1 ab
29 bd f3 99 30 76 8a 1a ae 59 60 d3 dd 9d fb 71 b0 e5 d5 98 22 2c a6 ae 77 3f 28 db b5 db 90 1
4 dd bf d3 b c f9 f0 93 a6 6a 6c fb 27 15 32 b3 50 37 9c 2c 52 f2 3e 88 5d 92 70 ec e2 bf b1 1
4 d2 fe 5a b8 a1 7c c8 67 3e 29 cb 1f 72 c4 db 95 c4 ad 47 73 3a a1 ed 36 e7 83 dc 68 21 bd 7c
```



```
 83 c8 a8 ba 49 8c 45 fa 10 63 4f 44 57 82 6c 62 ab b5 a9 14 dd b0 b1 ec 87 40 91 77 44 37 b0
80 87 3e de 22 40 31 86 1d dd a1 a3 c6 76 8d f7 7d 7c 5b 7a bb 99 5b f7 4 c9 a0 bd e6 5d 6f 63
 95 71 46 40 5b 7f ed 4a 70 95 b cb f7 6f 3a 4c f7 25 4e b1 3d 71 85 a8 fd c0 2c a0 fa 6c 9c a
d ec 14 89 31 4e 9a 76 c8 5a ae a1 9e d9 1 c9 23 40 79 7b 19 6d c0 d5 c5 c6 1f 9c d8 a7 ec 54
91 a7 8 14 ce 88 b3 a8 e8 63 81 51 41 9b 42 6f af be 8e 60 92 9e 44 a9 44 a9 4d 6b 8c bf af e1
 11 8b 87 68 20 78 72 32 2f 27 96 65 9c 27 ff 6b b9 2b 98 2d 36 31 b1 56 30 a5 3c fe d0 d9 9 3
9 84 d1 51 2b 78 d0 ae bd 84 be 61 c3 83 f2 79 46 76 d0 a1 5f d4 64 4 c8 65 5d 96 56 e1 71 4a
9d ed f5 74 41 f0 6e 60 10 c0 f1 35 5e 4a c4 37 c8 31 a4 ea 61 ee 76 37 86 91 39 a1 f9 79 e7 c
9 25 dd 5b dc 53 21 1e 9 d5 35 37 55 84 ad 76 4 15 5f 82 b9 ba 7 3d d0 5a c1 66 56 8d d9 5b 85
 5f d9 17 78 1a d6 b7 c 77 3c 8d cb 5d 3b 2b c1 57 14 e8 27 51 f6 46 a7 48 7f a4 ff 46 3c d2 7
8 fc ba 18 70 c5 ec 9c fc 4c a6 2 6e b6 e3 74 ef 3a c7 e0 ac f7 b9 3a 15 bb 5f c7 e4 37 37 67
20 2a 2e 5f 75 90 7b b1 fb 10 c 27 3d 6b 64 13 ed fb 77 29 d4 58 2e 7e c2 47 e8 b8 37 85 54 c8
 43 c1 ed b1 bf 22 de a2 49 12 a9 55 c9 58 32 b6 40 b8 a9 2c d3 e3 94 e0 a6 7d 1a 26 ee 8b 35
dc 6a af ca dd 96 f1 b5 cc 4f a3 93 98 1c 37 e 88 8f 4 a6 bd ce 2a 4a 5a b2 85 a f3 d 65 42 17
 2c 51
Time taken by serial nine rounds serial: 0.002281s
neelchoksi19bce0990@neel:~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fin
alImplementation/serial_code$ 
```

6.1.2. 9 rounds moving ahead in rounds , maintaining correctness

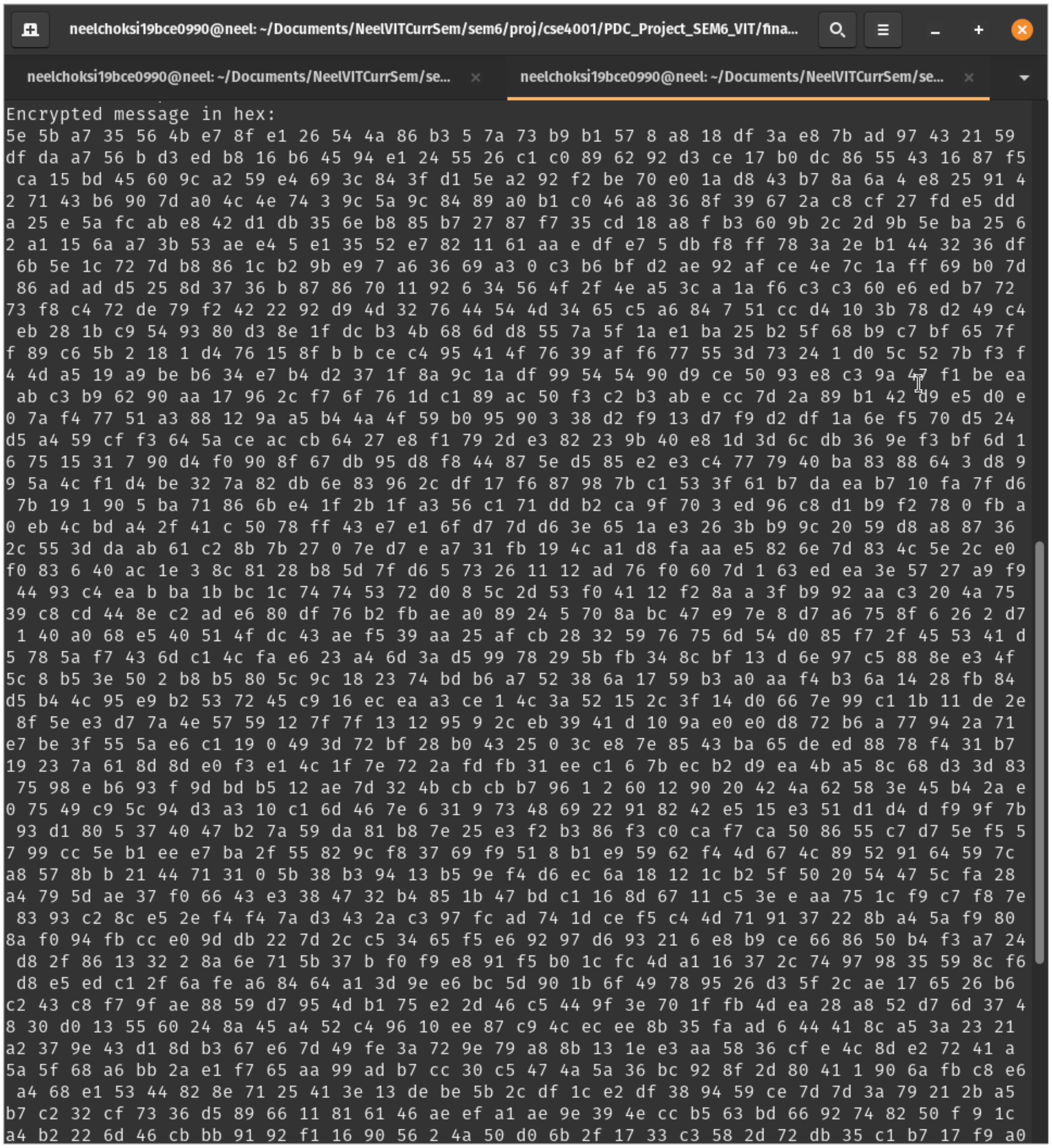
Parallel :



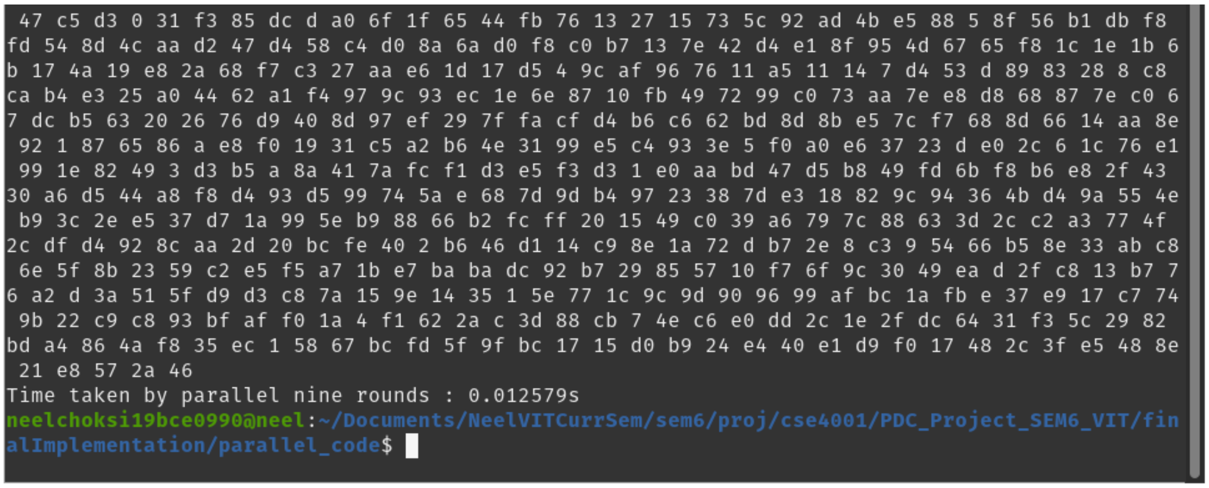
neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fina...

neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/se... | neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/se...

```
neelchoksi19bce0990@neel:~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fin
alImplementation/parallel_code$ g++ -fopenmp 06_getting_ahead_correctness_parallel.cpp
neelchoksi19bce0990@neel:~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fin
alImplementation/parallel_code$ ./a.out
Enter the message to encrypt: Sample text for Roma : the novel of ancient Rome / Steven Saylor
.Bibliographic record and links to related information available from the Library of Congress
catalogCopyrighted sample text provided by the publisher and used with permission. May be inco
mplete or contain other coding.CounterChapter OneA Stop on the Salt Route1000 B.C.As they roun
ded a bend in the path that ran beside the river, Lara recognized the silhouette of a fig tree
 atop a nearby hill. The weather was hot and the days were long. The fig tree was in full leaf
, but not yet bearing fruit.Soon Lara spotted other landmarks—an outcropping of limestone besi
de the path that had a silhouette like a man’s face, a marshy spot beside the river where the
waterfowl were easily startled, a tall tree that looked like a man with his arms upraised. The
y were drawing near to the place where there was an island in the river. The island was a good
 spot to make camp. They would sleep on the island tonight.Lara had been back and forth along
the river path many times in her short life. Her people had not created the path—it had always
 been there, like the river—but their deerskin-shod feet and the wooden wheels of their handca
rts kept the path well worn. Lara’s people were salt traders, and their livelihood took them o
n a continual journey.At the mouth of the river, the little group of half a dozen intermingled
 families gathered salt from the great salt beds beside the sea. They groomed and sifted the s
alt and loaded it into handcarts. When the carts were full, most of the group would stay behin
d, taking shelter amid rocks and simple lean-tos, while a band of fifteen or so of the heartie
r members set out on the path that ran alongside the river.With their precious cargo of salt,
the travelers crossed the coastal lowlands and traveled toward the mountains. But Lara’s peopl
e never reached the mountaintops; they traveled only as far as the foothills.
Sample text for Roma : the novel of ancient Rome / Steven Saylor.Bibliographic record and link
s to related information available from the Library of Congress catalogCopyrighted sample text
 provided by the publisher and used with permission. May be incomplete or contain other coding
.CounterChapter OneA Stop on the Salt Route1000 B.C.As they rounded a bend in the path that ra
n beside the river, Lara recognized the silhouette of a fig tree atop a nearby hill. The weath
er was hot and the days were long. The fig tree was in full leaf, but not yet bearing fruit.So
on Lara spotted other landmarks—an outcropping of limestone beside the path that had a silhoue
tte like a man’s face, a marshy spot beside the river where the waterfowl were easily startled
, a tall tree that looked like a man with his arms upraised. They were drawing near to the pla
ce where there was an island in the river. The island was a good spot to make camp. They would
 sleep on the island tonight.Lara had been back and forth along the river path many times in h
er short life. Her people had not created the path—it had always been there, like the river—bu
t their deerskin-shod feet and the wooden wheels of their handcarts kept the path well worn. L
ara’s people were salt traders, and their livelihood took them on a continual journey.At the m
outh of the river, the little group of half a dozen intermingled families gathered salt from t
he great salt beds beside the sea. They groomed and sifted the salt and loaded it into handcar
ts. When the carts were full, most of the group would stay behind, taking shelter amid rocks a
nd simple lean-tos, while a band of fifteen or so of the heartier members set out on the path
that ran alongside the river.With their precious cargo of salt, the travelers crossed the coas
tal lowlands and traveled toward the mountains. But Lara’s people never reached the mountainto
ps; they traveled only as far as the foothills.
padded msg length : 1952
Time taken to perform 9 rounds :0.00299556
```

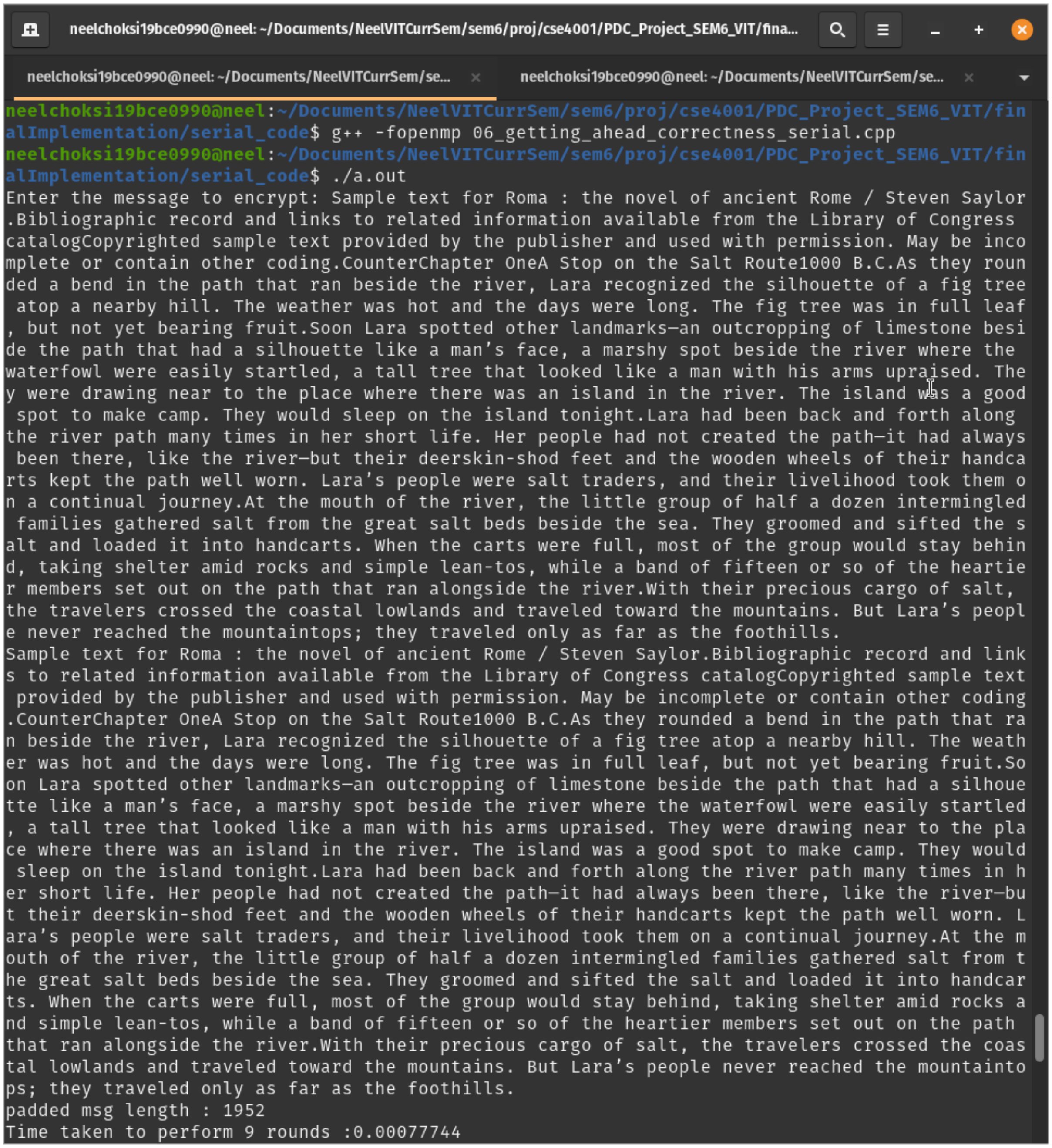

neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fina...
neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/se...
neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/se...

```
Encrypted message in hex:
5e 5b a7 35 56 4b e7 8f e1 26 54 4a 86 b3 5 7a 73 b9 b1 57 8 a8 18 df 3a e8 7b ad 97 43 21 59
df da a7 56 b d3 ed b8 16 b6 45 94 e1 24 55 26 c1 c0 89 62 92 d3 ce 17 b0 dc 86 55 43 16 87 f5
 ca 15 bd 45 60 9c a2 59 e4 69 3c 84 3f d1 5e a2 92 f2 be 70 e0 1a d8 43 b7 8a 6a 4 e8 25 91 4
2 71 43 b6 90 7d a0 4c 4e 74 3 9c 5a 9c 84 89 a0 b1 c0 46 a8 36 8f 39 67 2a c8 cf 27 fd e5 dd
a 25 e 5a fc ab e8 42 d1 db 35 6e b8 85 b7 27 87 f7 35 cd 18 a8 f b3 60 9b 2c 2d 9b 5e ba 25 6
2 a1 15 6a a7 3b 53 ae e4 5 e1 35 52 e7 82 11 61 aa e df e7 5 db f8 ff 78 3a 2e b1 44 32 36 df
 6b 5e 1c 72 7d b8 86 1c b2 9b e9 7 a6 36 69 a3 0 c3 b6 bf d2 ae 92 af ce 4e 7c 1a ff 69 b0 7d
 86 ad ad d5 25 8d 37 36 b 87 86 70 11 92 6 34 56 4f 2f 4e a5 3c a 1a f6 c3 c3 60 e6 ed b7 72
73 f8 c4 72 de 79 f2 42 22 92 d9 4d 32 76 44 54 4d 34 65 c5 a6 84 7 51 cc d4 10 3b 78 d2 49 c4
 eb 28 1b c9 54 93 80 d3 8e 1f dc b3 4b 68 6d d8 55 7a 5f 1a e1 ba 25 b2 5f 68 b9 c7 bf 65 7f
f 89 c6 5b 2 18 1 d4 76 15 8f b b ce c4 95 41 4f 76 39 af f6 77 55 3d 73 24 1 d0 5c 52 7b f3 f
4 4d a5 19 a9 be b6 34 e7 b4 d2 37 1f 8a 9c 1a df 99 54 54 90 d9 ce 50 93 e8 c3 9a 47 f1 be ea
 ab c3 b9 62 90 aa 17 96 2c f7 6f 76 1d c1 89 ac 50 f3 c2 b3 ab e cc 7d 2a 89 b1 42 d9 e5 d0 e
0 7a f4 77 51 a3 88 12 9a a5 b4 4a 4f 59 b0 95 90 3 38 d2 f9 13 d7 f9 d2 df 1a 6e f5 70 d5 24
d5 a4 59 cf f3 64 5a ce ac cb 64 27 e8 f1 79 2d e3 82 23 9b 40 e8 1d 3d 6c db 36 9e f3 bf 6d 1
6 75 15 31 7 90 d4 f0 90 8f 67 db 95 d8 f8 44 87 5e d5 85 e2 e3 c4 77 79 40 ba 83 88 64 3 d8 9
9 5a 4c f1 d4 be 32 7a 82 db 6e 83 96 2c df 17 f6 87 98 7b c1 53 3f 61 b7 da ea b7 10 fa 7f d6
 7b 19 1 90 5 ba 71 86 6b e4 1f 2b 1f a3 56 c1 71 dd b2 ca 9f 70 3 ed 96 c8 d1 b9 f2 78 0 fb a
0 eb 4c bd a4 2f 41 c 50 78 ff 43 e7 e1 6f d7 7d d6 3e 65 1a e3 26 3b b9 9c 20 59 d8 a8 87 36
2c 55 3d da ab 61 c2 8b 7b 27 0 7e d7 e a7 31 fb 19 4c a1 d8 fa aa e5 82 6e 7d 83 4c 5e 2c e0
f0 83 6 40 ac 1e 3 8c 81 28 b8 5d 7f d6 5 73 26 11 12 ad 76 f0 60 7d 1 63 ed ea 3e 57 27 a9 f9
 44 93 c4 ea b ba 1b bc 1c 74 74 53 72 d0 8 5c 2d 53 f0 41 12 f2 8a a 3f b9 92 aa c3 20 4a 75
39 c8 cd 44 8e c2 ad e6 80 df 76 b2 fb ae a0 89 24 5 70 8a bc 47 e9 7e 8 d7 a6 75 8f 6 26 2 d7
 1 40 a0 68 e5 40 51 4f dc 43 ae f5 39 aa 25 af cb 28 32 59 76 75 6d 54 d0 85 f7 2f 45 53 41 d
5 78 5a f7 43 6d c1 4c fa e6 23 a4 6d 3a d5 99 78 29 5b fb 34 8c bf 13 d 6e 97 c5 88 8e e3 4f
5c 8 b5 3e 50 2 b8 b5 80 5c 9c 18 23 74 bd b6 a7 52 38 6a 17 59 b3 a0 aa f4 b3 6a 14 28 fb 84
d5 b4 4c 95 e9 b2 53 72 45 c9 16 ec ea a3 ce 1 4c 3a 52 15 2c 3f 14 d0 66 7e 99 c1 1b 11 de 2e
 8f 5e e3 d7 7a 4e 57 59 12 7f 7f 13 12 95 9 2c eb 39 41 d 10 9a e0 e0 d8 72 b6 a 77 94 2a 71
e7 be 3f 55 5a e6 c1 19 0 49 3d 72 bf 28 b0 43 25 0 3c e8 7e 85 43 ba 65 de ed 88 78 f4 31 b7
19 23 7a 61 8d 8d e0 f3 e1 4c 1f 7e 72 2a fd fb 31 ee c1 6 7b ec b2 d9 ea 4b a5 8c 68 d3 3d 83
 75 98 e b6 93 f 9d bd b5 12 ae 7d 32 4b cb cb b7 96 1 2 60 12 90 20 42 4a 62 58 3e 45 b4 2a e
0 75 49 c9 5c 94 d3 a3 10 c1 6d 46 7e 6 31 9 73 48 69 22 91 82 42 e5 15 e3 51 d1 d4 d f9 9f 7b
 93 d1 80 5 37 40 47 b2 7a 59 da 81 b8 7e 25 e3 f2 b3 86 f3 c0 ca f7 ca 50 86 55 c7 d7 5e f5 5
7 99 cc 5e b1 ee e7 ba 2f 55 82 9c f8 37 69 f9 51 8 b1 e9 59 62 f4 4d 67 4c 89 52 91 64 59 7c
a8 57 8b b 21 44 71 31 0 5b 38 b3 94 13 b5 9e f4 d6 ec 6a 18 12 1c b2 5f 50 20 54 47 5c fa 28
a4 79 5d ae 37 f0 66 43 e3 38 47 32 b4 85 1b 47 bd c1 16 8d 67 11 c5 3e e aa 75 1c f9 c7 f8 7e
 83 93 c2 8c e5 2e f4 f4 7a d3 43 2a c3 97 fc ad 74 1d ce f5 c4 4d 71 91 37 22 8b a4 5a f9 80
8a f0 94 fb cc e0 9d db 22 7d 2c c5 34 65 f5 e6 92 97 d6 93 21 6 e8 b9 ce 66 86 50 b4 f3 a7 24
 d8 2f 86 13 32 2 8a 6e 71 5b 37 b f0 f9 e8 91 f5 b0 1c fc 4d a1 16 37 2c 74 97 98 35 59 8c f6
 d8 e5 ed c1 2f 6a fe a6 84 64 a1 3d 9e e6 bc 5d 90 1b 6f 49 78 95 26 d3 5f 2c ae 17 65 26 b6
c2 43 c8 f7 9f ae 88 59 d7 95 4d b1 75 e2 2d 46 c5 44 9f 3e 70 1f fb 4d ea 28 a8 52 d7 6d 37 4
8 30 d0 13 55 60 24 8a 45 a4 52 c4 96 10 ee 87 c9 4c ec ee 8b 35 fa ad 6 44 41 8c a5 3a 23 21
a2 37 9e 43 d1 8d b3 67 e6 7d 49 fe 3a 72 9e 79 a8 8b 13 1e e3 aa 58 36 cf e 4c 8d e2 72 41 a
5a 5f 68 a6 bb 2a e1 f7 65 aa 99 ad b7 cc 30 c5 47 4a 5a 36 bc 92 8f 2d 80 41 1 90 6a fb c8 e6
 a4 68 e1 53 44 82 8e 71 25 41 3e 13 de be 5b 2c df 1c e2 df 38 94 59 ce 7d 7d 3a 79 21 2b a5
b7 c2 32 cf 73 36 d5 89 66 11 81 61 46 ae ef a1 ae 9e 39 4e cc b5 63 bd 66 92 74 82 50 f 9 1c
a4 b2 22 6d 46 cb bb 91 92 f1 16 90 56 2 4a 50 d0 6b 2f 17 33 c3 58 2d 72 db 35 c1 b7 17 f9 a0
```

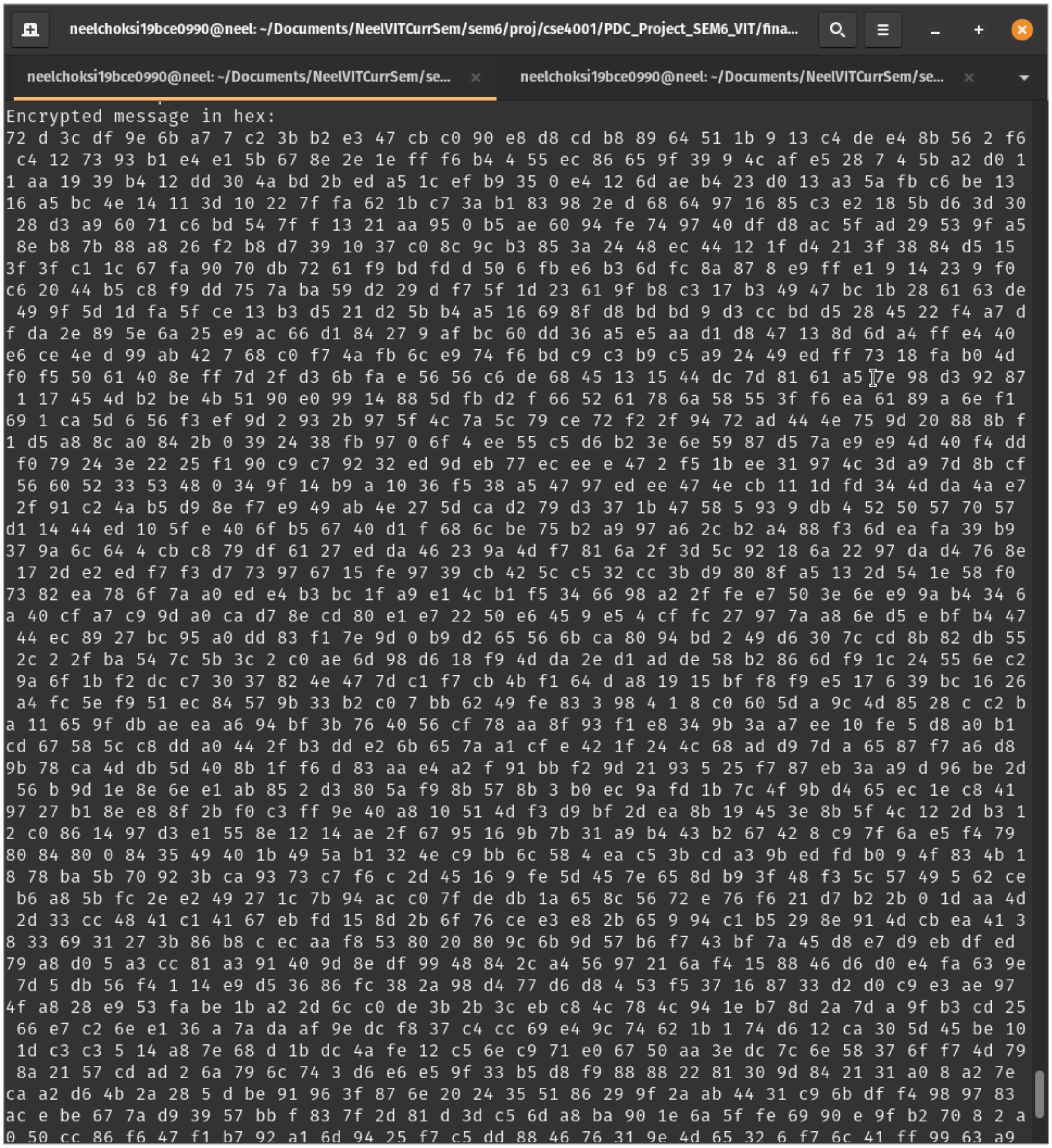


```
 47 c5 d3 0 31 f3 85 dc d a0 6f 1f 65 44 fb 76 13 27 15 73 5c 92 ad 4b e5 88 5 8f 56 b1 db f8
fd 54 8d 4c aa d2 47 d4 58 c4 d0 8a 6a d0 f8 c0 b7 13 7e 42 d4 e1 8f 95 4d 67 65 f8 1c 1e 1b 6
b 17 4a 19 e8 2a 68 f7 c3 27 aa e6 1d 17 d5 4 9c af 96 76 11 a5 11 14 7 d4 53 d 89 83 28 8 c8
ca b4 e3 25 a0 44 62 a1 f4 97 9c 93 ec 1e 6e 87 10 fb 49 72 99 c0 73 aa 7e e8 d8 68 87 7e c0 6
7 dc b5 63 20 26 76 d9 40 8d 97 ef 29 7f fa cf d4 b6 c6 62 bd 8d 8b e5 7c f7 68 8d 66 14 aa 8e
 92 1 87 65 86 a e8 f0 19 31 c5 a2 b6 4e 31 99 e5 c4 93 3e 5 f0 a0 e6 37 23 d e0 2c 6 1c 76 e1
 99 1e 82 49 3 d3 b5 a 8a 41 7a fc f1 d3 e5 f3 d3 1 e0 aa bd 47 d5 b8 49 fd 6b f8 b6 e8 2f 43
30 a6 d5 44 a8 f8 d4 93 d5 99 74 5a e 68 7d 9d b4 97 23 38 7d e3 18 82 9c 94 36 4b d4 9a 55 4e
 b9 3c 2e e5 37 d7 1a 99 5e b9 88 66 b2 fc ff 20 15 49 c0 39 a6 79 7c 88 63 3d 2c c2 a3 77 4f
2c df d4 92 8c aa 2d 20 bc fe 40 2 b6 46 d1 14 c9 8e 1a 72 d b7 2e 8 c3 9 54 66 b5 8e 33 ab c8
 6e 5f 8b 23 59 c2 e5 f5 a7 1b e7 ba ba dc 92 b7 29 85 57 10 f7 6f 9c 30 49 ea d 2f c8 13 b7 7
6 a2 d 3a 51 5f d9 d3 c8 7a 15 9e 14 35 1 5e 77 1c 9c 9d 90 96 99 af bc 1a fb e 37 e9 17 c7 74
 9b 22 c9 c8 93 bf af f0 1a 4 f1 62 2a c 3d 88 cb 7 4e c6 e0 dd 2c 1e 2f dc 64 31 f3 5c 29 82
bd a4 86 4a f8 35 ec 1 58 67 bc fd 5f 9f bc 17 15 d0 b9 24 e4 40 e1 d9 f0 17 48 2c 3f e5 48 8e
 21 e8 57 2a 46
Time taken by parallel nine rounds : 0.012579s
neelchoksi19bce0990@neel:~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fin
alImplementation/parallel_code$ 
```

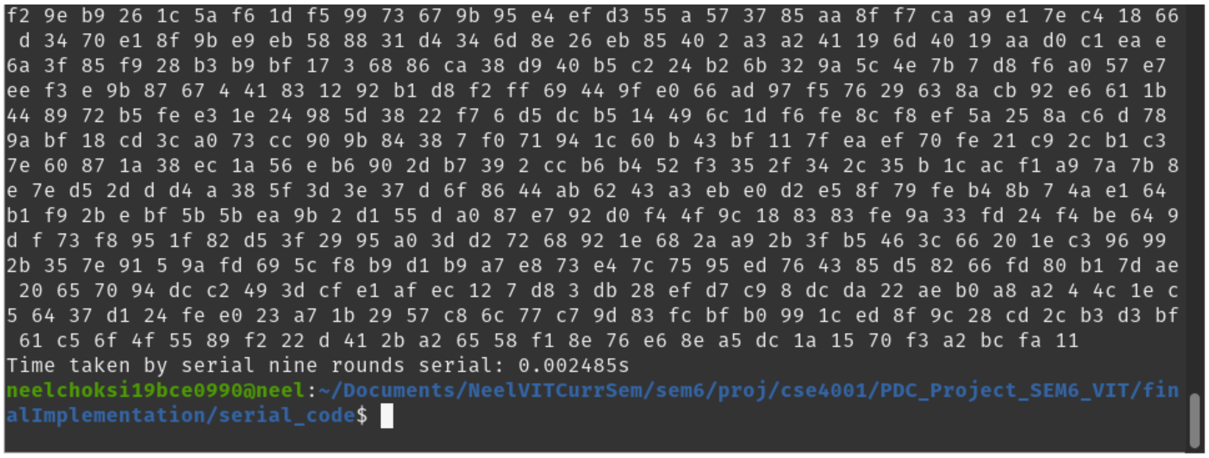
Serial :



```
neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fina...
neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/se...    neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/se...
neelchoksi19bce0990@neel:~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fin
alImplementation/serial_code$ g++ -fopenmp 06_getting_ahead_correctness_serial.cpp
neelchoksi19bce0990@neel:~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fin
alImplementation/serial_code$ ./a.out
Enter the message to encrypt: Sample text for Roma : the novel of ancient Rome / Steven Saylor
.Bibliographic record and links to related information available from the Library of Congress
catalogCopyrighted sample text provided by the publisher and used with permission. May be inco
mplete or contain other coding.CounterChapter OneA Stop on the Salt Route1000 B.C.As they roun
ded a bend in the path that ran beside the river, Lara recognized the silhouette of a fig tree
 atop a nearby hill. The weather was hot and the days were long. The fig tree was in full leaf
, but not yet bearing fruit.Soon Lara spotted other landmarks—an outcropping of limestone besi
de the path that had a silhouette like a man's face, a marshy spot beside the river where the
waterfowl were easily startled, a tall tree that looked like a man with his arms upraised. The
y were drawing near to the place where there was an island in the river. The island was a good
 spot to make camp. They would sleep on the island tonight.Lara had been back and forth along
the river path many times in her short life. Her people had not created the path—it had always
 been there, like the river—but their deerskin-shod feet and the wooden wheels of their handca
rts kept the path well worn. Lara's people were salt traders, and their livelihood took them o
n a continual journey.At the mouth of the river, the little group of half a dozen intermingled
 families gathered salt from the great salt beds beside the sea. They groomed and sifted the s
alt and loaded it into handcarts. When the carts were full, most of the group would stay behin
d, taking shelter amid rocks and simple lean-tos, while a band of fifteen or so of the heartie
r members set out on the path that ran alongside the river.With their precious cargo of salt,
the travelers crossed the coastal lowlands and traveled toward the mountains. But Lara's peopl
e never reached the mountaintops; they traveled only as far as the foothills.
Sample text for Roma : the novel of ancient Rome / Steven Saylor.Bibliographic record and link
s to related information available from the Library of Congress catalogCopyrighted sample text
 provided by the publisher and used with permission. May be incomplete or contain other coding
.CounterChapter OneA Stop on the Salt Route1000 B.C.As they rounded a bend in the path that ra
n beside the river, Lara recognized the silhouette of a fig tree atop a nearby hill. The weath
er was hot and the days were long. The fig tree was in full leaf, but not yet bearing fruit.So
on Lara spotted other landmarks—an outcropping of limestone beside the path that had a silhoue
tte like a man's face, a marshy spot beside the river where the waterfowl were easily startled
, a tall tree that looked like a man with his arms upraised. They were drawing near to the pla
ce where there was an island in the river. The island was a good spot to make camp. They would
 sleep on the island tonight.Lara had been back and forth along the river path many times in h
er short life. Her people had not created the path—it had always been there, like the river—bu
t their deerskin-shod feet and the wooden wheels of their handcarts kept the path well worn. L
ara's people were salt traders, and their livelihood took them on a continual journey.At the m
outh of the river, the little group of half a dozen intermingled families gathered salt from t
he great salt beds beside the sea. They groomed and sifted the salt and loaded it into handcar
ts. When the carts were full, most of the group would stay behind, taking shelter amid rocks a
nd simple lean-tos, while a band of fifteen or so of the heartier members set out on the path
that ran alongside the river.With their precious cargo of salt, the travelers crossed the coas
tal lowlands and traveled toward the mountains. But Lara's people never reached the mountainto
ps; they traveled only as far as the foothills.
padded msg length : 1952
Time taken to perform 9 rounds :0.00077744
```


neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fina...
neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/se...
neelchoksi19bce0990@neel: ~/Documents/NeelVITCurrSem/se...

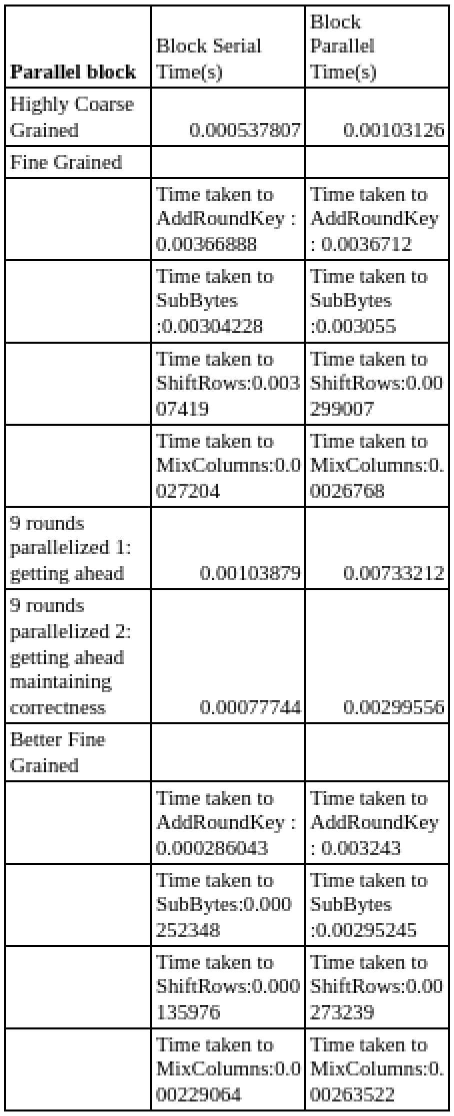
```
Encrypted message in hex:
72 d 3c df 9e 6b a7 7 c2 3b b2 e3 47 cb c0 90 e8 d8 cd b8 89 64 51 1b 9 13 c4 de e4 8b 56 2 f6
 c4 12 73 93 b1 e4 e1 5b 67 8e 2e 1e ff f6 b4 4 55 ec 86 65 9f 39 9 4c af e5 28 7 4 5b a2 d0 1
1 aa 19 39 b4 12 dd 30 4a bd 2b ed a5 1c ef b9 35 0 e4 12 6d ae b4 23 d0 13 a3 5a fb c6 be 13
16 a5 bc 4e 14 11 3d 10 22 7f fa 62 1b c7 3a b1 83 98 2e d 68 64 97 16 85 c3 e2 18 5b d6 3d 30
 28 d3 a9 60 71 c6 bd 54 7f f 13 21 aa 95 0 b5 ae 60 94 fe 74 97 40 df d8 ac 5f ad 29 53 9f a5
 8e b8 7b 88 a8 26 f2 b8 d7 39 10 37 c0 8c 9c b3 85 3a 24 48 ec 44 12 1f d4 21 3f 38 84 d5 15
3f 3f c1 1c 67 fa 90 70 db 72 61 f9 bd fd d 50 6 fb e6 b3 6d fc 8a 87 8 e9 ff e1 9 14 23 9 f0
c6 20 44 b5 c8 f9 dd 75 7a ba 59 d2 29 d f7 5f 1d 23 61 9f b8 c3 17 b3 49 47 bc 1b 28 61 63 de
 49 9f 5d 1d fa 5f ce 13 b3 d5 21 d2 5b b4 a5 16 69 8f d8 bd bd 9 d3 cc bd d5 28 45 22 f4 a7 d
f da 2e 89 5e 6a 25 e9 ac 66 d1 84 27 9 af bc 60 dd 36 a5 e5 aa d1 d8 47 13 8d 6d a4 ff e4 40
e6 ce 4e d 99 ab 42 7 68 c0 f7 4a fb 6c e9 74 f6 bd c9 c3 b9 c5 a9 24 49 ed ff 73 18 fa b0 4d
f0 f5 50 61 40 8e ff 7d 2f d3 6b fa e 56 56 c6 de 68 45 13 15 44 dc 7d 81 61 a5 7e 98 d3 92 87
 1 17 45 4d b2 be 4b 51 90 e0 99 14 88 5d fb d2 f 66 52 61 78 6a 58 55 3f f6 ea 61 89 a 6e f1
69 1 ca 5d 6 56 f3 ef 9d 2 93 2b 97 5f 4c 7a 5c 79 ce 72 f2 2f 94 72 ad 44 4e 75 9d 20 88 8b f
1 d5 a8 8c a0 84 2b 0 39 24 38 fb 97 0 6f 4 ee 55 c5 d6 b2 3e 6e 59 87 d5 7a e9 e9 4d 40 f4 dd
 f0 79 24 3e 22 25 f1 90 c9 c7 92 32 ed 9d eb 77 ec ee e 47 2 f5 1b ee 31 97 4c 3d a9 7d 8b cf
 56 60 52 33 53 48 0 34 9f 14 b9 a 10 36 f5 38 a5 47 97 ed ee 47 4e cb 11 1d fd 34 4d da 4a e7
 2f 91 c2 4a b5 d9 8e f7 e9 49 ab 4e 27 5d ca d2 79 d3 37 1b 47 58 5 93 9 db 4 52 50 57 70 57
d1 14 44 ed 10 5f e 40 6f b5 67 40 d1 f 68 6c be 75 b2 a9 97 a6 2c b2 a4 88 f3 6d ea fa 39 b9
37 9a 6c 64 4 cb c8 79 df 61 27 ed da 46 23 9a 4d f7 81 6a 2f 3d 5c 92 18 6a 22 97 da d4 76 8e
 17 2d e2 ed f7 f3 d7 73 97 67 15 fe 97 39 cb 42 5c c5 32 cc 3b d9 80 8f a5 13 2d 54 1e 58 f0
73 82 ea 78 6f 7a a0 ed e4 b3 bc 1f a9 e1 4c b1 f5 34 66 98 a2 2f fe e7 50 3e 6e e9 9a b4 34 6
a 40 cf a7 c9 9d a0 ca d7 8e cd 80 e1 e7 22 50 e6 45 9 e5 4 cf fc 27 97 7a a8 6e d5 e bf b4 47
 44 ec 89 27 bc 95 a0 dd 83 f1 7e 9d 0 b9 d2 65 56 6b ca 80 94 bd 2 49 d6 30 7c cd 8b 82 db 55
 2c 2 2f ba 54 7c 5b 3c 2 c0 ae 6d 98 d6 18 f9 4d da 2e d1 ad de 58 b2 86 6d f9 1c 24 55 6e c2
 9a 6f 1b f2 dc c7 30 37 82 4e 47 7d c1 f7 cb 4b f1 64 d a8 19 15 bf f8 f9 e5 17 6 39 bc 16 26
 a4 fc 5e f9 51 ec 84 57 9b 33 b2 c0 7 bb 62 49 fe 83 3 98 4 1 8 c0 60 5d a 9c 4d 85 28 c c2 b
a 11 65 9f db ae ea a6 94 bf 3b 76 40 56 cf 78 aa 8f 93 f1 e8 34 9b 3a a7 ee 10 fe 5 d8 a0 b1
cd 67 58 5c c8 dd a0 44 2f b3 dd e2 6b 65 7a a1 cf e 42 1f 24 4c 68 ad d9 7d a 65 87 f7 a6 d8
9b 78 ca 4d db 5d 40 8b 1f f6 d 83 aa e4 a2 f 91 bb f2 9d 21 93 5 25 f7 87 eb 3a a9 d 96 be 2d
 56 b 9d 1e 8e 6e e1 ab 85 2 d3 80 5a f9 8b 57 8b 3 b0 ec 9a fd 1b 7c 4f 9b d4 65 ec 1e c8 41
97 27 b1 8e e8 8f 2b f0 c3 ff 9e 40 a8 10 51 4d f3 d9 bf 2d ea 8b 19 45 3e 8b 5f 4c 12 2d b3 1
2 c0 86 14 97 d3 e1 55 8e 12 14 ae 2f 67 95 16 9b 7b 31 a9 b4 43 b2 67 42 8 c9 7f 6a e5 f4 79
80 84 80 0 84 35 49 40 1b 49 5a b1 32 4e c9 bb 6c 58 4 ea c5 3b cd a3 9b ed fd b0 9 4f 83 4b 1
8 78 ba 5b 70 92 3b ca 93 73 c7 f6 c 2d 45 16 9 fe 5d 45 7e 65 8d b9 3f 48 f3 5c 57 49 5 62 ce
 b6 a8 5b fc 2e e2 49 27 1c 7b 94 ac c0 7f de db 1a 65 8c 56 72 e 76 f6 21 d7 b2 2b 0 1d aa 4d
 2d 33 cc 48 41 c1 41 67 eb fd 15 8d 2b 6f 76 ce e3 e8 2b 65 9 94 c1 b5 29 8e 91 4d cb ea 41 3
8 33 69 31 27 3b 86 b8 c ec aa f8 53 80 20 80 9c 6b 9d 57 b6 f7 43 bf 7a 45 d8 e7 d9 eb df ed
79 a8 d0 5 a3 cc 81 a3 91 40 9d 8e df 99 48 84 2c a4 56 97 21 6a f4 15 88 46 d6 d0 e4 fa 63 9e
 7d 5 db 56 f4 1 14 e9 d5 36 86 fc 38 2a 98 d4 77 d6 d8 4 53 f5 37 16 87 33 d2 d0 c9 e3 ae 97
4f a8 28 e9 53 fa be 1b a2 2d 6c c0 de 3b 2b 3c eb c8 4c 78 4c 94 1e b7 8d 2a 7d a 9f b3 cd 25
 66 e7 c2 6e e1 36 a 7a da af 9e dc f8 37 c4 cc 69 e4 9c 74 62 1b 1 74 d6 12 ca 30 5d 45 be 10
 1d c3 c3 5 14 a8 7e 68 d 1b dc 4a fe 12 c5 6e c9 71 e0 67 50 aa 3e dc 7c 6e 58 37 6f f7 4d 79
 8a 21 57 cd ad 2 6a 79 6c 74 3 d6 e6 e5 9f 33 b5 d8 f9 88 88 22 81 30 9d 84 21 31 a0 8 a2 7e
ca a2 d6 4b 2a 28 5 d be 91 96 3f 87 6e 20 24 35 51 86 29 9f 2a ab 44 31 c9 6b df f4 98 97 83
ac e be 67 7a d9 39 57 bb f 83 7f 2d 81 d 3d c5 6d a8 ba 90 1e 6a 5f fe 69 90 e 9f b2 70 8 2 a
0 50 cc 86 f6 47 f1 b7 92 a1 6d 94 25 f7 c5 dd 88 46 76 31 9e 4d 65 32 6 f7 6c 41 ff 99 63 a9
```



```
f2 9e b9 26 1c 5a f6 1d f5 99 73 67 9b 95 e4 ef d3 55 a 57 37 85 aa 8f f7 ca a9 e1 7e c4 18 66
 d 34 70 e1 8f 9b e9 eb 58 88 31 d4 34 6d 8e 26 eb 85 40 2 a3 a2 41 19 6d 40 19 aa d0 c1 ea e
6a 3f 85 f9 28 b3 b9 bf 17 3 68 86 ca 38 d9 40 b5 c2 24 b2 6b 32 9a 5c 4e 7b 7 d8 f6 a0 57 e7
ee f3 e 9b 87 67 4 41 83 12 92 b1 d8 f2 ff 69 44 9f e0 66 ad 97 f5 76 29 63 8a cb 92 e6 61 1b
44 89 72 b5 fe e3 1e 24 98 5d 38 22 f7 6 d5 dc b5 14 49 6c 1d f6 fe 8c f8 ef 5a 25 8a c6 d 78
9a bf 18 cd 3c a0 73 cc 90 9b 84 38 7 f0 71 94 1c 60 b 43 bf 11 7f ea ef 70 fe 21 c9 2c b1 c3
7e 60 87 1a 38 ec 1a 56 e b6 90 2d b7 39 2 cc b6 b4 52 f3 35 2f 34 2c 35 b 1c ac f1 a9 7a 7b 8
e 7e d5 2d d d4 a 38 5f 3d 3e 37 d 6f 86 44 ab 62 43 a3 eb e0 d2 e5 8f 79 fe b4 8b 7 4a e1 64
b1 f9 2b e bf 5b 5b ea 9b 2 d1 55 d a0 87 e7 92 d0 f4 4f 9c 18 83 83 fe 9a 33 fd 24 f4 be 64 9
d f 73 f8 95 1f 82 d5 3f 29 95 a0 3d d2 72 68 92 1e 68 2a a9 2b 3f b5 46 3c 66 20 1e c3 96 99
2b 35 7e 91 5 9a fd 69 5c f8 b9 d1 b9 a7 e8 73 e4 7c 75 95 ed 76 43 85 d5 82 66 fd 80 b1 7d ae
 20 65 70 94 dc c2 49 3d cf e1 af ec 12 7 d8 3 db 28 ef d7 c9 8 dc da 22 ae b0 a8 a2 4 4c 1e c
5 64 37 d1 24 fe e0 23 a7 1b 29 57 c8 6c 77 c7 9d 83 fc bf b0 99 1c ed 8f 9c 28 cd 2c b3 d3 bf
 61 c5 6f 4f 55 89 f2 22 d 41 2b a2 65 58 f1 8e 76 e6 8e a5 dc 1a 15 70 f3 a2 bc fa 11
Time taken by serial nine rounds serial: 0.002485s
neelchoksi19bce0990@neel:~/Documents/NeelVITCurrSem/sem6/proj/cse4001/PDC_Project_SEM6_VIT/fin
alImplementation/serial_code$ 
```

## 6.2. Output – in terms of performance metrics

The parallelized code in coarse grained approach runs slower than the serial code due to parallel overheads.In the Fine Grained approach, the time taken by AddRoundKey,SubBytes operation in parallel is almost equivalent to that of serial . The time taken by ShiftRows and MixColumns in parallel is less than that in serial . The 9 Rounds irregular parallelism approach takes longer to execute than in serial .In the Better fine grained approach where a thread is given more work , time taken to execute AddRoundKey,SubBytes,ShiftRows, MixColumns in parallel is more than that in serial.



| **Parallel block** | Block Serial Time(s) | Block Parallel Time(s) |
|---|---|---|
| Highly Coarse Grained | 0.000537807 | 0.00103126 |
| Fine Grained | | |
| | Time taken to AddRoundKey : 0.00366888 | Time taken to AddRoundKey : 0.0036712 |
| | Time taken to SubBytes :0.00304228 | Time taken to SubBytes :0.003055 |
| | Time taken to ShiftRows:0.00307419 | Time taken to ShiftRows:0.00299007 |
| | Time taken to MixColumns:0.0027204 | Time taken to MixColumns:0.0026768 |
| 9 rounds parallelized 1: getting ahead | 0.00103879 | 0.00733212 |
| 9 rounds parallelized 2: getting ahead maintaining correctness | 0.00077744 | 0.00299556 |
| Better Fine Grained | | |
| | Time taken to AddRoundKey : 0.000286043 | Time taken to AddRoundKey : 0.003243 |
| | Time taken to SubBytes:0.000252348 | Time taken to SubBytes :0.00295245 |
| | Time taken to ShiftRows:0.000135976 | Time taken to ShiftRows:0.00273239 |
| | Time taken to MixColumns:0.000229064 | Time taken to MixColumns:0.00263522 |

Table 1. Serial vs Parallel execution times of blocks of code that are parallelized.

## 6.3. Performance comparison with existing works

The highly coarse grained approach provides the best speedup amongst all the approaches, this approach was suggested earlier in paper [2] . Our approach of getting ahead in 9 rounds proved to be the faster approach than the fine grained and better fine grained approach suggested in paper [3] .

|  | T(parallel) in s | T(serial) in s | speedup |
|---|---|---|---|
| highly coarse grained | 0.002232 | 0.001536 | 0.688172043 |
| fine grained | 0.043992 | 0.003132 | 0.07119476268 |
| getting ahead of rounds | 0.028421 | 0.002281 | 0.08025755603 |
| getting ahead maintaining correctness | 0.012579 | 0.002485 | 0.1975514747 |
| better fine grained | 0.022922 | 0.003072 | 0.134019719 |

Table 2. Speedup for 5 approaches to parallelise AES Algorithm.

# CONCLUSION AND FUTURE DIRECTIONS

## 7. Conclusion and Future Directions

We implemented 5 approaches of parallelizing AES which involved a coarse grained approach where the data blocks were encrypted fully by a thread, fine grained approaches where the internal operations of AES were encrypted by two threads and four threads , we also implemented our approach of encrypting the 9 rounds by executing the threads independently from which we could identify the data dependency and control dependency , since the approach was incorrect we implemented another approach which utilized the partial independence of the data identified in the previous wrong approach to parallelize most of the parts of a round and made moving ahead in rounds possible.

AES in parallel could be deployed in the iot devices as suggested in the paper [5] , we could implement a simulated environment . We would like to integrate the AES parallel algorithms and Modular Exponentiation which is used in key exchange as shown in paper [12]

We would also like to try an another approach of parallelizing the 9 rounds which includes saving the result after each operation so that the threads can be independently executed and waiting and locking functionality could be used to make the thread wait till the other thread evaluates the result for that iteration , locking can be used to lock the state when the thread from previous iteration is writing to the state and the thread on another iteration wants to access the state.

# REFERENCES

## 8. References

[1] C. Duta, G. Michiu, S. Stoica and L. Gheorghe, "Accelerating Encryption Algorithms Using Parallelism," 2013 19th International Conference on Control Systems and Computer Science, 2013, pp. 549-554, doi: 10.1109/CSCS.2013.92.https://ieeexplore.ieee.org/document/6569318

[2] N. -P. Tran, M. Lee, S. Hong and S. -J. Lee, "Parallel Execution of AES-CTR Algorithm Using Extended Block Size," 2011 14th IEEE International Conference on Computational Science and Engineering, 2011, pp. 191-198, doi: 10.1109/CSE.2011.43.http://ieeexplore.ieee.org/document/6062872

[3] R. Ketata, L. Kriaa, L. A. Saidane and G. Chalhoub, "Detailed analysis of the AES CTR mode parallel execution using OpenMP," 2016 International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN), 2016, pp. 1-9, doi: 10.1109/PEMWN.2016.7842901.http://ieeexplore.ieee.org/document/7842901

[4] J. D. Dalal, S. S. Dayala and N. Shah, "Optimized AES algorithm using Galois field multiplication and parallel key scheduling," 2012 1st International Conference on Emerging Technology Trends in Electronics, Communication & Networking, 2012, pp. 1-5, doi: 10.1109/ET2ECN.2012.6470044.http://ieeexplore.ieee.org/document/6470044

[5] Ritambhara, A. Gupta and M. Jaiswal, "An enhanced AES algorithm using cascading method on 400 bits key size used in enhancing the safety of next generation internet of things (IOT)," 2017 International Conference on Computing, Communication and Automation (ICCCA), 2017, pp. 422-427, doi: 10.1109/CCAA.2017.8229877.http://ieeexplore.ieee.org/document/8229877

[6] A. Barnes, R. Fernando, K. Mettananda and R. Ragel, "Improving the throughput of the AES algorithm with multicore processors," 2012 IEEE 7th International Conference on Industrial and

Information Systems (ICIIS), 2012, pp. 1-6, doi: 10.1109/ICIInfS.2012.6304791.https://ieeexplore.ieee.org/document/6304791

[7] Tariq Sadiq, A., & Hadi Faisal, F. (2016). Modification AES algorithm based on Extended Key and Plain Text. Journal Of Advanced Computer Science And Technology Research, 5(4). Retrieved from http://www.sign-ific-ance.co.uk/index.php/JACSTR/article/view/1105

[8]Xinmiao Zhang and Keshab K.Parhi" Approaches for the advanced encryption Standard Algorithm" published by the ieee computer society,1531-636x/12/$10.00 © 2002 ieee.

[9] Narayanan, Manikandan & Subha, Srinivasan. (2018). Parallel AES algorithm for performance improvement in data analytics security for IoT. International Journal of Networking and Virtual Organisations. 18. 112. 10.1504/IJNVO.2018.10012669.

[10] Shi, Jingang & Wang, Shoujin & Sun, Limei. (2020). A Parallel AES Encryption Algorithms and Its Application. 10.1007/978-981-15-2568-1_24.
[11] Navalgund, Siddalingesh & Desai, Akshay & Ankalgi, Krishna & Yamanur, Harish. (2013). Parallelization of AES algorithm using OpenMP. Lecture Notes on Information Theory. 1. 144-147. 10.12720/lnit.1.4.144-147.

[12]Xinmiao Zhang and Keshab K.Parhi" Approaches for the advanced Q. Hu, M. Duan, Z. Yang, S. Yu and B. Xiao, "Efficient Parallel Secure Outsourcing of Modular Exponentiation to Cloud for IoT Applications," in IEEE Internet of Things Journal, vol. 8, no. 16, pp. 12782-12791, 15 Aug.15, 2021, doi: 10.1109/JIOT.2020.3029030.

[13] Rohit C.,Leonardo D.,Dave K., Dror M.,Jeff M., & Ramesh M., & Streefkerk, R. (2000). Parallel Programming in OpenMP . Morgan Kaufmann Publishers.

[14]Almuhammadi, Sultan & Alhejri, Ibraheem. (2017). A comparative analysis of AES common modes of operation. 1-4. 10.1109/CCECE.2017.7946655.

# APPENDIX

Appendix-A.

Github link for the screenshots of output of all approaches and serial,parallel code for all approaches. https://github.com/NeelChoksi/sem6_CSE4001_pdc