

# **PARALLELIZED IMAGE ENCRYPTION USING DES**

## **A PROJECT REPORT**

**Submitted by**

**S Patrick Raja (20BCE1058)**  
**Altrin Lloyd Hudson D (20BCE1250)**  
**Dinesh Ragav N (20BCE1723)**

Course Code: CSE 4001  
Course Title: Parallel Distributed Computing

*In partial fulfillment for the award of the degree of*

**B. Tech**  
**in**  
**Computer Science and Engineering**

Under the guidance  
**of Dr. Jaisakthi S M Associate Professor,**  
**SCOPE, VIT, Chennai**



**VIT<sup>®</sup>**  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

## **1. Abstract**

The Data Encryption Standard is the most commonly used cipher and is a symmetric key block encryption algorithm. DES was a cryptosystem widely used to protect secret data transmissions. This project uses the -DES algorithm for image file encryption and decryption. The software implementation results are produced in Python. A detailed analysis explains the test results and the proposed plan is capable of withstanding evolving attacks on image file transfer security. The collected results show that the DES algorithm can be used as a very secure algorithm.

## **2. Motivation**

Parallelism is used to speed up the execution of arbitrary program code. A major part of parallelism is being able to decide which parts of the program to parallelize. In image processing, an image can be read into memory as an array of data, and operations on these pixels or sets of pixels are typically parallelizable. The task needs to be analyzed to assess whether this parallelism is significant enough to warrant execution. If the task is computationally intensive, we recommend running it in parallel.

If tasks are isolated and require less communication overhead, they can be parallelized.

### **3. Objectives**

This project fully implements the task and proposes and implements how to parallelize it across distributed systems. This project uses image encryption as a test and benchmark algorithm. Images are uploaded by users for encryption or decryption. The encryption and decryption process itself is performed using the triple DES (Data Encryption Standard) algorithm implemented using Python.

### **4. Introduction**

Data security is now an important aspect of digital data transmission. Increase the importance and sensitivity of this information, such as account information, military information, sensitive data such as medical records, and multimedia data such as images, audio, or video. For this reason, satisfactorily successful cryptographic algorithms are required to protect these types of information transmissions from disclosure by unauthorized users. Then again in our lives the pace of innovation and improvement in the area of computer processing speed has turned out to be faster and faster. These extensions facilitate threats and attacks on information or data, gradually uncovers those secrets and loads a vast array of tests to fulfill its duty to protect communications. Best approach of security assurance is Encryption. For changing input image into another image, the encryption system used many techniques. So that changed image is difficult to understand by another unauthorized person and to maintain the secret of images between clients. Another advantage of encryption technique is that you can't access the image information without a decryption key. Main application of image encryption is multimedia framework. Here we use an

established encryption algorithm which is the Data Encryption Standard (DES). DES is a cryptosystem commonly used to protect marked information transfers. DES is a symmetric key cryptosystem, using the same secret key, which is used only for both encryption and decryption processes. Many algorithms retain DES as the core design of the cryptographic design. Upgrading DES security improves the security of other algorithms such as Triple-DES and IDEA.

In today's world almost all digital services like internet communication, medical and military imaging systems, multimedia system needs a high level security. There is a need for security level in order to safely store and transmit digital images containing critical information. This is because of the faster growth in multimedia technology, internet and cell phones. Therefore there is a need for image encryption techniques in order to hide images from such attacks. In this system we use Triple DES (Data Encryption Standard) in order to hide image. Such Encryption technique helps to avoid intrusion attacks.

## **5. Literature Survey**

Literature survey is being carried out with respect to the topics of different data encryption methods, parallel approaches, image encryption and decryption methods, and cryptography.

**R. Yadavi, M. Beg, and M. Tripathi [1]** in their research paper proposed literature review using multimedia data taking an input in the form of image

for encryption using different techniques and also introduced more information about image encryption with its merits and demerits.

**R. Pakshwar, V. Kumar Trivedi and V. Richhariya [2]** in their research presented different image encryption techniques over the survey of twenty five different research papers. They also presented the importance of multimedia data and encryption using those data. Basic element for the encryption is the pixel of many methods.

**J. Ahmad, S. Oun Hwang and A. Ali [3]** in this they have made the analysis of Advanced Encryption Standard (AES), scheme of compression friendly encryption, Chaotically Scheme of Coupled Chaotic Map Encryption and a Bernoulli Map Based Encryption Scheme.

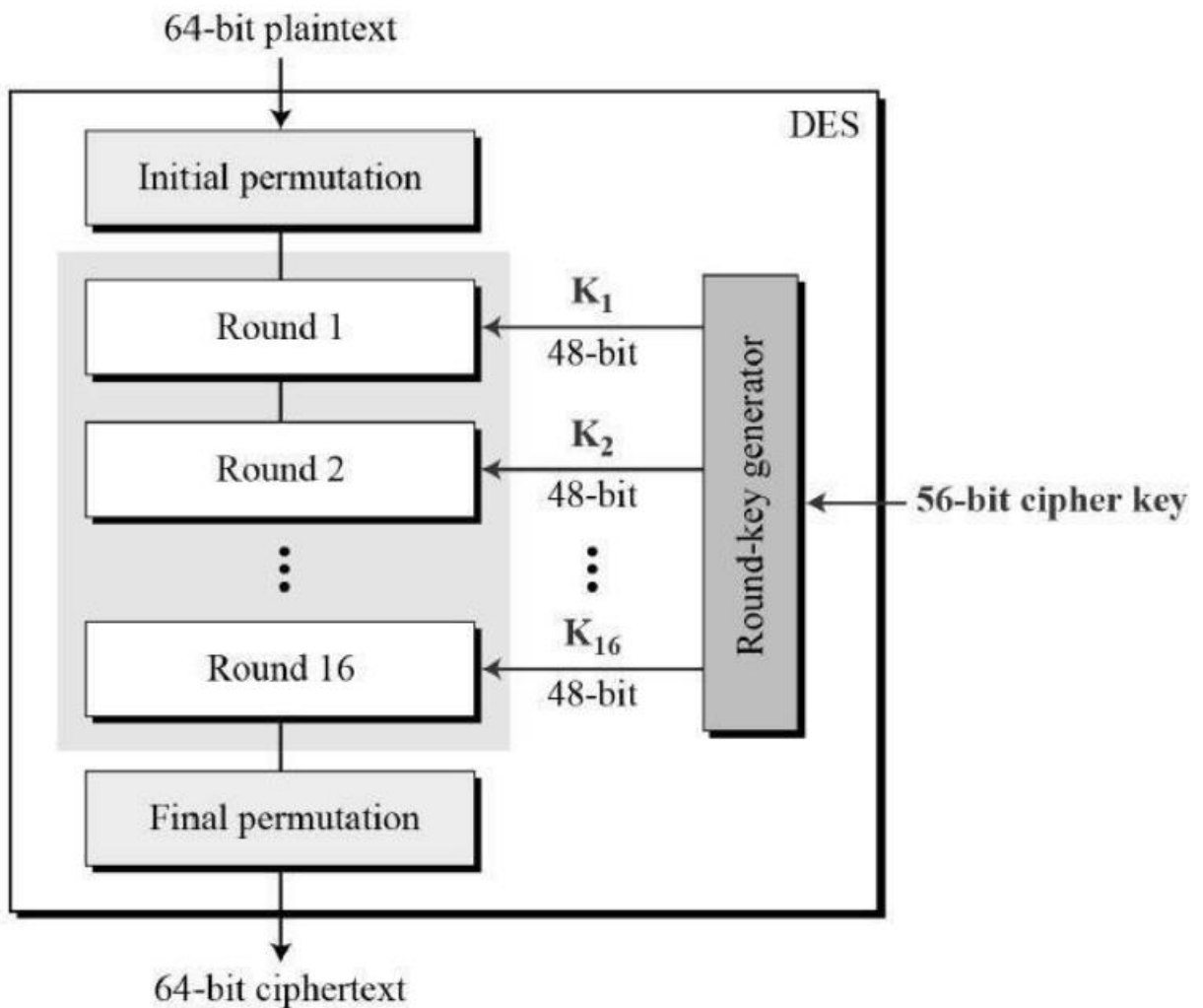
**Sivaguru J, Manikadan G, Karthikeyan S, and Sairamn [4]** provided a parallel method for cryptography in this article. You mentioned that running in parallel also speeds up encryption and decryption. They also used the concept of "slice and merge" to perform parallel ciphers and proved that outperformed many cipher algorithms.

**Salem Sheriff Elfurd [5]** proposed the use of cryptography in providing security services and a number of related applications of his cryptography in data security. He also presented information on encoding data using parallel paths and his linear Fibonacci formalism,

**Ravishankar K C and Venkatesh Murthy M G [6]**, and did image encoding in the region permutations. This method reflects the clutter in the visibility of the image. Randomness is the main purpose of and it spoils the big picture. This method can be parallelized using existing hardware.

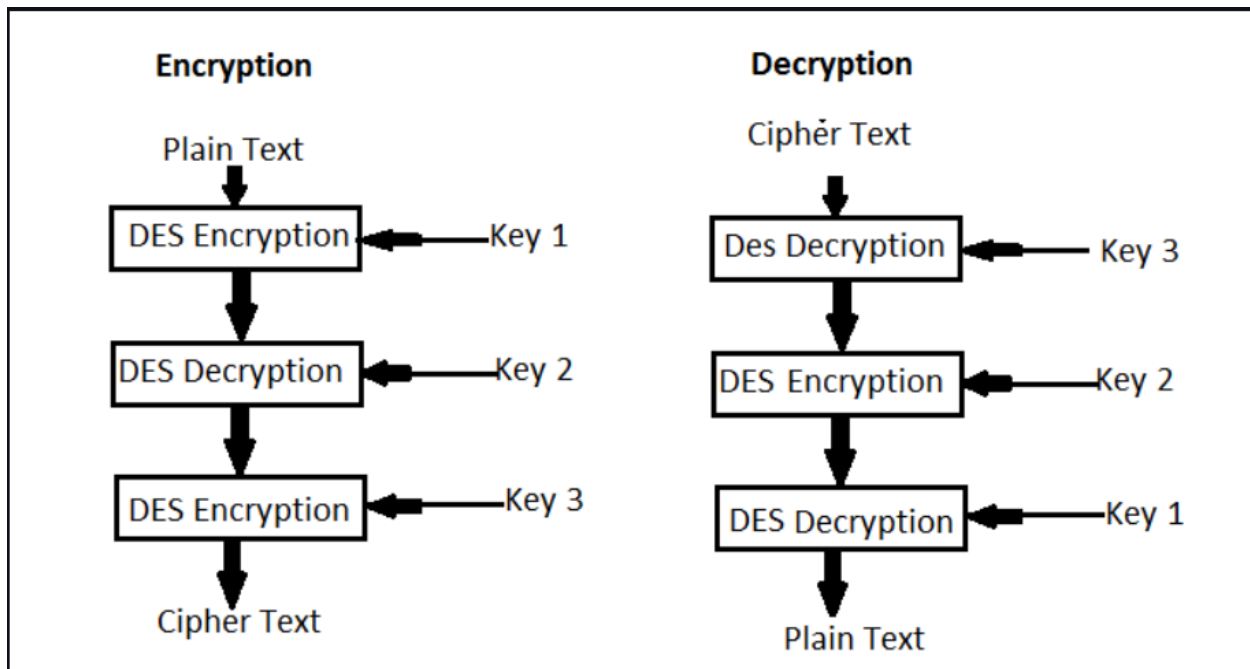
## 6. Proposed Methodology

### 6.1. Architecture Diagram

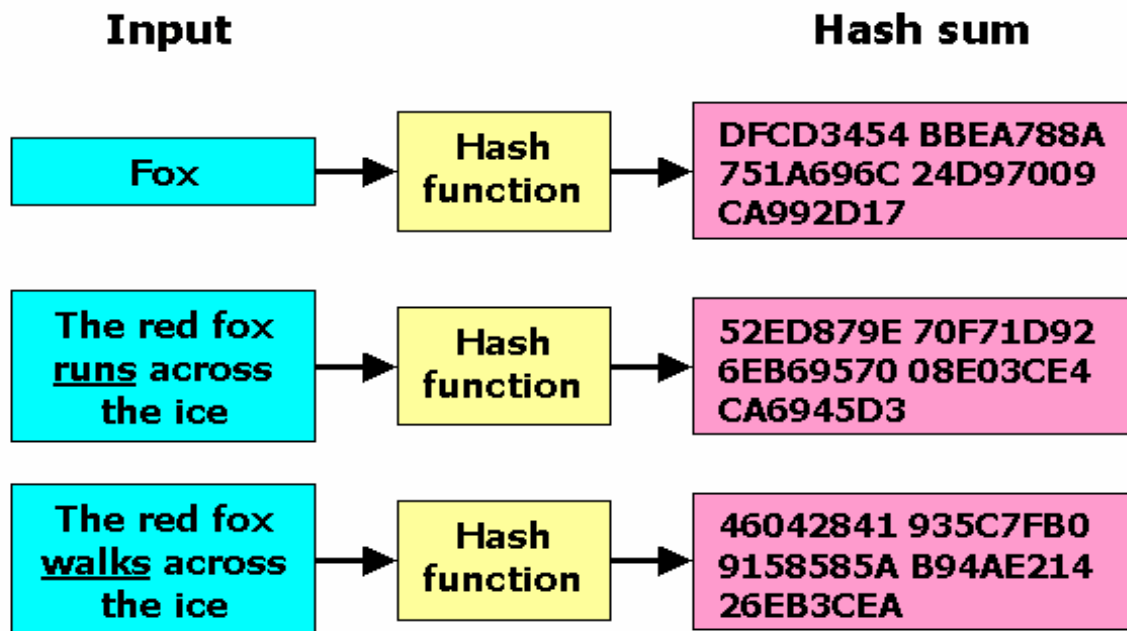


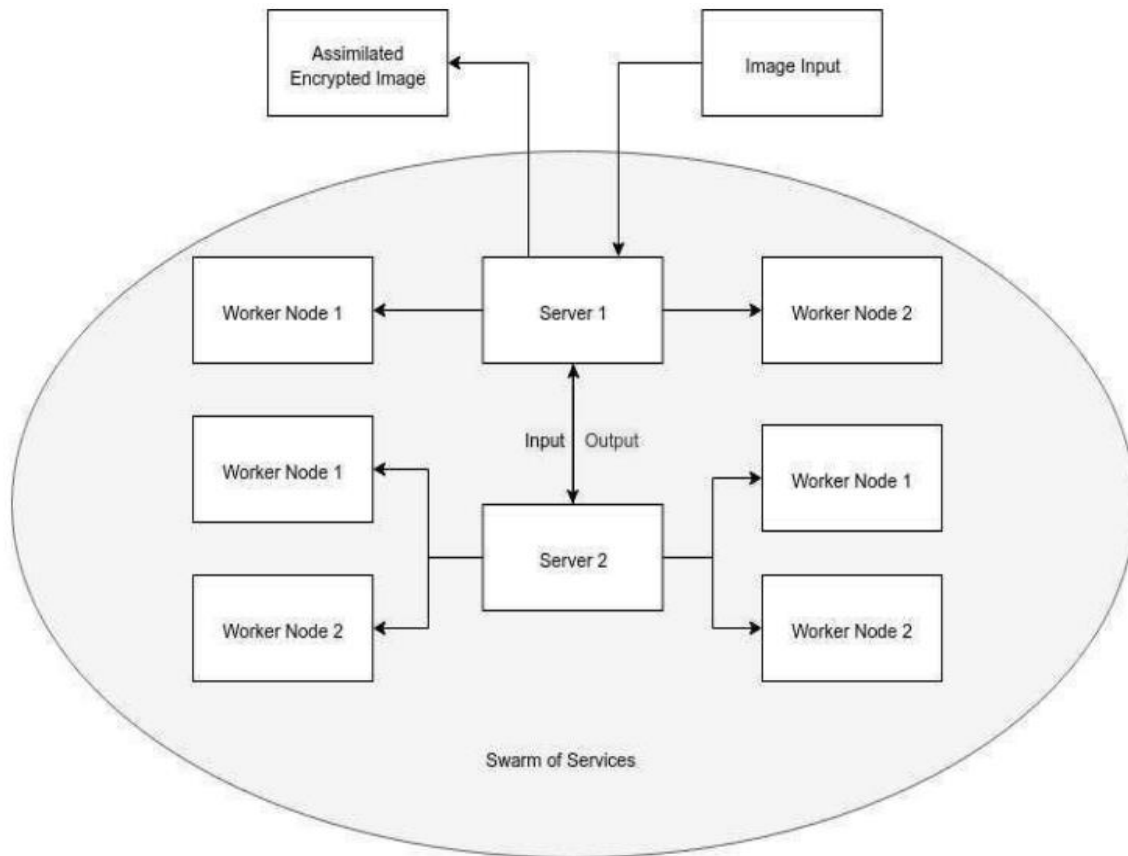
## 6.2. Module Diagram

### Triple DES:



### MD5 Hashing:





### 6.3. Experimental Setup

#### Requirements:

- Pycharm
- Des3 from crypto.cipher
- Md5 from hashlib
- Pycryptodome
- An image to encrypt and decrypt.

#### 6.4. Procedure

DES relies on the 2 basic terms of cryptography: substitution (also known as confusion) and transposition (also known as diffusion). DES consists of sixteen steps, every of that is named as a spherical. Transportation and substitution



steps are performed by every spherical. The broad-level steps in DES. 1. Within the opening, a block of 64-bit text is being handed over for the initial Permutation perform. 2. In the above step, initial permutation (IP) is done on the plain text. 3. Now the permuted block is made into two halves namely, Right and Left plain Text (RPT, LPT). 4. Currently every LPT and RPT to travel through sixteen rounds of cryptography method. 5. Finally, the split Right and Left plain text square measures are combined and forms a permutation which is final and is performed on one combined block. 6. The results of this method produce sixty four bit cipher text.

### **Encryption:**

The input image data that is taken as plain text is converted into cipher text in a parallel way by using threads. By reading the input image it is basically divided into so many parts. Each part have same size and total number of parts that are divided equals the total number of threads that are created. In the next step we allocate each thread to each part of input image that is divided. Now we perform the encryption part of DES algorithm on each image part which the whole process is done in a parallel way. After this step the output that is obtained by encryption of each image part is saved. All the parts of the image are combined to get the desired output. In order to do the encryption process in parallel way here we use thread that increases the performance of the process and the tasks to work in a parallel way.

The approach we followed here i.e in a parallel way is compared with performance of the process in a sequential way. From that we can performance of a process in a parallel way is better than performance in a sequential way.

### **Pseudo code for encryption of the image:**

Start

**Step 1:** Read input image.

**Step 2:** Split the image into many parts.

**Step 3:** For equal number of parts the corresponding equal number of threads are created.

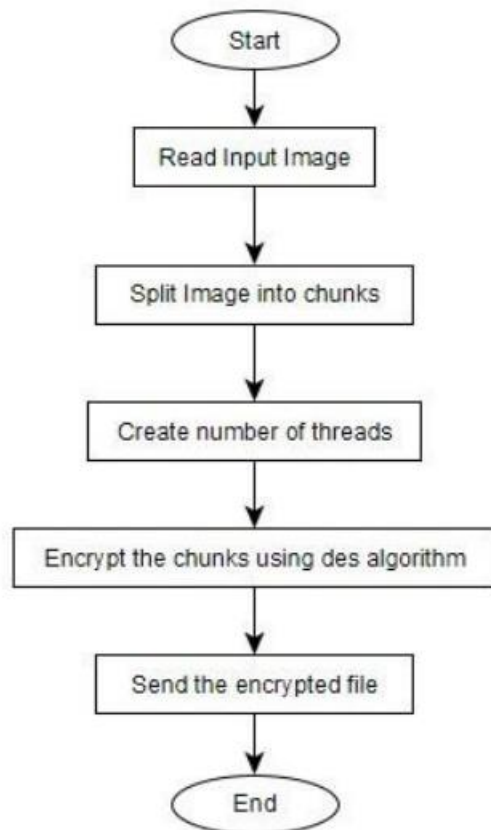
**Step 4:** Allocate each chunk to each thread that are created for encryption with the key.

**Step 5:** Now we will apply DES algorithm to all chunks that are allocated for encryption process.

**Step 6:** After the encryption of all chunks, we will combine all the chunks to single output.

**Step 7:** The total time that is taken for encryption process is noted.

**Step 8:** The final encrypted image is stored. Stop.



**Decryption:**

The Decryption process is defined as the inverse process to the Encryption process where the input image is taken as Encrypted image to get the output as original image. Now we perform the decryption part of DES algorithm on each image encryption part which the whole process is done in a parallel way. Here we allocate each thread to each encrypted part to do tasks in a parallel way. By using threads, we decrypt an encrypted image. The process follows the methodology as same as encrypted process

**Pseudo code for decryption of an image:**

Start

**Step 1:** Read the encrypted image

**Step 2:** Split the encrypted image into number of parts that are called chunks

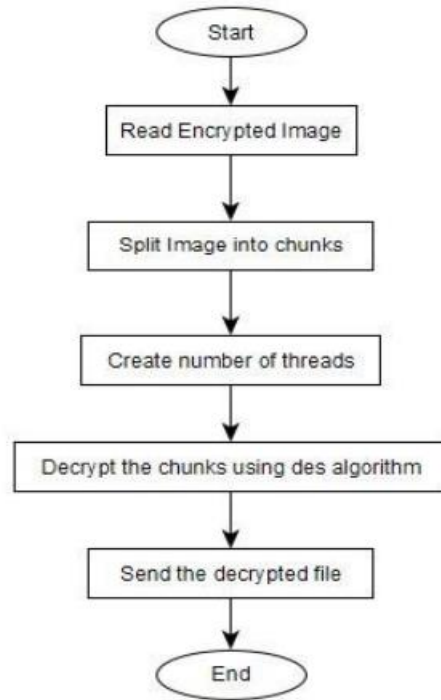
**Step 3:** Allocate each chunk to each thread that are created for decryption with the key.

**Step 4:** Now we will apply DES algorithm to all chunks that are allocated for decryption process.

**Step 5:** After the decryption of all chunks we will combine all the chunks to single output.

**Step 6:** The final decrypted image is stored.

Stop.



## 6.5. Result and Discussion

### Advantages

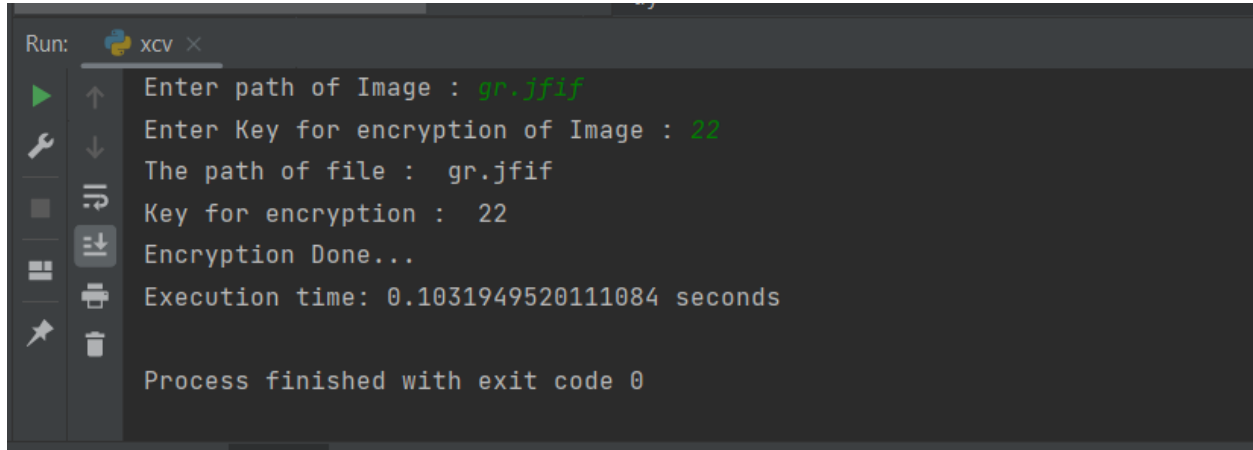
- The image can only be viewed by the receiver as the image is encrypted using Triple DES and the key is only known to the sender and receiver.
- Since the image is encrypted using Triple DES, it is more secure than DES.
- Since the key is entered by the sender and receiver and is not stored in the database, it makes the encryption and decryption more secure.

### Disadvantages

- The file size to be transmitted becomes large since it contains encrypted data.
- Since the file size is huge it can be suspected to contain some critical information.

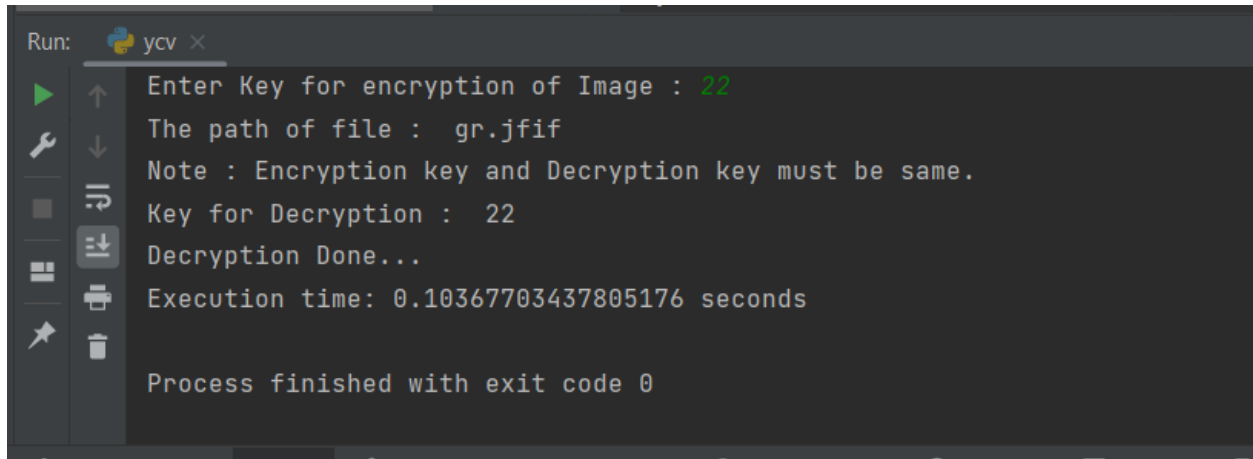
## Output Screenshots:

### Sequential Encryption:



```
Run: xcv
Enter path of Image : gr.jfif
Enter Key for encryption of Image : 22
The path of file : gr.jfif
Key for encryption : 22
Encryption Done...
Execution time: 0.1031949520111084 seconds
Process finished with exit code 0
```

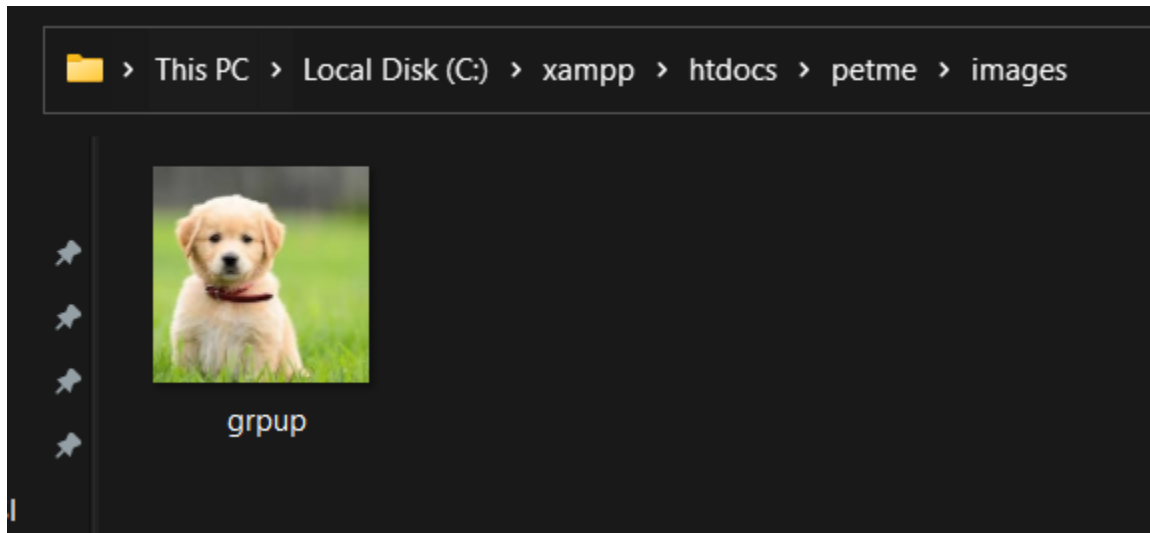
### Sequential Decryption:



```
Run: ycv
Enter Key for encryption of Image : 22
The path of file : gr.jfif
Note : Encryption key and Decryption key must be same.
Key for Decryption : 22
Decryption Done...
Execution time: 0.10367703437805176 seconds
Process finished with exit code 0
```

## Parallel execution:

### Image before encryption and path :



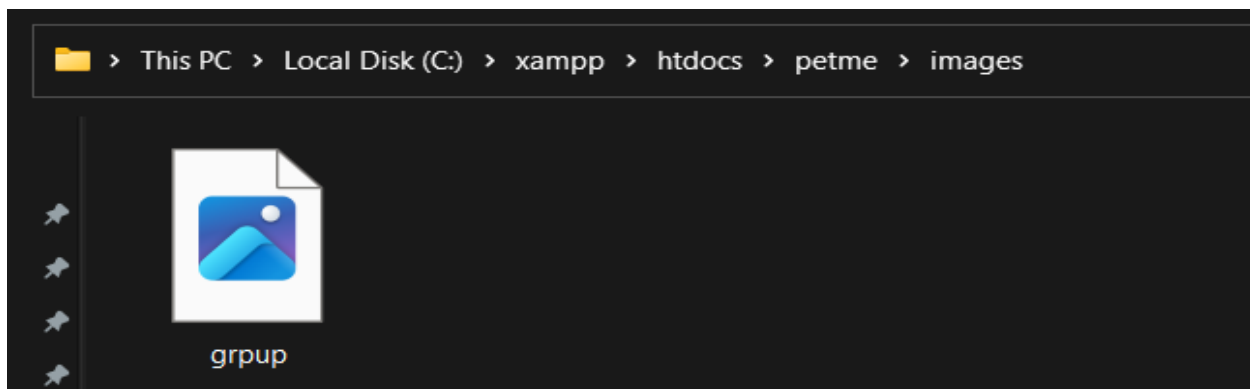
### Output by executing the parallel program using triple DES algorithm:

### Output while encrypting the same file:

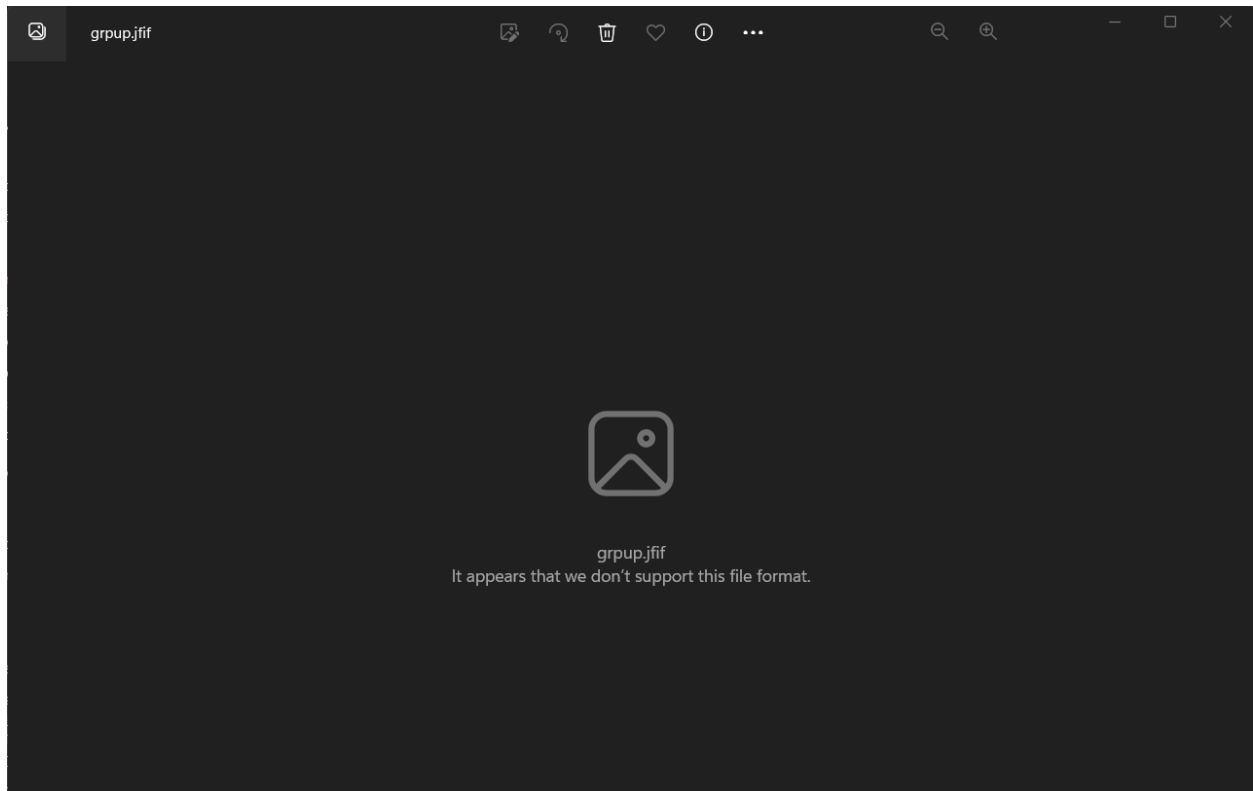
```
main x
C:\Python\Python38\python.exe "G:\documents\sem 5 FALL SEM\PDC\Python_DES3-main\Python_DES3-main\main.py"
Choose operation to be done:
  1- Encryption
  2- Decryption
Your Choice: 1
File path: grpup.jpg
TDES key: patrick
Encryption done!
Operation complete!
Execution time: 0.008994340896606445 seconds

Process finished with exit code 0
```

### Image after encryption:



## Output while decrypting the same file:

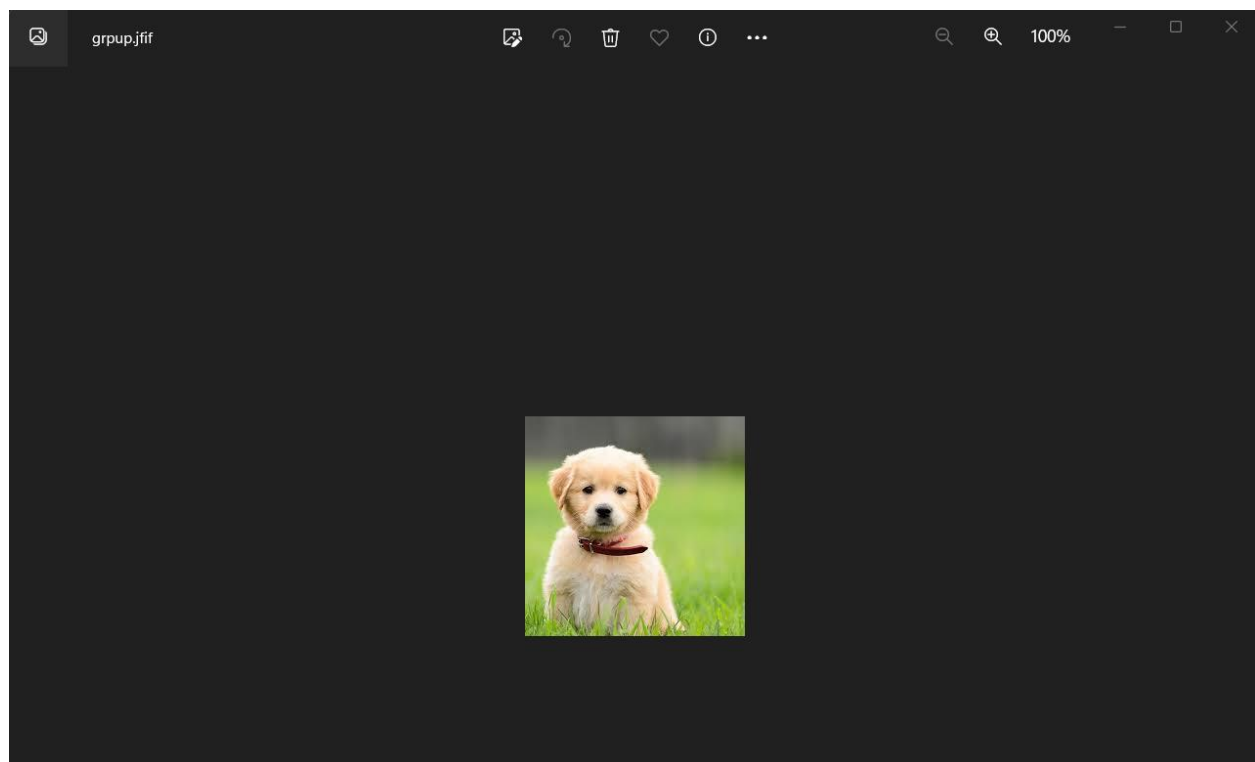
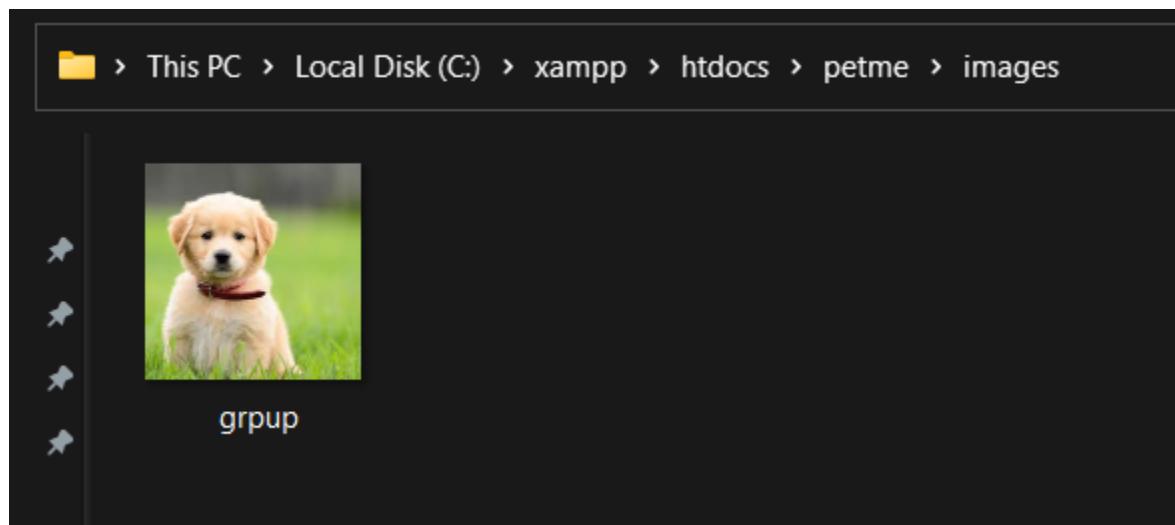


## Output while decrypting the same file:

```
main x
C:\Python\Python38\python.exe "G:\documents\sem 5 FALL SEM\PDC\Python_DES3-main\Python_DES3-main\main.py"
Choose operation to be done:
  1- Encryption
  2- Decryption
Your Choice: 2
File path: grpup.jfif
TDES key: patrick
Decryption done!
Operation complete!
Execution time: 0.01200103759765625 seconds

Process finished with exit code 0
```

## Image after decrypting:

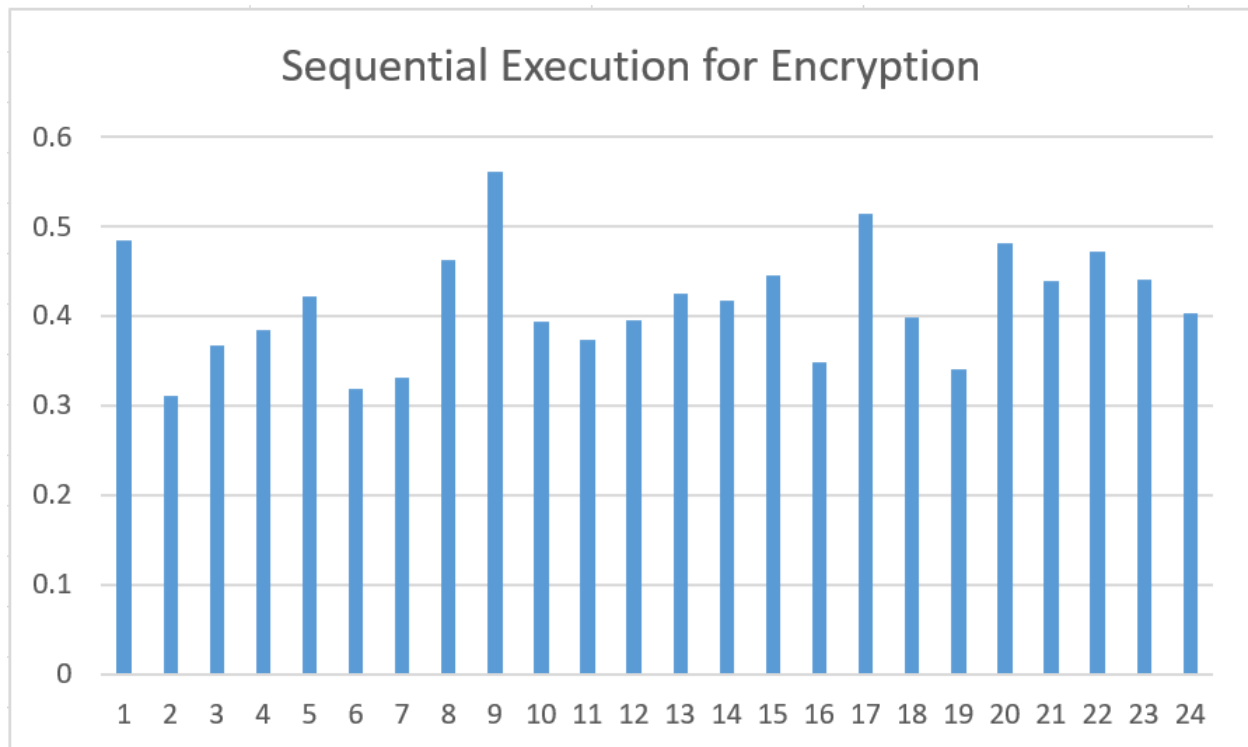




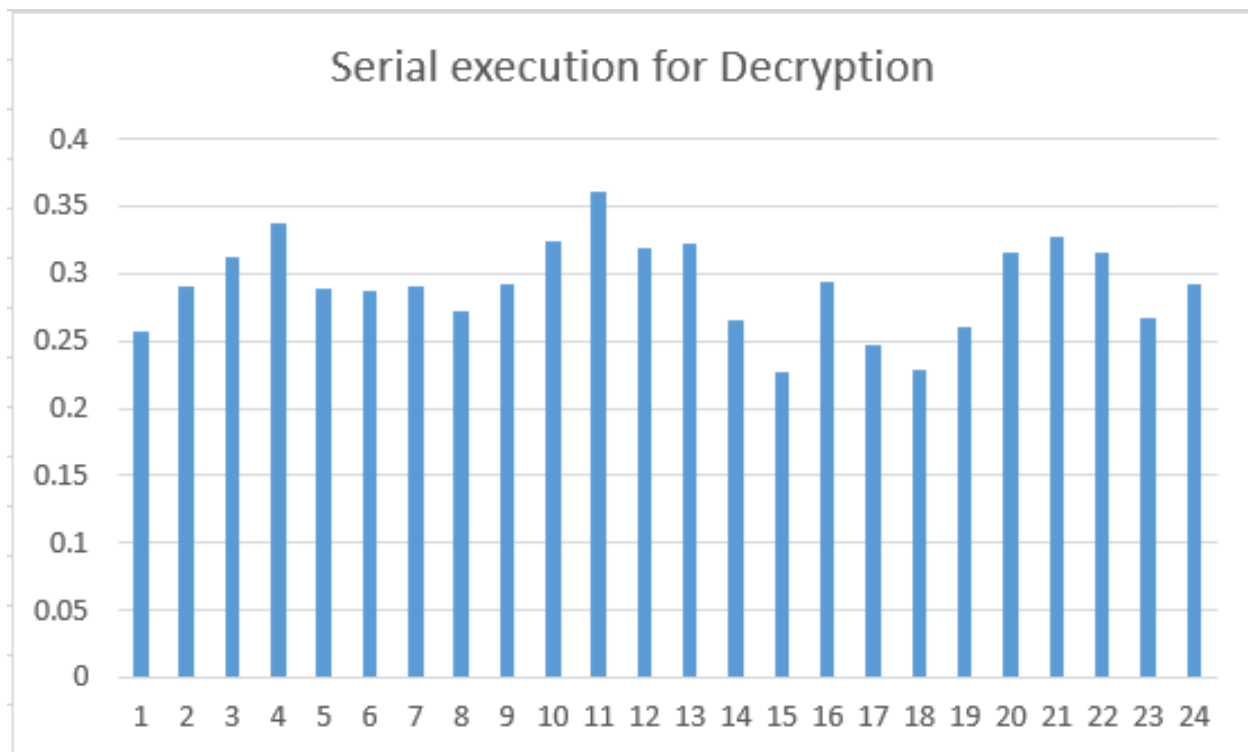
### Representation of the execution time:

Sequential execution		Parallel Execution			Speed up for encryption process	Speed up for decryption
Encryption	Decryption	Encryption	Decryption			
0.4842	0.2573	0.0113	0.0286		42.84955752	8.996503497
0.3102	0.291	0.0223	0.0594		13.9103139	4.898989899
0.3671	0.3127	0.0122	0.0213		30.09016393	14.68075117
0.3841	0.3371	0.0214	0.0203		17.94859813	16.60591133
0.4215	0.2896	0.0217	0.0124		19.42396313	23.35483871
0.319	0.2881	0.0124	0.0214		25.72580645	13.46261682
0.3317	0.2914	0.0159	0.0107		20.86163522	27.23364486
0.4621	0.2716	0.0167	0.0192		27.67065868	14.14583333
0.5621	0.2921	0.0109	0.0347		51.56880734	8.417867435
0.3945	0.324	0.0197	0.0271		20.02538071	11.95571956
0.3741	0.3612	0.035	0.0568		10.68857143	6.35915493
0.3958	0.3191	0.0212	0.0236		18.66981132	13.52118644
0.4258	0.3227	0.0246	0.0245		17.30894309	13.17142857
0.4171	0.2653	0.0231	0.0125		18.05627706	21.224
0.4462	0.227	0.0217	0.0225		20.56221198	10.08888889
0.3493	0.2935	0.0251	0.0395		13.91633466	7.430379747
0.514	0.2471	0.0271	0.0314		18.96678967	7.869426752
0.3983	0.2294	0.0194	0.0195		20.53092784	11.76410256
0.3412	0.2612	0.0173	0.0325		19.72254335	8.036923077
0.4812	0.3164	0.0281	0.0268		17.12455516	11.80597015
0.4398	0.3279	0.0245	0.0172		17.95102041	19.06395349
0.4725	0.3154	0.0257	0.0268		18.38521401	11.76865672
0.4413	0.2671	0.0296	0.0413		14.90878378	6.467312349
0.4027	0.2931	0.0175	0.0263		23.01142857	11.14448669
<b>AVERAGE</b>						
0.413991667	0.291720833	0.021016667	0.027345833		21.66159572	12.64452279

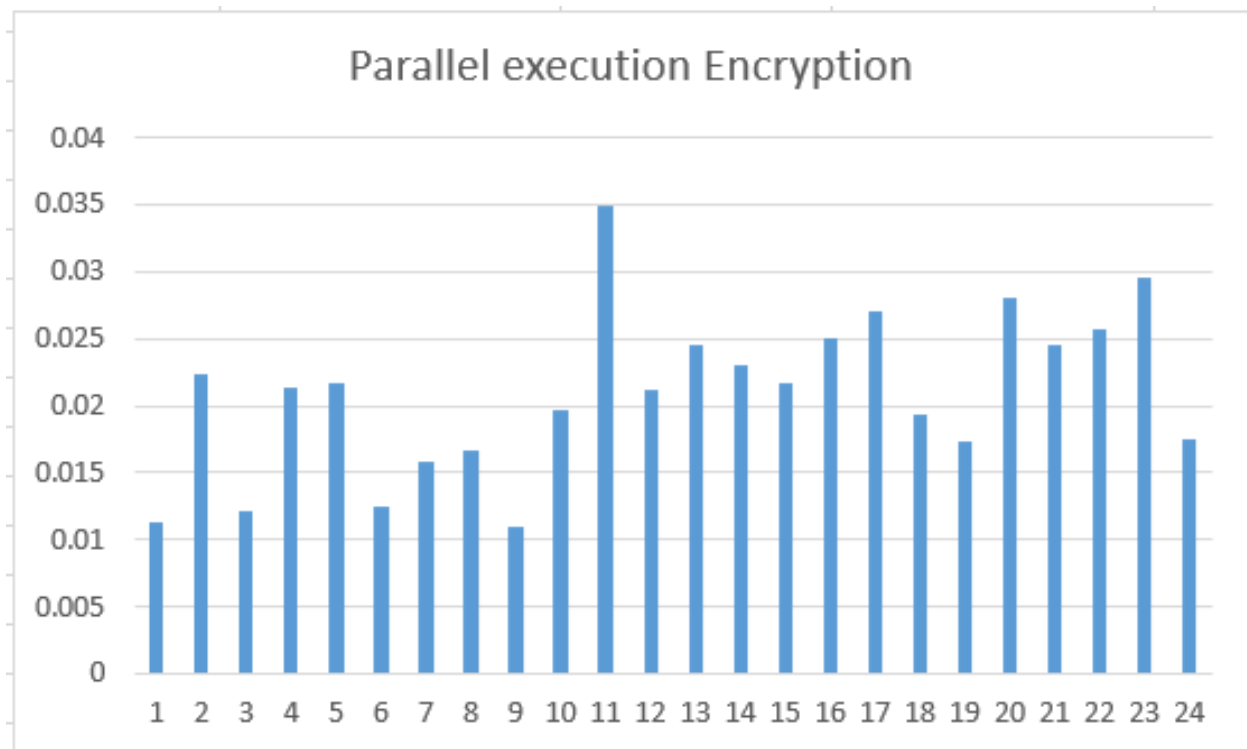
This table contains the time required for encryption and decryption of the image for both sequential algorithm and parallel algorithm. Finally, the average value for each type of execution is calculated.



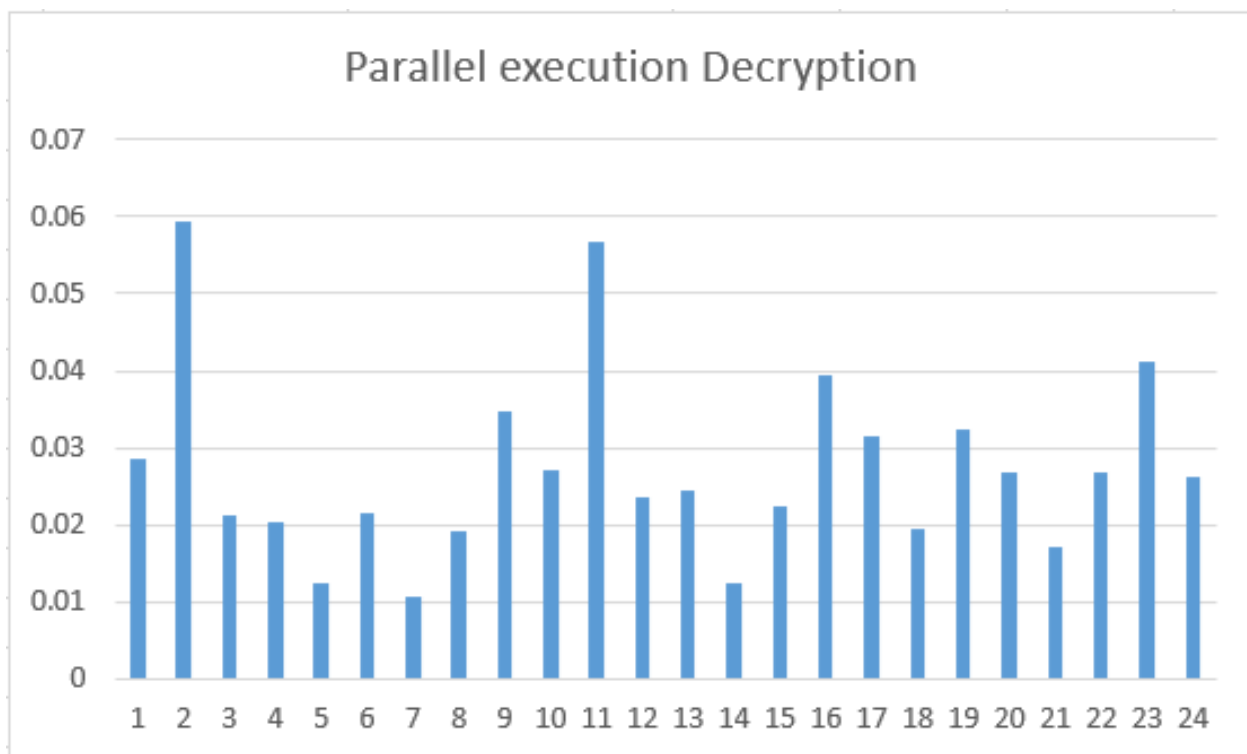
The average value for serial encryption is found out to be 0.413 seconds and this is almost constant and the variation in execution time occurs due to various factors.



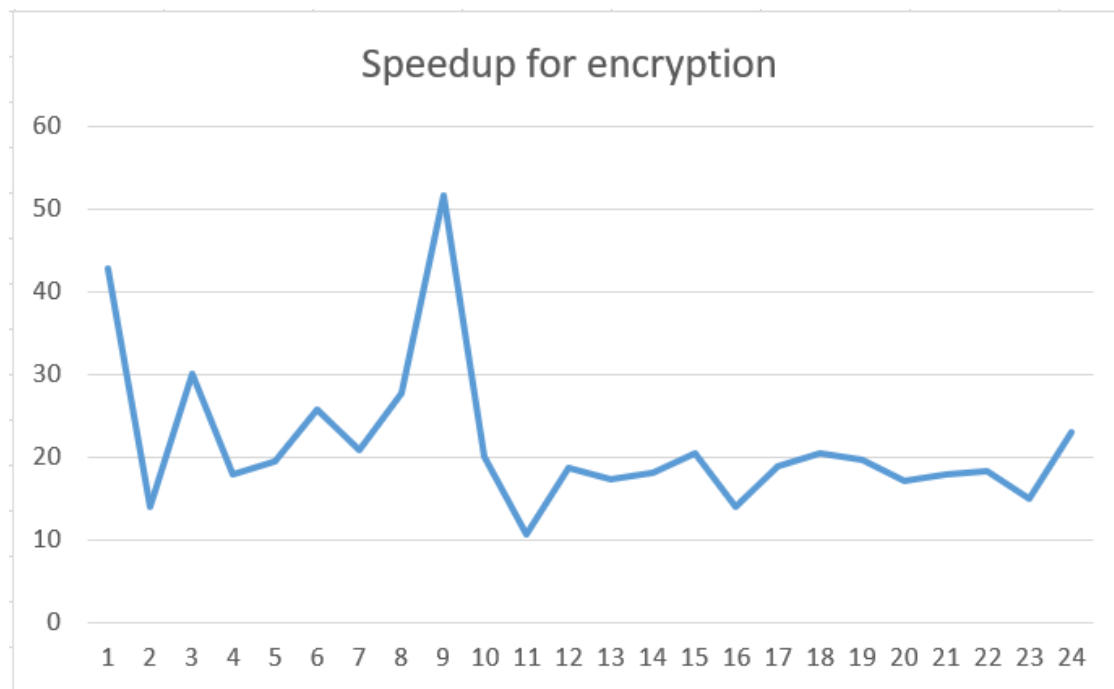
The average value for serial decryption is found out to be 0.291 seconds.



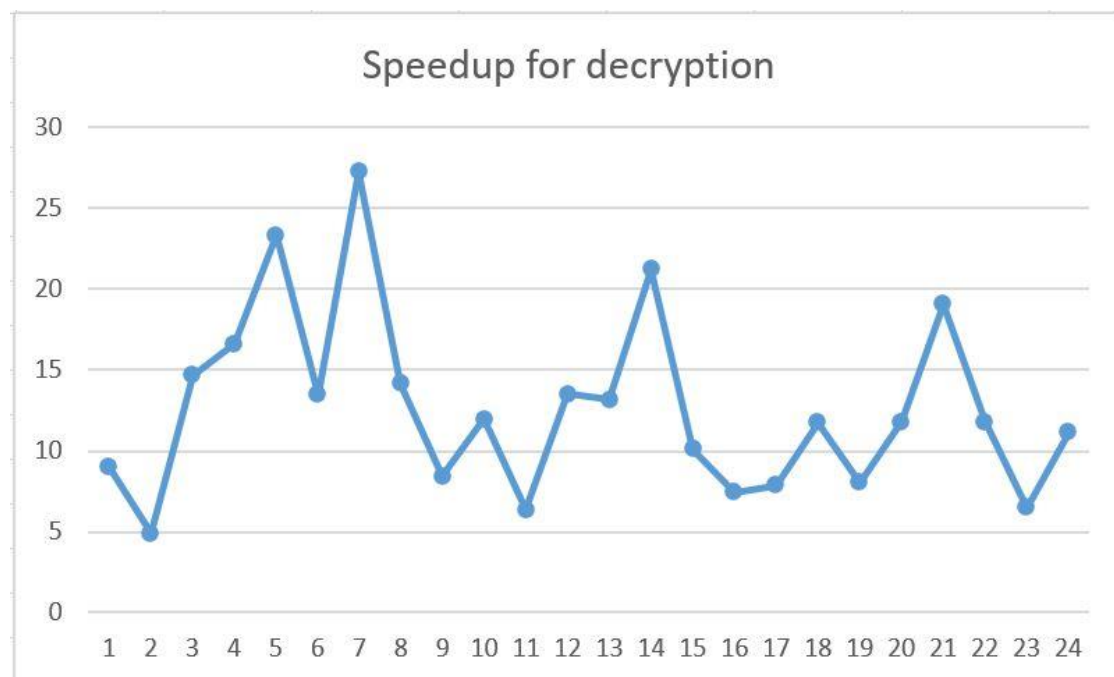
The average execution time is found out to be 0.021 seconds



The average execution time is found out to be 0.0273 seconds



There are a few outliers in the graph, but these outliers can be ignored because it's out of range. The average value for speedup for encryption is found out to be 21.66. This much speed up is achieved by parallelizing the code.



The average speedup for decryption is found out to be 12.64. This much speed up is achieved by parallelizing the code.

## 7. Conclusion

This project proposes and implements a way to fully implement the task and parallelize it. This project uses image encryption as a test and benchmark algorithm. Images are uploaded by users for encryption or decryption. The encryption and decryption process itself is performed using the DES (Data Encryption Standard) algorithm implemented in Python. Encryption and decryption of color images is done using the DES-algorithm by providing the required security for the images between two authorized users or clients. In our project, DES guarantees unbreakable security for color images. In Project, image encryption is performed using the DES algorithm. The experimental results of the proposed DES algorithm are very motivating. The implementation approach shows encrypted and decrypted images, and historical analysis is also performed using improved techniques. Future work in our project is to implement a DES technique that iterates the DES key three times to generate three sets of keys for the TDES algorithm, in order to generate a more secure T-DES algorithm. is based on improving the T-DES security level. . Another plan for the future is to use this proposed DES method to encrypt video files and provide secure transmission over communication channels.

## 8. Reference

1. Blelloch, G.E. *et al.* (no date) *An experimental analysis of parallel sorting algorithms - theory of Computing Systems*, SpringerLink. Springer-Verlag. Available at:  
<https://link.springer.com/article/10.1007/s002240000083> (Accessed:

November 16, 2022).

2. *An overview of encryption algorithms in color images* (no date).

Available at:

<https://www.sciencedirect.com/science/article/pii/S016516841930218>

X (Accessed: November 16, 2022).

3. *Color image encryption and decryption using DES algorithm* (no date). Available at: [https://www.irjet.net/archives/V3/i7/IRJET-](https://www.irjet.net/archives/V3/i7/IRJET-V3I7322.pdf)

V3I7322.pdf (Accessed: November 16, 2022).

4. Archana Swaminathan (2020) *Image encryption and decryption using Artificial Neural Networks*, Archana Swaminathan. Archana

Swaminathan. Available at:

<https://archana1998.github.io/project/encryption-decryption/>

(Accessed: November 16, 2022).

5. *Image encryption techniques: A critical comparison - researchgate* (no date). Available at:

[https://www.researchgate.net/publication/235950301\\_IMAGE\\_ENCRYPTION\\_TECHNIQUES\\_A\\_CRITICAL\\_COMPARISON](https://www.researchgate.net/publication/235950301_IMAGE_ENCRYPTION_TECHNIQUES_A_CRITICAL_COMPARISON)

(Accessed: November 16, 2022).

## 9. Appendix:

### Program:

### Sequential Encryption:

```
# try block to handle exception
import time
try:
    # take path of image as a input
    path = input(r'Enter path of Image : ')

    # taking encryption key as input
    key = int(input('Enter Key for encryption of Image : '))

    # print path of image file and encryption key that
    # we are using
    print('The path of file : ', path)
    print('Key for encryption : ', key)
    # get the start time
    st = time.time()
    # open file for reading purpose
    fin = open(path, 'rb')

    # storing image data in variable "image"
    image = fin.read()
    fin.close()

    # converting image into byte array to
    # perform encryption easily on numeric data
    image = bytearray(image)

    # performing XOR operation on each value of bytearray
    for index, values in enumerate(image):
        image[index] = values ^ key

    # opening file for writing purpose
    fin = open(path, 'wb')

    # writing encrypted data in image
    fin.write(image)
    fin.close()
    print('Encryption Done...')
    # get the end time
    et = time.time()
    # get the execution time
    elapsed_time = et - st
    print('Execution time:', elapsed_time, 'seconds')

except Exception:
    print('Error caught : ', Exception.__name__)
```

## Sequential Decryption:

```
import time

try:
    # take path of image as a input
    path = input('Enter path of Image : ')

    # taking decryption key as input
    key = int(input('Enter Key for encryption of Image : '))

    # print path of image file and decryption key that we are using
    print('The path of file : ', path)
    print('Note : Encryption key and Decryption key must be same.')
    print('Key for Decryption : ', key)
    # get the start time
    st = time.time()
    # open file for reading purpose
    fin = open(path, 'rb')

    # storing image data in variable "image"
    image = fin.read()
    fin.close()

    # converting image into byte array to perform decryption easily on
    numeric data
    image = bytearray(image)

    # performing XOR operation on each value of bytearray
    for index, values in enumerate(image):
        image[index] = values ^ key

    # opening file for writing purpose
    fin = open(path, 'wb')

    # writing decryption data in image
    fin.write(image)
    fin.close()
    print('Decryption Done...')
    # get the end time
    et = time.time()
    # get the execution time
    elapsed_time = et - st
    print('Execution time:', elapsed_time, 'seconds')

except Exception:
    print('Error caught : ', Exception.__name__)
```



## Parellel encryption and decryption:

```
import multiprocessing

# Then import DES3 for Encryption and md5 for key
from Crypto.Cipher import DES3
from hashlib import md5
from multiprocessing import Pool
import time

def image_encrypt_decrypt():

    # Encode given key to 16 byte ascii key with md5 operation
    key_hash = md5(key.encode('ascii')).digest()

    # Adjust key parity of generated Hash Key for Final Triple DES Key
    tdes_key = DES3.adjust_key_parity(key_hash)

    # Cipher with integration of Triple DES key, MODE_EAX for
    Confidentiality & Authentication
    # and nonce for generating random / pseudo random number which is used
    for authentication protocol
    cipher = DES3.new(tdes_key, DES3.MODE_EAX, nonce=b'0')

    # Open & read file from given path
    with open(file_path, 'rb') as input_file:
        file_bytes = input_file.read()

    if operation == '1':
        # Perform Encryption operation
        new_file_bytes = cipher.encrypt(file_bytes)
        print('Encryption done!')
    else:
        # Perform Decryption operation
        new_file_bytes = cipher.decrypt(file_bytes)
        print('Decryption done!')

    # Write updated values in file from given path
    with open(file_path, 'wb') as output_file:
        output_file.write(new_file_bytes)
        print('Operation complete!')

print('Choose operation to be done:\n\t1- Encryption\n\t2- Decryption')
operation = input('Your Choice: ')

# Image / File Path for operation
file_path = input('File path: ')

# Key for performing Triple DES algorithm
key = input('TDES key: ')

# get the start time
st = time.time()

image_encrypt_decrypt()
```

```
# get the end time
et = time.time()
# get the execution time
elapsed_time = et - st
print('Execution time:', elapsed_time, 'seconds')
```

```
# Code Complete!
```