Volume: 10 Issue: VI Month of publication: June 2022

DOI: https://doi.org/10.22214/ijraset.2022.44575

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VI June 2022- Available at www.ijraset.com

Node to Node Communication Security in IOT Networks

Jenisha. R[1], Mariya Desona. S[2], Priyanga. V[3]
Electronics and Communication Engineering, Jeppiaar Engineering College, ANNA University: Chennai

Abstract: The Internet of Things (IoT) describes the network of physical objects or things that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. Although several IoT devices are openly accessible to all in the network, it is extremely vital to be aware of the security risks and threats of cyber- attacks; therefore, it should be secured. In Cryptography, plain text is converted to encrypted text before it is sent, and it is converted to plain text after communication on the other side. Steganography is a method of hiding secret data, by embedding it into an audio, video, image, or text file. One technique is to hide data in bits that represent the same color pixels repeated in a row in an image file. In this project it proposes to encrypt the IoT networks data by Cryptography method and hide the encrypted message inside an image file using Steganography method as well increases the number of bits that can be saved within a pixel of an image. To incorporate the usage of convolutional neural networks in traditional steganography method to drastically increase the payload that can be transmitted through an image. Thus, in this project the convolutional networks algorithm will be developed and trained in such a way to increase the payload of the data to be encrypted as well as safely decrypted to view the original message.

I. INTRODUCTION

A. General Introduction

The Internet of Things (IoT's) is the concept of connecting any device (so long as it has an on/off switch) to the Internet and to other connected devices. The IoT is a giant network of connected things and people – all of which collect and share data about the way they are used and about the environment around them. IoT devices contain sensors and mini-computer processors that act on the data collected by the sensors via machine learning. IoT devices share the sensor data they collect by connecting to an IoT gateway or other edge device where data is either sent to the cloud to be analyzed or analyzed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices for instance, to set them up, give them instructions or access the data.

The internet of things helps people live and work smarter, as well as gain complete control over their lives. In addition to offering smart devices to automate homes, IoT is essential to business. IoT provides businesses with a real- time look into how their systems really work, delivering insights into everything from the performance of machines to supply chain and logistics operations. IoT enables companies to automate processes and reduce labour costs. It also cuts down on waste and improves service delivery, making it less expensive to manufacture and deliver goods, as well as offering transparency into customer transactions. As such, IoT is one of the most important technologies of everyday life, and it will continue to pick up steam as more businesses realize the potential of connected devices to keep them competitive.

1) Advantages and Disadvantages

Some of the advantages of IoT include the following,

a) Ability to access information from anywhere at any time on any device;
b) Improved communication between connected electronic devices;
c) Transferring data packets over a connected network saving time and money;
d) Automating tasks helping to improve the quality of a business's services and reducing the need for human intervention.

Some disadvantages of IoT include the following:

- As the number of connected devices increases and more information is shared between devices, the potential that a hacker could steal confidential information also increases.
- Enterprises may eventually have to deal with massive numbers -- maybe even millions -- of IoT devices, and collecting and managing the data from all those devices will be challenging.

- If there's a bug in the system, it's likely that every connected device will become corrupted.
- Since there's no international standard of compatibility for IoT, it's difficult for devices from different manufacturers to communicate with each other.

B. Technologies Used

1) Cryptography

Cryptography or cryptology derived fromGreek word "kryptos", which means "hidden" and "graphein", means "writing", and is the practice and study of techniques for secure communication in the presence of adversarial behaviour.
Cryptography refers to secure information and communication techniques derived from mathematical concepts and a set of rule-based calculations called algorithms, to transform messages in ways that are hard to decipher. A cryptographic algorithm, or cipher, is a mathematical function used in the encryption and decryption process. A cryptographic algorithm works in combination with a key a word, number, or phrase to encrypt the plaintext. The same plaintext encrypts to different cipher text with different keys. The security of encrypted data is entirely dependent on two things: the strength of the cryptographic algorithm and the secrecy of the key. These deterministic algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on the internet and confidential communications such as credit card transactions and email.The first known use of a modern cipher was by Julius Caesar (100 B.C. to 44 B.C.), who did not trust his messengers when communicating with his governors and officers. For this reason, he created a system in which each character in his messages was replaced by a character three positions ahead of it in the Roman alphabet.

Cryptographic techniques concern themselves with three basic purposes:

- Authentication Verifying the identity of a user or computer.
- Confidentiality keeping the contents of the data secret.

Integrity Ensuring that data doesn't change between the time it leaves the source and the time it reaches its destination.

2) Steganography

Steganography has been derived from two Greek words "Stego" which means "Covered" and "Graphia" which means "writing", thus translating to 'covered writing', or 'hidden writing'. Steganography is a method of hiding secret data, by embedding it into an audio, video, image, or text file. It is one of the methods employed to protect secret or sensitive data from malicious attacks. In modern digital steganography, data is first encrypted or obfuscated in some other way and then inserted, using a special algorithm, into data that is part of a particular file format such as a JPEG image. The secret message can be embedded into ordinary data files in many different ways. While cryptography and steganography are related, there is a difference between the two. Cryptography is used to scramble messages so that they cannot be understood. It does not hide the fact that the message exists. Steganography, on the other hand, conceals the fact that the message exists by hiding the actual message in another.In the 20th century, invisible inks were widely used. Fluids like vinegar, milk and fruit juices were used to write messages that could not be seen unless subjected to special treatment. When heated, these fluids turn dark and the message can be read. The current form of this technique is also quite similar, the idea being to prevent the detection of a message by hiding it rather than distorting it.
Steganography's ultimate objectives, which are

- Undetectability,
- Capacity of the hidden data is the main factors and Robustness There are various methods of steganography:
- Least significant bit (LSB) method
- Transform domain techniques
- Statistical methods
- Distortion techniques
- Hash-LSB and RSA algorithm

3) Deep Learning Process

A deep neural network provides state-of-the-art accuracy in many tasks, from object detection to speech recognition. They can learn automatically, without predefined knowledge explicitly coded by the programmers.

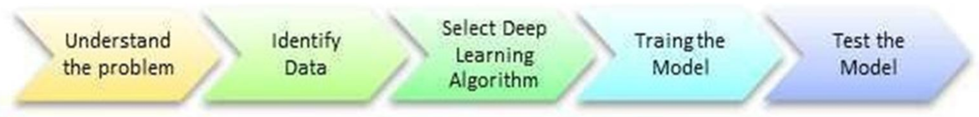


Figure 1.1 Deep Learning Process

To grasp the idea of deep learning Process fig 1.1 as shown, imagine a family, with an infant and parents. The toddler points objects with his little finger and always says the word 'cat.' As its parents are concerned about his education, they keep telling him 'Yes, that is a cat' or 'No, that is not a cat. The infant persists in pointing objects but becomes more accurate with cats. The little kid, deep down, does not know why he can say it is a cat or not. He has just learned how to hierarchies complex features coming up with a cat by looking at the pet overall and continue to focus on details such as the tails or the nose before to make up his mind. A neural network works quite the same. Each layer represents a deeper level of knowledge, i.e., the hierarchy of knowledge. A neural network with four layers will learn more complex feature than with that with two layers.

The learning occurs in two phases

- The first phase consists of applying a nonlinear transformation of the input and creates a statistical model as output.
- The second phase aims at improving the model with a mathematical method known as derivative.

The neural network repeats these two phases hundreds to thousands of time until it has reached a tolerable level of accuracy. The repeat of this two-phase is called an iteration.

C. *Objectives of the Project*

- To effectively develop a technological solution to transfer data securely.
- To implement steganography to transfer data through image files.
- To effectively increase the payload of data while transferring through image files.

D. *Scope Of The Project*

- Implementing IoT devices to securely transmit IoT network data
- Securing the Communication in IoT's using innovative cryptography and steganography.

II. LITERATURE SURVEY

A. *Literature Survey*

1) *Steganalysis of Digital Images Using Deep Fractal Network*

In the recent literature on steganalysis, it has been observed that a deeper network is, in general, preferred for detecting low tone embedding noise, e.g., SRNet. However, very recently, a deep model, called FractalNet, became popular, which is based on selfsimilarity and grows deeper and wider by maintaining a balance between depth and width using a recurrent adaptation of a fundamental building block. In this work, the concept of the FractalNet model has been exploited for steganalytic detection, where the embedded image has been used as input. In a practical scenario, it has been observed that steganalytic detection for test images is increasing if the width of the network can be increased with a certain proportion to the depth. The proposed deep network is designed by repeating a basic fractal block in such a way that a balance between the depth and width of the overall network can be maintained. A comprehensive set of experiments reveals that the proposed model outperformed the state-of-the-art results. An ablation study is also included to justify the proposed architecture in favor of its performance. The heavy data augmentation, FractalNet still cannot have the best result when comparing with those with other algorithms.

2) *Securing Data in Internet of Things (IoT) Using Cryptography and Steganography*

Internet of Things (IoT) is a domain wherein which the transfer of data is taking place every single second. The security of these data is a challenging task; however, security challenges can be mitigated with cryptography and steganography techniques. These techniques are crucial when dealing with user authentication and data privacy. In the proposed work, the elliptic Galois cryptography protocol is introduced and discussed. In this protocol, a cryptography technique is used to encrypt confidential data that came from different medical sources. Next, a Matrix XOR encoding steganography technique is used to embed the encrypted data into a low complexity image.

The proposed work also uses an optimization algorithm called Adaptive Firefly to optimize the selection of cover blocks within the image. Based on the results, various parameters are evaluated and compared with the existing techniques. Finally, the data that is hidden in the image is recovered and is then decrypted. It increases the size of the encrypted message significantly more than RSA encryption. Furthermore, the EGC algorithm is more complex and more difficult to implement.

3) Adaptive Payload Distribution in Multiple Images Steganography Based on Image Texture Features

With the coming era of cloud technology, cloud storage is an emerging technology to store massive digital images, which provides steganography a new fashion to embed secret information into massive images. Specifically, a resourceful steganographer could embed a set of secret information into multiple images adaptively, and share these images in cloud storage with the receiver, instead of traditional single image steganography. Nevertheless, it is still an open issue how to allocate embedding payload among a sequence of images for security performance enhancement. This paper formulates adaptive payload distribution in multiple images steganography based on image texture features and provides the theoretical security analysis from the steganalyst's point of view. Two payload distribution strategies based on image texture complexity and distortion distribution are designed and discussed respectively. The proposed strategies can be employed together with these state-of-the-art single image steganographic algorithms. The comparisons of the security performance against the modern universal pooled steganalysis are given. Furthermore, this paper compares the per image detectability of these multiple images steganographic schemes against the modern single image steganalyzer. Compared with Embedding Strategy Based on Distortion Distribution (ES-DD) and IMS, the security performance of ES-ITC is slightly low, especially for higher payload rates.

4) Deep Residual Network for Steganalysis of Digital Images

Steganographydetectors built as deep convolutional neural networks have firmly established themselves as superior to the previous detection paradigm – classifiers based on rich media models. Existing network architectures, however, still contain elements designed by hand, such as fixed or constrained convolutional kernels, heuristic initialization of kernels, the threshold linear unit that mimics truncation in rich models, quantization of feature maps, and awareness of JPEG phase. In this paper, we describe a deep residual architecture designed to minimize the use of heuristics and externally enforced elements that is universal in the sense that it provides state-of-theart detection accuracy for both spatial-domain and JPEG steganography. The key part of the proposed architecture is a significantly expanded front part of the detector that "computes noise residuals" in which pooling has been disabled to prevent suppression of the stego signal. Extensive experiments show the superior performance of this network with a significant improvement especially in the JPEG domain. Further performance boost is observed by supplying the selection channel as a second channel.Low accuracy for complex data and the process is complicated and also time consuming.

B. Conclusion

From the above reference papers, we can conclude that the following are the major disadvantages:

- In the existing system, with heavy data augmentation, FractalNet still cannot have the best result when comparing with those with other algorithms.
- Fractal networks are resistant to being too deep; extra depth may slow training, but does not impair accuracy.
- One of the biggest challenges in training any steganalytic model is the time that it takes to converge for the images with a low payload, such as 0.1 or 0.05 bpp. Sometimes, the model failed to converge at all.

III. PROPOSED SYSTEM

A. Existing System

In the recent literature on steganalysis, it has been observed that a deeper network is, in general, preferred for detecting low tone embedding noise, e.g., SRNet. However, very recently, a deep model, called FractalNet, became popular, which is based on selfsimilarity and grows deeper and wider by maintaining a balance between depth and width using a recurrent adaptation of a fundamental building block. In this work, the concept of the FractalNet model has been exploited for steganalytic detection, where the embedded image has been used as input. In a practical scenario, it has been observed that steganalytic detection for test images is increasing if the width of the network can be increased with a certain proportion to the depth. The proposed deep network is designed by repeating a basic fractal block in such a way that a balance between the depth and width of the overall network can be maintained. A comprehensive set of experiments reveals that the proposed model outperformed the state-of-the-art results. An ablation study is also included to justify the proposed architecture in favor of its performance.

1) Disadvantages Of Existing System

- One of the biggest challenges in training any steganalytic model is the time that it takes to converge for the images with a low payload, such as 0.1 or
  0.05 bpp. Sometimes, the model failed to converge at all.
- Fractal networks are resistant to being too deep; extra depth may slow training, but does not impair accuracy.
- With heavy data augmentation, FractalNet still cannot have the best result when comparing with those with other algorithms.

B. Proposed System

The Internet of Things (IoT) describes the network of physical objects or things that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. Although several IoT devices are openly accessible to all in the network, it is extremely vital to be aware of the security risks and threats of cyber-attacks; therefore, it should be secured. In Cryptography, plain text is converted to encrypted text before it is sent, and it is converted to plain text after communication on the other side. Steganography is a method of hiding secret data, by embedding it into an audio, video, image, or text file. One technique is to hide data in bits that represent the same color pixels repeated in a row in an image file. By applying the encrypted data to this redundant data in some inconspicuous way, the result will be an image file that appears identical to the original image but that has "noise" patterns of regular, unencrypted data. In this project we are going to encrypt the IoT networks data by Cryptography method and hide the encrypted message inside an image file by Steganography method as well increases the number of bits that can be saved within a pixel of an image. Thus, in this project the convolutional networks algorithm will be developed and trained in such a way to increase the payload of the data to be encrypted as well as safely decrypted to view the original message.

1) Advantages of Proposed System

- Provides security to IoT network data.
- Incorporates deep learning into traditional steganography methods.
- Increases the payload that can be transmitted via an image.
- Securely transmits the data.
- Real time encoding and decoding of data will be performed.

C. Architecture Diagram

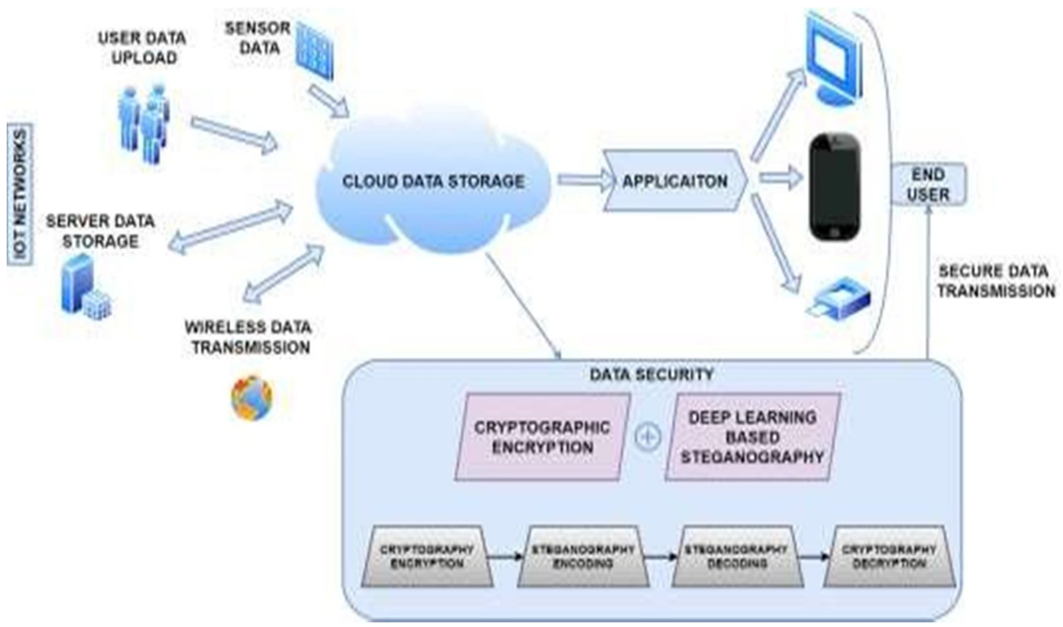


Figure 3.1 IoT Networks

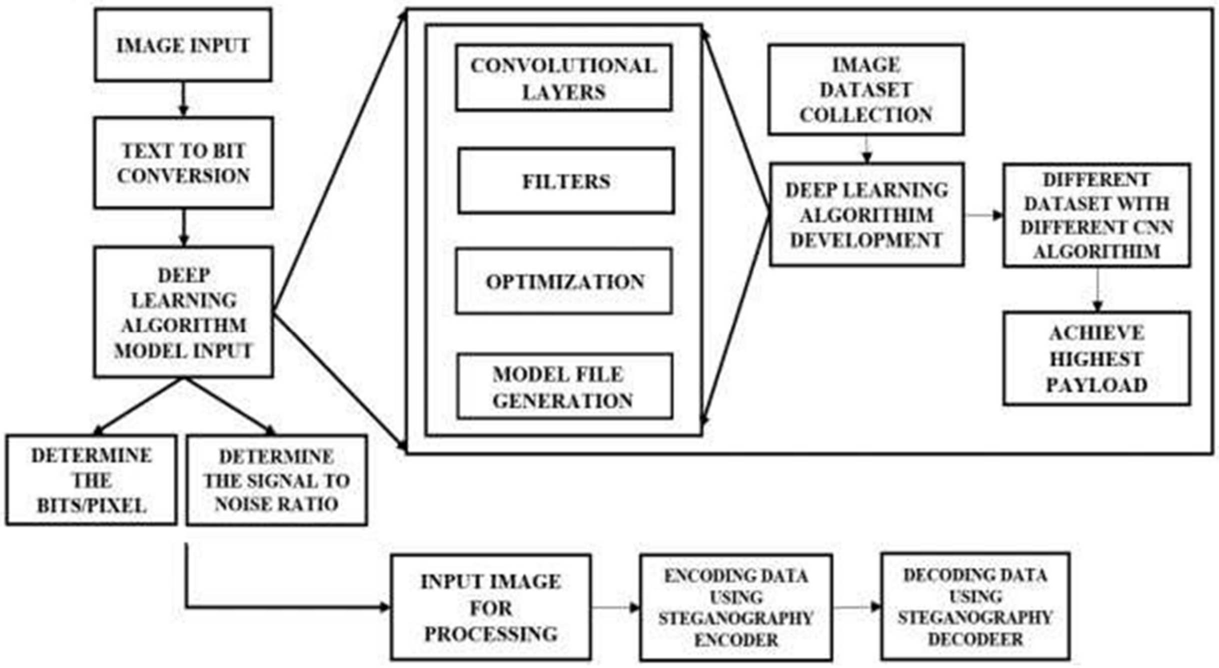


IMAGE INPUT
TEXT TO BIT CONVERSION
DEEP LEARNING ALGORITHM MODEL INPUT
DETERMINE THE BITS/PIXEL
DETERMINE THE SIGNAL TO NOISE RATIO
CONVOLUTIONAL LAYERS
FILTERS
OPTIMIZATION
MODEL FILE GENERATION
IMAGE DATASET COLLECTION
DEEP LEARNING ALGORITHIM DEVELOPMENT
DIFFERENT DATASET WITH DIFFERENT CNN ALGORITHIM
ACHIEVE HIGHEST PAYLOAD
INPUT IMAGE FOR PROCESSING
ENCODING DATA USING STEGANOGRAPHY ENCODER
DECODING DATA USING STEGANOGRAPHY DECODEER

Figure 3.2 Proposed System Architecture

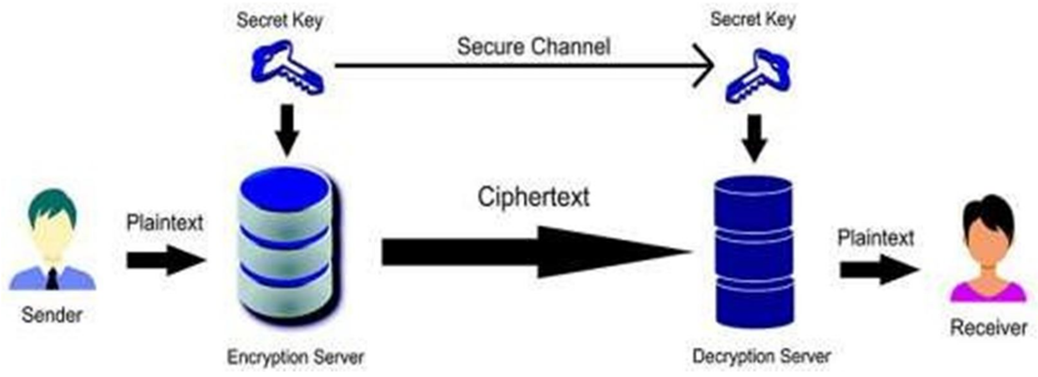


Figure 3.3 Steganography encoding and decoding

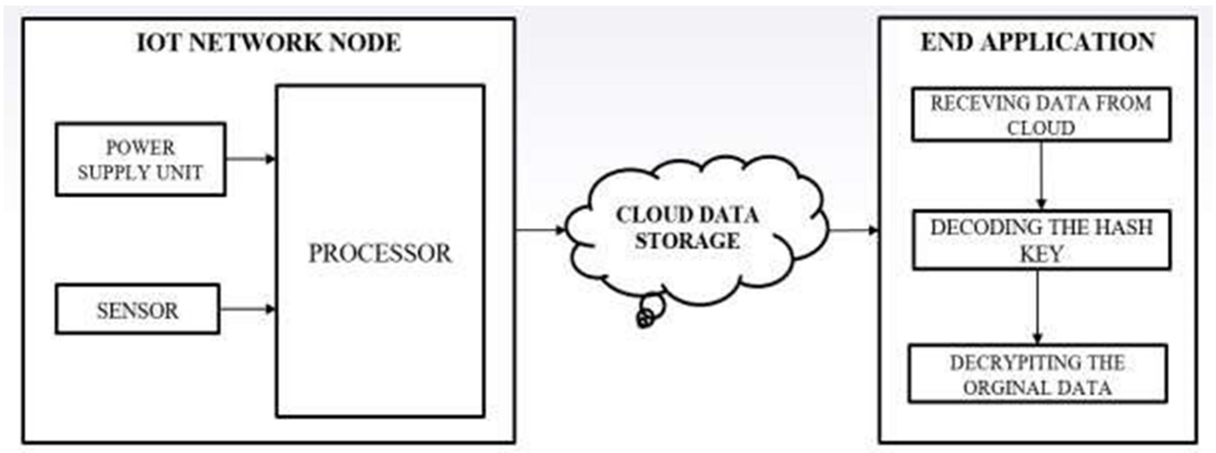


Figure 3.4 Hardware Block Diagram

D. Working

In this project, it proposes system to secure army data stored in network inside an image file using Steganography method as well increases the number of bits that can be saved within a pixel of an image.The usage of convolutional neural networks in traditional steganography method to drastically increase the payload that can be transmitted through an image. So, the first step in theproject will be collecting the DIV2K dataset and then will be separating these datasets into training as well as testing dataset where the testing dataset will be kept separate and the training dataset will be used to train the model. For training the dataset, encoding, decoding and critic modules are used. For training the algorithm 20 to 100 epochs are used. The Encoder module takes an cover image and a data tensor and combines them into a steganographic image. The Critic module takes an image and predicts whether it is a cover image or a steganographic image (N, 1). The Decoder module takes a steganographic image and attempts to decode the embedded data tensor. Then calculating the metric values such as payload, bits per pixel and accuracy, will be used to predict the accuracy of the model. After that, algorithm is enhanced to increase the payload or storage capacity of the message inside the image. The normal storage capacity is 0.5 bits per pixel. By modifying the existing algorithm in a such way to increase the bpp to more than 2 to 3 bpp, it will be trained and compared to the existing system using the metrics. After performing cryptography, using AES algorithm the data has been encrypted and the hash key has been generated. This hash key has been encoded inside an image using steganography and encoding the message in the image, the original image and the encoded image will looks the same. Whenever the data needs to be decrypted the hash is been decoded from the image using steganography decryption and cryptography decryption is applied to get the actual data. Also it will be safely decrypted using AES to view the original message.Finally a sensor will be connected to get the real time values and these data will be stored in the cloud for future purpose. Whenever the data needs to be decrypted from the cloud the hash will be decoded and applied to get the actual data.

E. System Analysis

1) Modules Description

- AES Cryptography Encryption
- AES Cryptography Decryption
- Encoding and Decoding of Image using Steganography

a) AES Cryptography Encryption

The features of AES are as follows

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details

AES Cryptoraphy Encryption fig 3.5 is an iterative rather than Feistel cipher. It is based on 'substitution– permutation network'. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations). Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix. Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

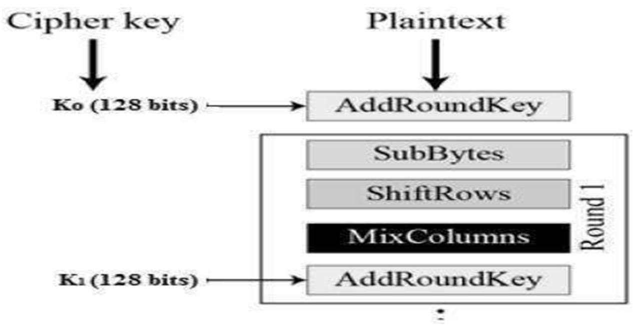


Figure 3.5 AES cryptography encryption

b) AES Cryptography Decryption

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order.

- Add round key Mix columns
- Shift rows
- Byte substitution
- Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be separately implemented, although they are very closely related.

c) Encoding and decoding of Image

Steganography is the process of embedding the data into the object without using any private keys. This type of steganography entirely depends upon the secrecy. This type of steganography uses a cover image in which data is to be embedded, personal information to be transmitted, and encryption decryption algorithms to embed the message into image. Many different carrier file formats can be used, but digital images are the most popular because of their frequency on the internet. For hiding secret information in images, there exist a large variety of steganography techniques some are more complex than others and all of them have respective strong and weak points. Different applications may require absolute invisibility of the secret information, while others require a large secret message to be hidden. Steganography in the transform domain involves the manipulation of algorithms and image transforms. These methods hide messages in more significant areas of the cover image, making it more robust. Many transform domain methods are independent of the image format and the embedded message may survive conversion between lossy and lossless compression.

- *Encoding:* The encoding function as shown in the fig 3.6 determines the message type and length and encodes this information as header information. Then the function sequentially encodes the message values across the RGB channels. This means that every message using this function is encoded from the top left pixel and is coded from top to bottom, left to right. It employed the as the encoding function along with a secured encryption key.

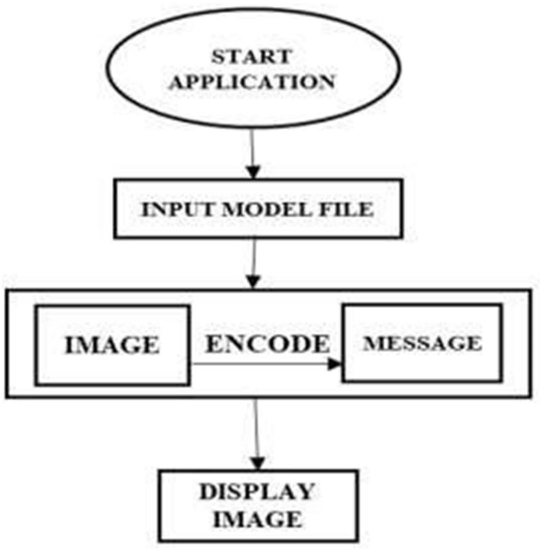


Figure 3.6 Encoding Process

- *Decoding:* The decoding process as shown in the fig 3.7 of the proposed system uses the reverse procedure of encoding to extract the embedded data from stego- images. The encryption key used by the encoder to embed the secret data is shared with the decoder for the retrieval of hidden message. The decoding function recovers a sequentially encoded message from the stego-images. This function takes in the stego-image and encryption key, decode the header to determine the message type and message length, and sequentially decodes and recovers the message from the pixel values of the input image.

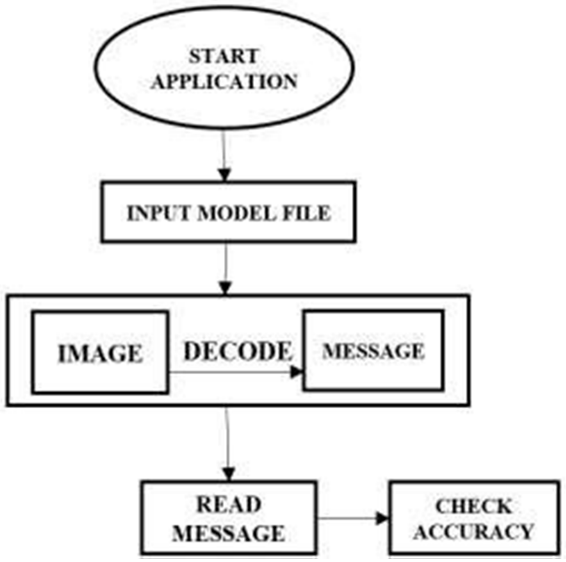


Figure 3.7 Decoding Process

F. Hardware Description

- Raspberry Pi
- Power Supply
- DHT 11 Sensor

1) *Raspberry Pi:* The Raspberry Pi hardware has evolved through several versions that feature variations in memory capacity and peripheral-device support.

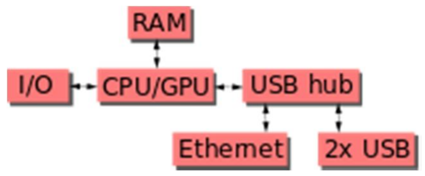


Figure 3.8 Block Diagram

This block diagram fig 3.8 describes Model B and B+; Model A, A+, and the Pi Zero are similar, but lack the Ethernet and USB hub components. The Ethernet adapter is internally connected to an additional USB port. In Model A, A+, and the Pi Zero, the USB port is connected directly to the system on a chip (SoC). On the Pi 1 Model B+ and later models the USB/Ethernet chip contains a five-port USB hub, of which four ports are available, while the Pi 1 Model B only provides two. On the Pi Zero, the USB port is also connected directly to the SoC, but it uses a micro USB (OTG) port.

Figure 3.9 Raspberry Pi

The Broadcom BCM2835 SoC used in the first generation as shown in the fig 3.9 Raspberry Pi includes a 700 MHz ARM11 76JZF-S processor, VideoCore IV graphics processing unit (GPU), and RAM. It has a level 1 (L1) cache of 16 KB and a level 2 (L2) cache of 128 KB. The level 2 cache is used primarily by the GPU. The SoC is stacked underneath the RAM chip, so only its edge is visible. The 1176JZ(F)-S is the same CPU used in the original iPhone, although at a higher clock rate, and mated with a much faster GPU. The earlier V1.1 model of the Raspberry Pi 2 used a Broadcom BCM2836 SoC with a 900 MHz 32-bit, quad-core ARM Cortex-A7 processor, with 256 KB shared L2 cache. The Raspberry Pi 2 V1.2 was upgraded to a Broadcom BCM2837 SoC with a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor, The Raspberry Pi 3+ uses a Broadcom BCM2837B0 SoC with a 1.4 GHz 64-bit quad-core ARM CortexA53 processor, with 512 KB shared L2 cache. Raspberry Pi provided a real-world performance roughly equivalent to 0.041 GFLOPS. On the CPU level the performance is similar to a 300 MHz Pentium II of 1997–99. Raspberry Pi 2 V1.1 included a quad core Cortex-A7 CPU running at 900 MHz and 1 GB RAM. To be highly dependent upon task threading and instruction set use Benchmarks showed the Raspberry Pi 3 to be approximately 80% faster than the Raspberry Pi 2 in parallelised tasks.

2) DHT11 - Humidity and Temperature Sensor

As shown in the fig 3.10.DHT11Sensor is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). The only real downside of this sensor is you can only get new data from it once every 2 seconds.

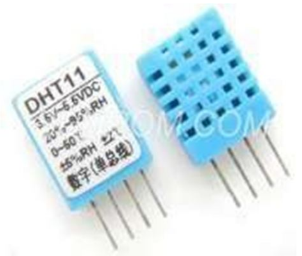


Figure 3.10 DHT 11 Sensor

Features

- Full range temperature compensated
- Relative humidity and temperature measurement
- Extra components not needed
- Low power consumption

G. Software Description

The purpose of the Software Requirement Specification is to produce the specification of the analysis task and also to establish complete information about the requirement, behaviour and also the other constraint like functional performance and so on. The main aim of the Software Requirement Specification is to completely specify the technical requirements for the software product in a concise and in unambiguous manner.

1) *Python:* Python supports the use of modules and packages, which mean that programs can be designed in a modular style and code, can be reused across a variety of projects. Once we've developed a module or package we need, it can be scaled for use in other projects, importantly, it is an interpreted language, which means that the written code is not actually translated to a computer- readable format at runtime. Whereas, most programming languages do this conversion before the program is even run. This type of language is also referred to as a "scripting language" because it was initially meant to be used for trivial projects. Python can be used to process text, display numbers or images, solve scientific equations, and save data. In short, it is used behind the scenes to process a lot of elements we might need or encounter on our device(s) - mobile included.
2) *Google Colab:* In this project, google colab is used as an open-source IDE.Google Colaboratory is a free online cloud-based Jupyter notebook environment that allows us to train our machine learning and deep learning models on Google Colab gives us three types of runtime for our notebooks:

- CPUs,
- GPUs, and
- TPUs

After that, the whole virtual machine is cleared and we have to start again. We can run multiple CPU, GPU, and TPU instances simultaneously, but their resources are shared between these instances.

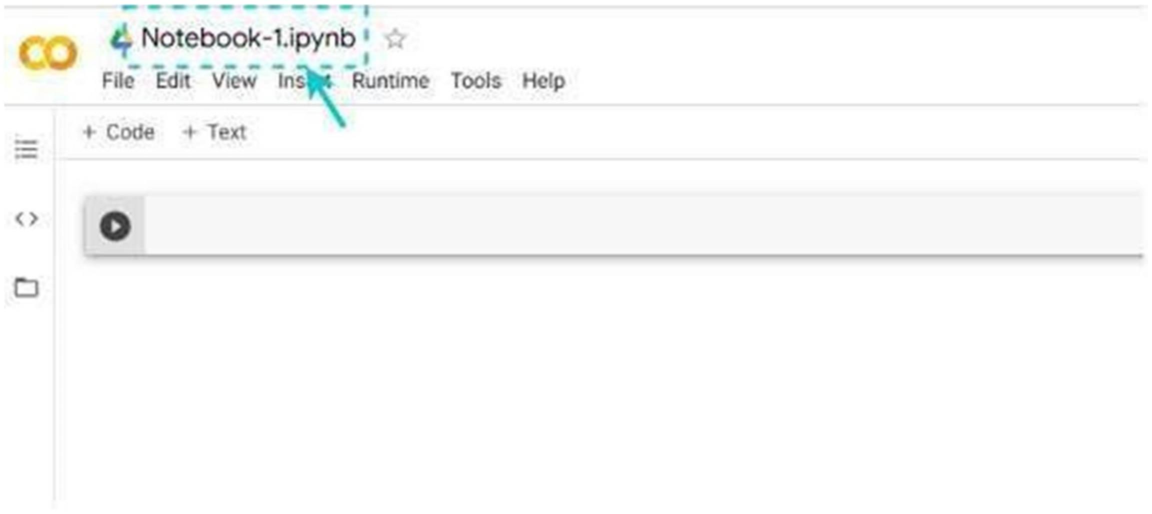


Figure 3.11 Google Colab Notebook

Colab notebooks allow us to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When we create our own Colab notebooks, they are stored in our Google Drive account. To learn more, see Overview of Colab. To create a new Colab notebook we can use the File menu above, or use the following link: create a new Colab notebook. Colab notebooks are Jupyter notebooks that are hosted by Colab. As a developer, we can perform the following using Google Colab;

- Write and execute code in Python
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets
- Integrate PyTorch, TensorFlow, Keras, OpenCV

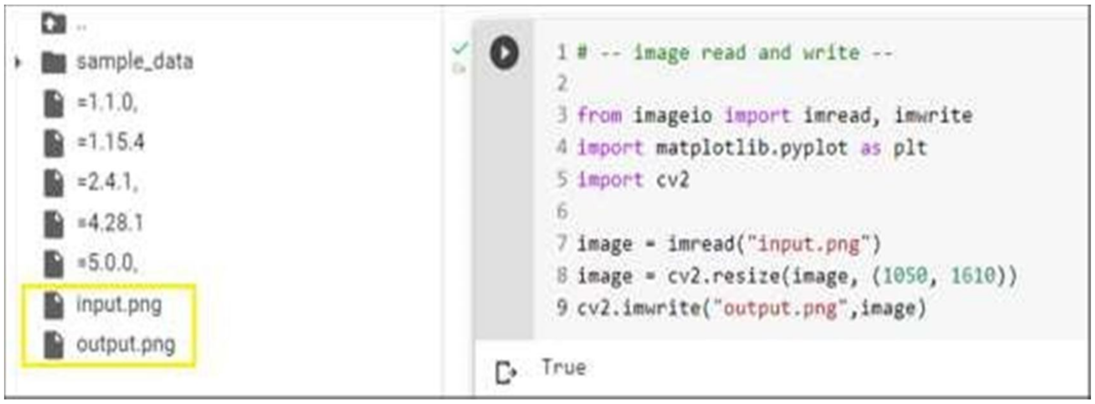
IV. RESULTS AND DISCUSSIONS

A. Introduction

This chapter discusses about the practical results obtained while implementing the project.

B. Results Obtained

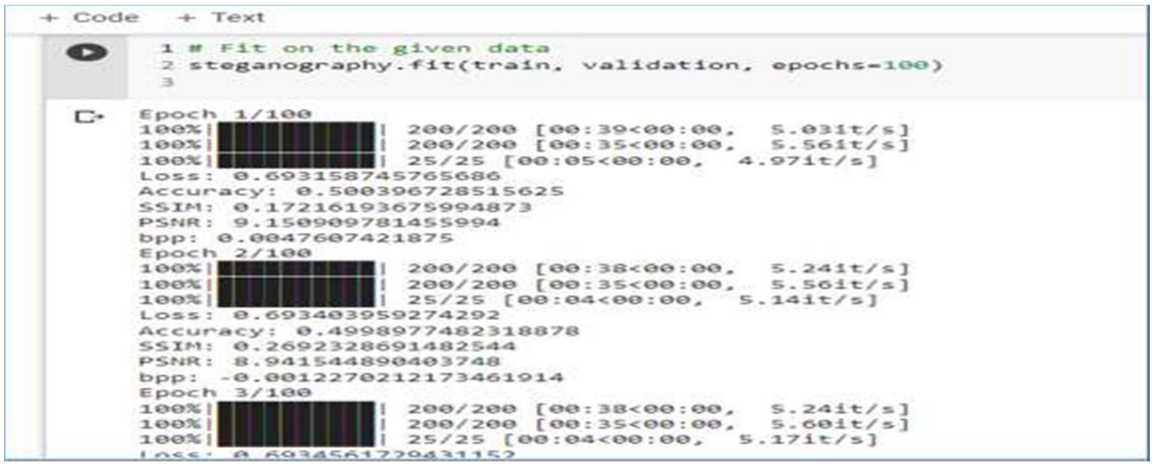
To begin with, testing of the trained model, we can split our project into modules of implementation that is done.



```
# -- image read and write --

from imageio import imread, imwrite
import matplotlib.pyplot as plt
import cv2

image = imread("input.png")
image = cv2.resize(image, (1050, 1610))
cv2.imwrite("output.png",image)
```

```
True
```

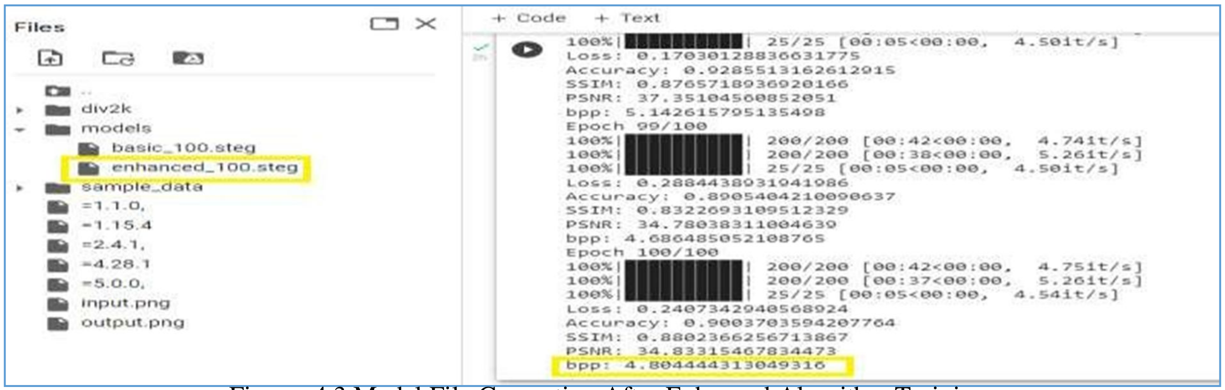
Figure 4.1 Image Read And Write



```
# Fit on the given data
steganography.fit(train, validation, epochs=100)

Epoch 1/100
100%|██████████| 200/200 [00:39<00:00,  5.03it/s]
100%|██████████| 200/200 [00:35<00:00,  5.56it/s]
100%|██████████| 25/25 [00:05<00:00,  4.97it/s]
Loss: 0.693158745765686
Accuracy: 0.500396728515625
SSIM: 0.17216193675994873
PSNR: 9.150909781455994
bpp: 0.0047607421875
Epoch 2/100
100%|██████████| 200/200 [00:38<00:00,  5.24it/s]
100%|██████████| 200/200 [00:35<00:00,  5.56it/s]
100%|██████████| 25/25 [00:04<00:00,  5.14it/s]
Loss: 0.693403959274292
Accuracy: 0.4998977482318878
SSIM: 0.2692328691482544
PSNR: 8.941544890403748
bpp: -0.0012270212173461914
Epoch 3/100
100%|██████████| 200/200 [00:38<00:00,  5.24it/s]
100%|██████████| 200/200 [00:35<00:00,  5.60it/s]
100%|██████████| 25/25 [00:04<00:00,  5.17it/s]
Loss: 0.6934561729431152
```

Figure 4.2 Deep Learning Algorithm Training

After training with basic deep learning algorithm gets completed, the model file has been generated via which the output for encoding and decoding can be checked.

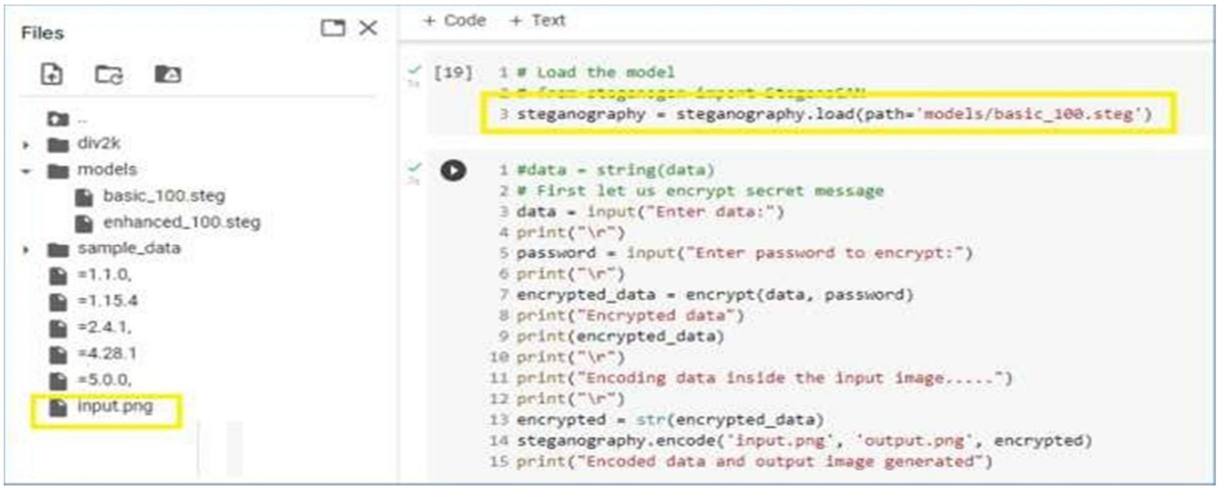


```
Files
..
div2k
models
    basic_100.steg
    enhanced_100.steg
sample_data
=1.1.0,
=1.15.4
=2.4.1,
=4.28.1
=5.0.0,
input.png
output.png
```

```
100%|██████████| 25/25 [00:05<00:00,  4.50it/s]
Loss: 0.17030128836631775
Accuracy: 0.9285513162612915
SSIM: 0.8765718936920166
PSNR: 37.35104560852051
bpp: 5.142615795135498
Epoch 99/100
100%|██████████| 200/200 [00:42<00:00,  4.74it/s]
100%|██████████| 200/200 [00:38<00:00,  5.26it/s]
100%|██████████| 25/25 [00:05<00:00,  4.50it/s]
Loss: 0.2884438931941986
Accuracy: 0.8905404210090637
SSIM: 0.8322693109512329
PSNR: 34.78038311004639
bpp: 4.686485052108765
Epoch 100/100
100%|██████████| 200/200 [00:42<00:00,  4.75it/s]
100%|██████████| 200/200 [00:37<00:00,  5.26it/s]
100%|██████████| 25/25 [00:05<00:00,  4.54it/s]
Loss: 0.2407342940568924
Accuracy: 0.9003703594207764
SSIM: 0.8802366256713867
PSNR: 34.83315467834473
bpp: 4.804444313049316
```

Figure: 4.3 Model File Generation After Enhanced Algorithm Training

After the training process and the metrics are also calculated. The bits per pixel which is the aim pf the project to achieve more than 3 bits per pixel is been successfully achieved



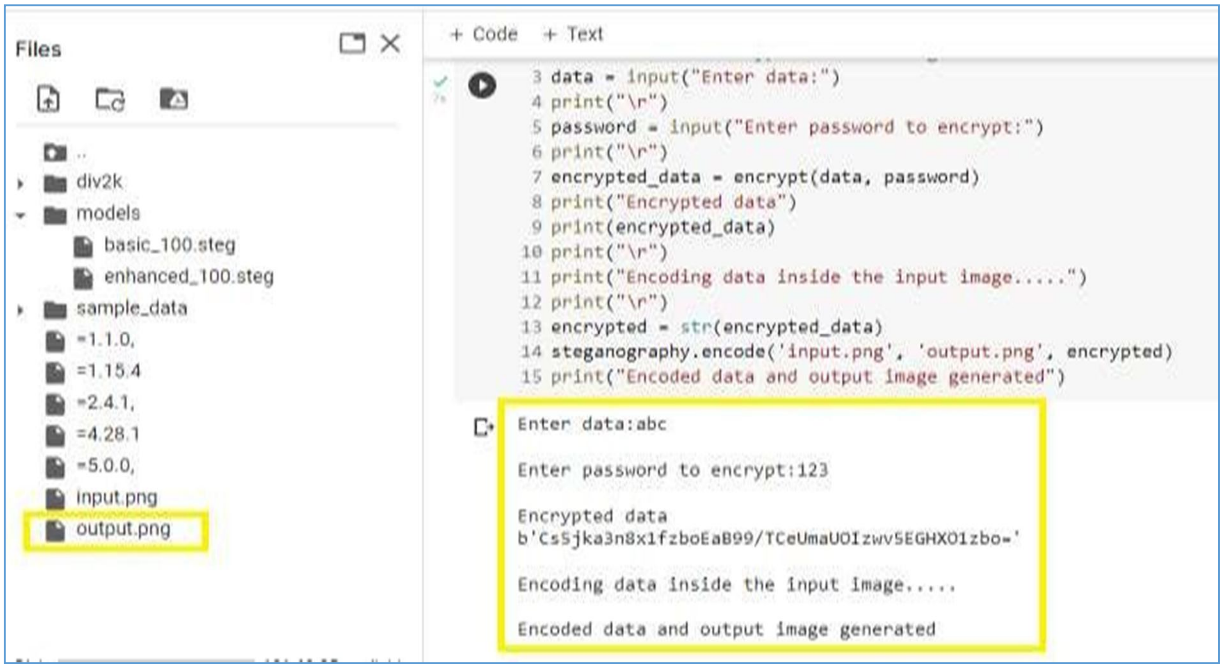
```
Files
..
div2k
models
    basic_100.steg
    enhanced_100.steg
sample_data
=1.1.0,
=1.15.4
=2.4.1,
=4.28.1
=5.0.0,
input.png
```

```
[19]
# Load the model
~~# from steganogan import SteganoGAN~~
steganography = steganography.load(path='models/basic_100.steg')
```

```
#data = string(data)
# First let us encrypt secret message
data = input("Enter data:")
print("\r")
password = input("Enter password to encrypt:")
print("\r")
encrypted_data = encrypt(data, password)
print("Encrypted data")
print(encrypted_data)
print("\r")
print("Encoding data inside the input image.....")
print("\r")
encrypted = str(encrypted_data)
steganography.encode('input.png', 'output.png', encrypted)
print("Encoded data and output image generated")
```

Figure: 4.4 Loading Basic Model File For Encoding

After the data gets converted to a cipher text, it is been hided inside the input image and an output image is been generated successfully which can be seen in the below fig 4.5.

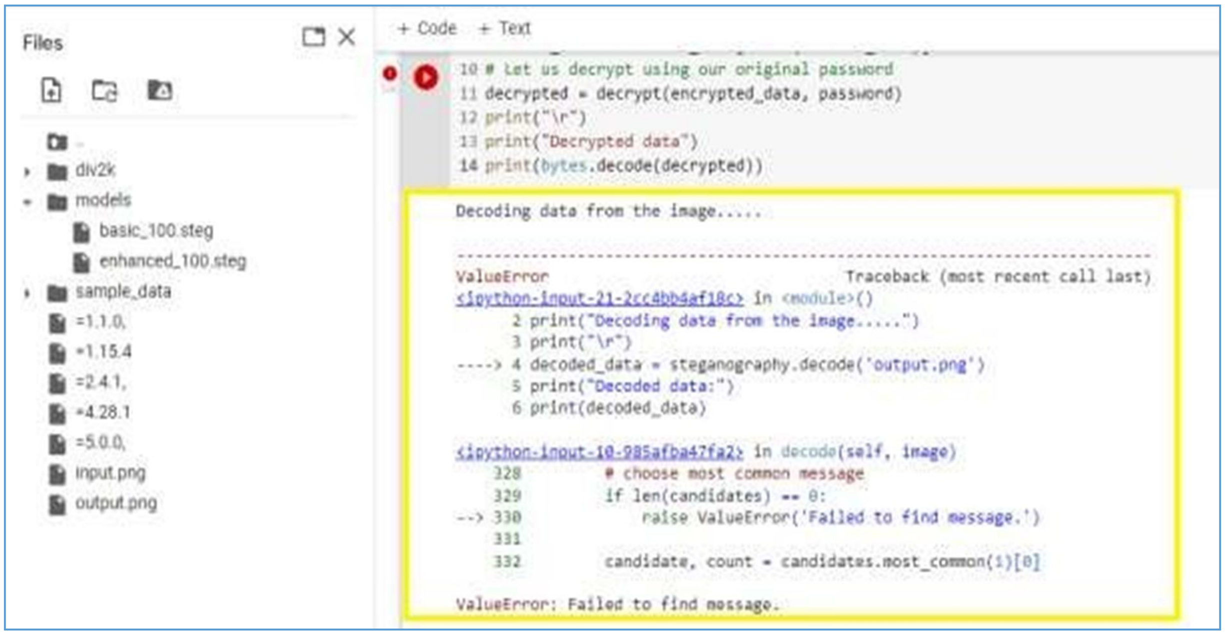


```
data = input("Enter data:")
print("\r")
password = input("Enter password to encrypt:")
print("\r")
encrypted_data = encrypt(data, password)
print("Encrypted data")
print(encrypted_data)
print("\r")
print("Encoding data inside the input image.....")
print("\r")
encrypted = str(encrypted_data)
steganography.encode('input.png', 'output.png', encrypted)
print("Encoded data and output image generated")
```

```
Enter data:abc

Enter password to encrypt:123

Encrypted data
b'Cs5jka3n8x1fzboEaB99/TCeUmaUOIzwv5EGHXO1zbo='

Encoding data inside the input image.....

Encoded data and output image generated
```

Figure: 4.5 Encrypting And Encoding Data Using Basic Model

Now the decoding is performed for the generated output image file which can be seen in the below fig 4.6.

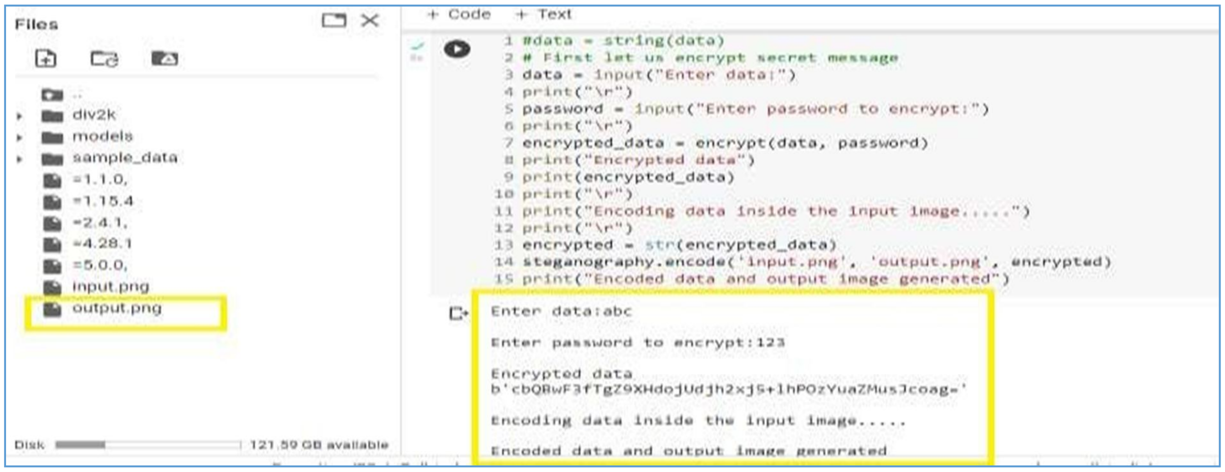


```
# Let us decrypt using our original password
decrypted = decrypt(encrypted_data, password)
print("\r")
print("Decrypted data")
print(bytes.decode(decrypted))
```

```
Decoding data from the image.....

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-21-2cc4bb4af18c> in <module>()
      2 print("Decoding data from the image.....")
      3 print("\r")
----> 4 decoded_data = steganography.decode('output.png')
      5 print("Decoded data:")
      6 print(decoded_data)

<ipython-input-10-985afba47fa2> in decode(self, image)
    328         # choose most common message
    329         if len(candidates) == 0:
--> 330             raise ValueError('Failed to find message.')
    331 
    332         candidate, count = candidates.most_common(1)[0]

ValueError: Failed to find message.
```

Figure: 4.6 Decrypting And Decoding Data Using Basic Mode

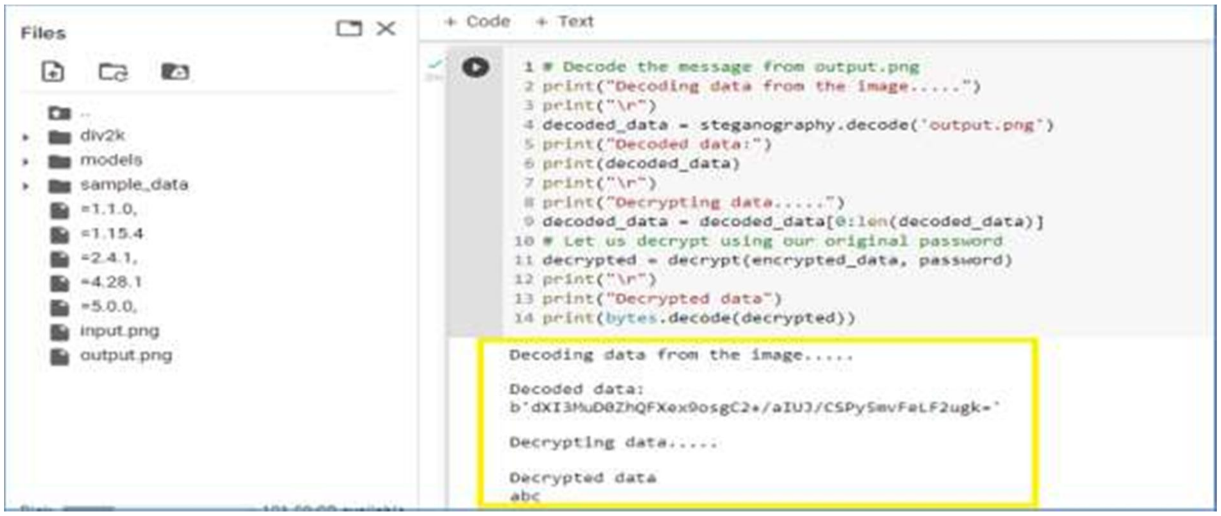
As the image file is processed for getting the data hidden inside, it throws error as no data found, which is because the bits per pixel was very less and hence data was not encoded successfully. Now, the enhanced model file is been loaded to perform the encoding of the data which can be seen in the below figure. As the encoding starts, the data gets converted to cipher text and an output image file is generated successfully which can be seen from the below fig 4.7.



```
#data = string(data)
# First let us encrypt secret message
data = input("Enter data:")
print("\r")
password = input("Enter password to encrypt:")
print("\r")
encrypted_data = encrypt(data, password)
print("Encrypted data")
print(encrypted_data)
print("\r")
print("Encoding data inside the input image.....")
print("\r")
encrypted = str(encrypted_data)
steganography.encode('input.png', 'output.png', encrypted)
print("Encoded data and output image generated")
```

```
Enter data:abc

Enter password to encrypt:123

Encrypted data
b'cbQRwF3fTgZ9XHdojUdjh2xj5+1hPOzYuaZMusJcoag='

Encoding data inside the input image.....

Encoded data and output image generated
```

Figure: 4.7 Encrypting And Encoding Data Using Enhanced Model



```
# Decode the message from output.png
print("Decoding data from the image.....")
print("\r")
decoded_data = steganography.decode('output.png')
print("Decoded data:")
print(decoded_data)
print("\r")
print("Decrypting data.....")
decoded_data = decoded_data[0:len(decoded_data)]
# Let us decrypt using our original password
decrypted = decrypt(encrypted_data, password)
print("\r")
print("Decrypted data")
print(bytes.decode(decrypted))
```

```
Decoding data from the image.....

Decoded data:
b'dXI3MuD0ZhQFXex9osgC2+/aIUJ/CSPySmvFeLF2ugk='

Decrypting data.....

Decrypted data
abc
```

Figure: 4.8 Decrypting And Decoding Data Using Enhanced Model

After successful working of the project, it is deployed in raspberry pi and DHT11 sensor is used to get the real time sensor data and successfully encoded and decoded the data to and from the image. the image generated during the encoding and decoding process is shown in the below fig 4.9.

Figure: 4.9 Encoding and Decoding process

In the existing system, with heavy data augmentation, FractalNet still cannot have the best result when comparing with those with other algorithms and Fractal networks are resistant to being too deep; extra depth may slow training. Thus, from the above result we have successfully achieved the results with more than 3 bits per pixel along with the real time encoding and decoding of the data.

V. CONCLUSION AND FUTURE SCOPE

Steganography is not intended to replace cryptography but rather to supplement it. If a message is encrypted and hidden with a steganographic method it provides an additional layer of protection and reduces the chance of the hidden message being detected. This project proposed that encrypt the IoT networks data by Cryptography method and hide the encrypted message inside an image file using Steganography method as well increases the number of bits that can be saved within a pixel of an image using the currently prevailing deep learning approach. Using which we have developed an algorithm effectively protect and secure the data. In the coming future, we will be implementing bits storage increasing and cryptography encryption and decryption performed in the next phase. We review the application of the data to be encrypted and decrypted for the data security field and it can promote to protect and secure the data with more accuracy. In this field there are more chance to develop or convert this project in many ways. The accuracy of the prediction will be increased by using different efficient techniques and algorithms. Thus, this project has an efficient scope in coming future to increase the payload of the data to be encrypted as well as safely decrypted to view the original message.

REFERENCES

[1] Brijesh Singh, Arijit Sur, PinakiMitra (2021),"Steganalysis of Digital Images Using Deep Fractal Network"Vol. 8, Issue: 3,PP:599-606.
[2] Hassaballah, Mohamed Abdel Hameed, Ali Ismail Awad, Khan Muhammad (2021),"A Novel Image Steganography Method for Industrial Internet of Things Security"Vol. 17, Issue: 11.
[3] Khari, Aditya Kumar Garg, Amir H. Gandomi, Rashmi Gupta, RizwanPatan, BalamuruganBalusamy (2020),"Securing Data in Internet of Things (IoT) Using Cryptography and Steganography"Vol. 50, Issue: 1,PP:73-80.
[4] Mehdi Boroumand, Student Member, IEEE, Mo Chen, Member IEEE, and Jessica Fridrich, IEEE (2019),"Deep Residual Network for Steganalysis of Digital Images" Vol.14, Issue: 5,PP:1181-1193.
[5] Pietro Tedeschi, Savio Scian calepore, Areej Eliyan, Roberto Di Pietro (2020),"LiKe: Lightweight Certificateless Key Agreement for Secure IoT Communications" Vol. 7, Issue: 1,PP:621-638.
[6] Ru Zhang, Feng Zhu, Jianyi Liu, Gongshen Liu (2020),"Depth-wise separable convolutions and multi-level pooling for an efficient spatial CNN-based steganalysis " Vol.15.
[7] Songtao Wu, Sheng-huaZhong, Yan Liu (2020), "A Novel Convolutional Neural Network for Image Steganalysis with Shared Normalization",Vol.22, Issue: 1.
[8] Wei Lu, Junjia Chen, Junhong Zhang, Jiwu Huang, Jian Weng, Yicong Zhou (2020) "Secure Halftone Image Steganography Based on Feature Space and Layer Manju".
[9] Weixuan Tang, Bin Li, Shunquan Tan, Mauro Barni, Jiwu Huang (2019), "CNN-based Adversarial Embedding for Image Steganography"Vol.14, Issue:8.