Content Adaptive Steganography based on Randomized Algorithm and Sequence to Sequence Model Text Generator

by

Examination Roll:
Fabliha Noshin Sithi(172470)
Shawlat Hilmee(172477)
MD Nuruzzaman Sojib(172496)

A Researech Project Report submitted to the
Institute of Information Technology
in partial fulfillment of the requirements for the degree of
Bachelor of Science in
Information Technology

Supervisor: K M Akkas Ali

Institute of Information Technology
Jahangirnagar University
Savar, Dhaka-1342

March 2022

DECLARATION

We hereby declare that this thesis is based on the results found by ourselves. Materials of work found by other researchers are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Fabliha Noshin Sithi	Shawlat Hilmee	MD Nuruzzaman Sojib
Roll:172470	Roll:172477	Roll:172496

CERTIFICATE

This is to certify that the thesis entitled **Content Adaptive Steganography Based on Randomized Algorithm and Sequence to Sequence Model Text Generator** has been prepared and submitted by **Fabliha Noshin Sithi**, **Shawlat Hilmee**, and **MD Nuruzzaman Sojib** in partial fulfillment of the requirement for the degree of Bachelor of Science (honors) in Information Technology on March 2022.

K M Akkas Ali
Supervisor

Accepted and approved in partial fulfilment of the requirement for the degree Bachelor of Science (honors) in Information Technology.

Prof. Dr. Mohammad Abu Yousuf	Dr. Shahidul Islam	Dr. Rashed Mazumder
Chairman	Member	Member

Prof. Dr. Md. Hasanul Kabir
External Member

ABSTRACT

In the modern world, data is the most important asset that an individual or an organization can possess. As a result, data security has turned out to be the most vital issue in data communication. Data security measurements are essential to restrict data from being misused in any aspect. Cryptography and steganography are two of the most popular security methods. Cryptography deals with data encryption whereas steganography handles data hiding. Though both cryptography and steganography can be used individually to provide security for data, combining these techniques turns out to be much more resilient to attacks. This paper describes a combined technique of cryptography and steganography in addition to randomized algorithm and automatic generation of cover texts using sequence to sequence model. For embedding, The proposed method firstly converts each character to a random numerical counter-part. Each of these numbers point to a unique phrase or group of words which is provided into the text generator for generating a whole sentence that works as the cover text.For extracting the secret text, the exact opposite of the embedding process is performed to extract the secret text.

Keywords: Cryptography, Steganography, Sequence to Sequence Model, Randomized Algorithm, GPT-2, Transfer Learning, Text Generation.

LIST OF ABBREVIATIONS

IIT	Institute of Information Technology
JU	Jahangirnagar University
ML	Machine Learning
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
GPT-2	Generative Pre-trained Transformer 2
TL	Transfer Learning
ER	Embedding Rate

LIST OF FIGURES

Figure

TABLE OF CONTENTS

CHAPTER I

Introduction

1.1 Overview

In a world where data is the key feature in every aspect, it is essential to provide security measures while transmitting data over an open channel. The primary security standards that a system has to follow are described by three characteristics: integrity, confidentiality, availability. Security of data can be compromised in the forms of data theft, data tampering, falsifying identities, password-related threats, etc.

The most significant factor in data security challenges is considered to be data leak prevention, with 88 per cent of critical and very important challenges. Likewise, Data Protection and Segregation carry 92 per cent significance on security challenges[1]. Between 25 May 2018 and 27 January 2020, organizations identified a total of 160,921 personal data breaches[2]. The average price of a security breach has already been estimated to be 3.86 million[2].] Data encryption is an appropriate fix to overcome data security challenges. It is effective to encrypt data or take some other measurements to hide the data before sending it to the receiver.

1.1.1 Cryptography

Cryptography is mainly a process where a message scrambles and can not be understood by anyone. It is primarily a secure message exchange among transmitter and recipient. Without a transmitter or a recipient, it can not be understood by anyone else. To understand this message, one needs to know the Key where the Key helps to decrypt the message. This encrypted message is basically used for confidential data transfer and security purposes. In our daily life, many people use cryptography; they don't even know they are using it. For security purposes or sending any confidential information, many people use cryptography. It can be used for good and bad purposes.It can be used in destructive works, which is really not good. For its extremely useful, it is also considered highly fragile, as cryptographic systems can become included due to a single programming or specification error[3]. Two types of cryptography systems are exist. There are two types of cryptography: symmetric cryptographic encryption and asymmetrical key

cryptography[4]. Among these two, symmetric cryptography is the best and oldest technique.

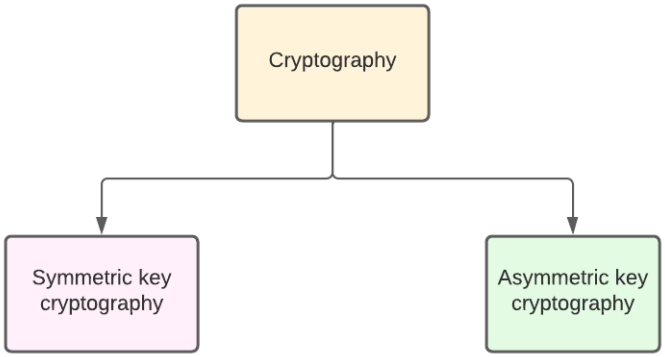

Figure 1.1: Classification of cryptography

- Symmetric Key Cryptography
  Symmetric Key is known as the Secret Key. It's also called shared key cryptography. In this process, when a sender sends a message, he sends a key and then the receiver receives the same Key for encryption and decryption.

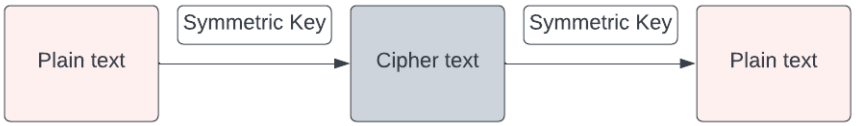

Figure 1.2: Symmetric cryptography

- Asymmetric Key Cryptography
  Asymmetric Key is a process where the sender sends a secret key to the receiver; then the receiver receives a different key. In asymmetric encryption and decryption, here we used two different keys. One is called a private Key, and another is a public Key. These two keys linked each other mathematically[5].

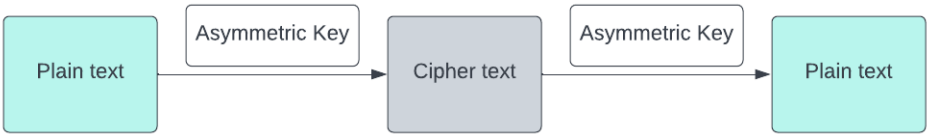

Figure 1.3: Asymmetric cryptography

1.1.2 Steganography

Steganography is mostly used to portray impenetrable communication. With the domains of data security and cryptography, we primarily encrypt information until no one except the sender and recipient can decrypt it; however, in steganography, we aim to conceal data instead of encrypt it. We can call them cousins in a Family because both of their majors is secure, confidential data. We can call it a hide and seek process. This hiding process occurs by changing the last bit of each of those bytes to hide one bit of data. Basically, steganography is an archaic subject, but often we are given in terms of the prisons problem in modern steganography. In the prison's problem, two inmates named Alice and Bob wanted to escape the prison, so they wanted to communicate with each other, but the warden always kept an eye on prisoners. If he suspects anything about their escape plan, he will punish them. So Alice and Bob had to communicate with each other with secret messages, and that's why they embedded their message and made it invisible. Thus they communicated with each other, and the warden didn't find it out. Relying on the Algorithm's secret is typically not seen as a good idea. Alice and Bob share a secret key in private key steganography, which is used to encrypt the communication. In private, Key basically uses a password which is used to seed a pseudorandom generator to choose where the secret message should be included in the cover object. In a public Key, the sender and receiver know each other's public Key[6].

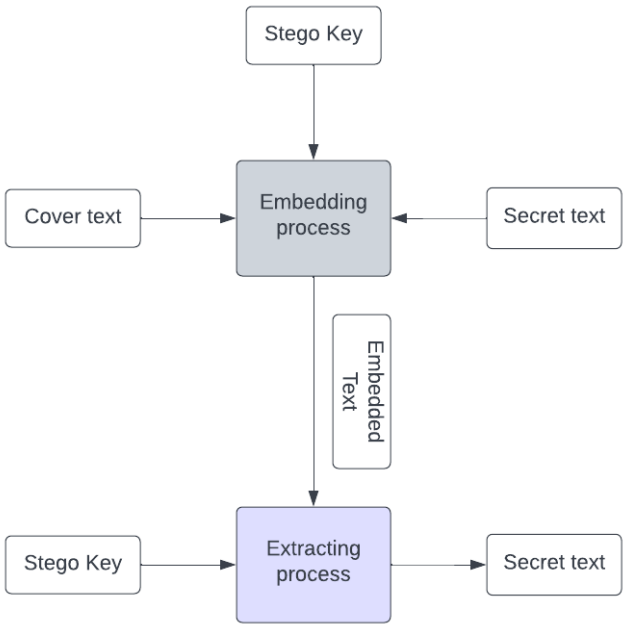

Figure 1.4: The process of steganography

Steganography is classified into various types according to its importance and goals. These are based on cover media. In-Text steganography payload is hidden by text. This text can

hide by many shifts such as line shifting, word shifting. In Image, steganography payload is hidden by Image. This can be hidden by image files such as JPEG, GIF, PNG. In the video, the steganography payload is hidden by Mp4. Video is a combination of images and video. So this file is huge in size. This huge amount of secret data can be embedded in video files[7]. In audio, the steganography payload is hidden by map3 or Wav. In this process, the message is embedded by shifting the binary sequence of that file.

In-Network steganography payload is hidden by a network protocol such as TCP, IP, UDP. The ISO layer network model includes conversion channels which can be utilized for steganography of network protocols[7].

1.1.3 Adaptive Steganography

In steganography, we hide confidential data for security. For hiding data, we use Image, audio, video, and network steganography. Using these sometimes faced some difficulties like embedding model textured or some noisy regions. To prevent these problems, we actually use adaptive steganography. The cost of altering each pixel is generally defined first, and then the adaptive steganographic strategy is executed. Encoding the hidden message while keeping the total to a minimum of all modified pixel costs. Efficient coding algorithms can incorporate the targeted payload with such an estimated distortion around the associated rate-distortion bound's minimum feasible values. Is it mainly a comparison between stego objects and cover objects[6].

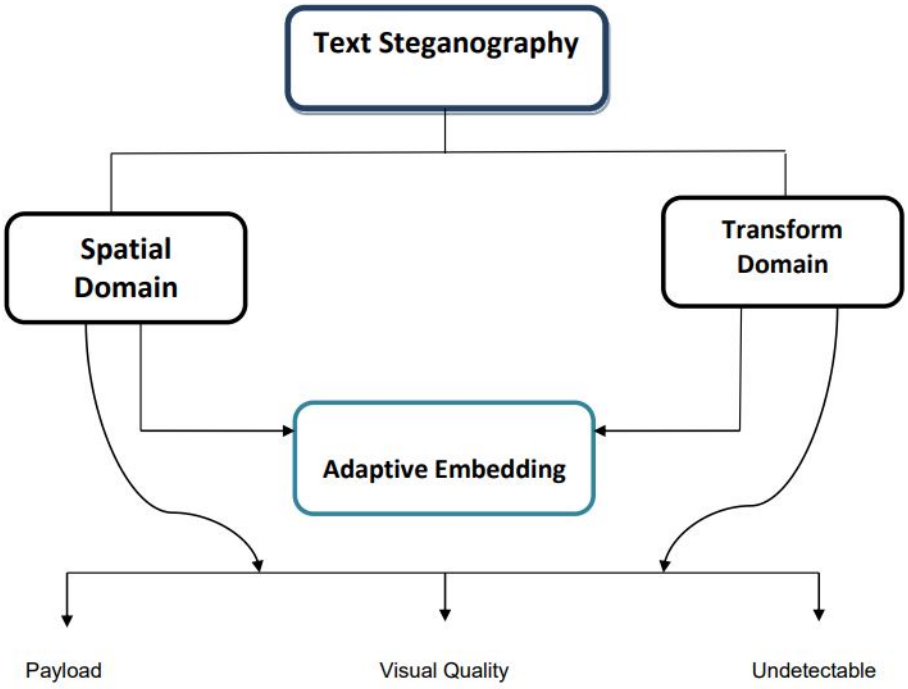

Figure 1.5: Adaptive steganography

1.1.4 Machine Learning and Text Generation Steganography

Machine learning and deep learning models can be very convenient in terms of information hiding and generating stego texts or images. The Algorithm used to hide information can be as crucial as the carrier[8]. Models of deep learning work as black boxes where the processing of output from the input is complicated as unexpected outcomes may arrive. However, in detecting steganography images, deep learning models are used [9].

Because of higher-level security and visible quality, adversarial architecture is extensively used [10]. The heart of the adversarial architecture happens to be an adversarial neural network that can be tuned up from a regular steganalysis network or a methodical CNN[11].

Synthesis technology is another intriguing study area in deep hiding in safe steganography. Unlike the embedding-based systems, synthesis technology does not require any change since containers are formed straight by secret[12].

LFM (light field messaging)[13] is a practical application for information concealment since it embeds, transmits, and receives concealed data in the form of an image displayed on the screen, which is subsequently captured by the camera. LFM is frequently referred to as photographic steganography.

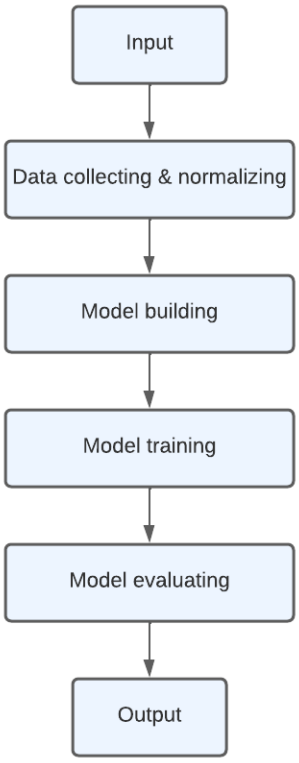

Figure 1.6: Machine Learning Process for Steganalysis

As natural language processing(NLP) is progressing over the years, generating stego text auto-

matically to hide information is becoming famous. These steganographic algorithms work without inherent moderation of cover text. Steganographic texts are generated according to the data that has to be embedded throughout the process. In comparison with the traditional approaches, these steganographic meth-ods generally provide well-built anti-interference and better embedding capability[14].

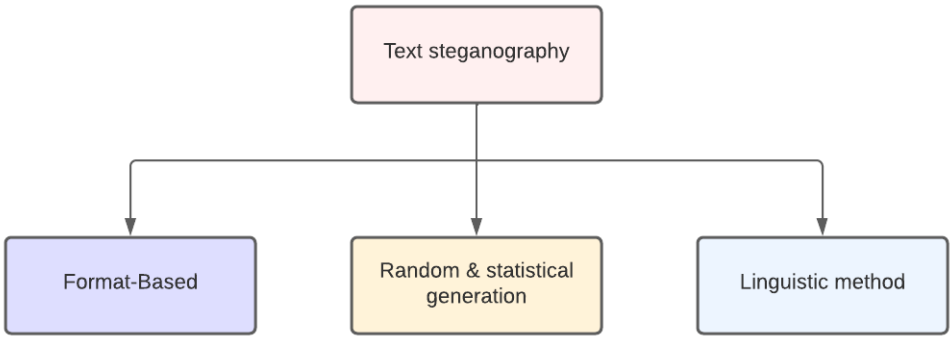

Figure 1.7: Types of Text Steganography

1.2 Problem statement

This article proposes an end-to-end message passing security system that consists of encryption, randomization, and machine learning-based steganography. Both the encoder and decoder section of the proposed method consists of these three parts. While steganography or encryption alone provides sufficient protection against several cyberattacks, merging several methods can result in a better and more secure message passing protocol.

1.3 Limitations of Existing Works

The following drawbacks in present systems have been discovered by examining several concurrent publications of similar topics.

i Higher space per bit embedding ratio

ii May not always provide security of information.

iii The ability to hide information is limited.

iv Manually generated stego texts may easily be detected.

1.4 Objective

This proposed system will serve the following objectives:

i To combine the idea of data encryption,randomization and data hiding for better security.

ii To generate cover texts automatically.

iii To provide better resistance against steganalysis and cryptanalysis.

1.5 Research Outline

The remainder of this essay is organized as follows. **Chapter II** examines the literature search as describes the previously published works. **Chapter III** introduces the proposed system framework, and finally, a conclusion is drawn in **Chapter IV** .

CHAPTER II

Literature Review

2.1 Introduction

For many years many have worked in this field of data security. From the start of data transferring, it is considered a vital issue. This data security mainly hides confidential data. Here we encrypt data with two processes. Data is encrypted by two processes. One is cryptography, and another one is steganography. Cryptography is mainly a secret conversion between sender and receiver. A message is sent with a secret key. And only the sender and receiver are aware of this Key. Steganography is a technique in which the sender conceals data using text, image, video, audio, or a network.

2.2 Cryptography and Steganography

Khari et al.[15] proposed a data security scheme for IOT which uses EGC (Elliptic Galois Cryptography) to encrypt input data and a Matrix XOR steganography method to hide the encrypted input inside a low resolution image. Their proposed method also applies Adaptive Firefly algorithm to minimize the choice of cover pieces in an image.

An adaptive least significant bit(LSB) matching comparison method was suggested by Luo et al.[16]. Their method detects horizontal and vertical edges as the edge regions in an image are found by calculating the divergence between successive pixels.

Hunan et al.[17] proposed an adaptive steganography method in terms of text steganography. They have proposed a distortion that includes statistical and linguistic distortion. Word frequency is the key measure in statistical distortion. To lessen the embedding impact, the Syndrome-trellis code has been used.

Por et al.[18] came up with a format oriented steganography method where the idea is to make use of the space character in the cover text. By regulating the alteration, the space symbol is altered to insert hidden information. The visual quality of the stego text may be maintained by adjusting the magnitude of the space character mutation.

2.3 Machine learning in Steganography

Goodfellow et al.[19] utilized an artificial neural network to recreate a game that included an image generating network and a discriminatory network that determined if a picture was genuine or synthetic. Generative Adversarial Networks is the name given by the authors to this type of network (GAN approach).

Yedroudj et al.[20]have employed a fundamental model in a simple meta-architecture and immediately added an adversarial discriminator to enhance the model's resilience to steganalysis. Furthermore, because of discriminator's effectiveness is constrained by the antagonistic training strategy, these strategies do not overcome separately trained reliable findings.

Li et al.[21] have used a deep learning model, which dynamically creates a systematic emotional dictionary that serves as the foundation for text emotional steganography information security. It also considers the information concealment algorithm, which is based on emotive word replacement, matrix encoding, and other techniques.

2.4 Text generation based steganography with Machine Learning

To create text steganography, Yu et al.[22] recommended using a unique type of poetry known as Song Ci, which originated in ancient China's Song Dynasty. This was the first text steganography method to use a Chinese artistic technique. Their approach, on the other hand, merely recombines a new Ci-poetry by selecting relevant words from an existing Song-Ci poem. Because it does not create new Ci-poems from scratch, its use may be limited. Furthermore, throughout the creation process, their technique picks words at random, disregarding word collocations and line connections. This frequently leads to the lack of a core subject for Ci-poems created by their algorithms which might prompt suspicion and decrease security.

LI et al.[23] have suggested a method to create a particular topic graph and use a graph encoding structure to encode the idea and content of the graph into vectors. The stego text is then generated and transmitted to the decoder. The transformer architecture is used to encode the secret data bitstream and a particular matter graph using a graph embedding structure.They use the decoder secret sentence to focus on information integration encodings at each decoding sampling interval. The output is picked from the language of the decoder or by transferring the produced vectors.

Yang et al.[24] have developed a linguistic steganography based on the RNN-Stega algorithm (Recurrent Neural Networks). It can build high-quality text covers automatically from a secret bitstream that will be embedded.But there is some limitations as the standard range of contextual information that RNNs can access is in practise is quite limited.

Yang et al.[25] presented a steganography method that employs Knowledge graph in the assistance of creating sentences that hides hidden information. This approach grants authorisation

to a certain extent over the semasiology of the created texts. They have attempted to enhance the grammatical accuracy and semantic quality of the steganographic texts compared to certain formerly existing methods.

CHAPTER III

System Model

In this paper, a new technique of text steganography is presented. This approach uses the Generative Pre-trained Transformer 2 (GPT-2) and Transfer learning (TL) to generate cover texts using randomly selected words and phrases to hide the payload. This method uses a collection of randomly selected phrases and words previously generated, and a set of lists contains several lists of various lengths. Each list contains randomly generated numbers within a pre-defined range. The cover text is generated in such a way that it remains grammatically coherent and meaningful. It is nearly impossible to detect the words of phrases from the whole sentence that hides the intended message.

3.1 Introduction

The system workflow is discussed in section 3.2 and proposed system architecture and the mechanisms used for implementing the proposed work is discussed in section 3.3 and 3.4.

3.2 Basic Workflow

For simplicity, the basic workflow of the implemented method is divided into two blocks: embedding and extracting.

3.2.1 Embedding

Embedding is the process where the secret text is encrypted and hidden in a cover text generated with the GPT-2 text generator. The process of embedding can be described in four phases:

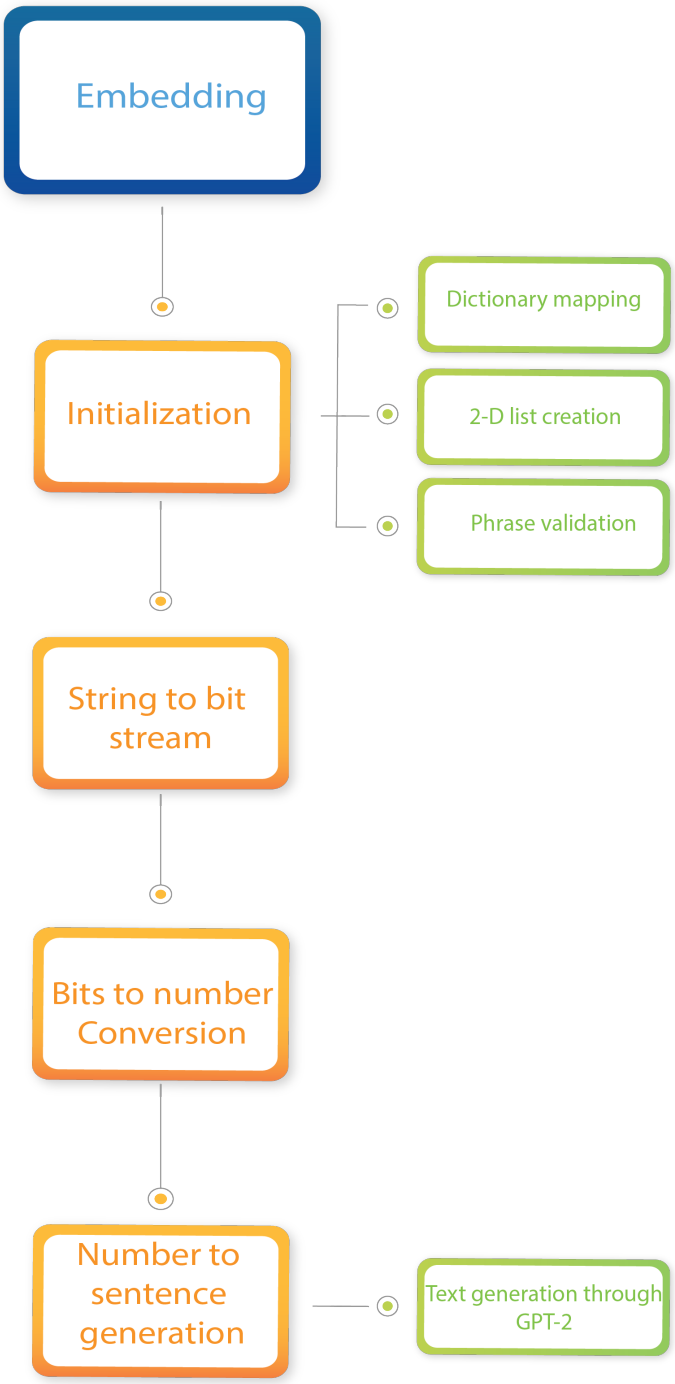

Figure 3.1: Embedding process

3.2.1.1 Initialization

At first, we initialize the process by defining some necessary global variables and lists. A two dimensional list is created as a storage for random variables between a pre-defined range. A specific number of unique phrases are taken from a large collection file of phrases for sentence generation. The phrases are mapped and inverse mapped into two dictionaries against a set of random generated numbers in such a way that each key and value pairs remain exactly the same for both dictionaries.

3.2.1.2 String to bit stream

The characters that are to be hidden are converted to a binary stream according to their ASCII values. The converted bits are added one after another to form a bit sequence.

3.2.1.3 Bits to number conversion

In each iteration, n numbers of sequential bits are taken from the binary stream and converted into corresponding decimal values. Each decimal value corresponds to a list index from the set of lists. Finally, a random decimal value is selected from each corresponding list. Thus, only one randomly generated decimal value is selected from the pre-defined set of lists for each character of the input text.

3.2.1.4 Number List to generated paragraph

Each of the selected numbers is then used to point a unique group of words from the pre-defined collection of phrases by appropriate number to string mapping. Each group of words is then fed into the GPT-2 model for an entire sentence generation. Semantic coherence is found as previously generated sentences are also provided into the text generation model along with the following selected group of words.

3.2.2 Extracting

Extracting is the process of finding out the secret text from the embedded text. The process of extracting can be described in four phases:

Figure 3.2: Extracting process

3.2.2.1 Sentence to number

Each sentence of the embedded paragraph is compared with the String to number dictionary to find out the matching phrase in the sentence. The value from for the matching phrase is taken and added to a list.

3.2.2.2 Number to index

The characters that are to be hidden are converted to a binary stream according to their ASCII values. The converted bits are added one after another to form a bit sequence.

3.2.2.3 Index to binary

Each selected index (integer value) is converted into n bit binary equivalent and all the bits are added to form a binary sequence.

3.2.2.4 Binary to character

In each iteration, 8 bits are taken sequentially and converted to equivalent character. All the characters are appended together to form the extracted secret text.

3.3 Hash Table

The hash matrix is a table which defines the conceptual data type linked array, which is a format that may map names to entries. A hash table makes use of a hash function to generate an index, often referred like a hash code, within a collection of bins or positions where the desired item can be found.

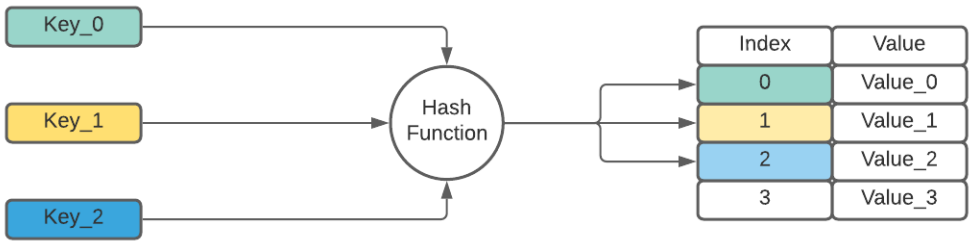

Figure 3.3: Hash Table

3.4 Sequence to Sequence Model

When the input length and output length are both variable, a model series to sequencing attempts to translate an input data horizontally to a given outcome. Pattern to models are required for services like google Translate, dialogue devices, including online conversations.

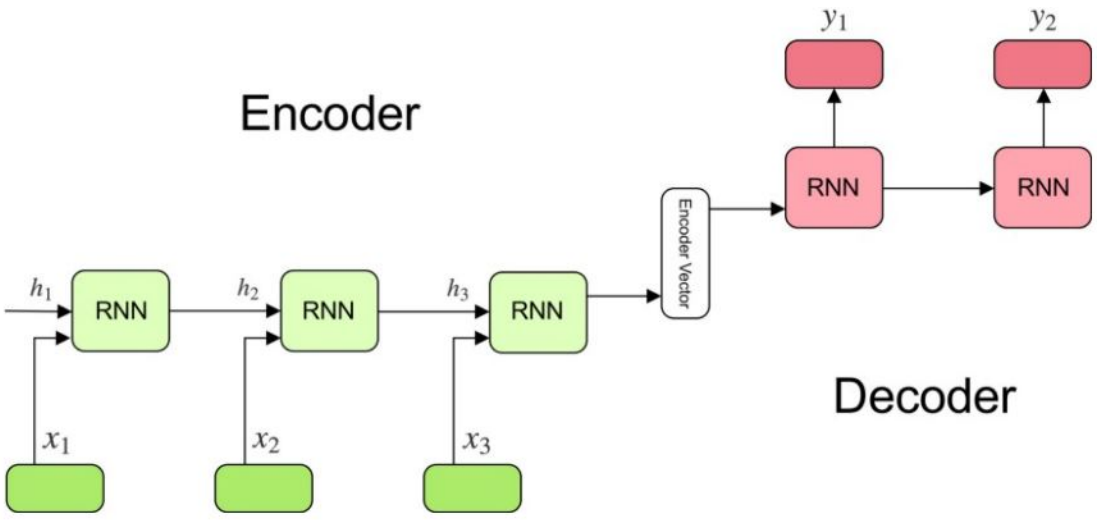

Figure 3.4: Sequence to Sequence Model

The model is made up of three pieces: encoder, vector encoder and decoder.

- Encoder
  A stack of several repeating devices where each receives a single input sequence element and

gathers and spreads information for the input sequence. In the responding of questions, all terms in the query are collected inside the input sequence. The i of each x_1 is the order of that word.The hidden h_i states are calculated using the following formula

$$h_t = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t)$$

We merely apply the weight to $h_{(t-1)}$ and input vector x_t in a previously concealed state.

- Encoder Vector
  This is the ultimate hidden state generated by the model's encoder. The above formula is calculated to find it out. In order to assist the decoder to accurately anticipate this vector it will contain the information for all input components. The decoder portion of the model functions as the initial hidden state.

- Decoder
  a stack of many repeating units in which a y_t output is predicted at a time t. Each unit accepts the secret state of the preceding unit and creates and produces and produces its own hidden condition. The output sequence is a collection of all words from the response in the question answering issue. Every word is shown as y_i where i is the order. Any hidden h_i state is calculated with the formula

  $$h_t = f(W^{(hh)}h_{t-1})$$

  We use the previous concealed state to calculate the next one, as we can see. The output y_t is calculated with the formula at step t

  $$y_t = softmax(W^S h_t)$$

  Softmax function is a generalization of logistic functions used to construct a vector to decide the ultimate performance.

3.4.1 Generative Pretrained Transformer 2

GPT-2 is an artificial intelligence language model that implements a deep neural network, a transformer model to be precise. A language model is an extension of a machine learning model that can predict the next word by analyzing a part of a sentence. GPT-2 is open-source and was developed by OpenAI in 2019. It is trained on a vast dataset. The architecture of the GPT-2 is quite identical to that of the decoder-only transformer. GPT-2 can predict the next item in a random sequence. It is often used as a text generator, a text translator, and a paragraph summarizer. Unlike LSTM and RNN, GPT-2 implements an attention mechanism that provides attention to predicting the most correlated portions of the input text.

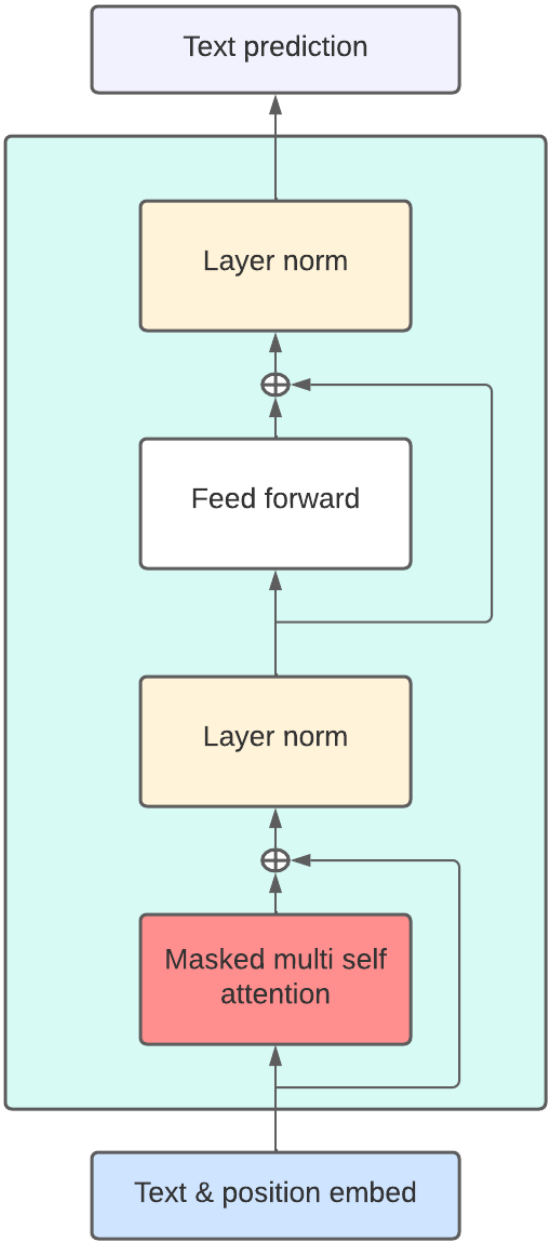

Figure 3.5: GPT-2 Architecture

CHAPTER IV

System Environment and Tools

4.1 Python

Python is an object-oriented, strong programming language with dynamic typing and is interpreted. Its high-level data structures and dynamic semantics make it particularly well-suited for faster development and usage as a scripting or connector language for connecting elements. Python's straightforward, easy-to-learn format puts a premium on readability, which reduces the value of program maintenance. In addition, Python allows modules and packages, which promotes the modularity and reuse of code in programs.

4.2 Numpy

NumPy, which refers to Numerical Python, is a package that contains multidimensional array structures and algorithms for handling them. NumPy enables the execution of mathematical and logical computations on arrays. NumPy's main significant object is the ndarray type, which would be an N-dimensional array. It is a term that refers to a group of identical goods. A zero-based address is used to retrieve items from the collection. Each element inside andarray has the same physical dimensions as the memory block. Each item in the ndarray would be a data-type object (called a dtype). Any element retrieved out of andarray object (by splitting) is expressed in Python by only a Python object from one of the array scalar types. .

4.3 Pandas

Pandas is a Python library that provides a quick, versatile, and creative data model that makes it natural and straightforward to deal with related or marked data. It is intended to serve as the foundational high-level structural unit for performing real-world information processing in Python. Furthermore, it aspires to stand as the most robust and versatile open-source data processing tool accessible in any language. It is now well on its path to accomplishing this objective. The two

fundamental representations of data structures in pandas are Series (1-D) and DataFrame (2-D). They both support the great majority of everyday use cases in mathematics, social studies, economics, and various engineering disciplines.

4.4 Request module

The applications module allows users to send Requests to the server using Python.The Requests library is a fundamental feature of Python that allows you to make HTTP requests to an apply Different. Whether it's Http APIs or web harvesting, a basic understanding of requests is required before progressing with these platforms. When a request is made to a URI, it responds with a respond. Python requests include built-in management functionality for both the client and the server.

4.5 Regular expression

Python's regex capability is contained in the re module. The re module offers a plethora of useful features and procedures, the majority of which will be covered in the following tutorial in the sequence A RegEx, or Notation, is a collection of symbols that together define a search query. RegEx could be used to determine whether or not a string matches a specified search query.

4.6 Json module

JSON is a standardized text-based method for encoding structured data that is based on the Object notation language. It is frequently used for data transmission in web applications.Python includes a library called json that handles JSON. To make use of this feature, we need to install the json module into our Python script. The content in JSON is represented by a quoted string containing the value from the vital mapping contained within

CHAPTER V

Implementation of Proposed System Model

5.1 Embedding secret text

The implementation algorithm of embedding is divided into several functional blocks for better understanding. These functional blocks are described with appropriate algorithms. All the blocks work together, complete proper actions and communicate with each other to cumulatively embed the secret text.

Secret text	Binary stream	Decimal equivalent for each n(8) bits	Selected random number	Selected group of words	Generated sentence
b	01100010	98	686	Abstraction is often	Abstraction is often used to restrict access to goods on the basis of gender.
ba	0110001001100001	98 97	188 402	He waited for the bus Watching the	He waited for the bus to arrive and was surprised to see that it was headed to the hotel. Watching the bus leave, he finally decided that he would not be leaving in one frame when he opened the doors.
bag	011000100110000101100111	98 97 103	543 224 485	Cats are good Everyone says They say that dogs	Cats are good for them. Everyone says we should just do this to them because you're going to find a new breed when you start taking the first steps. They say that dogs aren't as stupid.

Figure 5.1: Embedding process

5.1.1 Embed

In this section, all the global variables are initialized. The secret text is taken as the input. Process initialization is done through invoking the method Initialize (). The *StrToBit()* function is called and it takes the secret text as the only parameter. This function returns a binary bit sequence *BitStream* by converting the characters of the secret text into binary equivalents. Another function *BitsToNum*() is invoked. It takes the *BitStream* as its parameter, takes n bits at a time sequentially, converts it to corresponding decimal value, adds it to a list and returns the list as *NumberList*. Finally, this *NumberList* is passed as an argument in the function *GeneratedSentece()*. This function produces the generated text output by performing appropriate actions based on each value in the *NumberList*.

Algorithm: Embedding

Result: EmbeddedOutput

Data: StrIn

initialization;

$n \leftarrow$ Number of bits to represent a character;

$AvgLen \leftarrow$ Average length of lists;

$Deviation \leftarrow$ a small number to randomize length of lists;

$c \leftarrow$ Constant value ;

$StrIn \leftarrow$ Input string of characters;

$BitStream \leftarrow$ Empty string variable;

$Range \leftarrow 2^n * AvgLen * c$;

$RandomNumbers \leftarrow$ 1-D list of random numbers from 0 to $Range$;

$NList \leftarrow$ Converted 2-D list from $RandomNumbers$ with 2^n 1-D lists;

$MapNumStr \leftarrow$ 1-D Empty list;

$MapStrNum \leftarrow$ 1-D Empty list;

$EmbeddedOutput \leftarrow$ 1-D Empty list;

$NumberList \leftarrow$ 1-D Empty list;

$Phrases \leftarrow$ 1-D Empty list;

$ExtractedNumbers \leftarrow$ 1-D Empty list;

$BitStream2 \leftarrow$ Empty string variable;

$ExtractedOutput \leftarrow$ Empty string variable;

Initialize();

$BitStream \leftarrow$ StrToBit(StrIn);

$NumberList \leftarrow$ BitsToNum(BitStream);

$EmbeddedOutput \leftarrow$ GeneratedSentence(NumberList);

5.1.2 StrToBits()

On being invoked, *StrToBits()* takes the secret input text as its argument, takes each character of the string and converts it to equivalent binary byte. It return a string *BitStream* the binary values of each characters are added up.

Algorithm: StrToBits()

Result: BitStream

Data: StrIn

$BitStream \leftarrow$ Binary conversion of StrIn;

5.1.3 BitsToNum()

When invoked, this function takes n number of bits sequentially from the *BitStream* in each iteration, converts the n number of bits to corresponding decimal value, and adds those values into a list *NumberList*.

Algorithm: BitsToNum()

Result: NumberList

Data: BitStream

$i \leftarrow 0$;

while $i < length(Bitstream)$ **do**

$bits \leftarrow BitStream[i : i + n]$;

$dec \leftarrow$Decimal equivalent of $bits$;

$list \leftarrow NList[dec]$;

$size \leftarrow$ Length of $list$;

$index \leftarrow$ Random integer between 0 and $size$;

$NumberList \leftarrow$ Append $list[index]$ to $NumberList$;

$i \leftarrow$i+n;

end

5.1.4 RandomNumberGeneration()

There is a block method *RandomNumberGeneration()* that returns a random number between 0 and Range. Range is directly related to the number of bits used to represent a character (n), average length(*AvgLen*) of a list in the NList, and a constant defined value(c). Moreover, the random values that are generated are also added to a global list named *RandomNumbers*.

Algorithm: RandomNumberGeneration

Result: number

number ← A random generated integer number between 0 to Range;
if *number does not exists in RandomNumbers* **then**
    RandomNumbers ←Add *number* to *RandomNumbers*;

5.1.5 FillupNList()

The 2-D list *NList* can be considered as a collection of 2^n number of 1-D Lists, where each list is of random length. This technique adds an extra level of randomness which makes the algorithm much robust against detection. Each element of every the 1-D lists are unique and generated by invoking the *RandomNumberGeneration()* method. Finally, all the single dimensional lists are made a collection to build up the *NList*.

Algorithm: FillupNList()

i ← 0;
while *i*< 2^n **do**
    RandomLength= *AvgLen*+ (Random generated integer between -*Deviation* and +*Deviation*);
    Tlist ←Empty lists;
    while *RandomLength* > 0 **do**
        RandomNumber ←RandomNumberGeneration();
        Tlist ←Append *RandomNumber* to *Tlist*;
        RandomNumber ←RandomNumber−1;
    end
    NList ←Append *Tlist* to *NList*;
    i ←i-1;
end

5.1.6 Initialize()

The first thing Initialize() does is it invokes another method called *FillupNList()*. NList is initialized as an empty 2-dimensional List and *FillupNList()* generates and places random values in the *NList*. After that, a specific number(same as the length of *RandomNumbers*) of phrases(collection of words) from are loaded and uniquely mapped into a dictionary *MapNumStr* against the values of the list *RandomNumbers*. Similarly, the values of the *RandomNumbers* are uniquely mapped into another dictionary *MapStrNum* against the loaded phrases.*MapNumStr* and *MapStrNum* are mapped exactly inverse of one another. It means that each key and value pair of both the dictionaries will be exactly the same.

Algorithm: Initialize()

FillupNList();
$Len \leftarrow$ Length of $RandomNumbers$;
$Phrases \leftarrow Len$ Number of phrases;
$MapNumStr \leftarrow$ Converted to dictionary{Keys: $RandomNumbers$, Values: $Phrases$};
$MapStrNum \leftarrow$ Converted to dictionary{Keys: $Phrases$, Values: $RandomNumbers$};

5.1.7 GeneratedSentence()

This method serves an important purpose of selecting appropriate phrases for sending into the GPT-2 text generator through another function TextGeneration(). For a value in *RandomNumbers*, first we find out the single dimension list from NList that has the same index as the value in *RandomNumbers*. Then we select a value from that single dimensional list in a random manner and find the phrase that is mapped against this found value. Finally, this phrase is sent to the Sequence to Sequence text generator model for an entire sentence generation.The same exact process is repeated for every single value in *RandomNumbers*. We can find noticeable semantic correlation between the generated sentences because we send the previously generated sentences with the selected phrases in every iteration, except for the first.Each phrases or group of words are unique. If there are more than two common words in two phrases, one of them is slightly changed by adding a 's' or 'e' at the end of it in appropriate scenarios that doesn't significantly change the meaning or structure of the phrase.

Algorithm: GeneratedSentence()

Result: EmbeddedOutput

Data: NumberList

LastSentence ← Empty string;

$i \leftarrow 0$;

while $i < length(NumberList)$ **do**

  word ←*NList*[*i*];

  NewSentece ←word;

  CurrentSentece ← Empty String

  if *CurrentSentence*==*EmptyString* **then**

    $CurrentSentence = TextGeneration(NewSentence, 0) +' .'$;

  else

    $NewSentence = LastSentence + word$

    $CurrentSentence = TextGeneration(NewSentence, 1) +' .'$;

  end

  id ←Empty List j←Length of *word*;

  while *j<length of CurrentSentence* **do**

    if $CurrentSentence[j] ==' '$ *or* $CurrentSentence[j] ==' .'$ **then**

      break;

    id ←*j appended to id;*

    if *Length of id>0* **then**

      CurrentSentence =word+CurrentSentence[id[length of id-1] : length of CurrentSentence]

    end

    EmbeddedOutput ←*CurrentSentence apeended to EmbeddedOutput;*

    CurrentSentence ←*LastSentence*

    j ←*j+1*

  end

  i ←*i+1*

5.1.8 TextGeneration()

This method binds the GPT-2 model with our algorithm through a API call. It takes two arguments, a phrase and a position. The phrase is fed to the model for generating complete sentences that will work as our cover media. The position is used for returning a particular sentence which has the same index as the position for further sentence generation.

Algorithm: TextGeneration()

Result: Words, Position

Data: Sentence

$ModelOutput \leftarrow$ Output from the GPT-2 model with input $Words$;

$JSON \leftarrow$ json file from $ModelOutput$;

$AllSentences \leftarrow$ List of all sentences from $JSON$;

$Sentences \leftarrow$Sentence with index $Position$ from $AllSentences$;

5.2 Extracting secret text

The implementation algorithm of extracting is divided into several functional blocks for better understanding. These functional blocks are described with appropriate algorithms. All the blocks work together, complete proper actions and communicate with each other to cumulatively extract the secret text from the embedded text.

Embedded text	Matched group of words	Number selected by reverse mapping(x)	Index of the list that contains x	Converted binary stream	Secret text
Abstraction is often used to restrict access to goods on the basis of gender.	Abstraction is often	686	98	01100010	a
He waited for the bus to arrive and was surprised to see that it was headed to the hotel. Watching the bus leave, he finally decided that he would not be leaving in one frame when he opened the doors.	He waited for the bus Watching the	188 402	98 97	0110001001100001	ba
Cats are good for them. Everyone says we should just do this to them because you're going to find a new breed when you start taking the first steps. They say that dogs aren't as stupid.	Cats are good Everyone says They say that dogs	543 224 485	98 97 103	011000100110000101100111	bag

Figure 5.2: Extracting process

5.2.1 SentenceToNumber()

This function takes *EmbeddedOutput* as it's argument. *EmbeddedOutput* is *splitted* into each sentences. Each sentence is compared with the *MapStrNum* to find out the matching phrase in the sentence. The key value from the *MapStrNum* for the matching phrase is taken and added to a list called *ExtractedNumbers*.

Algorithm: SentenceToNumber()

Result: ExtractedNumbers

Data: EmbeddedOutput

Sentences ←List of all the sentences in *EmbeddedOutput* ;

i ←0 ;

while *i*<*Length of Sentences* **do**

  p ←Matched phrase from *Phrases* that exists in Sentence[i] ;

  num ← Corresponding number of *p* in *MapStrNum*;

  ExtractedNumbers ←Add *num* to *ExtractedNumbers* ;

  i ←i+1 ;

end

5.2.2 IndexToBinary()

This method takes the *ExtractedNumbers* as the only parameter. For each value of *ExtractedNumbers*, the single dimensional list is selected from *NList* which contains that value. The index of the single dimensional list from *NList* is selected, converted to n bit binary, and added to *BitStream2*.

Algorithm: IndexToBinary()

Result: BitStream2

Data: ExtractedNumbers

i ←0;

while *i* < *Length of ExtractedNumbers* **do**

  l ←1-D list in *NList* that contains *ExtractedNumbers[i]*;

  index ← Index of *Nlist[l]*;

  *BitStream*2 ← Add converted binary equivalent of *index* to *BitStream2*;

end

5.2.3 BinaryToChar()

From the *BitStream2*, 8 bits are taken in each iteration, converted to equivalent character, and added to the *ExtractedOutput*.

Algorithm: BinaryToChar()

Result: ExtractedOutput

Data: BitStream2

i ←0;

while *i* < *Length of BitStream2* **do**

  byte ←8 bits from position i in *BitStream2*;

  character ← Convert byte into corresponding charachter;

  ExtractedOutput ← Add *character* to *ExtractedOutput*;

  i ←i+8;

end

CHAPTER VI

Result Analysis

The performance of the implemented approach was evaluated by experimentation in this section of the article.

6.1 Embedding rate

The percentage of concealed bits in a total generated text is referred to as embedding rate, also known as embedding capacity. For a steganography algorithm to be called efficient, it must have a high embedding rate. According to [26] The mathematical expression of ER is expressed as:

$$\mathrm{ER} = \frac{1}{L}\sum_{k=1}^{L}\frac{(X_k - 1)}{P_{s_k}}$$

Where L is the Number of sentences generated, X_k is the length of the $k-th$ sentence and the number of bits utilized by the $k-th$ sentence in the system is P_{sk}.

We have compared our method with three other existed steganography algorithms to show that we have achieved a better ER.

Table 6.1: ER comparison

Methods	ER(%)
[27]	0.33
[28]	1.0
[26]	2.73
Our method	7.39

6.2 Embedding time and Time complexity

Embedding time is the time taken to hide or embed certain number of secret bits into appropriate cover media. The less the embedding time is, the faster the algorithm operates. An efficient steganography algorithm tend to keep it's time complexity near O(n). We have calculated the runtime of our method against the values of N, where N is the number of bits to represent each character while embedding in our method.

Table 6.2: Time against N

N	Time(in sec)
1	124.76 ± 43.23
2	49.16 ± 17.7
3	34.86 ± 5.40
4	19.81 ± 2.21
5	17.91 ± 3.81
6	12.29 ± 0.014
7	12.25 ± 1.35
8	8.21 ± 1.47

The relation between time and N can be made more clear with a chart.

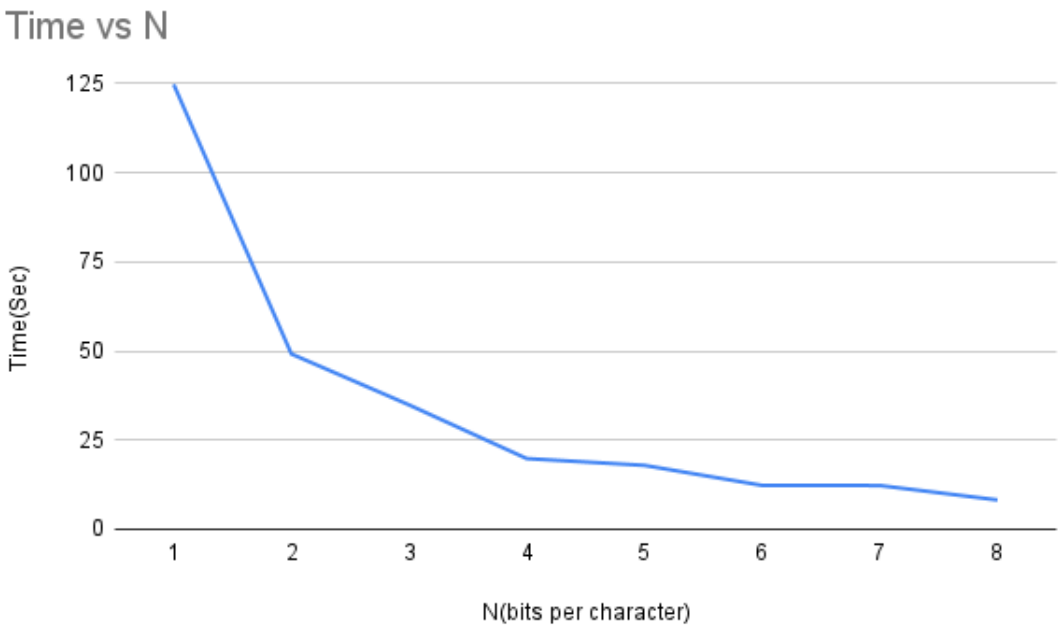

Figure 6.1: Change of embedding time to the size of N

CHAPTER VII

Conclusion and Future Work

This paper proposed and implemented a steganography method that includes encryption, randomization and deep learning text generator model. It is shown in our method that the sequence to sequence generator models can perform well with a good embedding rate and time complexity in terms of hiding data for the purpose of secure transmission. This method was compared to some existing similar methods and showed a better embedding rate. However, there are a lot of rooms of improvements in the implemented method. For future works, we intend to train the model more via transfer learning and assure more semantic coherence. And at the same time we look forward to reduce the embedding time for increasing efficiency.

References

[1] R. V. Rao and K. Selvamani, "Data security challenges and its solutions in cloud computing," *Procedia Computer Science*, vol. 48, pp. 204–209, 2015.

[2] Gatefy, "7 key findings from the 2020 cost of a data breach report," Mar 2021.

[3] A. M. Qadir and N. Varol, "A review paper on cryptography," in *2019 7th international symposium on digital forensics and security (ISDFS)*, pp. 1–6, IEEE, 2019.

[4] A. Joseph and V. Sundaram, "Cryptography and steganography–a survey," 2011.

[5] U. SenthilKumar and U. Senthilkumaran, "Review of asymmetric key cryptography in wireless sensor networks," *International Journal of Engineering and Technology*, vol. 8, no. 2, pp. 859–862, 2016.

[6] R. Chandramouli, G. Li, and N. D. Memon, "Adaptive steganography," in *Security and Watermarking of Multimedia Contents IV*, vol. 4675, pp. 69–78, International Society for Optics and Photonics, 2002.

[7] H. Singh, "Analysis of different types of steganography," *International Journal of Scientific Research in Science, Engineering and Technology*, vol. 2, no. 3, pp. 578–582, 2016.

[8] D. Zhu, "Hiding information in big data based on deep learning," *arXiv preprint arXiv:1912.13156*, 2019.

[9] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2545–2557, 2017.

[10] J. Hayes and G. Danezis, "Generating steganographic images via adversarial training," *arXiv preprint arXiv:1703.00371*, 2017.

[11] K. A. Zhang, A. Cuesta-Infante, L. Xu, and K. Veeramachaneni, "Steganogan: High capacity image steganography with gans," *arXiv preprint arXiv:1901.03892*, 2019.

[12] D. Hu, L. Wang, W. Jiang, S. Zheng, and B. Li, "A novel image steganography method via deep convolutional generative adversarial networks," *IEEE Access*, vol. 6, pp. 38303–38314, 2018.

[13] E. Wengrowski and K. Dana, "Light field messaging with deep photographic steganography," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1515–1524, 2019.

[14] H. Kang, H. Wu, and X. Zhang, "Generative text steganography based on lstm network and attention mechanism with keywords," *Electronic Imaging*, vol. 2020, no. 4, pp. 291–1, 2020.

[15] M. Khari, A. K. Garg, A. H. Gandomi, R. Gupta, R. Patan, and B. Balusamy, "Securing data in internet of things (iot) using cryptography and steganography techniques," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 1, pp. 73–80, 2019.

[16] W. Luo, F. Huang, and J. Huang, "Edge adaptive image steganography based on lsb matching revisited," *IEEE Transactions on information forensics and security*, vol. 5, no. 2, pp. 201–214, 2010.

[17] H. Huanhuan, Z. Xin, Z. Weiming, and Y. Nenghai, "Adaptive text steganography by exploring statistical and linguistical distortion," in *2017 IEEE Second International Conference on Data Science in Cyberspace (DSC)*, pp. 145–150, 2017.

[18] L. Y. Por, K. Wong, and K. O. Chee, "Unispach: A text-based data hiding method using unicode space characters," *Journal of Systems and Software*, vol. 85, no. 5, pp. 1075–1082, 2012.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[20] H. Ruiz, M. Chaumont, M. Yedroudj, A. O. Amara, F. Comby, and G. Subsol, "Analysis of the scalability of a deep-learning network for steganography "into the wild"," in *International Conference on Pattern Recognition*, pp. 439–452, Springer, 2021.

[21] F. Li, H. Tang, Y. Zou, Y. Huang, Y. Feng, and L. Peng, "Research on information security in text emotional steganography based on machine learning," *Enterprise Information Systems*, vol. 15, no. 7, pp. 984–1001, 2021.

[22] Y. Tong, Y. Liu, J. Wang, and G. Xin, "Text steganography on rnn-generated lyrics," *Mathematical Biosciences and Engineering*, vol. 16, no. 5, pp. 5451–5463, 2019.

[23] Y. Li, J. Zhang, Z. Yang, and R. Zhang, "Topic-aware neural linguistic steganography based on knowledge graphs," *ACM/IMS Transactions on Data Science*, vol. 2, no. 2, pp. 1–13, 2021.

[24] Z.-L. Yang, X.-Q. Guo, Z.-M. Chen, Y.-F. Huang, and Y.-J. Zhang, "Rnn-stega: Linguistic steganography based on recurrent neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1280–1295, 2018.

[25] Z. Yang, B. Gong, Y. Li, J. Yang, Z. Hu, and Y. Huang, "Graph-stega: Semantic controllable steganographic text generation guided by knowledge graph," *arXiv preprint arXiv:2006.08339*, 2020.

[26] N. Wu, P. Shang, J. Fan, Z. Yang, W. Ma, and Z. Liu, "Research on coverless text steganography based on single bit rules," in *Journal of Physics: Conference Series*, vol. 1237, p. 022077, IOP Publishing, 2019.

[27] R. Stutsman, C. Grothoff, M. Atallah, and K. Grothoff, "Lost in just the translation," in *Proceedings of the 2006 ACM symposium on Applied computing*, pp. 338–345, 2006.

[28] X. Chen, H. Sun, Y. Tobe, Z. Zhou, and X. Sun, "Coverless information hiding method based on the chinese mathematical expression," in *International conference on cloud computing and security*, pp. 133–143, Springer, 2015.