Fall 2012

Stealthy Plaintext

Naidele Katrumane Manjunath
San Jose State University

Stealthy Plaintext

A Project Report

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Naidele Katrumane Manjunath

December 2012

SAN JOSE STATE UNIVERSITY

The Undersigned Project Committee Approves the Project Titled

“Stealthy Plaintext”

by

Naidele Katrumane Manjunath

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

December 2012

Dr. Soon Tee Teoh, Department of Computer Science

Date

Dr. Mark Stamp, Department of Computer Science

Date

Hariharan Sudagar, Broadcom

Date

APPROVED FOR THE UNIVERSITY

Associate Dean Office of Graduate Studies and Research

Date

Table of Contents

List of Figures

ABSTRACT

Stealthy Plaintext

By Naidele Katrumane Manjunath

Correspondence through email has become a very significant way of communication at workplaces. Information of most kinds such as text, video and audio can be shared through email, the most common being text. With confidential data being easily sharable through this method most companies monitor the emails, thus invading the privacy of employees. To avoid secret information from being disclosed it can be encrypted. Encryption hides the data effectively but this makes the data look important and hence prone to attacks to decrypt the information. It also makes it obvious that there is secret information being transferred. The most effective way would be to make the information seem harmless by concealing the information in the email but not encrypting it. We would like the information to pass through the analyzer without being detected. This project aims to achieve this by "encrypting" plain text by replacing suspicious keywords with non-suspicious English words, trying to keep the grammatical syntax of the sentences intact.

ACKNOWLEDGEMENTS

This project is made possible through the help and support from my parents, professors, family, and friends. Especially, I would like to dedicate my acknowledgment of gratitude towards the following significant advisors and contributors.

First and foremost, I would like to thank my advisor Dr. Soon Tee Teoh, for his support, encouragement and technical advice that made this project possible.

Secondly, I would like to thank my committee member Dr. Mark Stamp, for giving me the idea for this project, providing valuable advices and inputs to improve my project.

I would also like to show my appreciation to Hariharan Sudagar, for the help and inspiration he extended.

Finally, I sincerely thank my mother, family, and friends, who provided me advice and unconditional support. This would not be possible without all of them.

1. Introduction

Employees in most large companies have access to the internet. This brings about a lot of advantages for a company in terms of increasing the scope of communication with tools such as email, instant messaging and video conferencing.

One of the most common uses of internet at work is for communication through email. Email access is provided as a method for internal and external communication at work. It saves money for the company by making less use of paper and labor for delivery of mails and providing faster communication, hence increasing effective usage of time. However employees with access to email tend to use it for personal purposes also. At times employee's emails may be monitored. This raises concerns of privacy. Although the Federal Electronic Communications Privacy Act provides some privacy to employee electronic information at work, it fails to protect all data. Emails may be monitored for reasons such as security, productivity measures, Performance Review, legal liabilities and compliance.

Most people today would agree with the fact that no privacy can expected at their workplace. Every conversation from phone calls to emails, every letter typed on the keyboard, every location the employee visits with the company badge on can be and are mostly monitored. All this is mostly considered to be legal. Most companies have their employees sign a legal document giving their consent for these details to be monitored by their employers. A 2007 American Management Association survey

found that two-thirds of employers monitor employee's Internet use [1]. Employers use different techniques to monitor employees such as video surveillance, internet surveillance, desktop surveillance etc.

Employees mostly get into trouble due to the conversations they have. Email scanning is the method by which emails are passed through filtering software or email monitors which are used to analyze the content of the email. It is unusual for humans to actually go through e-mails manually. Surveillance of communication between employees and external communication being a very common practice at most companies employees tend to be cautious.

Email scanners are used to detect suspicious mails being sent across. Emails both inbound and outbound are monitored. Why would emails be monitored at work?

- It would ensure increased security for the company.
- Increase in employee productivity.

In a lot of cases, employees wouldn't want their emails being monitored. They may like to have discussions about new job prospects or leakage of confidential information at work to go unnoticed.

One of the methods that are used to maintain privacy is to use encryption. On being encrypted it becomes obvious that there is important information being shared. Stealthy Ciphertext [2] is a SJSU project which tries to hide the fact that data has been encrypted. It converts the cipher text into ordinary looking plain text. The grammar may not be right but this may pass through a test for encryption when passed through a trained analyzer.

Our project will deal with developing a method to get emails across the email monitors without being detected. We want the fraudulent or stealthy email to be accepted as a genuine message by the analyzer.

These emails will not be encrypted; instead, they will be converted to another form of plain text. It will be an application of the idea presented in Stealthy Ciphertext. This project will optimize their approach by giving a better entropy value for the converted email. This can be considered as a form of steganography.

The first and the simplest approach to do this would be to replace keywords with less important looking data. Randomly replacing words can cause the analyzers to raise an alarm since this will increase the entropy of the data and make it look like it is encrypted. Encrypted data attracts unwanted attention. We try to make the data look as English like as possible. The "Englishness" of the converted text will be checked using a Hidden Markow Model. The next step in improving the "stealthy email" is to make it confirm to English grammar syntax. Although this may not be achieved to the fullest, that is humans will be able to tell the difference, email analyzers will not be able to detect the difference. Thus we aim to make communication at workplace worry free.

2. Background

The following section defines 4 important aspects which are used and implemented in this project:

1. Encryption.
2. Encryption Detection.
3. Steganography.
4. Entropy.
5. HMM.
6. Grammatical model of English text

2.1 Encryption

Encryption is the process of encoding information in such a way that only the intended parties can read or understand the information and no unauthorized parties such as eavesdroppers or hackers can read it. To encrypt a message in readable format, also known as plaintext, an encryption algorithm is used. This algorithm is used to convert plaintext into encrypted format called cipher text. This encryption is done usually with the help of an encryption key from the sending party. Any third party seeing the encrypted text should not be able to determine the contents of the encrypted information. On the receiving party side, the cipher text needs to be decoded into its original form. This is done with the use of secret decryption key.

There are two basic encryption schemes:

1. **Private-key or Symmetric encryption:** In this encryption scheme, the encryption and decryption keys are the same. Both the sending and the receiving parties need to agree on a common key before they communicate with each other.

2. **Public-key or Asymmetric encryption:** In this encryption scheme, there are two keys: the public key, which is used for encrypting the data and a private key, which is used for decrypting the data. As the names suggest, any one or everyone has access to the public key, therefore, anyone can encrypt the information, but only the receiving party has access to the decryption or private key [5].

With the use of transmission of large amounts of data via the internet, where information travels through a lot of connecting nodes before it reaches from the sending party to the receiving party, encryption plays a very important role to protect data in transit.

2.2 Encryption Detection

Encrypted data may be very effective in hiding valuable or secretive information but is equally vulnerable to being detected. Hence it gets unwanted attention, thus leading to attacks. Encryption of information leads to increase in randomness/entropy of the content.

In a research by Eric B Cole for his patent, "Methodology, System and Computer Readable Medium for Detecting File Encryption", he tried to illustrate this randomization property of encrypted text. Figure 1 shows the histogram of a regular text file, where the x axis shows the

ASCII values of the characters present in the text file, whereas the y axis represents the frequency of the characters. If you take a look at the figure 1, you will observe a huge spike at 32, which is due to the constant use of spacebar keystroke in the text file, illustrating common behavior. The second figure shows the histogram of the same text file as input and encrypted with PGP encryption. The histogram for the PGP encrypted text file looks flatter as compared to the regular text file, it is because encryption of text has increased the randomness in the file, and not many encrypted characters are repeated, thereby uniformly distributing the frequency of characters. This increase in randomness, which leads to uniformity in the frequency distribution of byte values in a file, can help in detecting if a file has been encrypted or is in its regular form [7].

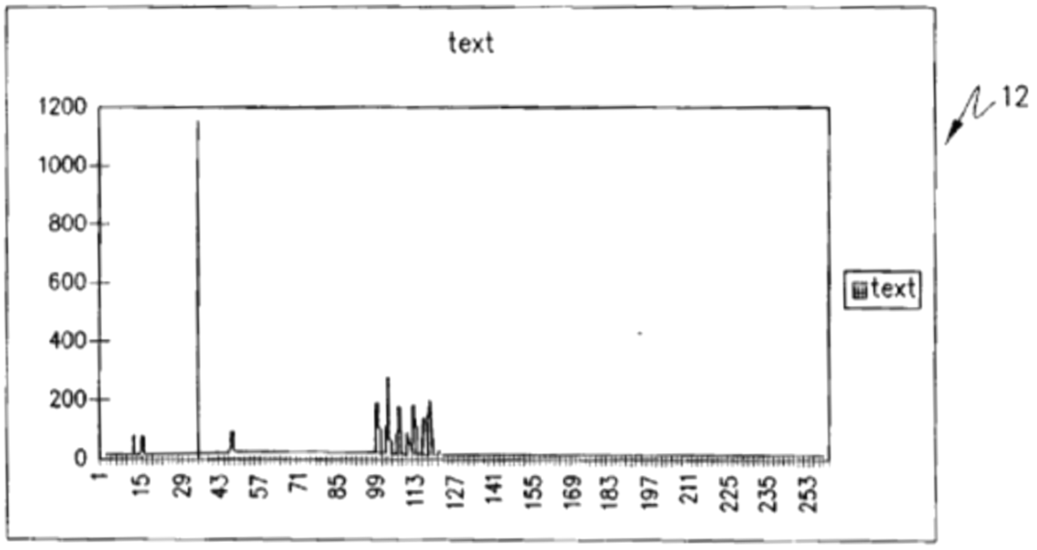


Figure 1 : Histogram of regular text file; *Referenced from Cole [7]*

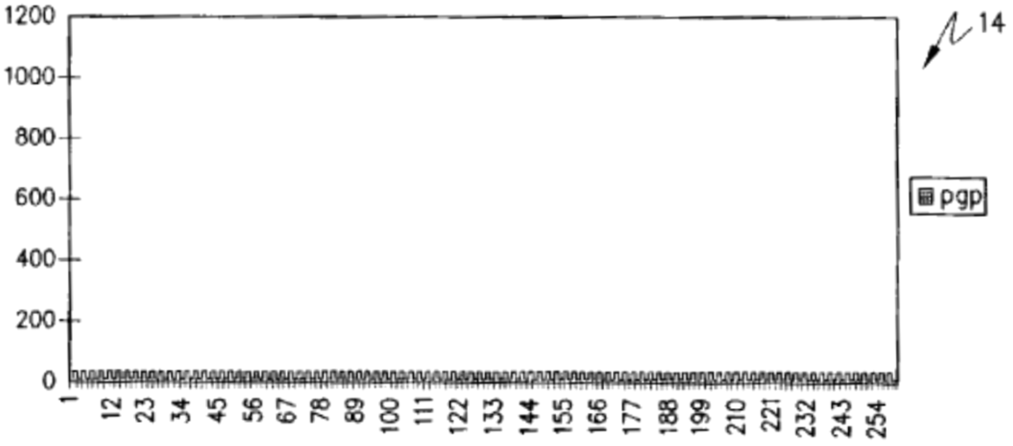


Figure 2 : Histogram of encrypted text file; Referenced from Cole [7]

Another approach to detection of encryption by examining packet headers was proposed by Bruce Schneier in his book Applied Cryptography, 2nd Ed. Most strong encryption algorithms compress the data before performing encryption. Compression before encryption is good because it reduces the redundant information in data and also reduces the time needed for encryption. Redundant data makes the job of cryptanalysts easy hence its removal increases the efficiency of the encryption algorithm.

According to Schneier's method if a file is encrypted using a good encryption algorithm it cannot be compressed much further. Such a file can be considered to be encrypted [5].

2.3 Steganography

Steganography is a method in which important information is concealed in seemingly innocent data. The art of invisible communication is called steganography. It provides security by using techniques to conceal the communication itself from the observer also known as "information hiding". Use of steganography can be seen from historical times and dates back to 480 BC where Demaratus, a Greek sent a warning about a pending attack to the Spartans by writing on the wooden surface of a wax tablet and then applying beeswax on the surface. Since the tablets looked blank to the guards the secret information could be communicated without being detected. In the early 20th century, some of the tricks used were invisible inks, difference in size of the text and microdots during the world wars by the Germans.

Internet is the largest source of information where information is available mostly at no cost. It is also one of the most effective and easy means of communication and can used be for sending and receiving large sizes of information. The presence of the large amount of multimedia data on the internet makes it a good avenue to use steganography. Internet steganography is the exploitation of Internet elements and protocols for the purpose of covertly communicating supplementary data [**8**]. The presence of text, images, videos and audio on the internet means different kinds of steganography over the internet. Below are a few types of steganography:

2.3.1 Hiding a message inside text:

There are a lot of proposed algorithms to hide a message inside a plain text. Although it is effective, the down side is that plain text can be read by anyone. Most of these algorithms have a pattern, which could be found out easily by reading through the plain text. Once the pattern is found out, the secret message inside the text can be recovered, making this technique non-robust.

There are further modifications to hiding a secret message in plaintext that try and make it hard to decode, such as using every nth character, and altering the number of whitespaces between lines and words. Another way was to generate a secret key from a public source of information, such as books or newspapers. You could generate the secret key keeping in mind a combination of page numbers, paragraphs, line number and character number. To decode information hidden in this form, a person should be able to get access to both the secret key and the source. The downside of this is that the secret key and source information has to be sent from the sender to the receiver in a secure manner. This leads to the conclusion that although easy to encode and decode, hiding a secret message in plaintext is not secure.

Our project can be considered to be performing text steganography since we are hiding the original email in the converted email.

2.3.2 Image Steganography:

Since the advent of internet and digital technology, the method of hiding secret messages has shifted its paradigm from text methods to other forms, such as messages hidden in bit form in things like audio and images. Though the practical use of image stenography was found to be limited by a research by German steganography expert Niels Provos, who created a scanning cluster to scan through more than a million pages in various newsgroups on the internet and couldn’t find any hidden messages in them, the increase in concerns for privacy and anonymity are showing an increasing trend in the use on Image Steganography to communicate messages secretly between two parties. Even though there is concern for the use of image steganography for malicious purposes, such as hackers to spread Trojans and viruses, and terrorists to exchange information, it can be used for constructive purposes too, such as digital watermarking for copyright purposes. One of the other main uses for Image Steganography is for the transportation of high-level or top-secret documents.

To hide a message in an image, you need to modify the bits in the image, and you need to do so in such a way that it should not alter the visible properties of the image. The best way to do that is to modify the image in its “noisy” spectrum, where the color variations are more, there-by reducing chances of detection. The common methods of hiding data in images are to modify the Least Significant Bit (LSB) of a byte, masking, filtering etc. The type of image file being used also contributes to the effectiveness of data hiding. Image steganography is gaining popularity nowadays, because graphics standards are improving, thereby leading to have more gradations of color than what the human eye can notice, and data at the receiving end can be stripped out. A 1024 * 1024 grey scale picture can store a 64 KB message. [14]

2.3.3 Implementation

Image steganography can be done by one of the 3 methods listed below:

1. **LSB Substitution:** LSB stands for Least Significant Bit, the title itself is self-explanatory as to what the algorithm does. Information is embedded in the least significant bit of the cover or source image. On an average, this technique leads to the modification of 50% of all the LSB’s in the image, the image is not distorted to the human eye.

Example: Let us say this is the image represented in bits:

(00101101 00011100 11011100)

(10100110 11000100 00001100)

(11010010 10101101 01100011)

The information we want to send in bit format is 11001000, after LSB substitution, the image in bitwise format will be:

(0010110**1** 0001110**1** 1101110**0**)

(1010011**0** 1100010**1** 0000110**0**)

(1101001**0** 1010110**0** 01100011)

2. **Blocking:** This works by breaking an image into blocks and using DCT's (Discrete Cosine Transforms).

3. **Palette Modification**: Palette modification works by taking advantage of the fact that number of colors in an image is limited. For example: If we take a GIF image, a very popular format used in the internet images, it has a depth of 8, that means it cannot have more than 2^8 =256 colors. The colors are stored in a color lookup table or palette. A single byte is used to represent a pixel. This data is used to index the color palette [9].

2.4 Entropy

Entropy is used to measure the randomness of data. The entropy of English is very low, that is, it is less random, and hence, fairly predictable. Application of common knowledge of English is required to make a fairly reliable assumption that in sentences , there will be more a's and e's than y's and z's, certain letter such as "the" are used more often in combination than any other words. Text in English has entropy of one bit for each byte (eight bits) of message. According to Shannon's experiments, it was concluded that the entropy rate of English text is between 1.0 and 1.5 bits per letter, or as low as 0.6 to 1.3 bits per letter. Entropy should not be too low or too high, there is an optimal entropy range. [13]

Adi Shamir and Van Someren, in the paper "Hide and Seek With Stored Keys" try to find out methods and techniques to efficiently find hidden cryptographic keys within large amounts of data like file systems, such as algebraic attacks to find out RSA keys in strings and statistical attacks to find arbitrarily hidden keys in data.

The basis of all applications to find encrypted information hidden in large amounts of data is that entropy or randomness of encrypted data is comparatively more that non-encrypted data. Therefore, one way to find keys in data would be to divide the data into small sections, calculate the entropy of that section and plot it in a graph. If the section consists of hidden key data, it can be detected easily in the graph. We need not get a true measure of entropy of complete data to differentiate between key and non-key data. Shamir and Someren carried out an experiment with a 64 byte sliding window and found the number of unique byte values within it. To give precise values, the first code which they analyzed in such fashion yielded just 30 unique values on an average, with a deviation of 10. In comparison, with sections having key data, the average was 60 unique values, which made things quite evident. Final results for their experiment on 300 KB of data yielded only 23 windows (64 bytes in each window) having a value of 50 or more, in which 20 of these were consecutive and comprised of cryptographic key data.

2.5 HMM

A Markov process or model is a stochastic model that is based on the Markov property. Markov property is a property that can be seen in a set of stochastic processes. The process is memory less that is the present state can predict the future states as well as the past states. A Markov chain is a process that consists of a finite number of states and some known probabilities p_{ij}, where p_{ij} is the probability of moving from state j to state i. That is the past affects the future through the present. These hidden states form the Markov chain.

A hidden Markov model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states [15]. Only the output of the HMM is visible, the state or states which the process goes through before reaching the output is not visible to the observer.

HMM has been extensively used for several decades and has various applications such as speech recognition, handwriting recognition by automatically recognizing repeated strokes and molecular biology for gene finding. Studies have shown HMMs to be effective in solving these kinds of problems.

A hidden Markov model comprises of five-tuples namely X, O, A, B and pi. The value of X and O are fixed. $\lambda = \{A, B, \pi\}$ will be the parameters for a given HMM.

Example of HMM:

Let us consider two people, A and B. A is the employer and B is the employee. A is on vacation and is out of the city. At the end of the day, B has to report back to A on what he did, his job is to do 3 chores: mow the lawn, put heating on and clean the pool. The choice of what is done on a given day is decided by the temperature that day. When employee B reports back to A at the end of the day, A tries to guess weather that day was hot or cold.

A considers the temperature as a discrete Markov Chain. There are 2 states which as hidden, namely “Hot” and “Cold”. B does one of three chores listed out above in a single day based on the temperature that day, which is “mow”, “put heating on” and “clean pool”. These three things are the observations, as B reports back to A with what he does at the end of the day and A learns about it from B. Let’s represent the chores as M, H and C respectively.

A knows about the general temperature trends, and what does B do on an average each day. These things would form the parameters for the Hidden Markov Model (HMM).

1) states = ('Cold', 'Hot')

2) observations = ('mow lawn', 'clean pool ', 'put heating on')

3) start_probability = {'Cold': 0.6, 'Hot': 0.4}

4) transition_probability = {
'Cold' : {'Cold': 0.7, 'Hot': 0.3},
'Hot : {'Cold': 0.4, 'Hot': 0.6} }

5) emission_probability = {
'Cold' : {'mow lawn': 0.1, 'clean pool': 0.4, 'turn heating on ': 0.5},
'Hot' : {'mow lawn': 0.6, 'clean pool': 0.3, 'turn heating on': 0.1}
}

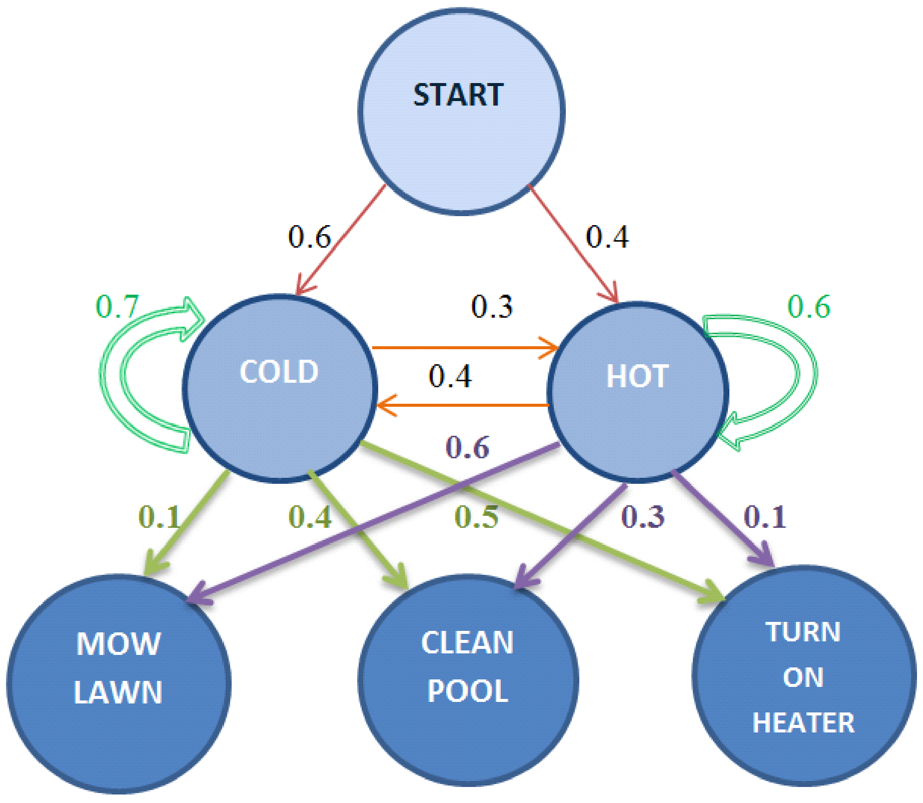


Figure 3 : Example of HMM

We can understand the following from the above example:

1. The start_probability is the probability of the temperature being hot or cold on an average. This is the state of the HMM initially. We can arrive at the initial state matrix from (3)

$$\pi = \begin{pmatrix} 0.6 & 0.4 \end{pmatrix}$$

2. The transition_probability is the probability of the change in temperature in the Markov chain. In the above example, the possibility of it being a hot day if the previous day was cold is only 30%. We can arrive at the state transition matrix from (4).

$$A = \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix}$$

3. The emission_probability is the probability of B performing a specific chore based on what the temperature is. In the above example, the possibility of B mowing the lawn if the temperature is cold is just 10%. We can arrive at the observation matrix from (5)

$$B = \begin{pmatrix} 0.1 & 0.4 & 0.5 \\ 0.6 & 0.3 & 0.1 \end{pmatrix}$$

Suppose there are four days when B has reported that he moved the lawn, cleaned the pool, moved the lawn, turned the heater on. That is the observation sequence is {M, C, M, H}. ‘A’ wants to determine what the temperatures were on these four days using the observations made. Here ‘A’ is trying to get the state sequence of the Markow process given the observation sequence.

The following notation is used to represent the observation and the states of HMM:

T = length of the observation sequence (training sequence)

N = number of states in the model (they are the hidden states, we may know this number, if we do not know the number we make a guess)

M = number of observation symbols

X = {X_0, X_1,….., X_{N-1}} = distinct states of the Markov process

O = {0, 1,….., T - 1} = set of possible observations

A = state transition probabilities

B = observation probability matrix

π = initial state distribution

A general HMM is illustrated below:

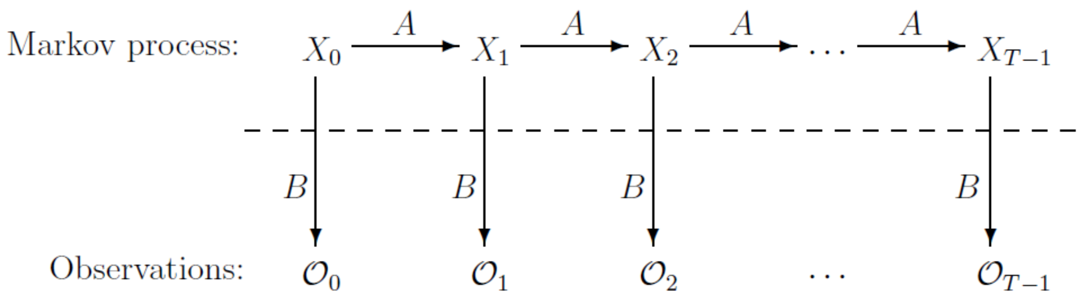


Figure 4 : General representation of HMM; *Referenced from Stamp [11]*

The symbols in the figure are as described in the notations section above. X_i represents the hidden state at time ‘t’ and O_i represents the observation state at time‘t’. The hidden states are above the dashed line and 'A' is the probability of transition from one state $X_{i \text{ to }} X_{j.}$ The Markow process is determined by the current state and the matrix 'A'. [11]

There are three kinds of problems which can be solved efficiently using HMM:

1. **The Evaluation Problem**: For a given observation sequence, *O=O1, O2…OT,* and the complete parameter set of an HMM, λ= *{A, B, π},* what is the probability (*P (O|λ))* that the observation sequence can be generated using the parameter set?
2. **The Decoding Problem**: For a given an observation sequence *O* and the parameter set of HMM λ, what is the optimal state sequence *X=X1X2…XT* that can generate the observation sequence.
3. **The Training Problem**: For an observation sequence *O,* known number of states, and known number of observation symbols what is the optimal model λ which maximizes the probability of observing the given sequence *P(O|λ)*? [16]

We are going to be evaluating the grammar of our converted text using HMM. We use the algorithm of the training problem for training the HMM and then use the algorithm for the evaluation problem to determine if the stealthy plaintext confirms to English language grammar.

2.6 Grammatical model of English text

We try to understand the syntax of English language, i.e., how words are organized in relation to each other.

2.6.1 Grammatical Syntax

Like every language English also uses certain rules when forming a sentence. We need to keep these in mind when doing the substitution. Natural language models assign certain probability to words.

The main assumption we make is that the analyzer will look for randomness in data and certain keywords. We need to understand certain rules in English before going any further.

According to grammar rules we can divide the words into: verbs, nouns, pronouns, adverbs, adjectives, prepositions, conjunctions, and interjections.

If a statistical analyzer looks for syntax then the rules it will follow are very similar to as described in Figure 5. In English sentence structure we can see the five patterns as described below.

1. Subject-Verb = Noun-Verb
   Example: Jill came.

2. Subject-Verb-Object: Noun-Verb-Noun
   Example: Jack climbed the hill

3. Subject-Verb-Adjective = Noun-Verb-Adjective
   Example: The food is delicious

4. Subject-Verb-Adverb = Noun-Verb-Adverb
   Example: Jack runs sometimes.

5. Subject-Verb-Object = Noun/Pronoub-Verb-Noun
   Example: He is running to the car.

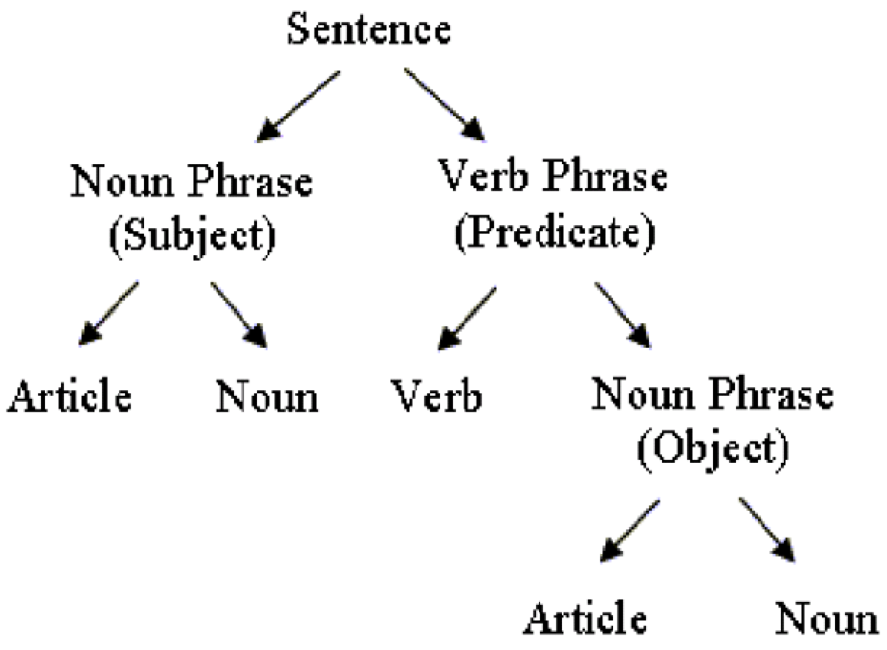


Figure 5 : Components of a grammatically correct sentence

While picking out words to replace keywords we have to make sure that certain syntax rules are being followed. Simova's describes her observations as depicted in the figure below. She saw that if a verb followed a noun then the verb would be followed by an adverb/preposition/period depending on its position in the sentence. We would not need to understand the syntax of the English language much in depth since we use a simple dictionary to spot words and then substitute them with relevant words.

3. Design and System Overview

The goal of this project is to design a method by which we can bypass an analyzer looking for encrypted and "suspicious" information being shared via email. Consider a scenario where Trudy works for Bob's company ABC. Trudy has some confidential information belonging to ABC which she wants to share with Alice only. Trudy composes an email, encrypts it and sends it to Alice. Meanwhile Bob receives an alert from the email analyzer stating that there is some encrypted information being sent through company email. Trudy makes up an excuse which Bob believes and hence Trudy is spared. Trudy now has to come up with an idea to share the information while maintaining its confidentiality as well as a method using which she can send the information while not being undetected by the analyzer. Trudy knows that the company analyzer looks out for keywords which the company wants to maintain confidential. Using this information Trudy comes up with an idea to send the information to Trudy without encrypting it. She also makes sure the confidentiality of the information is maintained. Our project is based on this scenario and implementation of the method which Trudy came up with. We will discuss in the following sections how the method is implemented, assumptions on what is based, its strengths and weaknesses.

System Overview

Phase 1:

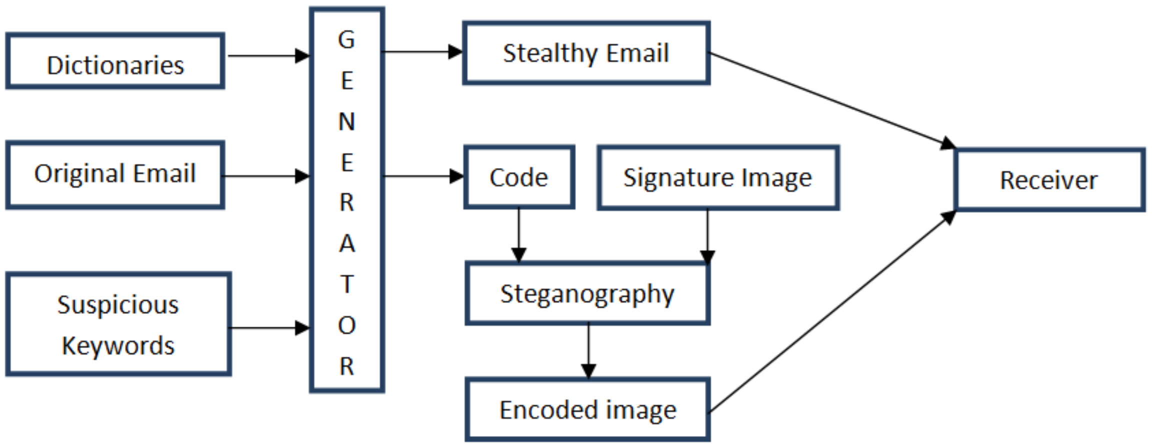


Figure 6 Phase 1- Encryption

Phase 2

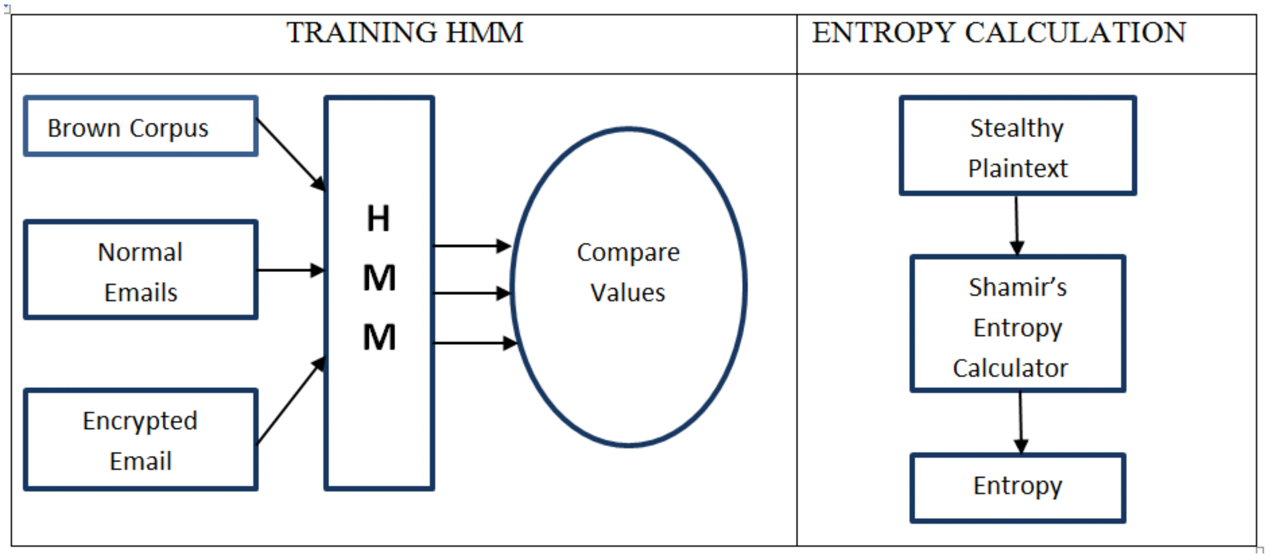


Figure 7 Training and Calculating Entropy

The project has two phases. The main goal of this project is to avoid encryption and make the plain text merge with common data as well as possible. In the first phase of the project we will be developing a generator for generating the stealthy email/plaintext. The generator along with the stealthy email/plaintext also generates a code. This code is embedded into an image using LSB image steganography. Most emails contain the signature of the person sending the email. The signature would usually contain a small image or symbol representing the company or person. We embed the code into this image. The email is then sent. At the receiver's side the code is extracted from the image and it is used to get the original image.

The second phase of the project will be to show that our implementation will be able to bypass the analyzer. The entropy of the stealthy plaintext is calculated using Shamir's entropy. We show that the generated email confirms to the English grammatical model using HMM.

4. Implementation

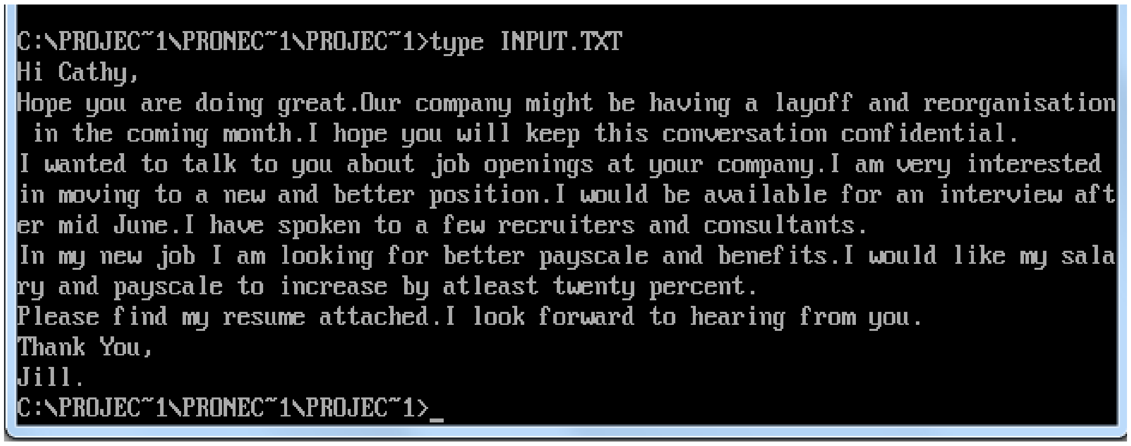
An email containing text can be encrypted and the encrypted information can be sent across to the receiver to provide confidentiality. In this project we are trying to achieve confidentiality when the information is being sent, but we also would like the information exchange to be unnoticeable. For achieving this we would need to bypass the email analyzer.

The approach which is being used makes use of the grammatical model of English text. We prepared different sets of dictionaries for the parts of speech such as nouns, verbs, adjectives etc. There is a dictionary of suspicious keywords which has a sample of words which the analyzer looks for. We work on the assumption that the analyzer will look out for keywords and the entropy of the information being sent. These suspicious keywords file is assumed to be shared between the sender and receiver. These files/dictionaries are read in as arrays by our code. Our method will look out for the suspicious keywords by traversing the email and then replacing them with a word from the dictionary which corresponds to the keywords parts of speech.

Each time a word is chosen for replacing the index of the word in the email is noted and index of the word in the dictionary is also noted. These two numbers form a pair in the code which is sent to the receiver. The below figures illustrate the generation of the code, the stealthy email and decoding. We have used C programming language to implement the stealthy plaintext generator. Below are snapshots of how the generator works.

Encoding

1. Input email: This is plain text. “ input.txt”

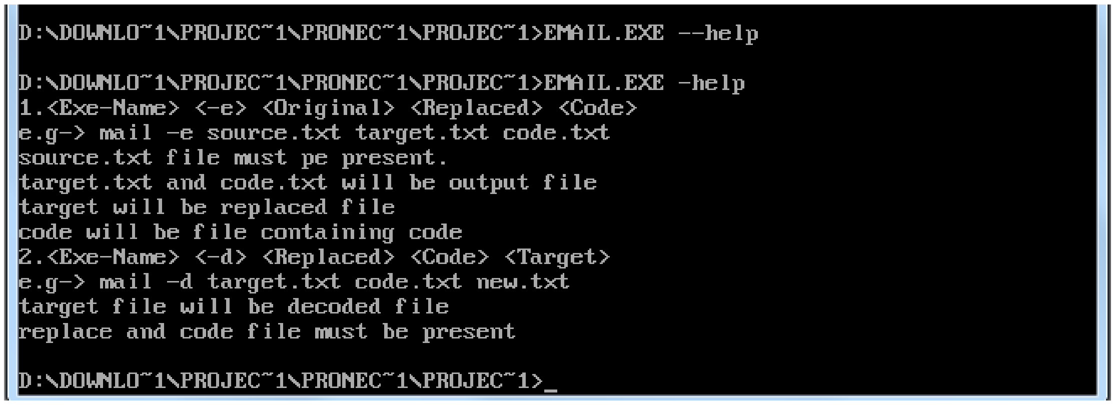


```
C:\PROJEC~1\PRONEC~1\PROJEC~1>type INPUT.TXT
Hi Cathy,
Hope you are doing great.Our company might be having a layoff and reorganisation
 in the coming month.I hope you will keep this conversation confidential.
I wanted to talk to you about job openings at your company.I am very interested
in moving to a new and better position.I would be available for an interview aft
er mid June.I have spoken to a few recruiters and consultants.
In my new job I am looking for better payscale and benefits.I would like my sala
ry and payscale to increase by atleast twenty percent.
Please find my resume attached.I look forward to hearing from you.
Thank You,
Jill.
C:\PROJEC~1\PRONEC~1\PROJEC~1>_
```

Figure 8 : Original email

2. Compile the code to generate exe. In this case the exe name is email.exe.

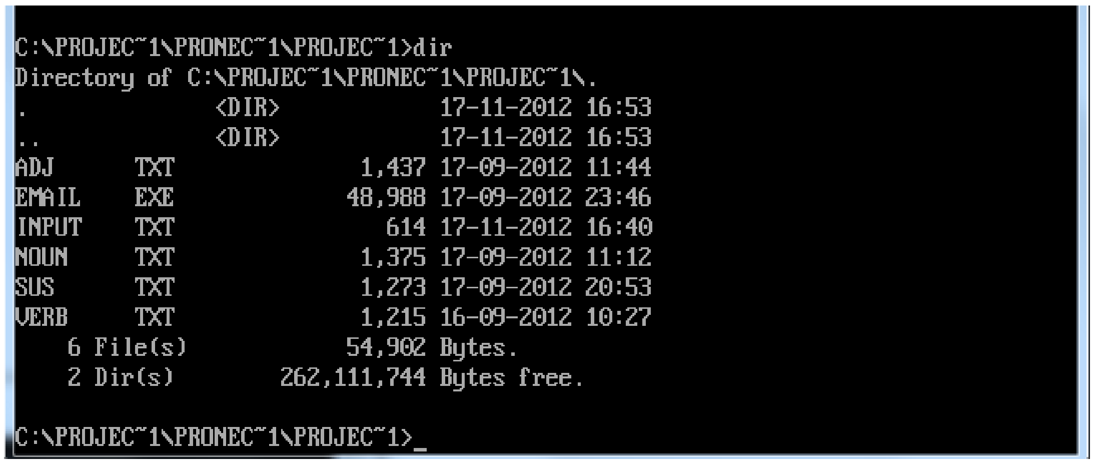
3. Here is a look at the help file for the project. The command is: email.exe –help. You should have the suspicious keyword, verb, noun and adjective list in the same directory level as the exe is.



```
D:\DOWNLO~1\PROJEC~1\PRONEC~1\PROJEC~1>EMAIL.EXE --help

D:\DOWNLO~1\PROJEC~1\PRONEC~1\PROJEC~1>EMAIL.EXE -help
1.<Exe-Name> <-e> <Original> <Replaced> <Code>
e.g-> mail -e source.txt target.txt code.txt
source.txt file must pe present.
target.txt and code.txt will be output file
target will be replaced file
code will be file containing code
2.<Exe-Name> <-d> <Replaced> <Code> <Target>
e.g-> mail -d target.txt code.txt new.txt
target file will be decoded file
replace and code file must be present

D:\DOWNLO~1\PROJEC~1\PRONEC~1\PROJEC~1>_
```

Figure 9 : Email generator help file

4. Here is a snapshot of the files present in the directory before running the exe:

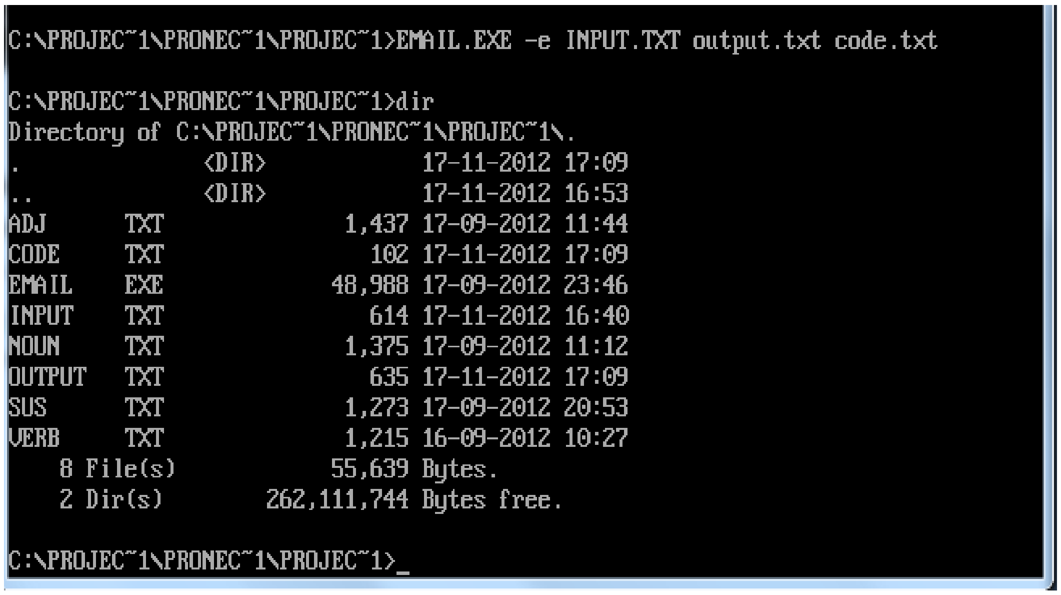


```
C:\PROJEC~1\PRONEC~1\PROJEC~1>dir
Directory of C:\PROJEC~1\PRONEC~1\PROJEC~1\.
.              <DIR>            17-11-2012 16:53
..             <DIR>            17-11-2012 16:53
ADJ      TXT             1,437 17-09-2012 11:44
EMAIL    EXE            48,988 17-09-2012 23:46
INPUT    TXT               614 17-11-2012 16:40
NOUN     TXT             1,375 17-09-2012 11:12
SUS      TXT             1,273 17-09-2012 20:53
VERB     TXT             1,215 16-09-2012 10:27
    6 File(s)           54,902 Bytes.
    2 Dir(s)       262,111,744 Bytes free.

C:\PROJEC~1\PRONEC~1\PROJEC~1>_
```

Figure 10 : Files present in directory before encryption

5. Now, to encode, we will run the exe with the command: email.exe –e input.txt output.txt code.txt, where:
   a) input.txt – Input file.
   b) output.txt- Encoded Output file.
   c) code.txt – code file generated during encoding.

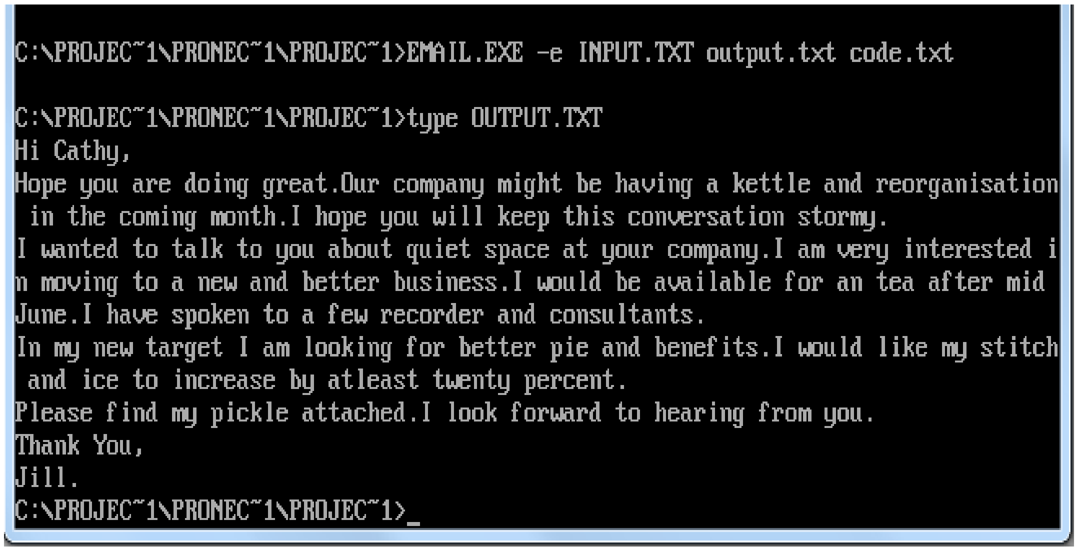
   As you will see in the figure below, there are two additional files generated, which are output.txt and code.txt.



```
C:\PROJEC~1\PRONEC~1\PROJEC~1>EMAIL.EXE -e INPUT.TXT output.txt code.txt

C:\PROJEC~1\PRONEC~1\PROJEC~1>dir
Directory of C:\PROJEC~1\PRONEC~1\PROJEC~1\.
.              <DIR>            17-11-2012 17:09
..             <DIR>            17-11-2012 16:53
ADJ      TXT              1,437 17-09-2012 11:44
CODE     TXT                102 17-11-2012 17:09
EMAIL    EXE             48,988 17-09-2012 23:46
INPUT    TXT                614 17-11-2012 16:40
NOUN     TXT              1,375 17-09-2012 11:12
OUTPUT   TXT                635 17-11-2012 17:09
SUS      TXT              1,273 17-09-2012 20:53
VERB     TXT              1,215 16-09-2012 10:27
    8 File(s)            55,639 Bytes.
    2 Dir(s)        262,111,744 Bytes free.

C:\PROJEC~1\PRONEC~1\PROJEC~1>_
```

Figure 11 : Files generated after encryption

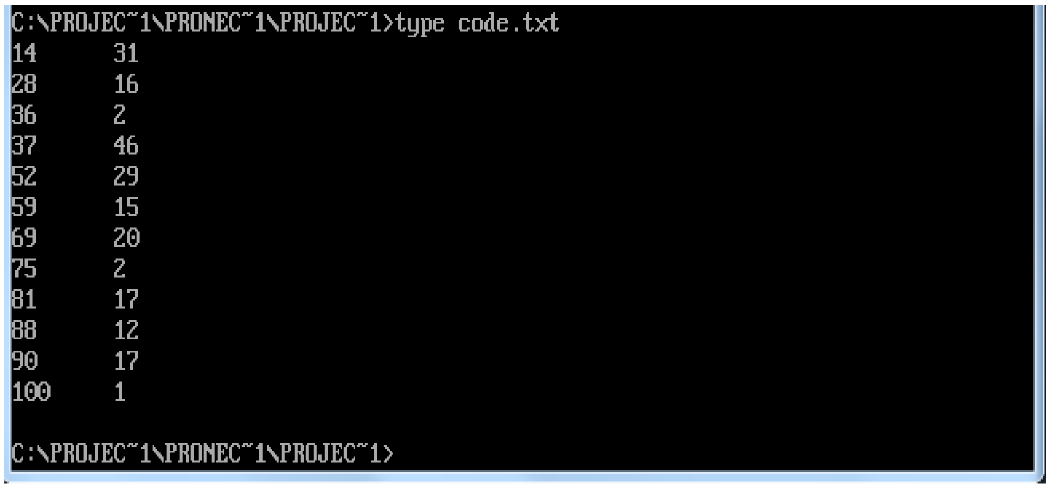
6. Once we run the command, the output file will have the keywords matching any words in the input email replaced by corresponding words with respect to their types, i.e., verb, noun or adjective. As you can see, layoff is a noun in the sus.txt file, replaced by the word kettle, present in the noun.txt file. The contents of the output file are:



```
C:\PROJEC~1\PRONEC~1\PROJEC~1>EMAIL.EXE -e INPUT.TXT output.txt code.txt

C:\PROJEC~1\PRONEC~1\PROJEC~1>type OUTPUT.TXT
Hi Cathy,
Hope you are doing great.Our company might be having a kettle and reorganisation
 in the coming month.I hope you will keep this conversation stormy.
I wanted to talk to you about quiet space at your company.I am very interested i
n moving to a new and better business.I would be available for an tea after mid
June.I have spoken to a few recorder and consultants.
In my new target I am looking for better pie and benefits.I would like my stitch
 and ice to increase by atleast twenty percent.
Please find my pickle attached.I look forward to hearing from you.
Thank You,
Jill.
C:\PROJEC~1\PRONEC~1\PROJEC~1>_
```

Figure 12 : Stealthy encrypted email

7. The code.txt generated is as follows:



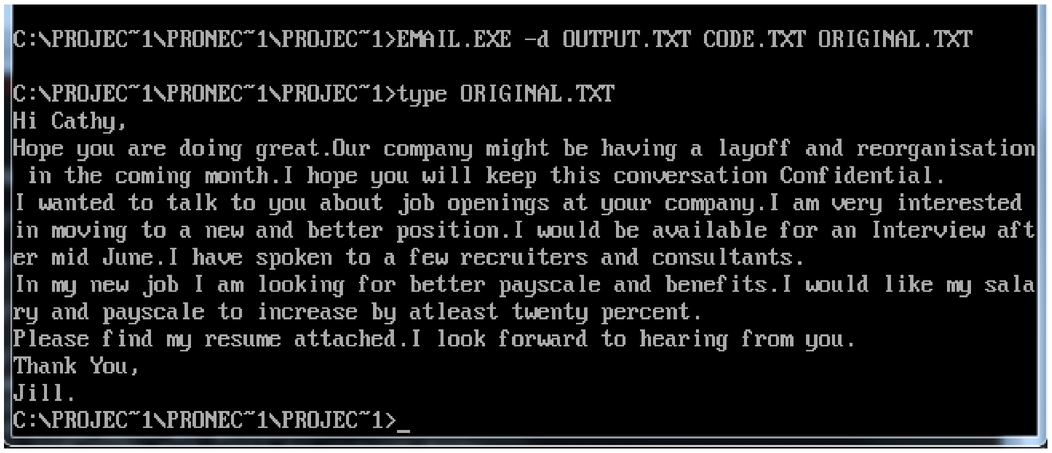
```
C:\PROJEC~1\PRONEC~1\PROJEC~1>type code.txt
14      31
28      16
36      2
37      46
52      29
59      15
69      20
75      2
81      17
88      12
90      17
100     1

C:\PROJEC~1\PRONEC~1\PROJEC~1>
```

Figure 13 : Code text file generated

The first number in each line represents the word replaced from the sus.txt file, i.e., layoff is the 14th word in the sus.txt file replaced by the 31st word, which is kettle, present in the noun.txt file.

Decoding:

1. Decoding is done with the same exe file used for encoding with the –d flag.
2. The decoding process is the reverse of the encoding process, here the input is the encoded output.txt file and code.txt file, and the output is the original email.

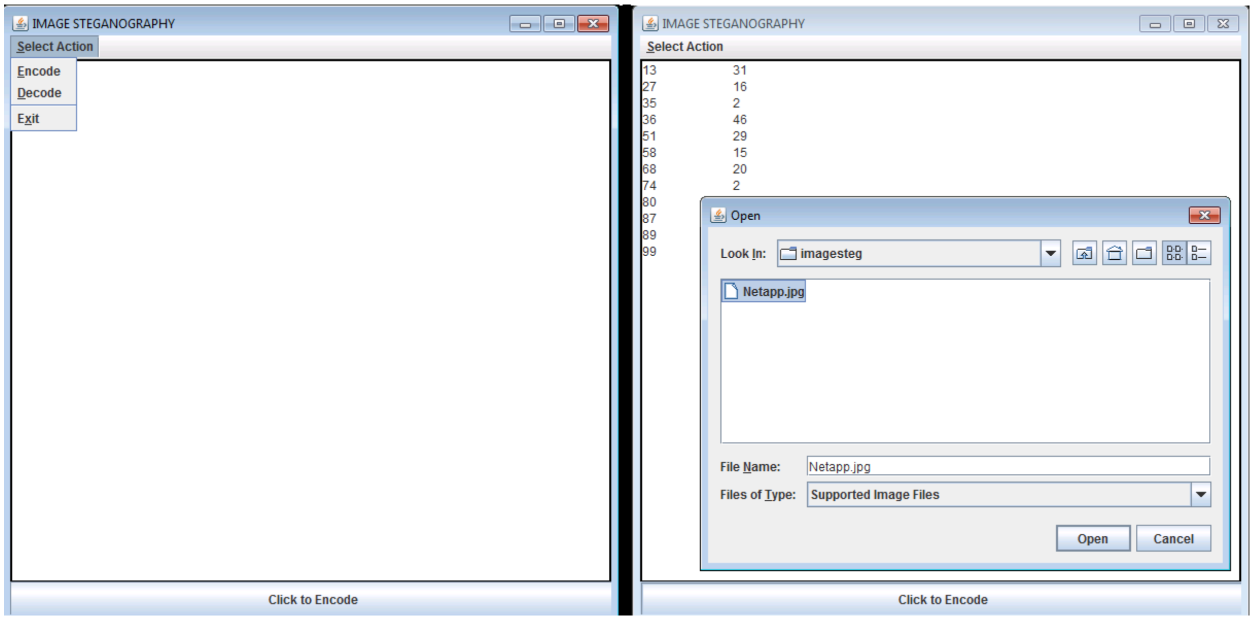


```
C:\PROJEC~1\PRONEC~1\PROJEC~1>EMAIL.EXE -d OUTPUT.TXT CODE.TXT ORIGINAL.TXT

C:\PROJEC~1\PRONEC~1\PROJEC~1>type ORIGINAL.TXT
Hi Cathy,
Hope you are doing great.Our company might be having a layoff and reorganisation
 in the coming month.I hope you will keep this conversation Confidential.
I wanted to talk to you about job openings at your company.I am very interested
in moving to a new and better position.I would be available for an Interview aft
er mid June.I have spoken to a few recruiters and consultants.
In my new job I am looking for better payscale and benefits.I would like my sala
ry and payscale to increase by atleast twenty percent.
Please find my resume attached.I look forward to hearing from you.
Thank You,
Jill.
C:\PROJEC~1\PRONEC~1\PROJEC~1>_
```

Figure 14 : Original email generated from stealthy email

Encoding the image

We have implemented least significant bit image steganography using Java. As illustrated in the figures below, the user has a choice of choosing what information is to be encoded and into which image. The information is read from the file and displayed on the screen. In the below figure we can see the code displayed after being read. From the figure below we can see there is no distortion visible in the image. The file size of the image also does not increase greatly.

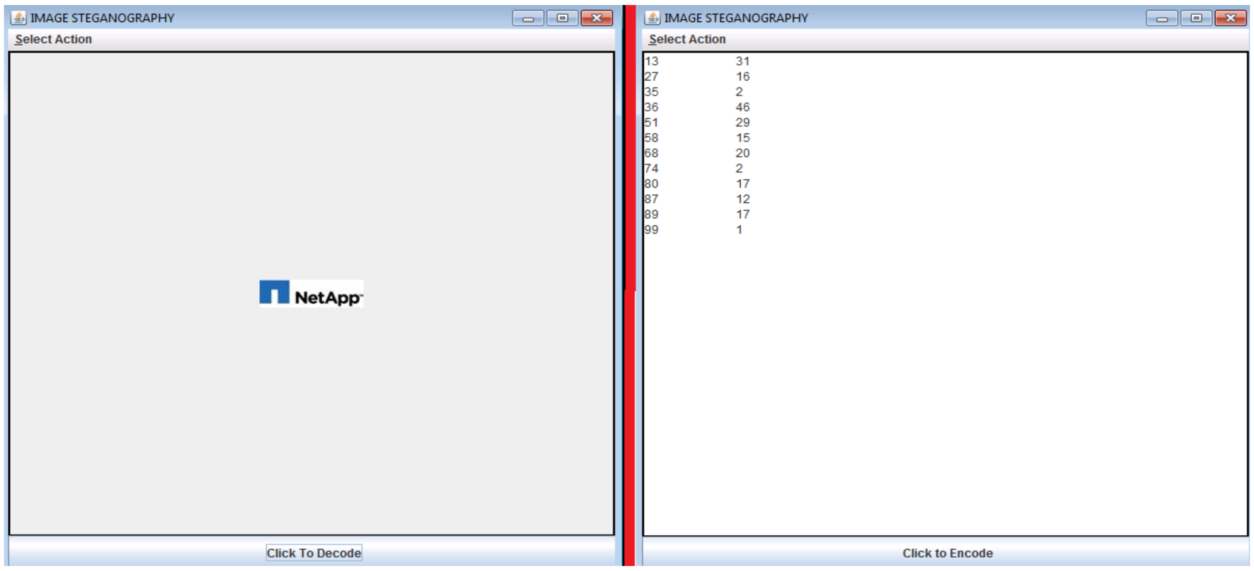


```
13          31
27          16
35          2
36          46
51          29
58          15
68          20
74          2
80
87
89
99
```

File Name: Netapp.jpg

Files of Type: Supported Image Files

Figure 15 : Image Encoding



```
13          31
27          16
35          2
36          46
51          29
58          15
68          20
74          2
80          17
87          12
89          17
99          1
```

Figure 16 : Code being put into image

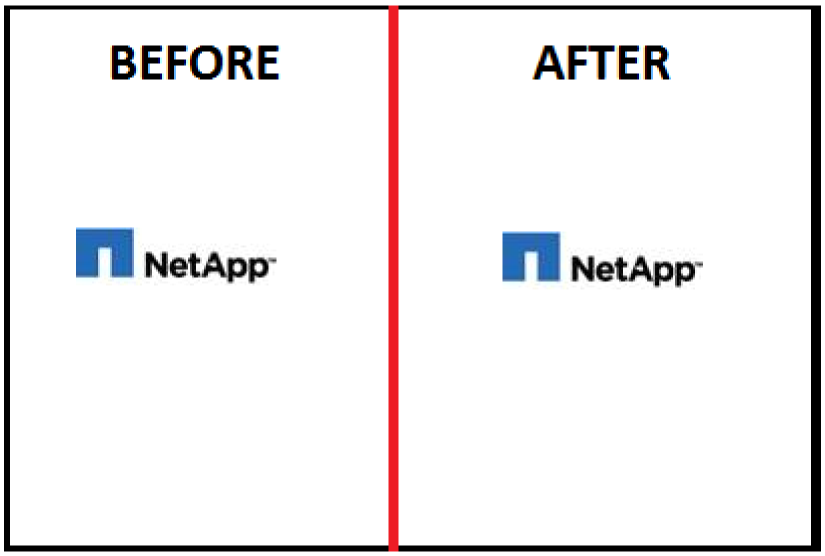


Figure 17 : Comparison of image-before and after encoding

5. Evaluating Stealthy Plaintext

We said earlier that the analyzer would be looking for suspicious keywords. We have taken care of this by making sure that there are no suspicious keywords present at all in the cipher text generated.

Now we show that our method generates cipher text with entropy that matches English language.

5.1 Test for Randomness

Random data can be detected because of its higher entropy. We make sure that the entropy of our cipher text is low. The optimal value for English text from the implementation of Shamir's method gives a value of 26 unique bytes for an average window of 64 bytes [4]. In our experiment we first calculated the entropy of Brown Corpus for which we found a value of 24.7. We then took five files of plain text and found their entropy. They fell in the range of 25 to 28. The same files when encrypted showed a remarkable change in the entropy value. The values ranged in between 41 and 43. This shows that using Shamir's method we can clearly distiguish between encrypted and plain text. We then took five emails and found their entropy and compared that with the entropy of the same emails converted into stealthy plaintext uisng our method. There was a very slight deviation of approximately 0.50.

The entropy for our cipher text gives a very good matching value which lies in the range of 26 to 27. This method has hence proved itself very effective in keeping the entropy same as English text.

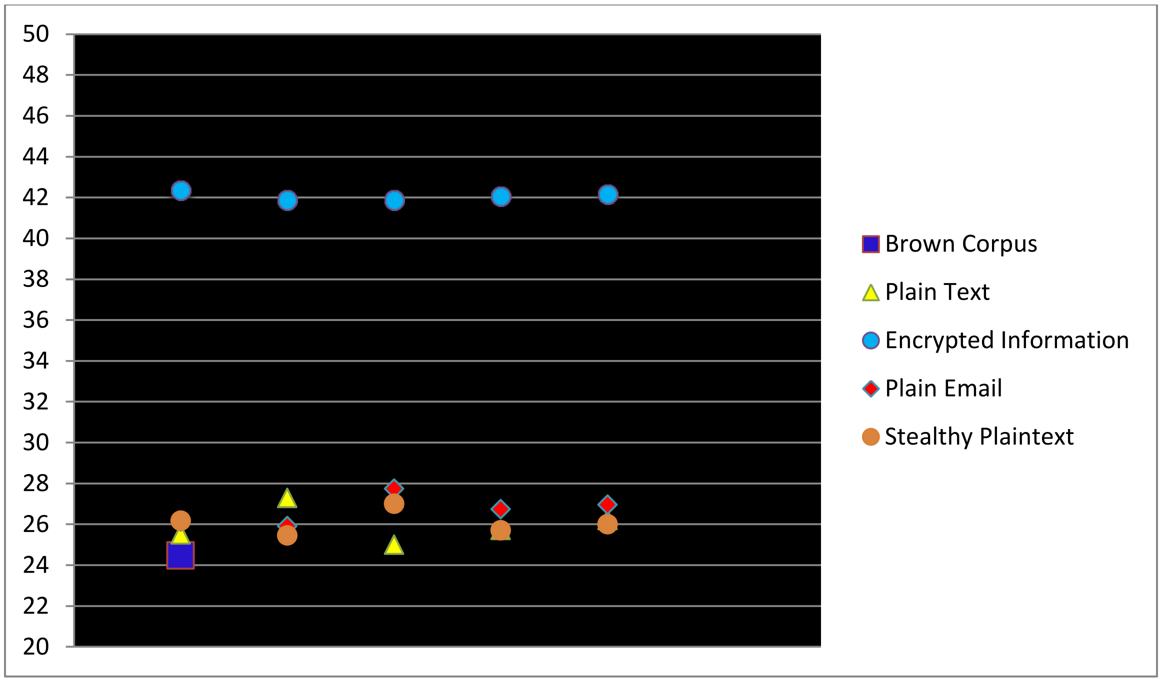


Figure 18 : Test for randomness

5.2 Test for Englishness

We make use of HMM to come up with a table of probabilities for the five sets of patterns or sequences seen in English grammar. The HMM was constructed based on the algorithm described by Stamp in [11]. It was trained with the Brown Corpus, which is well structured English to get a model for our testing. The probabilities of these sequences were then used to score the stealthy plaintext grammar. A high score or probability indicates high similarity between the training data and the test data, while a low score indicates the opposite.

Consider an example where a noun is followed by a verb. The HMM would give values such as there is a 40% chance of a verb appearing after the noun. Similarly the HMM can learn the probabilities of longer sequences. HMM thus trains itself with the probabilities of these sequences and gives us the probabilities of the parts of speech appearing in a particular sequence. We then used these probabilities to get a score for the Brown Corpus.

Specifications:

Training the HMM: We provide the HMM with the brown corpus as input text. The HMM used for training is based on the solution which is used for solving problem (3) (Training problem) described in the HMM section. The solution for the problem (1) (Evaluation problem) is then used to evaluate stealthy plain texts match to English.

Test Data: The observation symbols consisted of nine values namely noun, conjunction, interjection, verb, adjective, preposition, adverb, pronoun, and period. Five hidden states were assumed based on the five patterns seen in English grammar which is described in the Grammatical Syntax section. The words are classified into the parts of speech based on the grammar dictionaries by the HMM. We have created dictionaries for eight of the observation symbols excluding period. The dictionary consists of words varying approximately from 500(conjunction, interjection) to 3000(verb, noun, adverb etc). The observation sequence was of length 10,000 and the number of iterations was 3000.

From the training, HMM produced the following state and transition matrices.

The state transition matrix A:

$$A = \begin{pmatrix} 0.435277, & 0.156975, & 0.118584, & 0.166901, & 0.122217 \\ 0.116642, & 0.063435, & 0.365502, & 0.196055, & 0.258308 \\ 0.051165, & 0.000000, & 0.010732, & 0.863517, & 0.074597 \\ 0.203585, & 0.394978, & 0.032964, & 0.281989, & 0.086434 \\ 0.732479, & 0.117054, & 0.002351, & 0.130420, & 0.017661 \end{pmatrix}$$

The observation matrix from training the HMM model is as below:

$$\begin{pmatrix} 0.153436, & 0.008479, & 0.140808, & 0.830515, & 0.175811, & 0.619238, & 0.020123, & 0.000000, & 0.016244 \\ 0.000000, & 0.038024, & 0.004599, & 0.060040, & 0.121584, & 0.000000, & 0.014217, & 0.106753, & 0.011927 \\ 0.000000, & 0.149909, & 0.001302, & 0.508158, & 0.010095, & 0.001878, & 0.000000, & 0.098006, & 0.011093 \\ 0.011581, & 0.023697, & 0.000879, & 0.040683, & 0.009050, & 0.204300, & 0.000000, & 0.666089, & 0.006830 \\ 0.032616, & 0.561287, & 0.000000, & 0.000000, & 0.028268, & 0.299142, & 0.000000, & 0.006086, & 0.007308 \end{pmatrix}$$

Using the above matrices as input for the Evaluation problem HMM we found the probabilities of different texts confirming to English grammar. The different input files used were a plain email file of ~500 words, four grammatically correct plain text files, a file of 1000 nouns, a file of 1000 verbs, a file 150 interjections and then finally the stealthy plaintext. The scores have been plotted in the graph shown in Figure: 19

For each of the plain text files and the plain text emails the probabilities overlapped with very close values ranging from 98.85 to 98.88.

The files with only nouns, verbs, interjections showed probabilities which lie very far from 98 such as 92.61, 94.35 and 61 respectively.

Thus the probability helps us to clearly distinguish grammatically correct text from those which are not. Our stealthy plaintext email scored 98.88 which clearly shows that the grammar is right.

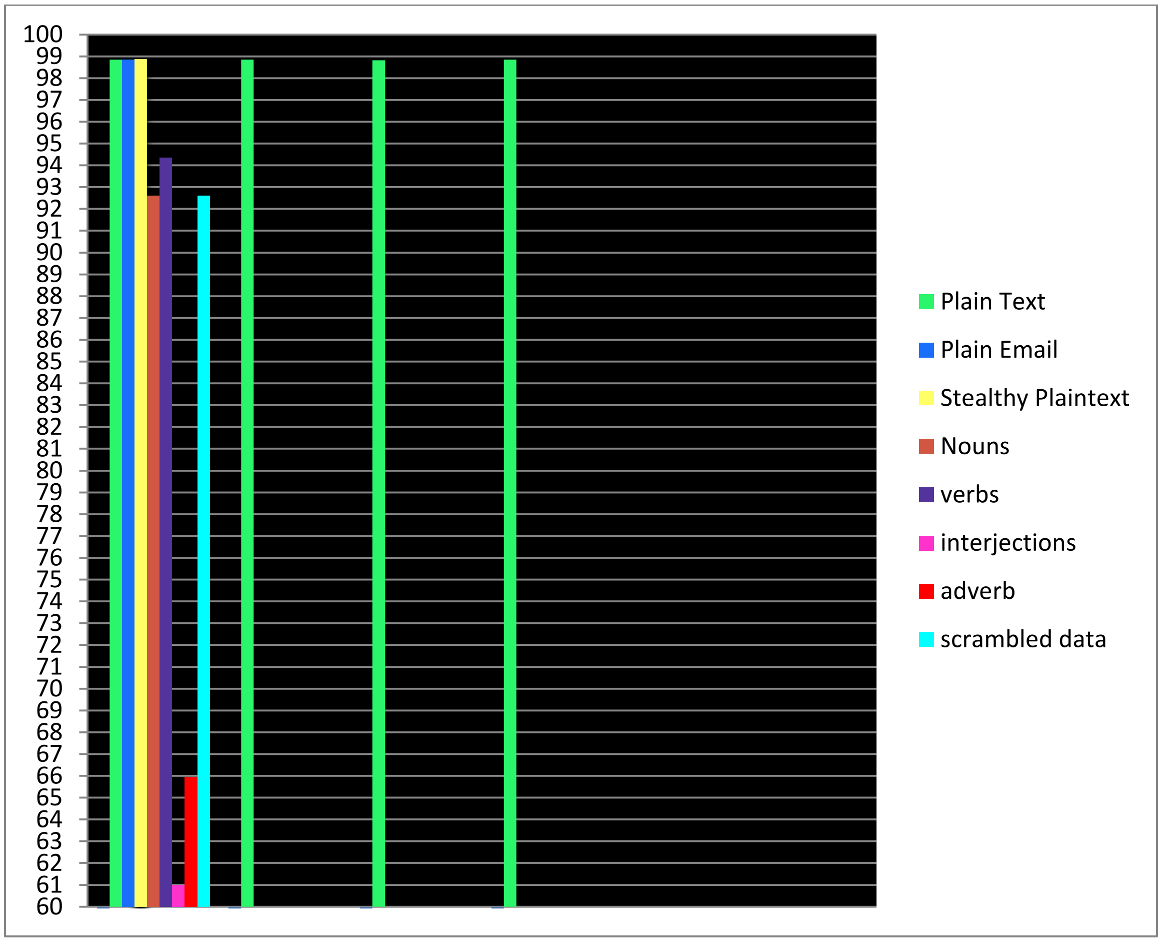


Figure 19: Probability of input text being English with 5 hidden states

Analysis of stealthy plaintext based on the understanding of syntax of English text by Simova:

In the paper "Stealthy Ciphertext" Simova describes the flow of English grammar. This is illustrated in the Figure: 20 [2].

Test Data: The observation symbols consisted of nine values namely noun, conjunction, interjection, verb, adjective, preposition, adverb, pronoun, and period as before. Three hidden states were assumed. The hidden states are:

State 1: noun, pronoun

State 2: verb, preposition, adverb, conjunction, period

State 3: adjective, interjection

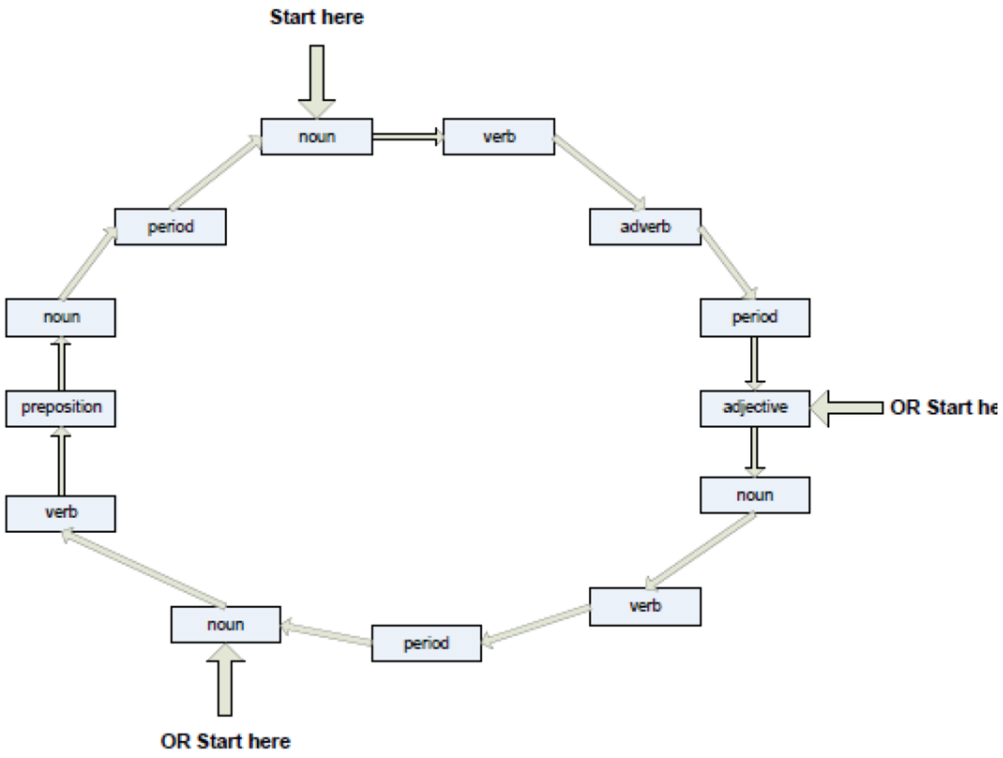


Figure 20: Flow of grammar in an English sentence; *Referenced from Simova [2]*

This HMM was trained with the same set of data and values as in the previous experiment. We thus obtained the following values for the state and transition matrices.

The state transition matrix A:

$$A = \begin{pmatrix} 0.514939, 0.083035, 0.402039 \\ 0.974965, 0.011807, 0.013204 \\ 0.414368, 0.505277, 0.080325 \end{pmatrix}$$

The Observation Matrix:

$$\begin{pmatrix} 0.527730, & 0.089955, & 0.000000, & 0.232258, & 0.028804, & 0.232487, & 0.081825, & 0.029259, & 0.005183 \\ 0.019509, & 0.055235, & 0.088286, & 0.000341, & 0.000000, & 0.377598, & 0.050055, & 0.011777, & 0.036441 \\ 0.073469, & 0.408696, & 0.009963, & 0.000000, & 0.376279, & 0.021471, & 0.014839, & 0.000000, & 0.228559 \end{pmatrix}$$

These matrices were then used for initializing the matrices in the evaluation problem. The input files used for testing were the same as the ones used in the HMM model with 5 hidden states. The scores have been plotted in the graph shown in Figure: 20

For each of the plain text files and the plain text emails the probabilities overlapped with very close values ranging from 96.07 to 98.71 percent.

The files with only nouns, verbs, interjections showed probabilities which lie very far from 96 such as 70.15, 86.13 and 88.01 respectively.

A scrambled email in which words were organized in random was used as input and we saw a score of 70.16.

Stealthy plaintext email scored a high value of 95.91 percent which shows that it is very similar to the training data and the plain text emails, proving that it has good grammatical structure.

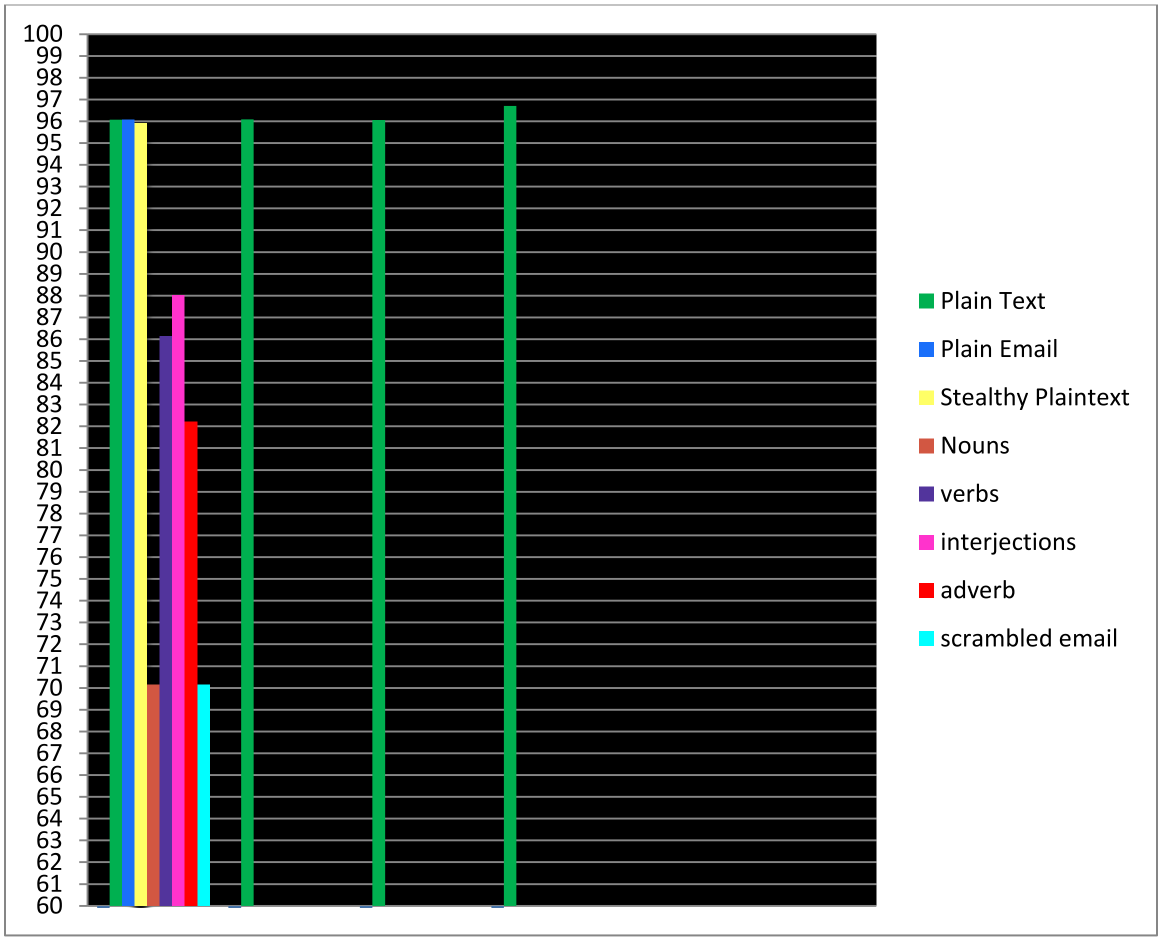


Figure 21: Probability of input text being English with 3 hidden states

6. Conclusion and Future Work

To summarize, we have developed a stealthy plain text generator which takes as input an email which contains “suspicious keywords” and converts it into an innocent looking email. The human mind would be able to recognize the awkwardness in the meaning of the stealthy plaintext since the sentences may not make complete sense, but after the evaluation of the converted text based on our assumptions about the automated analyzer, we are safe in assuming that we can easily pass off our stealthy plaintext email as a simple email. The English language has endless words and possible states in which we can combine and arrange words to form a sentence. Developing a complete grammatically correct and completely changed email while keeping the size of the original email intact is a very challenging problem. We understand that this is just the basis for a more sophisticated converter and can be improved in many ways so it is more efficient in hiding the transmission of data.

The basis of our project is stealthy encryption, where plain text replaces plain text. We have seen in the evaluation section that our project will be able to bypass an automated email analyzer which is based on our assumption that an analyzer looks for and flags random encrypted information. To validate our assumption, we conducted experiments which showed us that the entropy of the data does not change much when compared to plain text and lies far beyond the score of encrypted information.

We also took care of not letting the email look suspicious by encrypting our code generator file, which considered a list of numbers used to encode/decode into an image by LSB steganography, and could be identified as suspicious text by an automated email analyzer. The image, even after encoding the text, was minimally distorted, and the size of the image to be used for this could be small enough to pass of as part of the signature in an email, a common practice nowadays in the corporate world is to add in the company logo to emails.

A Hidden Markov Model was used to evaluate the Englishness of the stealthy plain text generated. We experimented with different kinds of texts as input to the HMM. A higher score in by the HMM evaluation means there is a high probability that the given text confirms to the grammatical syntax. We saw that the stealthy plaintext scored above 95 percent which is a distinguishable score and lies close to the score of plain English.

The techniques used by the analyzer may vary from one company to another. A project or program always has scope for improvement, and ours is no different. There could be lot of potential improvements to the “stealthy plaintext generator”. We need to share the suspicious keywords file initially between the sender and receiver. Future work can be to come up with a technique to handle the sharing in a more secure manner. There are many machine learning techniques which also can be used to test the Englishness of our converted text.

Currently our project has two parts before an email can be sent. We generate the stealthy plaintext and then encode the code into an image. This was done as the project started out with just one part and branched out to a second part, which required more interaction, making it necessary to add a graphical user interface for the latter. To make this tool easy to use for the user we could handle both these processes with a single process. Improvements to the project can be made such that a better technique could be used to perform the image steganography. We also see a limitation in the vocabulary which is confined to the dictionary size created. The number of states considered for the Hidden Markov Model can be further increased to get more consistent HMM scores. We plan to improvise on this in the future.

The idea implemented in this project is just the beginning for much more sophisticated systems. As the detection techniques get more elaborate we would have to come up with better ways of hiding data without encryption.

References

1. *American management association press room*. (2008, February 28). Retrieved from http://press.amanet.org/press-releases/177/2007-electronic-monitoring-surveillance-survey/

2. Martina Simova, Chris Pollett, Mark Stamp, "*Stealthy Ciphertext*" Department of Computer Science, San Jose State University

3. Susan E.Gindin, "*Guide to email and the internet in the workplace*", URL: http://www.info-law.com/guide.html#email (Accessed: Aug 30, 2010)

4. Shamir Adi, van Someren Nicko. 1998. *Playing hide and seek with stored keys*. URL: http://www.ncipher.com/resources/downloads/files/white_papers/keyhide2.pdf

5. Schneier, B. (1996). " Detecting Encryption." *Applied cryptography, protocols, algorithms, and source code in c*. John Wiley & Sons Inc.

6. Stamp, M. (2005). *Information security: Principles and practice*. Hoboken, New Jersey: John Wiley & Sons, Inc.

7. Cole, Eric B. *Methodology, System and Computer Readable Medium for Detecting File Encryption*. Sytex, Inc., assignee. Patent 7564969. 21 July 2009. Print.

8. Kundur, D., Ahsan, K., 2003: *Practical Internet Steganography: Data Hiding in IP*, Proceedings of the Texas Workshop on Security of Information Systems, April 2nd, 2003

9. Silman, J., "*Steganography and Steganalysis: An Overview*", SANS Institute, 2001

10. James C. Judge (2009). *Steganography: Past, Present, Future*. [URL] http://www.sans.org/reading_room/whitepapers/stenganography/steganography-past-present-future_552

11. M. Stamp, *A revealing introduction to hidden Markov models*. http://www.cs.sjsu.edu/faculty/stamp/Hampton/HMM.pdf

12. Francis, W. N. and H. Kučera. 1964. *Manual of Information to accompany A Standard Corpus of Present-Day Edited American English*, for use with Digital Computers. Providence, Rhode Island: Department of Linguistics, Brown University. Revised 1971. Revised and amplified 1979.

13. Shannon, Claude E.: Prediction and entropy of printed English, *The Bell System Technical Journal*, 30:50–64, January 1951.

14. T Morkel, JHP Eloff and MS Olivier, "*An Overview of Image Steganography*," in Proceedings of the Fifth Annual Information Security South Africa Conference (ISSA2005), Sandton, South Africa, June/July 2005 (Published electronically)

15. "*Hidden Markov Model*." Wikipedia. Wikimedia Foundation, 19 Nov. 2012. Web. 19 Nov. 2012. <http://en.wikipedia.org/wiki/Hidden_Markov_model>.

16. Han Shu, On-line handwriting recognition using hidden markov models, Master's thesis, Department of Electrical Engineering and Computer Science, the Massachusetts Institute of Technology, 1996.

17. Rabiner, L.R (1989). *"A Tutorial on Hidden Markov Models and selected applications in speech recognition"*