Classical Ciphers and Cryptanalysis

Brian Carter and Tanja Magoc

September 11, 2007

1 Introduction

Cryptography is the study of transmitting secret messages securely from one party to another. To accomplish this task, the original text, called plaintext, is "translated" into an encrypted version, called ciphertext, which is sent to the intended recipient. The recipient decrypts the text to obtain the original message. Cryptanalysis is process of analyzing the "hidden" message to learn information about the cryptosystem used to produce the ciphertext. Cryptanalysis includes studying the safety of these systems to prevent attacks by unauthorized malicious users who try to break the message before it reaches the intended recipient.

The main tool used for encrypting messages is a cipher. A cipher is a well defined procedure that states how to "hide" each character from the original text in the encrypted text and how the recipient should decrypt the ciphertext to read the original message. Since the early stages of cryptography development, around the year 500 BCE, different ciphers have become used, in practice. Each cipher is defined by a key, which describes the procedure of encrypting and decrypting the text. If the key is known, it is easy to decrypt the text "hidden" behind the transmitted symbols. The complexity and the security of ciphers range from very simple ones, whose keys are easily breakable these days, such as the classical ciphers, through more complex ciphers, such as Data Encryption Standard (DES) and Advanced Encryption Standard (AES), to the complex and yet not vulnerable ciphers, such as RSA ([12], [13]).

The classical ciphers are the absolute simplest forms of encryption; they have been around for thousands of years. There are essentially only two principles that drive all of the classical ciphers: substitution and transposition. Substitution ciphers are simply those that replace symbols in plaintext with another symbol of the same alphabet. To reverse the process, each ciphertext symbol is simply replaced with the corresponding original plaintext symbol. A transposition cipher simply rearranges the symbols in plaintext to produce ciphertext.

These principles are extremely basic at their cores and are simple to understand. However, the principles of substitution and transposition form the basis for many of today's encryption standards. By combining substitution and transposition in creative ways, it is possible to produce extremely secure cryptosystems.

1.1 Definition

A cryptosystem can be defined as following [1]:

- $\mathcal{M}$, the plaintext message space: this is the set of strings over some alphabet
- $\mathcal{C}$, the ciphertext message space: this is the set of possible ciphertext messages of the message space $\mathcal{M}$

- $\mathcal{K}$, an encryption key space: this is the set of possible encryption keys
- $\mathcal{K}'$, a decryption key space: this is the set of possible decryption keys
- $\mathcal{G} : \mathbb{N} \mapsto \mathcal{K} \times \mathcal{K}'$, a key generation algorithm
- $\mathcal{E} : \mathcal{M} \times \mathcal{K} \mapsto \mathcal{C}$, an encryption algorithm
- $\mathcal{D} : \mathcal{C} \times \mathcal{K}' \mapsto \mathcal{M}$, a decryption algorithm.

Given an integer 1^ℓ, $\mathcal{G}(1^\ell)$ outputs a key pair $(ke, kd) \in \mathcal{K} \times \mathcal{K}'$ of length ℓ.

For $ke \in \mathcal{K}$ and $m \in \mathcal{M}$, we define $c = \mathcal{E}_{ke}(m)$ to be the encryption transformation. Obviously, $c \in \mathcal{C}$. Similarly, for $kd \in \mathcal{K}'$, we define $m = \mathcal{D}_{kd}(c)$ to be the decryption transformation. In a cryptosystem, for each $m \in \mathcal{M}$ and $ke \in \mathcal{K}$, there exists $kd \in \mathcal{K}'$ such that $\mathcal{D}_{kd}(\mathcal{E}_{ke}(m)) = m$.

2 Substitution Ciphers

Substitution ciphers are the simplest ciphers used in cryptography. The most obvious substitution cipher is the Caesar cipher, which was in fact used by Julius Caesar to communicate with his army. The Caesar cipher is a typical example of the simplest class of substitution ciphers, the monoalphabetic substitution ciphers.

2.1 Monoalphabetic Substitution Ciphers

A monoalphabetic substitution cipher [1] is one for which $\mathcal{E}_k(m)$ is a simple substitution function which replaces each $m \in \mathcal{M}$ with a corresponding $c \in \mathcal{C}$ according to the cipher key k. Similarly, $\mathcal{D}_k(c)$ simply performs the reverse substitution of $\mathcal{E}$.

For illustration purposes, let's consider the Caesar cipher [2]. The cipher's alphabet is the 26 capital Roman letters and its key is the shift to the right by three positions. Thus, the Caesar cipher defines $\mathcal{E}_k(m)$ as the following mapping:

$$\begin{bmatrix} b \in m & : & \text{A} & \text{B} & \text{C} & \text{D} & \text{E} & \text{F} & \text{G} & \text{H} & \text{I} & \text{J} & \text{K} & \text{L} & \text{M} \\ & & \text{N} & \text{O} & \text{P} & \text{Q} & \text{R} & \text{S} & \text{T} & \text{U} & \text{V} & \text{W} & \text{X} & \text{Y} & \text{Z} \\ \mathcal{E}_k(b) & : & \text{D} & \text{E} & \text{F} & \text{G} & \text{H} & \text{I} & \text{J} & \text{K} & \text{L} & \text{M} & \text{N} & \text{O} & \text{P} \\ & & \text{Q} & \text{R} & \text{S} & \text{T} & \text{U} & \text{V} & \text{W} & \text{X} & \text{Y} & \text{Z} & \text{A} & \text{B} & \text{C} \end{bmatrix}.$$

Then, of course, the corresponding decryption algorithm $\mathcal{D}_k(c)$ is

$$\begin{bmatrix} b \in c & : & \text{D} & \text{E} & \text{F} & \text{G} & \text{H} & \text{I} & \text{J} & \text{K} & \text{L} & \text{M} & \text{N} & \text{O} & \text{P} \\ & & \text{Q} & \text{R} & \text{S} & \text{T} & \text{U} & \text{V} & \text{W} & \text{X} & \text{Y} & \text{Z} & \text{A} & \text{B} & \text{C} \\ \mathcal{D}_k(b) & : & \text{A} & \text{B} & \text{C} & \text{D} & \text{E} & \text{F} & \text{G} & \text{H} & \text{I} & \text{J} & \text{K} & \text{L} & \text{M} \\ & & \text{N} & \text{O} & \text{P} & \text{Q} & \text{R} & \text{S} & \text{T} & \text{U} & \text{V} & \text{W} & \text{X} & \text{Y} & \text{Z} \end{bmatrix}.$$

A plaintext message such as

```
THE FAULT DEAR BRUTUS LIES NOT IN OUR STARS BUT IN OURSELVES
```

would be encoded into the ciphertext

```
WKH IDXOW GHDU EUXWXV OLHV QRW LQ RXU VWDUV EXW LQ RXUVHOYHV
```

□

2.2 Polyalphabetic Substitution Ciphers

A polyalphabetic substitution cipher [1] is simply multiple monoalphabetic substitution ciphers applied to the same plaintext message m. By far, the best known polyalphabetic substitution ciphers is the Vigenère cipher ([3], [1]).

To use the Vigenère cipher, we must select a key string. The key string must be longer than a single symbol and may only contain symbols from the alphabet used to create the cipher. If ℓ denotes the key string's length, we divide the plaintext message into substrings of length ℓ with the possible exception of the last section, which may be shorter than ℓ symbols. Then, the Vigenère encryption algorithm applies the monoalphabetic substitution cipher to each symbol of the plaintext message with a shift of s, where s is the $n^{th} (\text{mod } \ell)$ symbol in the concatenation of the key string with itself until it is the same length or longer than the plaintext message.

The easiest way to visualize this algorithm on paper is to produce a tableau, an $n \times n$ matrix of each symbol of the alphabet (of cardinality n) being used with the cipher. The first row of the tableau will contain the alphabet in its usual order. The next row will contain the alphabet shifted to the right by one, and so on, until the last row contains the alphabet shifted to the left by one symbol. Then, the current symbol of the key string will be used as an index along the x-axis of the tableau and the current symbol of the plaintext will be used as an index along the y-axis of the tableau. The symbol at the intersection of the axes is the encrypted symbol. To decrypt the ciphertext, the process is repeated by locating the position in the tableau at which the row for the current symbol of the key string intersects with the current symbol of the ciphertext. This row's index is the correct symbol of the plaintext.

As an illustration, consider the plaintext message

```
THE FAULT DEAR BRUTUS LIES NOT IN OUR STARS BUT IN OURSELVES
```

again. Using the key string `SHINE`, we would break up the plaintext message as follows:

```
THEFA ULTDE ARBRU TUSLI ESNOT INOUR STARS BUTIN OURSE LVES
```

□

Then, concatenating our key string with itself until it is at least as long as our plaintext message

```
SHINE SHINE SHINE SHINE SHINE SHINE SHINE SHINE SHINE SHIN
```

We use each pair of symbols as indexes into our tableau and obtain the following ciphertext:

```
LOMSE MSBQI SYJEY LBAYM WZVBX AUWHV KAIEW TBBVR GBZFI DCMF
```

□

By applying the aforementioned process to the ciphertext, we again obtain our plaintext message.

3 Transposition Ciphers

A transposition cipher [1], also sometimes called a permutation cipher, is one for which applying $\mathcal{E}$ to plaintext produces ciphertext with the same symbols as the plaintext, but rearranged in different positions.

We must divide the plaintext message into message blocks. If b is the message block length, we define $\pi = (\pi(1), \pi(2), \ldots, \pi(b))$ such that $\pi(n)$ is the position within the message block that should be used for the plaintext symbol in the ciphertext. Obviously, $\pi(n)$ must be unique for

$1 \leq n \leq b$. This allows for $b!$ possible permutations of the key π. $\pi(n)^{-1}$, then, becomes the position in the plaintext at which each symbol of the ciphertext should be placed.

Similarly, we define $\pi^{-1} = (\pi(1)^{-1}, \pi(2)^{-1}, \ldots, \pi(b)^{-1})$ to be the decryption key.

For example, suppose that $b = 5$ and $\pi = \pi(\pi(1), \pi(2), \pi(3), \pi(4), \pi(5)) = \pi(2, 5, 4, 1, 3)$. We break up the plaintext message into blocks of the message length b

```
THEFA ULTDE ARBRU TUSLI ESNOT INOUR STARS BUTIN OURSE LVES
```

which can now be enciphered, producing

```
FTAEH DUETL RAUBR LTISU OETNS UIRON RSSAT IBNTU SOERU SL␣EV
```

□

In this case, a space ␣ occurs in the last message block to substitute for the missing 5^{th} character in that block. In practice, removal of any spaces, which makes deciphering the final message block using the same algorithm impossible, should be performed; with the spaces present, it is much easier to determine the encryption key.

4 Cryptanalysis

As mentioned earlier, cryptanalysis is the process of studying ciphers in order to avoid unauthorized users from reading the encrypted message. Even though the first developed ciphers, such as substitution ciphers, were thought to be safe from hackers, different techniques have been developed to easily find the key of the cipher and therefore, have the ability to decrypt the entire ciphertext. Several possible methods to break a substitution cipher include exhaustive search, frequency analysis, genetic algorithm, simulated annealing, tabu search, particle swarm optimization, and relaxation algorithm. These techniques are described in the following subsections.

4.1 Exhaustive Search

The exhaustive search method is the simplest to understand and implement out of all algorithms used to break substitution ciphers. This technique is possible since most cryptographic systems have a finite key space allowing for all possible keys to be checked until the correct one is found. This method could be an acceptable technique for breaking a monoalphabetic shift cipher since there are only as many possibilities for the key as the number of letters in the original alphabet (i.e. 26 possibilities for English alphabet). However, if we consider a monoalphabetic cipher in which any kind of permutation among letters is allowed, the number of possible keys rapidly increases to the factorial of the number of letters in the original alphabet (e.g., 26! possible keys for the English language). In the case of polyalphabetic ciphers, the keyword used for determining the next key in encrypting the text can be of a differing length, which leads to an even larger number of possible keys.

Even though at first, it seems that the brute force method is a reasonable way of trying to break simple substitution ciphers, in most cases it will end up being an unfeasible algorithm since the number of possible keys that might need to be checked is very large. Thus, practically, it would be impossible to do an exhaustive search in a reasonable amount of time. To overcome this deficiency, new algorithms have been developed to allow for faster breaking of the cipher [6].

4.2 Frequency Analysis Method

One of the algorithms to crack substitution ciphers much faster than the exhaustive search algorithm is the frequency analysis method ([1],[4], [5]). This is often thought of as the "classic" method of decrypting substitution ciphertext. Frequency analysis is the process of determining at which frequency each symbol of the encrypted message occurs within the ciphertext. This information can be used along with a knowledge of symbol frequencies within the language used in the cipher to help determine which ciphertext symbol maps to which corresponding plaintext symbol. For example, in the English language, the letter `e` is by far the most common.

It is possible to extend this basic principle: we can also determine the frequencies at which bigrams and trigrams (groups of two and three symbols, respectively) occur within the ciphertext and relate them to the known facts about the original language. For example, in the English language, `ng` and `the` are the most common bigrams and trigrams.

Essentially, decryption using frequency analysis involves making educated guesses of symbol mappings using knowledge of symbol, bigram, and trigram frequency. After obtaining a partial solution, the person analyzing the ciphertext can sometimes determine certain patterns that occur within the ciphertext. For example, it may be possible to infer that `the␣e` (where ␣ represents an unknown ciphertext symbol) might be the word `there` in plaintext. This is a painstakingly tedious process that often involves wrong guesses and backtracking. Success can vary dramatically based on the amount of available information about the cryptosystem used to produce the ciphertext.

4.3 Genetic Algorithm Method

Even though the frequency analysis method is a relatively effective way of breaking a substitution cipher, it has some weaknesses, the biggest one being that, in most cases, a human is needed to complete the algorithm. On the other hand, automated attack algorithms have been developed for which no human intervention is necessary. These automated methods are run on a computer and are finished either after a predetermined number of iterations or after a message has been successfully decrypted [7]. One of the automated attack algorithms that is widely used for cracking substitution ciphers is the genetic algorithm.

Genetic algorithms have found applications in many areas that involve search, optimization, and machine learning with the goal of maximizing the stated objective function. As the name suggests, the genetic method is based on human genetic processes. It starts with a random "population" of inputs (represented as a string) to the objective function. Then, it generates a new population from the existing population in a certain way that guarantees that the new population is on average better than the previous one. The changes made to a population to produce a better new population is the combination of selection, mating, and mutation, just as is the case in reproducing a human population. The selection process determines which strings will be used to produce a new generation. The strings which are currently producing the best results will have greater chance of being chosen for "reproduction." The mating stage decides how to combine the chosen strings. Usually predetermined positions of one string are kept and the rest is filled out by the corresponding elements from the other string. Finally, the mutation process determines which positions of a string will be changed to a different value. This general genetic algorithm is proven to be very efficient in many search problems [6].

Decrypting ciphertext can be viewed as a search for the correct key; thus, it is reasonable to believe that the genetic algorithm can be used to solve the problem of breaking a cipher. First of all, a random key is used to decrypt the ciphertext. The frequency analysis is applied to the "decrypted" text (i.e., the text produced by applying the chosen key) and it is compared to the frequency analysis

of the language used to encrypt the original message. The value of a fitness function is calculated. A higher result of the function implies a higher fitness of the chosen key, and therefore, a closer answer to the correct key. The fitness function can have different forms which are based on analysis of single characters (unigram analysis), pairs of characters (digram analysis), triplets of characters (trigram analysis), or any combination of any of these. The most commonly used fitness function for breaking the simple substitution ciphers is based only on unigram and digram analysis and has the form $fitness = (1 - \sum_{i=1}^{26}\{|SF[i] - DF[i]| + \sum_{j=1}^{26}|SDF[i,j] - DDF[i,j]|\}/4)^8$, where $SF[i]$ and $SDF[i,j]$ are standard frequencies of character i and the pair of symbols (i,j) in the original language, respectively, and $DF[i]$ and $DDF[i,j]$ are measured frequencies of character i and pair (i,j) in the "decrypted" text. The measured errors (i.e., the difference between standard frequencies and measured frequencies) are normalized and subtracted from 1, so that number closer to 1 represents the higher fitness. Also, the constants 4 and 8 are used to reduce sensitivity to large errors and to amplify small differences, respectively. Some fitness functions give more weight to one of the factors (unigram or digram) by inserting weight constants in front of the corresponding error calculation terms ([6], [7]).

To apply a genetic algorithm to crack a substitution cipher, a population of keys is randomly chosen and the output of the fitness function is calculated for each key in the population. The next step is to randomly select two keys from the current population of keys for mating. Even though the selection is random, it is more probable that keys with higher fitness will be chosen. The mating will generate two new keys, which will replace the old keys that have lower fitness. Mating is done in the following way: two strings are scanned from one end and compared character by character. When comparing two characters, the character that is more frequent in the ciphertext is chosen to build a new key. If the more frequent character is already used in the new key, then the other character is used. If both characters already appear in the new key, then a random character that has not yet been used is inserted in the new key. The second of the new keys is generated in the same way, but the scanning starts from the other end of the keys. Finally, the mutation process is applied. A character in the key is selected for mutation with a small probability. If a character chosen for mutation is in the lower part of the frequency analysis, it is swapped with a randomly chosen character; otherwise, it is swapped with the character to its right. The genetic algorithm is iterated a predetermined fixed number of times, and the key with the highest fitness in the final population pool will be used to decrypt the ciphertext.

Like all the other heuristic algorithms, the genetic algorithm does not always produce the exact answer, but rather it gives a solution that is close to the correct one. In the case of deciphering the ciphertext, after using the "best" key produced by genetic algorithm, most of the time, it is easy for a human to read the "decrypted" text and make a small changes to reproduce the correct plaintext. The experiments performed using the genetic method suggest that the fitness of about 0.9 is enough to determine the vowel substitutions and enough consonant substitutions after which the visual examination by a human can be used to decrypt the entire text. Also, it was deduced that producing about 100 generations is usually sufficiently enough to reach the fitness level of 0.9. Iterating the algorithm this many times takes a very short period of time, which suggests that in most cases, the genetic algorithm produces good results in a reasonable amount of time, and therefore is a good tool for attacking simple substitution ciphers [6].

4.4 Simulated Annealing

Another optimization algorithm used to break substitution ciphers is simulated annealing. This method is very similar to the genetic algorithm with the main difference being that the genetic algorithm holds a pool of possible keys at each moment, while the simulated annealing keeps only

one value at a time, which combined with a few other simplifications, makes this approach much simpler than the genetic algorithm ([7], [9]).

The simulated annealing algorithm mimics the physics process of annealing in metals; that is, slowly cooling a heated metal in order to attain a minimum energy state ([7], [9]). The idea of the annealing process is that particles in a metal move from energy level E_1 to level E_2 with probability $P(E_1, E_2) = \begin{cases} 1, & \text{if } E_2 - E_1 \leq 0 \\ e^{\frac{-(E_2-E1)}{kT}}, & \text{otherwise} \end{cases}$, where k is Boltzmann constant and T is the temperature. The algorithm starts by choosing a random solution as a current best solution and by initializing the temperature to some random value. At a given temperature, several attempts are made to perturb the current solution. For each attempted change, the change in cost $(E_2 - E_1)$ is calculated, and the probability of changing state to the new energy level is calculated based on the above probability equation. If the probability is greater then 0.5, the perturbation is accepted and the current solution is updated.

The temperature is reduced by a fixed amount when a predefined number of perturbation attempts is reached. The algorithm ends when no updates happen for a certain temperature or if the temperature falls below some value that is set in advance.

For the substitution cipher attack, the cost is calculated by the fitness function introduced in genetic algorithm section and the perturbations are obtained by swapping two randomly chosen characters. Thus, the algorithm starts with initializing the current solution by a random solution and calculating its fitness function. The initial temperature is set to a random value and the predetermined number of perturbations is applied to the current solution. The solution is updated if the new (perturbed) solution satisfies the conditions for updating solution–that is, if the probability of changing states is greater than 0.5. After the iterative loop is finished, the temperature is reduced by a predetermined value and the iterative loop is executed again. The algorithm ends when a satisfying solution is found or when the temperature has already been reduced a (predetermined) large number of times ([7], [9]).

In comparison to the genetic algorithm, the simulated annealing method is slightly weaker in terms of recovering the correct characters from the ciphertext. The genetic algorithm correctly matches more letters than the simulation annealing does for any length of the ciphertext; however, this difference is not too big, so the simulated annealing method is still a good tool for breaking simple ciphers. To reach the correct or almost correct solution, simulated annealing takes fewer iterations, but the total time it takes is longer than the time needed by genetic algorithm to reach the same result since the simulated annealing method spends long time in each iteration because of its detailed examination of each possible perturbation. Even though the genetic algorithm seems to be better than simulated annealing in all aspects, the main advantage of the simulated annealing is that it is much easier to implement and therefore, it still has an important role in cracking simple substitution ciphers ([7], [9]).

4.5 Tabu Search

Tabu search is an optimization technique used to improve the performance of genetic algorithms and simulated annealing in breaking substitution ciphers. The algorithm initializes the best solution with a random key and calculates its fitness by the fitness function discussed in the genetic algorithm section. A list of possible candidates is created by swapping two randomly chosen characters and the fitness of each candidate is calculated. If any of them has a higher fitness than the current best solution, the best solution is updated and the new pool of candidates is created based on the new current best key. Each candidate stays in the pool, called the tabu list, for a certain number of iterations, which ensures that search does not go back quickly to already checked solutions. The

algorithm ends after a predefined number of iterations or if there is no improvement in the best solution for a certain number of iterations ([7], [8]).

Comparing the results of the tabu search, the genetic algorithm, and simulated annealing leads to the conclusion that all three algorithms can almost equally well recover the correct characters from the ciphertext. The experimental results suggest that the genetic algorithm recovers slightly more characters than the other two algorithms, but each algorithm has its own advantages. Simulated annealing algorithm is much simpler to implement than genetic algorithms and the tabu search, while the tabu search has the advantage over the other two methods because it prevents the best current solution to be "stuck" in the local minimum for some time. This allows the tabu search to obtain the desired result faster than with the other two algorithms ([7], [8], [9]).

4.6 Particle Swarm Optimization

The particle swarm optimization method is another algorithm based on machine learning processes that is used for cracking substitution ciphers. The algorithm starts by choosing a random population of potential solutions, each of which is called a particle. Each particle keeps the coordinates of its best position found so far, called *pbest*. The best position of all particles is held in *gbest*. The particles change coordinates towards *gbest* according to the formula $P_{i,j} = P_{i,j} + v_{i,j}$, where $v_{i,j}$ is the velocity updated by $v_{i,j} = c_0 v_{i,j} + c_1 r_1 (Ppbest_{i,j} - P_{i,j}) + c_2 r_2 (Pgbest_{i,j} - P_{i,j})$. Here r_1 and r_2 are uniformly distributed random variables and c_0, c_1, and c_2 are learning factors. If the new coordinates of the particle produce a better result than the current *pbest*, then the *pbest* is updated accordingly and compared to *gbest* for possible update [11].

For cracking substitution ciphers, the coordinates of the "position" of a particle is a permutation of letters from the alphabet; the best position is defined by the fitness function described in genetic algorithm section. Generally, particles are considered in multidimensional space where dimensions do not depend on each other. However, in the problem of breaking substitution ciphers, the dimensions are not totally independent since each character from the alphabet should be used only once. Thus, a slightly modified version of particle swarm optimization is used. The higher velocity means that there is grater chance that the change of coordinates will happen rather than expecting a large coordinate change as is the case in general application of the particle swarm optimization technique. The coordinate change is just swapping two characters rather than an independent update of a single position.

The experimental results show that particle swarm optimization is a good tool for breaking simple substitution ciphers as long as bigram is used to calculate the fitness of particles. Using a unigram as a measure for fitness does not give any useful results for deciphering even the simplest ciphers. The combination of two options as well as using trigram has not been explored yet [11].

4.7 Relaxation Algorithm

Another method used to break substitution ciphers is the relaxation algorithm. The relaxation algorithm is a graph-based technique that relies on iterative parallel updating of values associated with each node. The nodes of the graph, v_i, are elements of the cipher alphabet; each node has a random variable, l_i, associated with it which represents the probabilities of the possible characters that this node represents. The random variables are initialized with the identity assumption– that is, the guess that each letter in ciphertext represents the exact same letter in the plaintext. Thus, the original probabilities assigned to the nodes are the frequency analysis results from the given language using unigram statistics. The probabilities of a node are updated based on the appearance of its two neighbors in the ciphertext and the trigram analysis of the original language.

The iterative procedure is performed to update probabilities of the possible candidates for each node in the graph. The algorithm ends when all letters are recovered correctly or after a predetermined number of iterations. The character that has the highest probability for a certain node at the end of the algorithm is assumed to be the correct letter represented by that node [10].

This method was applied to two different ciphers and showed pretty good results in recovering the original text. However, both examples seem to be using the monoalphabetic substitution cipher, so it is not quite proven if relaxation algorithm could be used to solve other types of substitution ciphers. Also, only two examples probably do not provide enough evidence for success of a method even though they do give an insight into a possibly applicable approach [10].

5 Conclusion

Since the early stages of transmitting secret information, different types of ciphers have been developed for this purpose. The substitution ciphers are the simplest form of "hiding" messages from unauthorized users. However, all types of substitution ciphers can be broken these days with very high accuracy. The most natural first attempt using the exhaustive search is probably not the best choice, since it is time consuming, but it decrypts the original text with 100% accuracy. The frequency analysis algorithm is surely the most accurate fast approach to decipher text. However, it requires the knowledge of the frequency statistics of the language used to write the original text. The biggest disadvantage of this process is that it relies on constant human interaction to determine the next move in the process.

To overcome the necessity for human intervention at all times, several heuristic approaches have been designed. Simulated annealing, a method based on the cooling down a heated metal, is the simplest one of optimization techniques used for cracking substitution ciphers. It takes a very detailed analysis at each step of iteration to produce a reasonable resultant key. This key is not always perfect, but with a little intervention from a human, the key produced by the algorithm can easily be modified to the correct key. Simulated annealing is a very easily implemented algorithm, and therefore, it has a wide application in breaking substitution ciphers. The genetic algorithm, which is based on the human reproduction process, offers an improvement to simulated annealing in that it recovers more correct characters than simulated annealing, and its running time is slightly lower. The tabu search is the fastest one of the optimization techniques with a little trade-off for accuracy of the located key.

The particle swarm optimization algorithm is based on physics and particle movement characteristics to progress toward a better solution. It gives good results when breaking substitution ciphers, but it seems that it has not been yet fully explored to give better results than the other heuristic methods. Another not completely examined algorithm that is used for hacking simple ciphers is the relaxation algorithm, which is a graph-based method that relies on parallel updating of probabilities for each node in the graph.

As described in the paper, there are several good approaches to break a substitution cipher, each of them having advantages and disadvantages over the other methods. Also, it has been presented that some algorithms have not yet been tested to find out their full capabilities. Thus, there is still lots of work to be done in this area even though the substitution ciphers are not used anymore these days for secure transmission of information. However, substitution ciphers represent the basic building blocks of more complex and more secure ciphers that are used today; thus, understanding the vulnerability of simple ciphers is important in using and building more complex ciphers.

References

[1] Mao, W., *Modern Cryptography: Theory & Practice*. Upper Saddle River, NJ: Prentice Hall PTR, 2004.

[2] Chris Savarese and Brian Hart, "The Caesar Cipher," *Historical Cryptography Web Site*, Trinity College, 1999, <http://starbase.trincoll.edu/∼crypto/historical/caesar.html> (6 September 2007).

[3] R. Morelli, "The Vigenere [*sic*] Cipher," *Historical Cryptography Web Site*, Trinity College, <http://starbase.trincoll.edu/∼crypto/historical/vigenere.html> (6 September 2007).

[4] "Cryptanalysis," *Wikipedia*, <http://en.wikipedia.org/wiki/Cryptanalysis> (6 September 2007).

[5] "Frequency analysis (cryptanalysis)," *Wikipedia*, <http://en.wikipedia.org/wiki/Frequency_analysis> (6 September 2007).

[6] Spillman, R., Jansses, M., and Kepner, M. *Use of genetic algorithm in the cryptoanalysts of simple substitution ciphers*, Cryptologia, vol. 17, issue 1, January 1993, 31-34.

[7] Clark, A., *Modern optimization algorithms for cryptanalysis*, Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems, Nov. 29-Dec, 2, 1994 Page(s):258 - 262.

[8] Verma, A. K., Dave, M., and Joshi, R. C., *Genetic algorithm and tabu search attack on the mono-alphabetic substitution cipher i adhoc networks*, Journal of Computer Science 3, 134-137, 2007.

[9] Dimovski, A. and Gligoroski, D., *Attacks on the transposition ciphers using optimization heuristics*, Proceeding of ICEST 2003, October 2003, Sofia, Bulgaria.

[10] Peleg, S. and Rosenfeld, A., *Breaking substitution ciphers using a relaxation algorithm*, Communications of teh ACM, NOvember 1979, vol. 22.

[11] Uddin, M. F. and Youssef, A. M., *Cryptanalysis of simple substitution ciphers using particle swarm optimization*, 2006 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, July 16-21, 2006.

[12] "History of cryptography", *Wikipedia*, <http://en.wikipedia.org/wiki/History_of_cryptography>

[13] "Cryptography", *Wikipedia*, <http://en.wikipedia.org/wiki/Cryptography>