STA4000 Report
Decrypting Classical Cipher Text Using Markov Chain Monte Carlo

Jian Chen

Supervisor: Professor Jeffrey S. Rosenthal

May 12, 2010

Abstract

In this paper, we present the use of Markov Chain Monte Carlo (MCMC) methods to attack the classical ciphers. We use frequency analysis as our basic tool. First we investigate the MCMC algorithm to break a simple substitution cipher. Then the algorithm is modified to break a simple transposition cipher. Our results on the transposition cipher are better than the existing methods such as simulated annealing, genetic algorithm and tabu search. We also extend the algorithms to break a simple substitution-transposition cipher up to key length 40.

1 Introduction

Cryptography is the study of the algorithms to encrypt and decrypt message between senders and receivers. The original text is called the plain text. The encrypted text is called the cipher text. The algorithms to perform encryption and decryption are refer to the ciphers. Usually a cipher contains one or two keys. In a symmetric key algorithm (e.g. DES), the decryption key is the same as the encryption key (or just the inverse function of it). In an asymmetric key algorithm (e.g. RSA), two different keys are used. The public key is used for encryption and a private key is for the decryption.

1.1 Classical Ciphers and Modern Ciphers

Ciphers can also be categorized in a different way, the classical cipher and the modern cipher. Substitution and transposition ciphers are 2 basic classical ciphers. The text are performed at the byte level.

A simple substitution cipher works by replacing each letter with another one. In our paper, we only substitute alphabetic letters. The spaces are leaved untouched. So the number of the possible keys are 26!. Table 1.1a illustrate an encryption and decryption example of a simple substitution cipher. For the encryption, all 'A's in the plain text are replaced by letter 'X', 'B's replaced by 'E', etc. For the decryption, all 'A's in the cipher text are replaced by 'I', 'B's replaced by 'C' etc. Note the encryption key is the inverse function of the decryption key.

Another classical cipher is the transposition cipher (also called the permutation cipher). The letters in the plain text stay the same but their positions are rearranged in a different

plain text	THE PROJECT GUTENBERG EBOOK OF OLIVER TWIST
encryption key	XEBPROHYAUFTIDSJLKZMWVNGQC
cipher text	MYR JKSURBM HWMRDERKH RESSF SO STAVRK MNAZM
decryption key	ICZNBKXGMPRQTWFDYEOLJVUAHS
decrypted text	THE PROJECT GUTENBERG EBOOK OF OLIVER TWIST

Table 1.1a: Simple substitution cipher encryption and decryption example.

order. A simple transposition cipher works by splitting the plain text into fixed sized blocks. The length of the key (also called the period) is the same as the size of the block. Letters in each block are permuted according to a same pattern(the key). Table 1.1b illustrate an example of encryption and decryption by a transposition cipher with key length 10. Note the encryption key is the inverse function of the decryption key.

plain text	T	H	E		P	R	O	J	E	C
encryption key	1	9	3	7	0	4	5	8	6	2
cipher text	H	C		J	T	P	R	E	O	E
decryption key	4	0	9	2	5	6	8	3	7	1
decrypted text	T	H	E		P	R	O	J	E	C

Table 1.1b: Simple transposition cipher encryption and decryption example.

The product cipher combines a sequence of simple transformations such as substitution, transposition and other arithmetic. SP-network (Substitution-permutation network) is a special case of the product cipher. It only involves the substitutions (S-box) and permutations (P-box). SP-network is very common in the design of modern ciphers such as DES. For a simple substitution-transposition cipher, the plain text is encrypted by a substitution cipher followed by a transposition cipher.

Modern ciphers are more secure than the classical ciphers. The encryption and decryption are performed on the bit level. DES is a symmetric key algorithm. RSA is an asymmetric key algorithm.

1.2 Cryptanalysis of classical Ciphers - Frequency Analysis

Cryptanalysis is the study of methods to obtain the meaning of the encrypted text. An attack refers to an algorithm to break a certain type of cipher. There are several types of attacks. In this paper we focus on the ciphertext only attack for classical ciphers, which means we would like to reveal the original text by only a few available decrypted text. Frequency analysis is the basic tool for breaking the classical ciphers.

The frequency analysis is the study of the frequencies of letters or combination of letters in the cipher text. In a particular language (e.g. English) certain letters and there combinations occurs more frequently than others. The frequencies of letters are also called n-gram, i.e. unigram stand for single letter frequencies, bigram for combination of 2 letters, trigram for 3 letters. For example, the E.g. in English 'E' is the most occurred single letter while 'Z' is the least used single letter. 'TH' and 'ER' can often be seen in the text. The general approach to break a classical cipher is by comparing the letter frequencies of the cipher text to a reference text. The reference text is usually a large text such as *War and Peace.*

More information about Cryptography and Cryptanalysis can be found in the book Applied Cryptography by Bruce Schneier. The discussion of frequency analysis can be found in Shannon's paper Communication Theory of Secrecy Systems.

1.3 Current Works on Cryptanalysis of Classical Ciphers

The Metropolis algorithm (Metropolis et al., 1953) [3] is very widely used to approximately sample from complicated probability distributions (see e.g. Gilks et al., 1996; Roberts and Rosenthal, 2004). More recently, this algorithm has been used as a method to break simple substitution cipher codes, by approximately converging to a target density on decryption functions which is designed to be largest when the decrypted pair frequencies most closely match a large reference text like *War and Peace*. This idea was first introduced by Marc Coram and Phil Beineke for the Stanford statistical consulting service (see the Introduction to Diaconis, 2008), and later studied more systematically by Connor (2003). In this paper, we significantly extend this approach to deal with more complicated substitution-plus-transposition ciphers, and find that it works well even with key lengths as large as 40.

2 MCMC Approach to Cryptanalysis of Classical Ciphers

The state space of the Markov chain contains all possible combinations of the decryption keys (a finite state space). Each possible key is a state in the Markov Chain. Following is the general MCMC algorithm to break the classical ciphers.

- A initial state (initial key) is chosen.
- Repeat the following steps for certain iterations (e.g. 10,000 iterations).
- For a current key x, we propose a new key y. The proposal is usually symmetry so the chain is detailed balanced.
- We calculate a score function $\pi(x)$ base on the frequencies of the decrypted text and the reference text. We do the same calculation for key y and get $\pi(y)$. (The score function is maximized when the frequencies of cipher text matches that of the reference text.)
- This means we accept proposal y with probability $(\frac{\pi(y)}{\pi(x)})^p$ (the acceptance rate) where p is a scaling parameter.

By Markov chain convergence theorem, the Markov chain will converge to its stationary distribution. In our case it will converge to the correct decryption key.

We will have more details on the choice of initial state, the proposal, the score function, acceptance rate and the scaling parameter for each attack algorithm.

2.1 Systematic testing methodology

We use the following texts to test our attack algorithms (table 2.2a). We use *War and Peace* as the reference text, *Oliver Twist* as the plain text during the programming and the tests. All 4 texts are used to test our final attack algorithms.

To make it simple, all letters are converted to upper case. All other non-alphabetic characters (i.e. spaces, punctuation, numbers) are included in the frequency analysis, but they are all converted to spaces. So totally we have 27 characters (26 upper case English

Text	Author	Publish Date
War and Peace	Leo Tolstoy	1869
Oliver Twist	Charles Dickens	1838
Pride and Prejudice	Jane Austen	1813
Ice Hockey (Wikipedia Page)	Wikipedia	2010

Table 2.2a: Cipher texts and reference texts used in our attacks.

alphabets and a space character). We also create indexes for each character for our processing (table 2.2b).

character	'A'	'B'	...	'Z'	' '
index	0	1	...	25	26

Table 2.2a: Characters and their indexes

We perform a systematic testing to evaluate our results. We run the encryption and the decryption process 100 times for each attack algorithm. Each run a random key is generated to encrypt the plain text and the attack is performed on the cipher text. At the end of the attack, we compare the decrypted text with the plain text. If the decrypted text is the same as the cipher text, it is a successful run. It is obvious the more successful runs, the better our attack is.

Sometime we are not able to reveal the full plain text. But if we successfully guessed most of them. We may still say our attacks are good because the remaining cipher text can be easily figured out by human eyes. So we also want to measure how close the decrypted text is to the plain text. We call this measurement the accuracy.

For a substitution ciphers, The accuracy is defined as $\frac{m_s}{n_s}$. Where m_s is the number of letters correctly revealed, $n_s$is the number of available letters in the plain text (usually n is 26, but it may be less than 26 for short cipher text). Note a letter is correctly revealed if the position of its first appearance in the plain text is the same as that of the decrypted text.

For a transposition ciphers. We defined accuracy as $\frac{m_t}{n_t}$. Where m_t is the number of letters correctly placed in one period (the key length). A letter is correct positioned if it has the same neighbors in the decrypted text as in the plain text. n_t is equal to the key length minus 2. We do not count the letters in the start and end positions as they only have one neighbor.

We also measure how long it takes for our attacks to run. Since a good attack should finish within reasonable time. Our program is written in C++ and is running on a MacBook Pro with the following system configuration (Table 2.2c).

CPU	2.26 GHz Intel Core 2 Duo
Memory	4GB 1067 MHz DDR3 Memory
OS version	Mac OS X Version 10.6.3
Compiler	g++ i386-apple-darwin10-g++-4.2.1

Table 2.2c: System configuration of the machine running the attacks

3 Our Results for Substitution Cipher

3.1 Uni-gram Attack

We want to associate the frequencies of the cipher text with the ones in the reference text. The simplest attack will be replacing the most occurred letter in the cipher text by the most frequent occurred letter in the reference text. Then the second, the third most occurred letter until the least occurred one.

We try this attack on the encrypted Oliver Twist novel (full text). It doesn't work very well. We only reveal 16 out of 26 letters in the cipher text (Table 3.1). We suspect it is because some letters (e.g. 'R' and 'S', 'C' and 'W') have similar frequencies.

plain text	THE PROJECT GUTENBERG EBOOK OF OLIVER TWIST
decrypted text	THE PSOJEWT FUTEIBESF EBOOK OG OLNVES TCNRT

Table 3.1: Decryption of Oliver Twist full text by uni-gram attack.

3.2 Bi-gram Attack by MCMC algorithm

We want to use other statistics such as the bi-gram. But first we defined the MCMC model of our attack.

3.2.1 The States of the Chain

The length of the key is 26. So we have totally 26! possible states. We let the initial state be 'ABCD...XYZ' so the decrypted text by using this key is identical to the cipher text.

3.2.2 The Score Function

The better way to relate the frequencies of the cipher text and the plain text is to use a so called score function. The value of the score function depends on the frequencies of the decrypted text and plain text. The closer the decrypted text to the plain text, the bigger the score value will be. For a particular state x, we define the score function as follow:

$$\pi(x) = \prod_{\beta_1,\beta_2} f_r(\beta_1,\beta_2)^{f_d(\beta_1,\beta_2)} \tag{1}$$

The score is based on bi-grams of the cipher text and reference text. β_1 and β_2 are all possible combinations of alphabets. $f_r(\beta_1,\beta_2)$ is pair frequency of letters β_1 and β_2 for reference text. $f_d(\beta_1,\beta_2)$ is the frequency from the cipher text.

Note, in our program we use the log score function for easy calculation. The frequencies f_r and f_c are added by one to avoid zeros (non-occurrence of certain pairs of letters may make the above product zero).

3.2.3 The Proposed next move

A key part of the MCMC algorithm is to define a proposal so the chain is detailed balanced and guaranteed to converge to its stationary distribution.

We propose a new key by swapping 2 randomly selected letters in the current key. So the transition probability is $1/n^2$. Unlike Connor's paper, swapping 'A' and 'A' (which gives the same key as current key) is a valid move in our program. Note it's a symmetric move so it's detailed balanced.

3.2.4 Acceptance probability

If the current state is x, the proposed key is y. The acceptance probability is $[\frac{\pi(y)}{\pi(x)}]^p$ where p is the scaling parameter (more details on section 3.2.6).

Now we have the mathematical model for bi-gram attack. We can apply it to break the substitution cipher.

3.2.5 Our Result for Bi-gram Attack

Table 3.2 shows by increasing number of iterations, we improve the accuracy and no. of successful runs. But the accuracy doesn't change much after 10,000 iterations. The accuracy is high (greater than 90%) which mean most of the runs are already very close to the correct result. But the number of success runs are pretty low around 50-60 out of 100. Next we try to improve the algorithm by tuning different parameters.

iterations	accuracy	no. of successful runs
1,000	0.5196	0
2,000	0.7732	19
5,000	0.9060	47
10,000	0.9064	51
20,000	0.9348	59
50,000	0.8932	54

Table 3.2: Decrypt substitution cipher by bi-gram.
How the accuracy depends on the no. of iterations
Use the last function

3.2.6 Looking for the Optimal Scaling Parameter

The scaling parameter is very important in the MCMC algorithm. Larger scaling parameter gives lower acceptance rate. But if the acceptance rate is too low, the chain is moving too slow and it will take too long to converge. Smaller scaling parameter gives higher acceptance rate. But if acceptance rate is too high, the chain will move too often and may not always stay in the stationary distribution.

Usually the optimal acceptance rate for a MCMC algorithm is around 0.234. But this value doesn't work very well for our attack on substitution ciphers. The best result is when the scaling parameter is set to 1 and the acceptance rate is 0.04. And if we increase the acceptance rate by lower the scaling parameter, the chain will not converge (On average it only reveal 26% of the cipher text after 10,000 iterations). So we should keep the scaling parameter to be 1.

3.2.7 Improve the Attack by Remembering the Best Score Function

Tuning the scaling parameter won't improve the algorithm. But after further investigation we found many runs already reveal the plain text (e.g. "THE PROJECT") in the middle of

scaling parameter	acceptance rate	no. of successful runs	
0.05	0.27	0.2664	0
0.1	0.12	0.6184	0
1	0.04	0.9064	51

Table 3.2a: Bi-gram Attack for substitution cipher.
Result after 10,000 iterations for different scaling parameters.

the process, but jump out of it (e.g. "THE PROZECT") later. We know when the algorithm found the correct answer, the log score function is maximized. So instead of using the last key for decryption, we use the key which gives the largest log score as the decryption key. Table 3.2b shows we get much better results by remembering the key which gives the best log score.

iterations	accuracy	no. of successful runs
1,000	0.5300	1
2,000	0.7716	20
5,000	0.9172	87
10,000	0.9312	90
20,000	0.9148	87
50,000	0.9488	93

Table 3.2b: Bi-gram Attack on substitution cipher.
How the accuracy depends on the no. of iterations.
Use the key which maximize the log score. Cipher text size is 2000.

3.2.8 How Much Cipher Text is Needed

The time spend on the decryption does not depend on the length of the cipher text used. But does more cipher text gives better results? Table 3.2c indicate it may not be the case.

cipher text	accuracy	no. of successful runs	duration (in seconds)
1,000	0.7143	0	0.4441
2,000	0.9312	90	0.4442
5,000	0.8576	22	0.4436
full cipher text	0.9831	97	0.4381

Table 3.2c: Attack substitution cipher by bi-gram.
How the accuracy depends on the size of cipher text used in the attack.
Use the key which maximize the log score. No. iterations = 10,000

We are getting mixed result by using 2000 text from random position of the cipher text for the attack (see table 3.2d). Cipher text in some position are better for decryption.

cipher text	accuracy	no. of successful runs	duration (in seconds)
574798	0.9492	0	0.3906
416031	0.9488	90	0.3933
243158	0.9840	97	0.3932
551774	0.9452	0	0.3940
223511	0.9596	94	0.3939

Table 3.2d: Attack substitution cipher by bi-gram.
How the accuracy depends on the position of cipher text used in the attack.
Use the key which maximize the log score. No. iterations = 10,000
2000 cipher text

3.3 Tri-gram Attack

We modify the previous MCMC algorithm to use tri-grams (triple letters frequencies) to calculate the score function.

$$\pi(x) = \prod_{\beta_1,\beta_2,\beta_3} f_r(\beta_1,\beta_2,\beta_3)^{f_d(\beta_1,\beta_2,\beta_3)} \tag{2}$$

Where β_1, β_1 and β_3 are all possible combinations of alphabets. f_r and f_d are triple letter frequencies of the reference text and the decrypted text. The attack also works but the result is not as good as the attack using the bi-grams (Table 3.3).

iterations	accuracy	no. of successful runs	duration (in seconds)
1,000	0.5336	1	0.9089
2,000	0.7652	46	1.6892
5,000	0.7896	75	4.0435
10,000	0.8920	87	7.9283

Table 3.3: Attack substitution cipher by tri-gram.
How the accuracy depends on the no. of iterations.
Use the last key.

3.4 Attack for Substitution Cipher - Final Version

The optimal values for the parameters are listed in table 3.4a.

The final algorithm to attack the substitution cipher is:

- randomly select 2000 cipher text from available cipher text
- decrypt the selected cipher text with pair letter frequencies (using optimal parameter value)
- apply the decode function to the full cipher text and calculate the log score for full text
- repeat the above procedure 5 times, the final key is the key which gives the highest log score

To investigate how our program work. We apply the method to different cipher text and reference text. The result from table 3.4b shows a very good result.

parameter	optimal value
cipher text length	2,000
scaling parameter	1
no. of iterations	10,000

Table 3.4a: Optimal values for parameters

cipher text	reference text	accuracy	no. of successful runs
Oliver Twist	*War and Peace*	1.0000	100
Pride and Prejudice	*War and Peace*	1.0000	100
Ice Hockey (Wikipedia Page)	*War and Peace*	1.0000	100
Pride and Prejudice	*Oliver Twist*	1.0000	100
War and Peace	*Oliver Twist*	0.9977	97
Ice Hockey (Wikipedia Page)	*Oliver Twist*	0.9869	83
War and Peace	*Pride and Prejudice*	0.9977	97
Ice Hockey	*Pride and Prejudice*	0.9977	97
Oliver Twist	*Pride and Prejudice*	0.9985	98
Pride and Prejudice	*Ice Hockey (Wikipedia Page)*	1.0000	100
War and Peace	*Ice Hockey (Wikipedia Page)*	0.9938	92
Oliver Twist	*Ice Hockey (Wikipedia Page)*	0.9750	74

Table 3.4b: Try our final version on different cipher text and reference text
Key length is 20, 2000 cipher text, scaling parameter is 1.
5 repetitions, each repetition run 10,000 iterations.
Use the key which gives the highest log score.

4 Our Results for Transposition Cipher

4.1 The MCMC algorithm

Transposition Ciphers only move letters around. So we can't use the uni-gram statistics to break it since there is no change to the frequencies of single letters. So we need to use the bi-gram (the frequencies of pairs of letters, e.g. number of occurrence of 'TH' in the text) statistics will work.

The state space depend on the key length of the transposition cipher. For key length p, there are p! possible combinations of keys. e choose the initial decryption key to be '0123...p' so the decrypted text by using this key is identical to the cipher text.

The score function and acceptance rate are the same as the ones from bi-gram attack for the substitution cipher.

4.2 The Proposed Move: Swap Moves v.s. Slide Moves

For the proposal, first we try the same swap moves. But we found the swap moves are not very efficient in some case. E.g. If the plain text is "PROJECT". The current decryption key gives a decrypted text "ROJECTP" which is very close to the correct answer, but we need at least 6 swap moves find the correct decryption key.

So instead of swapping letters, we randomly take one letter out and insert back to a random position of the remaining text. We call this proposal a slide move. For example, if the key size p is 8, the letter at position 3 is taken out and inserted back to position 6 in the remaining text. The text "ABCDEFGH" will becomes "ABCEFGDH" by this slide move.

Sliding moves of a single letter works pretty well with small key length. But as the key length gets larger, it also becomes less efficient. E.g. the plain text is "THE PROJECT ". But the current key generate a decrypted text " PROJECT THE". To get the correct key by single letter slide move, we need at least 3 moves (move each letter in "THE" to the left of " PROJECT"). But each move may lower the score function since we are breaking the word "THE" so it will more likely be rejected by the algorithm.

This can be easily solved by slide move of a block of letters. We select a random block of letters and insert back to the remaining text. For the " PROJECT THE" example, the word "THE" can be moved to the left of "PROJECT" by just one move. The formal definition of the block slide move is given below.

For a key length p. The new proposal is to slide move a block of n letters from position p1 to p2. Then n is taken from Uniform(0, p-2), p1 is taken from Uniform(0, p-n+1) and p2 is taken from Uniform(0, p-n+1)

For example, for transposition cipher with key length 8, we propose to move 2 letters from position 3 to position 6. We set p=8, n=2, p1=3, p2=6, "01234567" will become "01256347" ('34' is moved after '6')

The table 4.2a, 4.2b show the comparison of swap move, single letter slide move and block slide move as the proposals. With key length 10, we see some improvement of the accuracy and success rate by switching from swap move to slide move algorithm.

As the key length increase to 20. The benefit of using slide moves is more significant. With 1,000 iterations, swap moves or single letter slide moves can't get a single success run. But block letter slide move performs very good. It reveals 95% of the cipher text and successfully break the transposition cipher in 90 out of 100 of the runs.

move	accuracy	no. of successful runs
swap move	0.6550	17
single letter slide move	0.9525	83
block letter slide move	0.9587	90

Table 4.2a: Comparison of swap move and slide move for key length 10
1000 iterations, scaling parameter is 1, 1000 cipher text

move	accuracy	no. of successful runs
swap move	0.4383	0
single letter slide move	0.6333	0
block letter slide move	0.8961	59

Table 4.2b: Comparison of swap move and slide move for key length 20
1000 iterations, scaling parameter is 1, 1000 cipher text

4.3 Looking for the best scaling parameter

But we found the best scaling parameter for the mcmc algorithm to break a transposition cipher is around 0.44 when the scaling parameter is greater than 1 (table 4.3). We choose the scaling parameter to be 1 because it gives the highest accuracy and produce highest successful runs.

power	acceptance rate	accuracy	no. of successful runs
0.01	0.975	0.013	0
0.1	0.690	0.084	0
1	0.439	0.9694	87
10	0.437	0.9389	77
1,000	0.431	0.9489	81
100,000	0.440	0.9489	81

Table 4.3: Looking for the best scaling parameter
Key length is 20, 1,000 iterations, use 1000 cipher text

4.4 Remember the best log score

For substitution cipher, we improved the algorithm by remembering the key which gives the best log score. But we found we don't need that for breaking the transposition cipher. Because with the right scaling parameter it's very hard for the chain to jump out of the stationary distribution (the correct answer). Our experiments confirmed this. In each of the 100 runs, the last key is the key which gives the best log score. But we still remember the best log score because it doesn't give much overhead of the algorithm and it guarantee we will use the key which will gives the best result.

4.5 How much cipher text is enough, is more cipher text better?

The table 4.4a investigate if increase the length of the cipher text will increase the accuracy and the success rate. We need at least 500 cipher text to break transposition cipher with key length 20 in 10,000 iterations. It looks like the 2000 cipher text is the best choice since more cipher text will need more time to process but the benefit of using more cipher text is very marginal.

cipher text length	accuracy	no. of successful runs	duration (in seconds)
100	0.0789	0	0.4640
200	0.2844	0	0.5093
500	0.9250	72	0.6526
1,000	0.9694	87	0.8856
2,000	0.9933	95	1.3500
5,000	0.9917	96	2.7557
10,000	0.9861	92	5.079

Table 4.4a: Does more cipher text gives more chance to break the cipher?
Key length is 20, scaling parameter is 1. 10,000 iterations

We also find out different part of the cipher text gives similar results (Table 4.4b).

cipher text	accuracy	no. of successful runs
830080	0.9917	97
254640	0.9933	96
568780	0.9906	96
634220	0.9972	98
366660	0.9928	96

Table 4.4b: Try to use different part of the Oliver Twist cipher text
Key length is 20, scaling parameter is 1, 10,000 iterations

4.6 No. of Iterations

With the slide move proposal and optimized scaling parameter, we would like to see how we can increase the accuracy by running more iterations. So we try the algorithm on 2000 cipher text encrypted by a key with length 20. Table 4.6 shows the accuracy and succuss rate increase steadily as we run longer iterations. Our algorithm is able to successfully break the transposition cipher with sufficient cipher text (2000) after 20,000 iterations.

no. of iterations	accuracy	no. of successful runs
1,000	0.6878	3
2,000	0.7944	17
5,000	0.9544	73
10,000	0.9911	95
20,000	1.0000	100
50,000	1.0000	100

Table 4.6: How is the no. of iterations affect the accuracy and no. of successful runs.
Key length is 20, 2000 cipher text, scaling parameter is 1.

4.7 What about we don't have enough cipher text

When we only have 1000 cipher text, it looks like the accuracy won't increase as we run more iterations. (Table 4.7a)

We borrow an idea from the parallel computing. Instead of a long run for 50,000 iterations. We run a 10,000 for 5 times and use the decryption key which gives the highest score. The table 4.7b give the comparison of the 5 runs for 10,000 iterations each and 1 run for 50,000 iterations.

5 small runs are better than one big run. By choosing the best answer from 5 runs, the successful rate is 100%. Because the successful rate for 1 run of 10,000 iterations is 83%. So if we try 5 runs (10,000 iterations per run), the chances not getting an exact answer is only $(1 - 0.87)^5 = 0.0037\%$ which is very low.

4.8 Attacks on Transposition Cipher with Longer Keys

The following table investigate how our attacks perform on transposition cipher with longer keys. We notice the time to break the transposition cipher depends on the no. of iterations. But it does not depend on the length of the key. (Table 4.8)

no. of iterations	accuracy	no. of successful runs
1,000	0.6333	1
2,000	0.7583	12
5,000	0.8961	59
10,000	0.9694	87
20,000	0.9872	95
50,000	0.9856	93

Table 4.7a: With less (1000) cipher text, we can not achieve 100% successful rate by just increasing the no. of iterations.
Key length is 20, 1000 cipher text, scaling parameter is 1.

no. of runs x no. of iterations	accuracy	no. of successful runs
20 x 2,500	0.9972	98
10 x 5,000	1.0000	100
5 x 10,000	1.0000	100
2 x 25,000	1.0000	100
1 x 50,000	0.9856	93

Table 4.7b: With less (1000) cipher text, we can still achieve 100% successful rate by choosing the best answer from 5 small runs.
Key length is 20, 1000 cipher text, scaling parameter is 1.

4.9 Our Final Attack for Transposition Cipher

The optimal value of the parameters are summarized in the table 4.9a.

We proposed the following as our final algorithm for the attack of the transposition cipher.

- Randomly select 2000 cipher text from available cipher text
- Attack the selected cipher text with bi-gram (using optimal parameter values)
- Apply the decryption key to the full cipher text and calculate the log score for full decrypted text
- repeat the above procedure 5 times, the final result is the repetition which gives the highest log score

Note the steps are identical with the final decryption key of the substitution cipher.

To investigate how our program work. We apply the method to different cipher text and reference text. Table 4.9b shows the results are very good.

5 Our Results for Substitution-Transposition Cipher

5.1 A Substitution-Transposition Cipher

Substitution-transposition cipher have 2 keys. The length of the substitution key is 26. The length of the transposition key is variable. The letters are switched then moved around.

key size	no. of iterations	accuracy	no. of successful runs	duration (in seconds)
10	2,000	0.9962	97	0.3424
20	10,000	0.9911	95	1.3736
30	50,000	0.9957	98	6.4034
40	50,000	0.9613	70	6.3969
50	100,000	0.9648	68	12.8497

Table 4.8: Attack transposition ciphers with longer keys
cipher text 2000, scaling parameter is 1

key length	parameter	optimal value
any	cipher text length	2,000
any	scaling parameter	1
10	no. of iterations	2,000
20	no. of iterations	10,000
30	no. of iterations	50,000
40	no. of iterations	50,000
50	no. of iterations	100,000

Table 4.9a: Optimal values for parameters

5.2 Attack for the Substitution-Transposition Cipher

We shall still use the frequency analysis to break the Substitution-Transposition cipher. We hope we can reuse the MCMC attack algorithms for the substitution cipher and the transposition cipher.

Our algorithms to attack the substitution cipher and transposition cipher has several similar characteristics. First they work on the same problem space. i.e. they are applied on the same 27 characters ('A' to 'Z' and ' '). Although they use different proposed moves, they actually use the same score function (by comparing the pair frequencies of the cipher text and the reference text). And the process of decrypting the cipher text is indeed to maximize the score function. The score function is maximized when the correct answer is found. This make us think we might be able to combine our attack algorithms for both ciphers to break the substitution-transposition cipher.

5.3 First Attempt

We try to break a substitution-transposition cipher with transposition key length 10. First we try to run a bi-gram attack on substitution cipher and then a bi-gram attack on transposition cipher.

We use the optimal values of parameters from our previous attacks (Table 3.3a and Table 4.7). 2000 cipher text is used in the attack. We run 10,000 iterations for bi-gram attack on a substitution cipher. Then we run 2,000 iterations for the bi-gram attack on a transposition cipher for key length 10. The result is present in table 5.3. We are not able to get good result by simply increasing the number of iterations. Switching the sequence of the attacks also does not help.

cipher text	reference text	accuracy	no. of successful runs
Oliver Twist	*War and Peace*	1.0000	100
Pride and Prejudice	*War and Peace*	1.0000	100
Ice Hockey (Wikipedia Page)	*War and Peace*	1.0000	100
Pride and Prejudice	*Oliver Twist*	1.0000	100
War and Peace	*Oliver Twist*	1.0000	100
Ice Hockey (Wikipedia Page)	*Oliver Twist*	1.0000	100
War and Peace	*Pride and Prejudice*	1.0000	100
Ice Hockey (Wikipedia Page)	*Pride and Prejudice*	1.0000	100
Oliver Twist	*Pride and Prejudice*	1.0000	100
Pride and Prejudice	*Ice Hockey (Wikipedia Page)*	1.0000	100
War and Peace	*Ice Hockey (Wikipedia Page)*	1.0000	100
Oliver Twist	*Ice Hockey (Wikipedia Page)*	1.0000	100

Table 4.9b: Try our final algorithm for transposition cipher on different cipher text and reference text. key length is 20, 2000 cipher text scaling parameter is 1, 5 repetitions, each repetition run 10,000 iterations. use the key which gives the highest log score.

no. of iterations (substitution/transposition)	accuracy	no. of successful runs
10,000/2,000	0.0600	3
10,000/5,000	0.0587	3
10,000/10,000	0.0338	2

Table 5.3: Attack on a substitution-transposition cipher with transposition key length 10, 2000 cipher text. Bi-gram attack on substitution cipher followed by bi-gram attack on transposition cipher

5.4 Improvement by Using Cycles of Small Runs

In section 4.7 we find sometime several small runs are better than one big run. So instead of big runs for the attack on substitution cipher and the transposition cipher. We run several small cycles. Each cycle we run a bi-gram attack on substitution cipher and a bi-gram attack on transposition cipher and use the result as a start point for next cycle. Table 5.4 shows the result is getting better by increase the no. of cycles. But it is not improving a lot after 3 cycles. We only get 63% accuracy on average and 62 success out of 100 runs even after running 10 cycles.

5.5 Improvement by Running Uni-gram Attack on Substitution Cipher First

Recall in section 3.1 we try the uni-gram attack on the substitution cipher but it only reveal part of the letters from the cipher text. But we can make use of this in the attack for breaking the substitution-transposition cipher. Even though the uni-gram attack does not reveal all the letters, it gives us a very good starting point. We run the uni-gram attack on substitution cipher before running cycles of bi-gram attacks. The results from table 5.5a(transposition key length 10) and table 5.5b(transposition key length 20) are very good.

cycles	accuracy	no. of successful runs	duration(in seconds)
1	0.0600	3	0.7446
2	0.3713	35	1.4919
3	0.5462	52	2.5222
5	0.5250	52	3.6850
10	0.6300	62	7.4400

Table 5.4: Attack on a substitution-transposition cipher with
transposition key length 10, 2000 cipher text, run bi-gram attacks in cycles
10,000 iterations for bi-gram attack on substitution cipher
2,000 iterations for bi-gram attack on transposition cipher

cycles	accuracy	no. of successful runs	duration(in seconds)
1	0.7937	68	0.8369
2	1.0000	100	1.5800

Table 5.5a: Attack on a simple substitution-transposition cipher with
transposition key length 10. 2000 cipher text. Run a uni-gram attack
on substitution cipher first. Then run bi-gram attacks in cycles.
10,000 iterations for bi-gram attack on substitution cipher for each cycle.
2,000 iterations for bi-gram attack on transposition cipher for each cycle.

5.6 Attacks on Substitution-Transposition Cipher with Longer Transposition Key

We combine the above improvements (Run a uni-gram attack on substitution cipher first. Then run bi-gram attacks in cycles.) and try the attack on the substitution-transposition cipher. The results are pretty good even with longer transposition key. (Table 5.6)

5.7 Final Algorithm

So we propose the following algorithm to attack the substitution-transposition cipher.

1. Run the uni-gram attack for substitution cipher on the cipher text, the result is text t_1.
2. Run the bi-gram attack for transposition cipher on text t_1, the result is text t_2.
3. Run the bi-gram attack for substitution cipher on text t_2, the result is text t_3.
4. Let t_1=t_3, run step 2 and 3 for several cycles (the no. of cycles depends on the transposition key length).

We try the final algorithm on substitution-transposition cipher with transposition key length 20 with different combinations of cipher text and reference. The overall results are very good (table 5.7). And we also find the choice of cipher text and reference text do affect the results. We notice that among the 4 texts we choose, only *Ice Hockey (Wikipedia Page)* is a modern text, all others are novels written a hundred years ago. The results from attacks by using *Ice Hockey (Wikipedia Page)* as the reference text or cipher text are worse than using other text.

cycles	accuracy	no. of successful runs	duration(in seconds)
1	0.2394	0	1.8440
2	0.9133	82	3.6105
3	0.9906	98	5.3200

Table 5.5b: Attack on a simple substitution-transposition cipher with transposition key length 20. 2000 cipher text. Run a uni-gram attack on substitution cipher first. Then run bi-gram attacks in cycles. 10,000 iterations for bi-gram attack on substitution cipher for each cycle. 2,000 iterations for bi-gram attack on transposition cipher for each cycle.

key length (subst./trans.)	no. of iterations (subst./trans.)	cycles	accuracy	no. of successful runs	duration
26/10	10,000/2,000	3	1.0000	100	3.93
26/20	10,000/10,000	3	0.8894	86	5.32
26/30	10,000/50,000	5	0.8618	85	34.08
26/40	10,000/100,000	5	0.7645	73	65.51

Table 5.6: Attacks on a simple substitution-transposition cipher
How does our algorithm work for longer transposition key. 2000 cipher text.
Use *Oliver Twist* as the cipher text. Use *War and Peace* as the reference text.

6 Our Attempt for Polygraphic Substitution Cipher

Polygraphic Substitution Cipher is a substitution cipher which operate on blocks of letters. A cipher which substitute pairs of letters are called the digraphic cipher.

Playfair cipher is one example of the digraphic cipher. It uses a 5x5 table for the encryption and decryption. Playfair cipher have an additional rules before doing substitution. It inserts a 'X' between any letters that appears twice. e.g. the text 'teeth' will become 'texeth' and then the pairwise substitution is performed.

We try to extend our algorithm for simple substitution cipher to attack the digraphic cipher. But the attempt is not successful. The frequency analysis does not work very well in this case.

The the frequency distribution of the bi-graphic cipher is much more flat than the one from a simple substitution ciphers. The bi-gram frequency distribution (pairs of letters) for a simple substitution cipher has only 676 entries (26 times 26, from 'A A' TO 'Z Z'). But the frequency distribution of a bi-graphic cipher (pairs of pairs of letters) has 456976 entries (676 times 676, from 'AA AA' to 'ZZ ZZ'). So even with the full *Oliver Twist* novel (863080 letters), the average frequency is only 1. It's even harder to compare the pair letter frequencies for the cipher text and the plain text.

7 Our Thoughts on Attacking DES

DES is a symmetric key algorithm. It perform bit level operations on a block of 64-bit or 8-byte of letters. SDES is a reduced version of the DES with the block size 8-bits or 1-byte. The differential cryptanalysis and linear cryptanalysis algorithms are used to break the DES and SDES (see reference [11] and [12]).

cipher text	reference text	accuracy	no. of successful runs
Oliver Twist	*War and Peace*	0.8894	86
Pride and Prejudice	*War and Peace*	0.8433	80
Ice Hockey (Wikipedia Page)	*War and Peace*	0.6811	58
Pride and Prejudice	*Oliver Twist*	0.9100	89
War and Peace	*Oliver Twist*	0.8367	78
Ice Hockey (Wikipedia Page)	*Oliver Twist*	0.7211	62
War and Peace	*Pride and Prejudice*	0.8189	74
Ice Hockey (Wikipedia Page)	*Pride and Prejudice*	0.6811	56
Oliver Twist	*Pride and Prejudice*	0.8244	81
Pride and Prejudice	*Ice Hockey (Wikipedia Page)*	0.7961	74
War and Peace	*Ice Hockey (Wikipedia Page)*	0.6761	64
Oliver Twist	*Ice Hockey (Wikipedia Page)*	0.7778	71

Table 5.7: Final algorithm on a simple substitution-transposition cipher
Test on different cipher text and reference text
key length 26/20, no. of iterations 10,000/10,000, 2000 cipher text

But we found SDES is just a special case of a simple substitution cipher (substitute 26 alphabetics with 26 1-byte binaries codes). DES is the special case of Polygraphic Substitution Cipher (substitute blocks of 8 alphabetics with 8-byte binary codes). By replacing the 1-byte binary codes with alphabets, SDES can be easily broken by our bi-gram attacks on substitution ciphers. But frequency analysis may not be feasible for breaking DES (As block size become bigger, the frequency distribution become much flatter. Please see section 6 for more details.).

8 Summary

In this paper, we successfully apply the MCMC algorithm to break the substitution cipher and transposition cipher. The attacks are based on the frequency analysis of the cipher text and the reference text.

The uni-gram attack on a simple substitution cipher only reveal part of the cipher text. The bi-gram attack gives good results within 10,000 iterations. We also find with only 2000 cipher text is needed for the bi-gram attack. The tri-gram attack also works but the result is not as good as the bi-gram attack.

We also propose an attack on a simple substitution-transposition cipher by combining uni-gram, bi-gram attack on the substitution cipher and bi-gram attack on the transposition cipher. We are able to break the simple substitution-transposition cipher with transposition key length up to 40.

References

[1] S. Conner (2003), *Simulation and solving substitution codes*, Master's thesis, Department of Statistics, University of Warwick.

[2] Persi Diaconis (2008), *The Markov Chain Monte Carlo Revolution*

[3] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller (1953), *Equations of state calculations by fast computing machines.* J. Chem. Phys. **21**, 1087–1091.

[4] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, ed. (1996), *Markov chain Monte Carlo in practice.* Chapman and Hall, London.

[5] G.O. Roberts and J.S. Rosenthal (2004), *General state space Markov chains and MCMC algorithms.* Prob. Surv. **1**, 20–71.

[6] Bruce Schneier (1996), *Applied Cryptography Second Edition*, John Wiley & Sons

[7] C. E. Shannon (1949), *Communication Theory of Secrecy Systems*, from book "Claude Shannon: Collected Papers", John Wiley and Sons

[8] A.Dimovski, D.Gligoroski (2003), *Attacks on the Transposition Ciphers Using Optimization Heuristics*, ICEST 2003, October 2003.

[9] R. Toemeh, S. Arumugam (2007), *Breaking Transposition Cipher with Genetic Algorithm*, Electronics And Electrical Engineering, ISSN 1392 C 1215 2007. No. 7(79)

[10] Matthew D. Russell, John A. Clark, Susan Stepney, *Making the Most of Two Heuristics: Breaking Transposition Ciphers with Ants.* (no year found in paper)

[11] Dr. K. S. Ooi, Brain Chin Vito (2002), *Cryptanalysis of S-DES*

[12] Poonam Garg (2009), *Cryptanalysis of SDES via Evolutionary Computation Techniques*, IJCSIS, Vol. 1, No. 1, May 2009

[13] Polygraphic Substitution
`http://en.wikipedia.org/wiki/Polygraphic_substitution`

[14] Playfair Cipher
`http://en.wikipediaorg/wiki/Playfair_cipher`

[15] Project Gutenberg
`http://www.gutenberg.org/wiki/Main_Page`

[16] Ice Hockey (Wikipedia Page)
`http://en.wikipedia.org/wiki/Ice_hockey`