*Single Columnar Transposition Cryptanalysis and keyless decryption*

Nikhil M Adyapak  
*Department of Computer Science Engineering*  
*PES University, Bengaluru*, *India*

*Email: nikhiladyapak31@gmail.com*

*Abstract*— Single columnar transposition is an encryption technique used in text encryption to produce cipher text which is a permutation of its original text. In this study, the single columnar transpositions ciphers cryptanalysis and decryption is performed without the use of the encrypting key.

*Keywords—Columnar transposition, longest prefix match, cipher text, plain text, compressed plain text, encryption key, ASCII*

# Introduction

Single columnar transposition [4] [7] is a type of cipher that follows a simple rule for mixing up characters in the plain text form the cipher text. The message is written out in rows of a fixed length, and read out again column by column, and the columns are chosen by some scrambled order. Both the length of the rows and the permutations of the columns are usually defined by a keyword. The columnar transposition requires both the encoder and the decoder to know the keyword when encoding the plain text to cipher text and vice versa. This technique was invented by the ancient Greeks used by the Spartans to send secret messages. This technique was used extensively in the early 1900’s by the Germans during the world war to send encrypted texts to military bases of their allies. The key was shared to all the cipher operators at the beginning of the day and was supposed to be discarded after memorizing it. The columnar transposition is very secure when encoded with long keys (around length 20), but much weaker if shorter keywords are used. If the length of the keyword can be known, by permutations and combinations, the plain text can be obtained by brute force technique.

# Methods of columnar transposition matrix population

There are various methods by which the single columnar transposition matrix could be constructed. There are three methods which are based on the arrangement of plain text inside the matrix. [2][6][7]

The first method of populating the matrix is with the help of an encryption key. The encryption key is a sequence of characters used to encrypt the given plain text. The encryption key is numbered according to alphabetical order of the characters appearance in the English dictionary. For example, the key ‘dcab’ would be translated to ‘4312’ in numbers. The plain text to be used is ‘attack postponed until two am’.

The matrix is constructed using the key translated to a number as individual columns of the matrix. Each individual character in the plain text is inserted into a row under their respective columns as they appear in the sentence.

The cipher text is taken from the matrix and read as the column elements from the columns in increasing order of numbers. In this particular case, the plain text fits perfectly into the matrix with the key length 4 as shown in Fig 1.

The cipher text for the matrix would be ‘tppdiha aooulrm tktette acsnnle’. This is done by reading the column elements from the top to the bottom under columns marked 1 through 4.

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 3 | 1 | 2 |
| a | t | t | a |
| c | k | p | o |
| s | t | p | o |
| n | e | d | u |
| n | t | i | l |
| l | t | h | r |
| e | e | a | m |

Fig. 1. Single columnar transposition matrix population method 1.

The second method of populating the matrix with the help of the encryption key is by the method of padding.[6]

The matrix might not be fully populated with plain text as in method shown in Fig 1. Sometimes, it so happens that the plain text for a particular key might not be fully accommodated in the matrix. In such cases ‘padding’ is required. Padding is used to fill up the matrix with null characters or ‘dummy’ characters. These characters can be any text or ASCII characters.

The plain text used in both the first and second methods is the compressed plain text. For example, if the plain text was ‘attack postponed until two am’, then the compressed plain text would be the plain text without white spaces, that is ‘attackpostponeduntiltwoam’.

The second method can be illustrated with an example by taking the encryption key to be ‘dcabefg’ and the plain text to be ‘attack postponed until two am’ making the compressed plain text to be ‘attackpostponeduntiltwoam’.

The key would translate to the key number ‘4312567’This compressed plain text does not fit into the matrix with the given key as seen in Fig 2.

The padding or the ‘dummy’ characters known as placeholders are therefore put into the matrix to satisfy the matrix’s completeness. The padding characters are ‘X’,’Y’ and ‘Z’ as shown in Fig 2.

For convenience, we take the padding characters to be underscore ‘\_’, so that there is a visible change in the cipher text produced in this second method.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | 3 | 1 | 2 | 5 | 6 | 7 |
| a | t | t | a | c | k | p |
| o | s | t | p | o | n | e |
| d | u | n | t | i | l | t |
| w | o | a | m | X | Y | Z |

Fig. 2. Single columnar transposition matrix population method 2.

The cipher text would be ‘ttna aptm tsuo aodw coi\_ knl\_ pet\_’ in the second method as seen in Fig 2. The cipher text here is again read in the same way as in the first method shown in Fig 1.

The third method of populating the matrix with the help of the encryption key is by the method of padding in addition of populating the matrix with the plain text including white or blank spaces between words.

The third method can be illustrated with an example by taking the encryption key to be ‘dcabefg’ and the plain text to be ‘attack postponed until two am’.

The key would translate to the key number ‘4312567’. The plain text is populated into the matrix as it is with the blank or white spaces. This plain text does not fit into the matrix with the given key as seen in Fig 3. Therefore, there is padding characters introduced similar to the second method.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | 3 | 1 | 2 | 5 | 6 | 7 |
| a | t | t | a | c | k | space |
| p | o | s | t | p | o | n |
| e | d | space | u | n | t | i |
| l | space | t | w | o | space | a |
| m | padding | padding | padding | padding | padding | padding |

Fig. 3. Single columnar transposition matrix population method 3.

Using the third method from the example shown in Fig 3, making the space and padding characters to be white spaces, the cipher text would be ‘ts t atuw tod apelmcpno kot nia’ .

In this study, the second method of populating the matrix is used extensively for method of decrypting the single columnar transposition ciphers without the use of the encryption key. The second method of populating the matrix using compressed plain text and padding characters is shown in Fig 2.

# Decoding columnar transposition

In knowing the length of the cipher and the key permutation ‘4 3 1 2 5 6 7’ example 2 shown in Fig 2 can be converted back to plain text. [7].

By dividing the total length of the cipher by the length of the key, number rows originally present in the matrix while encrypting can be calculated. Here, the number of rows is 4 that can be got from dividing 28 (length of text) by 7 (length of key).

Rearranging the cipher text by taking single letters at a time from the key permutations and representing the same as columns, the plain text is obtained. The cipher text being ‘ttna aptm tsuo aodw coi\_ knl\_ pet\_’.

From the cipher text and the key ‘4 3 1 2 5 6 7’, taking the first letters from element 4 which is ‘a’, element 3 which is ‘t’, element 1 which is ‘t’, element 2 which is ‘a’, element 5 which is ‘c’, element 6 which is ‘k’ element 7 which is ‘p’.

This is repeated row number of times for the next characters in the same order of the key permutation. The key permutation is the order of numbering that can be in any order with the numbers ranging between 1 to N where N stands for the length of the key.

# Cryptanalysis

Cryptanalysis is the study of ciphertext, ciphers and cryptosystems with the aim of understanding how they work and finding and improving techniques for defeating or weakening them.

The key is the most vulnerable part of this cipher because attackers just knowing the key length can decipher the cipher text into plain text by using brute force technique. [5] [8]For a key length of n, the number of possible key permutations is n! (n factorial).

In example 2, dummy characters ‘X’, ‘Y’ and ‘Z’ are added. Since the matrix was not filled completely, the strength of the columnar transposition cipher is increased by adding random dummy characters.

All substitution and transposition ciphers so far has been decoded and there are methods being devised by researchers and cryptography experts in this field where they use and a combination of different ciphers like hill ciphers, play fair cipher, Caeser cipher and different transposition ciphers like columnar transposition, rail fence transposition and double transposition.

The columnar transposition cipher on its own is not that difficult to decode by attackers. In combination with other ciphers, it becomes really difficult to decode the cipher into plain text.

In research papers [3] [4], there is a new bit level approach for encoding and decoding columnar transposition ciphers, combined columnar transposition with hill cipher, and [1] published its research in encoding plain text initially with rail fence or columnar transposition cipher and then applying another layer of security be coding the cipher text again with Caeser cipher.

Columnar transposition is very secure when key lengths above 20 are chosen. This is because to decode the cipher text, there is at least 20! Permutations of keys to be used. [6]

The number of permutations of larger encryption key lengths with its effect is shown in the Fig 4. Combination of ciphers for multiple encryptions along with the encryption key lengths greater than 20 makes it really hard for attackers to decode the cipher.

| **Key Length** | **No. of permutations** | **Examples** |
| --- | --- | --- |
| 2 | 2 | AB, BA |
| 3 | 6 | ABC, BAC, CBA, ... |
| 4 | 24 | ABCD, ABDC, ACBD, ... |
| 5 | 120 | ABCDE, ABCED, ... |
| 6 | 720 | ABCDEF, ABDCFE, ... |
| 7 | 5,040 | ABCDEFG, ABDCGEF, ... |
| 8 | 40,320 | ABCDEFGH, ... |
| 9 | 362,880 | ABCDEFGHI, ... |
| 10 | 3,628,800 | ABCDEFGHIJ, ... |
| 11 | 39,916,800 | ABCDEFGHIJK, ... |
| 12 | 479,001,600 | ABCDEFGHIJKL, ... |

Fig. 4. Key length and number of permutations table

Alternative approaches [8][5] had a few researchers using digraph frequencies from tables to guess the length and predict the way the columns were arranged to get back the plain text. This cryptanalysis study had a solution by anagramming that is making a word or portions of words by rearranging letters.

The researcher’s study was that he had to study the other pairings and then manually apply them as in a trial-and-error way and concluded that they would most probably get the right pairing on the first try. Once a probable pairing was obtained, the same process would be continued using digraph frequencies to select columns to add on to the left or right.

# Keyless decryption using python programming language

Component In this study, python is used as the programming language to implement the key less decryption of the single columnar transposition. Given any plain text converted into cipher text by single columnar transposition by using an encryption key and following the matrix population as done in Fig 2, the python code converts the cipher text back to the plain text without knowing the encryption key or the length of the same.

Not knowing the length of the encryption key can be very hard since the number of permutations increases with increase in the length of the key as show in Fig 4. The idea is to have been given a plain text such that all the words are present in the English language and an encryption key to be given as a set of letters that need not be a legitimate word.

The order or the numbering of the encryption key is noted to encrypt the plain text stored in the encryption matrix as compressed plain text and padding characters if necessary. Once this is encrypted using single columnar transposition, the resultant cipher text would be fed into the system without the encryption key or the key length. The python code deciphers the cipher text back to the plain text as initially entered.

# Working of the python code

The code first computes the length of the document, or the number of characters present in the cipher text. This will be used to predict the shape of the encrypting matrix initially used. If any two integers perfectly divided the size of the cipher text, then those two numbers would be taken as the probable shape of the matrix.

If the size of the cipher text was S and two numbers M and N ‘s product made S, then the matrix would assume the shape of MxN or NxM. In this example, N and M are columns of the two possible initial matrices by which the key length would N or M respectively. [5][8]

By knowing the encryption key length, all the possible permutations are tried out for that key length and all the possible plain texts are generated. It is to be noted that in all those plain texts, there would be only one among them that would be the original plain text. These plain texts generated by the decrypting program would generate them as compressed plain texts and would not make any sense in reading them.

The longest prefix match technique is used to split the compressed plain text into a sentence. For example, the longest prefix match of the compressed plain text, ‘ballooncupboard’, would generate the plain text, ‘balloon cupboard’. The example could have been split as ‘ball on cup board’ or any other combination. Longest prefix match is used to split the compressed plain text into words with the maximum length possible.

Even these words generated might not make complete sense and hence the enchant package in Python programming language is used for identifying whether the words got from the longest prefix match belong to the English dictionary. If any word does not belong to the English dictionary, the permutation of the encryption key that generated the plain text would be rejected.

The next encryption key permutation would be chosen to decrypt the cipher text and this process is continued until either all the words got from the cipher text to form the plain text are in the English dictionary or all the possible permutations have been tested out without a correct result for the combination of either NxM or the MxN matrix. [5][6]

If the combination of M and N that divided the cipher text of length did not generate the correct plain text, a new combination of M and N would be chosen that completely divides the cipher text of length S and the same process would repeat from choosing a key length and trying out all its permutations.

The process is stopped once all the words got from the longest prefix match of the possible compressed plain text all belong to the English dictionary. This plain text would indeed be the initial plain text before it was encrypted by the single columnar transposition.

# Assumptions and enhancements possible

In the study, the encryption key lengths decryptable is restricted to only 9 because of the execution time being high. However, this can be improved through optimizing the code to support multi-threading and using of powerful multi-core processors with high processing capability can decrypt even ciphers with unknown keys of key lengths of up to 20. [6]

The assumptions are that the plain text before encryption consists of legitimate words from a particular language because if words are mis-spelt, then the decryption would result in either a wrong or no output. In the study, for a given key, if the plain text does not fit into the matrix, then the plain text would be padded with dummy characters. For only the study purposes, the dummy characters are assumed to be white spaces.

This can however be replaced by any ASCII characters to increase the strength of the cipher. The keyless decryption can be further expanded to other languages and not restricted only to the English language. The padding or the ‘dummy’ characters can be made to dynamically change or be different for every single instance making the cipher even stronger and harder to decrypt without the encryption key.

# Technologies used

The computer on which the python code was run had an Intel Core i5 7th gen processor. The software requirements for the study were the Windows 10 operating system. The coding language used was Python (version 3.8.8) programming language. The tools used was Pycharm IDE (2019 community edition) to run the python code. The packages installed through python were Pandas, numpy, cryptography, enchant, itertools python packages.

# Encryption and keyless decryption code

The code for this study can be found at <https://github.com/NikhilAdyapak/SingleColumnarTransposition> .

# Acknowledgment

I would like to thank my project guide Prof. Vineetha B.

# References

1. Jawad Ahmad Dar and Amit Verma, “A Novel Approach for Encryption of Text Messages, Analysis and Implementation of Simple Columnar Transposition Cipher with Ceasar Cipher and Rail Fence Cipher in C/C++”, International Journal of Science and Research, ISSN (O): 2319-7064, Volume 4 Issue 4, April 2015.
2. Sumathy Kingslin and R.Saranya, “Evaluative Study on Substitution and Transposition Ciphers” , International Journal of Creative Research Thoughts, ISSN: 2320-2882, Volume 6, Issue 1 January 2018.
3. Sayantan Majumdar, Abhisek Maiti Biswarup Bhattacharyya and Asoke Nath, “A New Bit-level Columnar Transposition Encryption Algorithm”, International Journal of Advance Research in Computer Science and Management Studies, ISSN: 2321-7782, Volume 3, Issue 7, July 2015.
4. Columnar Transportation:

<https://studydriver.com/columnar-transportation/> .

1. Cryptanalysis of columnar transpositions:

<https://www.uni-mainz.de/> .

1. Single Columnar Transposition:

<http://practicalcryptography.com/> .

1. Transposition ciphers:

<https://wikipedia.org/> .

1. Chris Christensen, “Cryptanalysis of the columnar transposition cipher”, unpublished.