Classical Transposition Ciphers

Introduction

Up until now, we have looked at cryptographic protocols that change letters to other letters to obscure the meaning of a message. But, the same goal can potentially be achieved by simply reordering the letters in a message in such a way to render the message unreadable without knowing how to reorder the letters. (This is basically the definition to transposition; to reorder the letters of a message in some sort of methodical way.) Typically we can solve small anagrams, such as those presented in the Jumble section of the paper, but anagrams of hundreds of letters are more difficult to face.

A very simple system was used by the North during the Civil War that involved the transposition of entire words. The following plaintext message:

We	will	attack	at	dawn	with	the
first	division	on	the	right	and	the
second	division	on	the	left	surprise	is
important	so	make	all	preparations	under	the
cover	of	darkness	nulls	fill	this	up

would become

We first second important cover will division division so of attack on on make darkness at the the all nulls dawn right left preparations fill with and surprise under this the the is the up.

To slightly improve the security of the code above, we could substitute code words for important words like "attack."

However, this is about as weak as a transposition code could be.

The type of transposition shown above is a polygraphic transposition while one that reorders individual letters is a monographic transposition. A standard transposition is done by writing the elements (letters or words) in order from left to right, top to bottom in a grid format, and then reading them down the columns instead. Roughly speaking, rows and columns are switched. Here is a quick example with the plaintext: are you going to the art museum on Thursday night?

Write it out in a grid like this, with a key of 7:

```
A R E Y O U G
O I N G T O T
H E A R T M U
S E U M N T H
U R S D A Y N
I G H T X X X
```
, (Fill in the X's to fill up the last row of the grid…)

Also, note that it's possible to properly encrypt with transposition WITHOUT filling in the last few letters. You can do some modular arithmetic to figure out which columns are "full" and which ones aren't, when decrypting.

Now, to encrypt the message, just read down each column instead of each row. (You can do it in different ways, by reading the columns in a predetermined order, but for this example we'll just go left to right.)

AOHSUIRIEERGENAUSHYGRMDTOTTNAXUOMTYXGTUHNX

Now, if you know that there are seven columns, then you can take the total length of the message, 42, and divide it by seven to know that there are six rows. Then, just take the message and write it out by writing down each column in order from left to right. When you are done, you can just read the message by reading each row in order from top to bottom. (Alternatively, as previously mentioned, if the number is not evenly divisible, you can do some modular arithmetic to figure out the length of each column.)

Railfence Cipher

But, there are a couple other possibilities, like the rail-fence cipher or a triangle from transposition. (In the rail-fence you write the plaintext in zig-zag columns and read off the ciphertext in rows:

```
T   I   E
 H S S T S
  I   A   T, is the plaintext THISISATEST.

The ciphertext is TIEHSSTSIAT.
```

Triangular Transposition

In a triangle form transposition, you write the message in a triangular form, one row at a time and then read off the ciphertext, column by column:

```
   Y
  OUM
 USTDO
THATNOW, is the plaintext YOUMUSTDOTHATNOW

The ciphertext is TUHOSAYUTTMDNOOW.
```

Permutation Cipher

In this cipher, you pick a key length, n, and then a permutation of the numbers 1, 2, …, n. For example, for n=6, a valid permutation would be 5, 3, 6, 2, 1, 4.

For this key, the plaintext message

```
LUKEIA MYOURF ATHERZ
```

would be encrypted as:

```
IKAULE ROFYMU RHZTAE
```

In each block of 6, we place the 5th plaintext character 1st in the ciphertext, the 3rd plaintext character 2nd in the ciphertext, the 6th plaintext character 3rd in the ciphertext, etc.

To break this type of cipher, one must try different guesses of the keysize and write out the ciphertext in the appropriate columns. Then one should try to see if permuting the columns together will form rows that make sense. In our example above, we'd line up the ciphertext as follows:

```
I K A U L E
R O F Y M U
R H Z T A E
```

Seeing the Z in column 3 indicates that this column should go to the end:

```
I K U L E A
R O Y M U F
R H T A E Z
```

From here you try finding some sort of anagram for the first row (or any other row for that matter). If you pick out "LUKE", you get it:

```
L U K E I A
M Y O U R F
A T H E R Z
```

Column Permutation Cipher

A column permutation cipher is very similar to the generic transposition cipher described in the introduction. You write down the message in a grid, and then read off the cipher text in columns instead of rows. But, in the column permutation cipher, you don't go in order left to right. Instead, the key, a permutation of the integers 1 through n where n is the number of columns dictates which order to read the columns for the cipher text. Using our previous example, with the keyword "PAINTER" which corresponds to a key of 5,1,3,4,7,2,6, since P is the 5th letter alphabetically in the word, A is the 1st letter alphabetically in the work, I is the 3rd letter alphabetically in the word, etc. (Note: You need not have a keyword. Instead you can choose any permutation you like. A keyword makes it easier to remember a particular permutation, but perhaps slightly compromises the security of the cipher.)

```
5 1 3 4 7 2 6
A R E Y O U G
O I N G T O T
H E A R T M U
S E U M N T H
U R S D A Y N
I G H T X X X
```

The corresponding ciphertext is:
RIEERGUOMTYXENAUSHYGRMDTAOHSUIGTUHNXOTTNAX.

In cryptanalyzing the column permutation cipher you should count the total number of characters in the ciphertext. It follows that the number of columns used in the key is a divisor of the total number of characters. This reduces the search space for the key length.

To determine if a guess for the length is likely to be correct, we can utilize the following property of the English language:

About 40% of letters in any stretch of English text are vowels. (This obviously doesn't hold for a sample of letters taken from every fifth letter in English text, for example.) What we can do is make a guess at this length, and then for each row, calculate the percentage of vowels. We would each row to come relatively close to 40%. Try different numbers of columns and see which gives the closest match based on this criteria. In the example above, consider guessing 6 columns. Then we'd write down the following:

```
1 2 3 4 5 6    Number of Vowels
-------------------------------
R O A M U X    3
I M U D I O    4
E T S T G T    1
E Y H A T T    2
R X Y O U N    2
G E G H H A    2
U N R S N X    1
```

The two rows (row 3 and row 7) with one vowel in them only are slightly suspicious. When we look in those rows it seems like it will be difficult to find some anagram of those letter that would form a word. Also, we notice that the Xs are not all in the same bottom row. It seems like these should line up there. (This is why it's better to have regular English letters as null characters…) So 6 is not the correct number of columns. Now try seven:

```
1 2 3 4 5 6 7  Number of Vowels
------------------------------
R U E Y A G O        4
I O N G O T T        4
E M A R H U T        3
E T U M S H N        2
R Y S D U N A        2
G X H T I X X        1
```

Although the last row has only one vowel, we can see it also has 3 Xs, so it's likely that our guess is correct! The three Xs are the null characters, so it makes sense that we only have one vowel in the four real characters on this row. Furthermore, we know that the X's map to the last three columns.

Try out different possibilities to find that the first two words are "are you" and once you determine this, switching the necessary columns will reveal the plaintext.

<u>Double Transposition Cipher</u>
Here we take the original plaintext P and encipher it using a column transposition with one keyword creating an intermediate ciphertext C'. Then we will encipher C' using a second keyword in a column transposition creating the final ciphertext C. It is not necessary for the two keywords to be of the same length. If necessary, we can pad C' with null characters so that it becomes the appropriate length. In decrypting, these nulls would have to be deleted after properly decrypting the first step.

Here is an example of double transposition:

Keyword #1: HOMES Keyword #2: PICNIC

Plaintext: ITBETTERSTOPRAININGBEFORETHEGAME

First, store the plaintext in an array with 5 columns:

1	3	2	0	4	
H	**O**	**M**	**E**	**S**	**= Keyword #1**
I	T	B	E	T	
T	E	R	S	T	
O	P	R	A	I	
N	I	N	G	B	
E	F	O	R	E	
T	H	E	G	A	
M	E	X	X	X	

Now, to get the intermediate ciphertext, read off the columns in the designated order, based on the keyword:

Intermediate Ciphertext: ESAGRGXITONETMBRPNOEXTEPIFHETTIBEAX

Now, copy these into an array with 6 columns:

5	2	0	4	3	1	
P	**I**	**C**	**N**	**I**	**C**	**= Keyword #2**
E	S	A	G	R	G	
X	I	T	O	N	E	
T	M	B	R	R	N	
O	E	X	T	E	P	
I	F	H	E	T	T	
I	B	E	A	X	X	

To get the final ciphertext, read these off of the columns in the designated order.

Final Ciphertext: ATBXHEGENPTXSIMEFBRNRETXGORTEAEXTOII

To cryptanalyze the double transposition cipher, collect several ciphertexts of the same length and line them up, one right underneath the other. Then attempt to permute the columns in such a way that all of the rows make sense. In doing this, it still makes sense to try to utilize the information about the percentage of vowels in a piece of English, though since there are many more letters to permute, the task is most definitely much more difficult than breaking a single column transposition.

Also, the book does not go through this analysis, but it might be useful in a regular column permutation to realize which indexes are possible for a letter to be transposed to.

In the preceding example, consider 'G', which is in index 18 of the original message. This is row 3, column 3. (Note that we are assuming that row 0, column 0 is where the first letter of the message is.) What we'd like to do is determine where this G will end up after the column permutation. Basically, we will be switching row and column. The old column value (which is 3), along with the permutation function will tell us when we will "read" that column. In our example above, the permutation function is described by the word "HOMES" as follows:

Function: $P_1 = \begin{array}{ccccc} \underline{1} & \underline{3} & \underline{2} & \underline{0} & \underline{4} \\ H & O & M & E & S \end{array}$ $P(0) = 1$ $P(1) = 3$ $P(2) = 2$ $P(3) = 0$ $P(4) = 4$

Thus, to determine WHEN the row with 'G' will be read, we need the column that it is currently in, which is 3, and we actually want P(3), which gives us 0, meaning that it is the first column "read" into the ciphertext. (If it weren't 0, we'd have to multiply the number of the column to be read off by the number of values in a column. The number of values in a column is the message length divided by the number of columns.) Then, we must add how far down the column the letter appears. This is simply the old row. So, to determine the index of the placement of the letter 'G' in the ciphertext, we compute 0x(35/5) + 3 = 3. In general, here is the computation:

$$\textbf{New index} = \underset{\substack{\text{Column in which} \\ \text{This letter will be}}}{\mathbf{P(i \% K)}} \, \underset{\substack{\text{\# of letters} \\ \text{in column}}}{\mathbf{(L / K)}} + \mathbf{(i / K)}$$

where i is the index of the letter in the plaintext, K is the length of the key, and L is the length of the message. We note that if we know the length of the keyword, then there are only K possibilities of where any plaintext letter can end up in the ciphertext.