[Permutation Cipher]

A Permutation cipher is another form of transposition cipher. In several ways, it is similar to columnar transposition in that the columns are written in the same way, including how to use keyword. However, instead of the whole ciphertext, the permutation cipher operates on a letter block. Permutation encryption scheme in the encryption scheme in cryptography is about rearranging a set of elements.

A simple permutation can be written as below.

$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$$

This tells us that the first element moves to the third position, the second element moves to the first position, and the third element moves to the second position. If the sets bigger, we apply the permutation to the smaller bits.

For example, set {a, b, c, d, e, f} becomes {b, c, a, e, f, d}.

To encrypt, we choose a keyword then split the plaintext into blocks that are the same length as the keyword.

For instance, if there is a 'iamstudyingcryptography' plaintext and if we use the simple permutation that we mentioned earlier, the first things we need to do is dividing the plaintext into 3 letter blocks; Iam stu dyi ngc ryp tog rap hyx. As you see the last block is padded with 'x'. Then we rearrange according to the permutation. We will obtain 'amitusyidgcnyprogtapryxh' as our ciphertext. As we have seen the ciphertext we just obtained, we should keep in mind that the distribution of the frequency of the alphabet between plaintext and ciphertext remains the same. Because of this permutation cipher can be easily detected by the cryptanalyst by doing a frequency count. To avoid weakness, we often used permutation cipher with other encryption schemes such as a substitution cipher.

To decrypt, we just need to apply the inverse of the permutation. We can start divide ciphertext into 3 letter blocks just like we did it for encryption; 'ami tus yid gcn ypr ogt apr yxh'. Now we can reorder the letters of the keyword to form the actual keyword. We also can reorder the blocks of ciphertext in the same way. We can retrieve the same plaintext which is 'iamstudyingcryptography'.

Also, We would like to introduce columnar transposition. Columnar transposition edit in a columnar transposition, the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both widths of rows and permutation of the columns are commonly defined by a keyword. The word ZEBRAS, for instance, is 6 lengths (so the rows are 6 lengths), and the permutation is determined by the alphabetical order of the letters in the keyword. In this case, the order would be "6 3 2 4 1 5". In a regular columnar transposition cipher, any spare space will be filled with nulls; in an irregular columnar transposition cipher, the spaces are left blank. Finally, the message is read off in columns, in the order specified by the keyword. For example, suppose we use the keyword ZEBRAS and the message WE ARE DISCOVERED. FLEE AT ONCE. In a regular columnar transposition, we write this into the grid as:

```
6 3 2 4 1 5
W E A R E D
I S C O V E
R E D F L E
E A T O N C
E Q K J E U
```

Note that provided five nulls (QKJEU) at the end. The ciphertext is then read off as:

EVLNE ACDTK ESEAQ ROFOJ DEECU WIREE
In the irregular case, the columns are not completed by nulls:

```
6 3 2 4 1 5
W E A R E D
I S C O V E
R E D F L E
E A T O N C
E
```

This results in the following ciphertext: EVLNA CDTES EAROF ODEEC WIREE
To decipher it, the recipient has to work out the column lengths by dividing the message length by the key length. Then he can write the message out in columns again, then re-order the columns by reforming the keyword.

Theoretically, all transposition ciphers are a type of permutation cipher where the length of the key is the same as the length of the plaintext. This is a rather meaningless generalization, and it is almost always easier to find some other rule to describe the transposition than the rather cumbersome permutation that would be required in actual pracitce. We can also perform permutation several times. Some permutations will return the plaintext when done twice. This is clear when we reverse the permutation, as we get the same permutation as we started with. The way to break a permutation cipher is to try out different permutations until we find a first block that makes sense. Once we have worked out the permutation, we can then use this to decrypt the rest of the intercept easily. We will discuss breaking permutation cipher next chapter in detail.

[Cryptanalysis Approaches for Permutation Ciphers]

As we discussed earlier, permutation ciphers do not affect the frequency of individual symbols. Simple transposition can be easily detected by the cryptanalyst by doing a frequency count. It is most likely a transposition if the frequency distribution of ciphertext displays very similar to plaintext. This can often be attacked by anagramming that sliding pieces of ciphertext around then looking for sections that look like anagrams of English words, and solving the anagrams. Once such anagrams have been found they reveal information about the pattern of transposition. And this can be extended as a consequence. Simpler transpositions also often suffer from the property that will reveal long sections of legible plaintext interspersed by gibberish with keys very close to the correct key. There are several ways to break permutation ciphers.

First, we want to introduce brute-force attacks. By using this method, we can simply try out all the possible combinations of characters. Alternatively, the attacker can attempt to guess the key which is typically created from the password using a key derivation function. This is known as an exhaustive key search. In general, the adversary uses a high-performance computer that conducts a lot of calculations per second and can thus evaluate a large number of variations as fast as possible. For example, this method is successfully deployed for short ciphertext because it will be greatly decreasing the number of possible combinations and making it easier to guess. But for longer ciphertext or keys make it too long to decrypt it since there are more possible values, making them exponentially more difficult to break than the shorter ones.

The second one we want to introduce is the Genetic Algorithm. To overcome the disadvantage of brute-force attack high computational complexity, a genetic algorithm is often used. This is a general method of solving problems for which there is no obvious solution that exists and based on Darwinian evolution theory that the principle of emulating the evolution of a species in nature, such that aspects of natural evolution are approximately equivalent to the different components of the algorithm. To simulate Darwinian survival of the fittest some representation of the fitness of the individuals must be generated.

The genetic algorithm contains three operators - selection, crossover, and mutation.

Selection is the process of choosing parents from the population for mating. This method randomly selects chromosomes out of the population according to its evaluation function. The higher the fitness function, the more chance an individual has to be chosen. The pressure of selection is defined as the degree to which the better individuals are favored. Therefore the higher the selection pressure, the more the better individuals are favored. The genetic algorithm is motivated by this selection pressure to enhance population fitness over successive generations.

Crossover is the process that taking two parent solutions and producing two children from them. The population is enriched with better individuals through the selection process.

The mutation is used to randomly select a portion of an individual to produce a new individual.

Following is the general flow of the genetic algorithm.

1. Initialize algorithm variables: G the maximum number of generations to consider, M the solution pool size, and any other problem-dependent variables.
2. Generate an initial solution pool containing M candidate solutions.
3. For G iterations, using the current pool:

- Select a breeding pool from the current solution pool and make pairings of parents.
- For each parental pairing, generate a pair of children using a suitable mating function.
- Apply a mutation operation to each of the newly created children.
- Evaluate the fitness function for each of the children.

- Based on the fitness of each of the children and the fitness of each of the solutions in the current pool, decide which solutions will be placed in the new solution pool. Copy the chosen solutions into the new solution pool.
- Replace the current solution pool with the new one. So, the new solution pool becomes the current one.

4. Choose the fittest solution of the final generation as the best solution.

Note that while we use a genetic algorithm, fitness measure is used to compare candidate keys is to compare n-gram statistics of the decrypted message with those of the language.

In conclusion, since the number of possible keys for permutation cipher with N key size is N!, the time to break the key is less than the time in the brute-force attack.

[Reference]

- Breaking Transposition Cipher with Genetic Algorithm[ISSN 1392 – 1215]
- USING GENETIC ALGORITHMS TO BREAK A SIMPLE[ICIT 2013 The 6th International Conference on Information Technology]
- Attacks on the Transposition Ciphers Using Optimization Heuristics[in Proceedings of ICEST 2003]
- https://en.wikipedia.org/wiki/Transposition_cipher
- https://en.wikipedia.org/wiki/Brute-force_attack
- https://crypto.interactive-maths.com/
- https://www.password-depot.de/en/know-how/brute-force-attacks.htm