Automated Cryptanalysis of Classical Ciphers

Otokar Grošek
Slovak University of Technology, Slovakia

Pavol Zajac
Slovak University of Technology, Slovakia

INTRODUCTION

Classical ciphers are used to encrypt plaintext messages written in a natural language in such a way that they are readable for sender or intended recipient only. Many classical ciphers can be broken by brute-force search through the key-space. Methods of artificial intelligence, such as optimization heuristics, can be used to narrow the search space, to speed-up text processing and text recognition in the cryptanalytic process. Here we present a broad overview of different AI techniques usable in cryptanalysis of classical ciphers. Specific methods to effectively recognize the correctly decrypted text among many possible decrypts are discussed in the next part Automated cryptanalysis – Language processing.

BACKGROUND

Cryptanalysis can be seen as an effort to translate a ciphertext (an encrypted text) to a human language. Cryptanalysis can thus be related to the computational linguistics. This area originated with efforts in the United States in the 1950s to have computers automatically translate texts from foreign languages into English, particularly Russian scientific journals. Nowadays it is a field of study devoted to developing algorithms and software for intelligently processing language data. Systematic (public) efforts to automate cryptanalysis using computers can be traced to first papers written in late ’70s (see e.g. Schatz, 1977). However, the research area has still many open problems, closely connected to an area of Artificial Intelligence. It can be concluded from the current state-of-the-art, that although computers are very useful in many cryptanalytic tasks, a human intelligence is still essential in complete cryptanalysis.

For convenience of a reader we recall some basic notions from cryptography. Very thorough survey of classical ciphers is written by Kahn (1974). A message to be encrypted (plaintext) is written in the lowercase alphabet $\mathcal{P}$= {a, b, c… x, y, z}. The encrypted message (ciphertext) is written in uppercase alphabet $\mathcal{C}$= {A, B, C… X, Y, Z}. Different alphabets are used in order to better distinguish plaintext and ciphertext, respectively. In fact these alphabets are the same.

There is a reversible encryption rule (algorithm) how to transform the plaintext to the ciphertext, and vice-versa. These algorithms depend on a secret parameter K called the key. The set of possible keys $\mathcal{K}$ is called the key-space. Input and output of these algorithms is a string of letters from respective alphabets, $\mathcal{P}^*$ and $\mathcal{C}^*$. Both, sender as well as receiver, uses the same secret key, and the same encryption and decryption algorithms.

There are three basic classical systems to encrypt a message, namely a substitution, a transposition, and a running key. In a substitution cipher a string of letters is replaced by another string of letters using prescribed substitution of single letters, e.g. left ‘a’ to ‘A’, replacing letter ‘b’ by letter ‘N’, letter ‘c’ by letter ‘G’, etc. A transposition cipher rearranges order of letters according to a secret key K. Unlike substitution ciphers the frequency of letters in the plaintext and ciphertext remains the same. This characteristic is used in recognizing that the text was encrypted by some transposition cipher. A typical running key cipher is to derive from a main key K the running key $K_0 K_1 K_2 ... K_n$. If $\mathcal{P} = \mathcal{C} = \mathcal{K}$ is a group, then simply $y_i = e_K(x_i) = x_i + K_i$.

Thus it is convenient to define a ciphering algorithm for classical ciphers as follows:

Definition 1: A classical cipher system is a five-tuple $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$, where the following conditions are satisfied:

1. $\mathcal{P}$ is a finite set of a plaintext alphabet, and $\mathcal{P}^*$ the set of all finite strings of symbols from $\mathcal{P}$.
2. $\mathcal{C}$ is a finite set of a ciphertext alphabet, and $\mathcal{C}^*$ the set of all finite strings of symbols from $\mathcal{C}$..
3. $\mathcal{K}$ is a finite set of possible keys.
4. For each $K \in \mathcal{K}$, there is an encryption algorithm $e_K \in \mathcal{E}$, and a corresponding decryption algorithm $d_K \in \mathcal{D}$ such that $d_K(e_K(x)) = x$ for every input $x \in \mathcal{P}$ and $K \in \mathcal{K}$.
5. The ciphering algorithm assigns to any finite string $x_0 x_1 x_2 ... x_n$ from $\mathcal{P}^*$ the resulting ciphertext string $y_0 y_1 y_2 ... y_n$ from $\mathcal{C}^*$, where $y_i = e_K(x_i)$. The actual key may, or need not depend on the index i.

Another typical case for $\mathcal{P}$, and $\mathcal{C}$, are r-tuples of the Latin alphabet. For transposition ciphers, the key is periodically repeated for r-tuples. For substitution ciphers of r-tuples, the key is an r-tuple of keys. In the case of running keys, there is another key stream generator g: $\mathcal{K} \times \mathcal{P} \rightarrow \mathcal{K}$ which generates from the initial key K, and possibly from the plaintext $x_0 x_1 x_2 ... x_{n-1}$ the actual key K_n.

For classical ciphers, there are two typical situations when we try to recover the plaintext:

1. Let the input to decryption algorithm $d_K \in \mathcal{D}$ with unknown key K be a ciphertext string $y_0 y_1 y_2 ... y_n$ from $\mathcal{C}^*$, where $y_i = e_K(x_i)$. Our aim is to find the plaintext string $x_0 x_1 x_2 ... x_n$ from $\mathcal{P}^*$. Thus in each execution an algorithm is searching through Key-space $\mathcal{K}$,
2. The decryption algorithm $d_K \in \mathcal{D}$ and key K are unknown. Our aim is to find for the ciphertext string $y_0 y_1 y_2 ... y_n$ from $\mathcal{C}^*$, where $y_i = e_K(x_i)$, the plaintext string $x_0 x_1 x_2 ... x_n$ from $\mathcal{P}^*$. This requires a different algorithm than the actual $d_K \in \mathcal{D}$, as well as some additional information. Usually there is available another ciphertext, say $z_0 z_1 z_2 ... z_n$ from $\mathcal{C}^*$. Thus in each execution an algorithm is searching through possible substitutions which are suitable for both ciphertexts.

In both cases we need a plaintext recognition subroutine which evaluates a candidate substring of length v for a possible plaintext, say $c_t c_{1+t} c_{2+t} ... c_{v+t} := x_t x_{1+t} x_{2+t} ... x_{v+t}$. Such automated text recognition needs an adequate model of a used language.

AUTOMATED CRYPTANALYSIS

There are two straightforward methods for automated cryptanalysis. Unfortunately none of them is for longer strings applicable in practice. The first one is for transposition ciphers. When no other information about the cipher is known, we can use a general method, called anagramming, to decipher the message. In this method we are trying to assemble the meaningful string (anagram) from the ciphertext. This is accomplished by arranging the letters to words from the dictionary. When we find the meaningful word we process the rest of the message in the same way. When we are not able to create more meaningful words, we retrace our steps, and try other possible words until the whole meaningful anagram is found.

The second, and very similar, is for the substitution ciphers. Here we are trying to assemble the meaningful string (anagram) from the ciphertext by searching through all possible substitutions of letters to get words from dictionary of the used language. Although the size of the key-space is large, automated cryptanalysis uses many other methods based, e.g. on frequency distribution of letters. Automated cryptanalysis of simple substitution ciphers can decrypt most of the messages both with known word boundaries (Carrol & Martin, 1986), and without this information (Ramesh, Athithan & Thiruvengadam, 1993; Jakobsen, 1995). There are other classical ciphers, where transposition or substitution depends not only on the actual key, but also on a position within a block of letters of the string.

For effective automated cryptanalysis at least two layers of plaintext candidate processing, filtering and scoring, are required. Better results are achieved by additional filtering layers. This of course increases computational complexity. Bellow we give an overview of these filtering layers.

Automated Brute Force Attacks

The basic type of algorithm suitable for automated cryptanalysis is a brute force attack. As we have to search the whole key-space, this attack is only feasible when key-space is "not too large". Exact quantification of the searchable key-space depends on computational resources available to an attacker, and the average time needed to verify a candidate for decrypted text. Thus, the plaintext recognition is the most critical part of the algorithm from the performance point of view.