9 Pattern Search

Word Lists

The second basic approach to cryptanalysis of the monoalphabetic substitution is the search for patterns in the ciphertext that correspond to the patterns of

- supposed words (probable words),
- words from a list.

This method is cumbersome if done by hand but easy with computer support that completely searches lists of several 100000 words in a few seconds.

Searching for a probable word is a variant of pattern search. We search for the pattern of a word that we suspect from knowledge of the context as occuring in the plaintext.

Numerical Patterns for Strings

To normalize letter patterns we describe them by numbers. Here is an example: The word "`statistics`" defines the pattern `1232412451`. The general procedure is: Replace the first letter by 1. Then replace each following letter by

- the number that was assigned to this letter before,
- the next unused number, if the letter occurs for the first time.

Here is a formal definition:

Definition Let Σ be an alphabet. Let $a_1, \ldots, a_q$ be letters from Σ. The **pattern** belonging to the string $(a_1, \ldots, a_q)$ ist the q-tuple $(n_1, \ldots, n_q) \in \mathbb{N}^q$ of numbers that is defined recursively by

- $n_1 := 1$.
- For $k = 2, \ldots, q$:
  If there is an i with $1 \leq i < k$ and $a_k = a_i$, then $n_k := n_i$,
  else $n_k := 1 + \max\{n_i \mid 1 \leq i < k\}$.

Remarks

1. $n_i = n_j \iff a_i = a_j$ for $1 \leq i \leq j \leq q$.
2. $\{n_1, \ldots, n_q\} = [1 \ldots m]$ where $m = \#\{a_1, \ldots, a_q\}$ (= number of different letters in $(a_1, \ldots, a_q)$).

Algorithmic Description

Goal: Determine the numerical pattern of a string.

Input: The string as a list `string` = $(a_1, \ldots, a_q)$.

Output: The numerical pattern as a list `pattern` = $(n_1, \ldots, n_q)$.

Initial value: `pattern = empty list`.

Auxiliary variables:

- $n =$ current number, initial value $= 0$.
- `assoc` = list of processed letters.
  The index i belongs to the letter `assoc[i]`.
  Initial value: `assoc = empty list`.

Procedure: Loop over the letters in `string`. The current letter is `x`.

If there is an i with `x = assoc[i]`, then append i to `pattern`,

else increment n, append n to `pattern`, append `x` to `assoc`.

For a Perl program that implements this algorithm see the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/PattPerl.html`