

A Fast Method for the Cryptanalysis of Substitution Ciphers
 Thomas Jakobsen
 January 8, 1995
 Abstract It is possible to cryptanalyze simple substitution ciphers (both mono- and polyalphabetic) by using a fast algorithm based on a process where an initial key guess is refined through a number of iterations. The algorithm needs to compute the distribution matrix only once and subsequent plain text evaluation is done by manipulating this matrix only, and not by decrypting the ciphertext and reparsing the resulting plain text in every iteration. The paper explains the algorithm and it shows some of the results obtained with an implementation in Pascal.

In a polyalphabetic cipher more than one such mapping is used. We assume that the plain text is in English and thus as plain text symbols as well as ciphertext symbols we use the letters A to Z and space. Thus a monoalphabetic key is a permutation of these 27 ciphertext symbols. The ciphertext is obtained from the plain text by replacing each symbol by the corresponding ciphertext symbol in the key. In this way we implicitly get a frequency count on words beginning and ending with a particular letter, namely the "space-letter" and the "letter-space" digram frequencies. Ask etc of the algorithm The idea behind the algorithm can be used for ciphertext-only attacks on other simple ciphers as well. The algorithm starts by making an initial guess about what the key is.

Average English has a certain distribution of letters (space, E, T, A, O, N as the most frequent, in that order) and thus our initial guess is that the most common ciphertext symbol is equivalent to space, the second most common is equivalent to E etc. We now need a function, $f(t)$, which maps a text, t , into a number measuring "how close" the language of t is to the expected language of the plain text, i.e., a function we can use for describing how close we are to having recovered the original plain text. But first let us introduce some notation. Let m denote the plain text, c the ciphertext, and e_k

$f(t)$ the encryption function where k is the key used, so that $c = e_k$

$f(m)$. Correspondingly we have the decryption function d_k

$f(t) = e$

$?/1k$

Let $D(t)$ be the matrix in which the elements are the digram frequencies of a given text t . The row headings are the first symbols of the digram and the column headings are the second symbols. E denotes a matrix containing the expected digram frequencies of the language in which we assume the plain text is written. Digram frequencies for English can be obtained from several works, including [4] and [7]. We, however, compiled our own table using 300,000 characters of text from Melville's *Moby Dick* [6].

5 10 15 20 25 11.101.45

Number of incorrect

symbols in the key, $k_f(d(c,k))$ Figure 1: $f(d(c;k))$ as a function of an increasingly more inaccurate key. Now we can state the main parts of the algorithm more formally. Algorithm 1: 1. Construct an initial key guess, k , based up on the symbol frequencies of the expected language and the ciphertext. 2. Let $v = f(d(c;k))$. 3. Let $k_0 = k$. 4.

Find the corresponding distribution matrix. This problem is solved in the following section. 3 A fast approach Just after step 5 in Algorithm 1 we have

$k_0 = f(d(c;k$

$_0)) =$

$X_i;j$

$j D_{ij}$

$f(d(c;k$

$_0)) = E_{ij}$

$j =$

$X_i;j$

$j D$

$_0ij$

$f(d(c;k_0)) = E_{ij}$

j ; where D

$_0(t)$ is the distribution matrix, $D(t)$, for t with the modification that rows i and j have been swapped after which columns i and j have been swapped (or equiv

alternately swapping the columns π first, then the rows σ). The above fact can be used to optimize the algorithm so that we have to parse the text only once to find the distribution matrix. Construct an initial key guess, k , based up on the symbol frequencies of the expected language and the ciphertext. Let $D = D(d/c; k)$. Let $v =$

$P_i; j$

$j \text{ } D_{ij}$

Swap the elements π and σ in k

0.7.

The strategy used to choose the two elements π and σ , is described in the following. Let s denote the vector of ciphertext symbols ranked in order of descending frequency. Let $\pi = s_a$ and $\sigma = s_{a+b}$

Swap the symbols π and σ in k

0.6b. If $a + b < 2/7$ then go to step 7.6d. Let $a = b = 1$. In this way the frequencies of the key elements π first swapped in k will be close to each other; π first we try to exchange $s/1$

$= s/2$

$, s/2$

$= s/3$

$, s/3$

$= s/4$

$, s/2/6$

$= s/2/7$

$, \text{ then } s/1$

$= s/3$

$, s/2$

$= s/4$

$, s/3$

$= s/5$

$, s/2/5$

$= s/2/7$

/, then $s/1$

$\neq s/4$

/, $s/2$

$\neq s/5$

/, $s/3$

$\neq s/6$

/, $/./././, s/2/4$

$\neq s/2/7$

etc./, and finally $s/1$

$\neq s/2/7$

/- the pair with the largest difference regarding symbol frequency ./4 Poly alphabetic ciphers
For poly alphabetic ciphers with n alphabets the above algorithm is extended so that instead of one alphabet we have several/.

and the columns $/i$ and $/j$ of the distribution matrix of the previous active alphabet/. The evaluation function becomes $f(t) =$

$n \sum_{h=1}^H$

$\sum_{j=1}^J$

$\sum_{j=1}^J$

$f(h)_{ij}$

$f(t) = \sum_{i,j} E_{ij}$

$\sum_{j=1}^J$; and instead of terminating in step 6 if $b \neq 2/7$, we let $b = 1$. We terminate the algorithm when all the symbol pairs of all the alphabets are exhausted without finding any two symbols to swap and at the same time lowering the evaluation value./5
Maximum likelihood If we make the assumption that the digram frequencies of a language are independent and that each one follows a Gaussian distribution with mean μ_{ij}

$\neq E_{ij}$

and variance σ_{ij}^2

$/2ij$

for all i, j , and if the algorithm does not get stuck in local minima/(that is/, if it actually does minimize the evaluation function/) then we can find an evaluation function that results in a maximum likelihood key with respect to the digram

distribution/. For a maximum likelihood attack our job will be to maximize \sum_j

\sum_j

\sum_j

\sum_j

\sum_j

\sum_j

\sum_j

\sum_j

\sum_j

\sum_j

\sum_j

\sum_j

\sum_j

However, if the above mentioned assumptions hold, then using \sum_j should result in a more accurate algorithm (albeit perhaps slower due to the extra multiplications). It should be mentioned that the general problem of minimizing \sum_j or \sum_j given D and E can easily be shown to be at least as hard as the graph isomorphism problem (consider the incidence matrices of the two graphs) but apparently when the plain text is human language, cryptanalysis is quite feasible. As evaluation functions we also tried to use $f(t)$

\sum_j

\sum_j

\sum_j

and the obvious candidate $f(t)$

\sum_j

\sum_j

\sum_j

\sum_j

but it showed up that generally these perform much worse than \sum_j .

Results Fig. 2 shows how successful the algorithm was in attacking texts of various lengths encrypted using both mono- and poly alphabetic substitution and a random key.

Length of

ciphertextPercentage of ciphertext

correctly solved

80

70

60

50

40

30

20

1090100

100 200 400 500 600 700 800 900 300 10004 alphabets

5 alphabets1 alphabet

2 alphabets

3 alphabetsFigure 2/: The percentage of text correctly solved as a function of the length of the ciphertext symbols whereas 93% of the text was correctly resolved using the King-Bahler algorithm (according to [2] and [5]). It is also available in C. 7 ConclusionThe algorithm is quite fast compared to earlier results and it succeeds in cryptanalyzing relatively small texts though the lengths of solved texts are always much larger than the uncertainty distance of the ciphers we are dealing with. As mentioned, the generalized version of the algorithm (Algorithm 1) can be applied to other simple ciphers although it is probably not possible to optimize it so that one can avoid reparsing the text in each iteration (like in Algorithm 2).

the more modern type of encryption algorithms (IDEA, DES, etc.) Finally it should be noted, that there is room for other small improvements as well, such as using a better method for deciding which pairs to swap. Also we might improve the evaluation function so that it not only depends on digrams, but perhaps also on trigrams, probable words and soon. References [1] Carroll, J. M. and S. Martin. 1986. The Automated Cryptanalysis of Substitution Ciphers. Cryptologia 10(4): 193-209. [2] Carroll, J. and L. Robbins. 1987. The

Automated Cryptanalysis of Poly alphabetic Ciphers/. /1/1/(/4)/:
/1/9/3/-/2/0/5/. /[/3/] Carroll/, L/. /1/9/9/1/. A lic e in Wonderland /.

and mathematics/.