



10-2  
DK40-8

# THE HAGELIN CIPHER MACHINE (M-209) Cryptanalysis from Ciphertext Alone

*James Reeds*

University of California  
Berkeley, California 94720

*Dennis Ritchie*

*Robert Morris*

Bell Laboratories  
Murray Hill, New Jersey 07974

## ABSTRACT

This paper describes a method for reading messages enciphered by the Hagelin Cipher Machine. The method is purely statistical in nature and requires only a sufficient quantity of ciphertext. No use is made of probable words nor of known correspondence between cleartext and enciphered messages. The internal settings of the machine are generated in the process so that further messages using those settings can be read with little additional labor.

A computer program embodying this method requires between 300 and 450 words of ciphertext to do its work and runs in about two minutes on a minicomputer.

This paper describes a method for reading messages enciphered using Boris Hagelin's cipher machine, in its commercial version called the C-48, but more widely known as the M-209 Converter from its U. S. Army Signal Corps designation. The method is almost purely statistical in nature. Its successful application requires only a sufficient quantity of ciphertext; the only assumption is that the cleartext is English or some other natural language. In particular, no use is made of probable words nor of known correspondence between cleartext and enciphered messages. An earlier paper by Morris [1] presented a method of reconstructing the complete internal settings of the machine, given only about 75 characters of corresponding cleartext and ciphertext. The algorithm described here requires considerably more ciphertext but no cleartext whatsoever. The general approach is thought to be representative of cryptanalytic techniques actually used by professionals in the field.

Although this paper is rather theoretical in tone, all of the ideas in it have been tried out in practice. The method has been put into the form of a computer program and been tested on a large number of sample messages. This paper is based on an unpublished manuscript by Reeds written in 1970; Ritchie and Morris obtained a copy of the Reeds manuscript and wrote the computer program in 1976.

## Machine Description

A complete description of the operations of the machine was given by Morris [1], and so only the most relevant facts will be repeated. Briefly, the machine produces a polyalphabetic substitution in which each of the 26 possible alphabets is a reversed standard alphabet (called a Beaufort alphabet) with an offset. The key that selects the alphabet is produced internally from a set of six *wheels* around which are a number of *pin positions*; the six wheels have, respectively,

26, 25, 23, 21, 19, and 17 pin positions. In each pin position, a pin may be either *active* or *inactive*, and a selector mechanism along the axis of the wheels selects an encrypting alphabet depending only (during a single message) on the pattern of active and inactive pins in the pin positions opposite the mechanism. As each plaintext letter is enciphered, each wheel advances by one pin position with respect to the selector mechanism. Because the numbers of pin positions on the various wheels are relatively prime, the period is  $26 \times 25 \times 23 \times 21 \times 19 \times 17$  or 101,405,850. On the other hand, the keying sequence is partially periodic in that each wheel returns to the same position every  $n$  letters, where  $n$  is the number of pin positions on its circumference. It is this subtle regularity that Morris was able to exploit in [1], and it is also the basis for the attack described here.

The method that the selector mechanism uses to translate the pin patterns into letters of the keying alphabet can be described as follows: There are 27 sextuples of 0's and 1's in the selector mechanism. Each sextuple is called a *bar*, the 1's are called *lugs*, and the 0's are called *blanks*. However, each bar has no more than two lugs (i.e. 1's) on it. This means that there are only 22 different possible bar patterns; one of all 0's, six consisting of only one 1 and five 0's; and fifteen composed of two 1's and four 0's. (There are fifteen different ways of choosing pairs of objects from six objects in all.) To produce a letter of the keying sequence, each of the 27 bars is compared with the pattern presented by the pin wheels. If, for a given bar, at least one active pin hits a lug on the bar, then we say that that bar is *engaged*. The number of bars that are engaged is the *displacement* for that pin pattern; it can range from 0 to 27. This displacement is then reduced modulo 26, and converted into a Beaufort keying letter according to the scheme  $0=a, 1=b, \dots, 25=z$ .

### An Intuitive Method

The important fact is that the transformation produced by the machine is not statistically random. With almost any arrangement of internal settings, if a long message consisting only of A's is fed in, the letter-frequency distribution of the ciphertext will not be flat, but will be skewed in some way. The exact shape does not matter, and in fact it depends in a complicated way on the internal settings; it is sufficient that it is non-random. If the input text is not a sequence of identical letters, but is instead written in a natural language such as English, the output will still not have a perfectly flat frequency distribution, because the peaks and valleys in the letter distribution of the input will be reflected in very minor peaks and valleys in the distribution of the output. Although the departure from randomness is slight indeed, it is enough to permit an attack.

Suppose we wish to determine the pin settings on a particular wheel, say the 17 wheel for definiteness. If the ciphertext is written in rows of 17 letters, each row below its predecessor, we observe that each column corresponds to a unique pin position on the wheel. That is, ciphertext characters numbered 1, 18, 35, ..., etc. correspond to the first point on this wheel; characters 2, 19, 36, and so forth, correspond to the second pin. Thus the columns may be divided into two groups depending on whether the corresponding pin is active or inactive. The statistics of the columns associated with inactive pins are like those of a machine set so that all pins on the 17 wheel are inactive. Conversely, the letter distribution in the columns associated with active pins is just that of a machine with all pins set on this wheel. Neither distribution is flat, and the two distributions are not identical.

Sensitive statistical tests exist for determining whether two observed distributions are in fact drawn from the same population. By comparing the distributions of the pairs of columns, it is possible to classify the columns into two groups, one of which is associated with active pins and one with inactive pins. In the first stage of analysis, it is not possible to say which group is associated with active pins and which with inactive pins, but this does not matter. Various appropriate statistical tests are described and discussed in Good [2], Sinkov [3], Friedman [4], and in synoptic detail, in Kahn [5, pp.377-383].

We could proceed by comparing each pair of the 17 columns using one of these statistical tests. Pairs which appear to be similar could be put into the same class and the process

continued until each of the columns had been assigned to one of the two classes. The method to be described in the next section in rather sparse mathematical terms is merely a generalization of this intuitively obvious method of segregating the distributions into two classes depending on the "distance" between them given by a sensitive statistical test. The method we describe does not concern itself with examining particular pairs of pins, but rather does the analysis all at once.

### The Mathematical Basis

The basis of the method is simple. Let  $f_{i\lambda}$  stand for the number of occurrences of cipher-text letter  $\lambda$  with pin position  $i$ . Usually,  $\lambda$  ranges from 1 to 26 (the number of letters in the alphabet), but, as explained below, can also range from 1 to 52 or more. At any rate, let the number of letters be  $L$ . Also,  $i$  ranges from 1 to  $N$ , where  $N$  may be one of the numbers 17, 19, 21, 23, 25, or 26, depending on which wheel is being attacked. The intention is now to find two "ideal" frequency counts, associated with the two segregation classes of pins. Let the number of letters  $\lambda$  in these two ideal frequency distributions be  $c_\lambda$  and  $b_\lambda$ , respectively. Then, for each pin position  $i$ ,  $f_{i\lambda}$  is either closely approximated by  $c_\lambda$  or by  $b_\lambda$ . Pin position  $i$  will be considered to fall into the segregation class that its actual distribution most closely matches. That is, if pin  $i$  belongs to the first class,  $f_{i\lambda}$  will be close to  $c_\lambda$ ; if pin  $i$  belongs to the second segregation class,  $f_{i\lambda}$  will be close to  $b_\lambda$ . To quantify this effect, the variable  $x_i$  is introduced, where we intend that  $x_i = 1$  if pin  $i$  is in the first class, and  $x_i = 0$  if pin  $i$  is in the second class. The problem is then to find ideal frequency counts  $c_\lambda$  and  $b_\lambda$ , as well as values of  $x_i$ , so that  $f_{i\lambda}$  is closely approximated by the quantity

$$x_i c_\lambda + (1-x_i) b_\lambda.$$

Note that if  $x_i = 0$ , this expression reduces to  $b_\lambda$ ; if  $x_i = 1$ , it reduces to  $c_\lambda$ . Because the "ideal" frequency distributions do not exist, the most we can hope for is to find the best fractional values of the  $x_i$ . If, for example, we find that  $x_1 = .27$ , we might say that pin 1 is 27% like the first ideal segregation class, and 73% like the second class. We would most probably be correct in assigning pin position 1 to the second class. If, say,  $x_2 = .48$ , we would have no grounds for assigning pin 2 to either class.

The proposed method in effect finds the values of  $x_i$ ,  $b_\lambda$ , and  $c_\lambda$  so that  $f_{i\lambda}$  is most closely approximated by  $x_i c_\lambda + (1-x_i) b_\lambda$ . Here "best approximated" is taken in the least-squares sense: the quantity  $\Phi$  is minimized, where

$$\Phi = \sum_{i=1}^N \sum_{\lambda=1}^L \{f_{i\lambda} - [x_i c_\lambda + (1-x_i) b_\lambda]\}^2. \quad (1)$$

For the cryptanalytic problem, it turns out to be unnecessary to recover the actual values of  $c_\lambda$ ,  $b_\lambda$ , and  $x_i$ ; certain related quantities will do. Thus we will rewrite the expression for  $\Phi$ . Let  $a_\lambda = c_\lambda - b_\lambda$ . Then  $\Phi$  is given by

$$\Phi = \sum_{i=1}^N \sum_{\lambda=1}^L (f_{i\lambda} - x_i a_\lambda - b_\lambda)^2. \quad (2)$$

Then, differentiating  $\Phi$  with respect to  $b_\lambda$  and setting the result to 0,

$$\frac{\partial \Phi}{\partial b_\lambda} = -2 \sum_{i=1}^N (f_{i\lambda} - x_i a_\lambda - b_\lambda) = 0, \quad (3)$$

and thus

$$b_\lambda = \sum_{i=1}^N f_{i\lambda} / N - a_\lambda \sum_{i=1}^N x_i / N. \quad (4)$$

If we now let

$$g_{i\lambda} = f_{i\lambda} - \sum_{i=1}^N f_{i\lambda} / N,$$

we get:

$$\Phi = \sum_{i=1}^N \sum_{\lambda=1}^L \left( g_{i\lambda} - x_i a_\lambda + \left( \sum_{j=1}^N x_j / N \right) a_\lambda \right)^2 . \quad (5)$$

Now let

$$y_i = x_i - \sum_{j=1}^N x_j / N .$$

It is the quantities  $y_i$  that will actually be calculated. The  $y_i$  are just the  $x_i$  minus the average value of the  $x$ 's; unless this average value is known, the actual  $x_i$  cannot be recovered. But since for the purposes of solving the cryptanalytical problem we care only about which  $x_i$  are larger than average and which are smaller, this is a harmless loss of information.

Now  $\Phi$  is given by the formula

$$\Phi = \sum_{i=1}^N \sum_{\lambda=1}^L (g_{i\lambda} - y_i a_\lambda)^2 . \quad (6)$$

The problem of finding the  $y_i$  and the  $a_\lambda$  that minimize  $\Phi$ , as given in equation (6), is well known; it is treated, for example, in Good [2]. As explained there, the choice of  $y_i$  that minimizes  $\Phi$  is given by the eigenvector corresponding to the largest eigenvalue of the matrix  $\Gamma = [\gamma_{ij}]$ , where

$$\gamma_{ij} = \sum_{\lambda=1}^L g_{i\lambda} g_{j\lambda} . \quad (7)$$

That is, the  $y_i$  we are looking for are solutions of the equation

$$\sum_{j=1}^N \gamma_{ij} y_j = k y_i , \quad (8)$$

where  $k$  has the largest possible value consistent with this equation. Equation (8) is an example of an *eigenvector equation*, a subject which has been extensively studied. There are standard, efficient procedures for the solution of these equations.

So the recipe is:

- a) Calculate  $g_{i\lambda} = f_{i\lambda} - \sum_{j=1}^N f_{j\lambda} / N$ .
- b) Calculate  $\gamma_{ij}$ , using equation (7) above.
- c) Use a standard subroutine to solve the eigenvector equation (8) in order to find the eigenvector corresponding to the largest eigenvalue.

This procedure has a simple geometrical interpretation. If we regard a pin's frequency count as a point in an  $L$ -dimensional space, the eigenvector method solves the following geometrical problem: to find the line in this  $L$ -dimensional space for which the sum of the various squared distances of the points from the line is at a minimum. The sum of squared distances is actually the quantity  $\Phi$ ; the position and direction of the line are determined by  $a_\lambda$  and  $b_\lambda$ . Then one drops perpendiculars from each point to the line, and notes where the perpendiculars hit the line. The positions of the hit points along the line are given by the values of the  $y_i$ .

If the  $N$  points (pin positions) separate cleanly into two groups in  $L$ -space, so will their projections along the line. We will consider a pin position  $i$  to fall into one class if the value of  $y_i$  is greater than average, and into the other class if  $y_i$  is less than average. At the same time it is possible to calculate a measure of the significance of this segregation by observing how closely clustered the points are. One simple way to do this is to scale the  $y_i$  so they range in value from 0 to 1, and then to measure the root-mean-square deviation from the (scaled) mean. (The scaling is necessary because the method of generating  $y_i$  guarantees that the mean is 0 and the norm is 1).

### **An Improvement**

The possibility of segregating the pins on a particular wheel depends, as we have seen, on the fact that the randomizing effect of the other five wheels is not sufficient to wipe out the unevenness of the letter-frequency distribution of the input. Once the pins on one of the wheels have been segregated, we can use the information gained to reduce the randomizing effect and improve the sensitivity of the tests applied to the remaining wheels. Suppose we have segregated the pins on the first wheel and are about to attack the second. For each enciphered letter, it is known whether it was encrypted with an active or inactive pin on the first wheel. Now we double the size of the alphabet by attaching to each letter an indication of its segregation set on the first wheel. Thus, for example, if the letter A is observed in the output it will be called  $A_0$  if it is associated with a pin position in one of the segregation sets of the first wheel, and  $A_1$  if it appears in the other. This technique in effect subtracts out the contribution of the first wheel and makes the calculations for the second wheel more reliable. It is true that there is some loss of information; by doubling the size of the alphabet we have halved the average number of occurrences of each letter and rendered the distribution more subject to truly random perturbations.

Of course, this procedure can be applied more than once; after the pins on two wheels have been segregated, the ciphertext alphabet can be quadrupled by marking each letter with one of the four possibilities for the associated pin positions on the two known wheels, and so on.

### **The Pin Program**

The part of the decryption program that identifies pins operates according to the principles already described. It is iterative in the sense that it is willing to recalculate the pin segregation on a given wheel several times. The following example shows how this is useful. Suppose the pins on the first wheel, and then on the second have been segregated. Using the idea of alphabet-splitting described in the last section, it is likely that the segregation of the first wheel's pins can be improved (made more nearly correct) if the calculation for the first wheel is repeated using the information gained from the segregation of the pins on the second wheel. The iteration is guided by an estimate of the correctness (significance) of the segregation on each wheel calculated from the appropriate eigenvector as described above.

At each iteration, the program recalculates each wheel in turn. The four most significant wheels (other than the one being worked on) are used to generate a sixteen-fold splitting of the ciphertext alphabet. (The only reason for not using all five remaining wheels is that the amount of space required to handle all the alphabets would be excessive.) After each iteration over all six wheels, the sum of the significance measures is examined, and if it has increased since the last iteration, another iteration is performed; otherwise the pin settings are declared correct and the program moves on to the next stage.

### **Identification of the Alphabets**

Once all the pins have been identified, it remains to solve the actual message and to determine the machine settings. Each ciphertext letter can be marked with a six-bit subscript according to the pattern of pin positions opposite the selector mechanism when the letter is enciphered. Each of the 64 possible subscripts uniquely determines one of the 26 possible Beaufort alphabets used to generate the ciphertext, and it is only necessary to determine which subscript designates which alphabet.

The simplest idea that comes to mind is to observe that the letters enciphered with a particular subscript have been subjected to simple substitution, so it should be possible just to match the output letter frequencies to each of the 26 alphabets and select the best match (based on some test), on the assumption that the input language was, say, English. The approach is attractive because it requires no assumptions whatever about the workings of the selector mechanism, and it actually does work, but it turns out that the combined effects of small sample size (2048 characters divided by 64 subscripts gives only 32 letters per alphabet), and the

likelihood of a few errors in the pins to distort the situation, are enough to make a determination by this method rather inaccurate. Our program actually does take this approach, and in the next section we show how the choice of alphabets is improved so as to make the output text more readable.

### The Lug Program

In general, some of the 64 alphabets will be wrongly chosen. If most of the alphabets are correct, it will then be possible to make use of known characteristics of the selector mechanism to improve the results. In particular if only a few of the alphabets are incorrect, say no more than about 20% of them, then they can all be corrected.

The selector mechanism has the property that if no pins are set, then the displacement is zero. Each of the 27 bars has either zero, one, or two lugs on it which are set. Therefore there are 22 possible configurations for a bar and we need only try to count how many bars possess each configuration. The first step is to distinguish correctly which pins are set and which are not set, as this has not been correctly done up to now (and did not matter). For each of the six positions, we inspect each of the 32 pairs of displacements that differ only in the selected position. If this position tends generally to increase the displacement when its pin is set, then the alphabet is left alone; if it tends to decrease the displacement, then the settings in that position are reversed. Since there are 32 comparisons to be made, we make the determination on the basis of a majority vote.

Next we choose a pair of positions, and, keeping all other positions constant, there are four configurations of the two selected positions, namely, 00, 01, 10, and 11 (where 0=off; 1=on). If the four displacements are correct, then the function

$$F(0,1)+F(1,0)-F(0,0)-F(1,1)$$

of the four displacements  $F(i,j)$  is the overlap between the two positions, i.e. the number of bars with their two lugs in these two positions. In fact we get 16 versions of this overlap and we let them vote; the most frequent version of the overlap is selected. The process is repeated for each pair of positions.

Next we compute for every position a tentative number of bars that have a lug only in that position, by using the overlaps just computed to correct each of the 32 relevant displacements. Again we let them vote and the most frequent result is chosen as the number of unshared bars at the current position. It doesn't matter how many bars there are with zero lugs, since they do not enter into the encryption process.

Finally, we ignore the original displacements and re-create the displacements completely from scratch using the just-determined array of overlaps and the counts of unshared bars.

### Results

The program was tested by using ciphertexts encrypted by a Hagelin machine with 30 different internal settings. The settings were randomly chosen within the range of requirements believed to have been utilized by the Signal Corps during World War II. Only ten different cleartext passages were used, each one encrypted using three different internal settings. Each test was run several times, using initial sequences ranging in length from 1750 to 2750 characters in length. Three levels of performance were observed.

- 1) When the program made no headway, the output was gibberish.
- 2) When the settings of the machine were broken in principle, the output was not easily readable but had sufficient patches of cleartext to be able to apply the methods of the Morris paper [1] and reconstruct the correct settings.
- 3) The output was perfect or nearly so, with errors only every 10 or 20 characters. (Precisely, no more than 2 pin errors).

Of course classification 2) is a bit vague; it depends on how clever one is in pulling sense out of nonsense. An example of what we mean is given in the following few lines. This

example follows the convention adopted for the M-209 that blanks are represented by the letter 'z'. The M-209 does not print the letter 'z' during decryption; a blank appears in its place.

assumng it therefore as an esablished truth that  
uge several stater in base of disunionyor stch co  
mbinationt of tgem as might hapen to be formedaou  
taof thd wreck og uhe henerak confedesady would be  
subjectatp thpse vibissitudet of pface amd war of  
fsiendshipaand enmit xithaeachaothor whidh havfy  
falmen to the lpt ofyallaneighborioganatinns not u

With a little imagination the text is readable at sight; more precisely, it is clear that the approximately 75 correct characters needed to reconstruct the internal settings by the method of Morris [1] are easily available.

The following table summarizes the results.

length	gibberish (%)	readable (%)	perfect (%)
1500	97	0	3
1750	77	13	10
2000	53	20	27
2250	23	3	73
2500	17	0	83
2750	0	0	100

To some extent the table is pessimistic because two of the five messages decoded as "gibberish" when looking at 2500 characters were actually readable when only 2250 characters were used; so actually 90 percent of the 2500 character messages could be understood. Thus the amount of text required varies by a factor of about two, apparently depending on chance details of the machine settings and the nature of the cleartext.

These results are for one particular version of the general method of attack and slightly different implementations could well perform better. No great amount of time was spent in trying to improve the results, and it is quite likely that they could be improved. The authors felt that once a program had been developed that was capable of reading and automatically producing cleartext versions of encrypted messages of lengths less than 400 words, the point had been proved.

It seems safe to say that any message over 1750 characters long (about 300 words) carries a substantial risk of being read, and that any message over about 2500 characters (about 420 words) is almost sure to be readable. Messages of this length are somewhat longer than those that would be typical of front-line military traffic, but much, much shorter than diplomatic and peacetime military messages. Only one readable message is needed to determine the internal settings; when these are known, decryption of rather short messages using other keys is rather easy. Moreover, because digits and punctuation must be spelled out, the number of characters required for even short messages is inflated beyond what would be required for ordinary printed communication. Thus, one may say that use of this machine for messages much more lengthy than the traditional "attack at dawn" is unwise unless dawn is already at hand. The program takes about two minutes to produce a solution on a DEC PDP-11/70.

## References

- [1] Morris, Robert. "The Hagelin Cipher Machine (M-209): Reconstruction of the Internal Settings." *Cryptologia* 2, to appear.

- [2] Good, I. J. *The Estimation of Probabilities*, MIT Press, Cambridge, Mass, 1965.
- [3] Sinkov, A. *Elementary Cryptanalysis*, Random House, New York, 1968.
- [4] Friedman, W. F. *Military Cryptanalysis*, War Dept., Office of the Chief Signal Officer, U.S. Government Printing Office, Washington, D.C.
- [5] Kahn, David, *The Codebreakers*, Macmillan, New York, 1967.