CHAPTER 8

The Enigma Machine

Chapter Goals

- To explain the working of the Enigma machine.
- To explain how the German military used the Enigma machine during World War II, in particular how session keys were transmitted from the sender to the receiver.
- To explain how this enabled Polish and later British cryptanalysts to read the German traffic.
- To explain the use of the Bombe in mounting known plaintext attacks.

8.1. Introduction

With the advent of the 1920s people saw the need for a mechanical encryption device. Taking a substitution cipher and then rotating it was identified as an ideal solution. This idea had actually been used previously in a number of manual ciphers, but mechanization was able to make it far more efficient. The rotors could be implemented using wires and then encryption could be done mechanically using an electrical circuit.

By rotating the rotor we obtain a new substitution cipher. As an example, suppose the rotor used to produce the substitutions is given by the following values in the first position:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
TMKGOYDSIPELUAVCRJWXZNHBQF.
```

To encrypt the first letter we use the substitutions given above; i.e. we substitute B by M and Y by Q. However, to encrypt the second letter we rotate the rotor by one position, i.e. we move the bottom row one step to the left, and so use the substitutions

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
MKGOYDSIPELUAVCRJWXZNHBQFT,
```

whilst for the third letter we use the substitutions

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
KGOYDSIPELUAVCRJWXZNHBQFTM,
```

and so on. This gives us a polyalphabetic substitution cipher with 26 different alphabets.

The most famous of these machines was the Enigma machine used by Germany in World War II. We shall describe the most simple version of Enigma which only used three such rotors, chosen from the following set of five:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
EKMFLGDQVZNTOWYHXUSPAIBRCJ
AJDKSIRUXBLHWTMCQGZNPYFVOE
BDFHJLCPRTXVZNYEIWGAKMUSQO
ESOVPZJAYQUIRHXLNFTGKDCMWB
VZBRGITYUPSDNHLXAWMJQOFECK.
```


N.P. Smart, *Cryptography Made Simple*, Information Security and Cryptography, DOI 10.1007/978-3-319-21936-3_8

Machines in use towards the end of the war had a larger number of rotors, chosen from a larger set. Note that the order of the rotors in the machine is important, so the number of ways of choosing the rotors is

$$5 \cdot 4 \cdot 3 = 60.$$

Each rotor had an initial starting position, and since there are 26 possible starting positions for each rotor, the total number of possible starting positions is $26^3 = 17\,576$.

The first rotor would step on the second rotor on each full iteration under the control of a ring hitting a notch; likewise the stepping of the third rotor was controlled by the second rotor. Both the rings were movable and their positions again formed part of the key, although only the notch and ring positions for the first two rotors were important. Hence, the number of ring positions was $26^2 = 676$. The second rotor also had a "kick" associated with it, making the cycle length of the three rotors equal to

$$26 \cdot 25 \cdot 26 = 16\,900.$$

The effect of the moving rotors was that a given plaintext letter would encrypt to a different ciphertext letter on each press of the keyboard. Finally, a plugboard was used to swap letters twice in each encryption and decryption operation. This increased the complexity and gave another possible 10^{14} keys. The rotors used, their order, their starting positions, the ring positions and the plugboard settings all made up the secret key. Hence, the total number of keys was then around 2^{75}. To make sure encryption and decryption were the same operation a reflector was used. This was a fixed public substitution given by

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
YRUHQSLDPXNGOKMIEBFZCWVJAT.
```

To encrypt a plaintext character it would first be passed through the plugboard (thus possibly swapping it to another letter), then it passed forwards through the rotors, then through the reflector, then backwards through the rotors, and finally it would pass once more through the plugboard.

The operation of a simplified four-letter Enigma machine is depicted in Figure 8.1. By tracing the red lines one can see how the plaintext character A encrypts to the ciphertext character D. Notice that decryption can be performed with the machine in the same configuration as used for encyption. Now assume that rotor one moves on one step, so A now maps to D under rotor one, B to A, C to C and D to B. You should work out what happens with the example when we encrypt A again.

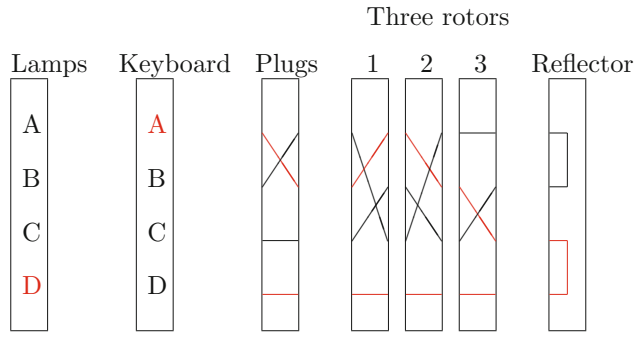

FIGURE 8.1. Simplified Enigma machine

In the rest of this chapter we present more details of the Enigma machine and some of the attacks which can be performed on it. However before presenting the machine itself we need to

fix some notation which will be used throughout this chapter. In particular lower-case letters will denote variables, upper-case letters will denote "letters" (of the plaintext/ciphertext languages) and Greek letters will denote permutations in S_{26} which we shall represent as permutations on the upper case letters. Hence x can equal X and Y, but X can only ever represent X, whereas χ could represent (XY) or (ABC).

Permutations will usually be given in cycle notation. One always has to make a choice as to whether we multiply permutations from left to right, or right to left. We decide to use the left-to-right method, hence

$$(ABCD)(BE)(CD) = (AEBD).$$

Permutations hence act on the right of letters, something we will denote by x^σ, e.g.

$$A^{(ABCD)(XY)} = B.$$

This is consistent with the usual notation of right action for groups acting on sets. See the appendix for more details about permutations.

We now collect some basic facts and theorems about permutations which we will need in the sequel.

Theorem 8.1. *Two permutations σ and τ which are conjugate, i.e. ones for which $\sigma = \lambda \cdot \tau \cdot \lambda^{-1}$ for some permutation λ, have the same cycle structure.*

We define the support of a permutation to be the set of letters which are not fixed by the permutation. Hence, if σ acts on the set of letters $\mathcal{L}$, then as usual we denote by $\mathcal{L}^\sigma$ the set of fixed points and hence the support is given by

$$\mathcal{L} \setminus \mathcal{L}^\sigma.$$

Theorem 8.2. *If two permutations, with the same support, consist only of disjoint transpositions then their product contains an even number of disjoint cycles of the same length. Conversely, if a permutation with support an even number of symbols has an even number of disjoint cycles of the same length, then the permutation can be written as a product of two permutations each of which consists of disjoint transpositions.*

Solving a Conjugation Problem: In many places we need an algorithm to solve the following problem: Given $\alpha_i, \beta_i \in S_{26}$, for $i = 1, \ldots, m$ find $\gamma \in S_{26}$ such that

$$\alpha_i = \gamma^{-1} \cdot \beta_i \cdot \gamma \text{ for } i = 1 \ldots, m.$$

Whilst there could be many such solutions γ, in the situations to which we will apply it we expect there to be only a few. For example, suppose we have one such equation with

$$\begin{aligned} \alpha_1 &= (AFCNE)(BWXHUJOG)(DVIQZ)(KLMYTRPS), \\ \beta_1 &= (AEYSXWUJ)(BFZNO)(CDPKQ)(GHIVLMRT) \end{aligned}$$

We need to determine the structure of the permutation γ such that

$$\alpha_1 = \gamma^{-1} \cdot \beta_1 \cdot \gamma.$$

We first look at what A should map to under γ. Suppose $A^\gamma = B$; then we have the equations

$$A^{\gamma \cdot \alpha_i} = B^{\alpha_i} = W \text{ and } A^{\beta_1 \cdot \gamma} = E^\gamma.$$

Thus we have $E^\gamma = W$. We then look at the equations

$$E^{\gamma \cdot \alpha_i} = W^{\alpha_i} = X \text{ and } E^{\beta_1 \cdot \gamma} = Y^\gamma.$$

So we have $Y^\gamma = X$. Continuing in this way via a pruned depth-first search we can determine a set of possible values for γ. Such an algorithm is relatively simple to write down in C, using a recursive procedure call. However, it is, of course, a bit of a pain to do this by hand, as would have been the only option in the 1930s and 1940s.

8.2. An Equation for the Enigma

To aid our discussion in later sections we now describe the Enigma machine as a permutation equation. We first assume a canonical map between letters and the integers $\{0, 1, \ldots, 25\}$ such that 0 is A, 1 is B etc. and we assume a standard three-wheel Enigma machine.

The wheel which turns the fastest we shall call rotor one, whilst the one which turns the slowest we shall call rotor three. This means that, when looking at a real machine rotor three is the leftmost rotor and rotor one is the rightmost rotor. Please keep this in mind as it can cause confusion (especially when reading day/message settings). The basic permutations which make up the Enigma machine are as follows.

Choice of Rotors: We assume that the three rotors are chosen from the following set of five rotors, presented in the table below. The Germans labelled these rotors I, II, III, IV and V, and they are the ones used in the actual Enigma machines. Each rotor also has a different notch position which controls how the stepping of one rotor drives the stepping of the others.

Rotor	Permutation Representation	Notch Position
I	$(AELTPHQXRU)(BKNW)(CMOY)(DFG)(IV)(JZ)$	16, i.e. Q
II	$(BJ)(CDKLHUP)(ESZ)(FIXVYOMW)(GR)(NT)$	4, i.e. E
III	$(ABDHPEJT)(CFLVMZOYQIRWUKXSG)$	21, i.e. V
IV	$(AEPLIYWCOXMRFZBSTGJQNH)(DV)(KU)$	9, i.e. J
V	$(AVOLDRWFIUQ)(BZKSMNHYC)(EGTJPX)$	25, i.e. Z

Reflector: A number of different reflectors were used in actual Enigma machines. In our description we shall use the reflector given earlier, which is often referred to as "Reflector B". This reflector has representation via disjoint cycles as

$$\varrho = (AY)(BR)(CU)(DH)(EQ)(FS)(GL)(IP)(JX)(KN)(MO)(TZ)(VW).$$

An Enigma Key: An Enigma key consists of the following information:

- A choice of rotors ρ_1, ρ_2, ρ_3 from the above choice of five possible rotors. Note that this choice of rotors affects the three notch positions, which we shall denote by n_1, n_2 and n_3. Also, as noted above, the rotor ρ_3 is placed in the left of the actual machine, whilst rotor ρ_1 is placed on the right. Hence, if in a German code book it says use rotors
$$I, II, III,$$
this means in our notation that ρ_1 is selected to be rotor III, that ρ_2 is selected to be rotor II and ρ_3 is selected to be rotor I.
- One must also select the ring positions, which we shall denote by r_1, r_2 and r_3. In the actual machine these are letters, but we shall use our canonical numbering to represent these as integers in $\{0, 1, \ldots, 25\}$.
- The plugboard is simply a product of disjoint transpositions which we shall denote by the permutation τ. In what follows we shall denote a plug linking letter A with letter B by $A \leftrightarrow B$.
- The starting rotor positions we shall denote by p_1, p_2 and p_3. These are the letters which can be seen through the windows on the top of the Enigma machine. Remember our numbering system is that the window on the left corresponds to p_3 whilst the one on the right corresponds to p_1.

The Encryption Operation: We let σ denote the shift-up permutation given by

$$\sigma = (ABCDEFGHIJKLMNOPQRSTUVWXYZ).$$

The stepping of the second and third rotor is probably the hardest part to grasp when first looking at an Enigma machine, however this has a relatively simple description when one looks at it in a mathematical manner.

Given the above description of the key we wish to deduce the permutation ϵ_j, which represents, for $j = 0, 1, 2, \ldots$, the encryption of the jth letter. We first set

$$\begin{aligned} m_1 &= n_1 - p_1 - 1 \pmod{26}, \\ m &= n_2 - p_2 - 1 \pmod{26}, \\ m_2 &= m_1 + 1 + 26 \cdot m. \end{aligned}$$

The values of m_1 and m_2 control the stepping of the second and third rotors.

We let $\lfloor x \rfloor$ denote the round towards zero function, i.e. $\lfloor 1.9 \rfloor = 1$ and $\lfloor -1.9 \rfloor = -1$. We now set, for encrypting letter j,

$$\begin{aligned} k_1 &= \lfloor (j - m_1 + 26)/26 \rfloor, \\ k_2 &= \lfloor (j - m_2 + 650)/650 \rfloor, \\ i_1 &= p_1 - r_1 + 1, \\ i_2 &= p_2 - r_2 + k_1 + k_2, \\ i_3 &= p_3 - r_3 + k_2. \end{aligned}$$

Notice how i_3 is stepped on every $650 = 26 \cdot 25$ iterations whilst i_2 is stepped on every 26 iterations and also stepped on an extra notch every 650 iterations.

We can now present ϵ_j as

$$\begin{aligned} \epsilon_j = \; & \tau \cdot (\sigma^{i_1+j} \cdot \rho_1 \cdot \sigma^{-i_1-j}) \cdot (\sigma^{i_2} \cdot \rho_2 \cdot \sigma^{-i_2}) \cdot (\sigma^{i_3} \cdot \rho_3 \cdot \sigma^{-i_3}) \cdot \varrho \\ & \cdot (\sigma^{i_3} \cdot {\rho_3}^{-1} \cdot \sigma^{-i_3}) \cdot (\sigma^{i_2} \cdot {\rho_2}^{-1} \cdot \sigma^{-i_2}) \cdot (\sigma^{i_1+j} \cdot {\rho_1}^{-1} \cdot \sigma^{-i_1-j}) \cdot \tau. \end{aligned}$$

Note that the same equation/machine is used to encrypt the jth letter as is used to decrypt the jth letter. Hence we have

$${\epsilon_j}^{-1} = \epsilon_j.$$

Also note that each ϵ_j consists of a product of disjoint transpositions. We shall always use γ_j to represent the internal rotor part of the Enigma machine, hence

$$\epsilon_j = \tau \cdot \gamma_j \cdot \tau.$$

8.3. Determining the Plugboard Given the Rotor Settings

For the moment, assume that we know values for the rotor order, ring settings and rotor positions; for this purpose we are given γ_j. We would like to determine the plugboard settings. The goal is therefore to determine τ given some information about ϵ_j for some values of j. One often sees it written that determining the plugboard given the rotor settings is equivalent to solving a substitution cipher. This is true, but the methods given in some sources are too simplistic.

Let m denote the actual message being encrypted, c the corresponding ciphertext, and m' the ciphertext decrypted under the cipher with no plugboard, i.e. with the obvious notation,

$$\begin{aligned} m &= c^{\epsilon}, \\ m' &= c^{\gamma}. \end{aligned}$$

The following is an example value of m' for a plugboard containing only one plug

```
ZNCT UPZN A EIME, THEKE WAS A GILL CALLED SNZW WHFTE.
```

I have left the spacings in the English words. You may then deduce that Z should really map to O, or T should really map to E, or E should really map to T, or maybe K should really map to R, etc. But which should be the correct plug setting to achieve this mapping? The actual correct plug setting is that O should map to Z; the other mappings are the result of this single plug setting. We now present some ways of obtaining information about the plugboard given various scenarios.

8.3.1. Ciphertext Only Attack: In a ciphertext only attack one can proceed as one would for a normal substitution cipher. We need a method to distinguish something which could be natural language from something which is completely random. The best option seems to be to use something called the Sinkov statistic. Let f_i, for $i = A, \ldots, Z$, denote the frequencies of the various letters in standard English. For a given piece of text we let n_i, for $i = A, \ldots, Z$, denote the frequencies of the various letters within the sample piece of text. The Sinkov statistic for the sample text is given by

$$s = \sum_{i=A}^{Z} n_i \cdot f_i.$$

The higher the value of s, the more likely that the text represents an extract from a natural language.

To mount a ciphertext only attack we let γ_j denote our current approximation for ϵ_j (initially γ_j has no plug settings, but this will change as the method progresses). We now go through all possible single-plug settings, $\alpha^{(k)}$. There are $26 \cdot 25/2 = 325$ of these. We then decrypt the ciphertext c using the cipher

$$\alpha^{(k)} \cdot \gamma_j \cdot \alpha^{(k)}.$$

This results in 325 possible plaintext messages $m^{(k)}$ for $k = 1, \ldots, 325$. For each one of these we compute the Sinkov statistic s_k, and keep the value of $\alpha^{(k)}$ which results in s_k being maximized. We then set our new γ_j to be $\alpha^{(k)} \cdot \gamma_j \cdot \alpha^{(k)}$ and repeat until no further improvement can be made in the test statistic.

This methodology seems very good at finding the missing plugboard settings. Suppose we are given that the day setting is

Rotors	Rings	Pos	Plugboard
III, II, I	PPD	MDL	Unknown

The actual hidden plugboard is given by $A \leftrightarrow B$, $C \leftrightarrow D$, $E \leftrightarrow F$, $G \leftrightarrow H$, $I \leftrightarrow J$ and $K \leftrightarrow L$. We obtain the ciphertext

```
HUCDODANDHOMYXUMGLREDSQQJDNJAEXUKAZOYGBYLEWFNWIBWILSMAETFFBVPR
GBYUDNAAIEVZZKCUFNIUTOKNKAWUTUWQJYAUHMFWJNIQHAYNAGTDGTCTNYKTCU
FGYQBSRRUWZKZFWKPGVLUHYWZCZSOYJNXHOSKVPHGSGSXEOQWOZYBXQMKQDDXM
BJUPSQODJNIYEPUCEXFRHDQDAQDTFKPSZEMASWGKVOXUCEYWBKFQCYZBOGSFES
OELKDUTDEUQZKMUIZOGVTWKUVBHLVXMIXKQGUMMQHDLKFTKRXCUNUPPFKWUFCU
PTDMJBMMTPIZIXINRUIEMKDYQFMIQAEVLWJRCYJCUKUFYPSLQUEZFBAGSJHVOB
CHAKHGHZAVJZWOLWLBKNTHVDEBULROARWOQGZLRIQBVVSNKRNUCIKSZUCXEYBD
QKCVMGLRGFTBGHUPDUHXIHLQKLEMIZKHDEPTDCIPF
```

The plugboard settings are found in the following order $I \leftrightarrow J$, $E \leftrightarrow F$, $A \leftrightarrow B$, $G \leftrightarrow H$, $K \leftrightarrow L$ and $C \leftrightarrow D$. The plaintext is determined to be:

```
ITWASTHEBESTOFTIMESITWASTHEWORSTOFTIMESITWASTHEAGEOFWISDOMITWA
STHEAGEOFFOOLISHNESSITWASTHEEPOCHOFBELIEFITWASTHEEPOCHOFINCRED
ULITYITWASTHESEASONOFLIGHTITWASTHESEASONOFDARKNESSITWASTHESPRI
NGOFHOPEITWASTHEWINTEROFDESPAIRWEHADEVERYTHINGBEFOREUSWEHADNOT
HINGBEFOREUSWEWEREALLGOINGDIRECTTOHEAVENWEWEREALLGOINGDIRECTTH
EOTHERWAYINSHORTTHEPERIODWASSOFARLIKETHEPRESENTPERIODTHATSOMEO
```

FITSNOISIESTAUTHORITIESINSISTEDONITSBEINGRECEIVEDFORGOODORFORE
VILINTHESUPERLATIVEDEGREEOFCOMPARISONONLY

8.3.2. Known Plaintext Attack: When one knows the plaintext there is a choice of two methods one can employ. The first method is simply based on a depth-first search technique, whilst the second makes use of some properties of the encryption operation.

Technique One: In the first technique we take each wrong letter in turn, from our current approximation γ_j to ϵ_j. In the above example of the encryption of the first sentences from "A Tale of Two Cities", we have that the first ciphertext letter H should map to the plaintext letter I. This implies that the plugboard must contain plug settings $H \leftrightarrow p_H$ and $I \leftrightarrow p_I$, for letters p_H and p_I with

$$p_H{}^{\gamma_0} = p_I.$$

In a similar manner we deduce the following further equations:

$$\begin{array}{lll} p_U{}^{\gamma_1} = p_T, & p_C{}^{\gamma_2} = p_W, & p_D{}^{\gamma_3} = p_A, \\ p_O{}^{\gamma_4} = p_S, & p_D{}^{\gamma_5} = p_T, & p_A{}^{\gamma_6} = p_H, \\ p_N{}^{\gamma_7} = p_E, & p_D{}^{\gamma_8} = p_B, & p_H{}^{\gamma_9} = p_E. \end{array}$$

The various permutations representing the first few γ_js for the given rotor and ring positions are as follows:

$$\begin{array}{rcl} \gamma_0 & = & (AW)(BH)(CZ)(DE)(FT)(GJ)(IN)(KL)(MQ)(OV)(PU)(RS)(XY), \\ \gamma_1 & = & (AZ)(BL)(CE)(DH)(FK)(GJ)(IS)(MX)(NQ)(OY)(PR)(TU)(VW), \\ \gamma_2 & = & (AZ)(BJ)(CV)(DW)(EP)(FX)(GO)(HS)(IY)(KL)(MN)(QT)(RU), \\ \gamma_3 & = & (AF)(BC)(DY)(EO)(GU)(HK)(IV)(JR)(LX)(MN)(PW)(QS)(TZ), \\ \gamma_4 & = & (AJ)(BD)(CF)(EL)(GN)(HX)(IM)(KQ)(OS)(PV)(RT)(UY)(WZ), \\ \gamma_5 & = & (AW)(BZ)(CT)(DI)(EH)(FV)(GU)(JO)(KP)(LN)(MX)(QY)(RS), \\ \gamma_6 & = & (AL)(BG)(CO)(DV)(EN)(FS)(HY)(IZ)(JT)(KW)(MP)(QR)(UX), \\ \gamma_7 & = & (AI)(BL)(CT)(DE)(FN)(GH)(JY)(KZ)(MO)(PS)(QX)(RU)(VW), \\ \gamma_8 & = & (AC)(BH)(DU)(EM)(FQ)(GV)(IO)(JZ)(KS)(LT)(NR)(PX)(WY), \\ \gamma_9 & = & (AB)(CM)(DY)(EZ)(FG)(HN)(IR)(JX)(KV)(LW)(OT)(PQ)(SU). \end{array}$$

We now proceed as follows: suppose we know that exactly six plugs are being used. This means that if we pick a letter at random, say T, then there is a $14/26 = 0.53$ chance that this letter is not plugged to another one. Let us therefore make this assumption for the letter T, in which case $p_T = T$. From the above equations involving γ_1 and γ_5 we then deduce that

$$p_U = U \text{ and } p_D = C.$$

We then use the equations involving γ_3 and γ_8, since we now know p_D, to deduce that

$$p_A = B \text{ and } p_B = A.$$

These last two checks are consistent, so we can assume that our original choice of $p_T = T$ was a good one. From the equations involving γ_6, using $p_A = B$ we deduce that

$$p_H = G.$$

Using this in the equations involving γ_0 and γ_9 we deduce that

$$p_I = J \text{ and } p_E = F.$$

We then find that our five plug settings of $A \leftrightarrow B$, $C \leftrightarrow D$, $E \leftrightarrow F$, $G \leftrightarrow H$ and $I \leftrightarrow J$ allow us to decrypt the first ten letters correctly. To deduce the final plug setting will require a longer piece of ciphertext, and the corresponding piece of known plaintext.

This technique can also be used when one knows partial information about the rotor positions. For example, many of the following techniques will allow us to deduce the differences $p_i - r_i$, but not the actual values of r_i or p_i. However, by applying the above technique, on assuming $r_i =$'A', we will at some point deduce a contradiction. At this point we know that either a rotor turnover has occurred incorrectly, or one has not occurred when it should have done. Hence, we can at this point backtrack and deduce the correct turnover. For an example of this technique at work see the later section on the Bombe.

Technique Two: A second method is possible when fewer than thirteen plugs are used. In the plaintext obtained under γ_j a number of incorrect letters will appear. Again we let m denote the actual plaintext and m' the plaintext derived under the current (possibly empty) plugboard setting. We suppose that there are t plugs left to find.

Suppose we concentrate on each place for which the incorrect plaintext letter A occurs, i.e. all occurrences of A in the plaintext m which are wrong in m'. Let x denote the corresponding ciphertext letter. There are two possible cases which can occur

- The letter x should be plugged to an unknown letter. In this case the resulting letter in the message m' will behave randomly (assuming γ_j acts like a random permutation).
- The letter x does not occur in a plugboard setting. In this case the resulting incorrect plaintext character is the one which should be plugged to A in the actual cipher.

Assuming ciphertext letters are uniformly distributed, the first case will occur with probability $t/13$, whilst the alternative will occur with probability $1 - t/13$. This gives the following method for determining the letter to which A should be connected. For all occurances of A in the plaintext m compute the frequency of the corresponding letter in the approximate plaintext m'. The one with the highest frequency is highly likely to be the one which should be connected to A on the plugboard. Indeed we expect the proportion of such positions with the correct letter to be given by $1 - t/13$, whilst all other letters we expect to occur in proportions of $t/(13 \cdot 26)$ each.

The one problem with this second technique is that it requires a relatively large amount of known plaintext. Hence, in practice the first technique is more likely to be used.

Knowledge of ϵ_j for Some js: If we know the value of the permutation ϵ_j for values of $j \in \mathcal{S}$, then we have the following equation

$$\epsilon_j = \tau \cdot \gamma_j \cdot \tau \text{ for } j \in \mathcal{S}.$$

Since $\tau = \tau^{-1}$ we can thus compute possible values of τ using our previous method for solving this conjugation problem. This might not determine the whole plugboard but it will determine enough for other methods to be used.

Knowledge of $\epsilon_j \cdot \epsilon_{j+3}$ for Some js: A similar method to the previous one applies in this case as well, since, if we know $\epsilon_j \cdot \epsilon_{j+3}$ for all $j \in \mathcal{S}$ and we know γ_j, we can use the equation

$$(\epsilon_j \cdot \epsilon_{j+3}) = \tau \cdot (\gamma_j \cdot \gamma_{j+3}) \cdot \tau \text{ for } j \in \mathcal{S}.$$

The utility of a method upon knowing $\epsilon_j \cdot \epsilon_{j+3}$ will become apparent in a little while.

8.4. Double Encryption of Message Keys

The Polish mathematicians Jerzy Różycki, Henryk Zygalski and Marian Rejewski were the first to find ways of analysing the Enigma machine. To understand their methods one must first understand how the Germans used the machine. On each day the machine was set up with a key, as above, which was chosen by looking in a code book; each subnet would have a different day key.

To encipher a message the sending operator decided on a message key. The message key would be a sequence of three letters, say DHI, which would need to be transported to the recipient. Using the day key, the message key would be enciphered twice. The double enciphering was to act as a

form of error control. Hence, DHI might be enciphered as $XHJKLM$. Note that D encrypts first to X and then to K.

The receiver would obtain $XHJKLM$ and then decrypt this to obtain DHI. Both operators would then move the wheels around to the positions D, H and I, i.e. they would turn the wheels so that D was in the leftmost window, H in the middle one and I in the rightmost window. Then the actual message would be enciphered. For this example, in our notation, this would mean that the message key was equal to the day key, except that $p_1 = 8$, i.e. I, $p_2 = 7$, i.e. H and $p_3 = 3$, i.e. D.

Suppose we intercept a set of messages which have the following headers, consisting of the encryption of the three-letter rotor positions, followed by its encryption again, i.e. the first six letters of each message are equal to

UCWBLR	ZSETEY	SLVMQH	SGIMVW	PMRWGV
VNGCTP	OQDPNS	CBRVPV	KSCJEA	GSTGEU
DQLSNL	HXYYHF	GETGSU	EEKLSJ	OSQPEB
WISIIT	TXFEHX	ZAMTAM	VEMCSM	LQPFNI
LOIFMW	JXHUHZ	PYXWFQ	FAYQAF	QJPOUI
EPILWW	DOGSMP	ADSDRT	XLJXQK	BKEAKY
.....	.....	.....	.....	.....
DDESRY	QJCOUA	JEZUSN	MUXROQ	SLPMQI
RRONYG	ZMOTGG	XUOXOG	HIUYIE	KCPJLI
DSESEY	OSPPEI	QCPOLI	HUXYOQ	NYIKFW

Let us take the last one of these and look at it in more detail. We know that there are three underlying secret letters, say l_1, l_2 and l_3. We also know that

$$ {l_1}^{\epsilon_0} = N,\ {l_2}^{\epsilon_1} = Y,\ {l_3}^{\epsilon_2} = I, $$

and

$$ {l_1}^{\epsilon_3} = K,\ {l_2}^{\epsilon_4} = F,\ {l_3}^{\epsilon_5} = W. $$

Hence, given that ${\epsilon_j}^{-1} = \epsilon_j$, we have

$$ N^{\epsilon_0\epsilon_3} = {l_1}^{\epsilon_0\epsilon_0\epsilon_3} = {l_1}^{\epsilon_3} = K,\ Y^{\epsilon_1\epsilon_4} = F,\ I^{\epsilon_2\epsilon_5} = W. $$

Continuing in this way we can compute a permutation representation of the three products as follows:

$$ \begin{aligned} \epsilon_0 \cdot \epsilon_3 &= (ADSMRNKJUB)(CV)(ELFQOPWIZT)(HY), \\ \epsilon_1 \cdot \epsilon_4 &= (BPWJUOMGV)(CLQNTDRYF)(ES)(HX), \\ \epsilon_2 \cdot \epsilon_5 &= (AC)(BDSTUEYFXQ)(GPIWRVHZNO)(JK). \end{aligned} $$

8.5. Determining the Internal Rotor Wirings

However, life was even more difficult for the Polish analysts as they did not even know the rotor wirings or the reflector values. Hence, they needed to break the scheme without even having a description of the actual machine. They did at least have access to a non-military version of Enigma and deduced the basic structure. In this they had two bits of luck:

(1) They deduced that the wiring between the plugboard and the rightmost rotor was in alphabetical order. Had this not been the case they would have needed to find an additional, hidden permutation.
(2) Secondly, the French cryptographer Gustave Bertrand obtained from a German spy, Hans-Thilo Schmidt, two months' worth of day keys. Thus, for two months of traffic the Poles had access to the day settings.

From this information they needed to deduce the internal wirings of the Enigma machine.

Note that in the pre-war days the Germans only used three wheels out of a choice of three, hence the number of day keys is actually reduced by a factor of ten. This is, however, only a slight simplification (at least with modern technology). To explain the method suppose we are given that the day setting is

Rotors	Rings	Pos	Plugboard
III, II, I	TXC	EAZ	$(AMTEBC)$

We do not know what the actual rotors are at present, but we know that the one labelled rotor I will be placed in the rightmost slot (our label one). So we have

$$r_1 = 2,\ r_2 = 23,\ r_3 = 19, p_1 = 25,\ p_2 = 0,\ p_3 = 4.$$

Suppose also that the data from the previous section was obtained as traffic for that day. Hence, we obtain the following three values for the products $\epsilon_j \cdot \epsilon_{j+1}$,

$$\begin{aligned}
\epsilon_0 \cdot \epsilon_3 &= (ADSMRNKJUB)(CV)(ELFQOPWIZT)(HY),\\
\epsilon_1 \cdot \epsilon_4 &= (BPWJUOMGV)(CLQNTDRYF)(ES)(HX),\\
\epsilon_2 \cdot \epsilon_5 &= (AC)(BDSTUEYFXQ)(GPIWRVHZNO)(JK).
\end{aligned}$$

From these we wish to deduce the values of $\epsilon_0, \epsilon_1, \ldots, \epsilon_5$. We will use the fact that ϵ_j is a product of disjoint transpositions and Theorem 8.2.

We take the first product and look at it in more detail. We take the sets of two cycles of equal degree and write them above one another, with the bottom one reversed in order, i.e.

$$\begin{array}{cccccccccc|cc}
A & D & S & M & R & N & K & J & U & B & C & V\\
T & Z & I & W & P & O & Q & F & L & E & Y & H
\end{array}$$

We now run through all possible shifts of the bottom rows. Each shift gives us a possible value of ϵ_0 and ϵ_3. The value of ϵ_0 is obtained from reading off the disjoint transpositions from the columns, the value of ϵ_3 is obtained by reading off the transpositions from the "off diagonals". For example with the above orientation we would have

$$\begin{aligned}
\epsilon_0 &= (AT)(DZ)(SI)(MW)(RP)(NO)(KQ)(JF)(UL)(BE)(CY)(VH),\\
\epsilon_3 &= (DT)(SZ)(MI)(RW)(NP)(KO)(JQ)(UF)(BL)(AE)(VY)(CH).
\end{aligned}$$

This still leaves us, in this case, with $20 = 2 \cdot 10$ possible values for ϵ_0 and ϵ_3.

Now, to reduce this number we need to rely on operational errors by German operators. Various operators had a tendency to always select the same three letter message key. For example, popular choices where QWE (the first letters on the keyboard). One operator used the letters of his girlfriend's name, Cillie, hence such "cribs" (or guessed/known plaintexts in today's jargon) became known as "cillies". Note, for our analysis here we only need one cillie for the day when we wish to obtain the internal wiring of rotor I.

In our dummy example, suppose we guess (correctly) that the first message key is indeed QWE, and that $UCWBLR$ is the encryption of QWE twice. This in turn tells us how to align our cycle of length 10 in the first permutation, as under ϵ_0 the letter Q must encrypt to U.

$$\begin{array}{cccccccccc}
A & D & S & M & R & N & K & J & U & B\\
L & E & T & Z & I & W & P & O & Q & F
\end{array}$$

We can check that this is consistent as we see that Q under ϵ_3 must then encrypt to B. Assuming we carry on in this way we will finally deduce that

$$\begin{aligned}
\epsilon_0 &= (AL)(BF)(CH)(DE)(GX)(IR)(JO)(KP)(MZ)(NW)(QU)(ST)(VY),\\
\epsilon_1 &= (AK)(BQ)(CW)(DM)(EH)(FJ)(GT)(IZ)(LP)(NV)(OR)(SX)(UY),\\
\epsilon_2 &= (AJ)(BN)(CK)(DZ)(EW)(FP)(GX)(HS)(IY)(LM)(OQ)(RU)(TV),\\
\epsilon_3 &= (AF)(BQ)(CY)(DL)(ES)(GX)(HV)(IN)(JP)(KW)(MT)(OU)(RZ),\\
\epsilon_4 &= (AK)(BN)(CJ)(DG)(EX)(FU)(HS)(IZ)(LW)(MR)(OY)(PQ)(TV),\\
\epsilon_5 &= (AK)(BO)(CJ)(DN)(ER)(FI)(GQ)(HT)(LM)(PX)(SZ)(UV)(WY).
\end{aligned}$$

We now need to use this information to deduce the value of ρ_1, etc. So for the rest of this section we assume that we know the ϵ_j for $j = 0, \ldots, 5$, and so we mark them in blue. Recall that we have

$$\begin{aligned}
\epsilon_j \;=\; & \tau \cdot (\sigma^{i_1+j} \cdot \rho_1 \cdot \sigma^{-i_1-j}) \cdot (\sigma^{i_2} \cdot \rho_2 \cdot \sigma^{-i_2}) \cdot (\sigma^{i_3} \cdot \rho_3 \cdot \sigma^{-i_3}) \cdot \varrho\\
& \cdot (\sigma^{i_3} \cdot {\rho_3}^{-1} \cdot \sigma^{-i_3}) \cdot (\sigma^{i_2} \cdot {\rho_2}^{-1} \cdot \sigma^{-i_2}) \cdot (\sigma^{i_1+j} \cdot {\rho_1}^{-1} \cdot \sigma^{-i_1-j}) \cdot \tau
\end{aligned}$$

We now assume that no stepping of the second rotor occurs during the first six encryptions under the day setting. This holds with quite high probability, namely $20/26 \approx 0.77$. Should the assumption turn out to be false we will notice in our later analysis and it will mean that we can deduce something about the (unknown to us at this point) position of the notch on the first rotor.

Given that we know the day settings, including τ and the values of i_1, i_2 and i_3 (since we are assuming $k_1 = k_2 = 0$ for $0 \leq j \leq 5$), we can write the above equation for $0 \leq j \leq 5$ as

$$\begin{aligned}
\lambda_j &= \sigma^{-i_1-j} \cdot \tau \cdot \epsilon_j \cdot \tau \cdot \sigma^{i_1+j}\\
&= \rho_1 \cdot \sigma^{-j} \cdot \gamma \cdot \sigma^{j} \cdot {\rho_1}^{-1}.
\end{aligned}$$

Where λ_j is now known and we wish to determine ρ_1 for some fixed but unknown value of γ. The permutation γ is in fact equal to

$$\gamma = (\sigma^{i_2-i_1} \cdot \rho_2 \cdot \sigma^{-i_2}) \cdot (\sigma^{i_3} \cdot \rho_3 \cdot \sigma^{-i_3}) \cdot \varrho \cdot (\sigma^{i_3} \cdot {\rho_3}^{-1} \cdot \sigma^{-i_3}) \cdot (\sigma^{i_2} \cdot {\rho_2}^{-1} \cdot \sigma^{i_1-i_2}).$$

In our example we get the following values for λ_j:

$$\begin{aligned}
\lambda_0 &= (AD)(BR)(CQ)(EV)(FZ)(GP)(HM)(IN)(JK)(LU)(OS)(TW)(XY),\\
\lambda_1 &= (AV)(BP)(CZ)(DF)(EI)(GS)(HY)(JL)(KO)(MU)(NQ)(RW)(TX),\\
\lambda_2 &= (AL)(BK)(CN)(DZ)(EV)(FP)(GX)(HS)(IY)(JM)(OQ)(RU)(TW),\\
\lambda_3 &= (AS)(BF)(CZ)(DR)(EM)(GN)(HY)(IW)(JO)(KQ)(LX)(PV)(TU),\\
\lambda_4 &= (AQ)(BK)(CT)(DL)(EP)(FI)(GX)(HW)(JU)(MO)(NY)(RS)(VZ),\\
\lambda_5 &= (AS)(BZ)(CV)(DO)(EM)(FR)(GQ)(HK)(IL)(JT)(NP)(UW)(XY).
\end{aligned}$$

We now form, for $j = 0, \ldots, 4$,

$$\begin{aligned}
\mu_j &= \lambda_j \cdot \lambda_{j+1},\\
&= \rho_1 \cdot \sigma^{-j} \cdot \gamma \cdot \sigma^{-1} \cdot \gamma \cdot \sigma^{j+1} \cdot {\rho_1}^{-1},\\
&= \rho_1 \cdot \sigma^{-j} \cdot \delta \cdot \sigma^{j} \cdot {\rho_1}^{-1},
\end{aligned}$$

where $\delta = \gamma \cdot \sigma^{-1} \cdot \gamma \cdot \sigma$ is unknown. Eliminating δ via $\delta = \sigma^{j-1} \cdot {\rho_1}^{-1} \cdot \mu_{j-1}\rho_1 \cdot \sigma^{-j+1}$ we find the following equations for $j = 1, \ldots, 4$,

$$\begin{aligned}
\mu_j &= (\rho_1 \cdot \sigma^{-1} \cdot {\rho_1}^{-1}) \cdot \mu_{j-1} \cdot (\rho_1 \cdot \sigma \cdot {\rho_1}^{-1}),\\
&= \alpha \cdot \mu_{j-1} \cdot \alpha^{-1},
\end{aligned}$$

where $\alpha = \rho_1 \cdot \sigma^{-1} \cdot {\rho_1}^{-1}$. Hence, μ_j and μ_{j-1} are conjugate and so by Theorem 8.1 have the same cycle structure. For our example we have

$$\begin{aligned}
\mu_0 &= (AFCNE)(BWXHUJOG)(DVIQZ)(KLMYTRPS),\\
\mu_1 &= (AEYSXWUJ)(BFZNO)(CDPKQ)(GHIVLMRT),\\
\mu_2 &= (AXNZRTIH)(BQJEP)(CGLSYWUD)(FVMOK),\\
\mu_3 &= (ARLGYWFK)(BIHNXDSQ)(CVEOU)(JMPZT),\\
\mu_4 &= (AGYPMDIR)(BHUTV)(CJWKZ)(ENXQSFLO).
\end{aligned}$$

At this point we can check whether our assumption of no-stepping, i.e. a constant value for the values of i_2 and i_3, is valid. If a step did occur in the second rotor then the above permutations would be unlikely to have the same cycle structure.

We need to determine the structure of the permutation α; this is done by looking at the four equations simultaneously. We note that σ and α are conjugates, under ρ_1, and we know that α has cycle structure of a single cycle of length 26, since σ is the shift-left permutation. In our example we only find one possible solution for α, namely

$$\alpha = (AGYWUJOQNIRLSXHTMKCEBZVPFD).$$

To solve for ρ_1 we need to find a permutation such that

$$\alpha = \rho_1 \cdot \sigma^{-1} \cdot {\rho_1}^{-1}.$$

We find there are 26 such solutions

$$
\begin{array}{l}
(AELTPHQXRU)(BKNW)(CMOY)(DFG)(IV)(JZ)\\
(AFHRVJ)(BLU)(CNXSTQYDGEMPIW)(KOZ)\\
(AGFIXTRWDHSUCO)(BMQZLVKPJ)(ENY)\\
(AHTSVLWEOBNZMRXUDIYFJCPKQ)\\
(AIZN)(BOCQ)(DJ)(EPLXVMSWFKRYGHU)\\
(AJEQCRZODKSXWGI)(BPMTUFLYHVN)\\
(AKTVOER)(BQDLZPNCSYI)(FMUGJ)(HW)\\
(AL)(BR)(CTWI)(DMVPOFN)(ESZQ)(GKUHXYJ)\\
(AMWJHYKVQFOGLBS)(CUIDNETXZR)\\
(ANFPQGMX)(BTYLCVRDOHZS)(EUJI)(KW)\\
(AOIFQH)(BUKX)(CWLDPREVS)(GN)(MY)(TZ)\\
(APSDQIGOJKYNHBVT)(CX)(EWMZUL)(FR)\\
(AQJLFSEXDRGPTBWNIHCYOKZVUM)\\
(ARHDSFTCZWOLGQK)(BXEYPUNJM)\\
(ASGRIJNKBYQLHEZXFUOMC)(DT)(PVW)\\
(ATE)(BZYRJONLIKC)(DUPWQM)(FVXGSH)\\
(AUQNMEB)(DVYSILJPXHGTFWRK)\\
(AVZ)(CDWSJQOPYTGURLKE)(FXIM)\\
(AWTHINOQPZBCEDXJRMGV)(FYUSK)\\
(AXKGWUTIORNP)(BDYV)(CFZ)(HJSLM)\\
(AYWVCGXLNQROSMIPBEF)(DZ)(HK)(JT)\\
(AZEGYXMJUVD)(BF)(CHLOTKIQSNRP)\\
(BGZFCIRQTLPD)(EHMKJV)(NSOUWX)\\
(ABHNTMLQUXOVFDCJWYZG)(EISP)\\
(ACKLRSQVGBITNUY)(EJXPF)(HOWZ)\\
(ADEKMNVHPGCLSRTOXQW)(BJY)(IUZ)
\end{array}
$$

These are the values of $\rho_1 \cdot \sigma^i$, for $i = 0, \ldots, 25$. So with one day's messages we can determine the value of ρ_1 up to multiplication by a power of σ. The Polish had access to two months' such data and so were able to determine similar sets for ρ_2 and ρ_3 (as different rotor orders are used on different days). Note that, at this point, the Germans did not use a selection of three from five rotors.

If we select three representatives $\hat{\rho}_1$, $\hat{\rho}_2$ and $\hat{\rho}_3$, from the sets of possible rotors, then we have

$$
\begin{aligned}
\hat{\rho}_1 &= \rho_1 \cdot \sigma^{l_1},\\
\hat{\rho}_2 &= \rho_2 \cdot \sigma^{l_2},\\
\hat{\rho}_3 &= \rho_3 \cdot \sigma^{l_3}.
\end{aligned}
$$

However, we still do not know the value for the reflector ϱ, or the correct values of l_1, l_2 and l_3. To understand how to proceed we present the following theorem.

Theorem 8.3. *Consider an Enigma machine $\mathcal{E}$ that uses rotors ρ_1, ρ_2 and ρ_3, and reflector ϱ. Then there is an Enigma machine $\hat{\mathcal{E}}$ using rotors $\hat{\rho}_1$, $\hat{\rho}_2$ and $\hat{\rho}_3$, and a different reflector $\hat{\varrho}$ such that, for every setting of $\mathcal{E}$, there is a setting of $\hat{\mathcal{E}}$ such that the machines have identical behaviour.*

Furthermore, $\hat{\mathcal{E}}$ can be constructed so that the machines use identical day settings except for the ring positions.

PROOF. The following proof was shown to me by Eugene Luks; I thank him for allowing me to reproduce it here. The first claim is that $\hat{\varrho}$ is determined via

$$\hat{\varrho} = \sigma^{-(l_1+l_2+l_3)} \cdot \varrho \cdot \sigma^{-(l_1+l_2+l_3)}.$$

We can see this by the following argument (and the fact that the reflector is uniquely determined by the above equation). Define the following function

$$\begin{aligned} P(\phi_1, \phi_2, \phi_3, \psi, t_1, t_2, t_3) = \tau \cdot (\sigma^{t_1} \cdot \phi_1 \cdot \sigma^{-t_1}) \cdot (\sigma^{t_2} \cdot \phi_2 \cdot \sigma^{-t_2}) \cdot (\sigma^{t_3} \cdot \phi_3 \cdot \sigma^{-t_3}) \cdot \psi \\ \cdot (\sigma^{t_3} \cdot \phi_3^{-1} \cdot \sigma^{-t_3}) \cdot (\sigma^{t_2} \cdot \phi_2^{-1} \cdot \sigma^{-t_2}) \cdot (\sigma^{t_1} \cdot \phi_1^{-1} \cdot \sigma^{-t_1}) \cdot \tau \end{aligned}$$

We then have the relation,

$$P(\hat{\rho_1}, \hat{\rho_2}, \hat{\rho_3}, \hat{\varrho}, t_1, t_2, t_3) = P(\rho_1, \rho_2, \rho_3, \varrho, t_1, t_2 + l_1, t_3 + l_1 + l_2).$$

Recall the following expressions for the functions which control the stepping of the three rotors:

$$\begin{aligned} k_1 &= \lfloor (j - m_1 + 26)/26 \rfloor, \\ k_2 &= \lfloor (j - m_2 + 650)/650 \rfloor, \\ i_1 &= p_1 - r_1 + 1, \\ i_2 &= p_2 - r_2 + k_1 + k_2, \\ i_3 &= p_3 - r_3 + k_2. \end{aligned}$$

The Enigma machine $\mathcal{E}$ is given by the equation

$$\epsilon_j = P(\rho_1, \rho_2, \rho_3, \varrho, i_1 + j, i_2, i_3)$$

where we interpret i_2 and i_3 as functions of j as above. We now set the ring positions in $\hat{\mathcal{E}}$ to be given by

$$r_1, r_2 + l_1, r_3 + l_1 + l_2$$

in which case we have that the output of this Enigma machine is given by

$$\hat{\epsilon}_j = P(\hat{\rho_1}, \hat{\rho_2}, \hat{\rho_3}, \hat{\varrho}, i_1 + j, i_2 - l_1, i_3 - l_1 - l_2).$$

But then we conclude that $\epsilon_j = \hat{\epsilon}_j$. □

We now use this result to fully determine $\mathcal{E}$ from the available data. We pick values of $\hat{\rho_1}$, $\hat{\rho_2}$ and $\hat{\rho_3}$ and determine a possible reflector by solving for $\hat{\varrho}$ in

$$\begin{aligned} \epsilon_0 = \tau \cdot (\sigma^{i_1} \cdot \hat{\rho_1} \cdot \sigma^{-i_1}) \cdot (\sigma^{i_2} \cdot \hat{\rho_2} \cdot \sigma^{-i_2}) \cdot (\sigma^{i_3} \cdot \hat{\rho_3} \cdot \sigma^{-i_3}) \cdot \hat{\varrho} \\ \cdot (\sigma^{i_3} \cdot \hat{\rho_3}^{-1} \cdot \sigma^{-i_3}) \cdot (\sigma^{i_2} \cdot \hat{\rho_2}^{-1} \cdot \sigma^{-i_2}) \cdot (\sigma^{i_1} \cdot \hat{\rho_1}^{-1} \cdot \sigma^{-i_1}) \cdot \tau \end{aligned}$$

We let $\hat{\mathcal{E}}^1$ denote the Enigma machine with rotors given by $\hat{\rho_1}, \hat{\rho_2}, \hat{\rho_3}$ and reflector $\hat{\varrho}$, but with ring settings the same as in the target machine $\mathcal{E}$ (we know the ring settings of $\mathcal{E}$ since we have the day key). Note that $\hat{\mathcal{E}}^1 \neq \hat{\mathcal{E}}$ from the above proof, since the rings are in the same place as in the target machine.

Assume we have obtained a long message, with a given message key. We put the machine $\hat{\mathcal{E}}^1$ in the message key configuration and start to decrypt the message. This will work (i.e. produce a valid decryption) up to the point when the sequence of permutations $\hat{\epsilon}_j^1$ produced by $\hat{\mathcal{E}}^1$ differs from the sequence ϵ_j produced by $\mathcal{E}$.

At this point we cycle through all values of l_1 and fix the first permutation (and also the associated reflector) to obtain a new Enigma machine $\hat{\mathcal{E}}^2$ which allows us to decrypt more of the long message. If a long enough message is obtained we can also obtain l_2 in this way, or alternatively

wait for another day when the rotor order is changed. Thus the entire internal workings of the Enigma machine can be determined.

8.6. Determining the Day Settings

Having now determined the internal wirings, given the set of two months of day settings obtained by Bertrand, the next task is to determine the actual key when the day settings are not available. At this stage we assume that the German operators are still using the "encrypt the message setting twice" routine. The essential trick here is to notice that if we write the cipher as

$$\epsilon_j = \tau \cdot \gamma_j \cdot \tau,$$

then

$$\epsilon_j \cdot \epsilon_{j+3} = \tau \cdot \gamma_j \cdot \gamma_{j+3} \cdot \tau.$$

So $\epsilon_j \cdot \epsilon_{j+3}$ is conjugate to $\gamma_j \cdot \gamma_{j+3}$ and so by Theorem 8.1 they have the same cycle structure. More importantly the cycle structure does not depend on the plugboard τ.

Hence, if we can use the cycle structure to determine the rotor settings then all that remains is to determine the plugboard settings. From the rotor settings we know the values of γ_j, for $j = 1, \ldots, 6$; from the encrypted message keys we can compute ϵ_j for $j = 1, \ldots, 6$ as in the previous section. Hence, the plugboard settings can be recovered by solving another of our conjugacy problems, for τ. This is easier than before as we have that τ must be a product of disjoint transpositions.

We have already discussed how to compute $\epsilon_j \cdot \epsilon_{j+3}$ from the encryption of the message keys. Hence, we *simply* compute these values and compare their cycle structures with those obtained by running through all possible

$$60 \cdot 26^3 \cdot 26^3 = 18\,534\,946\,560$$

choices for the rotors, positions and ring settings! Note that when this was done by the Polish analysts in the 1930s there was only a choice of the ordering of three rotors. The extra choice of rotors did not come in till a bit later. Hence, the total choice was 10 times less than this figure.

The above simplifies further if we assume that no stepping of the second and third rotor occurs during the calculation of the first six ciphertext characters. Recall this happens around seventy seven percent of the time. In such a situation the cycle structure depends only on the rotor order and the difference $p_i - r_i$ between the starting rotor position and the ring setting. Hence, we might as well assume that $r_1 = r_2 = r_3 = 0$ when computing all of the cycle structures. So, for seventy seven percent of days our search amongst the cycle structures is then only among

$$60 \cdot 26^3 = 1\,054\,560 \text{ (resp. } 105\,456)$$

possible cycle structures.

After the above procedure we have determined all values of the initial day setting bar p_i and r_i, however we know the differences $p_i - r_i$. We also know for any given message the message key p'_1, p'_2, p'_3. Hence, in breaking the actual message we only require the solution for r_1, r_2; the value for r_3 is irrelevant as the third rotor never moves a fourth rotor. Most German messages started with the same two-letter word followed by a space (space was encoded by 'X'). Hence, we only need to go through 26^2 different positions to get the correct ring setting. Actually one goes through 26^2 wheel positions with a fixed ring, and uses the differences to infer the true ring settings.

Once r_i is determined from one message the value of p_i can be determined for the day key and then all messages can be trivially broken. Another variant here, if a suitable piece of known plaintext can be deduced, is to apply the technique from Section 8.3.2 with the obvious modification to deduce the ring settings as well.

8.7. The Germans Make It Harder

In September 1938 the German operators altered the way that day and message keys were used. Now a day key consisted of a rotor order, the ring settings and the plugboard. But the rotor positions were not part of the day key. A cipher operator would now choose their own initial rotor positions, say AXE and their own message rotor positions, say GPI. The operator would put their machine in the AXE setting and then encrypt GPI twice as before, to obtain say $POWKNP$. The rotors would then be placed in the GPI position and the message would be encrypted. The message header would be $AXEPOWKNP$.

This procedure makes the analysis of the previous section useless, as each message would now have its own "day" rotor position setting, and so one could not collect data from many messages so as to recover $\epsilon_0 \cdot \epsilon_3$ etc. as in the previous section.

What was needed was a new way of characterizing the rotor positions. The strategy invented by Zygalski was to use so-called "females". In the six letters of the enciphered message key a female is the occurrence of the same letter in the same position in each substring of three. For example, the header $POWKNP$ contains no females, but the header $POWPNL$ contains one female in position zero, i.e. the repeated values of P, separated by three positions.

To consider what is implied by the existence of such females, firstly suppose we receive $POWPNL$ as above and that the unknown first key setting is x. Then we have that, if ϵ_i represents the Enigma machine in the day setting,

$$x^{\epsilon_0} = x^{\epsilon_3} = P,$$

that is

$$P^{\epsilon_0 \cdot \epsilon_3} = x^{\epsilon_0 \cdot \epsilon_0 \cdot \epsilon_3} = x^{\epsilon_3} = P.$$

In other words P is a fixed point of the permutation $\epsilon_0 \cdot \epsilon_3$.

Since the number of fixed points is a feature of the cycle structure and the cycle structure is invariant under conjugation, we see that the number of fixed points of $\epsilon_0 \cdot \epsilon_3$ is the same irrespective of the plugboard setting.

The use of such females was made easier by so-called Zygalski sheets. The following precomputation was performed, for each rotor order. An Enigma machine was set up with rings in position AAA and then, for each position A to Z of the third (leftmost) rotor a sheet was created. This sheet was a table of 51 by 51 squares, consisting of the letters of the alphabet repeated twice in each direction minus one row and column. A square was removed if the Enigma machine with first and second rotor with that row/column position had a fixed point in the permutation $\epsilon_0 \cdot \epsilon_3$. So for each rotor order there was a set of 26 sheets.

Note, we are going to use the sheets to compute the day ring setting, but the computation is done using different rotor positions but with a fixed ring setting. This is because it is easier with an Enigma machine to rotate the rotor positions than to change the ring settings. Then converting between ring and rotor settings is simple.

In fact, it makes sense to also produce a set of sheets for the permutation $\epsilon_1 \cdot \epsilon_4$ and $\epsilon_2 \cdot \epsilon_5$, as without these the number of keys found by the following method is quite large. Hence, for each rotor order we will have 26×3 perforated sheets. We now describe the method used by the Polish analysts when only three rotors were used (extending it to five rotors is simple but was time consuming at the time). We proceed via an example. Suppose a set of message headers are received in one day. From these we keep all those which possess a female in the part corresponding to the encryption of the message key. For example, we obtain the following message headers:

HUXTBPGNP	DYRHFLGFS	XTMRSZRCX	YGZVQWZQH
BILJWWRRW	QYRZXOZJV	SZYJPFBPY	MWIBUMWRM
YXMHCUHHR	FUGWINCIA	BNAXGHFGG	TLCXYUXYC
RELCOYXOF	XNEDLLDHK	MWCQOPQVN	AMQCZQCTR
MIPVRYVCR	MQYVVPVKA	TQNJSSIQS	KHMCKKCIL

LQUXIBFIV	NXRZNYXNV	AMUIXVVFV	UROVRUAWU
DSJVDFVTT	HOMFCSQCM	ZSCTTETBH	SJECXKCFN
UPWMQJMSA	CQJEHOVBO	VELVUOVDC	TXGHFDJFZ
DKQKFEJVE	SHBOGIOQQ	QWMUKBUVG	

Now assuming a given rotor order, say the rightmost rotor is rotor I, the middle one is rotor II and the leftmost rotor is rotor III, we remove all those headers which could have had a stepping action of the middle rotor in the first six encryptions. To compute these we take the third character of the above message headers, i.e. the position p_1 of the rightmost rotor in the encryption of the message key, and the position of the notch on the rightmost rotor assuming the rightmost rotor is I, i.e. $n_1 = 16$, i.e. Q. We compute the value of m_1 according to Section 8.2

$$m_1 = n_1 - p_1 - 1 \pmod{26}.$$

and remove all those for which

$$\lfloor (j - m_1 + 26)/26 \rfloor \neq 0 \text{ for } j = 0, 1, 2, 3, 4, 5.$$

This leaves us with the following message headers

HUXTBPGNP	DYRHFLGFS	YGZVQWZQH	QYRZXOZJV
SZYJPFBPY	MWIBUMWRM	FUGWINCIA	BNAXGHFGG
TLCXYUXYC	XNEDLLDHK	MWCQOPQVN	AMQCZQCTR
MQYVVPVKA	LQUXIBFIV	NXRZNYXNV	AMUIXVVFV
DSJVDFVTT	ZSCTTETBH	SJECXKCFN	UPWMQJMSA
CQJEHOVBO	TXGHFDJFZ	DKQKFEJVE	SHBOGIOQQ

We now consider each of the three sets of females in turn. For ease of discussion we only consider those corresponding to $\epsilon_0 \cdot \epsilon_3$. We therefore only examine those message headers which have the same letter in the fourth and seventh positions, i.e.

QYRZXOZJV	TLCXYUXYC	XNEDLLDHK	MWCQOPQVN
AMQCZQCTR	MQYVVPVKA	DSJVDFVTT	ZSCTTETBH
SJECXKCFN	UPWMQJMSA	SHBOGIOQQ	

We now perform the following operation, for each letter P_3. We take the Zygalski sheet for rotor order III, II, I, permutation $\epsilon_0 \cdot \epsilon_3$ and letter P_3 and we place this down on the table. We think of this first sheet as corresponding to the ring setting

$$r_3 = Q - Q = A,$$

where the Q comes from the first letter in the first message header, QYRZXOZJV. Each row r and column c of the first sheet corresponds to the ring setting

$$\begin{aligned} r_1 &= R - r, \\ r_2 &= Y - c. \end{aligned}$$

We now repeat the following process for each message header with a first letter which we have not met before. We take the first letter of the next message header, TLCXYUXYC, in this case T, and we take the sheet with label

$$P_3 + T - Q.$$

This sheet then has to be placed on top of the other sheets at a certain offset to the original sheet. The offset is computed by taking the top leftmost square of the new sheet and placing it on top of the square (r, c) of the first sheet where

$$\begin{aligned} r &= R - C, \\ c &= Y - L, \end{aligned}$$

i.e. we take the difference between the third (resp. second) letter of the new message header and the third (resp. second) letter of the first message header.

This process is repeated until all of the given message headers are used up. Any square which is now clear on all sheets then gives a possible setting for the rings for that day. The actual setting can be read off the first sheet using the correspondence above.

This process will give a relatively large number of possible ring settings for each possible rotor order. However, when we intersect the possible values obtained from considering the females in the 0/3 position with those in the 1/4 and the 2/5 positions we find that the number of possibilities shrinks dramatically. Often this allows us to uniquely determine the rotor order and ring setting for the day. We determine in our example that the rotor order is given by III, II and I, with ring settings given by $r_1 = A$, $r_2 = B$ and $r_3 = C$.

To determine the plugboard settings for the day we can either use a piece of known plaintext as before, or, if no such text is available, we can use the females to help drastically reduce the number of possibilities for the plugboard settings.

8.8. Known Plaintext Attack and the Bombes

Turing (among others) wanted a technique to break Enigma which did not rely on the way the German military used the system, which could and did change. Turing settled on a known plaintext attack, using what was known at the time as a "crib". A crib was a piece of plaintext which was suspected to lie in the given piece of ciphertext.

The methodology of this technique was, from a given piece of ciphertext and a suspected piece of corresponding plaintext, to first deduce a so-called "menu". A menu is simply a graph which represents the various relationships between ciphertext and plaintext letters. Then the menu was used to program an electro-mechanical device, called a Bombe. A Bombe was a device which enumerated the Enigma wheel positions and, given the data in the menu, deduced the possible settings for the rotor orders, wheel positions and some of the plugboard. Finally, the ring positions and the remaining parts of the plugboard needed to be found.

In the following we present a version of this technique which we have deduced from various sources. We follow a running example through so as to explain the method in more detail.

From Ciphertext to a Menu: Suppose we receive the following ciphertext

```
HUSVTNXRTSWESCGSGVXPLQKCEYUHYMPBNUITUIHNZRS
```

and that we know, for example because we suspect it to be a shipping forecast, that the ciphertext encrypts at some point the plaintext[1]

```
DOGGERFISHERGERMANBIGHTEAST
```

Now, we know that in the Enigma machine, a letter cannot decrypt to itself. This means that there are only a few positions for which the plaintext will align correctly with the ciphertext. Consider the following alignment:

```
HUSVTNXRTSWESCGSGVXPLQKCEYUHYMPBNUITUIHNZRS
-DOGGERFISHERGERMANBIGHTEAST---------------
```

then we see that this is impossible since the S in the plaintext FISHER cannot correspond to the S in the ciphertext. Continuing in this way we find that there are only six possible alignments of the plaintext fragment with the ciphertext:

```
HUSVTNXRTSWESCGSGVXPLQKCEYUHYMPBNUITUIHNZRS
DOGGERFISHERGERMANBIGHTEAST----------------
---DOGGERFISHERGERMANBIGHTEAST-------------
-----DOGGERFISHERGERMANBIGHTEAST-----------
--------DOGGERFISHERGERMANBIGHTEAST--------
----------DOGGERFISHERGERMANBIGHTEAST------
```

[1]This plaintext refers to sea regions in the BBC shipping weather forecast.

----------DOGGERFISHERGERMANBIGHTEAST

In the following we will focus on the first alignment, i.e. we will assume that the first ciphertext letter H decrypts to D and so on. In practice the correct alignment out of all the possible ones would need to be deduced by skill, judgement and experience. However, in any given day a number of such cribs would be obtained and so only the most likely ones would be accepted for use in the following procedure.

As is usual with all our techniques there is a problem if the middle rotor turns over in the part of the ciphertext which we are considering. Our piece of chosen plaintext is 27 letters long, so we could treat it in two sections of 13 letters (and drop the last letter). The advantage of this is that we know the middle rotor will only advance once every 26 turns of the first rotor. Hence, by selecting two groups of roughly 13 letters we can obtain two possible alignments, one of which we know does not contain a middle rotor movement.

We therefore concentrate on the following two alignments:

HUSVTNXRTSWESCGSGVXPLQKCEYUHYMPBNUITUIHNZRS
DOGGERFISHERG----------
----------ERMANBIGHTEAS----------

We now deal with each alignment in turn and examine the various pairs of letters. We note that if H encrypts to D in the first position then D will encrypt to H in the same Enigma configuration. We make a record of the letters and the positions for which one letter encrypts to the other. These are placed in a graph with vertices corresponding to the letters and edges being labelled by the positions of the related encryptions. This results in the two graphs (or menus) given in Figures 8.2 and 8.3:

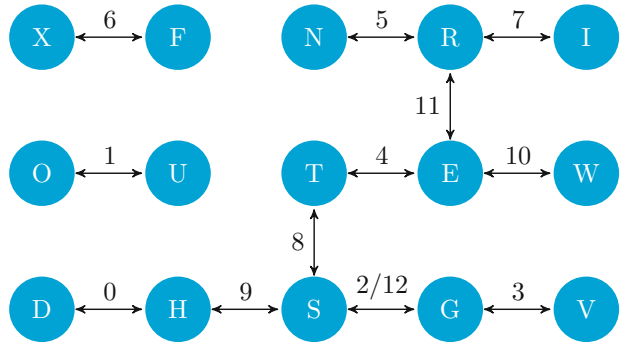

FIGURE 8.2. Menu 1

These menus tell us a lot about the configuration of the Enigma machine, in terms of its underlying permutations. Each menu is then used to program a Bombe. In fact we program one Bombe not only for each menu, but also for each possible rotor order. Thus if five rotor orders are in use, we need to program $2 \cdot 60 = 120$ such Bombes.

8.8.1. The Turing/Welchman Bombe: There are many descriptions of the Bombe as an electrical circuit. In the following we present the basic workings of the Bombe in terms of a modern computer; note however that in practice this is not very efficient. The Bombe's electrical circuit was able to execute the basic operations at the speed of light (i.e. the time it takes for a current to pass around a circuit), hence simulating this with a modern computer is very inefficient. The Bombe was in fact an electro-mechanical computer which was designed to perform a specific task – namely determining the wheel settings of the Enigma machine given one of the menus described

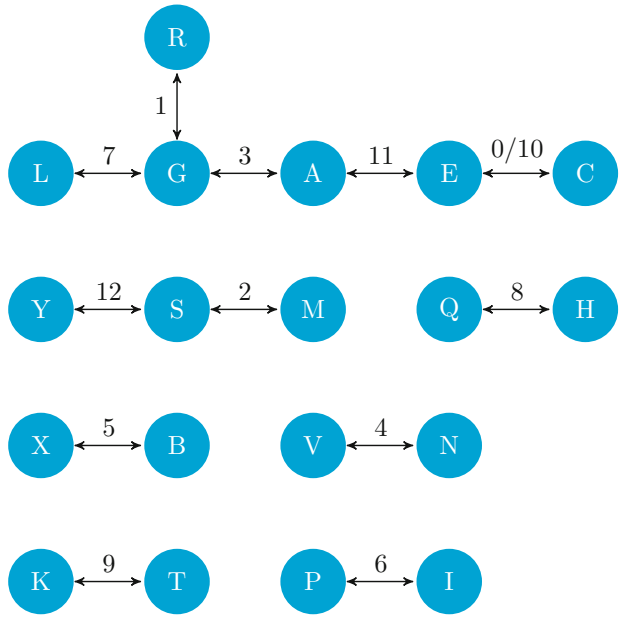

FIGURE 8.3. Menu 2

above. The initial design was made by Turing, but Welchman contributed a vital component (called the diagonal board) which made the process very efficient.

I have found that the best way to think of the Bombe is as a computer with 26 registers, each of which is 26 bits in length. In a single "step" of the Bombe a single bit in one register bank is set. Say we set bit F of register H; this corresponds to us wishing to test whether F is plugged to H in the actual Enigma configuration. This testing is done with the wheels in a given specific location. The Bombe then passes through a series of states until it stabilizes. In the actual Bombe this occurs at the speed of light; in a modern computer simulation it needs to be actually programmed and so occurs at the speed of a computer. Once the register bank stabilizes, each bit that is set means that if the tested condition is true, and the wheels are in the correct position, then so must this condition be true, i.e. if bit J of register K is set then J should be plugged to K in the Enigma machine. If we obtain a contradiction, then either the initial tested condition is false, or the wheels are in the wrong position. Continuing in this way with different wheel settings we either deduce the correct wheel settings, or deduce that the initial tested condition is false. Whilst the test of a specific configuration in the real Bombe happens at the speed of light, the moving to the next wheel setting happens mechanically. Thus despite, the test being faster on the Bombe than on a modern computer, the modern computer can outperform the Bombe for the overall algorithm.

In other words the Bombe deduces a "Theorem" of the form

$$\text{If } (F \to H \text{ and the wheels are in position } \mathcal{P}) \text{ Then } K \to J.$$

With this interpretation the diagonal board detailed in descriptions of the Bombe is then the obvious condition that if K is plugged to J, then J is also plugged to K, i.e. if bit J of register K is set, then so must be bit K of register J. In the real Bombe this is achieved using wires, however in a computer simulation it means that we always set the "transpose" bit when setting any bit in our register bank. Thus, the register bank is symmetric down the leading diagonal. The diagonal board drastically increases the usefulness of the Bombe in breaking arbitrary cribs.

To understand how the menu acts on the set of registers we define the following permutation for $0 \leq i < 26^3$, for a given choice of rotors ρ_1, ρ_2 and ρ_3. We write $i = i_1 + i_2 \cdot 26 + i_3 \cdot 26^2$, and define

$$\begin{aligned}\delta_{i,s} &= (\sigma^{i_1+s+1} \cdot \rho_1 \cdot \sigma^{-i_1-s-1}) \cdot (\sigma^{i_2} \cdot \rho_2 \cdot \sigma^{-i_2}) \cdot (\sigma^{i_3} \cdot \rho_3 \cdot \sigma^{-i_3}) \cdot \varrho \\ &\quad \cdot (\sigma^{i_3} \cdot {\rho_3}^{-1} \cdot \sigma^{-i_3}) \cdot (\sigma^{i_2} \cdot {\rho_2}^{-1} \cdot \sigma^{-i_2}) \cdot (\sigma^{i_1+s+1} \cdot {\rho_1}^{-1} \cdot \sigma^{-i_1-s-1}).\end{aligned}$$

Note how similar this is to the equation of the Enigma machine. The main difference is that the second and third rotors cycle through at a different rate (depending only on i). The variable i is used to denote the rotor position which we currently with to test and the variable s is used to denote the action of the menu, as we shall now describe.

The menu acts on the registers as follows: for each link $x \xrightarrow{s} y$ in the menu we take register x and for each set bit x_z we apply $\delta_{i,s}$ to obtain x_w. Then the bit x_w is set in register y and (due to the diagonal board) bit y is set in register x_w. We also need to apply the link backwards, so for each set bit y_z in register y we apply $\delta_{i,s}$ to obtain y_w. Then bit y_w is set in register x and (again due to the diagonal board) bit x is set in register y_w.

We now let l denote the letter which satisfies at least one of the following, and we hope all three:

(1) A letter which occurs more often than any other letter in the menu.
(2) A letter which occurs in more cycles than any other letter.
(3) A letter which occurs in the largest connected component of the graph of the menu.

In the above two menus we several letters to choose from in Menu 1, so we select $l = S$; in Menu 2 we select $l = E$. For each value of i we then perform the following operation

- Unset all bits in the registers.
- Set bit l of register l.
- Keep applying the menu, as above, until the registers no longer change at all.

Hence, the above algorithm is working out the consequences of the letter l being plugged to itself, given the choice of rotors ρ_1, ρ_2 and ρ_3. It is the third line in the above algorithm which operates at the speed of light in the real Bombe. In a modern simulation this takes a lot longer.

After the registers converge to a steady state we then test them to see whether a possible value of i, i.e. a possible value of the rotor position has been found. We then step i on by one, which in the real Bombe is achieved by rotating the rotors, and repeat. A value of i which corresponds to a valid value of i is called a "Bombe Stop".

To see what is a valid value of i, suppose we have the rotors in the correct positions. If the plugboard hypothesis that the letter l is plugged to itself is true, then the registers will converge to a state which gives the plugboard settings for the registers in the graph of the menu which are connected to the letter l. If, however, the plugboard hypothesis is wrong then the registers will converge to a different state, in particular the bit of each register which corresponds to the correct plugboard configuration will never be set. The best we can then expect is that this wrong hypothesis propagates and all registers in the connected component become set with 25 bits. The one remaining unset bit then corresponds to the correct plugboard setting for the letter l. If the rotor position is wrong then it is highly likely that all the bits in the test register l converge to the set position.

To summarize, we have one of the following situations upon convergence of the registers at step i:

- All 26 bits of test register l are set. This implies that the rotors are not in the correct position and we can step on i by one and repeat the whole process.
- One bit of test register l is set, the rest being unset. This is a possible correct configuration for the rotors. If it is indeed the correct configuration then, in addition, the set bit corresponds to the correct plug setting for register l, and the single bit set in the registers

corresponding to the letters connected to l in the menu will give us the plug settings for those letters as well.

- One bit of the test register l is unset, the rest being set. This is also a possible correct configuration for the rotors. If it is indeed the correct configuration then, in addition, the unset bit corresponds to the correct plug setting for register l, and any single unset bit in the registers corresponding to the letters connected to l in the menu will give us the plug settings for those letters as well.
- The number of set bits in register l lies in $[2, \ldots, 24]$. These are relatively rare occurrences, and although they could correspond to actual rotor settings they tell us little directly about the plug settings. The problem could be because the initial plug hypothesis is false. For "good" menus we find that stops like this are very rare indeed.

A Bombe stop is a position where the machine decides that it has reached a possibly correct configuration of the rotors. The number of such stops per rotor order depends on the structure of the graph of the menu. Turing determined the expected number of stops for different types of menus. The following table shows the expected number of stops per rotor order for a connected menu (i.e. only one component) with various numbers of letters and cycles.

	Number of Letters								
Cycles	8	9	10	11	12	13	14	15	16
3	2.2	1.1	0.42	0.14	0.04	≈ 0	≈ 0	≈ 0	≈ 0
2	58	28	11	3.8	1.2	0.3	0.06	≈ 0	≈ 0
1	1500	720	280	100	31	7.7	1.6	0.28	0.04
0	40000	19000	7300	2700	820	200	43	7.3	1.0

This also gives an upper bound on the expected number of stops for an unconnected menu in terms of the size of the largest connected component and the number of cycles within the largest connected component.

Hence, a good menu is not only one which has a large connected component but which also has a number of cycles. Our second example menu is particularly poor in this respect. Note that a large number of letters in the connected component not only reduces the expected number of Bombe stops but also increases the number of deductions about possible plugboard configurations.

8.8.2. Bombe Stop to Plugboard: We now need to work out how from a Bombe stop we can either deduce the actual key, or deduce that the stop has occurred simply by chance and does not correspond to a correct configuration. We first sum up how many stops there are in our example above. For each menu we specify, in the following table, the number of Bombe stops which arise and we also specify the number of bits in the test register l which gave rise to the stop.

	Number of Bits Set									
Menu	1	2	3	4	5-20	21	22	23	24	25
1	137	0	0	0	0	0	0	0	9	1551
2	2606	148	9	2	0	2	7	122	2024	29142

Here we can see the effect of the difference in size of the largest connected component. In both menus the largest connected component has a single cycle in it; in both cases being an edge with two different labels. For the first menu we obtain a total of 1697 stops, or 28.3 stops per rotor order. The connected component has eleven letters in it, so this yield is much better than the yield expected from Turing's earlier table. This is due to the extra two-letter component in the graph of Menu One. For Menu Two we obtain a total of 34 062 stops, or 567.7 stops per rotor order. The connected component in the second menu has six letters in it, so although this figure is bad it is in

fact better than the maximum expected from Turing's table. Again this is due to the presence of other components in the graph.

Given the large number of stops we need a way of automating the checking process. It turns out that this is relatively simple as the state of the registers allow other conditions to be checked automatically. Recall that the Bombe stop also gives us information about the state of the supposed plugboard. The following are so-called "legal contradictions", which can be eliminated instantly from the above stops, assuming the initial plug supposition is correct:

- If *any* Bombe register has 26 bits set then this Bombe configuration is impossible.
- If the Bombe registers imply that a letter is plugged to two different letters then this is clearly a contradiction.

Suppose we know that the plugboard uses a certain number of plugs (in our example this number is ten); if the registers imply that there are more than this number of plugs then this is also a contradiction. Applying these conditions means we are down to only 19 750 possible Bombe stops out of the 35 759 total stops above. Of these, 109 correspond to the first menu and the rest correspond to the second menu.

We clearly cannot cope with all of those corresponding to the second menu so let's suppose that the second rotor does not turn over in the first thirteen characters. This means we now only need to focus on the first menu. In practice a number of configurations could be eliminated due to operational requirements imposed on the German operators (e.g. not using the same rotor order on consecutive days).

8.8.3. Finding the Final Part of the Key: We will focus on the first two remaining stops for the first menu. Both of these correspond to rotor orders where the rightmost (fastest) rotor is rotor I, the middle one is rotor II and the leftmost rotor is rotor III.

The first remaining stop is at Bombe configuration $i_1 = p_1 - r_1 = Y$, $i_2 = p_2 - r_2 = W$ and $i_3 = p_3 - r_3 = K$. These follow from the following final register state in this configuration given in Table 8.1, where rows represent registers and columns the bits. The test register S has 25 bits

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	0	0	0	1	1	0	1	1	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	0	0
B	0	0	0	1	1	0	1	1	1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0
C	0	0	0	0	1	1	1	1	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	1	0	0
D	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
E	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F	0	0	1	1	1	1	1	1	1	0	0	1	1	0	0	0	0	1	1	1	0	1	1	1	1	0
G	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
H	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
I	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
J	0	0	0	1	1	0	1	0	1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0
K	0	0	0	1	1	0	1	1	1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0
L	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	0	1	1	1	1	0	0	0
M	0	0	0	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	1	1	1	1	1	0	0	0
N	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
O	0	1	0	1	1	0	1	1	1	1	1	1	0	1	0	0	1	1	1	1	1	0	1	1	1	1
P	0	0	0	1	1	0	1	1	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	0	0
Q	0	0	0	1	1	0	1	1	1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0
R	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
S	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
T	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
U	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	0	0	1	1
V	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
W	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
X	0	0	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	1	0	1	1	1	1	0
Y	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1	0	0
Z	0	0	0	1	1	0	1	1	1	0	0	0	0	1	1	0	0	1	1	0	1	1	1	0	0	0

TABLE 8.1. The registers on the first Bombe stop

set, so in this configuration each bit of the test register implies that this letter is not plugged to another letter. The plugboard setting is deduced to contain the plugs

$$C \leftrightarrow D,\ E \leftrightarrow I,\ F \leftrightarrow N,\ H \leftrightarrow J,\ L \leftrightarrow S,$$
$$M \leftrightarrow R,\ O \leftrightarrow V,\ T \leftrightarrow Z,\ U \leftrightarrow W,$$

whilst the letter G is known to be plugged to itself, assuming this is the correct configuration.

So we need to find one other plug and the ring settings. We can assume that $r_3 = 0 = A$ as it plays no part in the actual decryption process. Since we are using the rotor I as the rightmost rotor we know that $n_1 = 16$, i.e. Q, which, combined with the fact that we are assuming that no stepping occurs in the first thirteen characters, implies that p_1 must satisfy

$$j - ((16 - p_1 - 1) \pmod{26}) + 26 \leq 25 \text{ for } j = 0, \ldots, 12.$$

i.e. $p_1 = 0, 1, 2, 16, 17, 18, 19, 20, 21, 22, 23, 24$ or 25.

With the Enigma setting of $p_1 = Y$, $p_2 = W$, $p_3 = K$ and $r_1 = r_2 = r_3 = A$ and the above (incomplete) plugboard we decrypt the fragment of ciphertext and compare the resulting plaintext with the crib.

```
HUSVTNXRTSWESCGSGVXPLQKCEYUHYMPBNUITUIHNZRS
DVGGERLISHERGMWBRXZSWNVOMQOQKLKCSQLRRHPVCAG
DOGGERFISHERGERMANBIGHTEAST----------------
```

This is very much like the supposed plaintext. Examine the first incorrect letter, which occurs in position two. This error cannot be due to a second rotor turnover, because of our assumption, hence it must be due to a missing plugboard element. If we let γ_1 denote the current approximation to the permutation representing the Enigma machine for letter one and τ the missing plugboard setting then we have

$$U^{\gamma_1} = V \text{ and } U^{\tau \cdot \gamma_1 \cdot \tau} = O.$$

This implies that τ should contain either a plug involving the letter U or one involving the letter O; but both of these letters are already used in the plugboard output from the Bombe. Hence, this configuration must be incorrect.

The second remaining stop is at Bombe configuration $i_1 = p_1 - r_1 = R$, $i_2 = p_2 - r_2 = D$ and $i_3 = p_3 - r_3 = L$. The plugboard setting is deduced to contain the following plugs

$$D \leftrightarrow Q,\ E \leftrightarrow T,\ F \leftrightarrow N,\ I \leftrightarrow O,\ S \leftrightarrow V,\ W \leftrightarrow X,$$

whilst the letters G, H and R are known to be plugged to themselves, assuming this is the correct configuration. These follow from the final register state in this configuration given in Table 8.2. So we need to find four other plug settings and the ring settings. Again we can assume that $r_3 = A$ as it plays no part in the actual decryption process, and again we deduce that p_1 must be one of $0, 1, 2, 16, 17, 18, 19, 20, 21, 22, 23, 24$ or 25.

With the Enigma setting of $p_1 = R$, $p_2 = D$, $p_3 = L$ and $r_1 = r_2 = r_3 = A$ and the above (incomplete) plugboard we decrypt the fragment of ciphertext and compare the resulting plaintext with the crib.

```
HUSVTNXRTSWESCGSGVXPLQKCEYUHYMPBNUITUIHNZRS
DOGGERFISHERGNRAMNCOXHXZMORIKOEDEYWEFEYMSDQ
DOGGERFISHERGERMANBIGHTEAST----------------
```

We now look at the first incorrect letter, occurring in the 14th position. Using the same notation as before, i.e. γ_j for the current approximation and τ for the missing plugs, we see that if this incorrect operation is due to a plug problem rather than a rotor turnover problem then we must have

$$C^{\tau \cdot \gamma_{13} \cdot \tau} = E.$$

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	0	0	0	1	1	0	1	1	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	0	0
B	0	0	0	1	1	1	1	1	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	1	0	0
C	0	0	0	1	1	1	1	1	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	0	0
D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
E	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
F	0	1	1	1	1	0	1	1	1	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1	0
G	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
H	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
I	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
J	0	0	0	1	1	0	1	1	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	1	0	0
K	0	0	0	1	1	0	1	1	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	0	0
L	0	0	0	1	1	0	1	1	1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0
M	0	0	0	1	1	0	1	1	1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0
N	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
O	0	0	0	1	1	1	1	1	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	0	0	1
P	0	0	0	1	1	0	1	1	1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0
Q	0	0	0	0	1	0	1	1	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	1	0	0
R	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
S	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
T	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
U	0	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	0	1	1	1	0	1	1	0	0	1
V	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
W	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
X	0	1	0	1	1	1	1	1	1	1	0	0	0	1	0	0	1	1	1	1	0	1	0	1	1	0
Y	0	0	0	1	1	1	1	1	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	1	0	0
Z	0	0	0	1	1	0	1	1	1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0

TABLE 8.2. The registers on the second Bombe stop

Now, E already occurs on the plugboard, via $E \leftrightarrow T$, so τ must include a plug which maps C to the letter x where

$$x^{\gamma_{13}} = E.$$

But we can compute that

$$\gamma_{13} = (AM)(BE)(CN)(DO)(FI)(GS)(HX)(JU)(KP)(LQ)(RV)(TY)(WZ),$$

from which we deduce that $x = B$. So we include the plug $C \leftrightarrow B$ in our new approximation and repeat to obtain the plaintext

```
HUSVTNXRTSWESCGSGVXPLQKCEYUHYMPBNUITUIHNZRS
DOGGERFISHERGERAMNBOXHXNMORIKOEMEYWEFEYMSDQ
DOGGERFISHERGERMANBIGHTEAST----------------
```

We then see in the 16th position that we either need to step the rotor or there should be a plug which means that S maps to M under the cipher. We have, for our new γ_{15} that

$$\gamma_{15} = (AS)(BJ)(CY)(DK)(EX)(FW)(GI)(HU)(LM)(NQ)(OP)(RV)(TZ).$$

The letter S already occurs in a plug, so we must have that A is plugged to M. We add this plug into our configuration and repeat

```
HUSVTNXRTSWESCGSGVXPLQKCEYUHYMPBNUITUIHNZRS
DOGGERFISHERGERMANBOXHXNAORIKVEAEYWEFEYASDQ
DOGGERFISHERGERMANBIGHTEAST----------------
```

Now the 20th character is incorrect: we need P to map to I and not O under the cipher in this position. Again assuming that this is due to a missing plug we find that

$$\gamma_{19} = (AH)(BM)(CF)(DY)(EV)(GX)(IK)(JR)(LS)(NT)(OP)(QW)(UZ).$$

There is already a plug involving the letter I so we deduce that the missing plug should be $K \leftrightarrow P$. Again we add this new plug into our configuration and repeat to obtain

```
HUSVTNXRTSWESCGSGVXPLQKCEYUHYMPBNUITUIHNZRS
DOGGERFISHERGERMANBIXHJNAORIPVXAEYWEFEYASDQ
DOGGERFISHERGERMANBIGHTEAST----------------
```

Now the 21st character is wrong as we must have that L maps to G. We know, from the Bombe stop configuration that G is plugged to itself, and given

$$\gamma_{20} = (AI)(BJ)(CW)(DE)(FK)(GZ)(HU)(LX)(MQ)(NT)(OV)(PY)(RS),$$

we deduce that if this error is due to a plug we must have that L is plugged to Z. We add this final plug into our configuration and find that we obtain

```
HUSVTNXRTSWESCGSGVXPLQKCEYUHYMPBNUITUIHNZRS
DOGGERFISHERGERMANBIGHJNAORIPVXAEYWEFEYAQDQ
DOGGERFISHERGERMANBIGHTEAST----------------
```

All the additional plugs we have added have been on the assumption that no rotor turnover has yet occurred. Any further errors must be due to rotor turnover, as we now have a full set of plugs (as we know our configuration only has ten plugs in use). If when correcting the rotor turnover we still do not decrypt correctly we need to back up and repeat the process.

We see that the next error occurs in position 23. This means that a rotor turnover must have occurred just before this letter was encrypted. In other words we have

$$22 - ((16 - p_1 - 1) \pmod{26}) + 26 = 26.$$

This implies that $p_1 = 19$, i.e. $p_1 = T$, which implies that $r_1 = C$. We now try to decrypt again, and we obtain

```
HUSVTNXRTSWESCGSGVXPLQKCEYUHYMPBNUITUIHNZRS
DOGGERFISHERGERMANBIGHTZWORIPVXAEYWEFEYAQDQ
DOGGERFISHERGERMANBIGHTEAST----------------
```

But we still do not have the correct plaintext. The only thing which could have happened is that we have had an incorrect third rotor movement. Rotor II has its notch in position $n_2 = 4$, i.e. E. If the third rotor moved on at position 24 then we have, in our earlier notation

$$\begin{aligned} m_1 &= n_1 - p_1 - 1 \pmod{26} = 16 - 19 - 1 \pmod{26} = 22, \\ m &= n_2 - p_2 - 1 \pmod{26} = 4 - p_2 - 1 \pmod{26}, \\ m_2 &= m_1 + 1 + 26 \cdot m = 23 + 26 \cdot m \\ 650 &= 23 - m_2 + 650 \end{aligned}$$

This last equation implies that $m_2 = 23$, which implies that $m = 0$, which itself implies that $p_2 = 3$, i.e. $p_2 = D$. But this is exactly the setting we have for the second rotor. So the problem is not that the third rotor advances, it is that it should not have advanced. We therefore need to change this to say $p_2 = E$ and $r_2 = B$, (although this is probably incorrect it will help us to decrypt the fragment). We find that we then obtain

```
HUSVTNXRTSWESCGSGVXPLQKCEYUHYMPBNUITUIHNZRS
DOGGERFISHERGERMANBIGHTEASTFORCEFIVEFALLING
DOGGERFISHERGERMANBIGHTEAST----------------
```

Hence, we can conclude that, apart from a possibly incorrect setting for the second ring we have the correct Enigma setting for this day.

8.9. Ciphertext Only Attack

The following attack allows one to break the Enigma machine when only a single ciphertext is given. The method relies on the fact that enough ciphertext is given and that a full set of plugs is not used. Suppose we have a reasonably large amount of ciphertext, say 500-odd characters, and that p plugs are in use. If we knew the rotor positions, around $((26 - 2 \cdot p)/26)^2$ of the letters

would decrypt exactly, as these letters would not pass through a plug either before or after the rotor stage. Hence, one could distinguish the correct rotor positions by using some statistic to distinguish a random plaintext from a plaintext in which $((26-2\cdot p)/26)^2$ of the letters are correct.

Gillogly[2] suggests using the index of coincidence. To obtain this statistic we count the frequency f_i of each letter in the resulting plaintext of length n and compute

$$IC = \sum_{i=A}^{Z} \frac{f_i \cdot (f_i - 1)}{n \cdot (n-1)}.$$

For this approach we set the rings to position A, A, A and then run through all possible rotor orders and rotor starting positions. For each setting we compute the resulting plaintext and the associated value of IC. We keep those settings which have a high value of IC.

Gillogly then suggests for the settings which give a high value of IC, to run through the associated ring settings – adjusting the starting positions as necessary – with a similar test. The problem with this approach is that it is susceptible to the effect of turnover of the various rotors. Either a rotor could turn over when we did not expect it, or it could have turned over by error. This is similar to the situation we obtained in our example using the Bombe in a known plaintext attack.

Consider the following ciphertext, of 734 characters in length

```
RSDZANDHWQJPPKOKYANQIGTAHIKPDFHSAWXDPSXXZMMAUEVYYRLWVFFTSDYQPS
CXBLIVFDQRQDEBRAKIUVVYRVHGXUDNJTRVHKMZXPRDUEKRVYDFHXLNEMKDZEWV
OFKAOXDFDHACTVUOFLCSXAZDORGXMBVXYSJJNCYOHAVQYUVLEYJHKKTYALQOAJ
QWHYVVGLFQPTCDCAZXIZUOECCFYNRHLSTGJILZJZWNNBRBZJEEXAEATKGXMYJU
GHMCJRQUODOYMJCXBRJGRWLYRPQNABSKSVNVFGFOVPJCVTJPNFVWCFUUPTAXSR
VQDATYTTHVAWTQJPXLGBSIDWQNVHXCHEAMVWXKIUSLPXYSJDUQANWCBMZFSXWH
JGNWKIOKLOMNYDARREPGEZKCTZNPQKOMJZSQHYEADZTLUPGBAVCVNJHXQKYILX
LTHZXJKYFQEBDBQOHMXBTVXSRGMPVOGMVTEYOCQEOZUSLZDQZBCXXUXBZMZSWX
OCIWRVGLOEZWVVOQJXSFYKDQDXJZYNPGLWEEVZDOAKQOUOTUEBTCUTPYDHYRUS
AOYAVEBJVWGZHGLHBDHHRIVIAUUBHLSHNNNAZWYCCOFXNWXDLJMEFZRACAGBTG
NDIHOWFUOUHPJAHYZUGVJEYOBGZIOUNLPLNNZHFZDJCYLBKGQEWTQMXJKNYXPC
KAPJGAGKWUCLGTFKYFASCYGTXGZXXACCNRHSXTPYLSJWIEMSABFH
```

We run through all possible $60 \cdot 26^3$ possible values for the rotors and the rotor positions, with ring settings equal to A, A, A. We obtain the "high" values for the IC statistic given in Table 8.3. For the top 300 or so such high values we then run through all possible values for the rings r_1 and r_2 (note the third ring plays no part in the process) and we set the rotor starting positions to be

$$\begin{aligned} p_1 &= p'_1 + r_1 + i_1, \\ p_2 &= p'_2 + r_2 + i_2, \\ p_3 &= p'_3 \end{aligned}$$

The addition of the r_j value is to take into account the change in ring position from A to r_j. The additional value of i_j is taken from the set $\{-1, 0, 1\}$ and is used to accommodate issues to do with rotor turnovers which our crude IC statistic is unable to pick up.

Running through all these possibilities we present the configurations producing the highest values of IC in Table 8.4. Finally, using our previous technique for finding the plugboard settings given the rotor settings in a ciphertext only attack (using the Sinkov statistic), we determine that the actual settings are

ρ_1	ρ_2	ρ_3	p_1	p_2	p_3	r_1	r_2	r_3
I	II	III	L	D	C	Q	B	A

[2]See the references at the end of this chapter.

IC	ρ_1	ρ_2	ρ_3	p'_1	p'_2	p'_3
0.04095	I	V	IV	P	R	G
0.0409017	IV	I	II	N	O	R
0.0409017	IV	V	I	M	G	Z
0.0408496	V	IV	II	I	J	B
0.040831	IV	I	V	X	D	A
0.0408087	II	I	V	E	O	J
0.040805	I	IV	III	T	Y	H
0.0407827	V	I	II	J	H	F
0.040779	III	IV	II	R	L	Q
0.0407121	II	III	V	V	C	C
0.0406824	IV	V	III	K	S	D
0.0406675	IV	II	III	H	H	D
0.04066	III	I	IV	P	L	G
0.0406526	IV	V	II	E	E	O
0.0406415	I	II	III	V	D	C
0.0406303	I	II	IV	T	C	G
0.0406266	V	IV	II	I	I	A
0.0406229	II	III	IV	K	Q	I
0.0405969	V	II	III	K	O	R
0.0405931	I	III	V	K	B	O
0.0405931	II	IV	I	K	B	Q
⋮	⋮	⋮	⋮	⋮	⋮	⋮

TABLE 8.3. High IC values for ring setting A, A, A

with the plugboard given by the following eight plugs:

$$A \leftrightarrow B,\ C \leftrightarrow D,\ E \leftrightarrow F,\ G \leftrightarrow H,$$

$$I \leftrightarrow J,\ K \leftrightarrow L,\ M \leftrightarrow N,\ O \leftrightarrow P.$$

With these settings one finds that the plaintext is again the first two paragraphs of "A Tale of Two Cities".

Chapter Summary

- We have described the Enigma machine and shown how poor session key agreement was used to break into the German traffic.
- We have also seen how stereotypical messages were successfully used to attack the system.
- We have seen how the plugboard and the rotors worked independently of each other, which led to attackers being able to break each component separately.

IC	ρ_1	ρ_2	ρ_3	p_1	p_2	p_3	r_1	r_2	r_3
0.0447751	I	II	III	K	D	C	P	B	A
0.0444963	I	II	III	L	D	C	Q	B	A
0.0444406	I	II	III	J	D	C	O	B	A
0.0443848	I	II	III	K	E	D	P	B	A
0.0443588	I	II	III	K	I	D	P	F	A
0.0443551	I	II	III	K	H	D	P	E	A
0.0443476	I	II	III	K	F	D	P	C	A
0.0442807	I	II	III	L	E	D	Q	B	A
0.0442324	I	II	III	J	H	D	O	E	A
0.0442064	I	II	III	K	G	D	P	D	A
0.0441357	I	II	III	J	G	D	O	D	A
0.0441097	I	II	III	J	E	D	O	B	A
0.0441097	I	II	III	L	F	D	Q	C	A
0.0441023	I	II	III	L	C	C	Q	A	A
0.0440837	I	II	III	J	F	D	O	C	A
0.0440763	I	II	III	J	I	D	O	F	A
0.0440242	I	II	III	K	C	C	P	A	A
0.0439833	I	II	III	L	G	D	Q	D	A
0.0438904	I	II	III	L	I	D	Q	F	A
0.0438607	I	II	III	L	H	D	Q	E	A
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

TABLE 8.4. High IC values for different ring and rotor settings

Further Reading

The paper by Rejewski presents the work of the Polish cryptographers very clearly. The pure ciphertext only attack is presented in the papers by Gillogly and Williams.

J. Gillogly. *Ciphertext-only cryptanalysis of Enigma.* Cryptologia, **19**, 405–413, 1995.

M. Rejewski. *An application of the theory of permutations in breaking the Enigma cipher.* Applicationes Mathematicae, **16**, 543–559, 1980.

H. Williams. *Applying statistical language recognition techniques in the ciphertext-only cryptanalysis of Enigma.* Cryptologia, **24**, 4–17, 2000.