The Polish Cryptanalysis of Enigma

Richard E. Klima
klimare@appstate.edu
Department of Mathematical Sciences
Appalachian State University
Boone, North Carolina 28608
USA

Neil P. Sigmon
npsigmon@radford.edu
Department of Mathematics and Statistics
Radford University
Radford, Virginia 24142
USA

Abstract

The cryptanalysis of the Enigma cipher machine during World War II by British and American codebreakers led by Alan Turing at Bletchley Park has been well-documented, and rightfully recognized as one of the supreme achievements of the human intellect. However, without the successful cryptanalysis of an earlier version of Enigma by Polish codebreakers led by Marian Rejewski in the 1930s, the work of the British and Americans in the 1940s might have taken much longer, prolonging the war at the potential cost of untold additional lives. The mathematics integral to the Polish cryptanalysis of Enigma involved some basic theory of permutations. The purpose of this paper is to present an overview of these ideas and how they served to this effect. To assist in demonstrating this, technology involving Maplets will be used.

1 Introduction

In 1918, German electrical engineer Arthur Scherbius applied for a patent for a mechanical cipher machine. Later marketed commercially under the name *Enigma*, this machine was designed with electric current running through revolving wired wheels, called *rotors*. Scherbius offered Enigma to

the German military, who, after learning that their World War I ciphers had routinely been cracked, adopted and used Enigma as their primary field cipher prior to and throughout World War II.

In the early 1930s, due to suspicions that Germany was seeking to rearm and reclaim territories lost to Poland following World War I, the Poles began carefully monitoring German radio transmissions, which were encrypted using Enigma. Unable to decrypt these messages, the Polish government recruited mathematics students for the purpose of cryptanalyzing Enigma. Among these students were Marian Rejewski, Jerzy Różycki, and Henryk Zygalski, who became employees of the Cipher Bureau in Warsaw in the summer of 1932, coinciding with the beginning or their work on Enigma. Among Rejewski, Różycki, and Zygalski, the most renowned is Rejewski, who in particular pioneered the use of permutations in attacking Enigma.

In this paper, we will give an overview of the theory of permutations that Rejewski needed, and characterize the various components of Enigma and reasons for Germany's ill-fated confidence in its security. We will also describe and demonstrate several aspects of the successful efforts by the Polish codebreakers in cryptanalyzing the pre-war version of Enigma.

2 Permutations

Traditional collegiate abstract algebra courses often cover the basic theory of permutations, the details of which can thus be found in many resources typically used as textbooks in such courses. Due to their importance in the Polish method for cryptanalyzing Enigma, we will begin by giving an overview of some of the theory of permutations and their representations involving cycles.

A *permutation* on a set Ω is a function $\sigma : \Omega \to \Omega$ that is both one-to-one and onto. In this paper we will assume Ω is a finite set. As an example, if $\Omega = \{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D}\}$, then one permutation on Ω is the function σ with $\sigma(\mathtt{A}) = \mathtt{B}$, $\sigma(\mathtt{B}) = \mathtt{C}$, $\sigma(\mathtt{C}) = \mathtt{D}$, and $\sigma(\mathtt{D}) = \mathtt{A}$. Similarly, the function τ with $\tau(\mathtt{A}) = \mathtt{C}$, $\tau(\mathtt{B}) = \mathtt{B}$, $\tau(\mathtt{C}) = \mathtt{D}$, and $\tau(\mathtt{D}) = \mathtt{A}$ is a permutation on Ω. On the other hand, the function μ with $\mu(\mathtt{A}) = \mathtt{B}$, $\mu(\mathtt{B}) = \mathtt{A}$, $\mu(\mathtt{C}) = \mathtt{B}$, and $\mu(\mathtt{D}) = \mathtt{D}$ is not a permutation on Ω, since it is not one-to-one.

Permutations can be represented most efficiently using *cycle* notation. A permutation on a finite set $\{x_1, x_2, \ldots, x_n\}$ that maps $x_1 \mapsto x_2 \mapsto x_3 \mapsto \cdots \mapsto x_n \mapsto x_1$ is represented using cycle notation as $(x_1 x_2 \cdots x_n)$, with each element within the parentheses written to the right of the element from which it maps, until the parentheses close when the element at the "end" (the far right) maps back to the one at the "start" (the far left). So, for example, the permutation σ on $\Omega = \{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D}\}$ with $\sigma(\mathtt{A}) = \mathtt{B}$, $\sigma(\mathtt{B}) = \mathtt{C}$, $\sigma(\mathtt{C}) = \mathtt{D}$, and $\sigma(\mathtt{D}) = \mathtt{A}$ would be represented using cycle notation as $\sigma = (\mathtt{ABCD})$. The permutation τ on Ω with $\tau(\mathtt{A}) = \mathtt{C}$, $\tau(\mathtt{B}) = \mathtt{B}$, $\tau(\mathtt{C}) = \mathtt{D}$, and $\tau(\mathtt{D}) = \mathtt{A}$ would be represented as $\tau = (\mathtt{ACD})(\mathtt{B})$, which requires two "cycles," one containing the single element $\mathtt{B}$, since $\mathtt{B}$ maps to itself under τ, and thus must be at both the start and end of the cycle in which it is contained. Such cycles of "length" 1 are often not explicitly written in representations of permutations, since an element's absence from an expression can be interpreted as indicating that it maps to itself. Thus, τ could also be expressed as $\tau = (\mathtt{ACD})$. In this paper though we will include cycles of length 1.

Any pair of permutations σ and τ on a set Ω can be combined using the operation of function composition to form a new permutation, $\sigma \circ \tau$, on Ω, which is typically represented simply as $\sigma\tau$. The Polish codebreakers interpreted such compositions of permutations with the permutation listed on the left applied first, so for any $a \in \Omega$, the composition $\sigma\tau$ would be applied as $(\sigma\tau)(a) = \tau(\sigma(a))$.[1] As an example, for the pair of permutations $\sigma = (\mathtt{ABCD})$ and $\tau = (\mathtt{ACD})(\mathtt{B})$ on $\Omega = \{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D}\}$, we have the following:

$$\begin{aligned}
(\sigma\tau)(\mathtt{A}) &= \tau(\sigma(\mathtt{A})) = \tau(\mathtt{B}) = \mathtt{B},\\
(\sigma\tau)(\mathtt{B}) &= \tau(\sigma(\mathtt{B})) = \tau(\mathtt{C}) = \mathtt{D},\\
(\sigma\tau)(\mathtt{C}) &= \tau(\sigma(\mathtt{C})) = \tau(\mathtt{D}) = \mathtt{A},\\
(\sigma\tau)(\mathtt{D}) &= \tau(\sigma(\mathtt{D})) = \tau(\mathtt{A}) = \mathtt{C}.
\end{aligned}$$

That is, $\sigma\tau = (\mathtt{ABCD})(\mathtt{ACD})(\mathtt{B}) = (\mathtt{ABDC})$.

For a nonempty set Ω, the set of all permutations on Ω is a group with the operation of function composition. This guarantees that every permutation on Ω will be invertible, meaning that for every permutation σ on Ω, there must be a corresponding permutation σ^{-1} on Ω for which $\sigma\sigma^{-1}$ and $\sigma^{-1}\sigma$ are both the identity permutation (which maps every possible input to itself). To find the inverse of a cycle, we must only reverse its elements. For example, for the permutation $\sigma = (\mathtt{ABCD})$ on the set $\Omega = \{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D}\}$, the inverse is the permutation $\sigma^{-1} = (\mathtt{DCBA})$, or, equivalently, using a common convention of moving the first element in some understood order (e.g., alphabetical) to the start of a cycle, $\sigma^{-1} = (\mathtt{ADCB})$.

Similarly, for a permutation that requires more than one cycle to express in cycle notation, if the cycles are "disjoint" (i.e., if none of the elements appears in more than one of the cycles), then to find the inverse we must only reverse the elements in each of the cycles. For example, for the permutation $\sigma = (\mathtt{ADE})(\mathtt{BCG})(\mathtt{F})$ on $\Omega = \{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D},\mathtt{E},\mathtt{F},\mathtt{G}\}$, the inverse is $\sigma^{-1} = (\mathtt{EDA})(\mathtt{GCB})(\mathtt{F})$, or, equivalently, $\sigma^{-1} = (\mathtt{AED})(\mathtt{BGC})(\mathtt{F})$.

A pair of permutations σ and τ on a set Ω are said to be *conjugate* if there is a permutation ρ on Ω for which $\rho\sigma\rho^{-1} = \tau$. For example, the permutations $\sigma = (\mathtt{ADE})(\mathtt{BCG})(\mathtt{F})$ and $\tau = (\mathtt{BCD})(\mathtt{GAF})(\mathtt{E})$ on $\Omega = \{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D},\mathtt{E},\mathtt{F},\mathtt{G}\}$ are conjugate, since for the permutation $\rho = (\mathtt{ACDEFGB})$, it can be verified that $\rho\sigma\rho^{-1} = \tau$.

The *type* of a permutation is a list of the lengths of the cycles in nonincreasing numerical order in its representation as a single cycle or composition of disjoint cycles. For example, the permutations $\sigma = (\mathtt{ADE})(\mathtt{BCG})(\mathtt{F})$ and $\tau = (\mathtt{BCD})(\mathtt{GAF})(\mathtt{E})$ on $\Omega = \{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D},\mathtt{E},\mathtt{F},\mathtt{G}\}$ are both of type $(3,3,1)$, since as compositions of disjoint cycles both require two cycles of length 3 and one cycle of length 1. Similarly, the permutation $\rho = (\mathtt{ACDEFGB})$ on Ω is of type (7). The fact that the permutations σ and τ, which are conjugate, are also of the same type is not a coincidence, but rather is guaranteed by the following theorem, which turns out to be fundamental to the Polish method for cryptanalyzing Enigma.

[1] Many resources interpret compositions of permutations with the permutation listed on the right applied first, so $\sigma\tau$ would be applied as $(\sigma\tau)(a) = \sigma(\tau(a))$. We will use the interpretation of the Polish codebreakers though, with the permutation listed on the left applied first.

Theorem 2.1 *Conjugate permutations must be of the same type.*

Proof. Suppose σ and τ are conjugate permutations, with $\rho\sigma\rho^{-1} = \tau$, and let $(a_1 a_2 \cdots a_k)$ be a cycle for σ, so $\sigma(a_j) = a_{j+1}$ for $j = 1, 2, \ldots, k-1$, and $\sigma(a_k) = a_1$. For any $j = 1, 2, \ldots, k-1$, note that

$$\tau(\rho^{-1}(a_j)) = (\rho\sigma\rho^{-1})(\rho^{-1}(a_j)) = \rho^{-1}(\sigma(\rho(\rho^{-1}(a_j)))) = \rho^{-1}(\sigma(a_j)) = \rho^{-1}(a_{j+1}).$$

Similarly, $\tau(\rho^{-1}(a_k)) = \rho^{-1}(a_1)$, and so we see that $(\rho^{-1}(a_1)\rho^{-1}(a_2)\cdots\rho^{-1}(a_k))$ is a cycle for τ. Thus, σ and τ must be of the same type. ■

The proof of Theorem 2.1 shows that for permutations σ and ρ, to find the conjugate $\rho\sigma\rho^{-1}$, we must only translate the elements in the cycles of σ by ρ^{-1}. As an example, for the permutations $\sigma = (\mathtt{ADE})(\mathtt{BCG})(\mathtt{F})$ and $\rho = (\mathtt{ACDEFGB})$ on $\Omega = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{G}\}$,

$$\rho\sigma\rho^{-1} = (\rho^{-1}(\mathtt{A})\rho^{-1}(\mathtt{D})\rho^{-1}(\mathtt{E}))(\rho^{-1}(\mathtt{B})\rho^{-1}(\mathtt{C})\rho^{-1}(\mathtt{G}))(\rho^{-1}(\mathtt{F})) = (\mathtt{BCD})(\mathtt{GAF})(\mathtt{E}).$$

Finally, for a pair of permutations σ and τ on a set Ω, the inverse of $\sigma\tau$ will be $(\sigma\tau)^{-1} = \tau^{-1}\sigma^{-1}$, since both $(\sigma\tau)(\tau^{-1}\sigma^{-1}) = \sigma(\tau\tau^{-1})\sigma^{-1} = \sigma\sigma^{-1}$ and $(\tau^{-1}\sigma^{-1})(\sigma\tau) = \tau^{-1}(\sigma^{-1}\sigma)\tau = \tau^{-1}\tau$ will be the identity permutation. Since $(\sigma\tau)^{-1} = \tau^{-1}\sigma^{-1}$, we see that inverses of compositions of permutations satisfy the "reverse-order" law in mathematics. Notably, inverting a conjugate $\rho\sigma\rho^{-1}$ yields $(\rho\sigma\rho^{-1})^{-1} = (\rho^{-1})^{-1}\sigma^{-1}\rho^{-1} = \rho\sigma^{-1}\rho^{-1}$. That is, to invert a conjugate, we must only invert the permutation in the middle of the expression.

3 Description of Enigma

Enigmas contained various components, each contributing in their own way to the overall security of the machine. Figure 1 shows two photos of an Enigma with several of these components labeled.

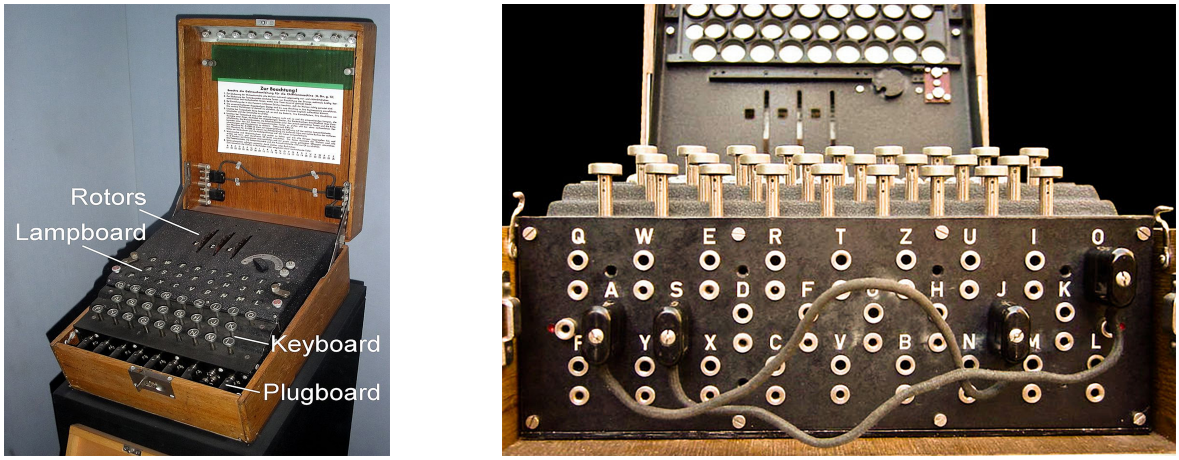

Figure 1: Photos of an Enigma with components labeled and the plugboard shown.

To encrypt or decrypt a letter using an Enigma, a key labeled with the letter was pressed on a keyboard, which launched electric current designating the letter from the keyboard. The current traveled first to the *plugboard*, where it changed to designate a different letter if a pair of sockets representing the input letter were connected by a cable to a pair of sockets representing a different letter. The current then passed right-to-left through a series of wheels, called *rotors*, three in the version of Enigma faced by the Poles, each of which could change the current to designate a different letter. The current then passed through another wheel called the *reflector*, which definitely changed the current to designate a different letter, back through the rotors but from left to right, back through the plugboard, and finally to the *lampboard*, where it lit a small bulb indicating the encrypted or decrypted letter.

The **plugboard** resembled a miniature old telephone switchboard, and was physically situated in the front of the machine. It had 26 pairs of sockets, one pair to represent each letter in the alphabet. Each socket pair could either be left open or connected to another socket pair by a short cable. If the socket pairs representing two letters were connected by a cable, then current designating either letter would change at the plugboard to designate the other letter. If the socket pair representing a letter was left open in the plugboard, then current designating that letter would leave the plugboard still designating the same letter.

The machine could be operated using anywhere from 0 to 13 plugboard cables. Varying the number of cables would have obviously maximized security, however standard German operating procedure was to use a fixed number of cables. With a fixed number of cables, we show in [3] that 11 would have maximized security, but for the version of Enigma faced by the Poles, 6 were always used. We actually show in [3] that for any $p = 1, 2, \ldots, 13$, the number of ways to connect $2p$ socket pairs using p cables is given by

$$N_p = \frac{26!}{(2p)!(26-2p)!} \cdot (2p-1) \cdot (2p-3) \cdots 1.$$

Using this formula, we find that the number of different ways the plugboard could be configured in the version of Enigma faced by the Poles is $N_6 = 100{,}391{,}791{,}500$.

Three **rotors**, each a circular disk about the size of a hockey puck, were lined up side-by-side in the back of the machine. The German Navy later employed a version of Enigma in which four rotors were used at the same time, but in the version of Enigma faced by the Poles, three were used.

We will call the flat sides of a rotor the *right* and *left* sides, since they could only be placed in the machine standing on end with each side facing in a particular direction. Both flat sides of a rotor contained 26 contact points, one to represent each letter in the alphabet, in alphabetical order around both sides of the rotor clockwise (when it was viewed from the right). Each contact on the right side of a rotor was connected to a contact on the left side by an electrical wire, although contacts that were connected did not usually represent the same letter. The idea was that current could enter a rotor on one side at a contact position designating one letter, and pass through and exit the rotor on the other side at a contact position designating a different letter.

There are an incredibly large number of ways in which the contacts within a rotor could have been connected ($26! \approx 4.03 \times 10^{26}$, to be precise), but because they had to be hard-wired, rotors with only a very small number of different wirings were ever actually produced. Rotors were eventually produced

with ten different wirings, but for the version of Enigma faced by the Poles, rotors with only three different wirings were available. While multiple rotors with identical wirings could have been used in the machine at the same time, in the version of Enigma faced by the Poles, exactly one rotor with each of the three different wirings available at the time was always used. These rotors could, however, be placed in the machine from left to right in any of $3! = 6$ possible orders.

One of the most remarkable achievements of the Polish codebreakers was that they were able to identify these three different rotor wirings from intercepted messages alone, without first physically handling a rotor. A detailed discussion of how they accomplished this can be found in [2]. These three different wirings are most easily expressed as permutations, taking as input the letter designated by current as it enters a rotor on the right, and giving as output the letter designated by the current as it exits the rotor on the left. In this manner, the wirings of the rotors, which we will denote as R_1, R_2, and R_3, can be expressed using cycle notation as follows.

$$R_1 = (\texttt{AELTPHQXRU})(\texttt{BKNW})(\texttt{CMOY})(\texttt{DFG})(\texttt{IV})(\texttt{JZ})(\texttt{S}) \quad (1)$$

$$R_2 = (\texttt{A})(\texttt{BJ})(\texttt{CDKLHUP})(\texttt{ESZ})(\texttt{FIXVYOMW})(\texttt{GR})(\texttt{NT})(\texttt{Q}) \quad (2)$$

$$R_3 = (\texttt{ABDHPEJT})(\texttt{CFLVMZOYQIRWUKXSG})(\texttt{N}) \quad (3)$$

For example, current designating A and entering R_1 on the right would exit R_1 on the left designating E, current designating J and entering R_2 on the right would exit R_2 on the left designating B, and current designating N and entering R_3 on the right would exit R_3 on the left still designating N. Note also that the permutation R_1 is of type $(10,4,4,3,2,2,1)$, the permutation R_2 is of type $(8,7,3,2,2,2,1,1)$, and R_3 is of type $(17,8,1)$.

Rotors were called "rotors" because they rotated inside the machine during its operation. Even before this though, as the machine was being set up, each rotor could be rotated into any of 26 distinct positions before being placed into the machine. This gave $26^3 = 17{,}576$ unique positions into which all three rotors could be rotated before being placed into the machine.

To assist Enigma operators with orienting rotors correctly, the letters A–Z (or sometimes the numbers 01–26) were etched into a ring around the outside of each rotor, listed in order clockwise (when the rotor was viewed from the right). For each of the three rotor slots, a small window was cut into the top of the machine to reveal the letter (or number) at a particular location on the ring. We will call this letter the *window letter*. The etched ring around the outside of each rotor was movable, and could itself be rotated into any of 26 distinct positions while the wired part of the rotor was held stationary. This was a complication, because rotating the ring while holding the wired part of a rotor stationary would move the window letter but not the wiring. A number from 1 to 26 called the *ring setting* indicated the position of the ring on a rotor.

The 26 possible ring settings for each rotor gave $26^3 = 17{,}576$ unique ring settings for all three rotors before they were placed into the machine. When counting the number of ways an Enigma could be set up prior to its operation, it is tempting to ignore the number of possible ring settings, since the movable rings did not change the fact that there were only 26 different wires for current to follow through a rotor. However, the number of possible ring settings cannot be ignored, since the rotation of the rotors during the operation of the machine depended only on the ring, not the full rotor. More

specifically, encrypting and decrypting messages using an Enigma was done one letter at a time, and each time the key for an input letter was pressed on the keyboard, the rightmost rotor would immediately (before current reached the rotors) rotate one position counterclockwise (when the rotor was viewed from the right). In addition, a notch was cut out from the ring around each rotor, and for the middle and rightmost rotors, when this notch was in one particular position in the rotor slot, if the rotor rotated one position counterclockwise it would cause the rotor to its left to also rotate one position counterclockwise.[2]

Combining the $3! = 6$ possible rotor orders, $26^3 = 17{,}576$ unique positions into which all three rotors could be rotated, and $26^3 = 17{,}576$ unique ring settings for all three rotors, we find that the total number of different ways that rotors could be configured in the version of Enigma faced by the Poles is $6 \cdot 17{,}576 \cdot 17{,}576 = 1{,}853{,}494{,}656$.

The **reflector** sat to the left of the rotors, and was also a circular disk with 26 contact points, one to represent each letter in the alphabet. The reflector only had contacts on its right side though, which were always wired to each other in 13 pairs. After its right-to-left journey through the rotors, current designating a letter would enter the reflector at one of its contact points, travel along one of the wires, and then exit the reflector at a different contact point, designating a different letter, to begin a left-to-right journey back through the rotors.

As with a rotor, there are a very large number of ways in which the contacts within a reflector could have been connected ($25 \cdot 23 \cdot 21 \cdots 1 = 7{,}905{,}853{,}580{,}625$, to be precise), but because they also had to be hard-wired, reflectors with only a very small number of different wirings were ever produced. Reflectors were eventually produced with five different wirings, but for the version of Enigma faced by the Poles, reflectors with only a single distinct wiring were available. This reflector was called reflector *A*, and it permuted the letters designated by current according to the following permutation.

$$A = (\texttt{AE})(\texttt{BJ})(\texttt{CM})(\texttt{DZ})(\texttt{FL})(\texttt{GY})(\texttt{HX})(\texttt{IV})(\texttt{KW})(\texttt{NR})(\texttt{OQ})(\texttt{PU})(\texttt{ST}) \tag{4}$$

For example, current entering this reflector designating `D` would exit the reflector designating `Z`, and vice versa. Also, although an Enigma could be operated with the reflector rotated into any of 26 distinct positions before being placed into the machine, reflectors were always placed into Enigma machines in one specific position, and once in the machine did not move. Thus, the number of different ways that the reflector could be configured in the version of Enigma faced by the Poles is 1.

With 100,391,791,500 ways to configure the plugboard, 1,853,494,656 ways to configure the rotors, and just 1 way to configure the reflector, the number of different ways that the full machine could be configured in the version of Enigma faced by the Poles is the following number.

$$100{,}391{,}791{,}500 \cdot 1{,}853{,}494{,}656 \cdot 1 = 186{,}075{,}649{,}051{,}516{,}224{,}000 \tag{5}$$

This number, which is approximately 1.86×10^{20}, or more than 186 *quintillion*, was much too large for a brute force attack on Enigma to have been possible at the time. However, through the work of Rejewski, Różycki, and Zygalski, this astronomical number was overcome.

[2]This means we could technically reduce the number of possible ring settings under consideration by a factor of 26, since the notch on the leftmost rotor was irrelevant, as there was no rotor to its left for it to influence. We will describe later though how the Poles were actually able to nullify the effect of all the ring settings altogether.

4 Operation of Enigma

Although all three of the rotors in an Enigma of the version faced by the Poles could rotate during the encryption of a message, for the rest of this paper we will assume that only the rightmost rotor rotated. This obviously significantly simplifies the machine, but it is a reasonable assumption, since the Polish method for cryptanalyzing Enigma would fail if any rotor except the rightmost rotated during the formation of the portion of a ciphertext used in determining the configuration of the machine.

Consider now an Enigma of the version faced by the Poles, with wired plugboard pairs A $\leftrightarrow$ Q, I $\leftrightarrow$ K, J $\leftrightarrow$ T, M $\leftrightarrow$ Z, O $\leftrightarrow$ S, and X $\leftrightarrow$ Y, and rotors in the left-to-right order R_3, R_1, R_2, with corresponding ring settings 14, 23, 5. Suppose also that the rotors are rotated into the positions for which the initial left-to-right window letters are AIM (called the *ground setting* of the machine). To use this machine to encrypt E, we would press the key labeled E on the keyboard. This would cause the rightmost rotor R_2 to rotate one position counterclockwise (before current reached the rotors), resulting in the new left-to-right window letters AIN, and send current initially designating E on a journey through the machine which is illustrated in Figure 2.

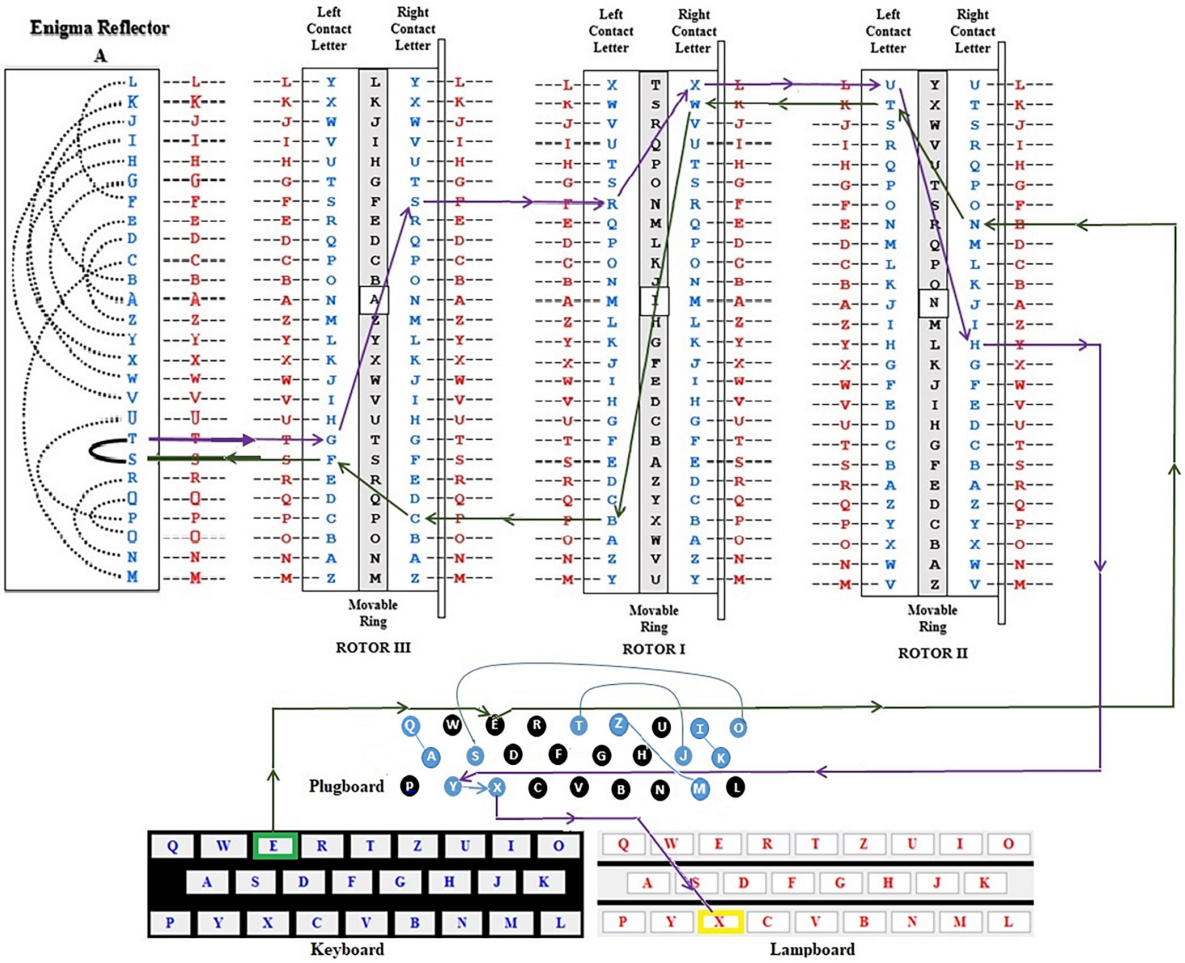

Figure 2: Diagram of current flow through an Enigma.

The path followed by the current through the machine illustrated in Figure 2 can be described explicitly using the language and terminology of permutations as follows.

- Current designating E travels first to the plugboard, which can be represented as the permutation $P = (\mathtt{AQ})(\mathtt{B})(\mathtt{C})(\mathtt{D})(\mathtt{E})(\mathtt{F})(\mathtt{G})(\mathtt{H})(\mathtt{IK})(\mathtt{JT})(\mathtt{L})(\mathtt{MZ})(\mathtt{N})(\mathtt{OS})(\mathtt{P})(\mathtt{R})(\mathtt{U})(\mathtt{V})(\mathtt{W})(\mathtt{XY})$. Since this permutation maps E to itself, the current leaves the plugboard still designating E.
- Current designating E travels next to the rightmost rotor R_2, which has ring setting 5 and window letter N. The effect of the ring setting and window letter can be combined in general into a single number from 0 to 25 called the *rotor offset*, which indicates the number of positions the wiring in the rotor is rotated from its "standard" position (for which this rotor would transform the letter designated by current exactly as in the expression for R_2 given in (2)) to reach its actual position. In particular, with window letter N, which is the 14th letter in the alphabet, and ring setting 5, the rotor offset is $14-5=9$, meaning that the wiring in the rotor is rotated 9 positions from its standard position to reach its actual position, and that current designating any letter would actually follow the wire through the rotor intended for the letter 9 positions later in the alphabet. This effect of the rotor offset can be represented as the permutation $O_9 = (\mathtt{AJSBKTCLUDMVENWFOXGPYHQZIR})$, in which each letter is followed by the letter 9 positions later in the alphabet, and the action of current entering and passing through the rotor can be represented by the permutation O_9R_2. However, because the current is actually designating the letter 9 positions earlier in the alphabet, the action of the current leaving the rotor also requires an application of the inverse O_9^{-1}. That is, the journey into, through, and out of the rightmost rotor can be represented in full by the permutation $R = O_9R_2O_9^{-1} = (\mathtt{AS})(\mathtt{BCYLGTU})(\mathtt{DNWZOMPF})(\mathtt{EK})(\mathtt{H})(\mathtt{IX})(\mathtt{JQV})(\mathtt{R})$. Since this permutation maps E to K, the current leaves the rightmost rotor designating K.
- Current designating K travels next to the middle rotor R_1, with ring setting 23 and window letter I, which combine to give the rotor offset $9-23=-14$, or 12 when adjusted by adding 26 to make the rotor offset be nonnegative. With $O_{12} = (\mathtt{AMYKWIUGSEQCO})(\mathtt{BNZLXJVHTFRDP})$, and the expression for R_1 given in (1), the journey into, through, and out of the middle rotor can be represented by $M = O_{12}R_1O_{12}^{-1} = (\mathtt{ACMQ})(\mathtt{BKPY})(\mathtt{DVELFIOSZH})(\mathtt{G})(\mathtt{JW})(\mathtt{NX})(\mathtt{RTU})$. Since this permutation maps K to P, the current leaves the middle rotor designating P.
- Current designating P travels next to the leftmost rotor R_3, with ring setting 14 and window letter A, which combine to give rotor offset $1-14=-13$, or 13 when adjusted by adding 26. With $O_{13} = (\mathtt{AN})(\mathtt{BO})(\mathtt{CP})(\mathtt{DQ})(\mathtt{ER})(\mathtt{FS})(\mathtt{GT})(\mathtt{HU})(\mathtt{IV})(\mathtt{JW})(\mathtt{KX})(\mathtt{LY})(\mathtt{MZ})$, and the expression for R_3 given in (3), the journey into, through, and out of the leftmost rotor can be represented by $L = O_{13}R_3O_{13}^{-1} = (\mathtt{A})(\mathtt{BLDVEJHXKFTPSYIZM})(\mathtt{CRWGNOQU})$. Since this permutation maps P to S, the current leaves the leftmost rotor designating S.
- Current designating S travels next to reflector A, whose expression as a permutation is given in (4). Since this permutation maps S to T, the current leaves the reflector designating T.
- Current designating T travels next back to the leftmost rotor, but since it is now moving in the opposite direction, it will now follow the path through the rotor indicated by the permutation $L^{-1} = (O_{13}R_3O_{13}^{-1})^{-1} = O_{13}R_3^{-1}O_{13}^{-1}$. That is, the journey into, through, and out

of the leftmost rotor in this opposite direction can be represented in full by the permutation $L^{-1} = O_{13}R_3^{-1}O_{13}^{-1} = (\texttt{A})(\texttt{BMZIYSPTFKXHJEVDL})(\texttt{CUQONGWR})$. Since this permutation maps T to F, the current leaves the leftmost rotor in this direction designating F.

- Current designating F travels next back to the middle rotor. The journey into, through, and out of the middle rotor in this opposite direction can be represented in full by the permutation $M^{-1} = O_{12}R_1^{-1}O_{12}^{-1} = (\texttt{AQMC})(\texttt{BYPK})(\texttt{DHZSOIFLEV})(\texttt{G})(\texttt{JW})(\texttt{NX})(\texttt{RUT})$. Since this permutation maps F to L, the current leaves the middle rotor in this direction designating L.
- Current designating L travels next back to the rightmost rotor. The journey into, through, and out of the rightmost rotor in this opposite direction can be represented in full by the permutation $R^{-1} = O_9R_2^{-1}O_9^{-1} = (\texttt{AS})(\texttt{BUTGLYC})(\texttt{DFPMOZWN})(\texttt{EK})(\texttt{H})(\texttt{IX})(\texttt{JVQ})(\texttt{R})$. Since this permutation maps L to Y, the current leaves the rightmost rotor in this direction designating Y.
- Current designating Y travels next back to the plugboard, which maps Y to X. The current thus leaves the plugboard designating X, headed on its way to the lampboard, whose bulb labeled X it lights.

The full journey through the machine can be expressed as $PRMLAL^{-1}M^{-1}R^{-1}P^{-1}$, or, equivalently, $(PRML)A(PRML)^{-1}$. This shows that as permutations, the full journey through the machine is conjugate to the reflector. By Theorem 2.1, we know then that the full journey through the machine is a permutation of the same type as the reflector.[3] This guarantees that the encryption and decryption processes were reciprocal, and also that it was not possible for a letter to encrypt as itself. These facts were also true of the more complex versions of Enigma the Germans used during World War II, and were exploited by the British and American codebreakers led by Alan Turing at Bletchley Park in their own successful cryptanalysis of Enigma.

We should also note that after encrypting E as described in this section, if we wished to encrypt a second letter, pressing its key on the keyboard would cause the rightmost rotor to rotate another position counterclockwise. This would result in the new rightmost window letter O, and new rightmost rotor offset 10. Then, with $O_{10} = (\texttt{AKUEOYISCMWGQ})(\texttt{BLVFPZJTDNXHR})$,[4] for the encryption of the second letter, the journey into, through, and out of the rightmost rotor the first time it was visited would be represented by $O_{10}R_2O_{10}^{-1} = (\texttt{ABXKFST})(\texttt{CMVYNLOE})(\texttt{DJ})(\texttt{G})(\texttt{HW})(\texttt{IPU})(\texttt{Q})(\texttt{RZ})$.

We will now demonstrate a Maplet[5] written by the authors that simulates the operations of encryption and decryption with an Enigma machine. This Maplet is available for download at the link labeled [S1] in Section 7. Enigma operators carried a codebook that indicated the common plugboard pairs, rotor order, ring settings, and ground setting for each day. However, given the volume of messages that were sent, and the fact that each one on any given day would be encrypted using the same codebook settings, to decrease vulnerability to frequency analysis, Enigma operators were instructed to choose

[3] Incidentally, for the example that is provided in this section, the full journey through the machine turns out to be $PRMLAL^{-1}M^{-1}R^{-1}P^{-1} = (\texttt{AG})(\texttt{BO})(\texttt{CU})(\texttt{DK})(\texttt{EX})(\texttt{FV})(\texttt{HM})(\texttt{IY})(\texttt{JP})(\texttt{LQ})(\texttt{NW})(\texttt{RS})(\texttt{TZ})$.

[4] This could be found directly, or, with $O_1 = (\texttt{ABCDEFGHIJKLMNOPQRSTUVWXYZ})$, as $O_{10} = O_1O_9$.

[5] A Maplet is like an applet, but uses (and requires) the engine of the computer algebra system Maple, and is written using Maple functions and syntax.

a separate, personal ground setting for each message they sent. To avoid confusion (with the ground setting given in the codebook), we will call this the *message setting*. When encrypting a message, an operator was to first configure their machine as directed by the codebook, including its ground setting, and use this configuration to encrypt the letters in their message setting, repeated to form a six-letter sequence so that transmission errors could be identified. The operator was then to manually turn the rotors so that their message setting was showing in the windows, and use this configuration to encrypt their actual message. They were then to transmit the six letters that resulted from encrypting their repeated message setting followed by the letters in their encrypted actual message.

An Enigma operator in receipt of an encrypted transmission was to first configure their machine as directed by the codebook, and use this configuration to decrypt the first six letters in the received transmission. This would reveal the message setting that had been used to encrypt the actual message. The operator was then to manually turn the rotors so that this message setting was showing in the windows, and use this configuration to decrypt the remainder of the transmission.

As an example, suppose an Enigma operator wishing to encrypt `ENIGMAXISXWORKING` chose message setting AIM, and found, for daily settings in the codebook, plugboard pairs `A` ↔ `Q`, `I` ↔ `K`, `J` ↔ `T`, `M` ↔ `Z`, `O` ↔ `S`, and `X` ↔ `Y`, rotor order R_3, R_1, R_2, ring settings 14, 23, 5, and ground setting TAG. The operator would first configure their machine as directed by the codebook, and encrypt `AIMAIM`. The result of using the Maplet to do this is shown in Figure 3.

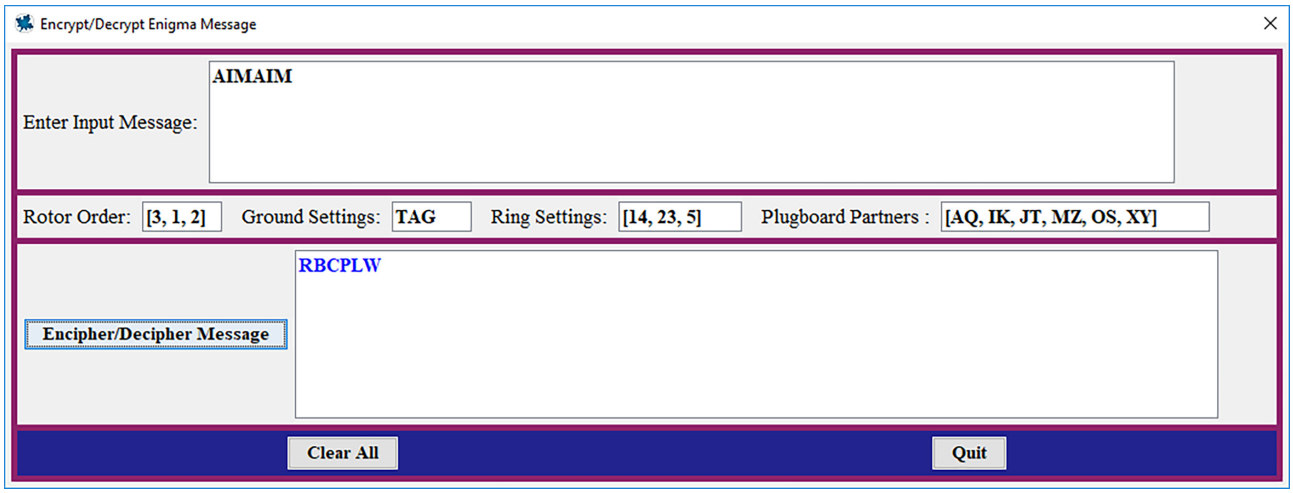

Figure 3: Using the Maplet to encrypt a message setting.

The operator would then turn their rotors so AIM was showing in the windows, and encrypt their actual message. This would result in `XYLFWMOJLVOQOBMAU`, and thus the operator would transmit the full ciphertext `RBCPLWXYLFWMOJLVOQOBMAU`.

An Enigma operator in receipt of this would decrypt the message in two steps. They would first configure their machine as directed by the codebook (which would match the codebook settings used by the originator of the message), and decrypt `RBCPLW`. The result of this would be `AIMAIM`. They would then turn their rotors so AIM was showing in the windows, and decrypt the remainder of the transmission. The result of using the Maplet to do this is shown in Figure 4.

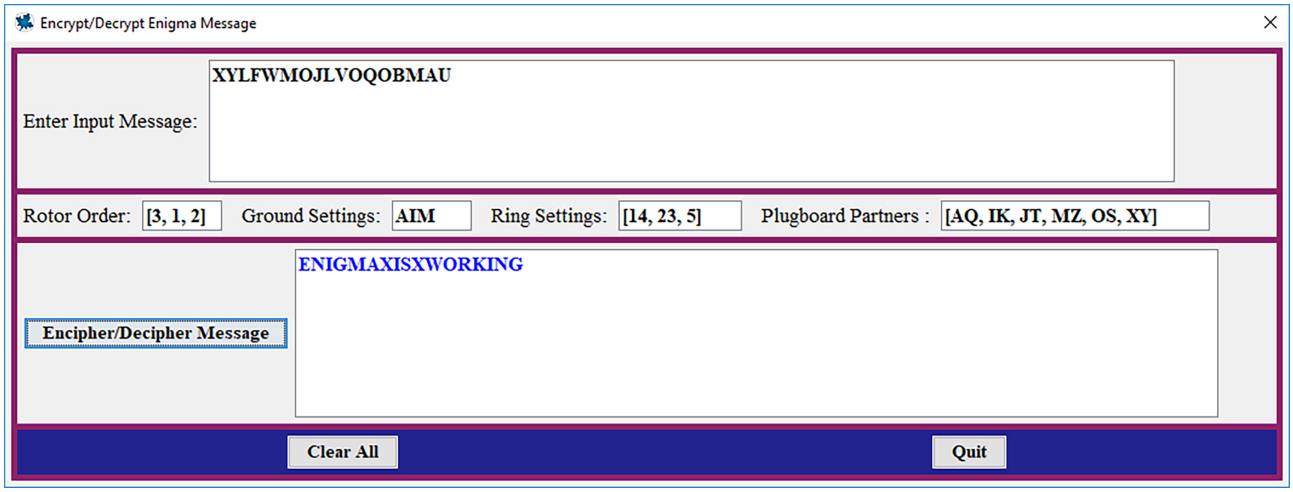

Figure 4: Using the Maplet to decrypt a message.

5 The Polish Cryptanalysis of Enigma

5.1 Analyzing Message Settings

The Polish codebreakers, working with the knowledge that the first six letters in intercepted transmissions resulted from encrypting the three letters in originators' message settings repeated, cleverly exploited this fact to discover the message settings. To see how they did this, consider the first six letters XXGCJQ in an intercepted transmission, formed using Enigma full-machine permutations labeled in order as P_1–P_6, applied to the repeated message setting AIMAIM, and with ground setting TAG. This portion of encryption can be summarized as follows.

Permutation:	P_1	P_2	P_3	P_4	P_5	P_6
Plaintext:	A	I	M	A	I	M
Window Letters:	TAH	TAI	TAJ	TAK	TAL	TAM
Ciphertext:	X	X	G	C	J	Q

Of course, at this point the Poles would not have known the actual plaintext or window letters. However, they did know the first and fourth plaintext letters had to match, from which they could glean some important information. In particular, due to the reciprocal nature of the machine, the letter to which P_1 mapped X would have to be mapped by P_4 to C. That is, $P_4(P_1(\mathrm{X})) = \mathrm{C}$, or, equivalently, $(P_1P_4)(\mathrm{X}) = \mathrm{C}$. Also, because they knew the second and fifth plaintext letters had to match, they knew $(P_2P_5)(\mathrm{X}) = \mathrm{J}$. Finally, because they knew the third and sixth plaintext letters had to match, they knew $(P_3P_6)(\mathrm{G}) = \mathrm{Q}$.

Building upon this idea, the Poles noticed certain patterns that occurred within the first six letters of intercepted transmissions originating from multiple Enigma operators all on the same day. For instance, consider the following list of 68 such examples.

```
ACFKVE  JAKJEB  SHDMZU  HEGNYQ  HOONPH  AXAKJG  BHLQZA  BNUQDO
BHGQZQ  KHMFZI  TAGYEQ  MFBGFN  BPKQMB  AYAKTG  BHMQZI  AXWKJT
CHHZZY  LHHIZY  UHFPZE  TGCYGW  XQVCIM  VZHBKY  BMNQXP  CHXZZX
DHBEZN  MEJGYC  VDJBQC  AIAKUG  PRITND  QSCHHW  BMOQXH  GHYOZZ
EHOLZH  NHHVZY  WHBWZN  XJECWV  BSJQHC  ATDKLU  GNPODL  BNZQDS
FAAAEG  OAJSEC  XBNCAP  AKAKCG  ATAKLG  BHHQZY  AIQKUJ
GHGOZQ  PAJTEC  YHDUZU  MLJGBC  HUUNRO  AXIKJD  AIRKUR
HCWNVT  QHGHZQ  ZNAXDG  BMBQXN  AVKKSB  BMJQXC  BSSQHF
IHDRZU  RHZDZS  ODNSQP  BNJQDC  TWJYOC  AIKKUB  CSTZHK
```
(6)

From the highlighted letters, for the permutations P_1 and P_4 resulting from the codebook settings on that particular day, we see that $(P_1P_4)(\mathtt{A}) = \mathtt{K}$, while $(P_1P_4)(\mathtt{K}) = \mathtt{F}$, and $(P_1P_4)(\mathtt{F}) = \mathtt{A}$. This means that $(\mathtt{AKF})$ is a cycle in the permutation P_1P_4. Similarly, using the letters in the first and fourth positions of these 68 six-letter sequences, we can see that $(\mathtt{BQHNV})$ is a cycle in P_1P_4, and in fact that in full, $P_1P_4 = (\mathtt{AKF})(\mathtt{BQHNV})(\mathtt{CZX})(\mathtt{DELIR})(\mathtt{GOSM})(\mathtt{J})(\mathtt{PTYU})(\mathtt{W})$. In the same manner, using the letters in the second and fifth positions of these 68 six-letter sequences, we can see that $P_2P_5 = (\mathtt{AEYTLB})(\mathtt{CVSHZK})(\mathtt{DQIURN})(\mathtt{F})(\mathtt{G})(\mathtt{JWOPMX})$, and using the letters in the third and sixth positions, we can see that $P_3P_6 = (\mathtt{AGQJCWTKBNPL})(\mathtt{DUOHYZSFEVMI})(\mathtt{R})(\mathtt{X})$. Notably, the permutation P_1P_4 is of type $(5,5,4,4,3,3,1,1)$, the permutation P_2P_5 is of type $(6,6,6,6,1,1)$, and P_3P_6 is of type $(12,12,1,1)$.

A couple of other insights dramatically reduced the number of possible machine configurations the Poles needed to consider. One was that since the expression $PRMLAL^{-1}M^{-1}R^{-1}P^{-1}$ summarizing the full journey through the machine can also be viewed as $P(RMLAL^{-1}M^{-1}R^{-1})P^{-1}$, the full journey through the machine is conjugate to the part of this journey that occurs between visits to the plugboard. By Theorem 2.1, we know then that the full journey through the machine is a permutation of the same type as the part of this journey that occurs between visits to the plugboard. That is, the plugboard had no effect on the cycle structure of the permutation representing the full machine. It is not hard to show that the plugboard would similarly have no effect on the cycle structure of the composition of two such permutations. As a result, ignoring the plugboard would not change the cycle structures of P_1P_4, P_2P_5, and P_3P_6 from what we observed them to be previously. Ignoring the plugboard though does significantly reduce the number of possible full machine configurations from its theoretical value given in (5) by the factor 100,391,791,500 that was contributed by the plugboard to this value.

Another insight was that since the Poles' method would fail if any rotor except the rightmost rotated during the formation of the portion of a ciphertext used in determining the configuration of the machine, and the rightmost rotor rotated in a predictable manner, they could simply ignore the ring settings. By assuming fixed common ring settings, then with regard to the rotors, the Poles only needed to consider the $26^3 = 17{,}576$ possible ground settings and $3! = 6$ possible rotor orders. This gives only $6 \cdot 17{,}576 = 105{,}456$ different ways that rotors could be configured, significantly less than the factor 1,853,494,656 that was contributed by the rotors to the theoretical value given in (5).

The Poles determined and recorded the cycle structures of P_1P_4, P_2P_5, and P_3P_6 for each of the 105,456 configurations they needed to consider. To do this, they built a machine called a *cyclometer*, which

consisted of two sets of three rotors with the reflector in between, and which they used to simulate the between-plugboard-visits operation of Enigma. When running the cyclometer, the Poles arranged both sets of rotors in one of the six possible orders, and moved the first set to reflect a combination of window letters (e.g., AIM) and the second set the combination with the rightmost rotor moved three positions forward (e.g., AIP). Current designating an input letter was then sent through the cyclometer, with the output letter recorded and itself put back into the machine to obtain a second output. This process was repeated until the original input was obtained as output, thus revealing a cycle in the permutation for the rotor order and window letters with which they started. Then, they would input a letter not in this cycle into the cyclometer and repeat the process, continuing to do this until every letter had occurred in a cycle. This would finally give the full representation of P_1P_4. They would then repeat the entire process with the window letters in the cyclometer positioned for the combinations with the rightmost rotors moved one position forward (e.g., AIN and AIQ) to find the representation of P_2P_5, and then repeat the entire process a final time with the window letters set for the combinations with the rightmost rotors moved one more position forward (e.g., AIO and AIR) to find the representation of P_3P_6.

The Poles completed this process for each of the 105,456 configurations they needed to consider, and recorded the results for P_1P_4, P_2P_5, and P_3P_6 in a (very large) table. Although no original copies of this table are known to still be in existence, we do know how the table was organized, specifically that permutations of the same type were grouped together, and listed alongside the window letters that were used on the first set of rotors in the cyclometer.

To demonstrate how this table could be used, consider Table 1, which gives a few samples of the cycles in the permutations of types $(5,5,4,4,3,3,1,1)$, $(6,6,6,6,1,1)$, and $(12,12,1,1)$ that result with the rotor order R_3, R_1, R_2 and assumed ring settings 1, 1, 1.

Table 1: Sample permutation types for rotor order R_3, R_1, R_2 and ring settings 1, 1, 1.

Permutation Type	**Window Letters**	**Cycles**
(5,5,4,4,3,3,1,1)	GDQ	(HKLQJ)(MXSYW)(BGDZ)(ENOV)(AFT)(CUI)(P)(R)
(5,5,4,4,3,3,1,1)	GED	(BIRPE)(HNVLQ)(DJXU)(GSOM)(AKZ)(CFY)(T)(W)
⋮	⋮	⋮
(5,5,4,4,3,3,1,1)	HPN	(CORLU)(DSHGZ)(BKWF)(EMNP)(AXT)(IYQ)(J)(V)
(5,5,4,4,3,3,1,1)	HYR	(ATXVB)(LOYMQ)(CRKJ)(EUNS)(DIF)(GWP)(H)(Z)
(6,6,6,6,1,1)	FYY	(ABFWNQ)(CHZOER)(DLTKGI)(PVUSXY)(J)(M)
(6,6,6,6,1,1)	GEE	(AEXJBL)(CVOHFK)(DMYTWS)(IURNPQ)(G)(Z)
⋮	⋮	⋮
(6,6,6,6,1,1)	HYJ	(AUBNOY)(CHIFKQ)(DLJTPE)(RZXWVS)(G)(M)
(6,6,6,6,1,1)	HYS	(AOSVIQ)(BMPUWN)(CRHFDT)(EYJLKZ)(G)(X)
(12,12,1,1)	GEC	(AKZUWGQRHJST)(CXIENFLYVMOD)(B)(P)
(12,12,1,1)	GEF	(AGQTCWJKLNDB)(EVMIPUSHXFOZ)(R)(Y)
⋮	⋮	⋮
(12,12,1,1)	HXU	(ANFHYXJRKLGT)(BPZEOVUQIMWC)(D)(S)
(12,12,1,1)	HYT	(BNPCWEOFXQMI)(DSUJRVTLKZYH)(A)(G)

In attacking the example given at the start of this section, we would first be looking for permutations in the full table with consecutive window letter settings that match the types of the permutations P_1P_4, P_2P_5, and P_3P_6 we discovered previously. Two candidates emerge from the small portion of the full table given in Table 1, the trio with consecutive window letter settings GED, GEE, and GEF, and the trio with consecutive window letter settings HYR, HYS, and HYT. Between these two candidates, we would then want the one whose cycles can be arranged to match up better with the cycles in P_1P_4, P_2P_5, and P_3P_6. To determine this, consider the following comparisons, with the cycles arranged to have common lengths and with the matching letters between corresponding cycles maximized.

P_1P_4: (AKF)(BQHNV)(CZX)(DELIR)(GOSM)(J)(PTYU)(W)
GED: (AKZ)(LQHNV)(CFY)(PEBIR)(GSOM)(T)(DJXU)(W)

P_2P_5: (AEYTLB)(CVSHZK)(DQIURN)(F)(G)(JWOPMX)
GEE: (AEXJBL)(CVOHFK)(PQIURN)(Z)(G)(TWSDMY)

P_3P_6: (AGQJCWTKBNPL)(DUOHYZSFEVMI)(R)(X)
GEF: (AGQTCWJKLNDB)(PUSHXFOZEVMI)(R)(Y)

P_1P_4: (AKF)(BQHNV)(CZX)(DELIR)(GOSM)(J)(PTYU)(W)
HYR: (DIF)(BATXV)(GWP)(MQLOY)(UNSE)(H)(CRKJ)(Z)

P_2P_5: (AEYTLB)(CVSHZK)(DQIURN)(F)(G)(JWOPMX)
HYS: (ZEYJLK)(CRHFDT)(BMPUWN)(X)(G)(QAOSVI)

P_3P_6: (AGQJCWTKBNPL)(DUOHYZSFEVMI)(R)(X)
HYT: (WEOFXQMIBNPC)(KZYHDSUJRVTL)(A)(G)

The cycles for GED, GEE, and GEF clearly match up better with the cycles in P_1P_4, P_2P_5, and P_3P_6 than do the cycles for HYR, HYS, and HYT. Thus, we would proceed assuming the ground setting that produced the six ciphertext letters in the example given at the start of this section was GEC, so that when the key labeled with the first plaintext letter on the keyboard was pressed, the rightmost window letter would rotate to D before current reached the rightmost rotor.

The reason why the cycles for GED, GEE, and GEF do not identically match the cycles in the permutations P_1P_4, P_2P_5, and P_3P_6 is that the latter were formed with the plugboard pairs connected, while the former were not. However, our comparisons between the cycles for GED, GEE, and GEF and those in P_1P_4, P_2P_5, and P_3P_6 actually reveal these plugboard pairs. Specifically, note that in our comparisons between P_1P_4 and GED, P_2P_5 and GEE, and P_3P_6 and GEF, the places where these comparisons fail to match all indicate the same six plugboard connections: B $\leftrightarrow$ L, D $\leftrightarrow$ P, F $\leftrightarrow$ Z, J $\leftrightarrow$ T, O $\leftrightarrow$ S, and X $\leftrightarrow$ Y.

We will now demonstrate a Maplet written by the authors that can be used to form the permutations P_1P_4, P_2P_5, and P_3P_6 from the sample intercepts given in (6), and then search a full table of the type created by the Poles in which they recorded results for each of the 105,456 configurations they needed to consider, looking for permutations with consecutive window letter settings that match the types of P_1P_4, P_2P_5, and P_3P_6. This Maplet is available for download at the link labeled [S2] in Section 7. We

begin by typing the sample intercepts given in (6) in a textbox at the top of the Maplet window, and clicking **Check Message Settings** to ensure that the intercepts provide enough information to give permutations P_1P_4, P_2P_5, and P_3P_6 that include all alphabet letters.[6] Next, we click **Compute Cycle Structure** to see the permutation types and cycles for P_1P_4, P_2P_5, and P_3P_6, and click **Check Table** to search the full table, looking for permutations with consecutive window letter settings that match the types of P_1P_4, P_2P_5, and P_3P_6.

The window to the right of **Check Table** then shows the combinations of rotor orders and ground settings from the full table for which permutations with consecutive window letter settings match the types of P_1P_4, P_2P_5, and P_3P_6. The numbers to the right of the ground settings in this window measure how well the letters between corresponding cycles match, with a number significantly higher than the rest indicating the likely best option. Since this number is highest for the ground setting GEC, we next type the rotor order and this ground setting in the given text boxes, and click **Compare Plugboard Cycle Structure**. This causes the cycles in P_1P_4, P_2P_5, and P_3P_6 and those from the table to be displayed together in the window, as shown in Figure 5.

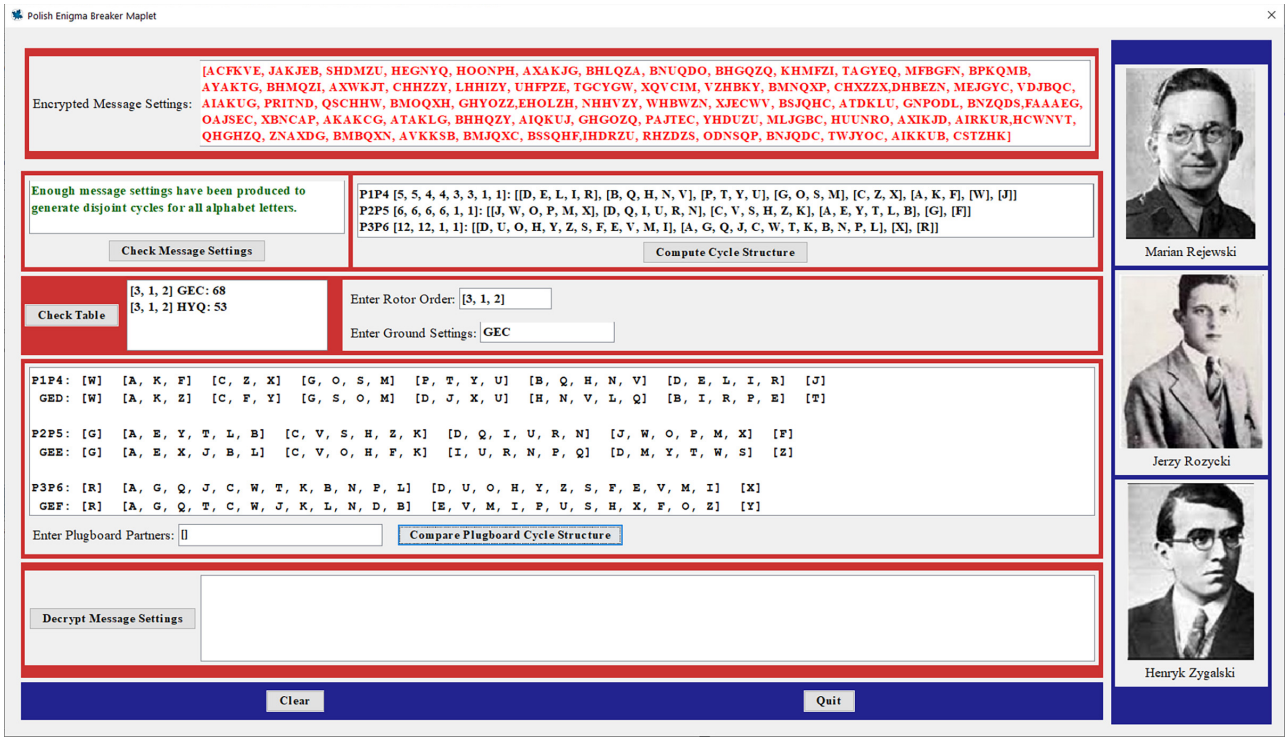

Figure 5: Using the Maplet to analyze encrypted message settings.

Seeing the cycles in P_1P_4, P_2P_5, and P_3P_6 and those from the table displayed together in the window as in Figure 5 enables us to identify the plugboard connections that would make these cycles identically match. Typing these plugboard connections in the **Enter Plugboard Partners** text box and again clicking **Compare Plugboard Cycle Structure** updates the cycles in the window, confirming when the cycles identically match that all plugboard connections have been identified correctly. Finally,

[6]Recall that this particular example involves 68 intercepts. Rejewski claimed that sometimes as many as 80 intercepts were necessary before the permutations P_1P_4, P_2P_5, and P_3P_6 would include all alphabet letters.

clicking **Decrypt Message Settings** reveals the decrypted message settings for each of the sample intercepts given in (6). The result of this is shown in Figure 6.

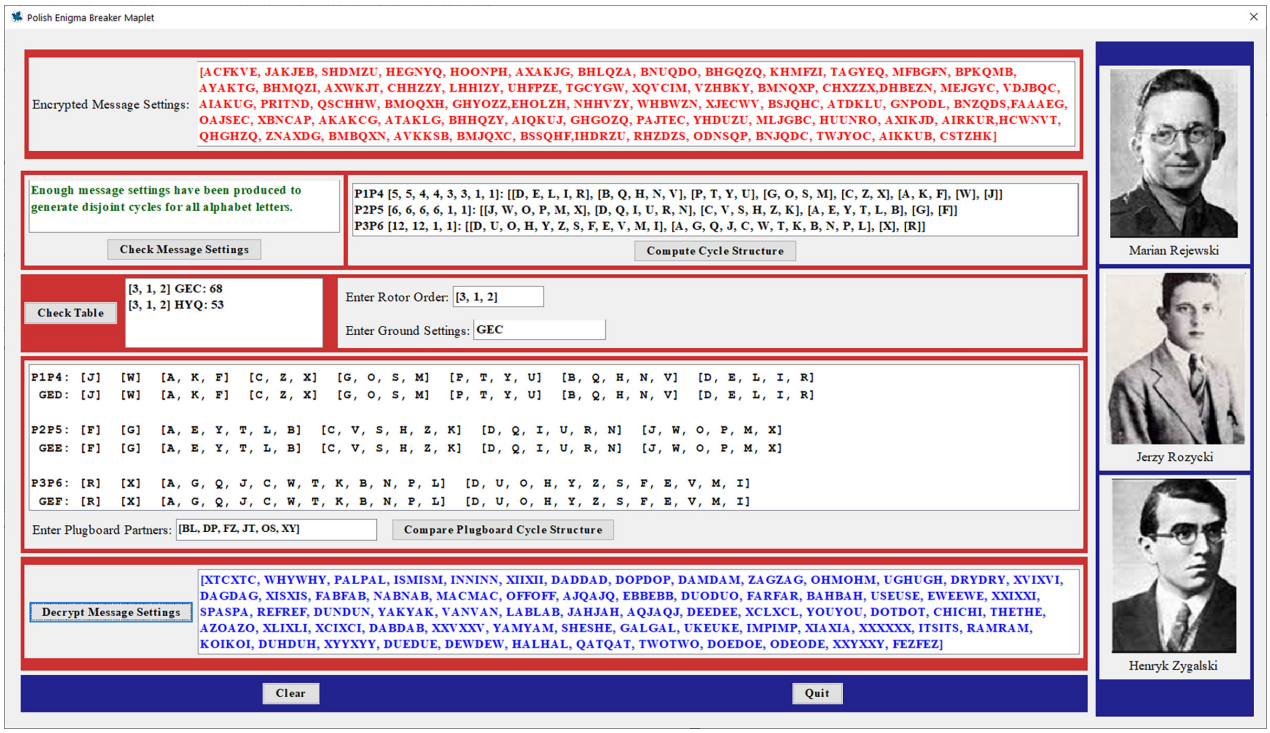

Figure 6: Using the Maplet to analyze encrypted message settings.

In this example, even using a full table of the type created by the Poles, only two combinations of a rotor order and ground setting resulted in permutations of the same type as P_1P_4, P_2P_5, and P_3P_6. Indeed, often for the Poles there were only a few such combinations. On some occasions though there were many, and sometimes several had cycles that matched up favorably and similarly with the cycles in P_1P_4, P_2P_5, and P_3P_6, making the task of finding the correct one more difficult. A detailed discussion of the cycle structures in the full table can be found in [1].

5.2 Analyzing Full Messages

To be able to decrypt full messages, the Poles still needed to figure out the ring settings given in the codebook, which to this point they had assumed to be some common fixed settings. To do this, the Poles discovered and exploited another mistake made by Enigma operators, specifically that many of their messages began with the crib TOX (actually, ANX, which would translate from German to English as TOX, the word "TO" followed by X acting as a space character). Using a message setting they had already determined though, the Poles could simply search for ring settings from among the $26^3 = 17{,}576$ possibilities that would decrypt the three letters directly following the sixth as TOX. Of course, with such a short crib, there were usually multiple ring settings that would decrypt the three letters directly following the sixth as TOX. However, each could be used in an attempt to decrypt a message in full, and then when the correct settings were found (i.e., when the full message

decrypted correctly), the ring settings from the codebook would be known, and could thus be used in the decryption of every full message intercepted on the same day.

As an example of this that demonstrates another Maplet written by the authors, which is available for download at the link labeled [S3] in Section 7, from the six-letter sequences given in (6) and their decrypted companions shown in Figure 6, consider the sequence CHHZZY and its decrypted companion FABFAB. Suppose also that the rest of the intercepted ciphertext (after the first six letters) was GCAHNYBCYHELYYDECLUQUVOGKGPLNXQXFPGLDQS, which would thus be known to have been formed using the message setting FAB, the rotor order R_3, R_1, R_2, and the plugboard connections B ↔ L, D ↔ P, F ↔ Z, J ↔ T, O ↔ S, and X ↔ Y. The Maplet window in Figure 7 shows, from among all of the 17,576 possible choices for the ring settings, the only 4 that would cause the first three encrypted message letters GCA to decrypt as the assumed crib TOX.

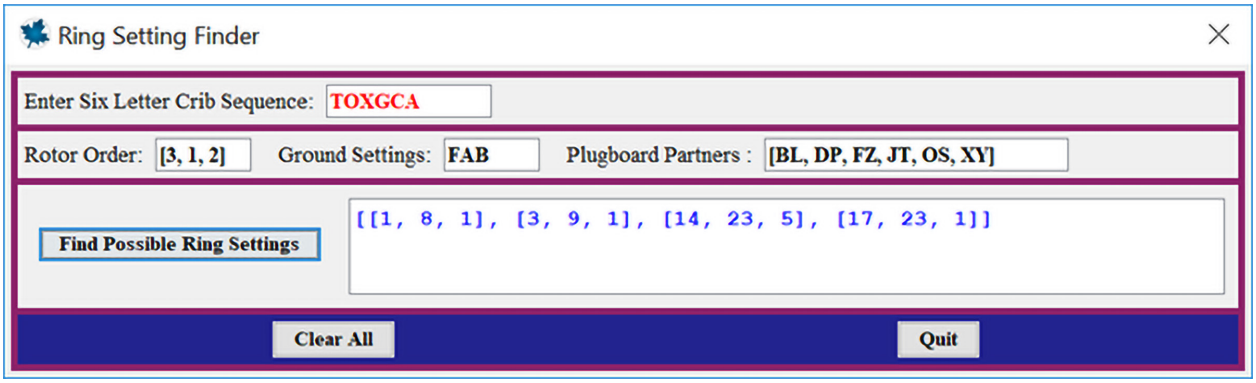

Figure 7: Using the Maplet to find ring settings.

Attempting to decrypt the full message using each of these four ring settings, which could be done using the Maplet demonstrated previously as in Figure 4, would then reveal which are actually correct, as well as the decrypted message. For example, the Maplet window in Figure 8 shows that the first choice 1, 8, 1 for the ring settings is not correct.

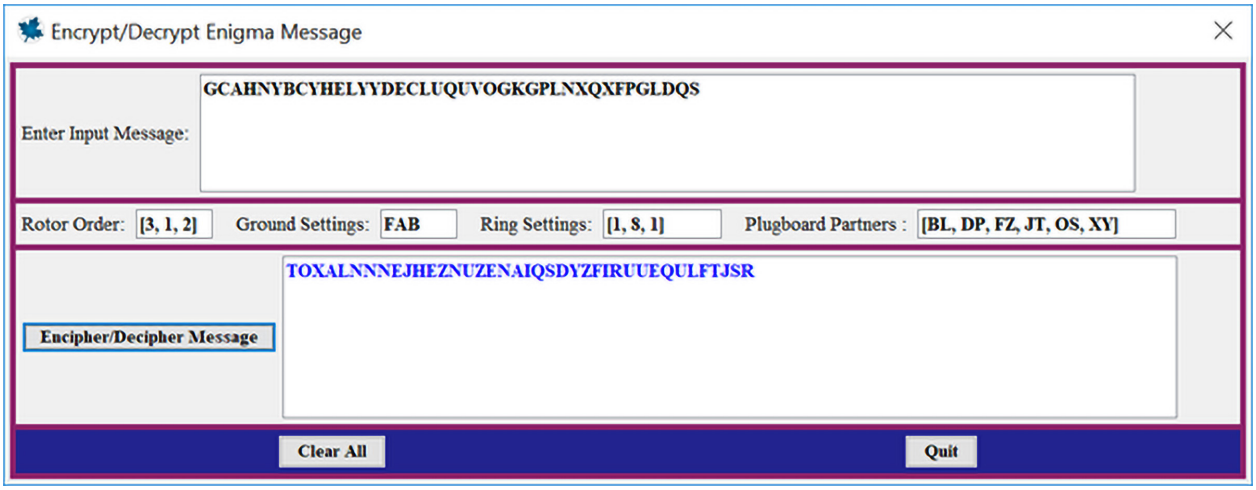

Figure 8: Using the Maplet to try to decrypt a message.

The Maplet window in Figure 9 though shows that the third choice 14, 23, 5 for the ring settings is correct, and that the full plaintext is TOXBEXORXNOTXTOXBEXTHATXISXTHEXQUESTION.

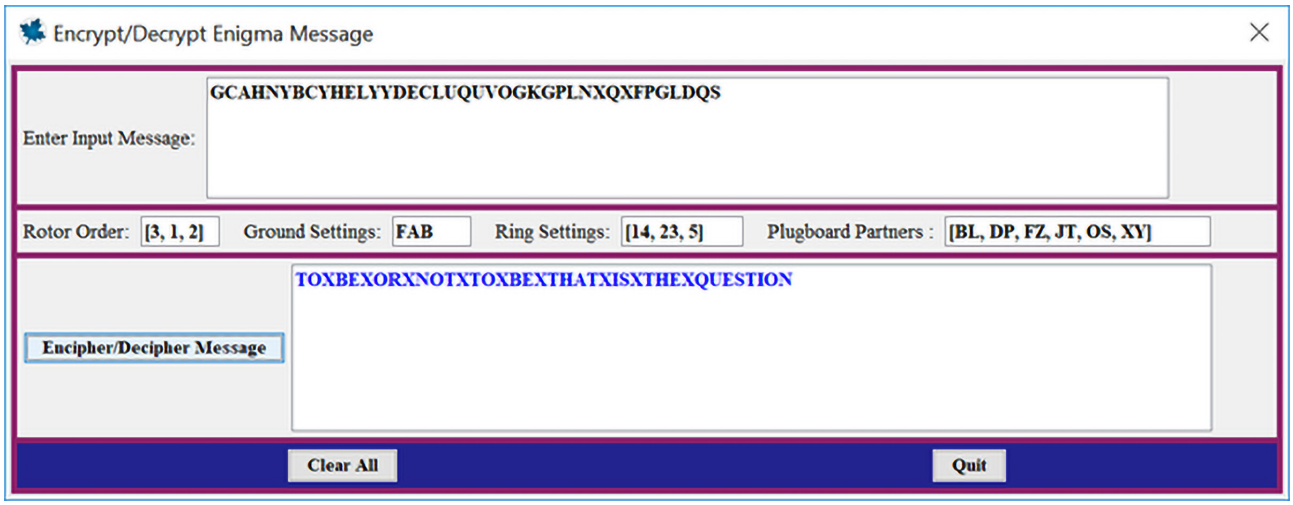

Figure 9: Using the Maplet to decrypt a message.

6 Conclusion

In this paper, we showed how Polish codebreakers led by Marian Rejewski used the mathematics of permutations to cryptanalyze German communications encrypted using the Enigma machine prior to World War II. This was demonstrated using Maplets, written by the authors, which are all available as a single download at [5]. More detailed historical descriptions of the work of the Polish codebreakers can be found in [1], [2], and [4]. In particular, the human side of the Polish codebreakers is very well portrayed in [4].

By 1938, Germany had made two additional rotor wirings available for use in their Enigma machines. This increased the number of possible rotor orders by a factor of 10 (from $3! = 6$ to $5 \cdot 4 \cdot 3 = 60$), too many for the Polish method for cryptanalyzing Enigma to be effective. However, British and American codebreakers led by Alan Turing at Bletchley Park later exploited other weaknesses in their own successful attack on the more complex machine. A detailed discussion of their efforts can be found in [3].

7 Supplementary Electronic Materials

[S1] Maplet that simulates the operations of encryption and decryption with an Enigma machine, as illustrated in Figures 3, 4, 8, and 9:
https://ejmt.mathandtech.org/Contents/v16n1p3/EnigmaMessageEncryptionDecryptionMaplet.mw.

[S2] Maplet that can be used to form the permutations P_1P_4, P_2P_5, and P_3P_6 from sample intercepts, and then search a full table of the type created by the Poles, as illustrated in Figures 5 and 6:
https://ejmt.mathandtech.org/Contents/v16n1p3/PolishEnigmaBreakerMaplet.mw.
This Maplet uses and requires the following table:
https://ejmt.mathandtech.org/Contents/v16n1p3/polishtable.mla.
This Maple also uses but does not require the following images:
https://ejmt.mathandtech.org/Contents/v16n1p3/Rejewski.jpg,
https://ejmt.mathandtech.org/Contents/v16n1p3/Rozycki.jpg,
https://ejmt.mathandtech.org/Contents/v16n1p3/Zygalski.jpg.

[S3] Maplet that can be used to find ring settings from sample intercepts and decrypted companions, as illustrated in Figure 7:
https://ejmt.mathandtech.org/Contents/v16n1p3/FindRingSettingsMaplet.mw.

References

[1] Craig P. Bauer, *Secret History: The Story of Cryptology*, Taylor & Francis, Boca Raton, FL, 2013.

[2] Chris Christensen, *Polish Mathematicians Finding Patterns in Enigma Messages*, Mathematics Magazine **80** (2007), no. 4, 247–273.

[3] Richard E. Klima and Neil P. Sigmon, *Cryptology, Classical and Modern, Second Edition*, Taylor & Francis, Boca Raton, FL, 2019.

[4] Hugh Sebag-Montefiore, *Enigma: The Battle for the Code, 75th Anniversary Edition*, Weidenfeld & Nicolson, London, 2011.

[5] Neil P. Sigmon, 2020. Maplet Download Page for Recognizing the Polish Efforts in Breaking Enigma. Available at: https://www.radford.edu/npsigmon/polish/paper.html.