ACTION THIS DAY

The Mathematics and Machinations that Bested the German Enigma

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of

Master of Science

in

Computer Science

by Jonah Weinbaum

Guarini School of Graduate and Advanced Studies
Dartmouth College
Hanover, New Hampshire

Examining Committee:

Christophe Hauser, Chair

Sergey Bratus

Sean Smith

Asher Auel

Peter Winkler

F. Jon Kull, Ph.D.
Dean of the Guarini School of Graduate and Advanced Studies

Abstract

This thesis presents a comprehensive and chronological overview of cryptographic techniques designed to break Enigma, beginning in 1932 and culminating in the creation of the Turing-Welchman Bombe. We discuss the mathematical theory and electromechanical implements used to decode one of history's greatest ciphers.

Reexamining the Bombe through the lens of modern group theory, we critique Alan Turing's estimation of the number of "stops" that the Bombe produces for various plaintext-ciphertext pairing structures. To address its limitations, we introduce a new framework for estimating the number of stops by extending John Dixon's theorem concerning the probability that uniformly distributed elements of S_n generate a transitive subgroup. Our formulation generalizes this result to compute the probability of transitivity when permutations are sampled from arbitrary distributions over conjugacy classes.

All results are supported by extensive simulation, and a full suite of implementations of various cryptographic techniques are made available via an open source repository to provide researchers with new tools to study the methods discussed herein.

Acknowledgments

I would first like to express my sincerest gratitude to the thesis committee members – Asher Auel, Sergey Bratus, Christophe Hauser, Sean Smith, and Peter Winkler. Further, this thesis would not have been possible without the extensive help and advice provided by Asher Auel, Peter Winkler, and Sean Smith.

The development of this thesis has taken roughly two years. During that time, it has been the support and encouragement of my friends and family that has allowed me to overcome the many challenges I have faced and see this work through to its completion. I would like to especially thank my father, Marc Weinbaum, for his historical expertise. Additionally, I would like to thank my best friend, George Stain, for their academic advisement. Most of all, I would like to thank my partner, Tabitha Klein, whose companionship, confidence in me, and strength carried me across hurdles I thought insurmountable. Her presence and insight has made this journey not just possible, but deeply meaningful.

In the thesis and narrative that follows, there are many figures, old and new, who pushed the bounds of cryptanalysis, mathematics, and statistics forward. It is their discovery and contributions that are at the heart of this thesis and I can only attribute the beauty and artistry of these insights to the minds which conceived them. It is also the combined efforts of modern preservationists and historians who allowed

this story to be told. Most notably I would like thank Tony Sale whose work in preserving historical documentation and rebuilding Colossus has done a great deal in keeping modern researchers engaged with this topic. I would also like to thank Steven Hosgood for his research into the relatively niche topic of Banburismus; without his prior work this section of the thesis would not have been possible. Both Tony Sale and Steven Hosgood have passed away, along with many preservationists who have kept this story alive. In that spirit, I write this thesis in an attempt to continue their legacy, honoring their work by contributing to the understanding and appreciation of this pivotal chapter in history.

Finally, I would like to dedicate this thesis to Sam Gawel who, though no longer with me, continues to guide me through even the most trying of times. His memory remains a well of faith when I face doubt and his spirit remains a compass when I lose my way.

Thesis Structure

It should be made clear that I am not a historian. As such, this thesis does not aim to be a perfect retelling of historical narrative; rather, this thesis aims to convey a progression of cryptographic attacks on Enigma. The thesis is structured to gradually build the reader's understanding of the various flaws in the Enigma machine, illustrating how and why the Bombe came to take the form that it did. Historical details are included to advance the mathematical narrative and are, to the best of my ability, confirmed to be historically accurate.

The first chapter serves as an introduction to the Enigma machine itself. We describe its construction, use, and cryptographic strength. We further provide mathematical description of the machine using permutation theory.

The second chapter discusses the first cryptographic attacks on Enigma which were employed by Polish cryptanalysts. We first discuss the reconstruction of the German Enigma via mathematical deduction about the internal wirings of the machine. We further discuss both manual and electromechanical methods for deducing daily keys.

The third chapter focuses on British cryptanalysis of the Enigma during the war. While we describe a variety of methods used by Bletchley Park to deduce daily keys, we primarily focus on the creation and construction of the Turing-Welchman Bombe.

We rigorously describe the mathematical reasoning implicit in the machine. We further describe extensions added to the original design of the Bombe. Finally, we describe the process by which naval Enigma messages were deciphered using the Bombe, known as Banburismus. We describe and justify this complex system of linguistic frequency analysis.

These three chapters function as an expository resource aimed at mathematically inclined readers seeking depth and rigor regarding cryptographic attacks on Enigma.

The fourth chapter examines the mathematical model by which the Bombe's efficacy was described by Turing. We first justify Turing's model using John Dixon's theorem regarding the probability that two uniformly distributed permutations in S_n form a transitive subgroup [16]. We also present several critiques of Turing's model and illustrate its inability to make predictions which align with the ground-truth operation of the Bombe. To remedy these issues, we present an alternative model which closely predicts the true behavior of the Bombe. To do this, we created a generalized version of Dixon's Theorem 4.5 which allows for the calculation of the probability that two permutations, pulled from a non-uniform distribution of cycle types in S_n, form a transitive subgroup.

The final chapter discusses directions for future research on the subjects and methods described herein. We also describe the open-source repository which is used throughout this thesis to generate tables, calculate relevant values, and simulate both manual and electromechanical methods of attacking Enigma.

Academic Contributions

This thesis presents a collection of novel contributions to the study of both cryptographic history and mathematics.

We present linear, rigorous, and extensive descriptions of cryptographic attacks on Enigma. To our knowledge no such prior compilation of mathematical materials exists on the subject of Enigma of comparable scope.

We provide a unique mathematical description of the Bombe, using group-theoretic language to formalize its electromechanical elements. This allows us to more precisely illustrate the mathematical contradictions the Bombe uses to eliminate certain rotor positions. This further allows us to abstract problems associated with analyzing the Bombe to explicit problems in the field of group theory.

We devise and prove a generalization of Dixon's Theorem. The original theorem statement regards the probability that two uniformly distributed permutations in S_n form a transitive subgroup [16]. We extend this theorem to allow for the calculation of the probability that two permutations, pulled from a non-uniform distribution of cycle types in S_n, form a transitive subgroup 4.5.

We further provide researchers with a repository of open-source tools used in the

creation of the thesis [39]. Of particular note, we have created a simulation of the Bombe which provides a detailed internal view of the wirings of the machine. Further, we have provided code which allows for the construction and use of Zygalski sheets. We additionally wrote several programs which allow for the computation of scoring, distance, dummyismus, and repeat sheets used for the Banburismus procedure. Most significantly, we implemented the generalized Dixon's Theorem as an algorithm in `python`, allowing for researchers to compute the probability of two permutations forming a transitive subgroup pulled from an arbitrary distribution over cycle types. Finally, we provide a script which allows researchers to quickly estimate the number of stops we expect the Bombe to encounter for a particular menu arrangement.

Requisite Background

The subject matter of this thesis concerns both the mathematical and mechanical means by which the Enigma encryption scheme was broken over time.

With respect to mathematical background, the reader is expected to have an introductory knowledge of abstract algebra, in particular a solid grasp of permutation theory. Further, many topics will delve into probabilistic and combinatoric analysis of language frequency, key spaces, and the effectiveness of various methods. For this reason the reader is expected to have a comfortable understanding of introductory combinatorics and Bayesian statistics.

With respect to mechanical background, the text requires limited knowledge of components such as transistors and relays. Circuit diagrams will be used sparingly, or will be abstracted, so as to reduce the need of an electrical engineering background.

Contents

Introduction

While the allied armies fought tooth and nail, in trenches and tanks, for the fate of the world, another army was amassed in the small radio factory of Bletchley Park. Mathematicians, engineers, linguists, and chess players, armed with paper and pencil, wire and punch-cards, took up their own battle.

Hitler's best-kept secrets lay hidden behind a math problem. What follows is the story of some of history's greatest minds and their work to solve that problem. It is a story of secrecy and spies; ingenious perspectives, laborious work, and machines built to think faster than any human ever had.

At the center of this story is the Enigma machine, a feat of cryptography so vast in its possible arrangements that it was thought unbreakable. Breaking it required more than mathematical trickery; it required coordination, luck, and a cryptographic intelligence team the likes of which the world had never seen. Winston Churchill saw this math problem for what it was: the key to saving millions of lives. Understanding the project's urgency, he issued a memo demanding that the utmost attention and resources be given to the Bletchley team. Churchill tagged this memo with a red stamp reserved for matters of the highest priority [9, p. 336]. It read:

Action This Day

1

The Enigma

The Enigma machine was used extensively by the Germans to encipher communications prior to and throughout World War II. German stratagems like the *blitzkrieg* required quick radio communication, so to ensure that the allied powers were not able to read messages, they encoded all radio signals using the Enigma machine. Breaking the Enigma would allow the allied powers to freely intercept and decrypt all naval, air force, and military command – offering them time to counter, defend, and retaliate appropriately.

Section 1.1

The Machine

Figure 1.1: Enigma model I [28]

Throughout this thesis, the model I Enigma is chosen to represent our canonical machine as this was the most common version used during World War II with over 20000 being produced [10]. Further, it was used by the *Heer* (Army), *Luftwaffe* (Air Force), and *Kriegsmarine* (Navy), making this a prime target for attack by cryptanalysts. Many models existed, each with varying layouts, key spaces, and use-cases; however, the central ideas that are discussed in this thesis can generally be adapted to work on other models.

At its most basic function, the Enigma (once set up) is a keyboard whose letters,

when depressed, illuminate bulbs of a corresponding keyboard layout. The operator presses keys of the desired plaintext and copies the output of the illuminated bulbs to obtain the enciphered text. The actual mechanism of this encipherment requires several mechanical components in a complex arrangement.

1.1.1. The Plugboard

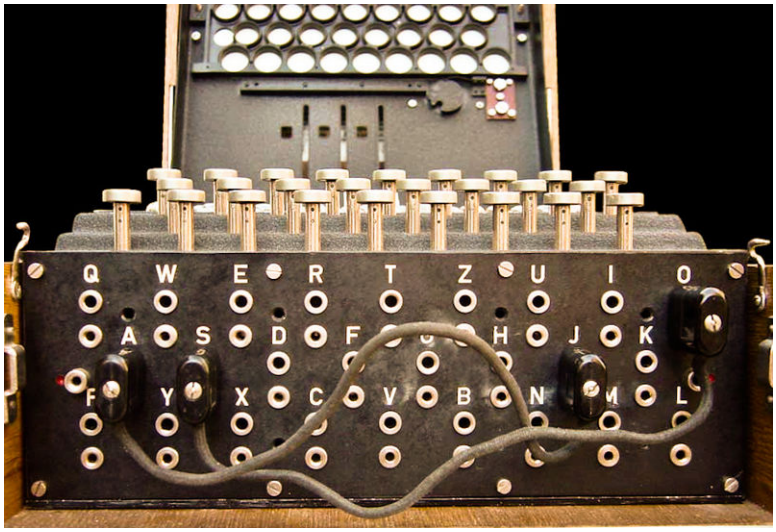


Figure 1.2: Enigma plugboard [25]

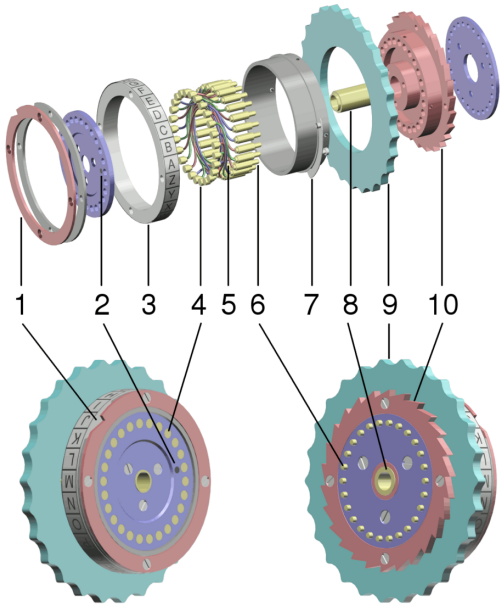
Upon a key press, the electrical current corresponding to this letter is sent to a mechanism known as the **plugboard**. From an operator's perspective, the plugboard was a series of ports, one for each letter, along with 10 cables which could connect these ports. When two letters are connected via a cable (e.g. A and Z), the plugboard will send current corresponding to a letter to the opposite letter (e.g. A goes to Z and vice versa). If no cable is plugged in to a letter (e.g. D has no cable), then the plugboard simply will return a current corresponding to this same letter (e.g. D). When a letter is connected to another through the plugboard, we will say that the letters are **steckered** to one another. Assuming all 10 cables are used, this means that the plugboard can be represented as a permutation on 26 letters with $2^{10}1^6$ cycle

type[1]. One such permutation could be

$$(\texttt{HR})(\texttt{AT})(\texttt{IW})(\texttt{SK})(\texttt{UY})(\texttt{DF})(\texttt{GV})(\texttt{LJ})(\texttt{BQ})(\texttt{MX})(\texttt{C})(\texttt{E})(\texttt{N})(\texttt{O})(\texttt{P})(\texttt{Z}).$$

In general, we will denote the permutation corresponding to a plugboard as S.

1.1.2. The Rotors



(a) Disassembled rotor [38]

(b) Enigma rotor V entry side [27]

(c) Enigma rotor V exit side [26]

Figure 1.3: Enigma rotor

[1]We will denote cycle types in this thesis in the format $\lambda_1^{m_1} \dots \lambda_k^{m_k}$ where λ_i is the length of a cycle and m_i is the multiplicity of this cycle length. We will often write the full notation $1^{m_1} \dots n^{m_n}$ in which m_i may be 0.

The Enigma model I used three rotors, selected from a larger set of rotors, the number of which changed depending on the military branch we examine. At a minimum, all three branches of the military eventually had access to five rotors labeled by their roman numeral equivalents (i.e. I, . . . , V). Each rotor encoded a unique permutation from 26 brass input contacts to 26 brass output contacts by simply connecting each input/output pair in the permutation with a wire as in Figure 1.3 (a)-(4,5,6). The contacts represent the letters in alphabetical order moving in a clockwise manner relative to the entry side of the rotor. The input contacts of the rotor were often marked with a white dot indicating which contact corresponded to A, but in general the A contact is found by looking at the contact immediately above the numeral indicator [12]. In Figure 1.3 (b), this is the topmost contact, located directly above the text V.

On their own, these rotors would prove to be very poor cryptographic devices, as they are just substitution ciphers, which are vulnerable to frequency analysis and, if found by the enemy, would serve no purpose whatsoever. Therefore, these rotors were designed to rotate, which served to change the substitution at each stage of encryption.

Because these rotors rotate, it is best to differentiate between contact letters and contact positions. When we give the permutation corresponding to a rotor as in figure 1.4 we are referring to the A contact as the specific contact denoted by the marker dot and the B contact as its next contact turning clockwise. When referred to in this context, we will use the word "contact." On the other hand, when we say that an electrical current corresponding to A enters a rotor, we mean that the current enters the contact at the topmost *position* of the rotor, even if the rotor has rotated now so

that that contact is not the contact with the marker dot. When referred to in this context we will use the word "position" so as to disambiguate from the prior context. That is to say, a contact and a position need not be the same. For example, contact `A` can be in position `B`. This occurs when the pin with the marker dot adjacent to it is one pin away from being at the top of the rotor.

Turnover. Each rotor had on its entry-side, a notch next to each contact as in Figure 1.3 (b). A pawl attempted to engage the notch and move the rotor forward by one contact each key press. On the exit-side, each rotor was equipped with a smooth ring with only a single notch breaking it as in Figure 1.3 (c). This is known as the **turnover notch**. Assuming the rotor functions in isolation the pawl will engage the entry notches during each key press thus moving the rotor forward by one until, after 26 presses, the rotor returns to its original position. However, if we have two rotors, say rotor M and N, such that rotor M has its entry contacts placed adjacent to the exit contacts of rotor N; then, the smooth ring of the rotor N will occlude the notches of rotor M thus preventing the pawl from engaging. That is, except at the location where the turnover notch is located. The pawl will then only be able to rotate rotor M if it aligns with the turnover notch of rotor N.

Now consider three rotors, rotors L, M, and N, arranged left to right from an operator's perspective. Then electrical current first enters rotor N, followed by rotor M, and finally rotor L. Rotor N will have no rotor's smooth ring occluding its notches so the pawl is free to engage rotor N at every key press. Thus rotor N will always turn at each press of the key. Rotor M, however, will only turn at the position at which rotor N's turnover notch aligns with the pawl (save for a case we will shortly discuss), meaning that for each full rotation of rotor N, rotor M will move by one contact. Finally, rotor L will only move when rotor M's turnover notch is aligned

with the pawl, meaning that rotor N must make 26 full rotations before rotor L will move by one contact.

$$\begin{pmatrix} \texttt{A} & \texttt{B} & \texttt{C} & \texttt{D} & \texttt{E} & \texttt{F} & \texttt{G} & \texttt{H} & \texttt{I} & \texttt{J} & \texttt{K} & \texttt{L} & \texttt{M} & \texttt{N} & \texttt{O} & \texttt{P} & \texttt{Q} & \texttt{R} & \texttt{S} & \texttt{T} & \texttt{U} & \texttt{V} & \texttt{W} & \texttt{X} & \texttt{Y} & \texttt{Z} \\ \texttt{V} & \texttt{Z} & \texttt{B} & \texttt{R} & \texttt{G} & \texttt{I} & \texttt{T} & \texttt{Y} & \texttt{U} & \texttt{P} & \texttt{S} & \texttt{D} & \texttt{N} & \texttt{H} & \texttt{L} & \texttt{X} & \texttt{A} & \texttt{W} & \texttt{M} & \texttt{J} & \texttt{Q} & \texttt{O} & \texttt{F} & \texttt{E} & \texttt{C} & \texttt{K} \end{pmatrix}$$

Figure 1.4: Rotor `V` permutation [12]

Double Stepping. Arguably the most confusing aspect of the mechanics of the Enigma machine is that of the **double step**. We just said that the each of the left two rotors (M and L) can *only* be moved when the turnover notch of its right-hand adjacent rotor (N and M respectively) is aligned with the pawl. For L this is true. However, for rotor M, we must consider that even if rotor N's turnover notch is *not* aligned with the pawl, rotor M's own turnover notch *may* be aligned with the pawl. In this case, the pawl will engage and move L; however, the turnover notch is still connected to rotor M, so when the pawl engages M's turnover notch it will not only move L, but it will also move M (just from the opposite side that we might expect).

We will illustrate this effect by example. Suppose our rotors N, M, and L had only three positions (1, 2, and 3). They will each have a turnover notch aligned with a pawl when at position 1. We will walk through a full cycle of turnover of the leftmost rotor L step-by-step. We have

- (Step 0) $L = 2$, $M = 2$, $N = 3$ – This represents our initial position
- (Step 1) $L = 2$, $M = 2$, $N = 1$ – Pawl engages N and steps it forward
- (Step 2) $L = 2$, $M = 3$, $N = 2$ – Pawl engages N and N's turnover notch engages M.

- (Step 3) $L = 2$, $M = 3$, $N = 3$ – Pawl engages N and steps it forward
- (Step 4) $L = 2$, $M = 3$, $N = 1$ – Pawl engages N and steps it forward
- (Step 5) $L = 2$, $M = 1$, $N = 2$ – Pawl engages N and N's turnover notch engages M.
- (Step 6) $L = 3$, $M = 2$, $N = 3$ – Pawl engages N. M's turnover notch engages L but this also has the effect of rotating M as well, even though N's turnover notch is *not* aligned. Observe that M has now moved twice in two steps, hence the name "double step".

From this example, we note that on an Enigma with 26 letters, the leftmost rotor L moves every $26 \cdot 25$ steps. We might expect this to occur every $26 \cdot 26$ steps, but because of double stepping the period is shortened.

Rotation. Consider the effect of a rotor turn on rotor `V`, whose internal wiring is described in figure 1.4. In a default position in which contact `A` is at position `A`. After pressing a key, the rotor will turn resulting in contact `B` now being in position `A`. This means that an input current entering at position `A` will go into contact `B`, be routed through the permutation and exit at contact `Z` which now is at position `Y` due to the rotation. This is to say that rotating the rotor has the effect of shifting an input letter forward by 1 (mod 26) and the output letter back by 1 (mod 26).

To encode the effect of rotation as a permutation consider

Definition 1.1. The *Caesar permutation* (denoted P) is the permutation taking a letter to the next letter in alphabetical order (mod 26). Its two-line permutation

notation is

$$\begin{pmatrix} \texttt{A} & \texttt{B} & \texttt{C} & \texttt{D} & \texttt{E} & \texttt{F} & \texttt{G} & \texttt{H} & \texttt{I} & \texttt{J} & \texttt{K} & \texttt{L} & \texttt{M} & \texttt{N} & \texttt{O} & \texttt{P} & \texttt{Q} & \texttt{R} & \texttt{S} & \texttt{T} & \texttt{U} & \texttt{V} & \texttt{W} & \texttt{X} & \texttt{Y} & \texttt{Z} \\ \texttt{B} & \texttt{C} & \texttt{D} & \texttt{E} & \texttt{F} & \texttt{G} & \texttt{H} & \texttt{I} & \texttt{J} & \texttt{K} & \texttt{L} & \texttt{M} & \texttt{N} & \texttt{O} & \texttt{P} & \texttt{Q} & \texttt{R} & \texttt{S} & \texttt{T} & \texttt{U} & \texttt{V} & \texttt{W} & \texttt{X} & \texttt{Y} & \texttt{Z} & \texttt{A} \end{pmatrix}.$$

If we denote the permutation corresponding to rotor `V` in default position as η. Then after r rotations, to get our new permutation we must first shift each input letter forward by r and each output letter backwards by r. This can be encoded via the Caesar permutation as follows

$$P^{-r}\eta P^{r}.$$

x It should be noted that current will only flow through the machine *after* the rotation has taken place. Thus encrypting a letter with the window indicating `AAA` is really going to send current through the rotors with the window indication `AAB` since the rightmost rotor will turn before the encryption happens.

Outer Ring. The rotors were additionally equipped with an outer ring with letters `A` to `Z` moving clockwise relative to the entry-side of the rotor as in Figure 1.3 (a)-(3). Alternately some rings had numerical values ranging from `01` to `26`. The outer ring was not fixed to the underlying rotor and could be moved. Most importantly, this outer ring is what contained the turnover notch – that is, the component in Figure 1.3 (a)-(3) was connected to the component (a)-(1). Thus, by moving the outer ring, we change where turnover occurs relative to the internal wiring of the rotor.

To consider the effect of this ring, consider if an operator were instructed to place the ring such that the outer ring's `02` label was placed over contact `A`. Once the rotor is closed inside the machine the operator can now only see the letters indicated by the outer ring appearing in a small window. If he moves the ring's label `01` to be in

the window, then contact `Z` is now in position `A`. This means that an input current entering at position `A` will go into contact `Z`, be routed through the permutation and exit at contact `K` which now is at position `L` due to the ring setting. This is to say that moving the ring setting has the effect of shifting an input letter back by 1 (mod 26) and the output letter forward by 1 (mod 26). As in the prior discussion on rotor rotation, if we denote the permutation corresponding to rotor `V` in default position as η. Then shifting the ring by r letters, we get a new permutation by first shifting each input letter backwards by r and each output letter forwards by r. This can be encoded via the Caesar permutation as follows

$$P^r \eta P^{-r}.$$

In this sense, we can think of rotor rotations and ring adjustments as having inverse effects. In fact, if we ignore turnover, setting the ring forward by r letters and then rotating the rotor forward by r letters is equivalent to having left the ring setting and rotor in its default position. That is to say, for cases where turnover does not occur, the ring setting (*Ringstellung*) and which letter we decide to show in our window (*Grundstellung*) really represent one singular component of our key space since we can always consider our ring setting as being at `A` by just shifting which letter we display in our window. However, consider that changing the ring setting also changes where the turnover occurs relative to the internal wiring of a rotor. This means that for rotors M and L which are affected by turnover, the ring setting does add to the key space since it has effects which are independent of the window setting.

1.1.3. The Reflector

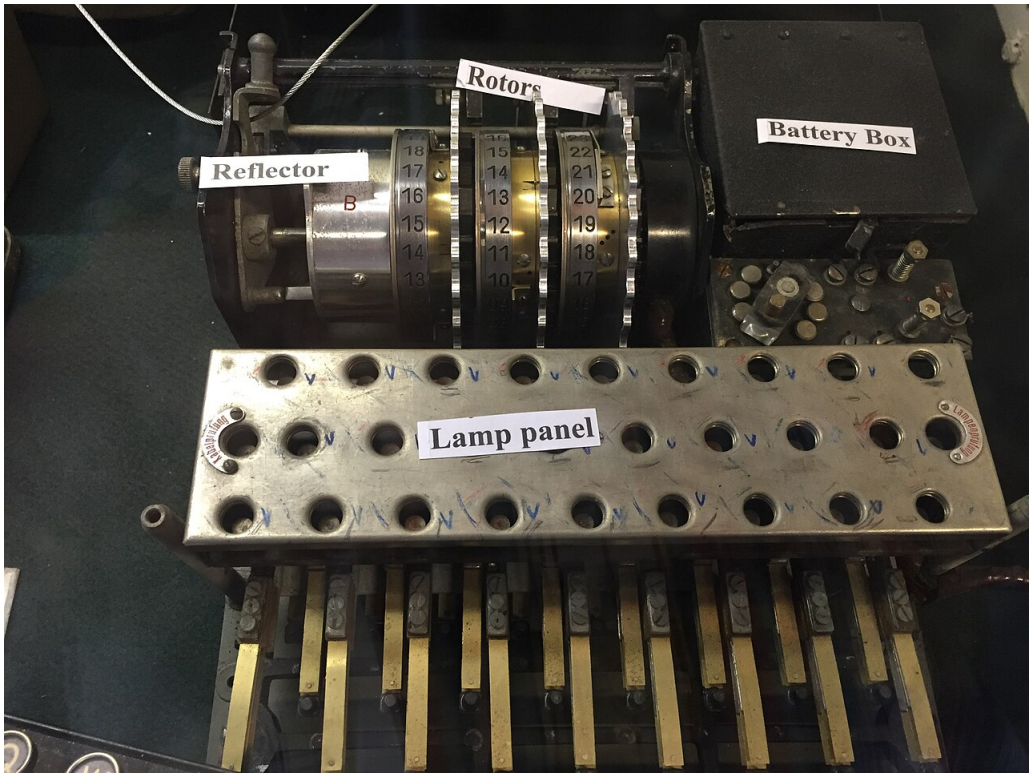


Figure 1.5: Enigma internals illustrating reflector placement [3]

After the current travels through all three rotors, it ends up at the reflector which we will denote R. The reflector is specially designed so that its permutation consists only of disjoint transpositions, that is it has a 2^{13} cycle type. This means that reflector simply swaps letters in a set of 13 pairs.

Rather than having an entry and an exit side each with their own contacts, the reflector has one side of contacts. Current enters at a particular contact and is routed through the permutation back out this same side at a different contact. The reason for this is that the reflector's job is to send current through the exact inverse of the process we have described up to this point. That is, after exiting the reflector, current

travels back through the rotors (now entering at the exit contacts of the leftmost rotor), travels back through the plugboard, and now ends up at a lamp light indicating the enciphered letter.

Section 1.2

Key Size

Plugboard Setting (*Steckerverbindungen*). 10 plugboard wires are selected in total. The first wire must connect 2 out of 26 letters giving us $\binom{26}{2}$ locations this wire can be placed. We now select a wire for the next two letters giving us $\binom{24}{2}$ remaining locations. We continue in this fashion until having placed 9 wires leaving us $\binom{8}{2}$ remaining choices. In total, we have found

$$\begin{aligned}&\binom{26}{2}\binom{24}{2}\cdots\binom{8}{2}\\&=\frac{26!}{2!\cdot 24!}\frac{24!}{2!\cdot 22!}\cdots\frac{8!}{2!\cdot 6!}\\&=\frac{26!}{2^{10}\cdot 6!}\end{aligned}$$

possible wire arrangements. Of course, the order in which we select the 10 wires does not matter so we have over-counted and must therefore divide this value by 10! wire orderings. Therefore, we have

$$\frac{26!}{2^{10}\cdot 10!\cdot 6!}$$

plugboard settings.

Rotor Selection (*Walzenlage*). From the 5 rotors in circulation, 3 rotors in some order were needed to operate the Enigma machine. There are thus $\binom{5}{3}$ total selections of 3 rotors each of which can be ordered in 3! ways, giving us $\binom{5}{3}\cdot 3!$ possibilities.

Ring Setting (*Ringstellung*). Recall that the only rotors for which the ring setting actually adds to the keyspace is the 2 rightmost rotors since this setting changes where turnover occurs and only the 2 leftmost rotors are affected by turnover. Since each ring can be setting corresponds a letter from the 26 letter alphabet, this gives us 26^2 possibilities.

Window Setting (*Grundstellung*). A window setting specified 3 letters from the 26 letter alphabet giving us 26^3 possibilities.

Reflector Selection. While there are multiple reflectors each of which saw varying levels of usage during World War II, most machines generally stuck to a fixed reflector known as `UKW-B`. Therefore, this does not factor into our key space but is worth noting if other reflectors are being considered.

Total Key Size. Putting together all these components of an Enigma's key settings we derive the following expression for the total number of keys

$$\frac{26!}{2^{10} \cdot 10! \cdot 6!} \cdot \binom{5}{3} \cdot 3! \cdot 26^2 \cdot 26^3 \approx 1.07 \cdot 10^{23} \approx 2^{77}$$

resulting in a roughly 77-bit key space.

With such a large key space, it seemed to the Germans that Enigma was unbreakable. This was an era before computers could churn through massive key spaces in a matter of hours. Any attempt to break Enigma was going to require an attack more intelligent than brute-forcing.

In Chapter 2, we will see how Polish cryptanalyst made use of a flaw in the Enigma protocol, along with some clever mathematics, to construct one of the earliest at-

tempts at breaking the cipher. To provide the requisite background, we will examine the protocol by which German Enigma operators sent messages, as well as provide a mathematical formalism for the machine.

Section 1.3

The Enigma Protocol

Suppose Alice and Bob are two radio operators (prior to September 15, 1938) who want to communicate securely [30, p. 546]. Each are supplied an Enigma machine along with a key sheet indicating the keys for a given day. The key sheet contains the following information:

- the choice and order of rotors, known as the *Walzenlage* – at this point only three rotors `I`, `II`, and `III` were in production [29, p. 214],
- the ring setting, known as the *Ringstellung*,
- the plugboard settings, known as the *Steckerverbindungen* – at this point only 6 jacks were used [24, p. 242],
- the window setting, known as the *Grundstellung*.

A key sheet at this time may have looked along these lines.

Datum	*Walzenlage*	*Ringstellung*	*Steckerverbindungen*	*Grundstellung*
31.	I II III	10 14 02	BF SD AY QN LP JE	VAR
30.	I II III	04 25 01	UE PL AY IN QR SZ	PAQ
29.	I II III	13 11 06	WJ VD PO TY BL NK	ZJB
⋮	⋮	⋮	⋮	⋮

Figure 1.6: Mock Enigma key sheet pre-1938

Alice wants to encrypt the message HELLO WORLD on the 31st of the month using the key sheet in figure 1.6. She opens the machine and places the rotors in the machine from left to right as I, II, III[2]. She then aligns the 10 indicator on the leftmost drum to be inline with the A contact, and similarly for the remaining ring settings. Alice now closes the machine and rotates the rotors so that they display VAR (or in numerals 22 01 18) in the window of each rotor. Finally, Alice connects B and F in the plugboard and similarly for the remaining plugboard settings. Alice now chooses a random message key, this is a three letter trigram called the *Spruchschlüssel*. Alice chooses her message key as LSG and is now ready to encrypt her message.

First, Alice will encrypt her message key LSG twice, producing the hexagram

MRF NZJ

Alice now sets her window setting to her message key LSG and begins enciphering her message HELLO WORLD to produce

JYOKZ OIBGO

Alice now sends the following message to Bob

MRF NZJ JYOKZ OIBGO

Bob receives this message. He gets his key sheet and sets up his machine as the key sheet describes for that day. He now types in the first six letters of the message to get back Alice's message key, which will look like

LSG LSG

We can now see why we doubly encoded the message key. If Bob did not see the same trigram repeated twice he would know that either he or Alice set up their machine

[2]Prior to January 1, 1936, the sequence of rotors was only changed once a quarter [29, p. 223].

incorrectly, or that Alice incorrectly typed in her message key. Bob could then tell Alice to correct the message. Now equipped with Alice's message key, Bob sets his machine window to `LSG` and begins typing in the remainder of the message to recover the plaintext

`HELLO WORLD`

It should be noted, the procedure described here is a facsimile of the actual procedure meant to only convey the components that are cryptographically significant. In practice, additional information was sent alongside the message such as time of transmission and radio station of origin. Further the message itself was to be encoded with particular rules, for example, spaces would be denoted with `X`. As we discuss changes to Enigma protocol throughout this thesis we will leave such details out.

Section 1.4

Enigma as a Permutation

Recall that from the keyboard, current will enter the plugboard (S), followed by the rightmost rotor (N), middle rotor (M), leftmost rotor (L), and the reflector (R), only to return through each of these components. Then at default position the Enigma machine can be represented as a permutation

$$\pi_0 \coloneqq S^{-1}N^{-1}M^{-1}L^{-1}RLMNS$$

Additionally recall that if we ignore turnover the ring setting and window setting effectively represent one singular setting. Then, by proper adjustment of permutations (L, M, and N) we can consider any Enigma setting as beginning in such a state as described by π_0. Further, each subsequent key-press will bring us to a new state with

a new permutation given by

$$\pi_i := S^{-1}P^iN^{-1}P^{-i}M^{-1}L^{-1}RLMP^{-i}NP^iS$$

where i describes the number of times we have depressed the keyboard. Of course, turnover does exist and does matter; however, if we are examining only the first few letters (say l) of a message being encrypted, we have a $\frac{25}{26}$ chance of no turnover occurring at each step and so we have a $(\frac{25}{26})^l$ chance of no turnover occurring during our initial stages on encryption. For small l this is a reasonably high probability. We will see that the first Enigma code-breakers made use of this fact to simplify their model of the machine to the above permutation π_i.

1.4.1. Cycle Type

Consider the structure of the permutation π_i. We have

$$\begin{aligned}\pi_i &= S^{-1}P^iN^{-1}P^{-i}M^{-1}L^{-1}RLMP^{-i}NP^iS \\ &= (LMP^{-i}NP^iS)^{-1}R(LMP^{-i}NP^iS).\end{aligned}$$

That is, π_i is simply the reflector permutation R pre-composed and post-composed with a permutation and its inverse respectively. This leads us to the following definition,

Definition 1.2. Let G be a group. We say two elements $a, b \in G$ are **conjugate** if $\exists\ g \in G$ s.t. $a = gbg^{-1}$ [4, p. 50].

In this way, we can shorten our above observation to say that $\forall\ i \in \mathbb{N}$ we have that π_i and R are conjugate permutations. This point is true regardless of turnover since the permutations being pre-composed and post-composed with R are always inverse

permutations. Now consider the following lemma,

Lemma 1.3. *Suppose* $\rho = (a_0 \dots a_{k-1}) \in S_n$[3] *is a* k*-cycle. Then* $\forall\ \tau \in S_n$ *we have* $\tau\rho\tau^{-1}$ *is a* k*-cycle.*

Proof. Since $\tau \in S_n$ is a bijection, to show how $\tau\rho\tau^{-1}$ acts on $\{1, \dots, n\}$ it suffices to show how it acts on $\{\tau(1), \dots, \tau(n)\}$. We consider how $\tau\rho\tau^{-1}$ acts on elements of the form $\tau(a_i)$ for a fixed $i \in \{0, \dots, k-1\}$.

$$\begin{aligned}\tau(a_0 \dots a_{k-1})\tau^{-1}(\tau(a_i)) &= \tau(a_0 \dots a_{k-1})(a_i) \\ &= \tau(a_{(i+1) \bmod k})\end{aligned}$$

Then we know $\tau\rho\tau^{-1}$ contains the cycle $(\tau(a_0) \dots \tau(a_{k-1}))$. Further, if $\tau\rho\tau^{-1}$ acts on the remaining elements $\tau(x)$ where $x \in \mathbb{N}_N - \{a_0, \dots, a_{k-1}\}$. Then we have

$$\begin{aligned}\tau(a_0 \dots a_{k-1})\tau^{-1}(\tau(x)) &= \tau(a_0 \dots a_{k-1})(x) \\ &= \tau(x)\end{aligned}$$

Meaning $\tau\rho\tau^{-1}$ acts as the identity on $\tau(x)$. From this we can deduce that $\tau\rho\tau^{-1}$'s cycle decomposition consists of a single cycle of length k. □

This lemma leads us to the following theorem that will be of deep relevance for the remainder of the thesis.

Theorem 1.4. $\forall\ \alpha, \beta \in S_n$ *we have*

[3] The symmetric group on n elements.

α and β are conjugates $\iff$ α and β have the same cycle type [17, p. 126].

Proof.

$\Rightarrow$) Suppose $\exists\ \tau \in S_n$ s.t. $\alpha = \tau\beta\tau^{-1}$. β has some cycle type $1^{m_1} \dots n^{m_n}$. We can decompose β into disjoint cycles $\beta_{i,j}$ for $i \in \mathbb{N}_n$ and $j \in \mathbb{N}_{m_i}$. That is, there are m_i cycles $\beta_{i,j}$ of length i. We then have,

$$\begin{aligned}
\alpha &= \tau\beta\tau^{-1} \\
&= \tau\big(\beta_{1,1} \dots \beta_{1,m_1} \dots \beta_{n,1} \dots \beta_{n,m_n}\big)\tau^{-1} \\
&= \tau\beta_{1,1}(\tau^{-1}\tau) \dots (\tau^{-1}\tau)\beta_{1,m_1}(\tau^{-1}\tau) \dots (\tau^{-1}\tau)\beta_{n,1}(\tau^{-1}\tau) \dots (\tau^{-1}\tau)\beta_{n,m_n}\tau^{-1} \\
&= (\tau\beta_{1,1}\tau^{-1}) \dots (\tau\beta_{1,m_1}\tau^{-1}) \dots (\tau\beta_{n,1}\tau^{-1}) \dots (\tau\beta_{n,m_n}\tau^{-1})
\end{aligned}$$

$\forall\ i \in \mathbb{N}_n$ and $j \in \mathbb{N}_{m_i}$ we have that $\beta_{i,j}$ is a cycle of length i so by Lemma 1.3 we have that $\tau\beta_{i,j}\tau^{-1}$ is an i cycle.

Then to show α has cycle type $1^{m_1} \dots n^{m_n}$ we need only show that each $\tau\beta_{r,x}\tau^{-1}$ and $\tau\beta_{s,y}\tau^{-1}$ are disjoint $\forall\ r, s \in \mathbb{N}_n$ and $x \in \mathbb{N}_{m_r}$, $y \in \mathbb{N}_{m_s}$ with $(r, x) \neq (s, y)$. Suppose not, that is $\tau\beta_{r,x}\tau^{-1}$ and $\tau\beta_{s,y}\tau^{-1}$ act non-fixedly on the same element. We write $\beta_{r,x}$ as $(a_0 \dots a_{r-1})$ and $\beta_{s,y}$ as $(b_0 \dots b_{s-1})$. These two cycles are disjoint as they are independent cycles in the disjoint cycle decomposition. Then $\forall\ p \in \{0, \dots, r-1\}$ and $q \in \{0, \dots, s-1\}$ we have that $a_p \neq b_q$. We have from Lemma 1.3 that

$$\begin{aligned}
\tau\beta_{r,x}\tau^{-1} &= (\tau(a_0) \dots \tau(a_{r-1})) \\
\tau\beta_{r,y}\tau^{-1} &= (\tau(b_0) \dots \tau(b_{s-1}))
\end{aligned}$$

By supposition there must $\exists\ \tau(a_p) = \tau(b_q)$; However, we know $a_p \neq b_q$ and τ is an

injection resulting in a contradiction. Then α has cycle type $1^{m_1} \dots n^{m_n}$.

$\Leftarrow$) Suppose α and β both have cycle type $1^{m_1} \dots n^{m_n}$. Then we can write α and β as cycle decompositions into m_i cycles of length i as,

$$\alpha = \prod_{i=1}^{n} \prod_{j=1}^{m_i} \alpha_{i,j}$$
$$\beta = \prod_{i=1}^{n} \prod_{j=1}^{m_i} \beta_{i,j}$$

For $i \in \mathbb{N}_n$ and $j \in \mathbb{N}_{m_i}$, we can denote

$$\alpha_{i,j} = (a_0^{(i,j)} \dots a_{i-1}^{(i,j)})$$
$$\beta_{i,j} = (b_0^{(i,j)} \dots b_{i-1}^{(i,j)})$$

Then we define $\tau : \mathbb{N}_n \to \mathbb{N}_n$ by

$$\tau(b_k^{(i,j)}) = a_k^{(i,j)}$$

for any valid indices $i \in \mathbb{N}_n$, $j \in \mathbb{N}_{m_i}$, and $k \in \{0, \dots, i-1\}$. We claim that $\alpha = \tau\beta\tau^{-1}$. We consider $\tau\beta\tau^{-1}$'s action on some $a_k^{(i,j)} \in \mathbb{N}_n$. Note that $a_k^{(i,j)}$ lies in some cycle of length i and by construction $b_k^{(i,j)}$ also lies in a cycle of length i.

$$\begin{aligned}
\tau\beta\tau^{-1}(a_k^{(i,j)}) &= \tau\beta\tau^{-1}(\tau(b_k^{(i,j)})) \\
&= \tau\beta(b_k^{(i,j)}) \\
&= \tau(b_{(k+1) \bmod i}^{(i,j)}) \\
&= a_{(k+1) \bmod i}^{(i,j)} \\
&= \alpha(a_k^{(i,j)})
\end{aligned}$$

Since this is true for all $a_k^{(i,j)}$ and this covers all of $\mathbb{N}_n$, then we have $\alpha = \tau\beta\tau^{-1}$ and α and β are conjugates. □

Corollary 1.5. *Given α and β of the same cycle type $1^{m_1} \dots n^{m_n}$, there are*

$$\prod_{i=1}^{n} i^{m_i} m_i!$$

permutations which conjugate α to β.

Proof. We have seen via the above proof that permutations with conjugate α to β consist of mappings between the cycles of α and β when cycles of the same length are written one over the other. Of course, we can reorder the cycles of length i in $m_i!$ ways. Further, for each i cycle (of which there are m_i) we can shift each element up to i times to get the same permutation. This gives i^{m_i} possible shifts of all i cycles. □

This further gives us that,

Corollary 1.6. *There are*

$$\frac{n!}{\prod_{i=1}^{n} i^{m_i} m_i!}$$

permutations with the cycle type $1^{m_1} \dots n^{m_n}$.

Proof. Let α be any fixed permutation with cycle type $1^{m_1} \dots n^{m_n}$. Every permutation with this cycle type is a conjugate of α. So, the number of such permutations is equal to the number of distinct conjugates of α.

Every permutation $\sigma \in S_n$ gives a conjugate $\sigma\alpha\sigma^{-1}$. However, different σ can produce the same permutation after conjugation. For two permutations σ and ρ to produce

the same permutation after conjugating α we must have

$$\sigma\alpha\sigma^{-1} = \rho\alpha\rho^{-1}$$
$$\Rightarrow \alpha = \sigma^{-1}\rho\alpha\rho^{-1}\sigma$$

In other words, $\rho^{-1}\sigma$ must conjugate α to α

From the previous corollary, we know there are

$$\prod_{i=1}^{n} i^{m_i} m_i!$$

permutations conjugating α to α.

Then number of distinct conjugates of α — and therefore the number of permutations with the same cycle type — is

$$\frac{n!}{\prod_{i=1}^{n} i^{m_i} m_i!}.$$

□

Returning to the Enigma, recall that at any fixed position of the Enigma machine, the permutation describing this position π_i is just a conjugate of the reflector permutation R. As the reflector permutation R has a cycle type of 2^{13} it must then be the case by Theorem 1.4 that at any given position the Enigma machine is simply a 2^{13} cycle. This gives two key properties at any given fixed position:

(1) The Enigma is an involution, meaning that $\pi_i(\pi_i(x)) = x$. This is actually a desired and arguably necessary property for the Enigma to work since we need to ensure that, when two machines are at the same position, a cipher letter $\pi_i(x)$ will decrypt to its plaintext counterpart x.

(2) The Enigma has no fixed points. Since π_i is composed of 13 disjoint transpositions, we can never have a letter x for which $\pi_i(x) = x$. Therefore, we will never see a letter encrypted to itself. This means that for *any* setting, repeatedly pressing a letter (e.g. `A`) on an Enigma machine will *never* produce the same letter (e.g. `A`) on the output bulbs.

2

Bomba Kryptologiczna

In the early 1930s, Polish cryptographer and mathematician Marian Rejewski had arguably the most difficult task in the process of breaking the Enigma. Not only did he need to determine a means to recover daily keys from limited intelligence supplied by a German spy but he additionally needed to recover the wirings of the rotors themselves and reproduce the Enigma machine being used by the Germans.

Section 2.1

Intelligence

Hans-Thilo Schmidt (codenamed Asché) was a German spy who worked at their cryptographic headquarters and provided integral documents to the French regarding the Enigma machine. On November 8, 1931, Schmidt made an exchange with French spy Rodolphe Lemoine (codenamed Rex). For 10,000 German marks, Asché allowed Rex to photograph two documents [24, pp. 16–21] [33, pp. 144–146]:

(1) Gebrauchsanweisung für die Chiriermaschine Enigma – Instructions for using the Enigma machine.

(2) Schlüsselanleitung für die Chiriermaschine Enigma – A manual which details

the actual encryption procedure for writing messages. This procedure is the same as that described in Section 1.3.

These documents did not contain any explicit wirings of rotors but they did illustrate how various machine settings were employed in practice. The Poles had an agreement with the French regarding military cooperation and thus these documents made their way to the Polish Cipher Bureau.

The head of the Polish Cipher Bureau, Karol Gwido Langer, continued to receive information from the French via their informant Asché. In fact, through Asché, Langer had received key sheets which totaled 38 months worth of daily keys. With these keys, there was no necessity for using cryptographic techniques to break the encryption, as operators could simply use the keys to decrypt traffic. However, Langer knew that war was looming and there would soon come a time where such information would become unavailable. He made the bold decision to keep these key sheets from cryptographers at the Bureau, knowing that without them, cryptographers would be forced to work out a means of deriving the daily keys, not through intelligence, but through cryptanalysis [33, p. 157].

Section 2.2

Characteristics

We will see that purely with knowledge of this procedure and some military intelligence, Rejewski was able to determine the rotor wirings necessary to make further cryptanalysis possible.

Consider the first six letters transmitted according to our encryption procedure outlined in Section 1.3. Operator Alice has some three letter private key (say `XYZ`) which

she encodes twice with the machine settings specified by her key sheet. This will give us six encrypted letters $\pi_1(\mathtt{X})\pi_2(\mathtt{Y})\pi_3(\mathtt{Z})\pi_4(\mathtt{X})\pi_5(\mathtt{Y})\pi_6(\mathtt{Z})$. Suppose these six letters are given as

$$\mathtt{ABC\ DEF}$$

That is

$$\pi_1(\mathtt{X}) = \mathtt{A}$$
$$\pi_2(\mathtt{Y}) = \mathtt{B}$$
$$\pi_3(\mathtt{Z}) = \mathtt{C}$$
$$\pi_4(\mathtt{X}) = \mathtt{D}$$
$$\pi_5(\mathtt{Y}) = \mathtt{E}$$
$$\pi_6(\mathtt{Z}) = \mathtt{F}$$

Since each π_i is represented by 13 disjoint transpositions we can deduce, for example, that

$$\pi_1\pi_4(D) = \pi_1(X) = A.$$

With a sufficient set of hexagrams from gathered messages, we could then fully deduce the permutation $\pi_1\pi_4$. Further, we could recover $\pi_2\pi_5$ and $\pi_3\pi_6$. Rejewski referred to these paired permutations $\pi_i\pi_{i+3}$, as **characteristics** [30, p. 217]. In practice, such recovered characteristics may look like

$$\pi_1\pi_4 = (\mathtt{DVPFKXGZYO})(\mathtt{EIJMUNQLHT})(\mathtt{BC})(\mathtt{RW})(\mathtt{A})(\mathtt{S}) \tag{2.1}$$
$$\pi_2\pi_5 = (\mathtt{BLFQUEOUM})(\mathtt{HJPSWIZRN})(\mathtt{AXT})(\mathtt{CGY})(\mathtt{D})(\mathtt{K}) \tag{2.2}$$
$$\pi_3\pi_6 = (\mathtt{ABVIKTJGFCQNY})(\mathtt{DUZREHLXWPSMO}). \tag{2.3}$$

Rejewski noted a key structural similarity between all such characteristics recovered in this fashion: in each characteristic, cycles of the same length appear in pairs.

To see why this happens, consider the following lemma,

Lemma 2.1. *Suppose* $(\alpha\beta)$ *appears in* π_i *for* $i \in \{1, 2, 3\}$. *Then* α *and* β *will appear in disjoint cycles of* $\pi_i\pi_{i+3}$ *of equal length [29, p. 550].*

Proof. We begin by noting that if $(\alpha\beta)$ is in π_{i+3} then π contains fixed points at α and β and our claim is true. Then without loss of generality we can arrange π_i and π_{i+3} (non-exhaustively) in the following way:

π_i	π_{i+3}
$(\alpha\beta)$	(βx_1)
(x_1x_2)	(x_2x_3)
$\vdots$	$\vdots$
$(x_{k-1}x_k)$	$(x_k\alpha)$

Then the product $\pi_i\pi_{i+3}$ will contain the cycles

$$(\alpha x_1x_3 \dots x_{k-1})(x_kx_{k-2} \dots x_2\beta)$$

and thus α and β end up in disjoint cycles of equal length. □

This lemma has several consequences:

(1) A characteristic like $\pi_1\pi_4$, seen in 2.1, which has two singletons `A` and `S` must have that both π_1 and π_4 share the transposition (`AS`).

(2) A characteristic like $\pi_3\pi_6$, seen in 2.2, (two disjoint 13 cycles) reduces the space of possible π_3's to just 13 permutations which will take the form

$$(\texttt{AD})(\texttt{BO})(\texttt{VM})\ldots(\texttt{YU})$$
$$(\texttt{AU})(\texttt{BD})(\texttt{VO})\ldots(\texttt{YZ})$$
$$\vdots$$
$$(\texttt{AO})(\texttt{BM})(\texttt{VS})\ldots(\texttt{YD})$$

and similarly for π_6.

Thus with absolutely no knowledge of the rotor wirings or the daily key, we can already tractably compute a searchable space of π_i's. To determine which π_i is the correct one, we make use of the most prevalent bug in cryptography – operator error.

2.2.1. Cillies

Enigma operators were instructed to construct random trigrams for their message keys – likely to prevent frequency analysis attacks on the hexagrams beginning messages; However, operators often chose the same trigrams for each message. Some examples might include:

- Initials or first letters of the operator's spouse. For example, `CIL` perhaps deriving from the name "Cecelia" being shortened to "cillie". Allegedly, for this reason, poor selections of trigrams from operators became known as **cillies** [8, p. 143].
- The same letter encoded three times such as `JJJ`. These were ultimately prohibited in 1933 [24, p. 241].
- Letters forming trigrams on the Enigma keyboard such as `ASD` and `PYX`. These were also later prohibited [24, p. 241].

By keeping track of various radio stations which used cillies, Rejewski had a reasonable guess as to what the message key used for a particular ciphertext was. Suppose we

received three hexagrams originating from a radio station which often used AAA as their message key:

SUG SMF

SJM SPO

SYX SCW.

We can then compare these message keys against our possibilities for each π_i to determine if AAA could have, in fact, been used to encipher these hexagrams. For example, in the the first hexagram we see that G appears as the third letter. We suspect our radio operator is using AAA as their message key, so this would mean π_3 contains the transposition (AG). However, in $\pi_3\pi_6$, A and G lie in the same cycle which would contradict Lemma 2.1. Thus we know that the first hexagram does not encode the message key AAA. In order for a hexagram to encode AAA we require that each letter does *not* appear in the same cycle as A corresponding characteristic $\pi_i\pi_{i+3}$, but *does* appear in a cycle of equal length to the cycle containing A. With this set strict of requirements, we can deduce that the third hexagram was most likely an enciphering of the repeated letter J since the first two cannot encode this message key.

If our hypothesis is correct and SYX SCW is actually an enciphering of AAA AAA, we significantly reduce our space of possible π_is. For example, in our case, such a hypothesis would completely determine which π_3 was used. We know π_3 contains the transposition (AX) and only one such possible π_3 contains this transposition, specifically,

$$\pi_3 = (\mathtt{AX})(\mathtt{BL})(\mathtt{VH})(\mathtt{IE})(\mathtt{KR})(\mathtt{TZ})(\mathtt{JU})(\mathtt{GD})(\mathtt{FO})(\mathtt{FM})(\mathtt{CS})(\mathtt{QP})(\mathtt{NP})(\mathtt{YW}).$$

Similarly we can determine π_6. For the remaining π_is we are only left with a small, searchable set of options.

Trying our reduced set of possible π_is on hexagrams from other radio stations suspected of using cillies we can find the most likely π_is. For example, if we use our set of possible π_is and find that for a particular choice of π_i and a particular hexagram, we decode `PPP PPP`, we have strong reason to believe that this is the correct choice of π_is. In this way, we can recover each π_i and thus recover any transmission's message key – all without any knowledge of internal wirings, plugboard settings, or daily keys.

Section 2.3

Recovering the Rotor Wirings

Equipped with a means to determine each day's π_is, Rejewski set himself to finding the internal wirings of the rotors [30, pp. 219–221]. The full recovery of rotor wirings and turnovers is out of scope of this thesis; however, we will provide a brief description to illustrate that with minimal intelligence information, entire rotor wirings were able to be deduced.

To perform this deduction we will assume no turnover occurs in the first six letters which, as we have discussed, is a reasonably likely assumption. We begin by expanding π_i to

$$\pi_i = S^{-1}P^{-(x+i)}N^{-1}P^{x+i}M^{-1}L^{-1}RLMP^{-(x+i)}NP^{x+i}S$$

where x accounts for the initial starting position of the rightmost rotor. Since turnover does not occur, $M^{-1}L^{-1}RLM$ does not change between π_is. Thus, we will denote this permutation Q, simplifying our earlier expression to

$$\pi_i = S^{-1}P^{-(x+i)}N^{-1}P^{x+i}QP^{-(x+i)}NP^{x+i}S$$

Rejewski knew the plugboard settings for two whole months, so he began shifting knowns and unknowns to opposite sides. Shifting things around gives us

$$\begin{aligned} & \pi_i = S^{-1}P^{-(x+i)}N^{-1}P^{x+i}QP^{-(x+i)}NP^{x+i}S \\ \Rightarrow\ & S\pi_i S^{-1} = P^{-(x+i)}N^{-1}P^{x+i}QP^{-(x+i)}NP^{x+i} \\ \Rightarrow\ & P^{(x+i)}S\pi_i S^{-1}P^{-(x+i)} = N^{-1}P^{x+i}QP^{-(x+i)}N \end{aligned}$$

To further simplify notation we will then define $\rho_i \coloneqq P^{(x+i)}S\pi_i S^{-1}P^{-(x+i)}$. Then we have now have

$$\rho_i = N^{-1}P^{x+i}QP^{-(x+i)}N$$

where ρ_is are known from the deduction of π_i and the key sheets Rejewski had access to. We will now eliminate this equation's dependence on Q by considering pairs of ρ_i and ρ_{i+1}

$$\begin{aligned} \rho_i\rho_{i+1} &= N^{-1}P^{x+i}QP^{-(x+i)}NN^{-1}P^{x+i+1}QP^{-(x+i+1)}N \\ &= N^{-1}P^{x+i}QP^{-(x+i)}P^{x+i+1}QP^{-(x+i+1)}N \\ &= N^{-1}P^{x+i}QPQP^{-(x+i+1)}N \\ &= N^{-1}P^{x+i}(QPQP^{-1})P^{-x+i}N \end{aligned}$$

Each $\rho_i\rho_{i+1}$ shares the common sub-expression $QPQP^{-1}$. We can eliminate this sub-expression by noting

$$\begin{aligned}
\rho_{i+1}\rho_{i+2} &= N^{-1}P^{x+i+1}(QPQP^{-1})P^{-x+i+1}N \\
&= N^{-1}P^{x+i+1}(P^{-(x+i)}NN^{-1}P^{x+i})(QPQP^{-1})(P^{-(x+i)}NN^{-1}P^{x+i})P^{-x+i+1}N \\
&= N^{-1}P^{x+i+1}P^{-(x+i)}N(N^{-1}P^{x+i}QPQP^{-1}P^{-(x+i)}N)N^{-1}P^{x+i}P^{-x+i+1}N \\
&= N^{-1}P^{x+i+1}P^{-(x+i)}N(\rho_i\rho_{i+1})N^{-1}P^{x+i}P^{-x+i+1}N \\
&= N^{-1}P^{-1}N(\rho_i\rho_{i+1})N^{-1}PN
\end{aligned}$$

We now have a relationship between each $\rho_i\rho_{i+1}$ and $\rho_{i+1}\rho_{i+2}$ by conjugating by $N^{-1}PN$. Now recall from Theorem 1.5 that if $\rho_i\rho_{i+1}$ has a reasonably large cycle structure, then there are only a limited number of possible permutations for $N^{-1}PN$. The relationship between $\rho_{i+1}\rho_{i+2}$ and $\rho_{i+2}\rho_{i+3}$ will further reduce these possibilities since, of course, $N^{-1}PN$ must be the same between these two relationships. Eventually we will be left with sufficiently few choices of $N^{-1}PN$ that we can fully deduce this permutation.

Let $\kappa = N^{-1}PN$. We now want to deduce N. We have that $P = N\kappa N^{-1}$. By Theorem 1.5 there are only 26 possible permutations for N^{-1}, and thus for N, which take κ to P via conjugation. These 26 candidates for N correspond to the 26 possible orientations of a single fixed wiring – that is, each represents the same rotor wiring with the entry side rotated by one position. Regardless of which we choose, we recover, up to some rotation of the entry side of the rotor, the wiring of the rightmost rotor. With a known key, we can easily brute-force these possible orientations to determine the true rightmost rotor wiring.

In a similar fashion, we can recover the remaining rotor wirings though often with the help of other mathematical tricks, not to mention military intelligence and luck. From this point forward, we assume that the cryptanalyst now has access to the wirings of all the rotors as well as the reflector.

Section 2.4

The Grill Method

We now have deduced all of the rotor permutations N, M, L, R, as well as each π_i for a given day. We will use this information to recover the daily keys.

For the moment, let us assume S is the identity permutation. Then rearranging π_i we get

$$Q = P^{-(x+i)} N P^{x+i} \pi_i P^{-(x+i)} N^{-1} P^{x+i} \tag{2.4}$$

Since Q must be the same for each such equation involving π_i, we will devise a manual means to deduce the correct starting position by find a value of x which produces a consistent permutation Q across all $i \in \{1, \ldots, 6\}$. We can pre-compute N, $P^{-1}NP$, $\ldots$, $P^4 N P^{-4}$ and arrange them in a large sheet of rows called the **bottom sheet**. Then, for each π_i, we write it out on a separate line with a blank row (a slit) beneath it to allow for alignment with a row of the bottom sheet. This forms the **top sheet**.

We will illustrate this for π_1 and denote the slit beneath with periods.

$$\begin{array}{rl} & \texttt{|ABCDEFGHIJKLMNOPQRSTUVWXYZ|} \\ \pi_1 & \texttt{|SRWIVHNFDOLKYGJTXBAPZECQMU|} \\ & \texttt{|.........................|} \end{array}$$

By sliding the bottom sheet beneath the top sheet we can test various values of $P^{-k}NP^k$ to see what values of Q they give. For example, for $k = 0$ we have

$$\begin{array}{rl} & \texttt{|ABCDEFGHIJKLMNOPQRSTUVWXYZ|} \\ \pi_1 & \texttt{|SRWIVHNFDOLKYGJTXBAPZECQMU|} \\ P^{-k}NP^k & \texttt{|KJPZYDTIOHXCSGUBRNWFMVEQLA|} \end{array}$$

We know that $Q(\texttt{A}) = (P^{-k}NP^k)\pi_i(P^{-k}NP^k)^{-1}(\texttt{A})$. Therefore to compute $Q(\texttt{A})$ we first begin at `A` in our bottom row to find that $(P^{-k}NP^k)^{-1}(\texttt{A}) = \texttt{Z}$. We now map `Z` through π_i by finding `Z` on the top row and seeing where it lands in the middle row, thus giving $\pi_i(P^{-k}NP^k)^{-1}(\texttt{A}) = \texttt{U}$. Finally, we see where $(P^{-k}NP^k)$ maps `U` by finding `U` on the top row and seeing where it lands in the bottom row, thus giving $Q(\texttt{A}) = (P^{-k}NP^k)\pi_i(P^{-k}NP^k)^{-1}(\texttt{A}) = \texttt{M}$. Thus for π_1 and $k = 0$ we get that

$$Q = (\texttt{AM})(\texttt{BF})(\texttt{CX})(\texttt{DI})(\texttt{EP})(\texttt{GT})(\texttt{HU})(\texttt{JN})(\texttt{KW})(\texttt{LS})(\texttt{OZ})(\texttt{QR})(\texttt{VY})$$

Continuing in this fashion, we can get candidate Qs generated

$$(P^{-1}NP^1)\pi_2(P^{-2}NP^2)^{-1}, \ldots (P^{-5}NP^5)\pi_6(P^{-5}NP^5)^{-1}$$

2.4.1. Recovering the Rotor Position and Order

Recall that Q must be consistent between each equation 2.4. To check this, we can simply construct our top sheet such that all six permutations π_i are written in a vertical stack, each with a slit beneath it. We slide the bottom sheet behind them corresponding to a hypothesis starting position x. Aligning each π_i with a possible $P^{-(x+i)}NP^{x+i}$, we recover the resulting Q for each row. If the computed Qs are consistent across all six π_i, we have identified the correct value of x and hence the starting position of the rightmost rotor.

If S were truly the identity then we would ultimately find an offset of the bottom sheet that generates identical Qs for each π_i. Of course, S will not be the identity. Thus, instead of looking for perfect consistency between each Q we are only looking for relative consistency between each Q where perhaps a majority of letters are mapped identically. The offset where we find the most consistency between the recovered Qs will give us a likely candidate for the value of x and thus the absolute position of the rightmost rotor.

Further, we can deduce with reasonable certainty the value of the Q by just considering where the majority of the Qs map `A`, `B`, and so on. At this time, very few letters were steckered, so most letters will be mapped without plugboard involvement whatsoever. Thus, with six candidates for Q we are likely to find that, for a given letter, most Qs map this letter consistently. This mapping is likely the true mapping of Q with no plugboard involvement.

Letters where these Qs fail to line up should correspond to which letters are steckered. We can potentially even recover some of the steckerings themselves by seeing where

we expect Q to map a letter and comparing this to where it actually mapped a letter for a particular π_i.

Such a method was called the **grill method** [30, pp. 221–222] and required tedious work and had many possibilities for mistakes; however, if done correctly, it could recover the absolute position of the rightmost rotor, the permutation Q, and some steckerings of the plugboard.

To determine the absolute positions and ordering of the remaining two rotors we can simply enumerate all $26^2 = 676$ positions of the left two rotors, for each of the two possible orderings of these rotors, until Q is produced. In practice, a catalogue was eventually compiled which associated each Q to a corresponding position and ordering of the left two rotors. Now equipped with each absolute rotor position, we still must determine the ring settings.

2.4.2. Recovering the Ring Setting

To determine the ring settings, we make use of another operator error. Often messages began with the letters `ANX`[1] which is the German word "to" along with `X` denoting a space [30, p. 223]. We could therefore set the ring setting to a fixed setting `AAA` (or $(\texttt{01}, \texttt{01}, \texttt{01})$) and then brute force all 26^3 possible rotor positions until the first letters in the deciphered message were `ANX`. Once we knew the absolute position at which this occurred, along with the message key, we can immediately determine what the ring settings must have been to produce this message.

[1] This practice was prohibited past 1940 by forcing messages to begin with a random word [24, p. 244].

To see this, suppose we observe that at a window setting of (a, b, c)[2] we decipher our desired trigram ANX. Our fixed ring setting was $(01, 01, 01)$. Our goal is to find the ring setting which, when paired with the actual message key position of (x, y, z), still produces the desired trigram. In order to maintain our permutation that produces the observed trigram, we need to ensure that as we change the window setting to correspond with the message key we correctly change the ring setting so as not to affect the permutation. Recall that the permutation of a rotor will not change if the ring setting and window setting change identically. That is, without turnover, moving the ring setting forward by k steps and moving the window setting forward by k steps will not change our permutation. Since our goal is to change our window setting from (a, b, c) back to the actual message key (x, y, z), we must change our window setting back by a displacement of

$$((a - x) \bmod 26, (b - y) \bmod 26, (c - z) \bmod 26)$$

Thus to get the value of the true ring setting, we compute

$$(01 - (a - x) \bmod 26, 01 - (b - y) \bmod 26, 01 - (c - z) \bmod 26).$$

This technique of fixing a ring setting and then computing the true ring setting needed to maintain a desired permutation while matching a desired window setting is used repeatedly throughout this thesis. We have chosen to provide a full explanation each time it becomes relevant, as understanding this technique is critical to understanding the Bombe which is our ultimate goal.

[2]We will often denote rotor positions as 3-vectors whose first coordinate represents the leftmost rotors position from 1 to 26 and each subsequent coordinates represents the next rotor to the right.

2.4.3. Recovering the Plugboard

With the ring setting, rotor position, rotor order, and some of the steckerings of the plugboard, we can easily determine any remaining steckerings by deciphering messages and finding German words in which a letter is swapped. By looking for consistent patterns of letter swaps within common German words, the full plugboard configuration can be uncovered. With that, the cryptanalyst is now able to recover the full daily key.

One key point regarding the grill method is that at the time when it was in use only 6 jacks were used in the plugboard. This means that the Qs generated by each π_i were relatively similar since the permutation S had less impact on average. However, the Germans eventually used up to 10 jacks making the grill method infeasible, since we can no longer confidently determine the true value of Q [24, pp. 237, 242]. This fact will later necessitate the development of an additional tool called the **Bomba**, later discussed in this chapter.

Section 2.5

The Clock Method

At this point Rejewski and his team were able to recover daily keys, but the above methods are extremely slow and inefficient. Many optimizations were made over the years. We will now examine one particular optimization.

While we do know the wirings of the rotors, we do not know when we begin our cryptanalysis what the order of these rotors were. At the time there were only three rotors in use (`I`, `II`, `III`) so one could simply try all 3 rotors as the rightmost one and repeat the above analysis. Of course, this makes the above method 3 times slower. In

practice, early Enigma daily keys kept rotor positions the same for an entire 3 month period meaning this analysis did not need to be done too frequently.

However, Jerzy Różycki worked out an efficient method to determine the rightmost rotor which he called the **clock method** [30, p. 223]. The clock method attempted to determine where the turnover notch was for the rightmost rotor. From this we could immediately determine which of the three rotors was used.

2.5.1. Index of Coincidence

In a string of random text from a 26 letter alphabet we get a uniform distribution of letters. Thus we expect each letter to appear with a $\frac{1}{26}$ probability. When we align two pieces of random text, it then follows that we expect 1 letter to match up between the two pieces of texts every 26 letters. However, text which encodes a language does not generate a uniform distribution. Distributions for various languages have been well studied and are the information needed for frequency analysis. Having a non-uniform distribution also implies that when we align two pieces of text which encode a language, the chance that a letter from the first piece of text will align at the same position with a letter from the second piece of text is non-uniform. This distribution is called the **index of coincidence**. In particular, for German, we find that the expected number of alignments (called **coincidences**) between two pieces of text 26 letters long is 2.05 [20, p. 5-3][3]. If these texts are enciphered with the *same* poly-alphabetic cipher, we expect to see the same distribution of coincidences. If they are enciphered with *different* poly-alphabetic ciphers, we expect to return to a uniform distribution.

[3]It should be noted that this data is from 1952 whereas the index of coincidence that was being used for German at this time seems to be roughly $\frac{26}{17}$.

To make use of this property, we can select two messages with message keys (determined as in Section 2.2.1) whose first two letters coincide (e.g. `QKA` and `QKJ`). This means that both messages were encoded with the only difference being in their rightmost rotor. If at some point during encoding the first message, its rotors align with the rotors used to encode the second message, we would expect to see the number of coincidences between the messages to suddenly spike. In this way, we can detect when the two messages have their rotors align. This does not work, however, if turnover occurs. In that case, one of the middle rotors will step meaning that the messages will not align in their rotor positions and we will see a random distribution of coincidences. If we align the message generated by `QKJ` on top of the message generated by `QKA`, we can slide the bottom message until it is 17 positions ($10 - 1 \bmod 26$) further than the top message,

```
QKJ: MDIJVZOMRJYNXTHLXHMMWGYDQJQKPBEOQWZOAGSVDUXAWCDNQBVPFCLEOCOUYPBGFSHKQJDYUCWBCU

QKA:                    NAZDCRRCHGUJBWQCHOJHXORLPZDKQOHDXEXJSUEIRFRFSWODSPSFAYGOQWKCN
```

at the point where they line up, we would expect the texts to both be encoded by `QKB`[4] and the number of coincidences (indicated by _) should spike as we see in the above diagram. Given that this occurred, this means turnover did *not* occur between the rightmost window displaying `J` and `A`. This, for example, eliminates rotor `I` as a candidate since its turnover occurs when `Q` is displayed in the window. Similarly, we can eliminate rotor `III` since its turnover occurs when `V` is displayed in the window. This leaves only rotor `II` with its turnover occurring when `E` is displayed in the window. With just two messages we can determine a likely candidate for the rightmost rotor being used that day. With more such message pairs, we strengthen our certainty that a particular rotor is being used as the rightmost rotor.

[4] We might expect this to be `QKA` but recall that the rotors move first before encipherment.

The clock method was a precursor to a method we will discuss later known as Banburismus. The important takeaway is that, while language frequency analysis may not be strong enough to decode Enigma messages, it may be strong enough to determine elements of the key like rotor choice or ordering.

Section 2.6

The Cyclometer

These manual methods of decryption became increasingly difficult as German operators were instructed to use more plugboard jacks and increased the rate at which daily keys changed. Rejewski wanted to produce a mechanical means of performing a similar deduction. He returned to the characteristics associated with a particular day's key as in Section 2.2. He noticed that the cycle structure of a characteristic did not regularly repeat (if at all). Then we perhaps could create a "fingerprint" of a key by noting the cycle structure of all three characteristics associated to the key. Now that Rejewski had the internal rotor wirings, he could build a machine that could immediately produce the cycle structure of characteristics for a given setting. This machine was dubbed the **cyclometer** [30, pp. 224–225].

Let us first understand the manual implementation of such a deduction. Assuming an identity plugboard S, we can use our internal rotor wirings to know the exact permutation π_1 and π_4 for any initial rotor position. In specific, suppose we have

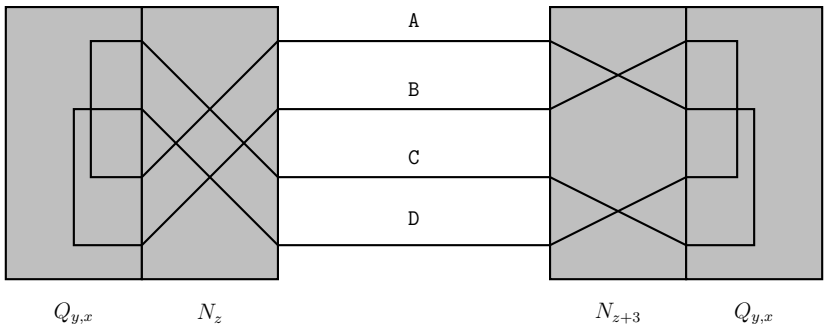
$$\begin{pmatrix} \texttt{A} & \texttt{B} & \texttt{C} & \texttt{D} & \texttt{E} & \texttt{F} & \texttt{G} & \texttt{H} & \texttt{I} & \texttt{J} & \texttt{K} & \texttt{L} & \texttt{M} & \texttt{N} & \texttt{O} & \texttt{P} & \texttt{Q} & \texttt{R} & \texttt{S} & \texttt{T} & \texttt{U} & \texttt{V} & \texttt{W} & \texttt{X} & \texttt{Y} & \texttt{Z} \\ \texttt{P} & \texttt{T} & \texttt{K} & \texttt{X} & \texttt{R} & \texttt{Z} & \texttt{Q} & \texttt{S} & \texttt{W} & \texttt{M} & \texttt{C} & \texttt{O} & \texttt{J} & \texttt{Y} & \texttt{L} & \texttt{A} & \texttt{G} & \texttt{E} & \texttt{H} & \texttt{B} & \texttt{V} & \texttt{U} & \texttt{I} & \texttt{D} & \texttt{N} & \texttt{F} \end{pmatrix}$$

π_1

$$\begin{pmatrix} \texttt{A} & \texttt{B} & \texttt{C} & \texttt{D} & \texttt{E} & \texttt{F} & \texttt{G} & \texttt{H} & \texttt{I} & \texttt{J} & \texttt{K} & \texttt{L} & \texttt{M} & \texttt{N} & \texttt{O} & \texttt{P} & \texttt{Q} & \texttt{R} & \texttt{S} & \texttt{T} & \texttt{U} & \texttt{V} & \texttt{W} & \texttt{X} & \texttt{Y} & \texttt{Z} \\ \texttt{J} & \texttt{W} & \texttt{V} & \texttt{R} & \texttt{O} & \texttt{S} & \texttt{U} & \texttt{Y} & \texttt{Z} & \texttt{A} & \texttt{T} & \texttt{Q} & \texttt{X} & \texttt{P} & \texttt{E} & \texttt{N} & \texttt{L} & \texttt{D} & \texttt{F} & \texttt{K} & \texttt{G} & \texttt{C} & \texttt{B} & \texttt{M} & \texttt{H} & \texttt{I} \end{pmatrix}$$

π_4

Then to find the cycle type of $\pi_1\pi_4$ we might do the following. We begin with A. We first run A through π_4 to get J. We then run J through π_1 to get M. Thus we see $\pi_1\pi_4(\texttt{A}) = \texttt{M}$. Now we could continue with M sending it back into π_4,. Eventually after jumping back and forth between π_1 and π_4 we we will have encountered all letters contained in the same cycle as A. We do, however, encounter some other letters as well. This method is effectively the same method used in lemma 2.1 to find the two disjoint cycles of equal length produced in a characteristic. In fact, by switching back and forth between π_1 and π_4 we will find alternating elements of the pair of cycles of equal length which contain A. We can mechanize this process via the following circuit[5]



In this circuit N_z represents the rightmost rotor at some position z and $Q_{y,x}$ represents our fictitious rotor (consisting of M, L, and R) at a fixed position y and x – we will denote this position (x, y, z). Note that right hand rotor in the permutation

[5]Throughout this thesis, diagrams will represent a simplified Enigma machine consisting of only four letters (A,B, C, and D):

depicted on the right is 3 steps further than the right hand rotor depicted on the left. When we now apply a current at `A` the current will travel back and forth through $N_z^{-1}Q_{y,x}N_z$ and $N_{z+3}^{-1}Q_{y,x}N_{z+3}$ and reach all the letters contained in the pair of cycles of equal length which contain `A`, that is, the number of wires electrified is double the length of the cycle containing `A`. If we connect each wire to a bulb, we can count the number of illuminated bulbs and quickly determine the length of the cycles in the permutation $\pi_1\pi_4$ at any given position of rotors (x, y, z). We can now rotate through all 26^3 positions of the rotors and create a catalogue of all characteristics $\pi_1\pi_3$ (note that $\pi_2\pi_5$ and $\pi_3\pi_6$ can be retrieved by just looking at the next two catalogue entries). We must also consider all 6 orderings of rotors `I`, `II`, and `III`. Thus we can laboriously generate a catalogue of all rotor orders and settings consisting of $6 \cdot 26^3 = 105456$ entries.

Now equipped with this catalogue, the cryptanalyst could use the traffic from a given day to determine the characteristics ($\pi_1\pi_3$, $\pi_2\pi_5$, and $\pi_3\pi_6$) as in Section 2.2, look up the cycle types in the catalogue, and immediately determine the rotor order and absolute rotor positions.

The plugboard settings could then be determined by comparing the characteristics recovered from that day's Enigma traffic, against the characteristics generated by the discovered rotor positions with no plugboard jacks inserted.

Finally, the ring position could be recovered as in Section 2.4.2 before by using known plaintext from the message.

This ultimately reduced the time to find a day's settings to roughly 15 minutes [30,

p. 225]. Note that the machine made finding cycles in permutations instantaneous by connecting each cycle in its own disjoint electrical circuit. This first attempt at mechanizing the process of decryption is a very early predecessor to the primary topic of this thesis, the Bombe. This method of enumerating cycles, as we will see, is the core of the Bombe's primary function.

Section 2.7

The Bomba

As mentioned prior, the addition of plugboard jacks made the grill method infeasible. Further, beginning on September 15, 1938, the German military changed the protocol by which Enigma messages were encoded, making both the grill method and cyclometer completely obsolete [31] [24, pp. 237, 244]. A new means of fingerprinting messages outside of characteristics needed to be developed. Further, compared to the grill method, such a new method would need to account for the fact that the plugboard swapped many more letters.

2.7.1. Changes to the Enigma Protocol

Starting in 1938, the German key sheets began removing an explicit window setting (*Grundstellung*) and added an indicator group known as the *Kenngruppen*. The indicator group was an extra identification added to messages to note which key was being used. It serves no cryptographic purpose so we will just discuss the way in which the message key was obtained.

Operators were now instructed to choose a random window setting (*Grundstellung*) for each message. The operator then sent this window setting in plaintext and proceeded to encode their message key (*Spruchschlüssel*) twice, enciphered with the

chosen window setting [30, p. 225–226].

On first view, this method seems to be more insecure than the previous method since we openly broadcast our window setting. However, because the window setting now changes between each message sequence, the cryptanalyst will never find enough messages with the same window setting to produce a set of characteristics needed for the cyclometer. Even with the window setting the cryptanalyst still knows nothing of the ring setting, plugboard setting, rotor position, or rotor order.

2.7.2. Females

Rejewski noticed that, given enough cipher material, we could find three messages with the following form seen below:

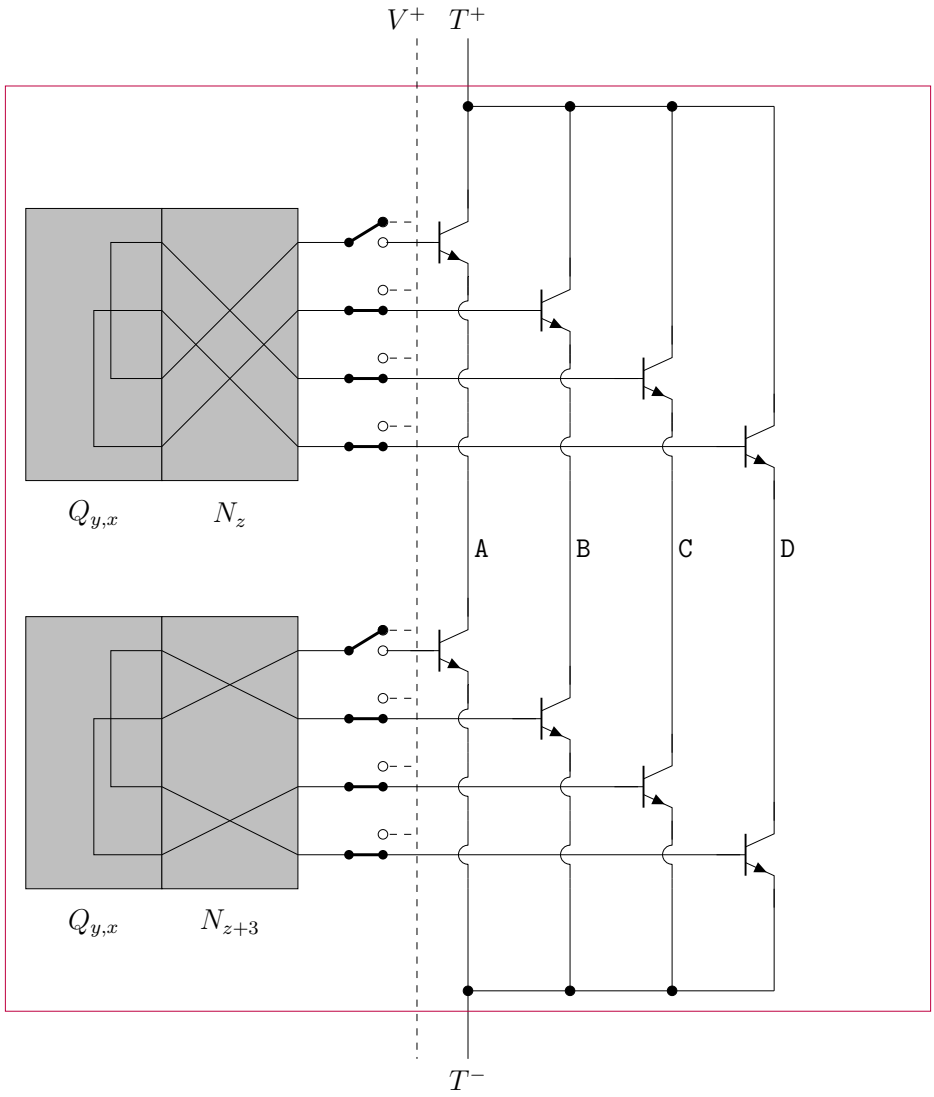
Grundstellung	*Spruchschlüssel*
RTJ	WAH WIK
HPN	RAW KTW
DQY	DWJ MWR

Notice that the first message has the letter W in both its first and fourth position. Cryptanalysts referred to such coincidences at a distance of three letters by the term **females** [24, p. 296] and to denote its location we could say this is a $1-4$ female. The second and third message also contain females, specifically $3-6$ and $2-5$ females respectively. In order for a daily key to be correct, it must produce females at the noted locations for the specified window setting. It turns out that such a set of females is a sufficiently unique "fingerprint" to reduce the number of possible rotor positions to a tractable number with which we can recover the daily key.

2.7.3. Bomba Kryptologiczna

We will first produce a mechanical means of detecting when three Enigma machines produce females at given locations – such a machine became known as the **Bomba Kryptologiczna** (Bomba for short). Let us begin by constructing a device that can detect a a single female, that is, it can determine for a given input letter when two sets of rotors three steps apart produce a female.

In this thesis, diagrams for the Cyclometer and the Bombe (later discussed) are abstracted so as to remove reference to specific circuitry elements since the reader is not expected to have a deep understanding of electrical engineering. However, compared to the Cyclometer and the Bombe, the Bomba requires more logical operations since it necessitates the equivalent of an AND gate. To maintain clarity for readers we will construct the Bomba piece by piece with relatively simplified circuit diagrams intended to give the reader an understanding of how such a machine could work in practice. Note that while we indicate transistors here for simplicity, the actual machine was an electromechanical computer which used relays and switches for these operations. Consider the following circuit:



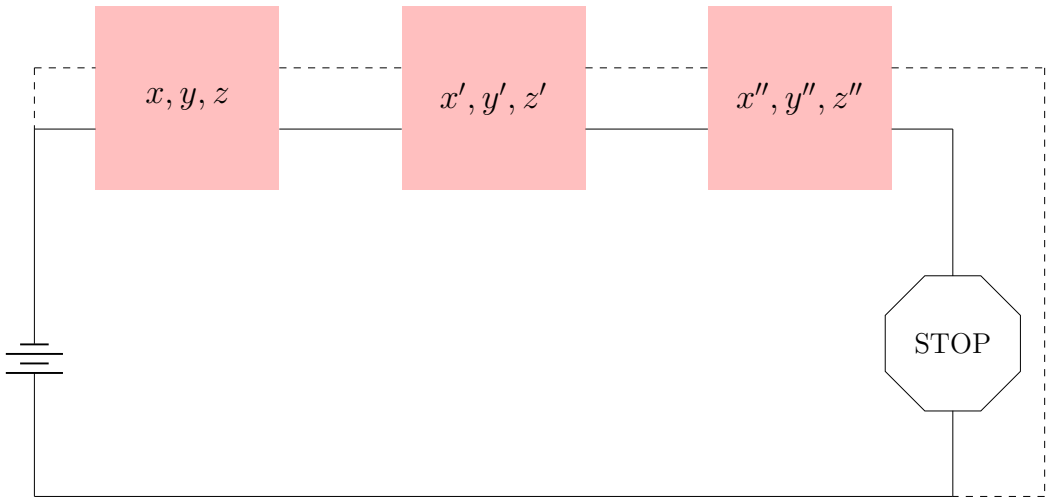
On the left we have our two sets of rotors represented by permutations $N_z^{-1}Q_{y,x}N_z$ and $N_{z+3}^{-1}Q_{y,x}N_{z+3}$ (three steps apart). Note that these permutations ignore the plugboard setting S, a caveat we will discuss later. Further, we will implicitly set the rotors' ring settings to `AAA` by just placing each letter above its corresponding contact on the rotor.

The dotted line represents our battery line. When current enters through the battery line, it is only able to enter at contacts which have their switches flipped, this is how

the operator can select the input letter. When the input letter then travels through the rotors, it will exit at some other location. If both $N_z^{-1}Q_{y,x}N_z$ and $N_{z+3}^{-1}Q_{y,x}N_{z+3}$ produce the same exit letter for the given input letter, then both exit letters will reach transistor gates along the same line (depicted with letters `A`, `B`, `C`, `D`). When this happens, current is freely able to flow from T^+ to T^-.

To see this, note that in our diagram we have `A` selected as the input letter. Following $N_z^{-1}Q_{y,x}N_z$ we can see that the rotor will now output `C`. Similarly, rotor $N_{z+3}^{-1}Q_{y,x}N_{z+3}$ will output `C`. Each of these will then travel to transistor gates of the line labeled `C`, thus opening both transistors in this line and allowing current to flow from T^+ to T^-. Thus, if we connect our battery in parallel to the line from T^+ to T^- and some current-triggered detector in series with the line from T^+ to T^-, the detector will only trigger when the input letter (in our case `A`) produces a female (in our case `C`) for the specified rotor position. We can abstract this entire circuit by noting that the wires exiting the enclosing box of the diagram are all that are needed to detect the presence of a female for the given rotor position and input. The box itself has settings consisting of the choice of rotors, rotor positions, and which switch we select for our input letter.

To allow this machine to detect three different pairs of females, we can simply align three female detectors in series, with three different rotor settings (x, y, z), (x', y', z'), and (x'', y'', z'') which indicate the start of that detector's particular female. We then wire the three detectors to a stopping mechanism (in the case of the Bomba this was an electromagnet), and the machine will stop whenever all three females are detected.



This is more or less the structure of the Bomba. Using our above construction we can select rotor positions corresponding to 3 females (thus representing 6 total sets of rotors) and we can select an input letter which we expect to produce these 3 females. The machine rotates all the rotors simultaneously so they maintain their relative distance from one another and as it churns through all 26^3 rotor positions. The machine will halt when it finds an absolute position corresponding to females in each set of rotors.

The origin of the name Bomba is disputed amongst records with the following claims to its namesake:

- A United States Army report explained that the original machines, which were turned by hand, would drop components on the floor when a solution was found producing a loud noise [1, p. 10].
- Colonel Lisicki of the Bureau explained that the name came from a popular ice-cream dessert of the same name which was consumed while the machine was first conceptualized [24, p. 63].
- Rejewski claims that the name was effectively random [30, p. 226].

From this machine we will now show that the daily key can be derived. We must first account for some caveats in our construction of the machine.

Turnover. In our construction we made the simplifying assumption that the rotor 3 steps apart from $N_z^{-1}Q_{y,x}N_z$ is $N_{z+3}^{-1}Q_{y,x}N_{z+3}$. Of course, if rotor turnover occurs between these two points, the middle rotor will have stepped making this assumption no longer true.

We found no explicit account of how turnover was handle in practice. It is possible that it was simply ignored since the likelihood of turnover in a 3 letter sequence of characters is $\frac{3}{26}$, meaning that there is a

$$(1 - \frac{3}{26})^3 \approx 0.69$$

chance that no turnover occurred in all three messages being examined. With a sufficient number of messages, simple trial and error will eventually yield legitimate results.

It is also possible that they explicitly handled turnover in their analysis. Earlier discovery of rotor wirings and ring notch locations meant that cryptanalysts knew, for a given rotor order, which window settings will produce turnovers in the 3 steps spanning the female being examined. Therefore, we definitively know whether or not a turnover occurred for a given message, so we can update the middle rotor as necessary to ensure the two rotors maintain a relative distance of 3 steps. This amounts to changing y to $y + 1$ for some rotor orders and messages – though for the sake of notational simplicity we will not account for this explicitly.

The Plugboard. Recall that our rotors are represented by permutations of the form $N_z^{-1}Q_{y,x}N_z$. Such permutations completely ignore the effect of the plugboard permutation S. However, consider a message with the keys

$$\texttt{RTJ} \quad \Big| \quad \underline{\texttt{W}}\texttt{AH } \underline{\texttt{W}}\texttt{IK}$$

For this message, denote the Enigma permutations at positions 1 and 4 as π_1 and π_4 respectively. Recall that W at positions 1 and 4 represent the same enciphered letter. Thus, we have

$$\pi_1(\texttt{W}) = \pi_4(\texttt{W}).$$

Consider the impact of the plugboard. Suppose W is fixed by the plugboard permutation S. We then have,

$$\begin{aligned}
&\pi_1(\texttt{W}) = \pi_4(\texttt{W}) \\
\iff &S^{-1}N_z^{-1}Q_{y,x}N_zS(\texttt{W}) = S^{-1}N_{z+3}x^{-1}Q_{y,x}N_{z+3}S(\texttt{W}) \\
\iff &S^{-1}N_z^{-1}Q_{y,x}N_z(\texttt{W}) = S^{-1}N_{z+3}x^{-1}Q_{y,x}N_{z+3}(\texttt{W}) \\
\iff &S^{-1}(N_z^{-1}Q_{y,x}N_z(\texttt{W})) = S^{-1}(N_{z+3}x^{-1}Q_{y,x}N_{z+3}(\texttt{W})).
\end{aligned}$$

Since S^{-1} is an injection, this occurs if and only if

$$N_z^{-1}Q_{y,x}N_z(\texttt{W}) = N_{z+3}^{-1}Q_{y,x}N_{z+3}(\texttt{W})$$

This means that for the purposes of testing for females on input W, our simplified rotor model works correctly so long as S fixes W. At the time when this machine was developed anywhere from $5-8$ plugboard jacks were being used, meaning that, on

average, half of the letters were fixed. Therefore, we have a reasonably high probability of our analysis being correct. In particular, around every other attempt should produce valid results.

This is also why our messages were chosen so that they all had the same letter `W` being repeated. We need only *one* letter fixed by S. We can then use this letter as the input for all our female detectors in the Bomba. Compared to the grill method, which relied on many letters being fixed by the plugboard, such a method was far more resilient to the addition of plugboard jacks.

2.7.4. Recovering the Ring Settings

In our Bomba we have 6 rotors, with 3 of them being paired such that they are 3 steps apart. Let us refer to these 6 rotors as,

$$R_{x,y,z} \text{ - } R_{x,y,z+3}$$
$$R_{x',y',z'} \text{ - } R_{x',y',z'+3}$$
$$R_{x'',y'',z''} \text{ - } R_{x'',y'',z''+3}$$

When our Bomba stops, we know the positions at which the females observed would occur, with a supposed ring setting of `AAA` (we will denote (`01`, `01`, `01`)). From this position, we will now attempt to recover the true ring setting. Suppose rotor $R_{x,y,z}$ stops at a position (a, b, c). We now know that with window setting (a, b, c) and ring setting $(\texttt{01}, \texttt{01}, \texttt{01})$ we have a set of permutations that generates the desired females. Of course, the actual window setting given in the message is (x, y, z). In order to maintain our permutations that produce the observed females we need to ensure that as we change the window setting to correspond with the known window setting, we correctly change the ring setting so as not to affect the permutation. Recall that the

permutation of a rotor will not change if the ring setting and window setting change identically, that is – without turnover, moving the ring setting forward by k steps and moving the window setting forward by k steps will not change our permutation. Since our goal is to change our window setting from (a, b, c) back to the actual window setting (x, y, z), we must change our window setting back by a displacement of

$$((a - x) \bmod 26, (b - y) \bmod 26, (c - z) \bmod 26).$$

Thus to get the value of the true ring setting, we compute

$$(\texttt{01} - (a - x) \bmod 26, \texttt{01} - (b - y) \bmod 26, \texttt{01} - (c - z) \bmod 26).$$

2.7.5. Recovering the Rotor Order

If we run our machine for all 6 possible rotor orderings, we now have candidate ring settings for each rotor ordering. Therefore, 6 Bomby (plural of Bomba) were constructed. We can now test which rotor ordering is correct by using our ring setting to attempt to decipher the doubly encoded message keys. If we were correct, then we should see the same trigram deciphered twice. Of course, because we have not yet set the plugboard we will not see the exact same trigram, but if we chose the correct rotor ordering we will still see many copies of letters at a distance of three from each other for any letter which was fixed by the plugboard. With enough deciphered message keys we can guess which rotor ordering was used that day. Given both this and the clock method described earlier we can recover the true rotor ordering.

2.7.6. Recovering the Plugboard

We now have the rotor order, the ring setting, and the window setting. We can then attempt to decrypt a message key. For example, we may decode the message key

from

LRJ | PAH IKO

and get

ANK LRK

We can assume K is correct since we expect it to be repeated at a distance of 3 letters apart, but the first two letters in each trigram are not identical indicating that they are being affected by the plugboard. We might guess therefore the the message should look like either something of the form

A.K A.K

or

L.K L.K

However, we have now deduced the entire permutation π_i, save for S. We can therefore derive which plugboard setting produces $\pi_1(\mathtt{P}) = \mathtt{L}$ or the alternate $\pi_4(\mathtt{I}) = \mathtt{A}$. This gives two plugboard possibilities. As we attempt to decrypt more message keys, we can find plugboard settings which mutually agree, and thus work out the entirety of the plugboard. With this, the cryptanalyst has recovered the entire daily key.

2.7.7. False Stops

It should be noted that a set of 3 females is not unique enough to pare down to a single setting. The Bomba could produce many stops, each of which produce females in all 3 pairs of rotors. These stops then need to be checked to see if the produce valid settings. This meant that finding keys with the Bomby could take nearly two hours [24, p. 242].

Section 2.8

Zygalski Sheets

2.8.1. Changes to the Enigma Protocol

While the Bomby made searching for possible ring settings more efficient, it still relied on the plugboard having a reasonable chance of having a fixed letter. On January 1, 1939, the Germans began to use 10 plugboard jacks [24, p. 242]. Thus, the probability that the letter being examined was fixed by the plugboard went down to $\frac{6}{26}$, meaning we would have to make use of the Bomby for $4-5$ different sets of females on average. Further, the Germans began to change the rotor order every month (eventually every day, and ultimately every 8 hours), meaning that we can no longer deduce one rotor order for the entire quarter [24, p. 242]. To make matters worse, the Germans introduced two new rotors (`IV` and `V`) meaning that the number of rotor orderings went from 6 to 60 [24, p. 243]. All in all, this meant that to use the Bomby would now require roughly 20 times as much work, making deciphering messages within the same day an unreasonable feat.

Around the same time as the Bomba was developed, Henryk Zygalski devised another means of using females in message indicators to recover the Enigma settings. Recall that we used the Cyclometer to generate a massive catalogue of each rotor position and ordering mapped to the corresponding cycle type of the characteristics generated by this setting. Zygalski intended to create a similar catalogue of positions and orderings, but now each entry related to the presence of a female for that setting. Consider a message with a $1-4$ female

BWY	FVJ FUE

Denote the Enigma permutation at the positions 1 and 4 respectively as π_1 and π_4. We know that F in the ciphertext at both position 1 and 4 represent the same letter (we will denote α). Then we have

$$\pi_1\pi_4(\mathtt{F}) = \pi_1(\alpha) = \mathtt{F}.$$

That is $\pi_1\pi_4$ contains the singleton cycle (F). The converse is also true, that containing a singleton cycle in $\pi_1\pi_4$ means that the letter contained in that singleton will produce a $1-4$ female.

We will denote π_i without the use of the plugboard as $\overline{\pi}_i$, that is

$$\pi_i = S^{-1}\overline{\pi}_i S$$

Then we note that

$$\begin{aligned}\pi_1\pi_4 &= S^{-1}\overline{\pi}_1 S S^{-1}\overline{\pi}_4 S \\ &= S^{-1}\overline{\pi}_1\overline{\pi}_4 S\end{aligned}$$

Then we have that $\pi_1\pi_4$ and $\overline{\pi}_1\overline{\pi}_4$ are conjugate permutations. Then by Theorem 1.4, these two must have the same cycle type. Thus, if either have a singleton cycle, they must both have a singleton cycle. All this is to ultimately say, if a particular Enigma setting produces a set of females (e.g. $1-4$), it will produce the same location of females (e.g. $1-4$) for any plugboard settings. Therefore, the possibility of producing a female is completely independent of steckerings.

With this in mind, Zygalski realized he could produce a catalogue of every rotor

position and ordering, along with whether or not their characteristic had a fixed point – that is, whether or not this setting could possibly generate a female.

2.8.2. Constructing Sheets

The catalogue was constructed as a series of sheets which are now called **Zygalski sheets**. Sheets produced in this thesis were constructed by code in our project's open-source repository [39]. To our knowledge this is the first open-source code available for creating and using Zygalski sheets.

To construct a sheet, we can set our Enigma to ring settings `AAA` and we can set our plugboard to the identity. We then choose a particular rotor ordering (e.g. `I III II`) and set our leftmost rotor's window position to a fixed letter (e.g. `B`). The sheet will be a 26 by 26 square of cells with row and columns labeled $\texttt{A} - \texttt{Z}$. For each cell location in row-column format (e.g. (`Y`, `W`)) we would set our machine to window setting `BWY` and examine if such a setting could produce a $1 - 4$ female. If a female is possible, we cut out a hole in the cell representing that setting. If a female is not possible, we leave that cell covered. We can denote such a sheet (`B - I III II`).

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

Figure 2.1: (B - I III II) Zygalski sheet: O represents a hole and . represents its absence

To maintain consistency with many digital recreations of Zygalski sheets, we use O to represent a hole (that is, a singleton cycle) and a . to indicate a covered cell.

Help from Bletchley. Originally as there were only 3 rotors, this meant they had to produce $6 \cdot 26 = 156$ such sheets. However, when the Germans added two new rotors, they now needed to produce ten times as many. Ultimately, the British at Bletchley Park ended up producing a full collection of these sheets and sent the collection to the Polish Cipher Bureau [30, p. 228].

2.8.3. Properties of Zygalski Sheets

The set of all sheets encodes, at every possible Enigma position, whether or not a $1-4$ female can be produced. Our ultimate goal is to recover the ring settings for a day's transmissions.

Suppose we receive the following indicators

`BWY`	`FVJ FUE`

We hypothesize that the rotor order is `I III II`, so we can get out the sheet (`B - I III II`) depicted in Figure 2.1 and examine row `Y` column `W` to see if window setting `BWY` with ring setting `AAA` can produce a $1-4$ female. At this cell, we will see it has no hole, indicating that such a setting cannot produce a female.

If we examine one cell to the right, we will be examining if the window setting `BXY` with ring setting `AAA` can produce a $1-4$ female. However, recall that, without turnover, this is the same as examining the window setting `BWY` with ring setting `AZA`.

By changing perspective, we can reinterpret this sheet. Rather than giving many possible window settings each with ring setting `AAA`, each cell can be read to be a reference to the window setting `BWY`, each with a different ring setting. In particular,

column c and row r will give us whether or not a window setting of BWY can produce a $1-4$ female with ring settings

$$(\mathtt{A}, \mathtt{A} - (c - \mathtt{W}), \mathtt{A} - (r - \mathtt{Y})).$$

Thus, such a sheet can investigate, for a given message key, all 676 possible ring settings with its leftmost letter being A. In our case, we can relabel Figure 2.1 as follows

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

Figure 2.2: Relabeled (B - I III II) representing window setting BWY with various ring settings starting with A

It is useful to note that location at which the ring setting ending in AA occurs. We will call this the sheets **origin** with respect to the indicator BWY.

Similarly, for the sheet one letter away from (B - I III II), that is, sheet (C - I III II), examining row Y column W will tell us, with window setting CWY and ring setting AAA, if we can produce a $1-4$ female. Of course, without turnover, this is the same as examining window setting BWY with ring setting ZAA. Thus moving to the sheet one letter forward is equivalent to examining our original indicator, but now with the ring setting's leftmost letter being Z. Thus, from the cell in sheet s, column c, and row r, we learn whether or not the window setting BWY with ring setting

$$(\mathtt{A}-(s-\mathtt{B}),\mathtt{A}-(c-\mathtt{W}),\mathtt{A}-(r-\mathtt{Y}))$$

can produce a $1-4$ female.

Other Females. While we may be able to find a sufficient number of messages with $1-4$ females to be able to make use of the sheets, our work would be much easier if we could include $2-5$ and $3-6$ females. Consider, for example, the indicators

PTC	PGA TGQ

Of course, this is exactly the same as a $1-4$ female with window settings PTD. In this sense, it suffices to only produce sheets for $1-4$ females.

2.8.4. Using Zygalski Sheets

In researching this thesis, no component was more troublesome than finding consistent and accurate information regarding Zygalski sheets. Many claims, often conflicting, were made by reliable sources. All the while, the closest description of how to actually use these sheets was given by Bletchley Park cryptanalyst, Gordon Welchman, in his book *The Hut Six Story* [40]. While Welchman does describe the underlying theory

behind the sheets, he does not explain explicitly how we test different configurations of the leftmost ring setting, nor how we deal with turnover. Further, I found no settings or means by which to reproduce the example sheets provided in the book, leading me to believe that they may be intended to give the reader an idea of what such a sheet would look like rather than representing an actual sheet. I suspect this is the case because I myself have done this for many examples provided in this thesis. Creating examples from authentic Enigma messages can require hundreds of messages to create a single example. The fact that some of these examples were fictitiously created does not change the underlying theory described.

However, in this section, I have attempted to explain a descriptive and clear procedure for usage of the sheets which is consistent with both Welchman and Rejewski's descriptions. In so doing, I will use sheets and indicators which were meticulously crafted from real Enigma settings to illustrate the true process of decryption. Some components which were not explicitly mentioned, I will attempt to fill in with educated conjecture. In the end, I was able to decipher example messages using the method described so, at a minimum, this is at least one way of using such sheets.

Consider the following set of indicators,

BWY	FNV FCF
AFQ	LOG LJU
LZX	IAP IYR
MHX	DLE DVE
RHT	WQH WAA
NSY	VCK VJR
QBW	OPA OVP
SBW	HRE HIK
VFS	SDL SAO
WFW	VEJ VDE
YKW	IGW IZZ

We begin with a large illuminated table. We now set up a frame with 26 by 26 letters indicating ring settings in the right two rotors from `A-Z`. This will serve as the window through which we will observe the sheets and ultimately, from this window, we will read the ring settings.

We will first test the assumption of the rotor order `I III II` and the leftmost ring setting `A`. We start with the first indicator. To get all possible ring settings beginning with `A` and window setting `BWY` we must get out the sheet (`B - I III I`). We place our sheet between the window and the table, and we now align the sheet such that its origin with respect to `BWY` matches the position (`A`, `A`) on the window. Thus, the window now reads consistently with the ring settings being tested for window setting `BWY`.

Note that if our Zygalski sheets were only 26 by 26, such a translation of the sheet would mean that there would be empty space in the window where no sheet was present as in Figure 2.3. Thus, to account for this, Zygalski sheets were actually 51 by 51, with cells repeating modulo 26 and labels on each axis from A-Z and then A-Y.

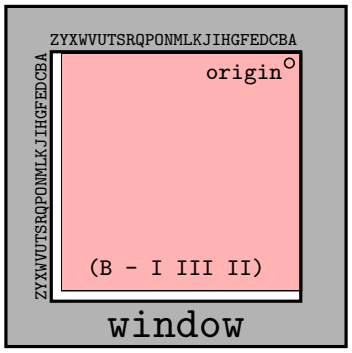


Figure 2.3: Sheet (B - I III II) shifted so that its origin is in location (A, A) in the window

We now consider our next indicator AFQ which corresponds to sheet (A - I III II). We place it in the window and translate it so that its origin aligns with (A, A). After doing this, the holes in both (B - I III II) and (A - I III II) will only line up at specific locations leading to a window with holes with light shining through as follows,

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

Figure 2.4: Window after lining up sheets for the first two indicators `BWY` and `AFQ`

Continuing in this fashion, placing the origin of each sheet corresponding to an indicator in the top right of our window, we are left with the following cells displayed,

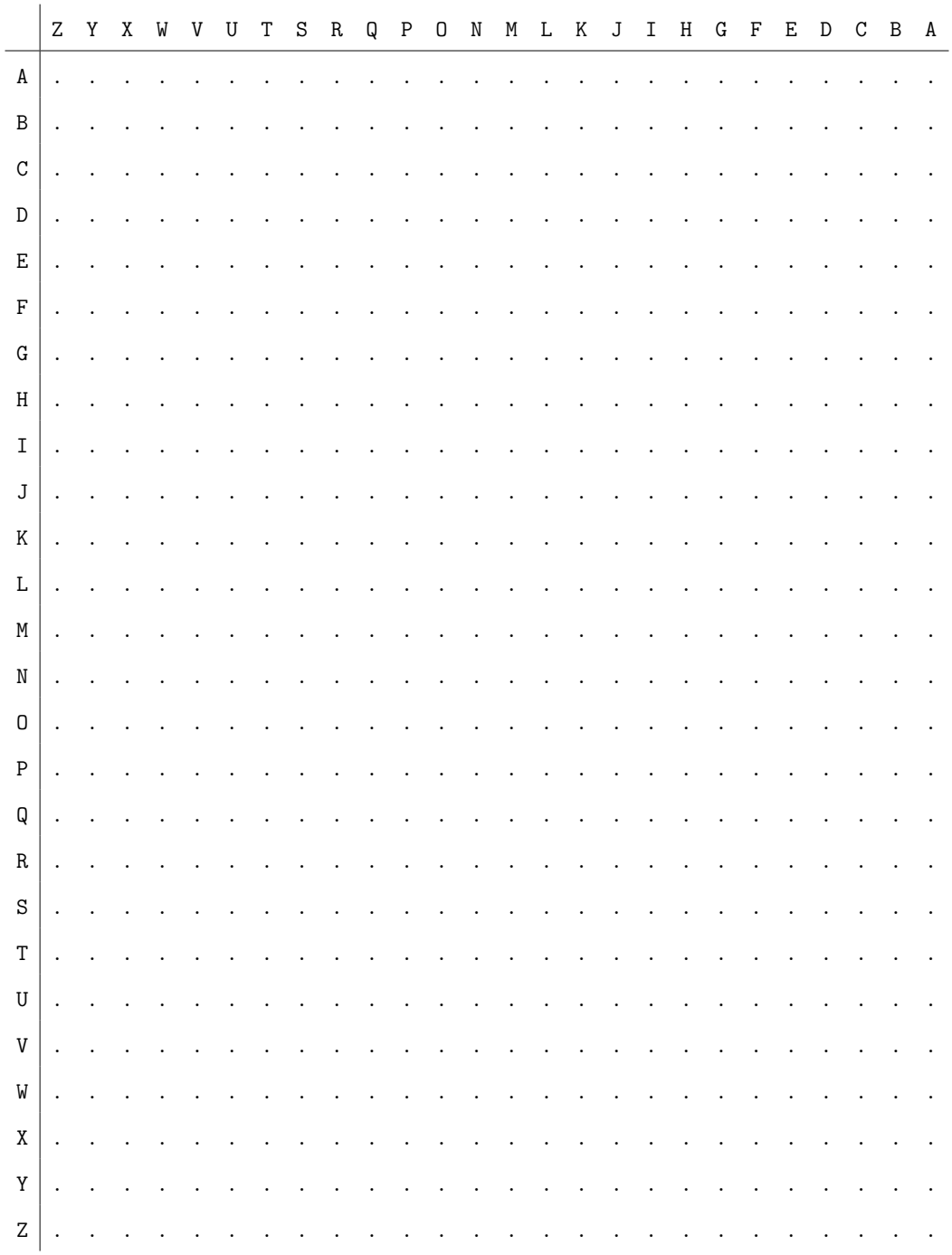


Figure 2.5: Window setting after overlaying sheets corresponding to all indicators

That is, all holes are blocked, and thus there is no position for which the rotors `I III II` and ring settings beginning with `A` can generate the observed females.

This only tested all ring settings beginning with `A`. To test the ring settings beginning with `B` we can move each sheet letter back by 1. That is, instead of sheet `(B - I III II)` use sheet `(A - I III II)`, and perform the exact same procedure. Iterating in this fashion through each possible letter in the leftmost ring setting, we will find some sheets which do have light peering through holes in all the sheets. Many of these settings, after testing, will be ruled out. However, eventually when we have moved the letter on each sheet back 3 times, which corresponds to the leftmost ring setting being `D`, we will see the following window,

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

This corresponds to the ring setting `DFX` which was, in fact, the rotor position that was used to encipher these indicators. This also means that our supposed rotor order is correct. If we iterate through every possible left ring setting and find no valid

settings, we must repeat this entire process for a new supposed rotor order. Though, by using the clock method, we can drastically decrease how many rotor orders we must try.

Recovering the Plugboard Setting. We can recover the plugboard settings exactly as we did for the Bomba described in Section 2.7.6. At this stage we will also uncover if we received a false setting from the window since we will find conflicting plugboard options.

Turnover. Much of this procedure was predicated on the fact that moving our window setting by some amount can be treated as moving the ring setting by an inverse amount. This is true, except when turnover occurs. This means that for some window settings of the right hand rotor we simply cannot interpret these cells as having the associated ring settings we described earlier. For a particular rotor order, this means there is a set of 4 adjacent rows on which we must not base our deductions.

To understand why there will be 4 such rows let us consider our sheet `(B - I III II)`. Rotor `II` will turnover whenever the window is at `E`. This means that if the righthand rotor begins at `B`, it will necessarily turnover while enciphering between the 1st and 4th position, meaning it will turnover while enciphering our females. This will similarly occur when the righthand rotor is at `C`, `D`, and `E`. This means that all four of these rows will not accurately represent the ring settings we label them with.

Programmatically this can be dealt with by ensuring that all 4 such rows have holes in every cell, thus ensuring that they do not accidentally rule out a possibly valid setting. How cryptographers actually dealt with this when using the physical sheets is unknown to me as no source referenced such an issue, save for Tony Sale in his

digitized implementation of the sheets, where he solved the problem as I did [36]. This is a question that warrants further historical research.

Expected Number of Sheets. We will now compute the expected number of sheets required to carry out the above operation. We first need to determine the probability that a particular Enigma setting can generate a female. Consider an Enigma permutation σ, our goal is to determine the probability that $\pi_1\pi_4$ has a singleton cycle. We will make the simplifying assumption that π_1 and π_4 are pulled independently from a uniform distribution over all 2^{13} cycles.

We will fix π_1 as containing 13 disjoint transpositions labeled τ_i, that is

$$\pi_1 = \tau_1 \dots \tau_{13}$$

$\pi_1\pi_4$ will contain a singleton cycle if and only if π_1 and π_4 contain at least one transposition in common. Then it suffices to determine the probability that π_4 contains at least one transposition from the τ_is.

We denote the event that τ_i is contained in π_4 as E_i. Then the probability that at least one τ_i is common between π_1 and π_4 is

$$\mathbb{P}\left(\bigcup_{i=1}^{13} E_i\right)$$

By inclusion-exclusion this gives us

$$\sum_{1\le i\le 13} \mathbb{P}(E_i) - \sum_{1\le i<j\le 13} \mathbb{P}(E_i \cap E_j) + \dots + \sum_{1\le i_1<\dots<i_{13}\le 13} \mathbb{P}(E_{i_1} \cap \dots \cap E_{i_{13}})$$

We will begin by considering $\mathbb{P}(E_i)$, that is, the probability that π_4 contains the transposition τ_i. Then to calculate this probability, we can imagine fixing τ_i and then we have free choice over the remaining 12 transpositions. This is the same as the number of 2^{12} cycles. We can then divide by the total number of π_4s which is just the number of 2^{13} cycles. This gives us

$$\mathbb{P}(E_i) = \frac{\#\text{ of } 2^{12} \text{ cycles}}{\#\text{ of } 2^{13} \text{ cycles}}$$

Using Lemma 1.6, we can compute

$$\begin{aligned}\mathbb{P}(E_i) &= \frac{\frac{24!}{2^{12}12!}}{\frac{26!}{2^{13}13!}} \\ &= \frac{2 \cdot (13)! \cdot (24)!}{(12)! \cdot (26!)}.\end{aligned}$$

We can write this as

$$\frac{2^1 \cdot (13)! \cdot (26 - 2(1))!}{(13-1)! \cdot (26!)}$$

which will be useful later when trying to determine a generic formula.

We now consider $\mathbb{P}(E_i \cap E_j)$, that is, the probability that π_4 contains both transpositions τ_i and τ_j. Similarly, we can imagine fixing τ_i and τ_j and then we have free choice over the remaining 11 transpositions. Following a similar argument as above

we derive

$$\begin{aligned}\mathbb{P}(E_i \cap E_j) &= \frac{\frac{22!}{2^{11}11!}}{\frac{26!}{2^{13}13!}} \\ &= \frac{2^2 \cdot (13)! \cdot (22)!}{(11)! \cdot (26!)} \\ &= \frac{2^2 \cdot (13)! \cdot (26 - 2(2))!}{(13-2)!(26!)}\end{aligned}$$

In general, we arrive at the formula for an arbitrary number of fixed transpositions $(\tau_{i_1}, \ldots, \tau_{i_k})$ with $k \leq 13$ as

$$\mathbb{P}(E_{i_1} \cap \cdots \cap E_{i_k}) = \frac{2^k \cdot (13)! \cdot (26-2k)!}{(13-k)! \cdot (26!)}$$

Then computing our original probability, we have

$$\begin{aligned}&\sum_{1 \leq i \leq 13} \mathbb{P}(E_i) - \sum_{1 \leq i < j \leq 13} \mathbb{P}(E_i \cap E_j) + \cdots + \sum_{1 \leq i_1 < \cdots < i_{13} \leq 13} \mathbb{P}(E_{i_1} \cap \cdots \cap E_{i_{13}}) \\ &= \sum_{i=k}^{13} (-1)^{k+1} \left(\sum_{1 \leq i_1 < \cdots < i_k \leq 13} \mathbb{P}(E_{i_1} \cap \cdots \cap E_{i_k}) \right) \\ &= \sum_{i=k}^{13} (-1)^{k+1} \left(\sum_{1 \leq i_1 < \cdots < i_k \leq 13} \frac{2^k \cdot (13)! \cdot (26-2k)!}{(13-k)! \cdot (26!)} \right) \\ &= \sum_{i=k}^{13} (-1)^{k+1} \binom{13}{k} \frac{2^k \cdot (13)! \cdot (26-2k)!}{(13-k)! \cdot (26!)} \\ &\approx 0.405\end{aligned}$$

Thus, any given setting has a roughly 40.5% chance of being able to have a female. Then the likelihood that n Enigma machines with randomized settings all produce a female is

$$(0.405)^n.$$

Then this is also the probability that, at a given location, there is a hole present in all n sheets. Given that there are 26^2 holes being examined, the expected number of holes peering through all n sheets is

$$(0.405)^n \cdot (26^2)$$

Ideally, we would like to reduce the possible settings down to just a single hole being examined, and thus we would need to find the number of sheets n such that

$$(0.405)^n \cdot (26^2) = 1$$

Solving this gives us that we need roughly $12-13$ sheets to produce a single surviving hole in our overlapped sheets. This means we need to find $12-13$ messages with indicators containing females.

3

Turing-Welchman Bombe

In the previous chapters, we have seen complicated manual cryptographic techniques give rise to a set of electromechanical tools meant to make such work tractable over short time periods. Changes to the Enigma protocol would soon necessitate the construction of one of the largest such electromechanical devices to date, known as the **Bombe**.

3.0.1. Changes to the Enigma Protocol

Recall that during the use of the Polish Bomba, Enigma operators were instructed to send their window setting (*Grundstellung*) in plaintext, and then use this to doubly encode their message key (*Spruchschlüssel*). Starting in 1940, a small but impactful change was made to the Enigma encryption protocol for the Army and Air Force – operators were instructed to send their encoded message key only once [31, pp. 331–332].

The Polish Bomba relied on relationships between the doubly encoded message key to function, without such relationships all the previously mentioned tools became useless. World War II had begun and the need for intelligence material was greater than ever. Thus, British intelligence brought a group of cryptanalysts, mathematicians,

historians, linguists, and more, to Bletchley Park to engage in a large-scale attempt to break the Enigma machine with this new protocol.

No name is more well known from this operation than that of Alan Turing, whose work went on to become the foundations of modern computer science. Turing needed to find a new means of attacking the Enigma encryption that could no longer rely on the doubly enciphered message key. Consider that the reliance on the doubly enciphered message key was really just a reliance on the knowledge that two letters encoded the same plaintext at specific intervals of distance. Thus, Turing began engineering a method of deducing daily keys via a known plaintext attack.

3.0.2. Herivel Tip

While Turing was developing his plaintext attack that ultimately culminated in the creation of the Bombe, there was a short period of time in which little to no Enigma messages could be deciphered [40, p. 102]. During this time, Bletchley Park cryptanalyst John Herivel had a simple idea. When operators set their ring settings as described by the key sheet they would place their wheels in the machine in the order described, and they would then turn the outer ring on each wheel so that the corresponding ring setting letter was at the top of the wheel. If the operator was particularly lazy, this was *all* they did, or if they were only slightly lazy, they may turn the wheels a couple turns forward or back. Regardless, this meant that many operators began enciphering their messages with a window setting that was nearly identical to the ring setting. Thus, by looking at several operators' first messages of the day, we would see a cluster around a particular setting that is likely to be the ring setting. Such an insight became known as the **Herivel tip** [8, p. 143].

With the Herivel tip, we could determine a likely ring setting. Then by either brute

force or use of the clock method, we could determine the rotor order. Then finally, as we did to recover the plugboard from the Bomba in Section 2.7.6, we could even recover the plugboard settings. Thus, purely by virtue of operator error, we could recover the entire daily key.

3.0.3. Parkerismus

Similar to the Herivel tip, Bletchely Park analyst Reg Parker found a shortcut to deduce some wheel orders, ring settings, and plugboard configurations. The presumption was that the German operators in charge of creating the key sheets needed to produce a large amount of random or pseudo-random numbers to create each daily key. As the number of unique keys increased, this task became cumbersome so, to reduce time, the operators creating the key sheets may reuse one or more columns of the key sheet from previously generated sheets. By keeping track of previously deduced daily keys, Parker found that, in fact, some columns from the daily sheets had been reused from previous months.

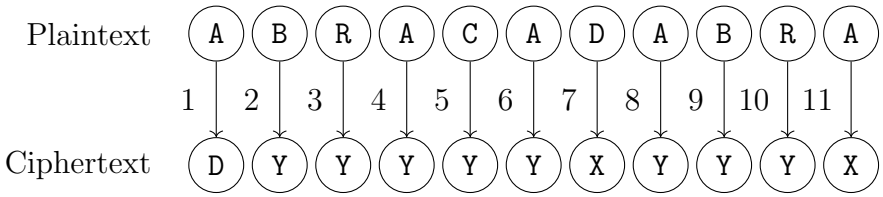
As parts of keys were broken, the relevant setting (e.g. wheel order, rotor order, plugboard configuration) were compared against Parker's records of prior keys to determine if that same setting appears in a prior sheet. If it was, Parker guessed that the entire column may actually be the same as the earlier month's and this would completely determine one component of the daily key for an entire month. As other components were deduced more columns could be determined in their entirety until, eventually, the entire month's key sheet could be determined in advance. This procedure became known as **Parkerismus** [40, pp. 130–131]. Welchman explained that at one point they had determined the entire month's worth of keys that Rommel would use in Africa just by use of Parkerismus [40, p. 131].

Ultimately, the Herivel tip and Parkerismus only helped them to decipher messages in particular circumstances. Herivel's trick was only used within a particular network of operators known as the "Red network" [40, p. 101]. Similarly, Parker's trick only functioned when certain columns in the key sheet had been reused. However, as we will later see, access to a set of plaintext from messages will become a vital component of our plaintext attack.

Section 3.1

Plaintext Attack

Suppose we knew the plaintext which had been enciphered into a particular Enigma transmission. Consider the following mapping,



where the top row indicates our plaintext message, the bottom row indicates the ciphertext, and the indices on the arrows indicate which step we are at while enciphering this message.

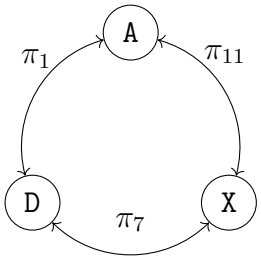
From this pairing we have that

$$\pi_1(\mathtt{A}) = \mathtt{D}$$

$$\pi_7(\mathtt{D}) = \mathtt{X}$$

$$\pi_{11}(\mathtt{X}) = \mathtt{A}$$

Thus, π_1, π_7, and π_{11} form a loop starting at $\mathtt{A}$. We visualize this loop as follows



with doubly linked arrows, since each π_i is an involution.

Recall that, ignoring turnover, each π_i is of the form

$$\pi_i = S^{-1}P^{-(x+i)}N^{-1}P^{x+i}M^{-1}L^{-1}RLMP^{-(x+i)}NP^{x+i}S$$

We will denote π_i separated from its plugboard as

$$\overline{\pi_i} = P^{-(x+i)}N^{-1}P^{x+i}M^{-1}L^{-1}RLMP^{-(x+i)}NP^{x+i}$$

that is $\pi_i = S^{-1}\overline{\pi_i}S$ (conversely, $\overline{\pi}_i = S^{-1}\pi_i S$)[1]. Then our loop is expressed by the fact that $\pi_{11}\pi_7\pi_1$ has a fixed point at A. We also note that the intermediate plugboard settings cancel out, that is,

$$\begin{aligned}\pi_1\pi_7\pi_{11} &= S^{-1}\overline{\pi_1}SS^{-1}\overline{\pi_7}SS^{-1}\overline{\pi_{11}}S \\ &= S^{-1}\overline{\pi_1\pi_7\pi_{11}}S.\end{aligned}$$

We will condense this notation by defining

$$\pi \coloneqq \pi_1\pi_7\pi_{11}$$

and

$$\overline{\pi} \coloneqq \overline{\pi_1\pi_7\pi_{11}}$$

[1]It should be noted that $S = S^{-1}$ since the plugboard is always an involution.

And thus we have shown $\pi = S^{-1}\overline{\pi}S$ (conversely, $\overline{\pi} = S^{-1}\pi S$).

Let us hypothesize that $\mathtt{A}$ is steckered in the plugboard to α – that is, $S(\mathtt{A}) = \alpha$ (conversely, $S(\alpha) = \mathtt{A}$). It then follows that for a fixed $i \in \mathbb{N}$

$$\begin{aligned}\overline{\pi}^i(\alpha) &= S\pi^i S(\alpha) \\ &= S\pi^i(\mathtt{A}) \\ &= S(\mathtt{A})\end{aligned}$$

and so we derive

$$S(\mathtt{A}) = \alpha \Rightarrow S(\mathtt{A}) = \overline{\pi}^i(\alpha) \ \forall \ i \in \mathbb{N}$$

Then we have that $\mathtt{A}$ must be steckered to all values in the set $\{\overline{\pi}^i(\alpha) \mid i \in \mathbb{N}\}$. We note that this set is that orbit of the element α under the group action of the subgroup $\langle\overline{\pi}\rangle$ – that is, $\langle\overline{\pi}\rangle \cdot \alpha$.

By construction of the Enigma machine, $\mathtt{A}$ cannot be steckered to more than one value at a time, so if $|\langle\overline{\pi}\rangle \cdot \alpha| > 1$ our initial hypotheses that $S(\mathtt{A}) = \alpha$ must have been incorrect. Further, the above argument also illustrates that $\mathtt{A}$ cannot be steckered to *any* element in the orbit of α since we would similarly find that the orbit of that element was not a singleton. Then we now have

$$|\langle\overline{\pi}\rangle \cdot \alpha| > 1 \Rightarrow \mathtt{A} \text{ cannot be steckered to any element in } \langle\overline{\pi}\rangle \cdot \alpha$$

thus eliminating several elements that $\mathtt{A}$ could be steckered to.

By representing $\overline{\pi}$ in its cycle notation, we can quickly see whether certain hypotheses are possible. For example, suppose we found that

$$\overline{\pi} = (\texttt{ABCDEF})(\texttt{GHIJK})(\texttt{L})(\texttt{MNOPQRSTUVWXYZ}).$$

If we suppose that A is steckered to any element in the cycle (ABCDEF) we find that this element has an orbit of length 6 in $\langle\overline{\pi}\rangle$ and thus A cannot be steckered to any element in this cycle. Then it is clear that A can only be steckered to L in this case.

3.1.1. Scanning Methods

Turing describes various methods of mechanizing the above analysis of cycle type to determine when we can eliminate rotor positions.

(a) If we examine a particular hypothesis, say A is steckered to K, we can rule out this steckering if we find that K is not in a 1-cycle, that is if $\overline{\pi}(\texttt{K}) \neq \texttt{K}$. If we mechanize this process we can eliminate rotor positions which do not satisfy this singular hypothesis. Turing called this method **single line scanning** [37, p. 104]. Note, however, that this method may eliminate rotor positions which do have valid steckerings, just not the particular steckering that we hypothesized.

(b) If we perform single line scanning in sequence, that is, for each steckering hypothesis we check if that hypothesis produces a fixed point in $\overline{\pi}$, we can rule out rotor positions which have all steckering hypotheses invalid. Turing called this method **serial scanning** [37, p. 104].

(c) Serial scanning requires a separate examination of each steckering hypothesis. Turing proposed a machine which could concurrently examine all steckering possibilities and eliminate rotor positions which had no valid steckerings. Turing called this method **simultaneous scanning** [37, p. 104].

(d) If we find $\overline{\pi}$ has a 26-cycle, then we must have that there are no 1-cycles and thus no valid steckerings. It then follows that the rotor position is incorrect. By constructing an electromechanical device which tests for this condition, we can

quickly eliminate *some* rotor positions which do not have valid steckerings. We will call this method **spider scanning**. Note, however, that this method would not, for example, detect that an $\overline{\pi}$ with cycle type 13^2 contains no valid steckerings. As Turing explained, "The ideal machine that Welchman was aiming at was to reject any position in which a certain fixed-for-the-time Stecker hypothesis led to any direct contradiction... The spider does more than this in one way and less in another. It is not restricted to dealing with one Stecker hypothesis at a time, and it does not find all direct contradictions" [37, p. 112]. Effectively, spider-scanning is like a form of simultaneous scanning which is restricted to examining only one cycle at a time.

Iterations of each scanning methods were proposed or designed, but in the end we find that the spider scanning method was used in the implementation of the Bombe.

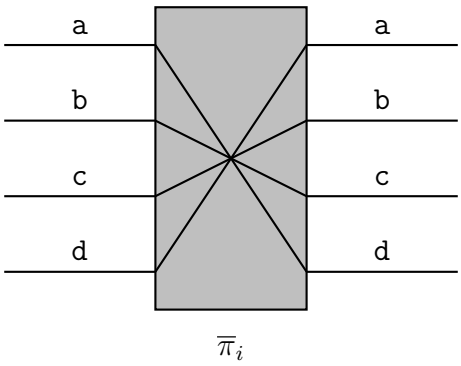
At a high level, we encode the structure of a plaintext-ciphertext pairing, and we input a hypothesis that a particular letter (e.g. `A`) is steckered to another (e.g. α). We then electrically produce the elements that must also be steckered to our test letter (e.g. $\langle\overline{\pi}\rangle \cdot \alpha$) if our hypothesis were to be true. If we find that this production generates a set of all letters, and thus disqualifies this rotor position from producing the plaintext we observed, we can then continue on to the next rotor position until we have no contradictory results from our hypothesis.

Section 3.2

The Bombe

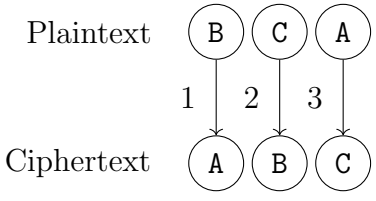
In the section, we outline the construction of a rudimentary Bombe. Our goal is to construct a machine using the above insight to quickly eliminate a rotor position based on contradictory hypotheses. We must shift from the above mathematical

construction to an electromechanical one. Imagine the following set of wires encoding a possible Enigma permutation we will denoted $\overline{\pi}_i$ (the bar indicates we are not yet considering the plugboard).

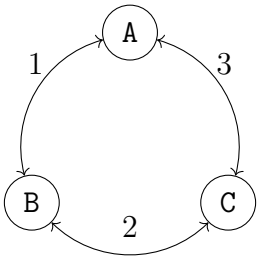


A couple quick notes about this abstraction. First, as these lines are simply wires, current can flow in either direction, left-to-right, or right-to-left. Second, we can apply current to multiple wires concurrently, for example, applying current at `a` and `c` will cause `d` and `b` to be live on the other side of the machine. Finally, the choice to use lower case letters will become clear further in this section as we want to separate the plugboard letters from the actual plaintext-ciphertext letters.

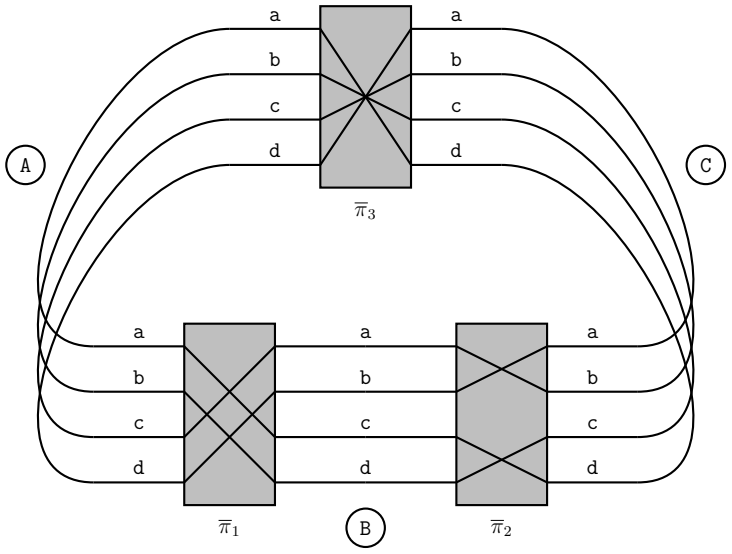
As described in our motivating example, we may want to connect Engima permutations in series to capture an underlying relationship between a plaintext-ciphertext pairing. Suppose we had the plaintext-ciphertext pairing,



Then we find that there exists a loop in the pairing as follows:



We can then think of this loop as a series of three Enigma permutations. At each step of enciphering the message, we have a new arrangement of the Enigma machine represented by the permutation π_i. Without the plugboard's involvement, these are denoted $\overline{\pi_i}$. For our plaintext-ciphertext pairing, we can imagine that the $\overline{\pi_i}$s for our loop of three Enigma machines may look as follows:



.

As before, let us make a hypothesis regarding steckering. Suppose $S(\mathtt{A}) = \mathtt{D}$. We know that A must map to A via $S^{-1}\overline{\pi}_3\overline{\pi}_2\overline{\pi}_1 S$. In our diagram, however, we have not accounted for the plugboard, so how can we represent the plugboard's involvement? In order to achieve this, *we* will function as the plugboard by applying voltage to the d line on the A cable, thus implicitly performing the plugboard mapping in which $S(\mathtt{A}) = \mathtt{D}$. Then this will go through the electrical mapping $\overline{\pi}_1$ arriving on the B cable, followed by $\overline{\pi}_2$ arriving on the C cable, followed by $\overline{\pi}_3$ arriving back on the A cable,

where *we* as the implicit plugboard know that the output of this electrical mapping must go through the plugboard again to get a final output of `A`.

We will denote each wire by a capital and lowercase letter which indicates both the cable and specific wire to which we refer. For instance, line `d` on cable `A` will be denoted `Ad`. Any time a wire `Xy` is live, implicitly this means that there is an intermediary plugboard which mapped $S(\texttt{X}) = \texttt{Y}$ or $S(\texttt{Y}) = \texttt{X}$. To see this electrical mapping occur let us visualize a current being sent through `Ad`.

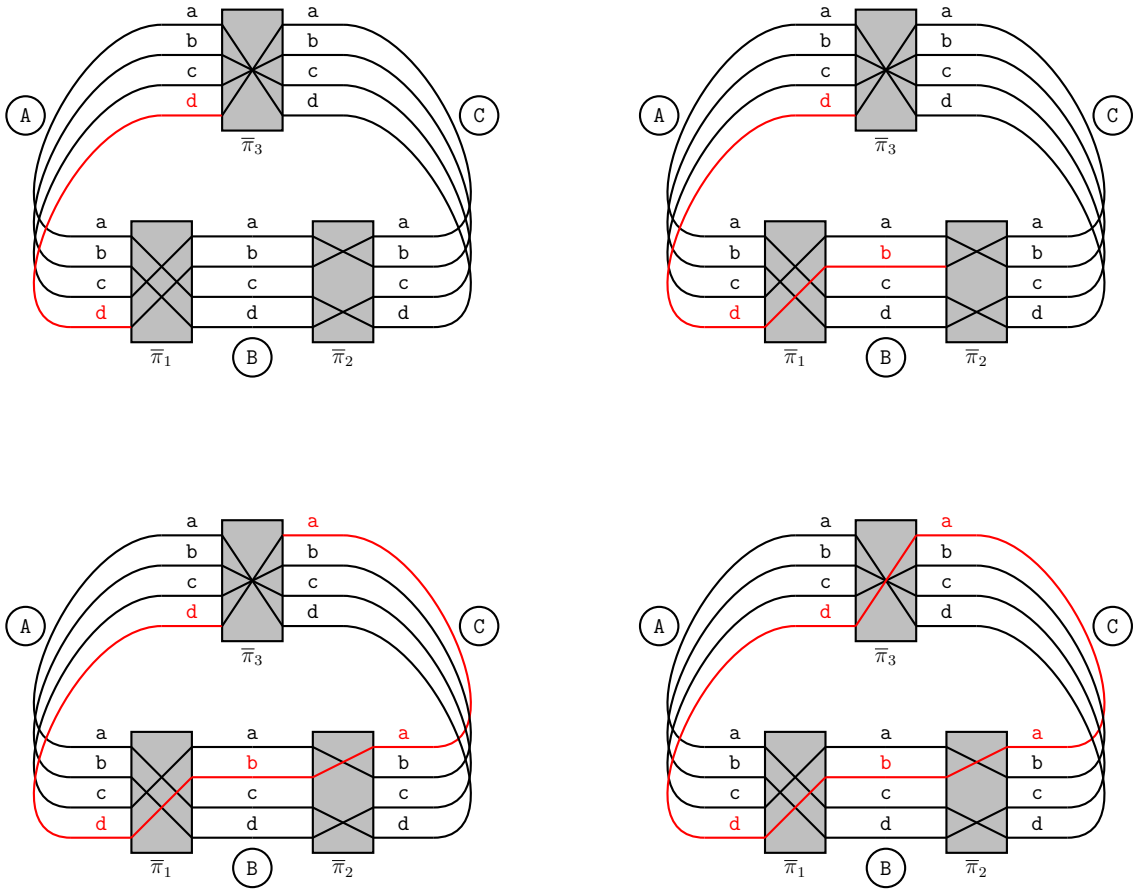


Figure 3.1: Current passing from wire `Ad` back to wire `Ad`

Following our hypothesis that $S(\texttt{A}) = \texttt{D}$ we find that after sending current through `Ad` we arrive after mapping through $\overline{\pi}$ back at `Ad`. Since we got an output of `Ad`, we know

that the plugboard needs to map D to A in order preserve our fixed point. Thus, we get the statement that $S(\mathtt{D}) = \mathtt{A}$ which is perfectly consistent with our original hypothesis.

On the other hand, if we change $\overline{\pi}$ and repeat this process which may end up in the following situation

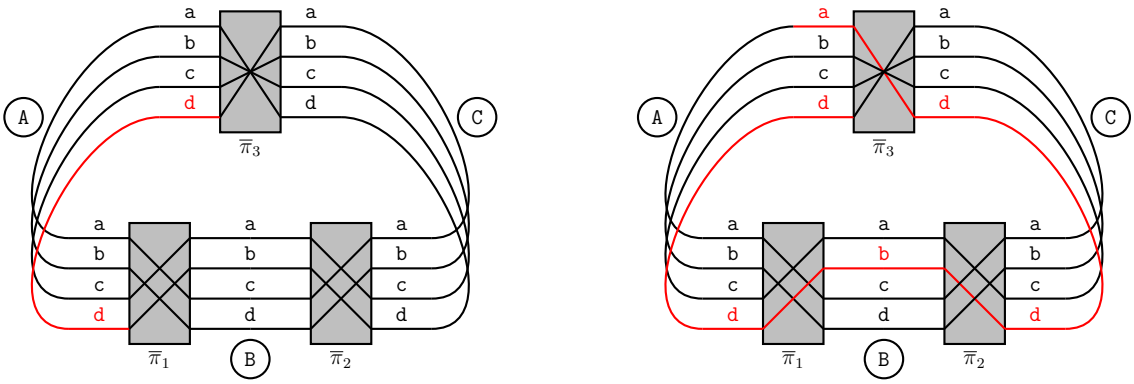
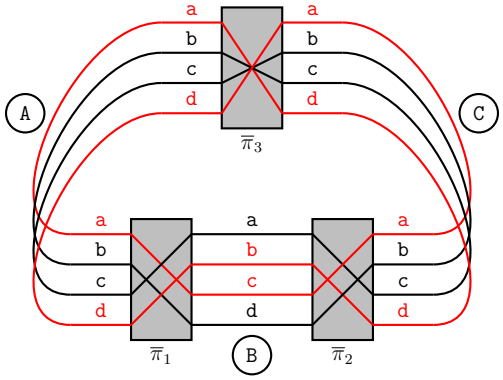


Figure 3.2: Current passing from wire Ad ending up at wire Aa

In this example, following our steckering hypothesis $S(\mathtt{A}) = \mathtt{D}$, results in Aa becoming live after mapping through $\overline{\pi}$ and thus we must have $S(\mathtt{A}) = \mathtt{A}$ as well, in order to preserve our fixed point at A. This creates an inconsistency between our hypothesis and deduction meaning that we can eliminate the both steckering possibilities $S(\mathtt{A}) \notin \{\mathtt{A}, \mathtt{D}\}$.

Effectively, this process is a mechanization of finding the cycle containing a particular letter. In the above Figure 3.2, sending current from Ad until we arrive back at Ad is akin to repeatedly applying $\overline{\pi}$ to D until we return to D. This can be visualized in our diagram as



Here we find that in $\overline{\pi}$ we have the cycle (AD) since Ad and Aa are the only two live wires in the A cable, after allowing currently to reach a steady-state. If we had instead applied current to Ab, we would find that Ab and Ac become live, thus giving us the full permutation $\overline{\pi} = (\mathtt{AD})(\mathtt{BC})$. The beauty of this design is that it is able to deduce the elements in a cycle of $\overline{\pi}$ nearly instantaneously since it only requires the circuit to reach a steady-state.

Equipped with this tool, spider-scanning becomes quite trivial. The goal of spider-scanning is to eliminate a rotor position by checking if the permutation has a 26-cycle. In our diagram, with four lines, this would be analogous to testing a steckering hypothesis (any hypothesis) and finding that the entire cable becomes live. Then all elements lie within the same cycle of $\overline{\pi}$ and thus all steckering possibilities are immediately eliminated.

The Bombe is designed to quickly move through all 26^3 possible window settings while maintaining the same relative distances between each Enigma permutation in our loop. At each configuration it applies current to test our steckering hypothesis and, only when an arrangement is such that it cannot be eliminated and further examination is needed, we will cause the machine to stop.

Scramblers. Thus far we have described the Bombe as connecting Enigma machines. In reality, the Bombe used a set of rotors known as **drums** mimicking the effect of each Enigma rotor. Thus three drums together produced the effect of an entire Enigma permutation. Such a set of drums was called a **Letchworth Enigma** (or **scrambler**) [37, p. 102] [40, p. 39] which effectively mimicked the function of the Enigma machine but was double-ended meaning current could be applied in either direction to achieve our Enigma permutation.

3.2.1. Stopping Mechanism

The Bombe would stop when it detected that any line (say `Ab`) is no longer live. To do this, we can place a differential relay such that it only engages when a current difference is present between `Ab` and some constant power supply line. Then, when current stops flowing through `Ab`, the relay will trigger. We can then wire the stopping mechanism to the contact terminal of the relay such that the stopping mechanism will only trigger when line `Ab` is not live (i.e. the relay is closed).

We can further extend this to detect when any line on the `A` cable is not live by having each line `Aa`, `Ab`, `Ac`, `Ad` wired in parallel, each to a separate relay, all comparing against the same constant supply voltage. If we wire our stopping mechanism to engage if any relay closes then our stopping mechanism will engage if any wire ceases to be live.

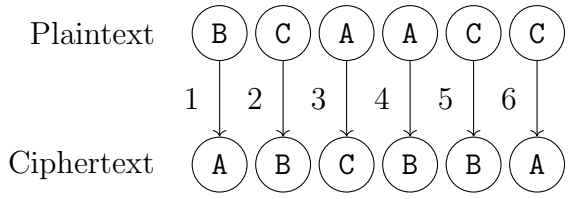
This raises the question of where to place this detecting circuit. If we are in the situation described above, a loop of Enigma permutations, then this choice does not matter. This is because if a 26-cycle is present in $\overline{\pi}_3\overline{\pi}_2\overline{\pi}_1$, it must also be present in $\overline{\pi}_2\overline{\pi}_1\overline{\pi}_3$ and $\overline{\pi}_1\overline{\pi}_2\overline{\pi}_1$. This is because all of these permutations are conjugates of one another, for example we have

$$\overline{\pi}_2\overline{\pi}_1\overline{\pi}_3 = \overline{\pi}_3^{-1}(\overline{\pi}_3\overline{\pi}_2\overline{\pi}_1)\overline{\pi}_3.$$

Since permutations in the same conjugacy class must have the same cycle type, it follows that no matter where in the loop we place our detector, if all wires become live on one cable they will become live on all cables. Thus, we can place our detector anywhere.

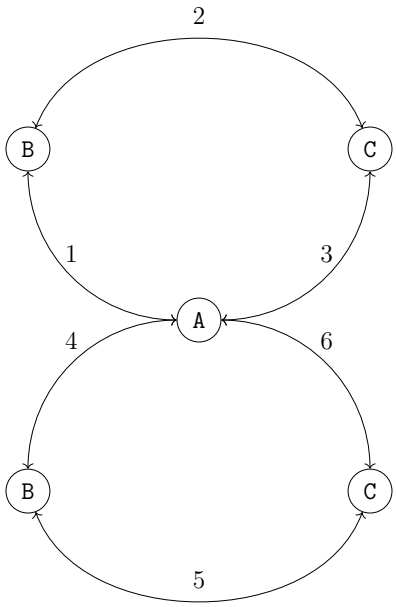
3.2.2. Multiple Loops

In our example we provided only a single loop encoded in the plaintext-ciphertext pairing. In practice, multiple loops could exist within such a pairing and we can electrically connect all loops together within a row on the Bombe to produce a single circuit. For instance, we might have a plaintext-ciphertext pairing as follows,



.

Thus encoding the following loops:



With our diagram format this could look as follows:

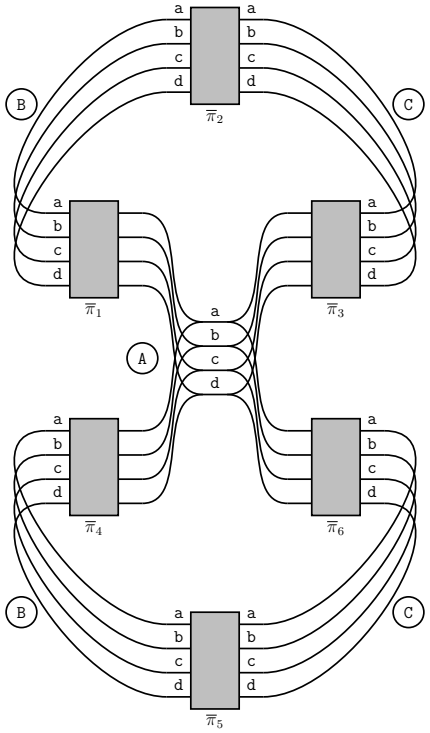


Figure 3.3: Multiple loops in the Bombe

.

This increased electrical connection results in fewer stops. In the next chapter, we will return to the effect that multiple loops have on the number of stops.

3.2.3. Indicator Drums and Ring Setting

The Bombe is set up with $k \leq 12$ scramblers

$$R_{x_1,y_1,z_1}, \ldots, R_{x_k,y_k,z_k}$$

intended to encode the various positions of our Enigma machine while enciphering our plaintext-ciphertext pairing. Each scrambler will have an implicit ring setting of ZZZ. We will assume that the window setting encoding the message began at ZZZ. The Bombe then runs through all 26^3 positions of the Enigma machine until it finds a stop.

Suppose that at a stop, the drums have positions

$$R_{a_1,b_1,c_1}, \ldots, R_{a_k,b_k,c_k}.$$

Recall that our ring setting is ZZZ (which we denote (26, 26, 26)). Then we know that a candidate with a ring setting of (26, 26, 26) and window setting of (a_1, b_1, c_1) should produce our expected plaintext at the location of the first letter included in our loop. Of course, we want the plaintext from the beginning of our plaintext-ciphertext pairing, so this should actually occur at a window setting of

$$(\mathtt{Z} + (a_1 - x_1) \bmod 26, \mathtt{Z} + (b_1 - y_1) \bmod 26, \mathtt{Z} + (c_1 - z_1) \bmod 26) \tag{3.1}$$

with a ring setting of (26, 26, 26). Recall that our assumed window setting at the beginning of our plaintext-ciphertext pairing is ZZZ. We want to move our window setting 3.1 to correspond with this, but in order to maintain the permutation which produced this stop, we must move the ring setting backwards by the same amount

we must move to get from 3.1 to ZZZ. This displacement is given by

$$((a_1 - x_1) \bmod 26, (b_1 - y_1) \bmod 26, (c_1 - z_1) \bmod 26)$$

since the Zs cancel out. Then we must move our assumed ring setting of (26, 26, 26) backwards by this same amount, thus giving a ring setting of

$$(26 - (a_1 - x_1) \bmod 26, 26 - (b_1 - y_1) \bmod 26, 26 - (c_1 - z_1) \bmod 26). \tag{3.2}$$

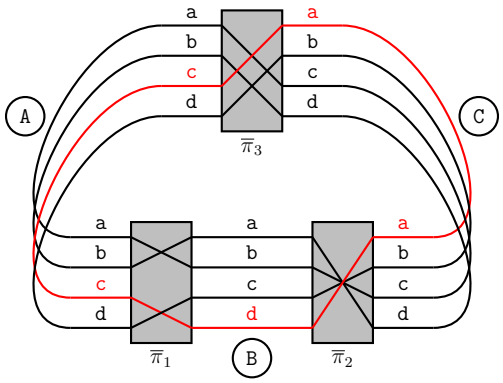
Thus with ring setting 3.2, and window setting ZZZ, we get the same permutation that caused the original stop, now properly adjusted so that it aligns with our assumed starting window setting.

Notice that $(a_1 - x_1, b_1 - y_1, c_1 - z_1)$ is just the number of steps our machine has made since it started. So, to get the ring position corresponding to the start of our plaintext-ciphertext pairing, we just subtract the number of steps each rotor has made from our assumed ring position (26, 26, 26). Rather than doing all this math, the Bombe was equipped with an additional set of **indicator drums** which began with labels ZZZ and for each step the machine moved *forward*, it moved the indicator drum *backward*. This meant that when the machine stopped, the indicator drums would immediately tell us the ring setting which corresponded to the start of our plaintext-ciphertext pairing. Then, with this ring setting, setting our machine to the window setting ZZZ would hopefully produce our expected plaintext-ciphertext pairing[2].

[2]The actual rotors used in the Bombe scramblers were incorrect and did not match the true Enigma rotors by varying degrees of rotation depending on the rotor in question. This meant operators did not set their window settings to ZZZ but rather some small offset depending on which rotors were included. This meant if we used rotors (II, V, III) our window setting would be YWY.

3.2.4. Recovering the Plugboard

Given a stop, we now know that for our hypothesized rotor order, a window setting of ZZZ, and ring setting given by the indicator drums, we should be able to produce our plaintext-ciphertext pair. However, we have yet to deduce the plugboard setting. The genius of the Bombe is that much of the deduction required to workout the plugboard setting has been done for us. Consider a possible stop of the Bombe



Our hypothesized plugboard setting was that A was steckered to C and we found no logical contradictions, thus producing a stop. This means that at a minimum we recover that $S(\texttt{A}) = \texttt{C}$. However, we may as well have hypothesis that B was steckered to D and thus we would have applied current to wire Bd. Note that this wire is included in our connected component so we would have recovered the exact same stop. That is, the statement $S(\texttt{B}) = \texttt{D}$ also produced no logical contradictions. Similarly, reading off cable C we would recover $S(\texttt{C}) = \texttt{A}$ though we already knew this from our first hypothesis. Thus, for each cable in our loop we recover a plugboard setting consistent with our initial hypothesis (though some of these may be redundant). For a sufficiently long loop we can recover several plugboard setting just by seeing which wires are electrified on which cables.

Checking Machines. While looking at the internal wiring will, in fact, derive such plugboard settings, from an operator's perspective the Bombe only displayed which

steckering our cable involved in our initial hypothesis produced. For this reason, a special device was produced called a **checking machine** whose sole purpose was to derive these plugboard settings from a stop of the Bombe. We will consider one such derivation.

The checking machine was effectively an Enigma machine with no plugboard attachment and no rotor advancement mechanism – effectively, the checking machine represents a fixed $\overline{\pi}_i$. Suppose the operator (known as a **wren**)[3] knew from the Bombe stop that our hypothesis letter A was steckered to C (i.e. $S(\mathtt{A}) = \mathtt{C}$). Further, from our loop, the wren knows that

$$\pi_1(\mathtt{A}) = \mathtt{B}$$

We then have that

$$\begin{aligned} S(\mathtt{B}) &= S(\pi_1(\mathtt{A})) \\ &= SS^{-1}\overline{\pi}_1 S(\mathtt{A}) \\ &= \overline{\pi}_1(\mathtt{C}) \end{aligned}$$

Then to determine what letter is steckered to B, the wren only needs to set their ring settings to that given by the indicator, the window settings to that corresponding to π_1, and input C. Now using this newly deduced steckering, the wren can produce the steckering of the next letter in the loop until we have deduced the steckerings corresponding to each letter in our loop.

Invalid Stops. If at this stage we find that a deduced plugboard setting contradicts an earlier plugboard setting we discovered, this means that the point at which the

[3]Wren comes from the abbreviation WRN of the Women's Royal Navy Service. Such members became the primary operators of Bombes at Bletchley Park [34, pp. 58, 97].

Bombe stopped is incorrect! Recall that the Bombe stops when there are no logical contradictions gleaned from its setup and the information in the loop – this does not mean no logical contradictions exist. Further, even if no logical contradictions existed that does not mean that this is the only such setting which could be logically consistent with our plaintext (though this is certainly less likely). The use of the Bombe is thus a trial-and-error procedure. We run the machine until we find a candidate stop, then we check for possible further logical contradictions, and if such contradictions are found, we continue from where we left off. Many additions we discuss later in this chapter are designed with the sole purpose of reducing the number of stops we need to check, down to a tractable number.

If we find our stop has valid plugboard settings, we can recover any remaining settings quite easily. We can attempt to decrypt our ciphertext with our window setting of ZZZ, our ring setting given by the indicator drums, and our plugboard settings we discovered from the stop. If we find that there are letters which do not match our expected plaintext we can deduce what plugboard settings need to be added to switch out incorrect letters with their correct plaintext letters. In this way, we can recover the entirety of the plugboard settings.

3.2.5. Recovering the Ring Settings

By this point we know that for our hypothesized rotor ordering, a window setting of ZZZ, a ring setting given by the indicator drums, and a plugboard setting deduced from the state of the stop, as well as additional logical induction, we can produce the desired plaintext-ciphertext pairing. However, this does not mean the ring setting that we have is the true ring setting. We assumed that turnover did not occur during our crib so in this case we can adjust our window setting and ring setting arbitrarily (so long as we keep their relative distances the same). Of course, eventually turnover

will occur, at which point the discrepancy between our ring setting and the true ring setting will become apparent.

We will first illustrate how to recover the true ring setting of the rightmost rotor. We begin by setting our Enigma as previously described and inputting the ciphertext. This should produce our expected plaintext. As we keep inputting ciphertext beyond our known plaintext we will recover additional plaintext in the message. Eventually, turnover will occur and we will begin deciphering gibberish since our ring setting is not the true ring setting. To account for this, we can move our ring setting forward by one step (which means we must also move our window setting forward by one to maintain our permutation) and we can perform the same procedure, investigating if we get gibberish. Eventually, we will have arrived at the true ring setting and our deciphering will begin producing additional plaintext.

To recover the ring setting of the other two rotors we employ the same procedure though it requires a great deal of ciphertext, as turnover becomes less and less frequent as we consider the middle and leftmost rotor respectively.

3.2.6. Recovering the Rotor Order

In its full form the Bombe was constructed with 3 rows each equipped with 12 scramblers. For a single plaintext-ciphertext pairing we could test multiple rotor orders simultaneously. If our plaintext-ciphertext pairing is shorter than 12 characters we could test 3 rotor orders concurrently, one on each row. If it is longer, then we may need to split it into two separate pairings spread across multiple rows, meaning that we may only test 1 or 2 rotor orders. Further, over the course of the war many Bombes were produced. This meant that it was feasible to test all 60 rotor orderings against our plaintext-ciphertext pairing concurrently to see which orderings generated valid

stops, thus allowing us to deduce the correct rotor order. In doing this, we have now managed to recover all of the relevant settings of the daily key.

Rotor Order Reuse. German Enigma protocol stipulated the following:

- No rotor setting could appear in two months' setting sheets [6]
- The same wheel could not be used in the same location two days in a row [6].
- In the same column, no wheel could be followed by a consecutive wheel (with `V` and `I` not regarded as consecutive) [6].

This meant that if Bletchley were able to regularly deduce Enigma settings, they could rule out many rotor orders which no longer needed to be tested in the next day or month.

3.2.7. Turnover

Recall that when we initially set up our Bombe, we set each scrambler to correspond to a location in our plaintext – the first location being `ZZZ`, the second being `ZZA`, and so on. As mentioned previously, the assumption that only the righthand rotor advanced in our window setting while enciphering the plaintext is only true if no turnover occurs. If turnover occurred during the message encipherment, our machine will not produce correct stops.

To account for this we will split up our message into multiple parts. Consider a plaintext-ciphertext pairing that is 26 characters long. We are guaranteed to have a turnover of the middle rotor during the encipherment of this message. However, we know that the turnover must have occurred in either the first half or the second half of the message. Thus, one half of the message assuredly had no turnover during its encipherment. This means that if we split our message into two 13 character

plaintext-ciphertext pairings, at least one of these will produce valid stops which will ultimately guide us to our daily key. In general, as long as our message is shorter than 26 characters we can split the message into components such that one component is guaranteed to not have turnover. If the message is longer than 26 characters there could be 2 or more turnovers during encipherment meaning that splitting the text in this way would no longer guarantee valid stops but, as we will see in our analysis in the next chapter, a plaintext-ciphertext pairing of length 26 is more than sufficient to produce a tractable number of stops, so this does not present an issue.

3.2.8. Cribs and Menus

We began with the supposition that the cryptographer is given a known plaintext-ciphertext pairing. Unless the cryptographer was able to intercept a set of communications before and after encryption this supposition seems unrealistic. However, cryptanalysts at Bletchley park employed two clever tricks that allowed for such pairings to be deduced with reasonable effectiveness.

Determining Plaintext. First, analysts were able to cross-reference known information to theorize likely phrases in a message. For instance, if the weather on a particular day was windy with a chance of rain, we would expect that the weather report (which intelligence knew arrived from specific channels at specific times) would likely contain phrases like "weather," "wind," or "rain."

Some sources of plaintext could be manufactured in a process known as **gardening** [32, p. 187] [22, p. 144]. Bletchley Park could request that the British Air Force "seed" a designated location with mines. It would then follow that many messages would make specific mention of that location or port which contained the mines.

Another source of plaintext lay in the message protocol. If a message was a continuation of another message, the message would begin with `FORT` (German for "continued") followed by the time at which the first message was sent, repeated twice. Numbers were ordered by the top row of the keyboard, that is, `Q` was 1, `W` was 2, and so on. Thus if a message was sent at 23 : 30, and a subsequent message was a continuation, it would begin with `FORTYWEEPYWEEPY`, where `Y` is used as a delimiting character [9, pp. 278–279].

Determining Cribs. The second trick used by analysts was the use of permutation theory to eliminate locations in the ciphertext where the plaintext could not feasibly have been enciphered.

Recall that one consequence of the Enigma machine being a conjugate of the reflector is that no letter can be enciphered to itself. On D-Day the weather report had the following ciphertext

`QFZWRWIVTYRESXBFOGKUHQBAISEZ`

Given that this was a report regarding the weather in the Bay of Biscay, we suspect it to contain the following plaintext

`WETTERVORHERSAGEBISKAYA`,

that is, "weather forecast Biscay" [18].

If we overlay our ciphertext on top of our plaintext as follows,

`QFZWRWIVTYRESXBFOGKUHQBAISEZ`

`WETTERVORHERSAGEBISKAYA.....`

we notice that at the 13th position, the letter S appears in both the plaintext and the ciphertext. We know that such an encipherment is impossible, and therefore, the suspected plaintext cannot begin at this location. If we slide the plaintext along our ciphertext we get the following,

```
QFZWRWIVTYRESXBFOGKUHQBAISEZ
WETTERVORHERSAGEBISKAYA.....
.WETTERVORHERSAGEBISKAYA....
..WETTERVORHERSAGEBISKAYA...
...WETTERVORHERSAGEBISKAYA...
....WETTERVORHERSAGEBISKAYA..
```

We can see that at offsets $0 - 3$ we have coincidences between the plaintext and the ciphertext that would make enciphering at this position impossible. Only at an offset of 4 do we observe no coincidences. Such a potentially valid location for enciphering is known as a **crib**. We may find many such cribs for a given transmission and suspected plaintext, but if our suspected plaintext is long, the likelihood of this occurring goes down significantly, so we expect to find only a few cribs.

Now equipped with our crib

```
RWIVTYRESXBFOGKUHQBAISE
WETTERVORHERSAGEBISKAYA
```

Figure 3.4: Example crib

We can generate a diagrammatic representation of our plaintext-ciphertext pairing

called a **menu** which will be used to wire up our Bombe. We assume the enciphering starts with window setting ZZZ so we will denote which window setting maps each letter to its plaintext counterpart [40, p. 306].

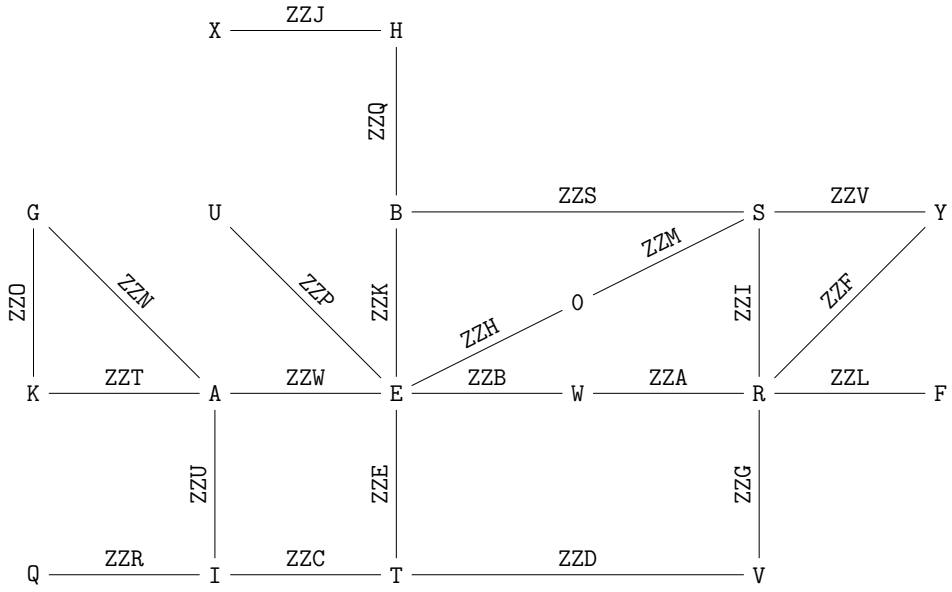


Figure 3.5: Example menu

Upon looking at this menu one might ask the question of why we include connections like the one between X and H if they are not included in any loop? From our earlier machine description its clear that such connection will never electrify additional wires. That is, they will only permute the incoming electrified wires but never increasing the quantity of electrified wires since there is no feedback. Thus, they add no additional value save for perhaps slightly more plugboard information. However, an addition to the machine made by Gordon Welchman made use of such connections and produced more electrified wires even within the loops themselves, thus reducing the total number of stops needing to be checked.

3.2.9. Diagonal Board

Recall that a wire `Xy` becoming electrified is consistent with the deduction that $S(\texttt{X}) = \texttt{Y}$. But this deduction is equivalent to the statement $S(\texttt{Y}) = \texttt{X}$. It follows then that if we apply current to `Yx` this will be entirely consistent with our current deductions and thus there will be no new logical contradictions introduced by this additional current. That is, for any wire `Xy` we can connect it directly to `Yx` through what was called a **diagonal board** [40, pp.301–305]. In our diagram, this can be viewed as follows:

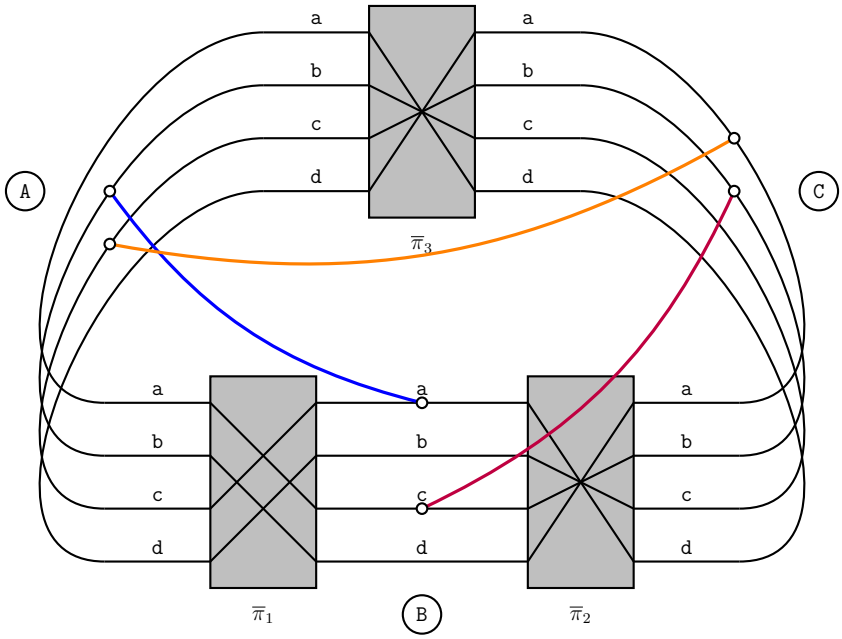


Figure 3.6: Diagonal wires

This significantly increases the number of connections, adding $\frac{k(k-1)}{2}$ new wires where k is the number of distinct letters in the menu. Adding more connections increases electrical connectivity thus reducing the number of stops. In fact, prior to the use of the diagonal board, often 3 loops needed to be present in the crib to have a tractable number of stops. With the addition of the diagonal board this could be significantly reduced and made the use of the Bombe practical [14].

For instance, in our example from Figure 3.6, without the diagonal such a state would produce a stop with hypothesis $S(\texttt{A}) = \texttt{D}$. However, adding the diagonal wires would cause all wires to have current flowing through them – meaning that our original stop must have been a false one, only revealed by the addition of new, consistent deductions which ruled out all possible plugboard settings.

This also meant that connections in our menu not included in a loop, such as the connection in 3.5 between X and H, could now produce feedback within our loops through the diagonal board connections. This made such connections valuable additions to a menu. This addition was so valuable in making the Bombe a functional means of decrypting Enigma traffic that the Bombe is often called the Turing-Welchman Bombe, as it was their combined contributions and insight that allowed the machine to so effectively produce daily keys.

While the name often includes only Turing and Welchman, the man who turned the mathematical theory of the Bombe into actual machinery was chief engineer of the British Tabulating Machine Company, Harold "Doc" Keen [40, p. 81].

3.2.10. Consecutive Stecker Knock Out

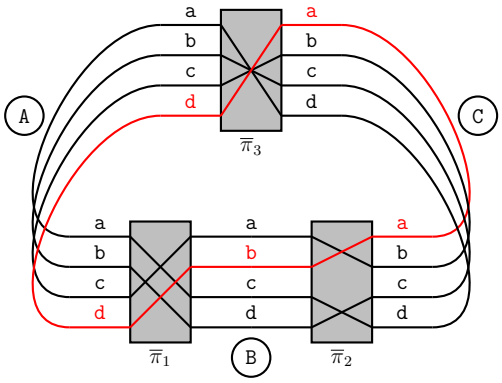
In the Air Force Enigma protocol, the plugboard setting was always such that no letter was connected to its neighboring letter in the alphabet (e.g. A cannot be connected to B). Some Bombes had a setting called **Consecutive Stecker Knock Out (CSKO)** that allowed for the elimination of some stops which implied consecutive steckerings. This was accomplished by wiring wire Ab to Bc to Dc and so on, with an optional jack that could complete the circuit on the back of the Bombe [8, p. 240].

In general, the CSKO circuit not directly eliminate consecutive steckerings. However, consider the case of a stop (which we will later see is quite common) in which every wire save for one is live. This implies that $\overline{\pi}$ has a cycle structure of $1^1 25^1$ and that this remaining singleton cycle is the only possible plugboard setting. If this dead wire representing the singleton does not imply any consecutive steckerings then the CSKO circuit will not come into play and the stop would be considered legitimate. If, however, a consecutive steckering like `Ef` is on the dead wire, we can see how the CSKO circuit will eliminate this stop.

We know that `Fe` is the dead wire on cable `F`, since `Ef` and `Fe` are connected via the diagonal board. Then any other wire on cable `F` must be live. In particular, cable `Fg` must be live. We know that the CSKO circuit connects `eF` to `Fg` so it follows that `eF` will now become live through the CSKO circuit. This will make live the only remaining wire, and so all wires become live. Thus, cases in which the machine would singularly imply consecutive steckering hypotheses would be eliminated. This reduces the number of stops to be checked for Air Force traffic.

3.2.11. The Machine Gun

Even with these additions, we still have more deductions that can be made to rule out certain settings. Consider our first stop we examined from Figure 3.1,



This stop may seem to imply we found no logical inconsistencies in our deductions

stemming from the hypothesis that $S(\texttt{A}) = \texttt{D}$. However, when a wren goes to check the settings found, they will discover that this setting produces a contradictory plugboard setting of $S(\texttt{A}) = \texttt{C}$ – thus ruling out this stop. Is there a means by which we could have detected this without having to manually work it out on a checking machine?

Recall that before we introduced the checking machine, we noted that the plugboard settings we would deduce from the checking machine were implicitly encoded in the internal state of the wires of the machine. In particular since wire `Ca` was active, we could have immediately deduced that $S(\texttt{A}) = \texttt{C}$ without having to employ a checking machine. However, these wires states are not easy for a human to read and would require complex manual analysis of the underlying cables. Instead, a device called the **machine gun** was developed which examined the state of the diagonal board at a given stop to determine if logical contradictions were present among the deduced plugboard settings. When a stop occurred, the machine gun would be brought in. The device then used a series of uniselectors to quickly switch through the diagonal board contacts to search for a pair of deduced steckerings that had a common letter (thus producing a logical contradiction). The machine got its name from the sudden loud burst of sound produced when the uniselectors ran through the diagonal board. The class of Bombes produced with the machine gun were dubbed **Jumbo Bombes** [8, p. 240].

Even with such a device many stops still needed to be checked by the wrens since the machine gun only detected logical inconsistencies in regards to plugboard setting overlap and thus still produced false stops.

Section 3.3

Banburismus

Early in this chapter, we discussed the changes made to the Enigma protocol that necessitated the use of the Bombe. The changes discussed applied only to the Army and Air Force. The Navy, on the other hand, had a completely different set of changes that required additional insight to allow us to make use of our Bombe on such naval messages.

3.3.1. The Naval Enigma

The German Navy used completely different Enigma models including `M1`, `M2`, `M3`, and ultimately `M4`, the last of which actually allowed for the use of four rotors [11]. The first three models are compatible with the Enigma model `I` that we have been discussing, so we will not be examining the differences between these machines in this thesis. Further, the `M4`'s additional rotor presented additional cryptographic challenges to cryptanalysts but the underlying theory described above is the same.

The most important difference of the Naval Enigma was the addition of three rotors – rotors `VI`, `VII`, and `VIII`. These rotors were exclusively used by the German Navy and they functioned differently than the aforementioned rotors `I-V`. In particular, these new rotors had 2 turnover notches, diametrically opposed, doubling the rate of rotor turnover. In theory, these new rotors increased the total rotor ordering possibilities to 336, but the naval operating procedure specified that one of the three rotors in a key sheet *must* be a naval rotor, thus reducing our rotor ordering space to only 276 [11].

The theory underlying the Bombe still allows for the naval Enigma to be broken,

but the additional rotors meant that the workload required to break such machines was vastly increased. Further, the additional turnover meant that cribs from naval transmissions no longer eliminated turnover by splitting the crib into two pieces. If instead we knew the rotor ordering, we could quickly determine cribs that ignored turnover and the amount of time we needed to use the Bombe would be vastly decreased. Turing invented a method by which cryptographers were able to reduce the possible rotor orderings to a set of only a few possibilities which became known as **Banburismus** [32, p. 76][4]. This method took inspiration from the clock method described in Section 2.5. To understand Banburismus, we must first understand the procedure by which naval Enigmas sent messages.

3.3.2. The Naval Enigma Protocol

Unlike Army and Air Force key sheets, which by this point had stopped including the window setting (*Grundstellung*) in their daily key, the Navy maintained this parameter of their daily key.

The naval operators were provided a large book of trigrams via a **K-Book** (*K-Buch*) as well as a large mapping from bigrams to other bigrams known as a **bigram table** [31, pp. 332, 334]. We will not discuss the exact details of choosing trigrams since they are not of cryptographic importance, but suffice it to say that the operator chose two trigrams from the K-Book (e.g. `CAC` and `ABB`) and arranged them in two rows as follows,

```
.CAC
ABB.
```

[4]The naming was due to cards used during the process being produced in Banbury, a small market town near Bletchley Park [32, p. 76].

The operator then chose two random letters (e.g. B and B) and filled in the remaining slots

BCAC

ABBB

Each vertical bigram would then be converted via the bigram table to a new bigram. That is

$$\texttt{BA} \mapsto \texttt{CC}$$

$$\texttt{CB} \mapsto \texttt{AA}$$

$$\texttt{AB} \mapsto \texttt{BC}$$

$$\texttt{CB} \mapsto \texttt{AA}$$

And these new bigrams were then arranged into tetragrams which in our example produces CCAA BCAA. These two tetragrams were then sent in plaintext at the start of our message. To actually encipher their message, the operator set their machine to the window setting specified by the key sheet and enciphered their second chosen trigram, ABB, to get their message key [31, pp. 332–336]. We will call the message key prior to being enciphered the **pre-message key** (in our case ABB). They then set their window setting to the message key and began enciphering their message.

The long short of this procedure is that if we have a copy of the bigram tables, we can work backwards to figure out the pre-message key for any naval message.

3.3.3. Pinches

The recovery of bigram tables was an incredible feat of the British Royal Navy. The Royal Navy boarded and recovered documents from a number enemy ships over the course of several years. The recovery of such documents became known as **pinches**. Some well known pinches are:

- The Narvik Pinch – This first pinch, on April 26, 1940, occurred when the *HMS Griffin* boarded the German *Polares* headed for the port of Narvik. Aboard, they found entire cribs, descriptions of the naval messaging protocol, and keylists [9, p. 259].
- The Krebs Pinch – On March 4, 1941, the *HMS Somali* boarded the German *Krebs*. Aboard, they found a keylist for the entire month of February [9, p. 260].
- The München Pinch – On May 7, 1941, the *HMS Somali* boarded the German weather ship *München*. Aboard, they found a keylist for the entire month of June [9, p. 260].
- The Lauenburg Pinch – On June 28, 1941, the *HMS Somali* boarded the German weather ship *Lauenburg*. Aboard, they found a keylist for the entire month of July [9, p. 260].
- The *U-110* Pinch – On May 9, 1941, the Royal Navy struck the German U-boat *U-110*. The crew abandoned the sinking ship, allowing members of the *HMS Bulldog* to board. Aboard, they recovered an Enigma machine and the bigram tables themselves (though they had already been reconstructed by Bletchley Park at this point) [9, p. 261].

These pinches provided Bletchley Park cryptanalysts with the information necessary to reverse engineer the bigram tables used by the German navy. With the bigram

tables in hand, we can assume that for any naval message we are able to deduce its pre-message key. With this assumption, we will now employ statistical analysis to determine which rotors were most likely used in the daily key for a particular day.

3.3.4. Repeats

Suppose we find the following two naval Enigma messages:

Pre-Message Key	Message
VFG	GXCYBGDSLVWBDJLKWIP...
VFX	YNSCFCCPVIPEMSGIZWF...

We know that after enciphering our pre-message key we will get our actual message key. Both VFG and VFX will be enciphered with the same settings (say π_i) so we know that they will be converted to

$$\texttt{VFG} \mapsto \pi_1(V)\pi_2(F)\pi_3(G)$$
$$\texttt{VFX} \mapsto \pi_1(V)\pi_2(F)\pi_3(X)$$

Given that both message keys begin with the letters $\pi_1(V)\pi_2(F)$, we know that they only differ by some unknown amount in their rightmost letter. Our first goal will be to determine the distance between $\pi_3(G)$ and $\pi_3(X)$. Note that we know such a distance ranges between -25 and 25 and we need not consider a distance of 0 as this would imply the pre-message keys are the same.

Recall our discussion from Section 2.5.1 regarding the index of coincidence. We know that when two texts represent the same poly-alphabetic cipher, the observed distribution of coincidences (Bletchley Park called these **repeats** [2, p. 95]) should

be much higher than two texts representing random text. Consider aligning our two texts with an offset of +11 letters, that is, the hypothesis that $\pi_3(\texttt{G}) = \pi_3(\texttt{X}) + 11$

```
...........GXCYBGDSLVWBDJLKWIPEHVYGQZWDTHRQXIKEESQSSPZXARIXEABQIRUCKHGWUEBPF
YNSCFCCPVIPEMSGIZWFLHESCIYSPVRXMCFQAXVXDVUQILBJUABNLKMKDJMENUNQ...........
```

In this case we find 3 repeats when considering an overlap of length 54, this repeat structure was denoted as 3/54.

For an offset of −9 we have

```
GXCYBGDSLVWBDJLKWIPEHVYGQZWDTHRQXIKEESQSSPZXARIXEABQIRUCKHGWUEBPF.......
.........YNSCFCCPVIPEMSGIZWFLHESCIYSPVRXMCFQAXVXDVUQILBJUABNLKMKDJMENUNQ
```

In this case we find 9 repeats when considering an overlap of 56 characters, but we additionally found two bigrams (ZW and QI) which were repeated. As bigrams are much more common in languages than random text, a point we will later discuss, we must factor this into our analysis. Thus we denote the repeat structure at this offset as $9^{\texttt{xx}}/56$, where each x denotes a bigram.

For any other n-gram repeat, we will use n in the exponent. For example, when considering 3 repeats in 57 letters, where we note that there is a trigram repeated we would denote this repeat structure as $3^3/57$. This is actually the case when examining our messages at an offset of −8.

Examining offsets from −9 to +9[5] we get the following chart of repeat structures

[5] In practice all offsets from −25 to +25 would be computed but the table has been shorted for brevity.

Offset	Repeat Structure
VFG = VFX-9	$9^{xx}/56$
VFG = VFX-8	$3^3/57$
VFG = VFX-7	$2/58$
VFG = VFX-6	$0/69$
VFG = VFX-5	$3/60$
VFG = VFX-4	$2/61$
VFG = VFX-3	$3/62$
VFG = VFX-2	$3/63$
VFG = VFX-1	$4^x/63$
VFG = VFX+1	$2/63$
VFG = VFX+2	$1/63$
VFG = VFX+3	$3/62$
VFG = VFX+4	$4/61$
VFG = VFX+5	$1/60$
VFG = VFX+6	$3/59$
VFG = VFX+7	$3/58$
VFG = VFX+8	$0/57$
VFG = VFX+9	$2/56$

Table 3.1: Repeat structure at various offsets [21, Section 2.3]

The question now arises, which of these offsets is most likely. Of course an offset of -9 has the most repeats, but perhaps the trigram at an offset of -8 makes this case more likely. In order to answer this question we will need a means to score these repeat structures. To do this, we must take a brief moment to discuss Bayesian statistics.

3.3.5. Bayesian Statistics

Turing thought of the relationships between texts at various offsets in terms of Bayesian statistics. Recall **Bayes' theorem** [23, p. 175],

Theorem 3.1. *Given events A and B, where $\mathbb{P}(B) \neq 0$ we have that*

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}.$$

This is given in terms of probability. We will see that the math becomes significantly easier to do in one's head if we think of these probabilities instead in terms of **odds**. That is,

Definition 3.2. For an event A with probability $\mathbb{P}(A)$ we say that the odds of A occurring are

$$\mathbb{O}(A) = \frac{\mathbb{P}(A)}{\mathbb{P}(\overline{A})}$$

where $\overline{A}$ denotes the probability of A not occurring, that is, $\mathbb{P}(\overline{A}) = 1 - \mathbb{P}(A)$.

Then we have

$$\begin{aligned}
\mathbb{O}(A|B) &= \frac{\mathbb{P}(A|B)}{\mathbb{P}(\overline{A}|B)} \\
&= \frac{\mathbb{P}(A)}{\mathbb{P}(\overline{A})} \cdot \frac{\mathbb{P}(B|A)\mathbb{P}(B)}{\mathbb{P}(B|\overline{A})\mathbb{P}(B)} \\
&= \frac{\mathbb{P}(A)}{\mathbb{P}(\overline{A})} \cdot \frac{\mathbb{P}(B|A)}{\mathbb{P}(B|\overline{A})} \\
&= \mathbb{O}(A) \cdot \frac{\mathbb{P}(B|A)}{\mathbb{P}(B|\overline{A})}.
\end{aligned}$$

We will denote $\frac{\mathbb{P}(B|A)}{\mathbb{P}(B|\overline{A})}$ as $\mathbb{F}(A, B)$. This is called the **Bayes factor** [21, Section 3.1]. The Bayes factor can be thought of as a measurement of how much the odds of an event increase when we add in a particular hypothesis. For a subsequent and independent event C we have,

$$\begin{aligned}
\mathbb{O}(A|B \wedge C) &= \mathbb{O}(A) \cdot \frac{\mathbb{P}(B \wedge C|A)}{\mathbb{P}(B \wedge C|\overline{A})} \\
&= \mathbb{O}(A) \cdot \frac{\mathbb{P}(B|A)}{\mathbb{P}(B|\overline{A})} \cdot \frac{\mathbb{P}(C|A)}{\mathbb{P}(C|\overline{A})} \\
&= \mathbb{O}(A) \cdot \mathbb{F}(A, B) \cdot \mathbb{F}(A, C).
\end{aligned}$$

Inductively, we have that for a series of independent events $B_1, \ldots, B_k$ we get,

$$\mathbb{O}(A|\wedge_{i=1}^{k} B_i) = \mathbb{O}(A) \cdot \prod_{i=1}^{k} \mathbb{F}(A, B_i).$$

If we are considering many events B_i, then we have to do a lot of multiplications by hand which are prone to error. For this reason, we consider the **log-odds** of these events occurring, turning these products into summations. In particular, for log-base b we have,

$$\log_b \mathbb{O}(A|\wedge_{i=1}^{k} B_i) = \log_b \mathbb{O}(A) + \sum_{i=1}^{k} \log_b \mathbb{F}(A, B_i).$$

This leads us to the following definition

Definition 3.3. A **ban** is a logarithmic unit of information gained from a Bayes factor $\mathbb{F}(A, B)$, such that 1 ban is equivalent to a Bayes factor which asserts that the odds of event A occurring are increased ten-fold when B is observed. That is

$$10^1 = \mathbb{F}(A, B)$$
$$\Rightarrow \log_{10}(\mathbb{F}(A, B)) = 1 \text{ ban}$$

Subsequently, a **deciban** is $\frac{1}{10}$ of a ban, representing an event's odds being increased by a factor of $10^{\frac{1}{10}}$, given B occurred. The unit we will be using is half of a deciban, known as a **hubdub (hdB)** [21, Section 4.1], representing an event's odds being increased by a factor of $10^{\frac{1}{20}}$, given B occurred. That is

$$20 \log_{10}(\mathbb{F}(A, B)) = 1 \text{ hdB}$$

In our case, if we use hubdubs, we arrive at the following relationship between our

odds

$$20 \log_{10} \mathbb{O}(A| \wedge_{i=1}^{k} B_i) = 20 \log_{10} \mathbb{O}(A) + \sum_{i=1}^{k} 20 \log_{10} \mathbb{F}(A, B_i)$$

Then a series of independent events $B_1, \ldots, B_k$ increase the odds of A by $\sum_{i=1}^{k} 20 \log_{10} \mathbb{F}(A, B_i)$ hdB. Note that if this value is positive, this implies that the event A is more likely to have occurred after observing each B_i. If this value is negative, then the event A is less likely to have occurred after observing each B_i. If it is 0, we have gained no information.

3.3.6. Scoring Charts

Bletchley Park cryptanalysts pre-compiled a large chart of how many hubdubs a particular observation of repeats was worth. Consider that at an offset of -13 our messages with pre-message keys `VFG` and `VFX` have a 6/52 repeat structure. That is, there are 6 monogram repeats occurring in 52 characters of observed text. We will call this observed event B. We will call the event that the our overlap is in-depth – that is, our rotors are aligned between both messages – as A.

To calculate how many hubdubs of information we gain we can compute.

$$20 \log_{10} \mathbb{F}(A, B) = 20 \log_{10} \frac{\mathbb{P}(B|A)}{\mathbb{P}(B|\overline{A})}$$

The denominator $\mathbb{P}(B|\overline{A})$ represents the probability that we observed 6 repeated monograms in 52 characters of text, given that our messages were not in-depth, that is, they both represent random streams of text. This is given by

$$\mathbb{P}(B|\overline{A}) = (\frac{1}{26})^6 (1 - \frac{1}{26})^{(52-6)}$$

The numerator $\mathbb{P}(B|A)$ represents the probability that we observed 6 repeated monograms in 52 characters of text, given that our messages are in-depth, that is, they both represent streams of enciphered German text. Based on observation of the corpus of naval Enigma messages, analysts determined that a monogram repeat occurs with a probability of $\frac{1}{17}$. Then our numerator is given by

$$\mathbb{P}(B|A) = (\frac{1}{17})^6(1 - \frac{1}{17})^{(52-6)}.$$

It then follows that the observation B, that is, that our texts are shifted at a distance of -13, is worth 13.59 (which Bletchley Park rounded to 13) hubdubs of information. That is, this new observation increases the odds of the messages being in-depth by a factor of $10^{\frac{13}{20}}$ – giving roughly 5 to 1 odds in favor of the text being in-depth. A scoring chart could be computed for all repeat structures of interest. Repeating this same calculation for various possible text lengths and number of repeated monograms, we may compute a table representing all hubdub values. The various tables and sheets which follow throughout this section, when not explicitly cited otherwise, were produced by our code available in the open-source project associated with this thesis [39].

	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
0	-7	-8	-8	-8	-8	-8	-9	-9	-9	-9	-9	-9	-10	-10	-10	-10	-10	-11	-11	-11	-11	-11	-12	-12
1	-4	-4	-4	-4	-4	-4	-5	-5	-5	-5	-5	-6	-6	-6	-6	-6	-7	-7	-7	-7	-7	-7	-8	-8
2	0	0	0	0	0	-1	-1	-1	-1	-1	-2	-2	-2	-2	-2	-2	-3	-3	-3	-3	-3	-4	-4	-4
3	4	4	3	3	3	3	3	2	2	2	2	2	1	1	1	1	1	1	0	0	0	0	0	0
4	8	7	7	7	7	7	6	6	6	6	6	6	5	5	5	5	5	4	4	4	4	4	3	3
5	11	11	11	11	11	11	10	10	10	10	10	9	9	9	9	9	8	8	8	8	8	8	7	7
6	15	15	15	15	15	14	14	14	14	14	13	13	13	13	13	13	12	12	12	12	12	11	11	11
7	19	19	19	19	18	18	18	18	18	18	17	17	17	17	17	16	16	16	16	16	15	15	15	15
8	23	23	23	23	22	22	22	22	22	21	21	21	21	21	20	20	20	20	20	20	19	19	19	19
9	27	27	27	26	26	26	26	26	25	25	25	25	25	25	24	24	24	24	24	23	23	23	23	23
10	31	31	30	30	30	30	30	30	29	29	29	29	29	28	28	28	28	28	27	27	27	27	27	27

Figure 3.7: Scoring sheet for Banburismus

Such a table of hubdub values were called **scoring sheets** (sometimes called **deciban sheets** from when calculations were done in decibans). The columns represent the length of text being observed and the rows indicate the number of repeated monograms. We now know how to score repeat structures which contain aligned monograms, but we have yet to account for bigrams or trigrams.

3.3.7. Bonus Scoring System

Bletchley Park cryptanalysts used a bonus system intended to account for the difference in frequency between monograms, bigrams, and trigrams in German text. The rule was – to find the value of a repeat structure find its row and column location, then add one row for each bigram and four rows for each trigram [2, p. 105].

We will illustrate this rule by example. To find the approximate hubdub value for an observed repeat structure of $6^{xx}/55$ we would examine row 6 column 55. However, to account for the additional two bigrams, we must moved down two rows (one for each bigram) to row 8 – giving a value of 20 hdB. Similarly, for a repeat structure

of $4^3/40$ we would examine row 4 column 40. Then, to account for the trigram we would move down four rows to row 8 – giving a value of 23 hdB.

The extent to which this bonus system was accurate as opposed to an arbitrary heuristic is hard to say. In order to determine the true hubdub value for a message containing bigrams and trigrams we would need to examine the entire corpus of Enigma message gathered for the purpose of Banburismus to determine the relative frequency of bigrams and trigrams. Bletchley Park cryptanalyst, Hugh Alexander, provides some n-gram frequencies [2, p. 96] as follows:

n-gram	**Probability**
Monogram	$\frac{1}{17}$
Tetragram	$100 \cdot \frac{1}{26^4}$
Hexagram	$15000 \cdot \frac{1}{26^6}$

Sadly, this information does not include bigram or trigram frequencies, and given the destruction of the vast majority of these intercepts [15, p. 29] [35, p. 193] it is unlikely that such a quantity will be amassed so as to be able to prove the accuracy of this bonus system.

There is no formal justification for the bonus system given in any historical accounts we examined. However, by working backwards from the bonus system we will attempt to reverse engineer the bigram frequency [13]. If this frequency lies within a reasonable estimate of bigram frequencies observed in German at-large, we can assume that this bonus system was likely based on careful analysis of actual bigram data in relation to the scoring table. To our knowledge, this is the first formal justification of the bonus scoring system used by Banburismus.

Given a bigram frequency of β in our Enigma corpus, we can approximate that a message with m monograms including b bigrams over n letters has a Bayes factor of

$$\frac{(\beta)^b(1-\beta)^{(n-1)-b}}{(\frac{1}{26^2})^b(1-\frac{1}{26^2})^{(n-1)-b}} \cdot \frac{(\frac{1}{17})^{(m-2b)}(1-\frac{1}{17})^{(n-m)}}{(\frac{1}{26})^{(m-2b)}(1-\frac{1}{26})^{(n-m)}}.$$

This makes several assumptions and overestimations. First, this assumes that having bigrams and monograms are independent events. Second, this ignores the fact that bigrams placed next to each other become higher order n-grams. However, this will do well enough to serve as justification for the bonus system.

By saying that such a Bayes factor is roughly equivalent to that of the Bayes factor of a message with $m+b$ monograms in n letters, we get

$$\frac{(\beta)^b(1-\beta)^{(n-1)-b}}{(\frac{1}{26^2})^b(1-\frac{1}{26^2})^{(n-1)-b}} \cdot \frac{(\frac{1}{17})^{(m-2b)}(1-\frac{1}{17})^{(n-m)}}{(\frac{1}{26})^{(m-2b)}(1-\frac{1}{26})^{(n-m)}} \approx \frac{(\frac{1}{17})^{m+b}(1-\frac{1}{17})^{n-m-b}}{(\frac{1}{26})^{m+b}(1-\frac{1}{26})^{n-m-b}}$$

$$\Rightarrow \frac{(\beta)^b(1-\beta)^{(n-1)-b}}{(\frac{1}{26^2})^b(1-\frac{1}{26^2})^{(n-1)-b}} \approx \frac{(\frac{1}{17})^{3b}(1-\frac{1}{17})^{-b}}{(\frac{1}{26})^{3b}(1-\frac{1}{26})^{-b}}$$

$$\Rightarrow \frac{(\beta)^b(1-\beta)^{(n-1)-b}}{(\frac{1}{26^2})^b(1-\frac{1}{26^2})^{(n-1)-b}} \approx (\frac{26}{17})^{3b}(\frac{17\cdot 25}{16\cdot 26})^b$$

$$\Rightarrow \frac{(\beta)^b(1-\beta)^{(n-1)-b}}{(\frac{1}{26^2})^b(1-\frac{1}{26^2})^{(n-1)-b}} \approx (\frac{26^2\cdot 25}{17^2\cdot 16})^b.$$

We will now make another assumption. We may not know β but we do expect it to be reasonably small. In an NSA paper regarding the index of coincidence, they estimate the probability of bigrams in German to be about 0.0097 [19, p. 4]. We may expect that β is reasonably close to, or at least in a similar magnitude to, this value. Given

its small magnitude, the ultimate application of $20\log_{10}$ to each side will render the term

$$\frac{(1-\beta)^{(n-1)-b}}{(1-\frac{1}{26^2})^{(n-1)-b}}$$

insignificant in comparison to $\frac{(\beta)^b}{(\frac{1}{26^2})^b}$. Thus, as a simplifying assumption this can be ignored. This leaves us with only a dependence on b, allowing us to approximately guess β. We have

$$\beta^b \approx (\frac{25}{17^2 \cdot 16})^b$$

Giving us $\beta \approx 0.00540$, or as Hugh Alexander might write, $\beta \approx 3.65 \cdot \frac{1}{26^2}$. Given all the simplifying assumptions made, this is still in the same magnitude of probability given by the NSA. Further, this estimate falls in line with the bonus system since it indicates that the index of coincidence of bigrams is roughly the cube of the index of coincidence of monograms – that is

$$\beta \cdot 26^2 \approx (\frac{1}{17} \cdot 26)^3,$$

hence the addition of an extra row for each bigram. Alternately, to put it in terms of logarithmic odds we have

$$20\log_{10}(\beta \cdot 26^2) \approx 3 \cdot 20\log_{10}(\frac{1}{17} \cdot 26)$$

that is, 1 bigram is worth 3 monograms in terms of hubdubs[6]. Thus we can say with reasonable confidence that the bonus system rule is grounded in legitimate data of bigram frequency. The same type of calculation can be done for trigrams, but the goal of this section was to give justification to the reader that the bonus system is

[6]In fact, this equation can be used to crudely estimate the bigram frequency which makes the bonus system exactly work. This is $\beta \approx 0.00529$, but the above estimate provides arguably more justification.

more than just a heuristic.

3.3.8. Distance Charts

Another factor we must account for is rotor turnover. We began with pre-message keys `VFG` and `VFX`, so all of this work has been assuming that only the last letter of the message key differs between our text; however, if the middle rotor turns over between the enciphering of our two pre-message keys, this assumption no longer holds. We must then ask the question, what is the likelihood that our two messages have message keys differing by only their last letter? If we assume that $\pi_3(\texttt{G}) = \pi_3(\texttt{X}) + n$ (thus corresponding to an offset of $+n$), then this is the same as asking what is the probability that a turnover did *not* occur between $\pi_3(\texttt{G})$ and $\pi_3(\texttt{X})$, that is, that a turnover did *not* occur over n letters distance of encipherment?

Let S represent the probability we are using a non-naval Enigma rotor as our rightmost rotor (rotors `I-V`). If we are in this case, then we have a $\frac{1}{26}$ chance of turnover at any given letter, so the likelihood of turnover over a distance of n letters is $\frac{n}{26}$. Thus the likelihood of *not* turning over is $\frac{26-n}{26}$.

If we are in the case with $1 - S$ probability that we are using a naval rotor (rotors `VI-VIII`), then we have a $\frac{1}{13}$ chance of turning over at any given letter, so the likelihood of *not* turning over is $\frac{13-n}{13}$. Note that for $n > 13$ we would have a probability of 1.

Then our probability of having *not* having rotor turnover occur over n letters distance

of encipherment, which we will denote τ_n, is

$$\tau_n = \begin{cases} S \cdot \dfrac{26-n}{26} + (1-S) \cdot \dfrac{13-n}{13}, & \text{for } n < 13 \\ S \cdot \dfrac{26-n}{26} & \text{for } 13 \leq n < 26 \end{cases}$$

Given that, by protocol, at least one rotor must be a naval rotor, we will now calculate S [21, Section 5.1].

We have $8 \cdot 7 \cdot 6$ possible arrangements of rotors for all possible naval rotors, but $5 \cdot 4 \cdot 3$ of these will contain no naval rotors. Additionally, only $7 \cdot 6 \cdot 3$ of these will have a naval rotor as their rightmost rotor, meaning that the probability of having a non-naval rotor as your rightmost rotor (i.e. $1 - S$) is

$$1 - S = \frac{7 \cdot 6 \cdot 3}{8 \cdot 7 \cdot 6 - 5 \cdot 4 \cdot 3}.$$

Thus $S \approx 0.543$. With this we can now compute the probability that a given offset represents message keys that only differ in their rightmost letter. In particular, we will generate a **distance chart** which associates for each offset, the hubdub value associated to this probability τ_N. We compute a chart with offsets from 0 to 25 and beneath we will show its associated hubdub value associated to the likelihood of turnover.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	-1	-1	-2	-2	-3	-4	-4	-5	-6	-7	-8	-10	-11	-12	-13	-14	-15	-16	-17	-18	-20	-22	-24	-28	-34

Figure 3.8: Distance chart

3.3.9. Distance Loss

We now need to consider how the probability of rotor turnover occurring effects our original estimates. The probability of observing a repeated monogram is now no longer just the frequency of monograms in German, but instead a weighted probability depending on the offset being examined. In particular we define for an offset of n

$$\ell_n \coloneqq \tau_n \cdot (\frac{1}{17}) + (1 - \tau_n)(\frac{1}{26})$$

This is the probability of finding a repeated German monogram weighted by the probability that the message produces a legitimate comparison of rotors with no turnover between encipherment of the pre-message keys. We add in the alternative possibility that rotor turnover did occur between encipherment of the pre-message keys in which case we expect the distribution of monograms to be random. We will say that ℓ_n represents the probability for a **loss** of $20 \log_{10} \tau_n$ hdB. Then, for each loss corresponding to a particular offset, we must recompute our scoring sheets – though now instead of using $\frac{1}{17}$ as our probability we will use ℓ_n. Here is the scoring table for a loss of -3 hdB,

	29	30	31	32	33	34	35	36	37	38	39	40
0	-4	-4	-4	-4	-4	-4	-5	-5	-5	-5	-5	-5
1	-1	-1	-1	-1	-1	-2	-2	-2	-2	-2	-2	-2
2	1	1	1	1	1	1	1	1	0	0	0	0
3	4	4	4	4	4	4	4	3	3	3	3	3
4	7	7	7	7	7	7	6	6	6	6	6	6
5	10	10	10	10	10	10	9	9	9	9	9	9
6	13	13	13	13	13	12	12	12	12	12	12	12
7	16	16	16	16	15	15	15	15	15	15	15	15
8	19	19	19	18	18	18	18	18	18	18	18	17
9	22	22	21	21	21	21	21	21	21	21	20	20
10	25	25	24	24	24	24	24	24	24	23	23	23
11	28	27	27	27	27	27	27	27	27	26	26	26
12	30	30	30	30	30	30	30	30	29	29	29	29
13	33	33	33	33	33	33	33	32	32	32	32	32
14	36	36	36	36	36	36	35	35	35	35	35	35

Figure 3.9: Scoring sheet for -3 hdB loss

3.3.10. n-grams

We have described scoring for monograms, bigrams, and trigrams, but messages can easily contain repeats of many more characters. We know that sheets existed for computing hubdub values of tetragrams and likely even higher n-grams, though how high we cannot definitively say without more historical sources. We also found no reference to tetragram sheets save for a couple values used for examples by Hugh Alexander, thus we will use these same examples later for reference.

What we do know is that in order to calculate the log-odds of tetragrams, messages were separated by the **crib room** into 20 categories labeled `I-XX`. Each category represented a message that was likely to contain a particular distribution of numerals (since many German numbers are 4 letters). The exact nature of these categories

were not discussed in any historical documentation found while researching for this thesis.

Within each category there are 13 subdivisions corresponding to where a tetragram occurred. The first 10 subdivisions represented a tetragram occurring in the first position, second position, and so on. The 11th subdivision corresponded to a tetragram occurring between the 11th and 30th position. The 12th subdivision corresponded to tetragrams occurring between the 30th position until the 30th to last position. The 13th subdivision corresponded to tetragrams occurring in the last 30 letters of the message. For each category and subdivision, a hubdub value was given [2, p. 107].

Filling in the values given by Hugh Alexander we provide a score sheet for tetragrams with missing values

	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV	XV	XVI	XVII	XVIII	XIX	XX
1	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
2	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
3	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
4	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
5	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
6	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
7	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
8	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
9	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
10	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
11	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
12	?	?	?	?	?	?	?	?	?	?	**19**	?	?	**23**	?	?	?	?	?	?
13	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Figure 3.10: Scoring sheet for tetragrams with the only known values filled in [2, p. 106]

In practice, to score a pair of messages with repeat structure $11^{4\mathbf{x}}/60$, we would first consult our tetragram scoring sheet for both our message categories and tetragram

locations to get our log-odds score by summing the two entries[7]. We would then consult the relevant score chart to score the remaining $7^{\mathbf{x}}/56$ repeat structure, adding these two scores together to get our total score for this repeat structure. With this, we can now score any repeat structure, save for one exception we will now discuss.

3.3.11. Dummyismus

There is one more caveat we need to address before we are ready to fully score our repeat structures. Based on observation, cryptanalysts at Bletchley Park noted that some percentage of messages were followed by "dummy" messages some way through. That is, short sections of German text were often followed by gibberish intended to confuse cryptanalysts. Dealing with such messages became known as **dummyismus** [21, Section 4.4].

Cryptanalysts in the crib room based on length, frequency, category, etc., would place a **blue line** at some point in each message, beyond which they estimated there was a 25% chance or higher of the message being a dummy. They noted this percentage on the sheet corresponding to each message [2, p. 104].

How they determined where to place the blue line and what percentage likelihood beyond this point the message was a dummy is not discussed by Hugh Alexander and likely depended on a number of additional factors which were known to the crib room but may remain a mystery to us.

In practice, if we are scoring a repeat structure of 0/30 before the blue line and $11^{3\mathbf{x}}/171$ after it, we would first score 0/30 as we have done before. We then must

[7] Technically we would need to add in the loss corresponding to our offset when computing our tetra scoring sheet. Alexander does not appear to do this but instead just subtracted the loss at the end. This approximation works sufficiently well in many cases but it is worth noting this discrepancy.

separately score the $11^{3\mathtt{X}}/171$ repeat structure on the sheet corresponding to its distance loss plus its dummyismus loss.

To see this, suppose our two messages have d_1 and d_2 probability, respectively, of being a dummy after the blue line. Then there is a

$$(1 - d_1) \cdot (1 - d_2)$$

chance of our messages both being genuine beyond the blue line. Thus we incur a loss of

$$20\log_{10}((1 - d_1) \cdot (1 - d_2))$$

hubdubs.

We can compute a lookup table in hdB for all percentages they encountered as follows,

	0%	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%	55%	60%	65%	70%
0%	0	0	-1	-1	-2	-2	-3	-4	-4	-5	-6	-7	-8	-9	-10
5%	0	-1	-1	-2	-2	-3	-4	-4	-5	-6	-6	-7	-8	-10	-11
10%	-1	-1	-2	-2	-3	-3	-4	-5	-5	-6	-7	-8	-9	-10	-11
15%	-1	-2	-2	-3	-3	-4	-5	-5	-6	-7	-7	-8	-9	-11	-12
20%	-2	-2	-3	-3	-4	-4	-5	-6	-6	-7	-8	-9	-10	-11	-12
25%	-2	-3	-3	-4	-4	-5	-6	-6	-7	-8	-9	-9	-10	-12	-13
30%	-3	-4	-4	-5	-5	-6	-6	-7	-8	-8	-9	-10	-11	-12	-14
35%	-4	-4	-5	-5	-6	-6	-7	-7	-8	-9	-10	-11	-12	-13	-14
40%	-4	-5	-5	-6	-6	-7	-8	-8	-9	-10	-10	-11	-12	-14	-15
45%	-5	-6	-6	-7	-7	-8	-8	-9	-10	-10	-11	-12	-13	-14	-16
50%	-6	-6	-7	-7	-8	-9	-9	-10	-10	-11	-12	-13	-14	-15	-16
55%	-7	-7	-8	-8	-9	-9	-10	-11	-11	-12	-13	-14	-15	-16	-17
60%	-8	-8	-9	-9	-10	-10	-11	-12	-12	-13	-14	-15	-16	-17	-18
65%	-9	-10	-10	-11	-11	-12	-12	-13	-14	-14	-15	-16	-17	-18	-20
70%	-10	-11	-11	-12	-12	-13	-14	-14	-15	-16	-16	-17	-18	-20	-21

Figure 3.11: Dummyismus scoring sheet

By adding the loss incurred from the dummyismus chart, to the loss incurred by the distance chart, we get the total loss corresponding to a particular repeat structure. To score this repeat structure we use the scoring sheet associated to that loss. We now have all the details necessary to score to messages whose pre-message key differs by only their last letter.

3.3.12. End-Wheel Comparison

We are finally ready to begin the process of **decibanning** [2, p. 105] based on end-wheel comparisons. Many of Hugh Alexander's scores do not align consistently with reproduced sheets and there are many points at which he likely switches between references to decibans and hubdubs. For this reason, we will be using numbers given by Steven Hosgood in "All You Ever Wanted to Know About Banburismus but Were Afraid to Ask" [21] as these more consistently align with the above description. The

reader is recommended to examine this resource for more detailed discussion about the scoring discrepancies between his analysis and those of Hugh Alexander. Regardless, the underlying theory is the same and the above discussion should give the reader and understanding of how such sheets looked and were used. We will give several examples to illustrate the various scenarios of decibanning.

TYQ = TYB + 5.

Suppose we are examining the hypothesis that pre-message keys TYQ and TYB represent an offset of +5. At this offset, we observe a repeat structure of $7^{xx}/32$, with no section after the blue line. Consulting our distance chart from Figure 3.8, a +5 offset gives us a loss of −3 hdB. We then get our score sheet calculated for a loss of −3 hdB from Figure 3.9 and consult row 7 column 32. Then, according to our bonus system, we move down one row for each bigram to row 9 – giving us a score of 21 hdB. We now enter this score in a **deciban sheet**. The sheet will show each offset and the associated score. Further, it will indicate which two letters are being compared, in our case Q and B. In our case we would fill in

	...	-1		+1	+2	+3	+4	+5	+6	...	
B	...	?	Q	?	?	?	?	22	?	...	B

Figure 3.12: End-wheel comparison for Q and B

If we had other pre-message keys which added observations regarding these distances (e.g. POQ and POB), then we would add another row below this and we could add up each column to get a more accurate score for each offset.

ASL = ASJ + 5.

Suppose we are examining the hypothesis that pre-message keys ASL and ASJ represent an offset of +5. At this offset, we observe a repeat structure of 0/30 before the blue

line, and $11^{3x}/171$ after the blue line. For the repeats before the blue line, consulting our distance chart from Figure 3.8, a +5 offset gives us a loss of −3 hdB. We then get our score sheet calculated for a loss of −3 hdB from Figure 3.9 and consult row 0 column 30 to get a score of −4 hdB. For the section after the blue line, the crib room estimates a −4 hdB loss due to the possibility that either is a dummy message. In this case, we must add this to our original loss of −3 due to distance, so we will ultimately consult a score sheet calculated for a loss of −7 hdB. Such a score sheet is

	167	168	169	170	171	172	173	174	175	176	177	178
10	5	5	5	5	5	5	4	4	4	4	4	4
11	7	7	7	7	7	6	6	6	6	6	6	6
12	9	9	9	9	8	8	8	8	8	8	8	8
13	11	11	11	11	10	10	10	10	10	10	10	10
14	13	13	13	12	12	12	12	12	12	12	12	12
15	15	15	14	14	14	14	14	14	14	14	14	14
16	17	16	16	16	16	16	16	16	16	16	16	16
17	18	18	18	18	18	18	18	18	18	18	18	18
18	20	20	20	20	20	20	20	20	20	20	20	19
19	22	22	22	22	22	22	22	22	22	22	21	21
20	24	24	24	24	24	24	24	24	24	23	23	23
21	26	26	26	26	26	26	26	26	26	25	25	25
22	28	28	28	28	28	28	28	28	27	27	27	27
23	30	30	30	30	30	30	30	29	29	29	29	29
24	32	32	32	32	32	32	31	31	31	31	31	31

Figure 3.13: Score sheet for −7 hdB loss

Given that our repeat structure after the blue line is $11^{3x}/171$, we first consult row 11 column 171. Then, according to our bonus system, we move down one row for the bigram and four rows for the trigram to row 16 – giving us a score of 16. Adding these two scores together we get a total score for this offset of 12 which we would

similarly record in the end-wheel comparison sheet for `L` and `J`.

Eventually, we will have computed scores across many messages and many offsets, and our deciban sheet will slowly be filled out. An example deciban sheet is given for comparing `V` and `W` derived from pre-message keys `ZOV/ZOW` and `GMV/GMW` respectively.

	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1		1	2	3	4	5	6	7	8	9	10	11	12	13	
V	-3	-3	-3	0	-8	6	-6	6	-12	-8	-8	23	0	W	3	-12	-8	7	-3	-6	-6	-2	1	-7	6	3	-2	V
W	-1	2	-2	9	-2	-1	-7	17	2	2	-5	4	-24	W	-6	-6	4	-3	-4	2	4	5	-1	-9	-7	-2	0	V
Sum	-4	-3	1	9	-10	5	-13	23	-1	-6	-13	**27**	-24		-3	-18	-4	4	-7	-4	-2	3	0	-16	-1	1	-2	Sum

Figure 3.14: End-wheel deciban sheet for `V` and `W` with summation on the bottom [21, Section 5.6]

After all this work we can now determine that the most likely candidate is that

$$\pi_3(\mathtt{W}) = \pi_3(\mathtt{V}) - 2$$

with a total score of 27 hdB. We can now add in the Bayesian prior representing the initial odds that any given offset is correct, which we denoted $\mathbb{O}(A)$. We have that there are 50 possible offsets, all of which are equally likely, giving us an extra factor of -34 hdB to get the true odds of our candidate as -7 hdB or roughly 2 to 1 odds against our candidate. These sheets will ultimately be used to disambiguate our deduced rotor possibilities. Later, when we show deciban sheets, we will show scores already being adjusted by the prior odds to make our work easier.

3.3.13. Middle-Wheel Comparison

So far we have only examined pre-message keys with their first two letters in common. In theory, there is no reason the same process cannot be applied to pre-message keys with only their first letter in common, although this requires significantly more offsets

being considered and significantly reduced odds.

We might think we need to handle all offsets from -650 to $+650$ but in practice, naval Enigma operating procedure restricted message lengths to 250 characters. Thus, we can only examine offsets of -250 to $+250$. This is still an incredible number of offsets to check by hand. To handle this, a section at Bletchley Park known as the **Freebornery** used tabulating machines to find repeats of length 4 or greater, since shorter n-grams were not worth the considerable odds against such a large array of candidates [2, pp. 102–103]. Messages with 4 or more repeats were compiled into a **tetra catalogue** as follows

Pre-Message Key	Position	Letters
AJU	4.8	LK TNSV RAJTLK
BYX	6.5	CS TNSV AVYTXK
ARX	5.6	AU TNTB LYDRMJ
AYR	11.11	XU TNTB LYDACV
CRS	6.1	NC TNXJ AVUSYM
CQT	8.4	NL TNXJ BRXSTO

Table 3.2: Example tetragram catalogue [2, p. 98]

Here we only underline repeats which are longer than 4 letters as these have the best chance at being legitimate. We think of each rightmost rotor position as having an alphabet of 26 letters before it turns over to a new alphabet of 26 letters. The left number in the position column denotes which alphabet we are in and the right number indicates which letter we are in that alphabet when the n-gram occurred. In this sense our first tetragram entry with a position of 4.8 means that the tetragram occurred in the 8th letter of the 4th alphabet, thus this occurred at the $26 \cdot 3 + 8$ or 86th letter of the message. It is worth noting that in this notation, for pre-message keys which differ by only their rightmost letter (as in the last section) an offset of $+n$ can be written as $+0.n$. In the end this will allow us to have one sheet with consistent

notation representing all our analyzed messages.

Left Rotor Distance Charts. To score these messages the first thing we must do is generate a new distance chart for all possible rotor turnovers between all -250 to $+250$ offsets. As before we would like to get a probability τ_n that our offset represents a legitimate comparison between message keys with their leftmost letter in common. This is the same as asking the probability that turnover of the leftmost rotor dd *not* occur. Now we must consider more scenarios than in our one rotor analysis. Recall that there are $8 \cdot 7 \cdot 6$ total arrangements of rotors, but $5 \cdot 4 \cdot 3$ of these are not allowed since they do not include any naval rotors. Then there are 276 total rotor combinations we are examining. The two rightmost rotors can be in one of the following cases

- Two non-naval rotors – For each of the 3 naval rotors as the lefthand rotor we have $5 \cdot 4$ possible non-naval rotor combinations for the right two rotors, giving 60 such combinations.
- Two naval rotors – We have $3 \cdot 2$ possible naval rotor combinations for the right two rotors, leaving 6 remaining rotor possibilities for the leftmost rotor, thus giving 36 such combinations.
- Middle naval rotor and rightmost non-naval rotor – We have $3 \cdot 5$ possible naval rotor combinations for the right two rotors, leaving 6 remaining rotor possibilities for the leftmost rotor, thus giving 90 such combinations.
- Middle non-naval rotor and rightmost naval rotor – As above this has 90 combinations.

For each of this cases we have varying likelihood of rotor turnover. Recall that when dealing with the leftmost rotors period of turnover we must consider double

stepping as described in section 1.1.2. Thus we have

- Two non-naval rotors – Leftmost rotor turns with a $26 \cdot 25 = 650$ period.
- Two naval rotors – Leftmost rotor turns with a $13 \cdot 12 = 156$ period.
- Middle naval rotor and rightmost non-naval rotor – Leftmost rotor turns with a $26 \cdot 12 = 312$ period.
- Middle non-naval rotor and rightmost naval rotor – Leftmost rotor turns with a $13 \cdot 25 = 325$ period.

Using the same arguments to compute τ_n as in Section 3.3.8 we get

$$\tau_n = \begin{cases} \frac{60}{276} \cdot \frac{650-n}{650} + \frac{90}{276} \cdot \frac{325-n}{325} + \frac{90}{276} \cdot \frac{312-n}{312} + \frac{36}{276} \cdot \frac{156-n}{156}, & \text{for } n < 156 \\ \frac{60}{276} \cdot \frac{650-n}{650} + \frac{90}{276} \cdot \frac{325-n}{325} + \frac{90}{276} \cdot \frac{312-n}{312}, & \text{for } 156 \leq n < 312 \\ \frac{60}{276} \cdot \frac{650-n}{650} + \frac{90}{276} \cdot \frac{325-n}{325}, & \text{for } 312 \leq n < 325 \\ \frac{60}{276} \cdot \frac{650-n}{650} & \text{for } 325 \leq n < 650 \end{cases}$$

This lets us compute τ_n from up to an offset of ± 650 but as mentioned we will only need this for ± 250. We can now generate a distance chart as before but now we consider each value from in the range -250 to $+250$.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-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	-1	-1	-1	-1	-1	-1	-1	-1	-2	-2
2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-5	-5	-5	-5
5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6
6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-7	-7	-7	-7	-7	-7	-7	-7	-7	-7	-7	-7	-7	-7	-7
7	-7	-7	-7	-7	-7	-7	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8
8	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-10	-10	-10	-10	-10	-10	-10	-10	-10
9	-10	-10	-10	-10	-10	-10	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11	-12	-12	-12	-12	-12	-12

Figure 3.15: Distance chart for left rotor turnover

With this chart we now know what the loss for our scoring sheet needs to be for a particular message pair. From here we can use our exact same scoring method to compute the log-odds of score for entries in the tetragram catalogue. To get the true odds of our candidate offset we must consider the Bayesian prior representing the initial odds that any given offset is correct. We have that there are 1300 possible offsets, all of which are equally likely, giving us an extra factor of -62 hdB to get the true odds of our candidate.

Score Conversion. A candidate offset on the tetragram catalogue does not just store information about the middle wheel but also additional information about the end wheel which we can use to further increase accuracy of our end wheel comparison. In order to do this, we need to determine our score with respect to the Bayesian prior for representing the initial odds of an offset in ± 25 being correct. Given that we have used the same scoring system regardless of analyzing middle-wheel or end-wheel comparisons, we have that the Bayesian factors we computed would be the same whether we were analyzing middle-wheel or end-wheel comparisons. Thus, we need only account for the difference in the prior odds between middle-wheel and end-wheel comparisons. We have -62 hdB prior log-odds for middle-wheel comparison and -34

hdB log-odds for end-wheel comparison. Thus to convert our score to that of an end wheel comparison we must incur a loss of $-34-62=-28$ hdB. If p is the probability that our messages are in-depth, then we have that the odds with a loss of -28 hdB incurred is

$$\frac{p\cdot 10^{\frac{-28}{20}}+(1-p)(1-10^{\frac{-28}{20}})}{(1-p)}$$
$$=10^{\frac{-28}{20}}\frac{p}{1-p}-10^{\frac{-28}{20}}+1$$

Note that $\frac{p}{1-p}$ is just our current odds and the above equation is our new odds. Then for a score of s, we have $10^{\frac{s}{20}}=\frac{p}{1-p}$. Thus, to convert from a middle-wheel to an end-wheel comparison, we compute

$$20\log_{10}(10^{\frac{s}{20}}\cdot 10^{\frac{-28}{20}}-10^{\frac{-28}{20}}+1).$$

We can compute a conversion chart from middle-wheel scores to end-wheel scores as follows.

Middle-wheel Score	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
End-wheel Score	1	1	1	2	2	2	2	3	3	4	4	4	5	5	6	6	7	8	8	9	10	10	11	12	13

Figure 3.16: Chart to convert middle-wheel scores to end-wheel scores

Now we can include pre-message keys with only their left letter in common in our deciban sheets for end-wheel analysis.

3.3.14. Scritchmus

We have developed a lot of theory to score repeat structures and various letter offsets so it is best to recap what we get at the end of this analysis. We have,

- A list of repeat structures and their scores which we will call a **catalogue**. This

is just a scored tetra catalogue combined with a scored end-wheel catalogue. An example is as follows[8]

	Catalogue		
(1)	ARX = AYR + 6.5	Octagram	Certain
(2)	NTP = NXU + 2.15	Enneagram	Certain
(3)	LLK = LAP + 9.24	Enneagram	Certain
(4)	VRN = VXR + 0.21	$16^6/95$	100 : 1 on
(5)	STK = STN + 0.7	$23^{33}/256$	15 : 1 on
(6)	RWL = RWC + 0.13	$14^4/100$	6 : 1 on
(7)	BVY = BLT + 1.6	$19^5/140$	5 : 1 on
(8)	UJA = UMY + 5.3	$18^5/210$	3 : 2 on
(9)	BAQ = BWS + 0.17	$20^7/274$	50 : 1 on

Figure 3.17: Catalogue of repeat structures [2, p. 99]

- A list of deciban sheets for a variety of letters combinations compiled from summed likelihoods of letter offsets gleaned from both end-wheel and middle-wheel comparisons. One such deciban sheet is as follows,

	-6	-5	-4	-3	-2	-1		1	2	3	4	5	6	
T	1	2	7	7	4	3	H	5	3	1	7	4	7	T

Figure 3.18: End-wheel deciban sheet for H and T [2, p. 101]

The process of combining these two pieces of information to recover the middle and end rotors became known as **scritchmus** [21, Section 7.0].

Determining π_3**.** We are now going to determine the mapping of the third letter in the pre-message key π_3. We begin by making a chain of letters whose distances are deduced from the likely candidates from our catalogue 3.17. Repeat (1) indicates

[8]For sufficiently high odds we just write "certain" as it may as well be.

that $\pi_3(\mathtt{X}) = \pi_3(\mathtt{R}) + 5$. Let us make an initial assumption that $\pi_3(\mathtt{R}) = \mathtt{M}$ then we deduce that $\pi_3(\mathtt{X}) = \mathtt{M} + 5 = \mathtt{R}$. We can place these in an alphabet as follows

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
............R....X........
```

In this way, adding in repeats (2)-(5) we get from our initial hypothesis $\pi_3(\mathtt{R}) = \mathtt{M}$,

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
P......N...UR.K..X........
```

Recall that π_3 must be an involution so if $\pi_3(\mathtt{R}) = \mathtt{M}$ we must also have $\pi_3(\mathtt{M}) = \mathtt{R}$, but of course this contradicts our derivation that $\pi_3(\mathtt{X}) = \mathtt{R}$. Then our initial hypothesis $\pi_3(\mathtt{R}) = \mathtt{M}$ is likely wrong. We must therefore continue through all hypotheses for the value of $\pi_3(\mathtt{R})$. We can eliminate possibilities as follows,

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	Notes
P	.	.	.	.	.	.	N	.	.	.	U	R	.	K	.	.	X	.	.	.	.	.	.	.	.	Contradicts reciprocal R and M
.	P	.	.	.	.	.	.	N	.	.	.	U	R	.	K	.	.	X	.	.	.	.	.	.	.	Contradicts reciprocal P and B
.	.	P	.	.	.	.	.	.	N	.	.	.	U	R	.	K	.	.	X	.	.	.	.	.	.	Contradicts reciprocal N and J
.	.	.	P	.	.	.	.	.	.	N	.	.	.	U	R	.	K	.	.	X	.	.	.	.	.	Contradicts reciprocal R and P
.	.	.	.	P	.	.	.	.	.	.	N	.	.	.	U	R	.	K	.	.	X	.	.	.	.	Contradicts reciprocal P and E
.	.	.	.	.	P	.	.	.	.	.	.	N	.	.	.	U	R	.	K	.	.	X	.	.	.	Contradicts R cannot be self-reciprocal
.	.	.	.	.	.	P	.	.	.	.	.	.	N	.	.	.	U	R	.	K	.	.	X	.	.	Contradicts N cannot be self-reciprocal
.	.	.	.	.	.	.	P	.	.	.	.	.	.	N	.	.	.	U	R	.	K	.	.	X	.	**Possible**
.	.	.	.	.	.	.	.	P	.	.	.	.	.	.	N	.	.	.	U	R	.	K	.	.	X	Contradicts reciprocal P and I
X	.	.	.	.	.	.	.	.	P	.	.	.	.	.	.	N	.	.	.	U	R	.	K	.	.	Contradicts U cannot be self-reciprocal
.	X	.	.	.	.	.	.	.	.	P	.	.	.	.	.	.	N	.	.	.	U	R	.	K	.	Contradicts reciprocal K and Y
.	.	X	.	.	.	.	.	.	.	.	P	.	.	.	.	.	.	N	.	.	.	U	R	.	K	Contradicts reciprocal X and C
K	.	.	X	.	.	.	.	.	.	.	.	P	.	.	.	.	.	.	N	.	.	.	U	R	.	Contradicts reciprocal X and D
.	K	.	.	X	.	.	.	.	.	.	.	.	P	.	.	.	.	.	.	N	.	.	.	U	R	Contradicts reciprocal P and N
R	.	K	.	.	X	.	.	.	.	.	.	.	.	P	.	.	.	.	.	.	N	.	.	.	U	**Possible**
U	R	.	K	.	.	X	.	.	.	.	.	.	.	.	P	.	.	.	.	.	.	N	.	.	.	Contradicts P cannot be self-reciprocal
.	U	R	.	K	.	.	X	.	.	.	.	.	.	.	.	P	.	.	.	.	.	.	N	.	.	Contradicts reciprocal X and H
.	.	U	R	.	K	.	.	X	.	.	.	.	.	.	.	.	P	.	.	.	.	.	.	N	.	Contradicts reciprocal R and D
.	.	.	U	R	.	K	.	.	X	.	.	.	.	.	.	.	.	P	.	.	.	.	.	.	N	**Possible**
N	.	.	.	U	R	.	K	.	.	X	.	.	.	.	.	.	.	.	P	.	.	.	.	.	.	Contradicts reciprocal K and H
.	N	.	.	.	U	R	.	K	.	.	X	.	.	.	.	.	.	.	.	P	.	.	.	.	.	Contradicts reciprocal U and F
.	.	N	.	.	.	U	R	.	K	.	.	X	.	.	.	.	.	.	.	.	P	.	.	.	.	**Possible**
.	.	.	N	.	.	.	U	R	.	K	.	.	X	.	.	.	.	.	.	.	.	P	.	.	.	Contradicts reciprocal N and D
.	.	.	.	N	.	.	.	U	R	.	K	.	.	X	.	.	.	.	.	.	.	.	P	.	.	Contradicts reciprocal X and O
.	.	.	.	.	N	.	.	.	U	R	.	K	.	.	X	.	.	.	.	.	.	.	.	P	.	Contradicts reciprocal R and K
.	.	.	.	.	.	N	.	.	.	U	R	.	K	.	.	X	.	.	.	.	.	.	.	.	P	Contradicts reciprocal U and K

In the end we arrive at only 4 possible alphabets. By filling in values given by the reciprocal nature of π_3 we arrive at these four alphabets [2, p. 100]:

	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	Notes
(a)	.	.	.	.	.	.	.	P	.	.	V	.	.	O	N	H	.	T	U	R	S	K	.	Y	X	.	Picks up (7), denies (8)
(b)	R	.	K	.	.	X	.	.	.	.	C	.	.	V	P	O	.	A	.	.	Z	N	.	F	.	U	Denies (6)/(8)
(c)	.	.	.	U	R	.	K	.	.	X	G	.	.	Z	.	S	.	E	P	.	D	.	.	J	.	N	Denies (9)
(d)	.	.	N	.	.	.	U	R	.	K	J	.	X	C	.	V	.	H	.	.	G	P	.	M	.	.	

After filling in the alphabets further, we notice that some alphabets either contradict (deny) or support (pick up) the remaining hypotheses we have not included, that is, hypotheses (6)-(9). As (6) and (9) are both strong hypotheses, this will discount the validity of alphabets (b) and (c). Leaving us with remaining alphabets (a) and (d). Note that the denial of (8) for alphabet (a) is not that strong of an indicator against this alphabet since (8) has roughly even odds of being correct and picking up (7) is such that it outweighs the denial of (8) anyways.

We now need to determine which of (a) or (d) is correct. To do this we can turn to our deciban sheets 3.18. For example, we note that alphabet (a) is such that $\pi_3(\mathtt{T}) = \pi_3(\mathtt{H}) + 2$ which is evidence 3 hdB in favor of this alphabet. Alexander's example found, that after adding up all the offsets in alphabet (a) against deciban sheets, we find that the alphabet has a total score of 54 hdB [2, p. 101], almost certainly indicating that alphabet (a) is our correct choice for π_3.

We can now further fill in the alphabet from hypotheses that got picked up. We begin by including hypothesis (9) and using π_3's reciprocal property to get the alphabet

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
.......P..VQ.ONHLTURSK.YX.
```

Now we that we have add L to our alphabet we can make use of hypothesis (6) to get

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
..DC...P..VQ.ONHLTURSK.YX.
```

With a large catalogue, we will often be able to fill out the entire alphabet in this way.

Recovering the End Rotor. We will now examine what possible end rotors could be used that corroborate our alphabet for π_3. Consider that for our end-wheel comparisons like hypothesis (5) and (6), we have strong indication that no turnover occurred during these repeat structures. Therefore we may assume that no turnover occurred between the encipherment of C and L, and no turnover occurred between N and K. Thus, we can assume no turnover occurred between C and K. Given that this is a stretch of 18 letters, we can immediately eliminate the possibility that the rightmost rotor is a naval rotor as it would have turnover at least every 13 letters. That leaves us with rotors I-V. Rotors I, II, and IV, all have turnover between C and K. Thus we are left with only the possibilities that the rightmost rotor is III or V – narrowing our possibilities from 8 to 2.

Recovering the Middle Rotor. Determining the middle rotor is much the same as the above procedure. Our goal is to get a likely alphabet for π_2 and determine where turnover of the leftmost wheel occurred so as to eliminate possible middle rotors. The only difference is that, based on our above analysis, we now strongly suspect that turnover of the rightmost rotor occurs between K and C. Thus, a hypothesis like (2) which asserts that, in the middle wheel alphabet $\mathtt{T} = \mathtt{X} + 2$, must be adjusted to state that $\mathtt{T} = \mathtt{X} + 3$, since we know turnover of the rightmost rotor occurred in

passing from P to U. Adjusting our offsets as necessary we can perform the exact same analysis as above to determine a small set of possibilities for the middle rotor.

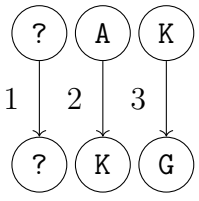
All in all, this process can reduce the 276 rotor orderings of the naval Enigma to only a handful of possibilities, thus making searching for naval settings on the Bombe tractable. However, in order to actually make use of the Bombe we still need a crib to determine valid settings.

3.3.15. Running the Bombe

By this point we have determined a handful of possible rotor orderings, and we have, with luck, determine the entirety of permutations π_2 and π_3. Note though that the permutations π_2 and π_3 are themselves plaintext-ciphertext pairings! For example, if we determine π_2 and π_3 to be

$$\begin{array}{rl} & \texttt{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \\ \pi_2 & \texttt{KJNRPQUOLBAISCHEFDMYGZXWTV} \\ \pi_3 & \texttt{OVJWLSKMUCGEHTAXRQFNIBDPZY}. \end{array}$$

Then we get a series of 2 letter plaintext-ciphertext pairings, such as



Using these plaintext-ciphertext mappings we can construct a menu as follows:

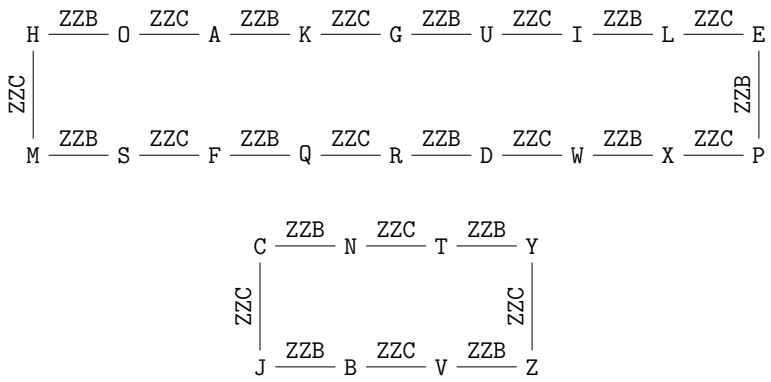


Figure 3.19: Menu constructed from Banburismus

Equipped with this menu, the Bombe can now be run for our limited set of rotor orderings and can deduce the steckerings and positions of rotors necessary to produce our menu – that is, the settings which convert pre-message keys into message keys.

The EINS Catalogue. Recall from Section 3.2.5, that the settings given by a stop of the Bombe do not necessarily correspond to the true ring settings. In order to determine the true ring settings we would normally use our plaintext-ciphertext pairing to determine at which point turnover occurred and adjust our ring setting as necessary. However, our cribs consist exclusively of two letter plaintext-ciphertext pairings, after which the message is enciphered with the message key and is thus meaningless. Therefore we can no longer deduce at what point during our message turnover occurs, and thus can no longer use this method to determine the true ring setting.

In analyzing naval Enigma traffic, cryptanalysts found that nearly all messages (around 90%) contained the word EINS (German for "one") at some point in the message [37, pp. 140–142]. This makes sense for a number of reasons:

(1) Naval messages regularly dealt with numbers, all of which needed to be typed out in full. This statistical observation is what made the use of the tetra cata-

logue so valuable [21, Section 9.3].

(2) An odd quirk known as **Benford's law** tells us that 1 appears more commonly than any other leading digit in numerical data [5, p. 556].

(3) `EINS` shows up naturally in many German words either as a sub-component of a word or when spanning across two words [21, Section 9.3].

Given that we have a known rotor ordering, steckering, and ring/window setting whose relative distance is correct, but may need to be adjusted to get the true ring setting, we can compile a list of all 26^3 encipherments of `EINS` which preserve the relative distance between our ring and window setting. Originally this process had to be done by hand with catalogues known as **corsets** but a machine called the **test-plate** or **baby** was created which automatically enciphered `EINS` at all such positions into an `EINS` **catalogue** [37, pp. 140–142].

The catalogue could then be used to determine locations within our message corpus, along with settings, that would produce `EINS`. If we then continue deciphering our message from the start of the `EINS` location, as in Section 3.2.5, we can decipher plaintext until we encounter gibberish and adjust the right, and subsequently middle, ring setting until we have deduced the entire ring setting. With this final component, we are now able to completely decipher naval Enigma messages.

It is clear from the amount of messages, sheets, and scores that need to be compared and tabulated that Banburismus was a great deal of work. Alexander estimates that an average day involved 400 messages with an average length of 150 characters. In total around 6000 comparisons needed to be made all of which needed to be scored and counted [2, p. 109]. Particular credit should be given to whom Alexander noted

was “one of the best Banburists”, Joan Clarke [2, p. 73]. Not only was she an exceptional Banburist, she was also one of only a handful of female cryptanalysts working at Bletchley Park.

The work done by Banburists provided crucial information on naval operations, including U-Boat attacks which were crippling supply lines and starving Britain.

4

Stops

The Turing-Welchman Bombe was remarkably adept at computing daily keys in a tractable manner. In optimal conditions, with a well-crafted and fortunate menu, the Bombe could run in as little as 20 minutes and produce exactly the daily key needed to decrypt messages [2, p. 12]. To approach these optimal conditions, we must consider what it means for one menu to be stronger than another. In this chapter, we will explore the relationship between menus and the number of stops the Bombe is expected to encounter. Equipped with this information, a cryptanalyst would be able to select the menu that yields the fewest stops.

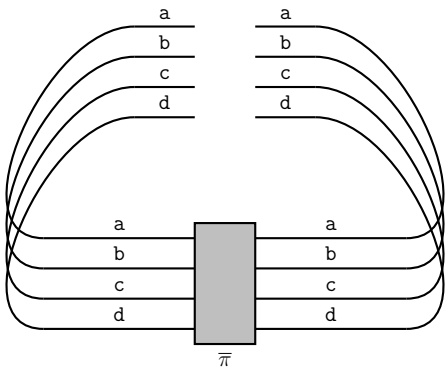
Each stop consumed precious time, as operators needed to verify whether they corresponded to a valid key. Therefore, menu selection becomes an integral factor in reducing the time between intercepting transmissions and recovering the key for that day. Even a reduction of minutes could provide the slight edge that the Allies needed to preempt an attack. Thus, developing an accurate model by which we can correlate menus to their expected number of stops is a matter of strategic urgency.

Section 4.1

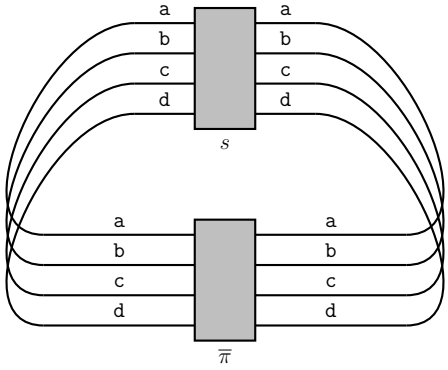
Turing's Model

We begin by describing the model for the expected number of stops given by Turing himself in *The Prof's Book* [37, pp. 112–116]. Turing approaches this problem not by computing the expected number of stops itself, but instead by computing the expected number of **normal stops** over all steckering hypothesis. These normal stops are described by Turing as, "positions at which by altering the point at which the current enters the diagonal board, one can make 25 relays close" [37, p. 112]. In the case of a single loop in our menu, this is equivalent to the statement that the resulting loop in the Bombe has a singleton cycle. If we apply current to this singleton cycle all the remaining relays will close. For now we will ignore the impact of the diagonal board.

Turing considered a menu in which no loops occurred which he called a **web**. In this case every position of the Bombe and every initial steckering hypothesis would create a normal stop, as there is no feedback necessary to electrify any additional wires on a given cable. Turing considers not only the 26^3 possible rotor positions of the Bombe, but also the 26 initial steckering hypotheses we could input. In our case, all steckering hypotheses and all rotor configurations produce a normal stop so we get 26^4 total normal stops over all steckering hypotheses. In our simplified model with only 4 characters this may look as follows,



Turing then considered the effect of adding an edge to our menu which would form a loop, he called such an edge a **chain-closing constatation**. He wanted to deduce the likelihood that adding such an edge turns our normal stops into anything other than a normal stop. We will denote this chain-closing edge as s. Our diagram would then be,



Suppose that before adding the edge s, a particular steckering hypothesis $S(x) = y$, produces a normal stop. What is the probability that by adding the permutation s we are now no longer in the situation of a normal stop?

Given that by supposition electrifying wire xy through $\overline{\pi}$ has only a single live wire, the permutation s need only connect the live wire to any of the 3 remaining non-electrified wires to arrive at anything other than a normal stop. Thus there is $\frac{3}{4}$ chance that $s \circ \overline{\pi}$ is no longer a normal stop. Conversely, there is a $\frac{1}{4}$ chance that s

fails to remove our normal stop.

In the case of a 26 character alphabet, the same logic follows, and we have a $\frac{1}{26}$ chance that a chain-closing constatations fails to remove a normal stop. Given that we originally expected 26^4 normal stops over all steckering hypotheses without any closures, we would now expect that adding a chain-closing constatation to our menu would have $\frac{1}{26}$ as many normal stops. Thus adding one closure to our menu now brings the expected number of normal stops over all steckering hypotheses down to $26^{4-1} = 26^3$.

Turing extends this argument to explain that, for a menu with c loops (which he called **closures**) we would expect 26^{4-c} normal stops over all steckering hypotheses [37, p. 116].

4.1.1. Dixon's Theorem

A natural question arises - why should the number of normal stops over all steckering hypotheses approximate the number of actual stops the Bombe encounters? First, we are only considering a subset of possible stops so, from this perspective, we might expect our approximation to be under-representing the true number of stops. Second, we are considering these stops over *all* steckering hypotheses so, from this perspective, we might expect our approximation to be around 26 times greater than the actual answer. Yet, testing Turing's formula against the actual Bombe, in many cases, we find that his approximation of 26^{4-c} falls roughly in line with the actual number of stops. To answer this question we must show the following:

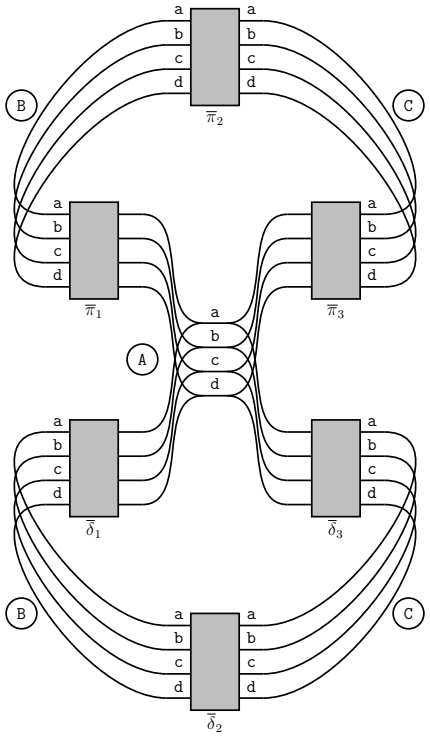
(1) The number of normal stops approximates the number of stops.

(2) The consideration of all 26 steckering hypotheses for each rotor position averages

out in such a way that we only account for 1 steckering hypothesis per each rotor position thus removing any double counting of a particular rotor position.

Transitive Subgroups. To show these two points, we require additional mathematical tools. We first need to translate our mechanical understanding of the Bombe's electrification of wires into a mathematical one. For a single loop this is quite trivial. If the permutation representing our loop is $\overline{\pi}$, then the number of normal stops over all steckering hypotheses is just the number of singleton cycles in $\overline{\pi}$'s cycle decomposition.

Once we add in additional closures this becomes slightly more complicated. Suppose we have two closures, with each separate closure being represented by the permutations $\overline{\pi}$ and $\overline{\delta}$ respectively.



How can we determine which wires are connected to which in this complex arrangement of permutations? To answer this we introduce

Definition 4.1. The subgroup of S_n generated by two permutations σ, $\tau \in S_n$ is

$$\langle \sigma, \tau \rangle \coloneqq \{\sigma^{a_1}\tau^{b_1} \dots \sigma^{a_k}\tau^{b_k} \mid k \in \mathbb{N},\ a_i, b_i \in \mathbb{Z}\}$$

This is the set of all group elements obtained by applying finite compositions of σ, τ, and their inverses [4, p. 47].

Consider how this relates to our two loop Bombe arrangement with $\overline{\pi}$ and $\overline{\delta}$. The electricity is free to flow through any number of iterations forwards and backwards through both $\overline{\pi}$ and $\overline{\delta}$. Applying a current at some input wire will propagate in such a way that it reaches possible letters which can be reached by some sequence of applications of $\overline{\pi}$, $\overline{\delta}$, and their inverses. Then on cable `A`, applying a current at some input x can only reach wire y on that cable if $\exists\ \sigma \in \langle \overline{\pi}, \overline{\delta} \rangle$ such that $\sigma(x) = y$. If we want to know for a given input wire, what other wires it can reach, we need to know which letters are connected to which through permutations in $\langle \overline{\pi}, \overline{\delta} \rangle$.

We note that $\langle \overline{\pi}, \overline{\delta} \rangle$ has a natural group action on $\mathbb{N}_4$ given by

$$\sigma \cdot x \coloneqq \sigma(x) \text{ for } \sigma \in \langle \overline{\pi}, \overline{\delta} \rangle \text{ and } x \in \mathbb{N}_4.$$

Consider an orbit for this group action. That is, for $x \in \mathbb{N}_4$ we have

$$\langle \overline{\pi}, \overline{\delta} \rangle \cdot x \coloneqq \{\sigma(x) \mid \sigma \in \langle \overline{\pi}, \overline{\delta} \rangle\}.$$

We can think of this orbit as all elements in $\mathbb{N}_4$ which x can reach via some sequence finite compositions of $\overline{\pi}$, $\overline{\delta}$, and their inverses. This is exactly analogous to the set of wires on cable `A` which can be reached from an input on wire x. Then these orbits partition $\mathbb{N}_4$ and tell us which wires are in a connected loop in our Bombe arrangement.

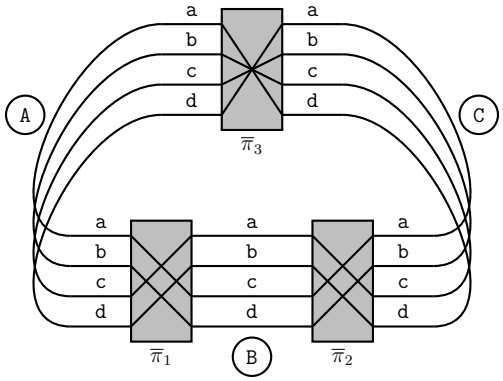
In this way, the partition given by the set of orbits $\mathbb{N}_4/\langle\overline{\pi},\overline{\delta}\rangle$ represents which wires are connected to which wires in our diagram. If $\mathbb{N}_4/\langle\overline{\pi},\overline{\delta}\rangle = \{\{\mathtt{a},\ldots,\mathtt{d}\}\}$ then we know that electrifying any wire on cable `A` will electrify all wires in the diagram. In such a case we say that $\langle\overline{\pi},\overline{\delta}\rangle$ forms a **transitive subgroup** of S_4. Formally,

Definition 4.2. Let $H \leq G$ be a subgroup of G which acts on a set X. We say H acts **transitively** on $S \subseteq X$ provided that

$$\forall\ a, b \in S,\ \exists\ \sigma \in H \text{ s.t. } \sigma(a) = b.$$

If H acts transitively on X we say that H is a **transitive subgroup of** G [4, p. 177].

Consider how this relates to our single loop example.



In this loop, we have that on cable `A`, wires `a` and `d` are connected, and wires `b` and `c` are connected. We explained that his could be simply determined by examining the cycle decomposition of $\overline{\pi}$ which in this case is

$$(\mathtt{ad})(\mathtt{bc}).$$

However, we can get the exact same information by simply considering the set orbits of $\langle\overline{\pi}\rangle$ acting on $\{\mathtt{a},\ldots\mathtt{d}\}$, which would give us

$$\{\{\mathtt{a},\mathtt{d}\},\{\mathtt{b},\mathtt{c}\}\},$$

giving us an equal picture of the connections between wires on the A cable. Additionally, shifting to this view means that rather than saying that the entire loop becomes electrified when $\overline{\pi}$ has a 4 cycle, we say that this occurs when $\langle\overline{\pi}\rangle$ forms a transitive subgroup of S_4.

This description is more robust in that it can handle any number of closures and gives us a picture of which wires are connected to which on a particular cable.

Distribution of Stops. Equipped with this mathematical framework, we are now ready to address Turing's model. We will classify stops by the cardinality and multiplicity of sets in the partition given by the orbits described above. For example, if a set of orbits is

$$\{\{\mathtt{a},\ldots,\mathtt{m}\},\{\mathtt{n},\ldots,\mathtt{z}\}\}$$

we would say this stop has a 13^2 stop type. In general, if the orbits of a subgroup are sets $\omega = \{\omega_1,\ldots,\omega_k\}$, we say that this represents a $1^{m_1}\ldots n^{m_n}$ stop, where m_i is the number of sets in ω with cardinality i – that is $m_i = |\{j \in \mathbb{N}_k \mid |\omega_j| = i\}|$.

This is effectively analogous to the cycle type of a permutation, but in order to extend this to multiple loops we must frame it in this description of partitions given by orbits of subgroups. Where context allows, we will refer to these interchangeably as either **orbit types**, **stop types**, or **partition types**.

In Turing's model, such a 13^2 stop type as above, would count as 0 normal stops over all steckering hypotheses. In the actual running of the Bombe this would constitute a single stop. Further, a $1^3 23^1$ stop type in Turing's model would be considered as 3 normal stops over all possible steckering hypotheses. In reality, this would only

constitute a single real stop in the running of the Bombe. This highlights the two qualms that we are trying to reconcile in Turing's model, the absence of abnormal stops, and the over-counting of normal stops.

We begin by considering the case of two closures, where at each rotor position of the Bombe, we have some permutations $\overline{\pi}$ and $\overline{\delta}$. What is the distribution of stop types? For now we will assume that any loop in a particular configuration of the Bombe represents a uniformly random permutation. We will later refine this, but as a heuristic it serves to justify Turing's model. We want to know what stop types are most common. That is, given two uniformly random permutations $\overline{\pi}$ and $\overline{\delta}$, what is the distribution of orbit types generated by the subgroup $\langle \overline{\pi}, \overline{\delta} \rangle$? This question was answered by John Dixon in 1969 in his paper "The Probability of Generating the Symmetric Group" [16].

Theorem 4.3 (Dixon's Theorem)**.** *Given two uniformly random permutations σ and τ in S_n, the probability that $\langle \sigma, \tau \rangle$ has an orbit type of $1^{m_1} \dots n^{m_n}$ is*

$$1 - \frac{1}{n!} \sum_{1^{\ell_1} \dots n^{\ell_n} \neq 1^{m_1} \dots n^{m_n}} \prod_{i=1}^{n} \frac{(i! t_i)^{\ell_i}}{\ell_i!}$$

where t_i is the probability that two uniformly random permutations in S_i form a transitive subgroup.

Proof. Let t_i be the probability that for two uniformly random $x, y \in S_i$, $\langle x, y \rangle$ acts transitively on $\mathbb{N}_i$.

Fix a partition $\omega = \{\omega_1, \dots, \omega_k\}$ of $\mathbb{N}_n$. We want to know the number of pairs $x, y \in S_n$ such that $\langle x, y \rangle$ have orbits which are exactly the sets in our partition. For $\langle x, y \rangle$ to contain an orbit ω_i, we must have that $\langle x, y \rangle$ acts transitively on ω_i. This

also means that the restriction of the action of $\langle x, y\rangle$ to ω_i is a transitive group action. The total number of ways to define such a restriction on ω_i, without any constraints on transitivity is $(|\omega_i|!)^2$. So, to get the number of restrictions which represent a transitive action we compute $(|\omega_i|!)^2 t_{|\omega_i|}$. To contain all orbits ω_i, we must have that when restricted to each ω_i, $\langle x, y\rangle$ acts transitively. Thus there are

$$\prod_{i=1}^{k}(|\omega_i|!)^2 t_{|\omega_i|}$$

permutations $x, y \in S_n$ such that $\langle x, y\rangle$ generates the orbits ω.

Supposing ω has partition type $1^{\ell_1} \dots n^{\ell_n}$, then our above argument can be framed as stating that there are

$$\prod_{i=1}^{n} ((i!)^2 t_i)^{\ell_i}$$

permutations $x, y \in S_n$ such that $\langle x, y\rangle$ has an orbit type equivalent to the partition type of ω.

To obtain the total number of such $x, y \in S_n$ that generate a subgroup with orbit type $1^{\ell_1} \dots n^{\ell_n}$, we multiply by the number of partitions of $\mathbb{N}_n$ with this type, of which there are

$$\frac{n!}{\prod_{i=1}^{n} (i!)^{\ell_i} \ell_i!}.$$ [1]

We then compute the probability that $x, y \in S_n$ do *not* generate orbits with orbit type $1^{m_1} \dots n^{m_n}$, by computing the total number of $x, y \in S_n$ which generate any other orbit type, and dividing by $(n!)^2$ (i.e. the total number of pairs of permutations

[1] This fact will be addressed later, but it follows from Corollary 1.6 with only minor argumentative changes.

in S_n). This is precisely,

$$\begin{aligned}
&\frac{1}{(n!)^2} \sum_{1^{\ell_1}\ldots n^{\ell_n} \neq 1^{m_1}\ldots n^{m_n}} \frac{n!}{\prod_{i=1}^{n} (i!)^{\ell_i} \ell_i!} \prod_{i=1}^{n} ((i!)^2 t_i)^{\ell_i} \\
= \; &\frac{1}{n!} \sum_{1^{\ell_1}\ldots n^{\ell_n} \neq 1^{m_1}\ldots n^{m_n}} \prod_{i=1}^{n} \frac{(i!)^{2\ell_i} t_i^{\ell_i}}{(i!)^{\ell_i} \ell_i!} \\
= \; &\frac{1}{n!} \sum_{1^{\ell_1}\ldots n^{\ell_n} \neq 1^{m_1}\ldots n^{m_n}} \prod_{i=1}^{n} \frac{(i!)^{\ell_i} t_i^{\ell_i}}{\ell_i!}.
\end{aligned}$$

The inverse of this probability is the probability that we generate exactly the orbit structure we specified, thus giving us the desired result,

$$\mathbb{P}(\langle \sigma, \tau \rangle \text{ has orbit type } 1^{m_1} \ldots n^{m_n}) = 1 - \frac{1}{n!} \sum_{1^{\ell_1}\ldots n^{\ell_n} \neq 1^{m_1}\ldots n^{m_n}} \prod_{i=1}^{n} \frac{(i! t_i)^{\ell_i}}{\ell_i!}.$$

□

It should be noted that the actual statement of this proof is a formula for t_n which is then shown to asymptotically approach 1 as n increases. In the context of the above theorem, t_n can be solved by computing the probability that $\langle \sigma, \tau \rangle$ has an orbit structure of n^1, that is, the subgroup formed is transitive. For such an orbit structure, each t_i given in our equation necessarily has $i < n$ so we can recursively solve for t_n. With the ability to solve for any t_k for $k \in \mathbb{N}$ we can then go and compute the probability of any particular orbit type via the above theorem.

4.1.2. Turing's Model's Accuracy

We can now compute, for the case of two loops, what the probability of a particular stop type is. Computing the above for a 26^1 orbit type tells us that roughly 95.9% of all rotor configurations in a two loop structure will electrify all wires, producing no stop whatsoever. The remaining 4.1% of rotor configurations account for all pos-

sible stops. Of these stops, we can compute that roughly 92.3% of these stops have a stop type of $1^1 25^1$. This is a *massive* proportion of the possible stops. Nearly all stops have this singular stop type, and herein lies the justification for Turing's method.

For the sake of argument, suppose *all* stops were of the type $1^1 25^1$. In this case, we note two observations:

(1) All stops have a steckering hypothesis producing a normal stop.

(2) The number of steckering hypothesis producing a normal stop for each stop type is 1.

It would then follow that if we could compute the number of normal stops over all steckering hypothesis we would get exactly the number of total stops. In such a case Turing's calculation of the expected number of normal stops over all steckering hypothesis is the same as the expected number of stops. As the proportion of stops whose stop type is $1^1 25^1$ increases, Turing's expected count becomes an increasingly accurate proxy for the actual number of stops. Given that 92.3% of stops have this stop type with just two closures, we can expect that Turing's estimate is very strong.

Moreover, as the number of closures increases, such an estimate improves. This is because for $\langle \sigma_1, \ldots, \sigma_k \rangle \leq S_{26}$, the expected size of orbits should increase as k increases. This can be shown heuristically via the Orbit-Stabilizer Theorem [4, p. 180] which tells us that,

Theorem 4.4 (Orbit-Stabilizer Theorem)**.** *For a group G acting on a finite set X, we have that for $x \in X$*

$$|G \cdot x| = \frac{|G|}{|G_x|}$$

where $G \cdot x$ is the orbit of x and G_x is the stabilizer of x.

We see that the size of the orbit of x is inversely proportional to the size of its stabilizer. As we increase the number of random permutations generating our subgroup, the constraints required for an element in $\langle \sigma_i \rangle$ to fix x become increasingly unlikely to be satisfied by independent random generators. Thus, we expect the average size of G_x to go down, and thus the average size of an orbit increases.

In our context, this implies that stop types tend to have larger components as the number of closures increases. Consequently, the stop type $1^1 25^1$—corresponding to a single isolated wire—becomes even more dominant among all stops. Thus, Turing's estimate should also improve as the number of generators, that is, the number of closures, increases.

Simulating Turing's Model. We will now show to what extent Turing's model aligns with ground-truth simulation. For testing during this thesis I created a simulation of both the Bombe and the Enigma machine. However, to ensure my implicit biases regarding my understanding of the Bombe's functions do not interfere with my analysis, simulated figures will be generated by Jean-François Bouchaudy's simulation of the Bombe [7]. Bouchaudy's simulation misses a number of stops that were detected by my simulation since, for poorer menus, it fails to propagate the input signal sufficiently to reach a steady state. If we increase the propagation in Bouchaudy's simulation to 100 loops per position of the Bombe, then my simulation falls in line with Bouchaudy's.

We can run Bouchaudy's simulation with randomly generated menu configurations and rotor choices, allowing us to compute the average number of stops along with a margin of error with 95% confidence.

For instance, running a Monte Carlo simulation of menus with three closures, we find that there are 30.26 ± 0.57 stops. Turing's estimate would give us 26^{4-3} or 26 stops which, though outside the margin of error, is still relatively close to the simulated ground-truth.

However, as we remove closures, we can expect that the distribution of stop types will diverge away from $1^1 25^1$ stops, and Turing's estimate will stray from the actual number of stops. This can be seen by running a menu with a single closure which produces 17003.50 ± 23.83 stops. Turing's estimate would give us 26^{4-1} or 17576 stops, which falls far outside the desired margin of error of simulation.

Turing's estimate is a strong heuristic model for the number of stops, but we can see that even with a large number of closures, his model is not accurate enough to fit within our desired margin of error. Were the Enigma machine to use more letters, or the number of closures to be significantly higher, we would expect Turing's model to fall within the margin of error; However, in the case of a 26 letter alphabet, with only a few closures, the number of abnormal stops and the number of stops with multiple steckering hypotheses producing normal stops are too high for the model to align with reality. In order to address these problems, we must develop a new model which does away with some of Turing's fundamental assumptions.

Section 4.2

The Cycle Type Model

The goal of Turing's model was to compute the expected number of normal stops over all steckering hypotheses. In the last section, we justified that for stronger menus,

this serves as a reasonable estimate for the true number of stops but still failed to capture the accuracy desired. The goal for this section is to produce a model which considers *all* stops, not just normal stops.

Further, in order to justify Turing's model, we made use of Dixon's theorem which showed that for two random permutations the probability they cause a $1^1 25^1$ stop type is sufficiently high to explain Turing's model's convergence to our simulation. In reality, the permutations formed by the scramblers on the Bombe are not random, but rather form a complex distribution that must be factored into our model to most accurately mimic the simulated behavior.

4.2.1. Single Loop

Consider a Bombe running a menu with a single closure. This closure is made up of some number of scramblers which we denote $\overline{\pi}_1, \ldots, \overline{\pi}_\ell$. In the loop, these form one permutation $\overline{\pi} = \overline{\pi}_1 \ldots \overline{\pi}_\ell$. The Bombe will stop whenever $\overline{\pi}$ fails to connect all wires in the Bombe. We have previously shown that this occurs whenever $\overline{\pi}$ is not a 26, or equivalently, whenever $\langle \overline{\pi} \rangle$ does not form a transitive subgroup of S_{26}.

We might expect that composing ℓ scramblers should produce a permutation $\overline{\pi}$ that is relatively random in S_{26}. If this were the case, we would expect $\overline{\pi}$ to be a 26 cycle with probability $\frac{1}{26}$ and thus we would expect a stop with probability $\frac{25}{26}$. Thus over 26^3 possible rotor positions, we would expect $26^2 \cdot 25 = 16900$ stops.

Running our simulation with $\ell = 5$ we find that there are 16224.77 ± 2.80 stops. This is significantly different from our prediction. Further, a simulation with $\ell = 4$ produces 17576 stops with no margin of error, meaning that *every* position on *every* menu with this structure produces a stop. Clearly, we are not dealing with permuta-

tions $\overline{\pi}$ that are uniformly random.

Parity. To make this clear, consider our case where $\ell = 4$. We have 4 permutations $\overline{\pi}_1, \ldots, \overline{\pi}_4$ which make up our loop $\overline{\pi}$. Each permutation $\overline{\pi}_i$ is a permutation representing an Enigma machine, meaning that it has a 2^{13} cycle type. This means that each permutation in our loop is made of 13 disjoint transpositions. In other words, each permutation $\overline{\pi}_i$ has an odd parity. Thus, the composition of 4 such permutations of odd parity must result in a permutation of even parity. Given that any 26 cycle is necessarily of odd parity, $\overline{\pi}$ can *never* have a cycle type of 26^1 and thus always produces a stop. In mathematical terms, we would say the distribution generated by composing 4 of these Enigma permutations is supported on A_{26} – the set of even permutations in S_{26}. In fact, this is true for any loop where ℓ is even.

Similarly, for any odd ℓ, $\overline{\pi}$ cannot produce any even parity cycle types, including our $1^1 25^1$ stop type that was so common in the two cycle case. In this case, the distribution is supported on odd permutations in $S_{26} \setminus A_{26}$.

The important takeaway is that we cannot just consider the number of closures in a menu – we must also consider the length of each loop within the closures of a menu. The length of a closure completely changes which set our distribution is supported on. Thus, our model will aim to express this by factoring in the additional variable of closure length.

Simulating the One-Loop Case. The distribution generated by composing permutations (in our case 2^{13} cycles) is a deeply complicated subject that is out of the scope of this thesis. The relationship between ℓ and the resulting distribution in S_{26} is so complex and vast in its possible enumeration that our best chance at getting a

reasonable grasp of this distribution is to simulate it.

The use of a simulation to get the expected number of stops for a single loop of length ℓ begs the following questions:

(1) If we can simulate the number of expected stops for one closure, why not just solve the entire problem of computing the expected number of stops via simulation?

While we might be able to run a lengthy Monte Carlo simulation of the Bombe over many different menu arrangements and shapes, the nature of the Bombe is so complicated that, in practice, there is no convergence to a steady state. Focusing on a single loop allows us to reach a distribution with a steady state fairly quickly. We can then rely on this distribution to compute more complicated menu arrangements which we will discuss in the following section.

In practice, this distribution was reached by running a Monte Carlo simulation. In each iteration, we initialize ℓ Enigma machines with the `pyenigma` package. Each had the same randomized rotor ordering and window setting, then each machine was given a randomized offset, and the resulting cycle type of their composition was collected. As we collect this distribution, we compute the total variation distance change between our distributions every 1000 iterations. If the total variation distance change between our distributions goes below 0.0001 for more than 30 comparisons, we deem that the distribution has reached a steady state.

Implicit in this simulation is the assumption that permutations produced by the Bombe are uniformly distributed within each cycle type. This is a reasonable assumption since all permutations for a given cycle type are the same up to relabeling.

Therefore, we expect no structural differences within the distribution for a fixed type.

Initially, we initialized ℓ permutations chosen randomly from the set of permutations with cycle type 2^{13} to perform our simulation. However, when we looked at the two closure case, certain edge cases, such as the case where one closure is of length 2, strayed from the simulated results. This is likely because the Enigma machines being represented in the Bombe all have the same initial configuration, only with different offsets, meaning that the permutations they represent are not purely independent 2^{13} permutations and there is some complicated underlying statistical relationship between the permutations in a loop.

(2) Would cryptanalysts at the time be able to simulate such a distribution?

While cryptanalysts at the time may not have been able to simulate as many iterations as we can with modern computers, we do know that such technology existed as would make this task reasonably achievable at the time. Recall from Section 2.6, that the Polish Cipher Bureau produced a device called the Cyclometer, whose sole purpose was to quickly determine the cycle type produced by composing two Enigma permutations. With simple changes, this could easily be extended to determine cycle types of arbitrary length loops of Enigma machines. With some random sampling and tedious cataloging, it is certainly feasible that cryptanalysts of the time could have computed a distribution close to the one discussed in this section.

One Loop Results. With both these points addressed, we can examine the results of simulating a single loop of length ℓ. After collecting the cycle type distribution for each length ℓ over a large Monte Carlo simulation, we can take the observed frequency of non-26^1 cycles to get the probability of a stop occurring. Multiplying this by 26^3 we get our expected number of stops. We then compare this with a small simulation

run via Bouchaudy's simulation as our ground-truth.

l	**Expected Stops**	**Simulated Stops**
2	17576.00	17576.00 ± 0.00
3	16246.63	16243.78 ± 6.91
4	17576.00	17576.00 ± 0.00
5	16224.88	16220.57 ± 6.66
6	17576.00	17576.00 ± 0.00
7	16222.69	16224.37 ± 6.63
8	17576.00	17576.00 ± 0.00
9	16223.75	16227.80 ± 6.92
10	17576.00	17576.00 ± 0.00
11	16224.53	16228.59 ± 6.95
12	17576.00	17576.00 ± 0.00

Table 4.1: Single-Loop Bombe Stop Probabilities for Loop Lengths $l = 2$ to $l = 12$

It is no surprise that these two values line up with each other, since they are both just simulated implementations of the Bombe; however, we will see that with just this small simulation, we are able to accurately estimate the stops for the case of two loops, without the need for any further simulation.

4.2.2. Two Loops

Consider a Bombe running a menu with two closures. These closures are made up of two sequences of scramblers we denote $\overline{\pi}_1, \ldots, \overline{\pi}_a$ and $\overline{\delta}_1, \ldots, \overline{\delta}_b$ – forming two loops with permutations $\overline{\pi}$ and $\overline{\delta}$. The Bombe will stop whenever these permutations fail to connect all wires in the Bombe. We have previously shown that this occurs whenever

$\langle \overline{\pi}, \overline{\delta} \rangle$ does not form a transitive subgroup of S_{26}.

To get the expected number of stops we need only know the probability that $\langle \overline{\pi}, \overline{\delta} \rangle$ form such a transitive subgroup. If $\overline{\pi}$ and $\overline{\delta}$ were uniformly distributed in S_{26} this question would be easily answered via Dixon's theorem; however, we just showed that these permutations are not evenly distributed and that their distributions are dependent on the loop lengths a and b. In the last section, we gathered the distribution of cycle types for each length ℓ, so we know the probability that a particular cycle type $1^{m_1} \dots 26^{m_{26}}$ occurs for a loop of length ℓ.

Therefore, we will approach this problem by finding the probability that two permutations pulled uniformly from two fixed cycle types form a transitive subgroup of S_{26} (or more generally S_n). In the argument that follows, we will construct a new generalization of Dixon's Theorem which will allow us to compute this probability by replacing the original strict requirement of uniformity, with a variable choice of distribution over cycle types.

We will first simplify our notation to make such a problem easier to state formally. Rather than denoting a cycle type as $1^{m_1} \dots n^{m_n}$, we will instead shorten this to the vector of multiplicities $m = (m_i)_{i \in \mathbb{N}_n}$, to mean that this permutation has m_i many i cycles. We will then define $t_{\alpha,\beta} = t_{(\alpha_i),(\beta_i)}$ as the probability that a random permutation with cycle type $\alpha = (\alpha_i)$, and a random permutation with cycle type $\beta = (\beta_i)$, form a transitive subgroup of S_n. This collapses our notations of cycle type into a single variable that will make notation significantly more condensed.

Assuming we have a means to compute these $t_{\alpha,\beta}$, we can then introduce our sin-

gle loop distribution from the last section to compute

$$\mathbb{P}(\overline{\pi} \text{ has cycle type } \alpha) \cdot \mathbb{P}(\overline{\delta} \text{ has cycle type } \beta) \cdot t_{\alpha,\beta}.$$

This is the probability that $\overline{\pi}$ has a fixed cycle type α and $\overline{\delta}$ has a fixed cycle type β, and that these two permutations form a transitive subgroup of S_n. To get the probability that any two $\overline{\pi}$ and $\overline{\delta}$ form a transitive subgroup, we can simply range over all possible cycle types in S_n to get

$$\begin{aligned} &\mathbb{P}(\langle \overline{\pi}, \overline{\delta} \rangle \text{ forms a transitive subgroup}) \\ =&\sum_{\alpha,\beta} \mathbb{P}(\overline{\pi} \text{ has cycle type } \alpha) \cdot \mathbb{P}(\overline{\delta} \text{ has cycle type } \beta) \cdot t_{\alpha,\beta}, \end{aligned}$$

and the complement of this probability will be the probability of a stop.

Then we need only to compute these $t_{\alpha,\beta}$ to solve the two loop case. In order to do this, we must generalize Dixon's Theorem to be expressed in terms of cycle types rather than the size of the ambient space S_n.

Computing $t_{\alpha,\beta}$**.** In order to solve this problem, just as in Dixon's Theorem, we will enumerate the various partition types of partitions of $\mathbb{N}_n$.

Let

$$\mathcal{I}_n \coloneqq \left\{ (\upsilon_1, \ldots, \upsilon_k) \in \mathbb{N}^k \,\middle|\, 1 \leq k \leq n,\ 1 \leq \upsilon_i,\ \upsilon_1 \leq \upsilon_2 \leq \cdots \leq \upsilon_k, \sum_{j=1}^{k} \upsilon_j = n \right\}$$

be the set of integer partitions of n. To align with our earlier multiplicity-based description of orbit, partition, and cycle types, we define a multiplicity function:

$$m^{(\cdot)} : \mathcal{I}_n \to \mathbb{N}^n$$
$$\upsilon = (\upsilon_j)_{j \in \mathbb{N}_k} \mapsto m^{(\upsilon)} = (m_i^{\upsilon})_{i \in \mathbb{N}_n}$$

where each $m_i^{(\upsilon)} \coloneqq |\{j \in \mathbb{N}_k \mid \upsilon_j = i\}|$.

We note that $\{m^{(\upsilon)} \mid \upsilon \in \mathcal{I}_n\}$ is exactly the set of partition types of partitions of $\mathbb{N}_n$ since the set partitions of $\mathbb{N}_n$ are surjective on to the integer partitions of n.

We showed in Corollary 1.6 that, for a particular cycle type $m = (m_i)_{i \in \mathbb{N}_n}$, the total number of permutations in S_n with such a cycle type is

$$\frac{n!}{\prod_{i=1}^{n} i^{m_i}(m_i!)}.$$

Similarly, for a particular partition type $m^{(\upsilon)} = (m_j^{(\upsilon)})_{j \in \mathbb{N}_n}$, the total number of partitions with partition type $m^{(\upsilon)}$ is

$$\frac{n!}{\prod_{i=1}^{n} (i!)^{(m_i^{(\upsilon)})}(m_i^{(\upsilon)}!)}.$$

We can arrive at this formula in a similar fashion to Corollary 1.6, but we must account for the fact that the order of each i cycle does not matter since we are treating them as sets, which gives us a factor of $i!$ in our formula as opposed to i.

To condense both of these factors, for a vector $x = (x_i)_{i \in \mathbb{N}_n}$, we introduce

$$Y(x) \coloneqq \prod_{i=1}^{n} (i!)^{x_i} \cdot x_i!, \qquad Z(x) \coloneqq \prod_{i=1}^{n} i^{x_i} \cdot x_i!.$$

Then with this notation, the total number of permutations with cycle type $c = (c_i)_{i \in \mathbb{N}_n}$ is

$$\frac{n!}{Z(c)}$$

and the total number of partitions with partition type $p = (p_i)_{i \in \mathbb{N}_n}$ is

$$\frac{n!}{Y(p)}$$

We will first illustrate the computation of $t_{\alpha,\beta}$ by example. Suppose we have two permutations σ and τ taken from a random distribution of permutations with cycle type $1^2 3^1$ (with corresponding α_is) and $2^1 3^1$ (with corresponding β_is) respectively in S_5. We want to find the probability that $\langle \sigma, \tau \rangle$ is a transitive subgroup of S_5.

The subgroup $\langle \sigma, \tau \rangle$ will form some set of orbits which partitions $\mathbb{N}_5$. Our goal is to find the probability that the orbit we generate is exactly $\{\mathbb{N}_n\}$. We will do this by enumerating all partition types which $\langle \sigma, \tau \rangle$ can form in its orbit structure.

In total, the are

$$\frac{(5!)^2}{Z(\alpha)Z(\beta)}$$

possible σ and τ combinations taken from their respective cycle types. This total must be equal to the sum, over each partition type of $\mathbb{N}_5$, of the number of ways that $\langle \sigma, \tau \rangle$ can produce an orbit structure mirroring that partition type.

The transitive case contributes exactly

$$\frac{(5!)^2}{Z(\alpha)Z(\beta)} \cdot t_{\alpha,\beta}$$

pairs $\sigma, \tau \in S_5$. Our ultimate goal will be to isolate this factor $t_{\alpha,\beta}$ to get our desired probability.

Now consider the case in which $\langle \sigma, \tau \rangle$ produces a set of orbits, say $\{A, B\}$ where $|A| = 3$ and $|B| = 2$ – that is, an orbit of type $2^1 3^1$. In order for this to occur, every cycle of σ and τ must be fully contained in either A or B since a cycle containing elements from both would necessarily mean that one set is reachable from the other, thus bridging the two sets into one orbit.

Thus, we must have that the cycle lengths of σ and τ be partitionable into two groups – one group of cycles whose total length is 3, and the other whose total length is 2. In our case, this can be achieved by splitting σ into a 3 cycle and 2 cycle, and splitting τ into a 3 cycle and two 1 cycles. The first group of cycles in S_3 can be denoted by cycle types $\alpha^{(3)} = (\alpha_i^{(3)})$ and $\beta^{(3)} = (\beta_i^{(3)})$. The number of pairs of cycles grouped in this way is

$$\frac{(3!)^2}{Z(\alpha^{(3)})Z(\beta^{(3)})}.$$

The second group of cycles in S_2 can be denoted by cycle types $\alpha^{(2)} = (\alpha_i^{(2)})$ and $\beta^{(2)} = (\beta_i^{(2)})$. The number of pairs of partitions grouped in this way is

$$\frac{(2!)^2}{Z(\alpha^{(2)})Z(\beta^{(2)})}.$$

We can then examine restrictions of σ and τ to each subset A and B and compute the probability that the restricted permutations generate a transitive subgroup on the

3 element set (that is $t_{\alpha^{(3)},\beta^{(3)}}$) and the probability that the restricted permutation generates a transitive subgroup on the 2 element set (that is $t_{\alpha^{(2)},\beta^{(2)}}$). Then the total number of pairs σ and τ with cycle types α and β which form orbits A and B is

$$\frac{(3!)^2}{Z(\alpha^{(3)})Z(\beta^{(3)})} \cdot \frac{(2!)^2}{Z(\alpha^{(2)})Z(\beta^{(2)})} \cdot t_{\alpha^{(3)},\beta^{(3)}} \cdot t_{\alpha^{(2)},\beta^{(2)}}.$$

If we wanted the total number of ways that σ and τ generate an orbit type of $2^1 3^1$, we must perform this computation over each possible partitions into sets $|A| = 3$ and $|B| = 2$, of which there are

$$\frac{5!}{(2!)(3!)}$$

total partitions. Since our count of σ and τ generating a fixed orbit with that structure was independent of the specific subsets within that orbit, we can simply multiply by the above factor to get the total σ and τ pairs generating orbits of the form $\{A, B\}$ with $|A| = 3$ and $|B| = 2$ as

$$\frac{5!}{(2!)(3!)} \cdot \frac{(3!)^2}{Z(\alpha^{(3)})Z(\beta^{(3)})} \cdot \frac{(2!)^2}{Z(\alpha^{(2)})Z(\beta^{(2)})} \cdot t_{\alpha^{(3)},\beta^{(3)}} \cdot t_{\alpha^{(2)},\beta^{(2)}}.$$

We note that, for a fixed cycle type $x = (x_i)_{i\in\mathbb{N}_n}$, and a partition type $\upsilon = \{\upsilon_1, \ldots, \upsilon_k\} \in \mathcal{I}_n$, the set

$$\mathcal{S}_{x,\upsilon} \coloneqq \left\{ \left(x_i^{(\upsilon_j)}\right)_{i\in\mathbb{N}_n,\, j=1,\ldots,k} \;\middle|\; \begin{array}{l} \text{For each } j, \sum_{i=1}^{n} i x_i^{(\upsilon_j)} = \upsilon_j, \\ \text{and for each } i, \sum_{j=1}^{k} x_i^{(\upsilon_j)} = x_i \end{array} \right\}.$$

is the set of groupings of the cycle type x into k buckets, such that the sum of the lengths of cycles in the jth bucket is exactly υ_j. Further, we require that for each $i \in \mathbb{N}$, $\sum_{j=1}^{k} x_i^{(\upsilon_j)} = x_i$ since we want to ensure that the total number of i cycles does not change between our original cycle type and this new grouping of cycles. This is

the set of all groupings of the cycle type x into distinct groups of cycles whose lengths correspond to the cardinalities of sets within a partition of $\mathbb{N}_n$.

In the example we just computed, we have that for $\upsilon = (2, 3)$ (i.e. our orbit has type $2^1 3^1$), $\alpha = (0, 1, 1, 0, 0)$ (i.e. σ has cycle type $2^1 3^1$) and $\beta = (2, 0, 1, 0, 0)$ (i.e. τ has cycle type $1^2 3^1$) , we have that

$$S_{\alpha,\upsilon} = \{((0, 1, 0, 0, 0), (0, 0, 1, 0, 0))\}$$

and

$$S_{\beta,\upsilon} = \{((2, 0, 0, 0, 0), (0, 0, 1, 0, 0))\}.$$

Note that if either $S_{\alpha,\upsilon}$ or $S_{\beta,\upsilon}$ were empty, then there is no way to partition both σ and τ's cycle types such that they can form orbits which are compatible with our partition type υ.

For our particular σ and τ, this is the case when we want to determine the number of pairs σ and τ generating an orbit of type $\rho = (1, 4) \in \mathcal{I}_n$. For this to occur, σ and τ must be partitionable into groups of cycles whose total lengths are 4 and 1. For τ this can be achieved by splitting the cycles into a 3 cycle and 1 cycle (totaling length 4), and a remaining 1 cycle (totaling length 1) For σ this is impossible to achieve since it clearly cannot be subdivided such that it has cycles totaling length 1. This is to say, for $\alpha = (0, 1, 1, 0, 0)$

$$S_{\alpha,\rho} = \emptyset.$$

We found that for a particular partition type υ, we have

$$\sum_{\alpha^{(\upsilon)} \in S_{\alpha,\upsilon}} \sum_{\beta^{(\upsilon)} \in S_{\beta,\upsilon}} \prod_{j=1}^{k} \frac{(\upsilon_j!)^2}{Z(\alpha^{(\upsilon_j)})Z(\beta^{(\upsilon_j)})} \cdot t_{\alpha^{(\upsilon_j)},\beta^{(\upsilon_j)}}$$

total ways in which σ and τ can produce this partition type in its orbits. For $\upsilon = (2,3) \in \mathcal{I}_5$, we find that there are 40 pairs of permutations in S_5 with cycle types $2^1 3^1$ and $1^2 3^1$ respectively that generate orbits with a set containing 3 elements and a set containing 2 elements.

If we enumerate over all partition types in $\mathcal{I}_5$, summing over the ways in which σ and τ can generate a set of orbits matching that partition type, we will get the total number of pairs of σ and τ. We have that

$$\frac{(5!)^2}{Z(\alpha)Z(\beta)} = \sum_{(\upsilon_1,\ldots,\upsilon_k) \in \mathcal{I}_5} \frac{5!}{Y(m^{(\upsilon)})} \sum_{\alpha^{(\upsilon_j)} \in S_{\alpha,\upsilon}} \sum_{\beta^{(\upsilon_j)} \in S_{\beta,\upsilon}} \prod_{j=1}^{k} \frac{(\upsilon_j!)^2}{Z(\alpha^{(\upsilon_j)})Z(\beta^{(\upsilon_j)})} \cdot t_{\alpha^{(\upsilon_j)},\beta^{(\upsilon_j)}}.$$

In general, for $n \in \mathbb{N}$, and two permutations σ and τ with cycle types α and β respectively, we have that

$$\frac{(n!)^2}{Z(\alpha)Z(\beta)} = \sum_{(\upsilon_1,\ldots,\upsilon_k) \in \mathcal{I}_n} \frac{n!}{Y(m^{(\upsilon)})} \sum_{\alpha^{(\upsilon_j)} \in S_{\alpha,\upsilon}} \sum_{\beta^{(\upsilon_j)} \in S_{\beta,\upsilon}} \prod_{j=1}^{k} \frac{(\upsilon_j!)^2}{Z(\alpha^{(\upsilon_j)})Z(\beta^{(\upsilon_j)})} \cdot t_{\alpha^{(\upsilon_j)},\beta^{(\upsilon_j)}}.$$

To solve for $t_{\alpha,\beta}$, we must isolate the partition $\{\mathbb{N}_n\}$, giving us

$$\begin{aligned}\frac{(n!)^2}{Z(\alpha)Z(\beta)} =& \frac{(n!)^2}{Z(\alpha)Z(\beta)} \cdot t_{\alpha,\beta} \\ &+ \sum_{(\upsilon_1,\ldots,\upsilon_k) \neq (n)} \frac{n!}{Y(m^{(\upsilon)})} \sum_{\alpha^{(\upsilon_j)} \in S_{\alpha,\upsilon}} \sum_{\beta^{(\upsilon_j)} \in S_{\beta,\upsilon}} \prod_{j=1}^{k} \frac{(\upsilon_j!)^2}{Z(\alpha^{(\upsilon_j)})Z(\beta^{(\upsilon_j)})} \cdot t_{\alpha^{(\upsilon_j)},\beta^{(\upsilon_j)}}.\end{aligned}$$

Solving for $t_{\alpha,\beta}$ we have

Theorem 4.5. *For $\sigma, \tau \in S_n$ with cycle types α and β respectively, we have*

$$t_{\alpha,\beta} = 1 - \frac{Z(\alpha)Z(\beta)}{(n!)^2} \sum_{(\upsilon_1,\ldots,\upsilon_k)\neq(n)} \frac{n!}{Y(m^{(\upsilon)})} \sum_{\alpha^{(\upsilon_j)}\in S_{\alpha,\upsilon}} \sum_{\beta^{(\upsilon_j)}\in S_{\beta,\upsilon}} \prod_{j=1}^{k} \frac{(\upsilon_j!)^2}{Z(\alpha^{(\upsilon_j)})Z(\beta^{(\upsilon_j)})} \cdot t_{\alpha^{(\upsilon_j)},\beta^{(\upsilon_j)}}.$$

This can be solved recursively since each factor $t_{\alpha^{(\upsilon_j)},\beta^{(\upsilon_j)}}$ is being computed for some $\upsilon \neq (n)$. Thus, each $\upsilon_j < n$ and we can compute the sub-problem for the case of permutations with cycle types $\alpha^{(\upsilon_j)}$ and $\beta^{(\upsilon_j)}$. Eventually, we will achieve the base case where $n = 1$, in which case transitivity is trivial.

In practice, we performed bottom-up recursion to get all possible values $t_{\alpha,\beta}$, for all pairs of cycle types α and β in $S_1, \ldots, S_{26}$ respectively. As a sanity check, we verified that for all cycle types α and β in S_n, that

$$t_n = \sum_{\alpha,\beta} \frac{1}{Z(\alpha)Z(\beta)} \cdot t_{\alpha,\beta}$$

where $n \in \{1, \ldots, 26\}$ and t_n is in reference to the original statement of Dixon's Theorem 4.3.

To the best of our knowledge, a general formula computing the probability that two permutations with fixed cycle types α and β generate a transitive subgroup of S_n has not appeared in academic literature. This contribution serves to extend Dixon's theorem to allow for cases in which $\sigma, \tau \in S_n$ are drawn from non-uniform distributions determined by their cycle types.

Two Loop Results. With these $t_{\alpha,\beta}$ in hand, we can now compute the probability that two Enigma permutations $\overline{\pi}$ and $\overline{\delta}$ do *not* form a transitive subgroup – thus producing a stop. By multiplying by 26^3 we find the expected number of stops.

What follows are two tables, the first represents the length of each closure in a two loop menu and the resulting number of expected stops as computed via our $t_{\alpha,\beta}$, the second is the number of stops observed via simulation.

	2	3	4	5	6	7	8	9	10	11	12
2	**781.69**	**749.40**	**750.42**	**751.24**	**750.81**	**750.08**	**750.78**	750.25	**750.05**	**750.24**	**750.13**
3		**704.13**	**705.75**	**706.53**	**706.12**	**705.39**	**706.07**	**705.55**	**705.35**	**705.56**	**705.40**
4			**707.42**	**708.20**	**707.79**	**707.05**	**707.74**	**707.21**	**707.02**	**707.22**	707.07
5				**708.98**	**708.57**	**707.83**	**708.51**	707.99	**707.80**	**708.00**	707.84
6					**708.16**	**707.42**	**708.11**	**707.59**	**707.39**	**707.59**	**707.44**
7						**706.68**	**707.37**	**706.85**	**706.65**	**706.85**	706.70
8							**708.05**	**707.53**	**707.33**	**707.54**	**707.38**
9								**707.01**	**706.81**	**707.02**	**706.86**
10									706.62	**706.82**	**706.67**
11										**707.02**	**706.87**
12											**706.72**

Table 4.2: Expected number of stops for two closures each ranging from length $\{2, \ldots, 12\}$. Values within the 95% margin of error of table 4.3 are noted in **bold**.

	2	3	4	5	6	7	8	9	10	11	12
2	781.08 ± 4.99	749.89 ± 3.58	752.43 ± 3.43	748.28 ± 3.46	749.09 ± 3.42	751.22 ± 3.47	749.06 ± 3.42	753.89 ± 3.58	749.75 ± 3.56	750.87 ± 3.49	750.22 ± 3.57
3		704.02 ± 1.73	705.49 ± 1.75	706.84 ± 1.69	706.50 ± 1.78	705.45 ± 1.68	705.42 ± 1.70	706.25 ± 1.69	706.54 ± 1.69	706.08 ± 1.62	705.98 ± 1.70
4			707.55 ± 1.63	707.93 ± 1.59	707.22 ± 1.60	706.95 ± 1.58	707.50 ± 1.66	708.45 ± 1.65	707.32 ± 1.63	707.85 ± 1.66	709.15 ± 1.62
5				708.55 ± 1.65	707.70 ± 1.61	708.97 ± 1.61	707.65 ± 1.65	706.33 ± 1.59	707.45 ± 1.66	707.84 ± 1.68	705.78 ± 1.57
6					706.69 ± 1.59	706.74 ± 1.61	706.55 ± 1.66	706.21 ± 1.60	707.41 ± 1.57	707.38 ± 1.62	707.28 ± 1.66
7						707.64 ± 1.61	708.29 ± 1.56	707.38 ± 1.60	708.17 ± 1.60	705.95 ± 1.58	708.60 ± 1.58
8							707.48 ± 1.63	707.15 ± 1.59	706.82 ± 1.66	707.60 ± 1.61	706.73 ± 1.61
9								706.91 ± 1.64	708.12 ± 1.64	706.56 ± 1.57	707.28 ± 1.60
10									708.33 ± 1.62	707.54 ± 1.58	707.88 ± 1.61
11										707.23 ± 1.63	707.18 ± 1.62
12											708.22 ± 1.60

Table 4.3: Simulated number of stops for two closures each ranging from length $\{2, \ldots, 12\}$.

4.2.3. Three or More Loops

Our generalization of Dixon's Theorem can be easily extended to work on an arbitrary number of permutations with arbitrary cycle types $\alpha_1, \ldots, \alpha_\ell$ where now we define $t_{\alpha_1,\ldots,\alpha_\ell}$ to be the probability that ℓ permutations, pulled uniformly from each of the cycle types $\alpha_1, \ldots, \alpha_\ell$, form a transitive subgroup of S_n. By similar derivation to Theorem 4.5 we find that

$$t_{\alpha_1,\ldots,\alpha_\ell} = 1 - \frac{\prod_{i=1}^{\ell} Z(\alpha_i)}{(n!)^\ell} \sum_{(\upsilon_1,\ldots,\upsilon_k)\neq(n)} \frac{n!}{Y(m^{(\upsilon)})} \sum_{\alpha_1^{(\upsilon_j)}\in\mathcal{S}_{\alpha_1,\upsilon},\ldots,\alpha_\ell^{(\upsilon_j)}\in\mathcal{S}_{\alpha_\ell,\upsilon}} \prod_{j=1}^{k} \frac{(\upsilon_j!)^\ell}{\prod_{i=1}^{\ell} Z(\alpha_i^{(\upsilon_j)})} \cdot t_{\alpha_1^{(\upsilon_j)},\ldots,\alpha_\ell^{(\upsilon_j)}}.$$

The problem with calculating the corresponding values of $t_{\alpha_1,\ldots,\alpha_\ell}$ is that of computational complexity.

While the implemented algorithm used many optimizations including caching lookups and pre-computing partitions, it is still sufficiently slow that the three loop case would likely require further optimization and more parallelization. Given the scale of combinatoric enumeration done to solve this problem, we used `python` to implement the algorithm. Certainly a language with less overhead like `C` or `Rust` could outpace our program. This examination of further closures presents an interesting research topic as it would illustrate whether or not the variance from the one loop estimates results in a degradation of the model for higher order loops.

5

Conclusion

Section 5.1

H-M Factor

This entire discussion of the expected number of stops has ignored the impact of the diagonal board. Once implemented, the diagonal board drastically reduced the number of stops for a run of the Bombe.

5.1.1. Turing's H-M Factor

To describe this effect, Turing introduced an additional term called the **H-M factor**, named after Cyril Holland-Martin the Technical Director of the British Tabulating Company which manufactured the Bombe [37, p. 116].

The H-M factor measures the proportion of normal stops over all steckering hypotheses that are maintained after introducing the diagonal connections. Given that the number of diagonal wires being used is dictated by the number of letters being considered in a menu, Turing gave separate H-M factors for menus containing various numbers of letters. We will denote the H-M factor for a menu with ℓ letters as $\mathcal{H}_\ell$.

Distinct Letters in Menu	$\mathcal{H}_\ell$
2	0.92
3	0.79
4	0.62
5	0.44
6	0.29
7	0.17
8	0.087
9	0.041
10	0.016
11	0.0060
12	0.0018
13	0.00045
14	0.000095
15	0.000016
16	0.0000023

Figure 5.1: The H-M factor as given by Turing in the *Prof's Book* [37, p. 116]

Supposing we had a menu with c closures containing ℓ distinct letters. Turing's model tells us that before the introduction of the diagonal board we expect 26^{4-c} normal stops over all steckering hypotheses. Given that $\mathcal{H}_\ell$ of these normal stops are maintain after introducing the diagonal board, Turing's model estimates that we expect

$$26^{4-c} \cdot \mathcal{H}_\ell$$

total normal stops over all steckering hypotheses when including the diagonal board.

The H-M factor has been discussed in academic literature regarding the Bombe, with researchers like John Wright [41] and Donald W. Davies [14] attempting to improve the accuracy of the H-M factor through either recursive formulae or explicit enumeration, respectively. Much of this research stems from the fact that Turing provides no formal justification for his computation of the H-M factor, stating simply that, "the method of constructing the [H-M factor] table is very tedious and uninteresting" [37, p. 116]. This understatement sparked a number of attempts to recreate, or improve upon, this "uninteresting" computation.

For the same reasons discussed in Section 4.1.2, this serves as a fair approximation of the actual number of stops since with more loops the proportion of stops with type $1^1 25^1$ increases. However, this also means that the calculation of $\mathcal{H}_\ell$ suffers from the inherent issues with Turing's model without the diagonal board – under-counting due to abnormal stops, and over-counting stops with multiple steckering hypotheses producing normal stops. As an example, Turing calculates that $\mathcal{H}_2 = 0.92$ [37, p. 116]. While this may serve as an accurate estimate of the number of normal stops over all steckering hypotheses, it has scarce relation to the actual number of stops. For instance, a menu with one closure and only two letters can never produce a stop – even with the introduction of the diagonal board.

5.1.2. Cycle-Type Based H-M Factor

As with our cycle-type model introduced in the last section, a proper calculation of $\mathcal{H}_\ell$ should not only take into account the single variable ℓ, it should also take into account an additional variable representing the partition type $\upsilon \in \mathcal{I}_n$ of a stop. Further, such an H-M factor would not give the proportion of normal stops over all steckering hypotheses that are maintained after the introduction of the diagonal board, it would give the proportion of *all* stops which are maintained. We will denote such a

hypothetical H-M factor as $\mathcal{H}_{\ell,\upsilon}$. While $\mathcal{H}_\ell$ is Turing's original scalar correction factor for menus of length ℓ, our refinement $\mathcal{H}_{\ell,\upsilon}$ introduces partition-aware correction accounting for the stop type of a stop.

Consider the case of two closures. We showed in the last section that we can compute the probability of a stop for two loops with cycle structures α and β as.

$$t_{\alpha,\beta} = 1 - \frac{Z(\alpha)Z(\beta)}{(n!)^2} \sum_{(\upsilon_1,\ldots,\upsilon_k)\neq(n)} \frac{n!}{Y(m^{(\upsilon)})} \sum_{\alpha^{(\upsilon_j)}\in S_{\alpha,\upsilon}} \sum_{\beta^{(\upsilon_j)}\in S_{\beta,\upsilon}} \prod_{j=1}^{k} \frac{(\upsilon_j!)^2}{Z(\alpha^{(\upsilon_j)})Z(\beta^{(\upsilon_j)})} \cdot t_{\alpha^{(\upsilon_j)},\beta^{(\upsilon_j)}}.$$

We can modify our version of Dixon's Theorem to give us the likelihood that two permutations $\sigma, \tau \in S_n$ with cycle type α and β are such that the orbits of $\langle \sigma, \tau \rangle$ have a particular partition type $\upsilon \in \mathcal{I}_n$. We denote this probability $t^{(\upsilon)}_{\alpha,\beta}$ and this is given as

$$t^{(\upsilon)}_{\alpha,\beta} = \frac{Z(\alpha)Z(\beta)}{(n!)^2} \frac{n!}{Y(m^{(\upsilon)})} \sum_{\alpha^{(\upsilon_j)}\in S_{\alpha,\upsilon}} \sum_{\beta^{(\upsilon_j)}\in S_{\beta,\upsilon}} \prod_{j=1}^{k} \frac{(\upsilon_j!)^2}{Z(\alpha^{(\upsilon_j)})Z(\beta^{(\upsilon_j)})} \cdot t_{\alpha^{(\upsilon_j)},\beta^{(\upsilon_j)}}.$$

We note that

$$t_{\alpha,\beta} = 1 - \sum_{\upsilon\neq(n)} t^{(\upsilon)}_{\alpha,\beta}.$$

Recall that the probability of a stop for loops of length a and b is

$$\begin{aligned}
& 1 - \sum_{\alpha,\beta} \mathbb{P}(\overline{\pi}_1 \dots \overline{\pi}_a \text{ has cycle type } \alpha) \cdot \mathbb{P}(\overline{\delta}_1 \dots \overline{\delta}_b \text{ has cycle type } \beta) \cdot t_{\alpha,\beta} \\
= \; & 1 - \sum_{\alpha,\beta} \mathbb{P}(\overline{\pi}_1 \dots \overline{\pi}_a \text{ has cycle type } \alpha) \cdot \mathbb{P}(\overline{\delta}_1 \dots \overline{\delta}_b \text{ has cycle type } \beta) \cdot (1 - \sum_{\upsilon \neq (n)} t^{(\upsilon)}_{\alpha,\beta}) \\
= \; & 1 - \sum_{\alpha,\beta} \mathbb{P}(\overline{\pi}_1 \dots \overline{\pi}_a \text{ has cycle type } \alpha) \cdot \mathbb{P}(\overline{\delta}_1 \dots \overline{\delta}_b \text{ has cycle type } \beta) \\
& + \sum_{\alpha,\beta} \sum_{\upsilon \neq (n)} \mathbb{P}(\overline{\pi}_1 \dots \overline{\pi}_a \text{ has cycle type } \alpha) \cdot \mathbb{P}(\overline{\delta}_1 \dots \overline{\delta}_b \text{ has cycle type } \beta) \cdot t^{(\upsilon)}_{\alpha,\beta} \\
= \; & \sum_{\alpha,\beta} \sum_{\upsilon \neq (n)} \mathbb{P}(\overline{\pi}_1 \dots \overline{\pi}_a \text{ has cycle type } \alpha) \cdot \mathbb{P}(\overline{\delta}_1 \dots \overline{\delta}_b \text{ has cycle type } \beta) \cdot t^{(\upsilon)}_{\alpha,\beta} \\
= \; & \sum_{\upsilon \neq (n)} \sum_{\alpha,\beta} \mathbb{P}(\overline{\pi}_1 \dots \overline{\pi}_a \text{ has cycle type } \alpha) \cdot \mathbb{P}(\overline{\delta}_1 \dots \overline{\delta}_b \text{ has cycle type } \beta) \cdot t^{(\upsilon)}_{\alpha,\beta}
\end{aligned}$$

This does not include our hypothetical H-M factor yet. Our factor $\mathcal{H}_{\ell,\upsilon}$, gives the proportion of stops of type υ that are maintained after the diagonal board is introduced. Then the total number of expected stops, with two closures with lengths a and b representing a menu of ℓ distinct letters (ℓ is not necessarily $a + b$ because the closures may share letters), including the diagonal board is,

$$\sum_{\upsilon \neq (n)} \mathcal{H}_{\ell,\upsilon} \sum_{\alpha,\beta} \mathbb{P}(\overline{\pi}_1 \dots \overline{\pi}_a \text{ has cycle type } \alpha) \cdot \mathbb{P}(\overline{\delta}_1 \dots \overline{\delta}_b \text{ has cycle type } \beta) \cdot t^{(\upsilon)}_{\alpha,\beta}.$$

While this provides a framework for understanding such a hypothetical factor $\mathcal{H}_{\ell,\upsilon}$ – we leave its explicit computation to future work.

In theory, $\mathcal{H}_{\ell,\upsilon}$ could be estimated via simulation; However, this would require that we are able to efficiently generate menus producing arbitrary orbit structures, many of which are extremely rare. This would likely result in high variance and poor convergence.

We suspect it is possible to perform an exact enumeration to compute $\mathcal{H}_{\ell,v}$, likely via similar methods used to compute $t_{\alpha,\beta}$, though we do not yet posses such a procedure. Developing a means to compute this factor would allow us to model the effective number of stops with high accuracy across nearly all Bombe configurations — with or without the diagonal board.

Section 5.2

Computational Methods

Many of the results in this thesis were obtained through simulations and computational analysis. To support reproducibility and encourage further exploration, this thesis is accompanied by a companion repository containing all relevant code. This repository can be found at https://github.com/JonahWeinbaum/building-a-bombe. This repository includes the scripts used for:

- Simulating the Bombe
- Simulating the Enigma
- Creating and using Zygalski Sheets
- Computing scoring, distance, dummyismus, and repeat sheets for Banburismus
- Capturing cycle distributions in the Bombe
- Computing probabilities $t_{\alpha,\beta}$
- Computing the expected number of stops in the Bombe for various menu arrangements

This repository is intended not only to allow readers to experiment with the methods described herein, but also to serve as a foundation for any future research building on this work.

Section 5.3

Future Work

This thesis provides several promising directions for exploration. We hope researchers can improve the estimation of stops in the Bombe by computing a cycle type based H-M factor as in Section 5.1.2. Further, we hope readers find novel use cases for the generalized Dixon's Theorem 4.5. This may serve to deepen investigations of transitivity in randomly composed permutations or, via reduction of the theorem statement, provide new results in other fields. In particular, we have begun investigating a relationship between the probability that a graph pulled from a uniform distribution over bipartite graphs is connected and the statement of the generalized Dixon's Theorem. Finally, through further optimization of the algorithm implementing the generalized Dixon's Theorem 4.5, researchers will be able to investigate the ways in which additional closures in a menu affect the number of stops the Bombe is expected to encounter.

Section 5.4

Conclusion

Throughout this thesis, we have described a battle of mathematics, a cat and mouse game between Allied cryptanalysts and Axis cryptographers attempting to thwart each other with a variety of cryptographic vulnerabilities and countermeasures. Our contribution to this story was to amalgamate and expand on the mathematical reasoning underlying this battle. We provide uniquely comprehensive and consistent

mathematical rigor meant to inform modern audiences about a key period in the history of computing and cryptography.

At the heart of this thesis is the belief that exploring subjects that may be considered dated or antiquated can still yield novel and modern insights. Learning about the problems that necessitated the creation of modern computing and understanding the thought processes of those who solved those problems can illuminate new perspectives and raise new questions in the field of computing. Examining this story provided us with a unique problem whose solution, in the form of the generalized Dixon's Theorem, offers a novel contribution to modern research in the field of abstract algebra. In analyzing and extending Turing's approach with modern tools, we not only gain greater understanding of the Bombe itself, but also of the mathematical structures that underlie its success.

This thesis serves to bridge a gap between historical cryptanalysis and contemporary mathematics. The stories we have examined are not relics of the past, but living lessons in perseverance and intellect which can shape our understanding of modern cryptanalysis and mathematics. The mathematical reasonings themselves are not bound by the time in which they were created; in contemporary reexamination, we find new avenues for exploration and discovery.

Bibliography

[1] 6812th Signal Security Detachment, U.S. Army. *Report on the British Bombe, 1944*. Archived by Tony Sale at Codes and Ciphers; accessed: 2025-08-15. 1944. URL: https://web.archive.org/web/20110807181352/http://www.codesandciphers.org.uk/documents/bmbrpt/bmbpg001.HTM.

[2] Hugh Alexander. *Cryptographic History of Work on the German Naval Enigma*. Digital facsimile. Declassified document, The National Archives, UK, Reference HW 25/1; accessed: 2025-08-15. Kew, Richmond, Surrey, TW9 4DU, c. 1945. URL: http://www.ellsbury.com/gne/gne-000.htm.

[3] ArnoldReinhold. *Enigma insides*. https://commons.wikimedia.org/wiki/File:Enigma_insides.agr.jpg. Uploaded to Wikimedia Commons on 20 September 2015; licensed under CC BY-SA 4.0; accessed: 2025-08-15. 2015.

[4] Michael Artin. *Algebra*. Boston: Pearson, 2011. ISBN: 9780130047632.

[5] Frank Benford. "The Law of Anomalous Numbers". In: *Proceedings of the American Philosophical Society* 78.4 (1938), pp. 551–572.

[6] Jean-François Bouch. *Key Rules*. Accessed: 2025-08-15. 2015. URL: http://www.jfbouch.fr/crypto/enigma/break/rules.html.

[7] Jean-François Bouchaudy. *Turing/Welchman Bombe Simulator (in C)*. Accessed: 2025-08-15. 2010. URL: http://www.jfbouch.fr/crypto/enigma/bombe/simu_c.html.

[8] Stephen Budiansky. *Battle of Wits: The Complete Story of Codebreaking in World War II*. New York: Free Press, 2000. ISBN: 9780684831305.

[9] B. Jack Copeland, ed. *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life*. Oxford University Press, 2004. ISBN: 9780198250807.

[10] Crypto Museum. *Enigma I*. Accessed: 2025-08-15. URL: https://www.cryptomuseum.com/crypto/enigma/i.

[11] Crypto Museum. *Enigma M4*. Accessed: 2025-08-15. 2025. URL: https://www.cryptomuseum.com/crypto/enigma/m4/.

[12] Crypto Museum. *Enigma wiring*. Accessed: 2025-08-15. URL: https://www.cryptomuseum.com/crypto/enigma/wiring.htm.

[13] Cryptography Stack Exchange contributors. *Banburismus and Bonus Scoring System*. Accessed: 2025-08-15. 2025. URL: https://crypto.stackexchange.com/questions/114964/banburismus-and-bonus-scoring-system.

[14] Donald W. Davies. "Effectiveness of the Diagonal Board". In: *Cryptologia* 34.3 (2010), pp. 229–239. DOI: 10.1080/0161-119991887865. URL: https://doi.org/10.1080/0161-119991887865.

[15] Harold C. Deutsch. *The Historical Impact of Revealing The Ultra Secret*. Unclassified Report DOCID: 3827029. Accessed: 2025-08-15. National Security Agency. URL: https://www.nsa.gov/portals/75/documents/news-features/declassified-documents/cryptologic-spectrum/ultra_secret.pdf.

[16] John D. Dixon. "The Probability of Generating the Symmetric Group". In: *Mathematische Zeitschrift* 110 (1969). See Theorem 2, pp. 199–205.

[17] David S. Dummit and Richard M. Foote. *Abstract Algebra*. 3rd. Wiley, 2004. ISBN: 9780471433347.

[18] Graham Ellsbury. *The Turing Bombe – Cribs and Menus*. Accessed: 2025-08-15. 1988. URL: http://www.ellsbury.com/bombe1.htm.

[19] William F. Friedman. *The Index of Coincidence*. Pamphlet. Declassified and approved for release by NSA on 03-10-2014; accessed: 2025-08-15. Fort Meade, MD, Mar. 1944. URL: https://www.nsa.gov/portals/75/documents/news-features/declassified-documents/friedman-documents/correspondence/FOLDER_152/41746979078617.pdf.

[20] William F. Friedman and Lambros D. Callimahos. *Military Cryptanalytics, Part I*. Accessed: 2025-08-15. Fort Meade, MD, 1952. URL: https://www.nsa.gov/portals/75/documents/news-features/declassified-documents/friedman-documents/publications/folder_242/41748189078749.pdf.

[21] Steven Hosgood. *All You Ever Wanted to Know About Banburismus but were Afraid to Ask*. Webpage, archived. Accessed: 2025-08-15. 2007. URL: https://web.archive.org/web/20160309155544/http://stoneship.org.uk/~steve/banburismus.html.

[22] David Kahn. *Seizing the Enigma: The Race to Break the German U-boat Codes, 1939-1943*. Houghton Mifflin Co., 1991. ISBN: 9780395518005.

[23] Maurice G. Kendall. *The Advanced Theory of Statistics, Volume 1*. 1st. London: Charles Griffin & Company Limited, 1948. ISBN: 9780852642429.

[24] Władysław Kozaczuk. *Enigma: How the German Machine Cipher Was Broken, and How It Was Read by the Allies in World War Two*. English; Polish. Frederick, Md.: University Publications of America, 1984. ISBN: 9780313270079.

[25] Bob Lord. *Enigma plugboard*. https://en.m.wikipedia.org/wiki/File:Enigma-plugboard.jpg. Uploaded to Wikimedia Commons on 16 February 2005; licensed under the GNU Free Documentation License and CC BY-SA 3.0. 2005.

[26] Bob Lord. *Enigma rotor flat contacts*. https://commons.wikimedia.org/wiki/File:Enigma-rotor-flat-contacts.jpg. Uploaded to Wikimedia Commons on 16 February 2005; licensed under the GNU Free Documentation License and CC BY-SA 3.0; accessed: 2025-08-15. 2005.

[27] Bob Lord. *Enigma rotor pin contacts*. https://commons.wikimedia.org/wiki/File:Enigma-rotor-pin-contacts.jpg. Uploaded to Wikimedia Commons on 16 February 2005; licensed under the GNU Free Documentation License and CC BY-SA 3.0; accessed: 2025-08-15. 2005.

[28] David Ng. *Enigma machine from World War II finds unlikely home in Beverly Hills*. Photograph of the German Enigma machine owned by David Bohnett; accessed: 2025-08-15. 2015. URL: https://www.latimes.com/entertainment/arts/culture/la-et-cm-imitation-game-enigma-machine-david-bohnett-20150122-story.html.

[29] Marian Rejewski. "An Application of the Theory of Permutations in Breaking the Enigma Cipher". In: *Applicationes Mathematicae* 16.4 (1980), pp. 543–559.

[30] Marian Rejewski. "How Polish Mathematicians Deciphered the Enigma". In: *Annals of the History of Computing* 3.3 (July 1981), pp. 213–234. ISSN: 0164-1239.

[31] Dirk Rijmenants. "Enigma Message Procedures Used by the Heer, Luftwaffe and Kriegsmarine". In: *Cryptologia* 34.4 (2010), pp. 329–339. DOI: 10.1080/

01611194.2010.486257. URL: https://doi.org/10.1080/01611194.2010.486257.

[32] Hugh Sebag-Montefiore. *Enigma: The Battle for the Code*. Weidenfeld Nicolson, 2000. ISBN: 9780297842514.

[33] Simon Singh. *The Code Book: The Secret History of Codes and Code-Breaking*. London: Fourth Estate, 1999. ISBN: 9781857028799.

[34] Christopher Smith. *The Hidden History of Bletchley Park: A Social and Organisational History, 1939–1945*. Palgrave Macmillan, 2015. ISBN: 9781349694891.

[35] Geoff Sullivan and Frode Weierud. "Breaking German Army Ciphers". In: *Cryptologia* 29.3 (2005). Accessed: 2025-08-15, pp. 193–232. DOI: 10.1080/01611190508951299. URL: https://www.tandfonline.com/doi/abs/10.1080/01611190508951299.

[36] Tony Sale. *Calculating the Zygalski Sheets*. Accessed: 2025-08-15. 2024. URL: https://www.codesandciphers.org.uk/virtualbp/poles/zyginfo.htm.

[37] Alan M. Turing. *The Prof's Book: Turing's Treatise on the Enigma*. English. Public domain; original English text. Public Domain (Creative Commons Public Domain Mark 1.0), 1940. ISBN: 9798372858404.

[38] Wapcaplet. *Enigma rotor exploded view*. https://en.wikipedia.org/wiki/File:Enigma_rotor_exploded_view.png. Uploaded to Wikimedia Commons on 11 June 2006; licensed under CC BY-SA 3.0; accessed: 2025-08-15. 2006.

[39] Jonah Weinbaum. *building-a-bombe*. GitHub repository. 2023. URL: https://github.com/JonahWeinbaum/building-a-bombe.

[40] Gordon Welchman. *The Hut Six Story: Breaking the Enigma Codes*. English. New York: McGraw-Hill, 1982. ISBN: 9780947712341.

[41] John Wright. “A recursive solution for Turing’s H-M factor”. In: *Cryptologia* 40.4 (2015), pp. 327–347. DOI: 10.1080/01611194.2015.1062318. URL: https://doi.org/10.1080/01611194.2015.1062318.