# The Enigma Machine

India Poetzscher

February 7, 2026

## 1 Introduction

The Enigma was a typewriter-esque cryptography device invented by German scientist Arthur Scherbius shortly after World War I. For many decades, the Enigma was considered the state-of-the-art method of encryption. As a result, Nazi Germany relieved heavily on the Enigma for internal communication throughout World War II [1].

Mathematically, it is useful to model the Enigma using symmetric groups, as we will explore. We will first define symmetric groups, then understand encryption as a proper involution, and finally, explain why this turned out to be the Enigma's fatal flaw, ultimately leading to its decryption by British codebreakers. In 1940, the Enigma was successfully decoded by Alan Turing and his team at Bletchley Park, and historians estimate that this work shorted the war by anywhere from two to four years [1][2].

## 2 The Enigma Machine

The Enigma machine consists of a keyboard, a plugboard, three rotors, a reflector, and a lampboard (see figure 1). To encrypt a letter, the operator types the plaintext letter on the keyboard, sending a signal that first travels through the plugboard, then to the back of the machine through rotors one, two, and three, before hitting the reflector. The signal then bounces off the reflector and travels in reverse through rotors three, two, and one, then through the plugboard again, and finally, the ciphertext letter gets lit up on the lampboard. This process is repeated for each letter of the plaintext message and the complete ciphertext is then sent via radio transmission.

The receiving party has an identical machine, set up in the exact same way (to be discussed). To decrypt, they simply type each letter of the received ciphertext onto the keyboard and watch as the plaintext appears on the lampboard.

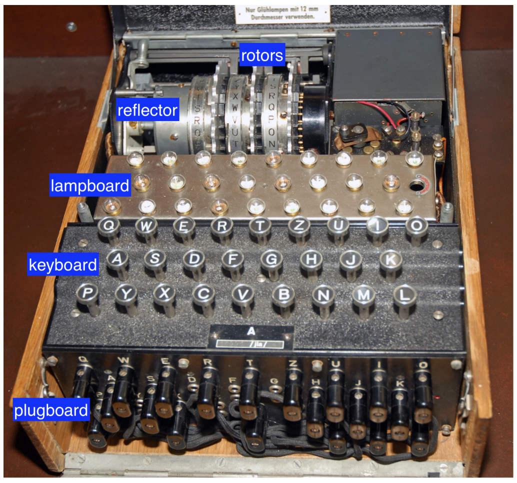

Figure 1: The components of the Enigma machine (with rotor cover removed).

## 3 Symmetric Groups

To analyze the Enigma, we must understand the notion of a symmetric group. First, recall that a group G is a set together with a binary operation, often denoted $\bigstar$, such that G is closed under $\bigstar$, and the associative, identity, and inverse laws hold [3]. As the name suggests, a symmetric group is a type of group. To understand what exactly it is, we must first introduce permutations.

**Definition 3.1.** A **permutation** is a bijection from some set X to itself.

**Example 3.2.** Let $X = \{1, 2, 3\}$. Then $f_1 : X \to X$ where all elements map to 1 is not a permutation, while $f_2 : X \to X$ defined by $f(1) = 2, f(2) = 3,$ and $f(3) = 1$ is a permutation.

**Definition 3.3.** Let $X$ be a set. The set of all permutations of $X$ together with the binary operation function composition is called a **symmetric group**, denoted $S_X$. Recall that the composition of two functions $f : A \to B$ and $g : B \to C$, written $g \circ f$, is defined by $(g \circ f)(x) = g(f(x))$.

**Theorem 3.4.** $S_X$ is a group.

*Proof.* We will show that $S_X$ satisfies all four group axioms. For closure, suppose $P_1, P_2 \in S_X$. We know that the composition of two permutations is also a permutation, so $(P_1 \circ P_2) \in S_X$.
To see associativity, suppose $P_1, P_2, P_3 \in S_X$ and $x \in X$. We will show $(P_1 \circ P_2) \circ P_3 = P_1 \circ (P_2 \circ P_3)$ by evaluating each expression at $x$ via repeatedly applying the definition of function composition. On the left, we have

$$((P_1 \circ P_2) \circ P_3)(x) = (P_1 \circ P_2)(P_3(x)) = (P_1(P_2(P_3(x))).$$

On the right, we have

$$(P_1 \circ (P_2 \circ P_3))(x) = P_1((P_2 \circ P_3)(x)) = (P_1(P_2(P_3(x))).$$

Since the left and right expressions are equivalent, associativity holds.
Next, let $e$ be our identity element defined by $e(x) = x$ for all $x \in X$. Then $e$ is a permutation of $X$, so $e \in S_X$. For all $P \in S_X$

$$(e \circ P)(x) = e(P(x)) = P(x) = P_1(e(x)) = (P \circ e)(x)$$

so the identity law holds.
Lastly, we show the existence of a unique inverse for each $P \in S_X$. Since $P \in S_X$, $P$ is a bijection, meaning it has a unique inverse $P^{-1}$. The inverse of a bijection is a bijection, so $P^{-1} \in S_X$. $\square$

Knowing that $S_X$ is a group will be critical to our analysis of the Enigma. Our focus will be on the symmetric group $S_{26}$, the set of all permutations of the set $\{1, 2, 3, ..., 26\}$, where we view each number from 1 to 26 in terms of its corresponding letter in the alphabet (i.e. $1 = A, 2 = B...$). For the remainder of this paper, let $X$ denote this set $\{1, 2, 3, ..., 26\}$.

## 4 Enigma as a Symmetric Group

Now we take a closer look into the Enigma's internal workings. Recall that the sequence of events for a letter leaving the keyboard is plugboard $\to$ 3 rotors $\to$ reflector $\to$ 3 rotors (in reverse order) $\to$ plugboard $\to$ lampboard.

### 4.1 Rotors

First, we will zoom in on the three rotors. Let $R_1$ be the right-most rotor, $R_2$ the middle rotor, and $R_3$ the left rotor. The electrical current leaves the plugboard and travels, from right to left, through each rotor in the order $R_1$, $R_2$, $R_3$. Each rotor has 26 contact points on the right (the incoming side) and 26 contacts on the left (the outgoing side). Inside the rotor is a mess of 26 wires that connects input and output contacts and acts as an "electrical implementation of a substitution cipher" [4]: the incoming letter gets mapped to some letter via the wiring connection, which becomes the input for the subsequent rotor. The physical set up of the wires ensures that each rotor $R_i : X \to X$ is a permutation, an element of $S_{26}$ [4].

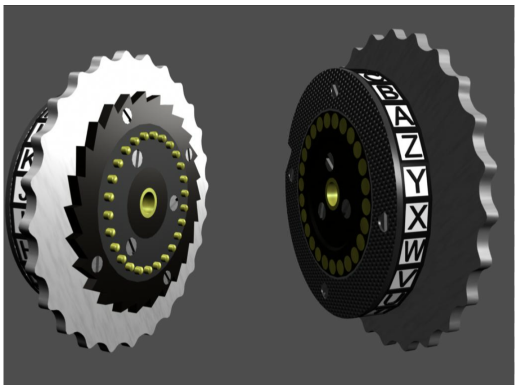

Figure 2: Two rotors and their contact points.

**Example 4.1.** Suppose $R_1$ maps the letter $A$ to $K$, $R_2$ maps $K$ to $Q$, and $R_3$ maps $Q$ to $J$. Then an $A$ enters the sequence of rotors and comes out as a $J$ the other end.

The crucial characteristic of the rotors is that they are *dynamic*—meaning they rotate. Specifically, each rotor can be in one of 26 positions, with its current position indicated by the letter sitting at the top of the rotor that peeks through the display window.

After a key is pressed on the keyboard, $R_1$ rotates one position, which changes its mapping. This means the same letter would now follow a *different path* through $R_1$. Specifically, that letter follows the subsequent letter in the alphabet's path. For example, an $A$ now follows $B$'s path, $B$ follows $C$'s path, and so on [5]. If previously $R_1$ mapped $A$ to $K$ and $B$ to $X$, it would now map $A$ to $X$. Essentially, $R_1$ becomes a different permutation, say $R_1'$. Each key press continues to shift $R_1$ forward one position and, by default, after $R_1$ has completed one full revolution returning to its initial position (after 26 key presses), it triggers the middle rotor $R_2$ to shift one position. Similarly, once $R_2$ has completed one revolution (after $26^2$ key presses), the left rotor $R_3$ shifts one position [6]. After $R_3$ has completed one revolution (after $26^3$ key presses), we are back to the initial state of the machine with all $A$'s displayed on top. Together, this means that the rotors encrypt each plaintext letter using a different permutation.

## 4.2 Reflector

After going through the three rotors, the signal hits the reflector, which is a crucial feature for our purposes. Recall that the same machine used for encryption is also used for decryption. The purpose of the reflector's design is to ensure this criteria is met and it achieves this by "pairing off" letters. The reflector $F : X \to X$ is then another permutation, as it maps letters to their "partner" and every letter has exactly one partner, which must be distinct from itself [7]. This idea will be explored in more depth later on.

**Example 4.2.** Suppose our reflector pairs $B$ with $F$. When a $B$ exits the third rotor and hits the reflector, it will then follow $F$'s path as it makes its way (in reverse) through the three rotors. Similarly, when an $F$ hits the reflector, it would now follow $B$'s path.

## 4.3 Plugboard

The plugboard is an additional layer of security that swaps the path of two letters right after being pressed on the keyboard and right before being displayed on the lampboard. The Germans had exactly 10 cables that each connected two letters together, leaving six letters untouched [8]. Therefore, the plugboard $P : X \to X$ is yet another permutation mapping letters to their "partner," but differing from the reflector $F$ in that there are six letters that don't get a "partner" and instead, remain unchanged.

**Example 4.3.** Suppose $A$ and $W$ are connected on the plugboard. Then, if you type an $A$ on the keyboard, you'll actually follow $W$'s path moving forward. Similarly, if you type a $W$, you'll actually follow $A$'s path. Further suppose that $T$ and $Q$ are connected on the plugboard. If that $W$-turned-$A$ turns into a $T$ after traveling through the rotors, the reflector, and reversing through the rotors, then the plugboard would convert that $T$ to a $Q$ and a $Q$ would light up on the lampboard.

The 10 plugboard cable connections could be changed at any time and were only known by the internal network. There are

$$\frac{26!}{6!10!2^{10}} = 150,738,274,937,250$$

possible ways to wire the plugboard [9]. The 26! comes from the ways of choosing from the set of 26 letters, but we must divide out 6! for the letters that are not involved. We also need to divide out 10! since even though 20 letters are involved, there are only 10 wires. Lastly, the ordering of a pairing does not matter ($T$ and $Q$ is the same as $Q$ and $T$), so we must divide by $2^{10}$. This massive number tremendously complicates the system.

## 4.4 Modeling Encryption

We are now ready to model the entire encryption process using our group operation.

**Theorem 4.4.** Let $R_i$ (for $i = 1, 2, 3$), $F$, and $P$, represent the rotor, reflector, and plugboard permutations, respectively. Then, encryption $E : X \to X$ (of a single letter) can be modeled as

$$(*)\ E = P^{-1} \circ R_1^{-1} \circ R_2^{-1} \circ R_3^{-1} \circ F \circ R_3 \circ R_2 \circ R_1 \circ P$$

Furthermore, we can simplify this as

$$(**)\ E = S^{-1} \circ F \circ S$$

where $S = R_3 \circ R_2 \circ R_1 \circ P$ represents the forward path [5].

*Proof.* The first five permutations in $(*)$ (recall function composition is applied right to left) directly correspond to the first half of the path that a letter takes through the Enigma. Now we turn to the second half (post-reflector $F$). The existence of the inverse permutations follows from the fact that permutations are bijective, and all bijections have a (unique) inverse. Left to prove is that these inverses *actually represent* the path that a letter takes on the back half of its journey through the Enigma. Consider a rotor in a fixed position. Initially, a letter enters the rotor from its right side and travels through the rotor's internal wiring exiting on the left (the forward direction, given by $R_i$). On its way back, however, it needs to enter the rotor from the left and exit on the right (backwards direction, which we claim is given by $R_i^{-1}$). When entering through the left contact $C_{i,L}$ for some $i \in \{1, 2, ....26\}$, it needs to exit the rotor through the contact point that $C_{i,L}$ is connected to via wiring $C_{i,R}$. This relationship is exactly given by the inverse permutation. If $R_i(a) = b$, then by definition of an inverse, $R_i^{-1}(b) = a$. For instance, say an $A$ traveled through $R_1$ becoming a $K$. Even though by the time that $K$ returns to $R_1$ (after going through the rest of the rotors and reflecting back) it is now a different letter, say $C$, to determine how $C$ will come out of $R_1$, we apply $R_1^{-1}$ to 3 ($C$'s position in the alphabet). The same applies to the plugboard permutation and its inverse. Thus, encryption of a single letter can be modeled as $(*)$. Finally, to simplify $(*)$ as $(**)$, recall that the inverse of a composition of bijections is each of their individual inverses composed in reverse order. $\square$

**Corollary 4.5.** Encryption $E : X \to X$ (of a single letter) is an element of $\mathrm{S}_{26}$ (a permutation).

*Proof.* Since each step in the encryption process (plugboard, rotor, reflector) is an element of $\mathrm{S}_{26}$ and because symmetric groups are closed under function composition, encryption is also an element of $\mathrm{S}_{26}$. $\square$

The above description of $E$ only models encryption for a single letter, or equivalently, one fixed set of rotor positions. Recall that the right rotor $R_1$ (and occasionally $R_2$ and $R_3$) steps forward after each key press, meaning $R_1$ changes to a new permutation $R_1'$. As we've touched on, this means the Enigma uses a different permutation when encrypting each letter, so encryption should be thought of as a *sequence* of elements in $S_{26}$, based on the initial settings of the machine. In other words, rather than a single permutation $E$ being able to capture the entire encryption of a plaintext message, we need to let $E_j$ denote the encryption of the $j$th letter after the initial state of the machine. Modeling $E_j$ looks like

$$E_j = P^{(j)-1} \circ R_1^{(j)-1} \circ R_2^{(j)-1} \circ R_3^{(j)-1} \circ F \circ R_3^{(j)} \circ R_2^{(j)} \circ R_1^{(j)} \circ P^{(j)}$$

where $R_i^{(j)}$ is the $i$th rotor permutation at time $j$ and $P^{(j)}$ is the plugboard permutation at time $j$ [5]. Then, encryption of a series of plaintext letters is the sequence: $E_1, E_2, ...E_n$ where $n$ is the number of letters of the plaintext. Thus, decryption now entails recovering an entire sequence of permutations rather than a single fixed permutation of the alphabet.

## 5 Complexity of Enigma

Let us now look at the complexity of the Enigma to analyze its security as a cryptography device. We know that the security of a cryptosystem relies substantially on how infeasible brute force is. Now that we understand the basic structure of Enigma, we'll explore all the ways it can be set up to help us understand the overall complexity of the machine.

So far, we know that the different rotor positions provide $26^3 = 17576$ states of our machine and the plugboard another 150 trillion or so. It turns out that the Germans actually had five different rotors to pick from. Of the five, they'd choose three to place into the machine and an order to place them in. Since there are five rotors to choose from for the first slot, then four for the second slot, and three for the third slot, that yields $5 \times 4 \times 3 = 60$ more options. In addition, there is an adjustable ring encasing each rotor that can be in one of 26 positions, which changes the way one rotor triggers the next rotor to rotate, meaning only the first two rotor rings are relevant. These rings add $26^2 = 676$ more possibilities [1]. All together, this totals

$$17576 \times 150,738,274,937,250 \times 60 \times 676 \approx 1.07458687 \times 10^{23}$$

possible settings of our machine, an incomprehensibly large number.

As mentioned, the three selected rotors can only be in one of $17,576$ possible positions—not a huge number, especially compared to the plugboard. Yet the addition of the dynamic rotors was critical and drastically increased the difficulty of decryption, not in terms of sheer number of permutations introduced but rather the means of deciphering. Without it, the Enigma would be, in essence, no different from a substitution cipher (like a Caesar cipher), which is vulnerable to frequency analysis. Now, decryption entails recovering an entire sequence of permutations, where the same letter will not necessarily encrypt the same way until after 17,576 key-presses—and no plaintext message contains that many letters!

Because of the massive and infeasible amount of work brute force would require, the Germans believed their machine was unbreakable. What would breaking Enigma even entail? Well, the Allies had their own Enigma machine—in fact Enigma machines were available to the public—but they needed their machine to be set up identically to the Germans' machines in order to decrypt their messages. Much of the German's set up remained constant throughout the war, but certain pieces changed on the daily. What remained fixed was the internal wirings of each rotor (the $R_i$ permutations for all five rotors), and with the help of Polish Cryptanalyst Rejewski's work, the Allies acquired this information [1]. What was left was the daily key: a sheet of paper sent to each German officer, which outlined the exact settings to put their machine on for that day before beginning decryption. This included information about which three of the five rotors to place into the machine and in which order, the starting positions of the three rotors, and the plugboard connections [10].

Note that with this information, anyone could set their machine up accordingly at the start of each message (by putting the chosen rotors in their described positions and wiring the plugboard) and decode all German messages for the day. So, the task of the cryptanalyst was to figure out the daily key. Since the daily key changed each day, ideally, you'd need a way to quickly and consistently figure out the daily key each morning. As we've calculated above, however, the combination of all of the Enigma's set-up options makes this an infeasible "guessing game."

## 6 Vulnerabilities and Cracking Enigma

Though brute force initially seemed infeasible, after modeling the Enigma as a symmetric group, code breakers identified a mathematical property which exposed a vulnerability—a design flaw of the machine that drastically reduced the search space needed for brute force. We now explore what that property was.

**Definition 6.1.** An **involution** is a function $f : A \to A$ which is its own inverse. That is, for all $x \in A, f(f(x)) = x$ or equivalently, $f^2 = e$ where $e$ is the identity map.

**Definition 6.2.** A **fixed point** of a function $f$ is an element $x$ in the domain that gets mapped to itself: $f(x) = x$.

**Definition 6.3.** A **proper involution** is an involution with no fixed points.

**Theorem 6.4.** The reflector permutation $F : X \to X$ is a proper involution [11].

*Proof.* Recall that $F$ is a bijection that "pairs off" elements, such that each element has exactly one partner (which is distinct from itself). In our set $X$, we have 13 pairings and $F$ maps each element to its partner, thus $F$ has no fixed points. To see that $F$ is an involution, we know that $F(x) = y$ for some $y \in X$ such that $F(y) = x$. Then $F(F(x)) = F(y) = x$. □

**Theorem 6.5.** For a fixed rotor configuration (no rotor stepping), encryption $E : X \to X$, given by

$$E = S^{-1} \circ F \circ S$$

is a proper involution.

*Proof.* Fix a rotor configuration. Then

$$\begin{aligned}
E^2 &= (S^{-1}FSS^{-1}FS) \\
&= S^{-1}F(SS^{-1})FS \text{ by associativity of function composition} \\
&= (S^{-1}FeFS) \text{ by definition of inverse} \\
&= (S^{-1}FFS) \text{ by definition of identity } e \\
&= (S^{-1}F^2S) \\
&= (S^{-1}eS) \text{ by Theorem 6.4} \\
&= (S^{-1}S) \text{ by definition of identity } e \\
&= e.
\end{aligned}$$

Since $E^2$ is the identity permutation, then $E(E(x)) = x$ for all $x \in X$.
To show that $E$ has no fixed points, suppose for a contradiction that $E(x) = x$ for some $x \in X$. Then

$$E(x) = S^{-1}(F(S(x)) = x$$

where the first equality holds by the definition of function composition and the second from our assumption. Applying $S$ to the right equality (and switching the order) we get

$$\begin{aligned}
S(x) &= S(S^{-1}(F(S(x))) \\
&= e(F(S(x))) \text{ by definition of inverse} \\
&= F(S(x)) \text{ by definition of identity } e
\end{aligned}$$

meaning $F$ maps $S(x)$ to itself, contradicting that $F$ has no fixed points. Thus, $E$ has no fixed points, so encryption at a fixed rotor position is a proper involution. □

**Corollary 6.6.** No letter is ever enciphered as itself [11].

*Proof.* Since encryption of a single letter is a proper involution, it has no fixed points, thus no letter is ever enciphered as itself. □

Why was this a vulnerability? Recall that the ultimate goal of the cryptanalyst is to guess the daily key, which consisted of the rotor ordering, the rotor starting positions, and the plugboard connections. The fact that no letter was ever encrypted as itself allowed for “crib-based” attacks: comparing portions of ciphertext with a known part of the plaintext, called a “crib.” Given a ciphertext, the cryptanalyst would start by guessing a word that they think will appear in that message (the crib).

**Example 6.7.** For instance, each day the Germans would send out a weather report at 6am which always followed the same format. We are pretty confident that the word “weather report”, “wetterbericht” in German, will appear somewhere in that message, making this our "crib" [12]. Since we know that no letter encrypts as itself, we can slide our crib back and forth under the ciphertext until that property holds true for all letters. For instance, the following is not a legal correspondence since we have an $E$ encrypting as an $E$

$$G\mathbf{E}QXLESQTPIXZ\ldots$$

$$W\mathbf{E}TTERBERICHT$$

but if we shifted our crib further we might get the perfectly legal match-up

$$\ldots MXLPISVBITTZH\ldots$$

$$WETTERBERICHT$$

which means "MXLPISVBITTZH" might be our ciphertext encoding of "WETTERBERICHT".

Since most messages were not that long, you could feasibly figure out the ciphertext encoding of your crib, or at worst, a few candidates. Once you have a crib and its ciphertext, we can work towards figuring out the plugboard combinations—the most important piece of the day key—using the "Bombe Machine."

The Bombe Machine was a huge machine built by Alan Turing and Gordon Welchman, aimed at figuring out the plugboard. From here, the rest of the daily key could be relatively-easily computed. At a high level, the Bombe Machine used the crib to speed up the process of guessing plugboard connections. The algorithm behind the Bombe Machine begins by making a guess about a single plugboard pairing and checks for contradictions based on that assumption [8].

Putting aside the Bombe Machine momentarily and taking our above example, suppose we guess that $M$ and $A$ are connected on the plugboard. Our crib tells us that $M$ (from "MXLPISVBITTZH") maps to $W$ (from "WETTERBERICHT") for some rotor position. So picking any initial rotor position, we begin to build up a chain of implications. We know

1. $M$ is encrypted as a $W$ (from our crib)
2. The plugboard turns $M$ into $A$ before entering the machine (our guess)
3. The rotors and reflector convert $A$ to a specific, known letter, say $K$ (known since the Allies had the rotor wirings)
4. $K$ goes through the plugboard and becomes a $W$ (from our crib)

Thus, we have just deduced another plugboard pairing, $K$ and $W$. We continue this process of deducing plugboard connections, and if at any point we reach a contradiction to our initial assumption (i.e. $M$ is connected to a letter other than $A$ on the plugboard), we know our initial guess was wrong, and we move on to the next guess—the next letter that $M$ could be connected to. If we go through all 26 possible connections (which includes the possibility that $M$ has no connection on the plugboard), then we step our right rotor forward one position [12]. If we don't reach a contradiction, we want to "halt" and add this plugboard connection to a list of valid possibilities to be checked later. Eventually, we will step through all 17,576 positions. Furthermore, we repeat this process for each of the 60 possible rotor orderings [8].

This process takes extremely long and seems like an infeasible amount of work, but Turing had two insights that drastically sped this process up. First, he realized that once you've found a contradiction, all other deductions made along the way must also be false, meaning these can all be thrown out, reducing the search space—they're all "fruit of a poison tree" [12]. Secondly, he realized you could create and evaluate this chain of implications instantaneously using electrical currents. Thus, the Bombe machine was born to do just that, and was able to go through all rotor positions and orderings in just 20 minutes, leaving a large, but feasible, list of possibilities to be checked by hand by human cryptanalyst [8]. From here, the cryptanalysts can figure out all the correct plugboard connections, compute the rest of the daily key, and begin decoding German messages.

## 7 Conclusion

The Enigma was regarded as one of the most advanced cryptosystems of its time, thought unbreakable by several of the world's most renowned code breakers. It was not until code breakers turned to group theory that its design flaws were identified and exploited. Group theory both beautifully models the Enigma and ultimately, was the key to cracking it.

## References

[1] Graham Ellsbury. The enigma machine: Its construction, operation and complexity. *ellsbury.com*, 2003.

[2] Jack Copeland. Alan turing: The codebreaker who saved 'millions of lives'. *BBC*, 2012.

[3] Jill Pipher Jeffrey Hoffstein and Joseph H. Silverman. *An Introduction to Mathematical Cryptography*. Springer, 2014.

[4] David Cash. Permutations and enigma. *https://people.cs.uchicago.edu/ davidcash/284-autumn-19/02-permutations-and-enigma.pdf*, 2019.

[5] Bill Casselman. Marian rejewski and the first break into enigma. *American Mathematical Society*, 2009.

[6] Eric Roberts and Jerry Cain. The enigma machine. *https://web.stanford.edu/class/cs106j/handouts/36-TheEnigmaMachine.pdf*, 2017.

[7] 101Computing. Code breaking during wwii. *https://www.101computing.net/enigma/enigma-instructions.html*, 2025.

[8] Kathleen Wang Angela Zou and Robby Huang. Bombe machine. *https://people.ece.cornell.edu/land/courses/ece5760/FinalProjects/s2022/* $az292_kw456_lh479/az292_kw456_lh479/index.html$, 2025.

[9] Numberphile. 158,962,555,217,826,360,000. *https://www.youtube.com/watch?v=G2*$_Q9FoD - oQ$, 2013.

[10] The National Museum of Computing. The turing-welchman bombe. *https://www.tnmoc.org/bombe*, 2005.

[11] Jeff Suzuki. The theorem that won the war: Part 2.4 – the message key. *Mathematical Association of America*, 2023.

[12] Numberphile. Flaw in the enigma code. *https://www.youtube.com/watch?v=V4V2bpZlqx8*, 2013.