Alan Turing and the cracking of the Enigma code

By Jan Brascamp. Version 3, January 2024

What were the inner workings of the Enigma machine that Germany used to encrypt its communications during World War Two, and what were the inner workings of the machine that Alan Turing designed to crack the Enigma cipher?

The Enigma machine

During World War Two the German forces used an electromechanical device to encode and decode their secret communications. The device, called the *Enigma*, had a keyboard with 26 keys, and 26 light bulbs above it (Figure 1). Each of the keys had a different letter of the alphabet printed on it, as did each of the bulbs. When you pressed one of the keys, one of the lights would come on, but the letter that lit up was not the same as the letter you pressed. Typing the message 'HELLO', for instance, might result in a sequence of lights that spelled 'PCUTM'. In other words, the machine could encode a typed message ('HELLO') into a ciphertext ('PCUTM'). One useful feature of the Enigma machine was that it could also apply the inverse operation: it could decode ciphertext back into the original message. This was important for German communication: one operator could encrypt a message using their Enigma machine and send the resulting ciphertext (e.g., 'PCUTM') safely over the airwaves, after which a second operator receiving that secret message could simply type it into their Enigma machine to have the original letters light up again (e.g., 'HELLO').

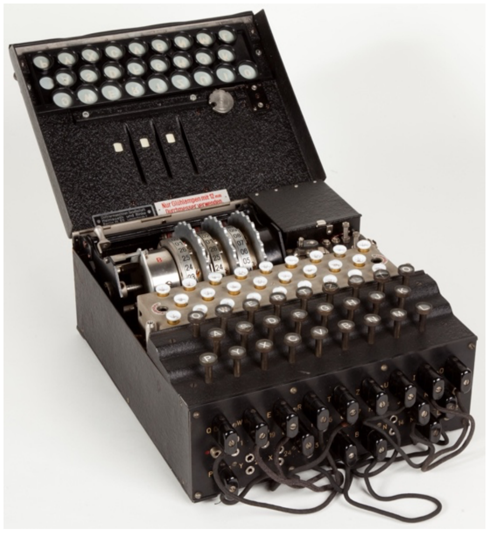
Figure 1. An Enigma Machine. Source: museo della Scienza e della Tecnologia "Leonardo da Vinci", CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons.

This procedure clearly had the benefit of efficiency, but just how safe it was for the Germans, depends on how hard it was for their enemies to crack an intercepted piece of ciphertext. In this context it is initially surprising that the coded text that Enigma machines produced, is basically what is called a *substitution cipher*. In substitution ciphers each letter in the original message is transposed to some other letter to produce the ciphertext. Substitution ciphers have been in use for many centuries, and in most cases it doesn't take much to crack them. One good starting point, when you get your hands on a long enough fragment of ciphertext, is to count how many distinct symbols are used in the fragment. If it turns out that this number is about 26, then each symbol in the ciphertext probably corresponds to a letter in the alphabet (plus perhaps some punctuation marks), and all you need to do is identify the correspondence. There are several ways you could go about this, and none of them are particularly complicated. For instance, you could tally up how often each possible symbol occurs in your sample of ciphertext. If the original text is in English, then the ciphertext symbol that occurs most often probably stands for the letter 'E'; the most common letter in English. From there you could move on to less common letters, and pretty soon you would be able to guess entire words like in a game of hangman, until you had identified the full correspondence between alphabet letters and ciphertext letters, and could reconstruct the original message.

Clearly, an ordinary substitution cipher is not sufficiently secure to be relied on for top-secret wartime communications. But Enigma's substitution cipher was not ordinary. For one thing, tallying up ciphertext symbols would get you nowhere in the case of Enigma. A closer look at the Enigma-encoded version of 'HELLO' in our first paragraph makes this instantly clear. The ciphertext in that example read 'PCUTM'. In an ordinary substitution cipher each alphabet letter corresponds to a specific ciphertext letter, yet in 'PCUTM' the two consecutive L's of 'HELLO' correspond to a sequence of *two different* symbols, U and T. How can this be?

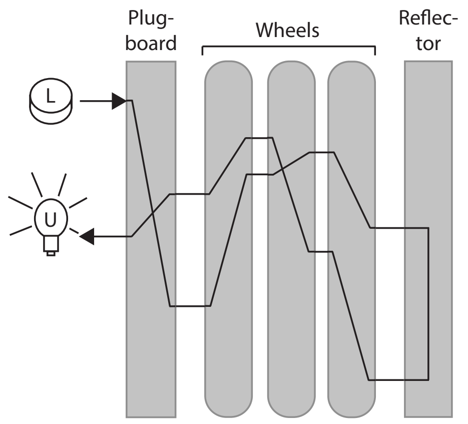

Figure 2. Basic organization of an Enigma machine.

To answer this question we need to look inside the Enigma machine, and follow the signal generated by a key press as it moves into the system (Figure 2). When one entered some letter by pressing its key on an Enigma machine, the first thing the letter's signal encountered on its way into the machine is the *plugboard*. This was an electrical circuit with on its outer side (the side connected to the machine's user interface) 26 contacts that we may label A through Z, and on its inner side another 26 contacts that we may label the same. In many cases a letter's signal simply passed through the plugboard unaltered, so that a signal entering at position L resulted in a signal coming out at position L. In other cases, however, the signal got rerouted so that an L routed to, say, a Q. In other words, the plugboard on its own implemented a basic substitution cipher. But in the Enigma machine this was just the beginning. After the plugboard, the signal (either rerouted or not) went into the first of three so-called *wheels*[1]. Like the plugboard, a wheel was also a circuit that mapped 26 possible inputs onto 26 possible outputs, and the first wheel's job was to route any letter that entered its circuit from the plugboard to some letter coming out. Moreover, the three wheels were mounted right next to each other, so that the letter that came out of one wheel formed the input to the next. All three wheels were basically the same, but each had a different wiring, so the wheels differed in terms of which input letters got rerouted to which output letters. Clearly, by the time a letter came out of the third wheel, it bore a convoluted relationship to the letter that was originally pressed on the keyboard: the original letter may have been converted to a different letter in the plugboard, and then got converted three more times as the signal passed through the wheels. But this does not conclude the scrambling process inside an Enigma machine, and when one pressed a key the letter that came out of the third wheel was usually not the letter that lit up on the outside of the machine. Instead, after the third wheel an Enigma machine contained a *reflector circuit*.

This circuit took the letter that emerged from the third wheel and routed it straight back into that same wheel, but at a different spot. There the letter started its journey back through the third wheel, through the second and through the first, with each passage meaning another rerouting. Then, after one final passage back through the plugboard, the signal finally reached the end of its circuitous journey through the machine, at a bulb on the outside that lit up to reveal the ciphertext letter that corresponded to the alphabet key that had been pressed.

But wait. Although the above description explains some of the Enigma's inner wiring, it does not explain how two L's in 'HELLO' could translate to two distinct symbols in Enigma ciphertext. If anything, the description suggests that Enigma code was simply a substitution cipher: that any L pressed on the keyboard should have made the same journey through the machine, no matter how circuitous, and ignite the same bulb. But this was not the case, and this is where the real ingenuity of the Enigma machine lies, and what made Enigma code so much harder to crack than many other substitution ciphers. The crux is this: each time the input generated by a key press had passed through the machine to light up a letter, the first of the three wheels rotated by 1/26th of a full cycle, so by one letter. This means that, the next time that same key was pressed, the resulting signal took a different route than it did the first time and – in all likelihood – ended up lighting up a different letter. When typing 'HELLO', in other words, the first and second L would pass through the plugboard in the same fashion but then their paths would diverge: the two would enter the first wheel at different spots that were one step removed from each other, and then come out of that wheel at spots that could be any number of steps from each other, depending on the wheel's wiring. From there on the routes taken by the two signals would remain separated, with the result that one press of the L key could make the U bulb light up, while the other could ignite the T.

The other two wheels were engaged in a stepwise rotation just like the first one, but they moved more slowly. Namely, each time the first wheel had taken 26 steps, thereby completing one full revolution, it pushed the second wheel up by 1/26th of *its* cycle. The third wheel moved 26 times slower again, taking one step for every revolution of the second wheel. This means that one had to press a key 26x26x26 times, or about eighteen thousand times, before the machine returned to a setting it had visited before. Put another way, if someone took a novel and typed it into an Enigma machine, they would be about 40 pages in before typing a letter that was encrypted using the same settings and, therefore, the same substitution alphabet, as the letter they began with. Clearly, any decryption attempt that assumed a consistent mapping between the letters in the ciphertext and those in the original message was bound to fail.

Enigma settings

This ever-changing encoding scheme made Enigma's ciphertext much harder to decode and importantly contributed to the Germans' confidence that they could freely transmit encoded messages over the airwaves. At the same time, it did not pose any problem to the German operators for whom the messages were intended. One reason is that these operators had their own Enigma machines, and the other is that they also had a top-secret codebook, distributed among all German communication stations. The book specified, for each given day, exactly how

German operators should configure their Enigma machines on that day. Recall that the Enigma machine was both an encoder and a decoder in one: a ciphertext produced on one Enigma could be typed into another Enigma to light up exactly the letters that were typed to produce the ciphertext to begin with. But this only worked if both machines were set to the same configuration: it would fail, for instance, if one machine's plugboard routed L to Q, while the other routed L to G, or if the two machines' wheel positions did not match. The secret book made sure that all settings did match among German Enigma machines on a given day, so that communications proceeded smoothly.

Among the settings specified in the codebook were the positions of the wheels: it is only a modest simplification[2] to say that all German operators used the same wheel positions at the start of each message sent or received on a given day. One can think of this as the codebook specifying which of the approximately eighteen thousand (26x26x26) encoding schemes applied to the first symbol in each message on that day, and automatically also which next scheme applied to the second symbol, to the third, etcetera. However, Enigma machines had several additional settings, to which the book also prescribed daily updates, and which further modified the encoding scheme. For one thing, an Enigma's wheels were removable, and each morning the operators used the book to decide which three wheels to place in the machine out of five uniquely wired wheels that were available, and in which left-to-right order. Information in the codebook further specified, for the first and second wheel, *at which point* in the 26-step revolution each of them should push its neighbor wheel one step up (determined by the so-called *ring settings*), as well as how the operator should wire up the plugboard that day. All in all, the codebook allowed the German operators to set their machines to one out of roughly a trillion times a trillion (10^{23}) configurations that were a priori possible. Recall that typing a piece of ciphertext into an Enigma machine would only produce the original message if the settings of the encoding machine and the decoding machine matched exactly, so this enormous number of possibilities formed a tremendous obstacle to those trying to interpret intercepted pieces of Enigma ciphertext.

The logic of Turing's approach: cribs, menus and loops

It was 1938, about a year before the war, when Alan Turing first became involved in the British effort of trying to crack the Enigma code. Before the British, the Polish secret service had been closely following the development of the German cipher, and had made several breakthroughs, including determining the internal wiring of each of the Enigma's wheels. By the time World War Two began in 1939, the British and French had received vital information from the Poles and had, in fact, each received a complete Enigma replica from them. Even that, however, did not allow Turing and his team to start decrypting Germany's communications. After all, without access to the Germans' secret codebook, the British still had to figure out which of 10^{23} possible Enigma configurations the Germans were using on a given day. What Turing and his colleagues are remembered for today, is their solution to this problem: a solution that enabled them to quickly reduce the humongous number of possible Enigma settings down to a small few that the Germans may have been using on a given day, thereby allowing that day's intercepted ciphertexts to be decrypted.

Although we established above that tallying up ciphertext symbols is not helpful for deciphering Enigma code, Turing's approach had one similarity with that approach, as well as with many other codebreaking techniques. This is the fact that it exploited *regularities* in the secret communications that were intercepted by the British: ways in which the communications deviate from randomness. In Turing's case, the starting point was formed by regularities introduced by the Germans' communication habits. In some cases, for instance, Turing could be confident that a sequence of 10 symbols somewhere toward the end of a piece of intercepted ciphertext corresponded to the German 'HEILHITLER'. Or, another example, he knew that every day around a certain time the Germans would broadcast a message that, in the original German, included the string 'WETTERFORHERSAGE' -- Wetterforhersage being the German word for weather forecast. Based on this type of knowledge, Turing and his team could highlight snippets of intercepted ciphertext and pair them up with the suspected German that they encoded. Such a piece of suspected German text was called a *crib*. Turing realized that a crib with its associated ciphertext said something about the configuration of the Enigma machine that produced the ciphertext. In very general terms this is obvious: imagine intercepting a ciphertext message that starts with 'OWWFOQE', and imagine knowing that sequence to stand for 'WETTERF'. This would tell you that the German operator who encrypted the message saw an O light up when they typed the first W into their machine, which rules out many configurations for that machine at that time.

A

Crib	W	E	T	T	E	R	F
Ciphertext	O	W	W	F	O	Q	E
Wheel position	X	X+1	X+2	X+3	X+4	X+5	X+6

B

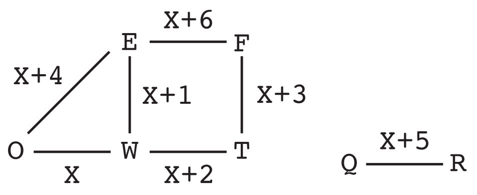

Figure 3. An example of a crib/cyphertext pair (panel A), and its associated menu (panel B).

That specific piece of information, however, would not be very useful. For one thing, the Enigma machines' ongoing wheel rotations ensured that a W-to-O mapping at the start of a message implied very little about what an O meant elsewhere in the ciphertext. More fundamentally, the amount of information provided by that single W-to-O mapping would be miniscule compared to the sheer number of possible Enigma settings that may have been used: above we saw that there are about 10^{23} possible Enigma settings total, and we may expect W to translate to O at about 1/26 of all those settings -- still an awful lot. In other words, when it comes to Turing's overall goal of quickly homing in on a small number of candidate Enigma settings, this specific piece of information would not get him very far. One of the important features of Turing's approach addressed this issue: rather than focusing on the crib-ciphertext mapping of any single letter in a pair like 'WETTERF'/'OWWFOQE', the approach managed to maximize information gain by considering the entire pair at once. One can think of this as Turing searching for settings at which it wasn't just the case that W translated to O, but at which it was *also* the case that a subsequent E -- encoded 1/26 rotations of the first wheel onward but otherwise at the same settings -- translated to W, *and* a T that followed yet 1/26 wheel rotations down translated to another W, etcetera. All seven requirements implied by this seven-letter crib/ciphertext pair are summarized in Figure 3A. The proportion of settings at which these requirements were *all* met, of course, would be much smaller than the proportion that met the W-to-O requirement on its own.

To understand Turing's approach, let's again consider the 'WETTERF'/'OWWFOQE' pair. Turing and his team summarized such pairs using schematics like the one shown in Figure 3B, called *menus*. This particular menu shows that there are two letters, W and O, that translate into each other at some wheel position labeled X. It also shows that E and W translate into each other one wheel step further, at position X+1; that T and W translate into each other at position X+2, etcetera until the end of the crib/ciphertext pair. There are two things to note about this menu. First, because the Enigma was both an encoding machine and a decoding machine the direction of the connecting lines does not matter: at position X a keyboard letter W translates to a lightbulb letter O (during encoding), but a keyboard letter O also translates to a lightbulb letter W (during decoding). The second thing to note is that this menu contains two of what are called *loops*. One can follow the first loop by tracing the triangle (clockwise) that runs from W to O at position X, then on to E at position X+4, and then back to W at position X+1 to complete the loop. The second loop is one step longer and runs from W, to E, to F, to T, and back to W.

Such loops were critical for Turing's approach. As mentioned above, the approach was designed to glean as much information as possible from a crib/ciphertext pair by considering all letter mappings in the pair at once. But it turns out that the amount of information contributed by a given letter mapping within the pair (or, in other words, the proportion of possible Enigma settings ruled out by that particular mapping) was especially large if that mapping closed a loop. For the following explanation of why that is, it is helpful to recall the distinction between the 26 keys and bulbs that connected to the outside of an Enigma's plugboard, and the 26 connectors on the plugboard's inner side that made contact with the first wheel. As mentioned above, sometimes the plugboard would simply link a given letter's key or bulb with the corresponding inner connector, so the W key, say, would activate the W connector. In other cases there would be a rerouting, so that the W's so-called *plugboard partner* would be a different letter. In our explanation we will use the prime symbol to denote a letter's plugboard partner, so W' indicates whichever inner connector is linked to the W key and bulb, irrespective of potential rerouting inside the plugboard (Figure 4). With this in mind a loop like Figure 3B's W-O-E-W loop shows that Turing would be looking for a wheel configuration X with the following properties.

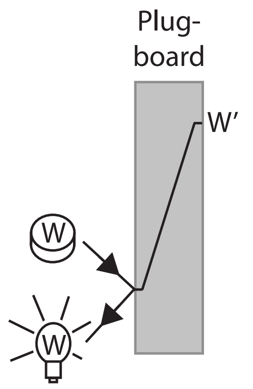

Figure 4. Illustration of terminology. The connector to which the plugboard routes the W key and W bulb, is designated the W' connector.

1. At configuration X, a signal at the W' connector results -- via the wheels and the reflector circuit -- in a signal at O'. Equivalently, a signal at O' results in a signal at W'.

2. Four rotation steps of the first wheel onward but otherwise at the same wheel configuration (i.e., at configuration X+4), that same O' connector now links to a third connector E'.

3. Three wheel steps before that latter position, but otherwise again at the same configuration (i.e., at configuration X+1), E' now links back to the same W' connector that we started with.

To see why the addition of property 3 is so important, let's consider what each property reveals about the wheels, if one does not know the plugboard mappings. More specifically, let's imagine trying each possible connector W' (i.e., each possible plugboard partner of the W key and bulb), and for each of these connectors evaluating how many candidate wheel settings can be ruled out based on these three properties. Property 1 on its own states that inner plugboard connector W' routes to a second inner plugboard connector, O'. Regardless of which inner plugboard connector one chooses for W', this first property on its own rules out no wheel configurations at all. After all, each inner plugboard connector routes to a second inner plugboard connector for all possible wheel settings, and if the plugboard mappings are unknown then this second connector may be O'. The addition of property 2 does rule out some wheel configurations, but not many. Properties 1 and 2 together state that W' routes to O' at setting X, and also that O' routes to a third connector, E', at setting X+4. As just pointed out in the context of property 1 on its own, no matter which wheel setting is X, any connector W' routes to a second inner plugboard connector. Similarly, no matter which wheel setting is X+4, this second connector O' also routes to some third inner plugboard connector. However, because this third connector may be any connector along the plugboard's inner surface other than O' itself, at about 1/25 of all possible wheel settings it is W': the connector that we started off with. Yet property 2 specifies that O' does not route straight back to W' at setting X+4: it routes to a different connector, E' (i.e., the plugboard partner of the E bulb and key; not the plugboard partner of the W bulb and key). In sum, for each possible connector W' that one may try, properties 1 and 2 together yield a contradiction for about 1/25 of all possible wheel settings, leaving only 24/25 as viable candidates. A similar logic shows why adding property 3, and thereby closing Figure 3B's W-O-E-W loop via X+1, is so informative. Let's say that, for some choice of connector W', one has restricted the possible wheel configurations X+1 to only those, for which properties 1 (at setting X) and 2 (at setting X+4) can both be true. For what proportion of those configurations can property 3 be true as well? In other words, for what proportion of possible settings X+1 does a signal to connector E' prompt a signal back at the original connector W'? Because E' can connect to any of the 25 connectors that are not E' itself, this proportion is only 1/25. In other words, when one closes the loop by adding property 3, the majority of wheel configurations can instantly be discarded for each given choice of connector W'.

Of course, even a reduction by 1/25 is not very large relative to the total number of a priori possibilities, but recall that Turing's approach made use of all letter mappings in a menu at the same time; not just of the ones that formed a single loop. The menu of Figure 3B has two different loops, corresponding to a larger reduction, and Turing worked with menus with more loops than that.

The implementation of Turing's approach: the Bombe

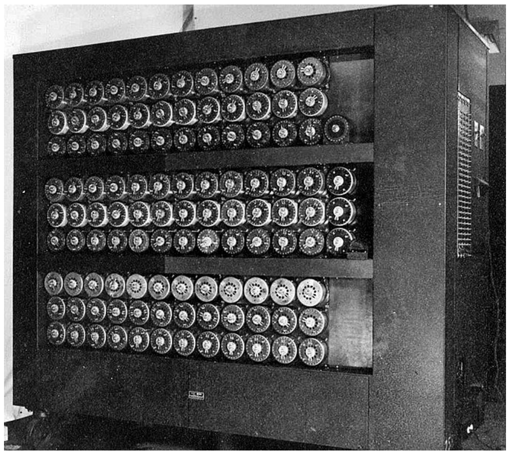
Figure 5. A Bombe. This image created by the United Kingdom Government is in the public domain.

The above illustrates that a sufficiently restrictive menu may help narrow down the number of possible Enigma wheel settings to a manageable few. But how could Turing's team do that in practice, and how could they do it quickly? That is where a machine called the Bombe comes in. A Bombe (Figure 5) was an electromechanical device, effectively a type of early computer, that Turing designed on the basis of earlier Polish machines, for the specific purpose of identifying viable Enigma settings based on menus like the one shown in Figure 3B. In other words, a Bombe could take the information contained in any menu that the British codebreakers constructed on the basis of a crib/ciphertext pair, and home in on those Enigma settings that could have produced that crib/ciphertext pair. To understand how a Bombe could do that, we need to look at its components and operation.

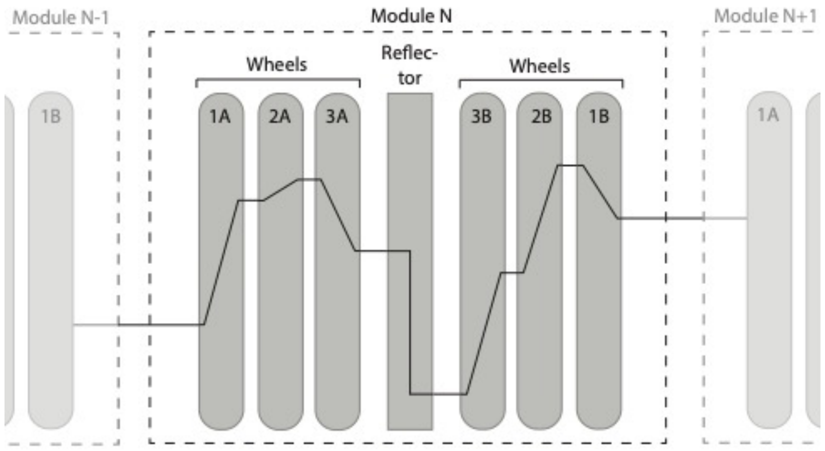

Figure 6. Schematic of a module inside a Bombe.

One can think of a Bombe as roughly a collection of Enigma machines, electrically connected end to end, so that an electrical signal coming out of one Enigma machine at a particular letter, entered the next Enigma machine at the same letter. More precisely, each module inside a Bombe mimicked the assembly of wheels and reflector circuit that could be found in an Enigma machine, but 'unfolded' by including two copies of each wheel (let's call them A and B), so that a signal could pass into the module via wheels 1A, 2A, and 3A to the reflector circuit, and then back out via wheels 3B, 2B, and 1B. Moreover, while in an Enigma machine the first wheel lined up with the 26 connectors that formed the inner side of the plugboard, in a Bombe module those same connectors were hooked up to the corresponding 26 connectors in the next module (Figure 6).

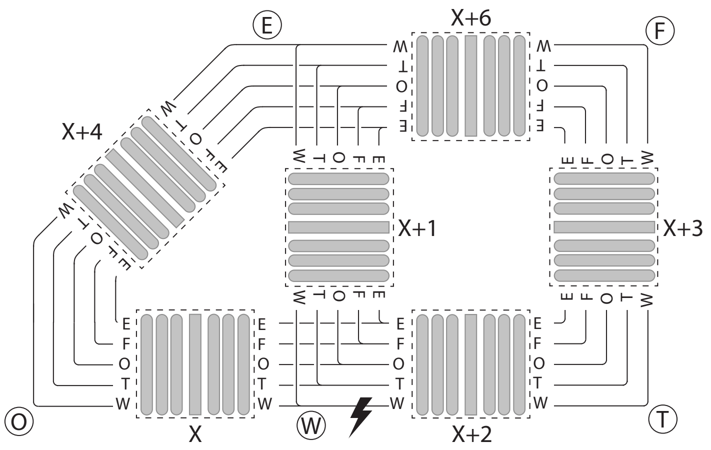

Figure 7. Schematic of a Bombe that has been set up according to the menu of Figure 3B.

Before setting a Bombe to work, Turing and his colleagues would connect its modules up according to a promising menu they had derived from an intercepted message. Figure 7 illustrates this for the menu of Figure 3. Each module in the Bombe's circuit takes the place of a line in the menu. As detailed below, just like each line connects two letters in the menu, each Bombe module was meant to mimic the encoding Enigma's wheel assembly at the moment when those two letters (one on the keyboard, the other on the lightboard) were translated into each other as the crib was being typed. Bundles of wires running between Bombe modules, in turn, take the place of the menu's letters. Accordingly, in Figure 7 each of these bundles is marked by a circled letter, denoting the Enigma interface letter that is at the corresponding menu location. The individual wires are also labeled (E, F, etc.), to denote how the wires correspond with the connectors on the inner surface of an Enigma plugboard. Bear in mind that this does not imply a correspondence with any particular interface letter: rerouting in the plugboard ensured that, say, inner connector E need not be inner connector E' (using Figure 4's terminology). To avoid clutter, in Figure 7 each bundle has not 26 but only five wires, which is all we will need for the explanations below.

Aside from the connections between the modules, each module individually would also be set up in accordance with the menu. Like the wheels in Enigma machines, those inside a Bombe module could rotate. While it was not important what specific positions Turing's team would set the modules' wheels to before starting the machine, it was critical that they set the modules’ wheel configurations correctly *relative to each other*: the number of wheel steps separating the individual modules' configurations should match those in the menu. In other words, the British could choose any wheel configuration for the module that is marked X in Figure 7, as long as the other modules were configured correctly relative to that (i.e., X+1, X+2, etc.) Maintaining these relative settings was important because, as mentioned, the purpose of the modules was to mimic the encoding Enigma's wheel settings as they were at various points in the process of typing a crib. And, while the absolute settings of this encoding machine were obviously not known, a menu did specify the relation between these settings at different points.

Once the modules inside a Bombe were set up according to a menu, 'pressing play' on the Bombe corresponded to applying a voltage to one of the 26 (5 in our example) wires that connected two modules. Critically, what happened next depended on whether the wheel configurations in the Bombe could explain the crib/ciphertext pair that was summarized by the menu or not. In other words, in our example it would distinguish whether the configuration of the module marked X could possibly match the Enigma wheel configuration when the initial W of ‘WETTERF’ was typed to make the O bulb light up. Before having a closer look at this, please note that Bombes featured no plugboard analog, and that we will initially pretend that Enigma machines had no plugboard either: we will act as if knowing which Enigma button was pressed or which light came on, was enough to know which of the 26 connectors carried a signal to or from the Enigma's first wheel. This simplifying assumption means that one could start a Bombe’s operation by applying a voltage to exactly the wire that corresponded to the first key of the crib that was used to configure the Bombe. Figure 7’s Bombe was set up according to the ‘WETTERF’/‘OWWFOQE' pair, so we will examine what happens in our example when a voltage (symbolized by a lightning bolt in the figure) is applied to the W wire inside the W bundle. Later we will take into account the Enigma plugboard, and examine how this changes things.

Figure 8A shows part of how this electrical signal propagates through the example Bombe's circuit if the wheel configuration of the module marked X happens to exactly match that of the Enigma that encrypted the initial W of the crib as an O. In that case the signal, after entering the X module via the W wire inside the W bundle, exits via the O wire inside the O bundle (still under the assumption that there is no plugboard). In the Bombe's circuit this O wire connects to the module marked X+4. One can apply the same reasoning to that module: if module X happens to be correctly configured, then module X+4 is also correctly configured: that module has the setting that applied when the Enigma encrypted the 5th letter of the crib, an E, as an O. This means that the signal would continue to the E wire inside the E bundle, as shown in Figure 8A. This E wire connects to two further Bombe modules, but Figure 8A focuses only on module X+1. Module X+1 corresponds to a position in the crib/ciphertext pair where an E translates to a W, so inside this module the E wire leads to the W wire. In fact, it leads to exactly the same W wire to which a voltage was applied to begin with. This is an important observation: in the scenario

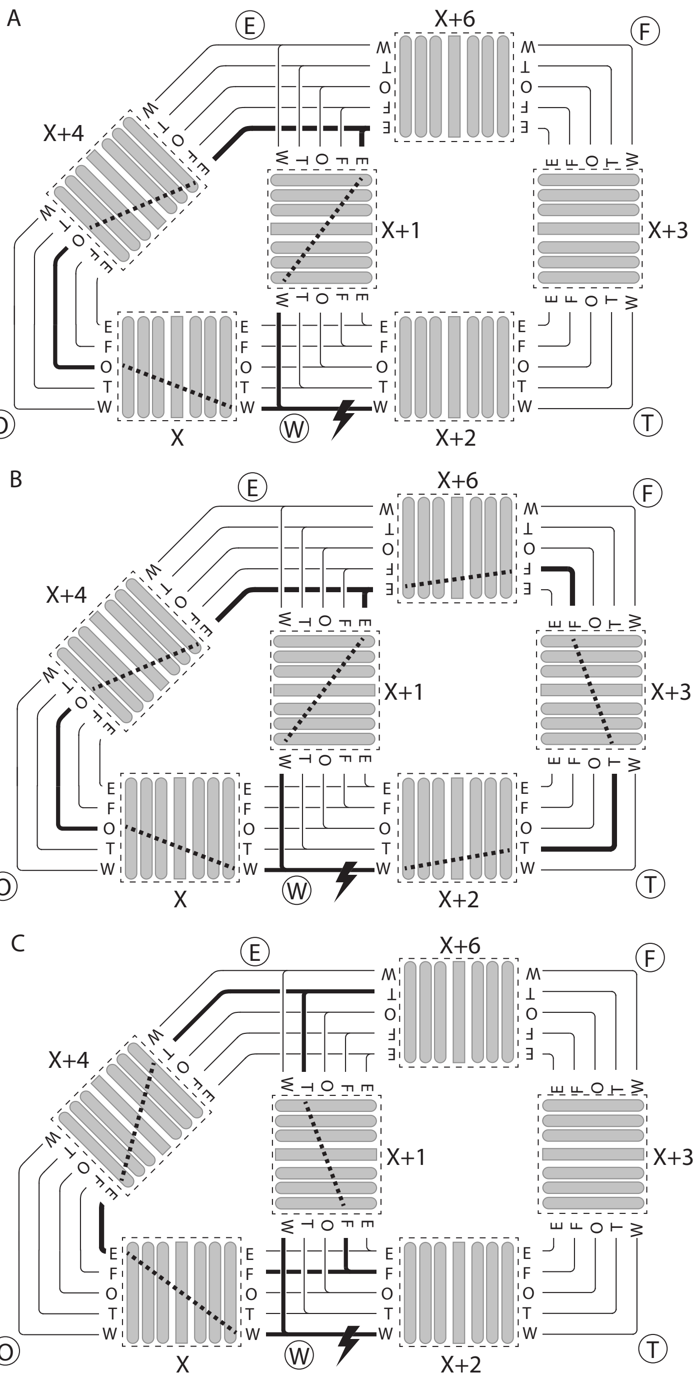

Figure 8. Illustration of signal propagation in the Bombe of Figure 7, in two different scenarios. Panels A and B: the Bombe's wheel settings match Enigma settings that can account for Figure 3A's crib/ciphertext pair. Panel C: they do not.

where the Bombe's wheel settings happen to be correct, there is a closed electric circuit that consistently involves only one connecting wire as we trace the loop along modules X, X+4, and X+1. Of course, Figure 8A ignores signal propagation in the second loop of this Bombe, but Figure 8B shows that the situation does not importantly change when that loop is included. The logic laid out above applies to the second loop, as well, so the modules and connecting wires inside that loop also form a closed electric circuit, and it remains true that only one connecting wire carries the signal from module to module anywhere in the Bombe.

The situation is very different in the scenario where the Bombe’s wheel settings do not match those of the Enigma that produced the ‘WETTERF’/‘OWWFOQE' crib/ciphertext pair. This is illustrated in Figure 8C. Again, a voltage is applied to the W wire of the W bundle, but now there is no guarantee that this will translate to a signal coming out at the O wire of the O bundle. Because the module can have any settings in this scenario, the signal is equally likely to exit at any of the wires. Let’s say it exits at the E wire and propagates from there to the X+4 module. Again, there is no way of telling where it will exit, and Figure 8C shows one option: the one where it exits on the T wire. The figure also shows the signal completing its lap via the X+1 module, which (in this example) it exits on the F wire inside the W bundle. Now we reach a critical observation: while in Figure 8A this was the end of the story, this time it is not. This time the signal does not come back on itself on the W wire inside the W bundle, thereby completing all signal propagation inside the X, X+4, X+1 loop. Instead, the signal returns to the X module via a new wire, the F wire, and thereby starts a new lap around this loop, along a different set of connecting wires. In other words, in contrast to the situation of Figure 8A, this time the signal does not remain restricted to a circuit of single wires that are electrically isolated from the others.

Of course, Figure 8C shows just one possibility, and there are countless other ones. Perhaps a signal that enters the X module at the W connection exits on the F wire rather than the E wire, etcetera. But regardless, the probability of the signal coming back on itself after one full lap around a loop is small: in our example it is 1/5 and in a real Bombe with all connecting wires it was 1/26. That probability increased slightly upon each further lap around a given loop, but it remained modest even after several laps (1/25 after two laps, 1/24 after three, etc.) Moreover, recall that useful menus had multiple loops: the one of Figure 3 has two, and the ones used by Turing’s team had more. Figure 8B illustrated that, when the Bombe’s settings happened to be correct, all loops converged right back onto the same wires, thereby keeping the signal restricted to that one, electrically isolated circuit. The chances of that happening at incorrect settings got smaller and smaller, the more loops the Bombe and its underlying menu contained. In fact, for sufficiently restrictive menus there was a high likelihood for incorrect settings to result in the signal propagating to every single connecting wire inside the Bombe.

In sum, after configuring a Bombe according to a menu derived from a crib/ciphertext pair, one could establish whether the Bombe's current wheel settings could account for the pair by applying a voltage to one wire inside the Bombe, and then measuring the voltages on all connecting wires between any given module pair. If only one of those wires carried the signal then the settings could account for the pair, and if all of them did then they could not. It is exactly this fact that was exploited in the design of Turing's Bombes. Recall that the wheels inside a Bombe could turn, just like the ones inside an Enigma did. The Bombes' design also featured a component that measured the voltages on all wires connecting a module pair, and this component fed into a set of electric motors that drove this wheel rotation. If this component measured that the signal was present on all wires, indicating that the current wheel settings were incorrect, then the wheel assemblies inside all modules would rotate simultaneously by a single step. In other words, then the Bombe would instantly proceed to the next possible wheel settings in order to test those. In this fashion the wheels would keep spinning, testing about 2 settings per second, or 7200 per hour, until the component measured a signal on only one connecting wire, thereby bringing the motors to a halt. At that point an operator could read the wheel settings inside the Bombe -- wheel settings that had a high likelihood of matching those used by the German operator who produced the intercepted ciphertext.

From obstacle to aid: the surprising role of the plugboard

The above paragraphs explained the basic principle of a Bombe's workings. But the explanation may have left you skeptical that Bombes were practically useful. For one thing, testing settings at a rate of 7200 per hour, although impressive, is not helpful if the total number of options is around 10^{23}: it would still take more time than the age of the universe to test them all. Additionally, the above explanation depended on the simplifying assumption that there was no plugboard. This made it possible to apply the initial voltage to a connecting wire (the W wire inside the W bundle) that corresponded to an element of the crib/ciphertext pair (the initial W of 'WETTERF'), and to then follow the resulting signal as it remained restricted to an electrically isolated circuit. Real Enigmas did have plugboards, so Turing and his colleagues could not know which Bombe wires corresponded to elements of the crib/ciphertext pair. If the W key at the start of a crib, for instance, routed to the F connection inside the Enigma plugboard, then the corresponding Bombe wire would be the F wire inside the W bundle. And then, as a final complicating factor, we should consider the ring settings. As briefly mentioned above, in an Enigma machine the ring settings determined at which position in its 26-step cycle the first wheel would nudge the second wheel one step up, and at which position in *its* 26-step cycle the second wheel would do the same to the third wheel. Importantly, these events could happen while a crib was being entered into an Enigma to produce the corresponding ciphertext, and this influenced whether different Bombe modules, for instance modules X and X+4 in Figure 8, should differ only in the positions of their first wheels, or also in those of the second and, perhaps, third.

The issue of the ring settings is related to a modification to Turing's original Bombe design that was introduced by Gordon Welshman, a mathematician and a wartime colleague of Turing's. This modification is called the *diagonal board*, and it will be discussed further down in this text. Before that, we will focus on the issues of the plugboard and of the sheer number of possible Enigma settings, which turn out to be related issues. Specifically, close consideration of the explanations surrounding Figure 8 reveals that plugboard settings were irrelevant to a Bombe's operation and could, in fact, be inferred along with the wheel settings after the Bombe had come to a halt. As a result, it is fair to say that a Bombe's only job was to find viable wheel settings, of which there were not nearly as many as Enigma settings in general.

To understand why the plugboard settings were irrelevant, consider a situation just like the one of Figure 8, except now the 'WETTERF'/'OWWFOQE' pair was produced on an Enigma with a plugboard that rerouted the signals passing between the Enigma's interface and its first wheel. As a result, when the initial W was typed into the Enigma, its signal did not enter the first wheel at the W connector on the plugboard's inner side: connector W was not connector W'. The question is how this would change our example Bombe’s behavior if we still, as in Figure 8, started its operation by applying a voltage to the W wire inside the W bundle. At incorrect wheel settings nothing important would change: the signal would still propagate throughout the Bombe's circuitry until it was present on all wires (if the menu was sufficiently restrictive). Things would change substantially, on the other hand, at correct wheel settings. Just as at incorrect wheel settings, there would now be no way of predicting where the signal would exit module X. After all, in this scenario the configuration of module X matches the wheel configuration of an Enigma in which the W key connects to the O bulb; not an Enigma in which the W connector on the plugboard’s inner side connects to any particular second connector. As a result, at correct wheel settings the signal would now start propagating across the Bombe’s wires, just like at incorrect settings. This seems worrisome, because then how can one distinguish correct wheel settings from incorrect ones in this scenario? The answer is that at correct settings signal propagation would be importantly restricted: there would be one circuit of single connecting wires that the propagating signal could not reach. If that seems surprising, then consider this. The reason that in Figures 8A-B the signal remained restricted to a circuit of single connecting wires, is that this circuit was electrically isolated from the other wires. The presence of a plugboard does not change the fact that such an isolated circuit exists inside our Bombe at correct wheel settings: it is still there, formed by those wires that correspond to the connectors on the plugboard’s inner surface that were activated when the crib was typed to produce the ciphertext (i.e., connectors W' and O' when the crib's initial W was encrypted as an O, etcetera). The only thing the plugboard does, is make it impossible to know ahead of time which wires those are, and to deliberately apply the initial voltage to one of them.

The upshot is that, relative to our Bombe's workings as laid out above, all that needs to change to accommodate the plugboard is the criterion that stops the wheels from turning: the wheels of a real Bombe stopped whenever a signal was measured on only one wire connecting two modules, but also when a signal was measured on all wires *but* one. Only if all wires carried the signal would the wheels keep spinning. Moreover, even though Bombes were designed, then, to crunch through candidate wheel settings exclusively, a Bombe’s state after it had halted also provided information on plugboard settings. Just like an operator could examine the modules of a halted Bombe to find viable wheel setting, they could find associated plugboard settings by measuring the voltages on connecting wires. For instance, imagine the Bombe of Figure 8 halting with a signal present on all wires inside bundle W except the F wire. This would mean that the F wire forms this section of the isolated circuit (rather than the W wire, as in Figures 8A-B), so it would point to a plugboard that routes the W key and bulb to the F connector. By also measuring the wires between other module pairs, the plugboard routings of the crib/ciphertext pair's remaining letters could be inferred in the same way.

In sum, not only did the plugboard do nothing to prevent Turing and his team from finding candidate wheel configurations; finding candidate wheel configurations is all they needed to do, because associated plugboard settings could be identified in the process. This greatly reduced the number of possible options to search through: from 10^{23} down to about one million (taking into account that each Enigma wheel had 26 possible starting positions, and that the German codebook also specified which three wheels to use to begin with, out of five available). With several Bombes working in parallel on the same menu and each of them crunching through about 7200 wheel settings per hour, therefore, Turing and his team had a solid chance of identifying a given day's Enigma settings. For example, eight Bombes working at the same time could examine roughly half of the possibilities within 10 hours.

What has not yet been considered in the above analysis is the ring settings: the settings that determined at which point in its cycle the first Enigma wheel would nudge up the second wheel by one step, and at which point the second wheel would nudge the third. A critical concept above was that of Bombe modules being displaced relative to each other by a given number of wheel steps. What was meant, for instance by a statement like 'one module has wheel configuration X, and the other has wheel configuration X+4', was that the modules' configurations differed by four steps of the first wheel, in an attempt to match the state difference within the encoding Enigma machine between the moments that it encrypted the crib's first letter and fifth letter. In reality, however, that state difference may have consisted of four steps of the first wheel as well as one step of the second (if the first wheel nudged the second one during this initial part of the crib). Or, in rare cases, four steps of the first, one step of the second, and also one step of the third. In short, the ring settings multiplied the one million or so possible settings that one might want to consider, and this multiplication (by a factor of 676) would seem to put a large dent in Turing's chances of finding the right settings in time.

One potential strategy to avoid this concern, was to use a relatively short crib, and to assume that it involved no movement of the encoding Enigma's second or third wheel. For a crib of only two letters, for instance, the chances of its creation having involved any movement of the Enigma's second wheel were only 1/26, and for the third wheel those chances were smaller still. A related but better approach, with a longer crib in hand, was to cut it into two consecutive fragments of 13 letters each, and to assign each fragment to a different group of Bombes. Because the second wheel moved only once for every 26 key presses, one could be certain that one of the fragments involved movement of only the first wheel, so only that first wheel would have to be considered in the Bombes' operation. But there was a drawback to the British using such approaches, and that was that shorter cribs were less restrictive: there were more possible Enigma settings that could account for a given crib/ciphertext pair if it was short, so for shorter cribs the Bombes would come up with more false alarms.

This dilemma limited the effectiveness of the British codebreaking effort until, as briefly mentioned above, Gordon Welshman proposed his extension to the Bombe design; an extension called the diagonal board. In essence, the diagonal board's value lay in the fact that it allowed more information to be squeezed from a given crib, thereby increasing how restrictive that crib was in terms of the number of Enigma settings that it ruled out. This, in turn, allowed the British to use shorter cribs, thereby mitigating the issues raised by the ring settings.

Intriguingly, what allowed the diagonal board to work was the plugboard, a component meant to thwart codebreaking efforts. Above we saw that the plugboard posed no obstacle in a Bombe's search for suitable wheel settings. With the addition of the diagonal board, the plugboard even made this search easier. The reason is a particular design feature of the plugboard. Namely, if the plugboard was set up such, that a given key and light bulb, say A, were routed to a different letter's connector on the plugboard's inner side, say B, then this would automatically also route the B key and bulb to the inner connector of letter A. It is not hard to see why the plugboard may have been designed this way: it avoided the possibility of multiple keys and bulbs leading to the same connector on the plugboard's inner surface, which would cause ambiguity when encoding or decoding a message. But the design feature also introduced into crib/ciphertext pairs something that was identified above as a foothold for codebreakers: deviation from randomness. In particular, if one somehow knew that key and bulb A routed to inner connector B, then this automatically ruled out all but one possible plugboard setting for key and bulb B. In their original design, Bombes did not use this information, but with the addition of Welshman's diagonal board they did.

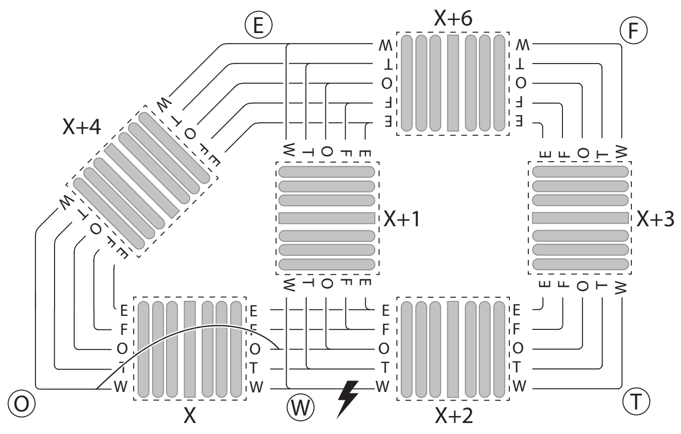

Figure 9. Illustration of the diagonal board. This schematic matches that of Figures 7 and 8, but with one wire of the diagonal board added.

Figure 9 illustrates the diagonal board. It shows the same Bombe as before, but now with an additional connection: between the O wire inside the W bundle, and the W wire inside the O bundle. What the diagonal board did was add connections between different parts of the Bombe’s circuitry, and this added connection would be one of them. Let’s consider what this connection would do in three different scenarios. The first is a scenario in which the wheel settings are incorrect. At incorrect settings, as we have seen, applying the initial voltage results in the signal propagating across all wires of the Bombe (if the menu is restrictive enough). In that case, what the added connection does is facilitate this propagation by providing the signal an additional route. This is why the diagonal board made short cribs more restrictive: it reduced the chances of some parts of the Bombe's circuitry not being reached by the signal when wheel settings were, in reality, incorrect.

The second scenario is one in which the wheel settings are correct and, moreover, the crib/ciphertext pair happens to come from an Enigma machine with a plugboard that routed the W key and W bulb to the O connection inside the plugboard. In the Bombe, this means that the O wire inside the W bundle must be part of the circuit of single wires that are electrically isolated from the rest of the Bombe. The added connection in Figure 9 routes this wire to the W wire inside the O bundle, so if that latter wire does not form a leg of that same circuit then the added connection would break the circuit's isolation, and the Bombe would no longer function as intended. The plugboard design, however, ensured that if an encoding Enigma's plugboard routed the W key and bulb to the O connection, as in this scenario, then it also routed the O key and bulb to the W connection. In other words, the W wire in the O bundle does form a leg of the isolated circuit, so all the added connection does here is link two wires that were already linked, as part of the isolated circuit of single wires.

Very similar reasoning applies to the third possible scenario, in which the wheel settings are also correct, but the crib/ciphertext pair comes from an Enigma machine with a plugboard that did not route the W key and bulb to the O connection inside the plugboard. This means that the O wire in the Bombe's W bundle cannot be part of the electrically isolated circuit of single wires, so the connection added by the diagonal board does no harm as long as the W wire inside the O bundle does not belong to that circuit either. The symmetric design of the plugboard again ensures that this requirement is met, as a plugboard that does not route the W key and bulb to the O connection, cannot route the O key and bulb to the W connection, either. As in the previous scenario, therefore, the added wire does nothing but link two wires that were already part of the same network of connected wires, this time the majority of wires outside of the isolated circuit.

The diagonal board did in a systematic way what the added connection between the W bundle's O wire and the O bundle's W wire does in Figure 9. If we use the letters X and Y as placeholders for all letters present in a crib/ciphertext pair, then for each X bundle the diagonal board connected each Y wire to the corresponding X wire of the Y bundle. For each of those numerous connections the same logic applies as the one laid out above, so the diagonal board reduced the rate at which a Bombe would halt at incorrect wheel settings, without preventing it from halting at correct wheel settings. A major benefit of this, to return to the topic that started our discussion of the diagonal board, was that it allowed the British to use shorter cribs, and thereby avoid problems posed by the ring settings.

The legacy of Turing and his Bombe

The achievements of Turing, Welshman, and their colleagues played a large role in the Allied war effort, by some estimates shortening World War Two by years and saving millions of lives. Rather than being celebrated as a war hero, however, Turing lived a post-war life that was short and marked by tragedy. This was in part because the confidential nature of his codebreaking work limited the general public's awareness of Turing's contributions. More importantly, Turing was a homosexual and homosexual relations were illegal in Britain at the time. A few years after the war, in 1952, Turing was charged with 'gross indecency', and he was convicted shortly after. As a result of this conviction, the nation that was so indebted to Turing, had him undergo hormone injections to suppress his libido. Turing's life ended in 1954, when he was found dead at age 41 after a suspected suicide.

In more recent years Turing has received his due recognition. In the early twenty-first century, the British government apologized for Turing's treatment and pardoned his conviction. There is also general agreement on the importance of Turing's intellectual contributions -- not only as a codebreaker during World War Two, but also as a foundational figure in the fields of computer science and artificial intelligence.

Further reading

I first became interested in the Enigma and the Bombe years ago when reading Simon Singh's *The code book: the science of secrecy from ancient Egypt to quantum cryptography*. I remember that book as a great read and a good introduction to cryptography in general. The book gave me a general sense of how the Enigma and the Bombe worked, and piqued my interest in them. This prompted me to later pick up *Enigma: the battle for the code* by Hugh Sebag-Montefiore. That book provides much more detail about the Bombe and its historical context, and it allowed me to fill in some blanks in my understanding. But I wasn't able to really put it together until I came across http://www.ellsbury.com/enigmabombe.htm, a website that has the best explanation of the Bombe's workings that I have seen.

Footnotes

1. There were actually several Enigma variants that differed in design aspects like the number of wheels, and also in the security of the code they produced. The variant used by the German navy, for instance, had a number of wheels that reached up to eight in later designs, and produced code that was particularly hard to crack.

2. This is a simplification. It is not strictly true that the wheel settings were the same for each message on a given day. As an added security measure, German operators were instructed to randomly choose the rotation positions of the three wheels at the start of each message they sent, but to include three symbols that conveyed those positions at the start of the message in question. What was prescribed by the codebook were the so-called *ring settings*, which (aside from other things, discussed in the main text) determined which three symbols an operator would see, when looking through windows positioned over the wheels, at a given set of wheel positions. In other words, the ring settings determined the relation between the three symbols that the operator included at the start of a message, and that message's wheel positions. Because of this, and even though the wheel settings themselves did not match between messages on a given day, the important point remains that cracking the wheel settings for one message made it straightforward to identify the wheel settings of other messages sent on the same day.