**Description of the Enigma Machines**

**by**

**Tom Cuff**

Table of Contents

(Hint: Ctrl + left-click on the page number to immediately go to the start of that section.)

[1 - Introduction 1](#_Toc353182437)

[2 - Functional Description of the Encryption and Decryption Process 22](#_Toc353182438)

[3 - Top-Level Description of the Software Used to Simulate an Enigma Machine 23](#_Toc353182439)

[4 - How to Test the Software Used to Simulate an Enigma Machine 25](#_Toc353182440)

[5 - Breaking the Enigma Machine Cipher – If it can be done 29](#_Toc353182441)

# 1 - Introduction

We are required for CIS208 (Spring 2013)Team Project to either create software version of an Enigma machine to encode plaintext or a software version of the Polish Bombe used to decipher the Enigma ciphertext. No matter which choice we make, we need to have a good understanding of how Enigma machines function. To this end, I have decided to canvass the literature on these machines and give an overview of their construction and function.

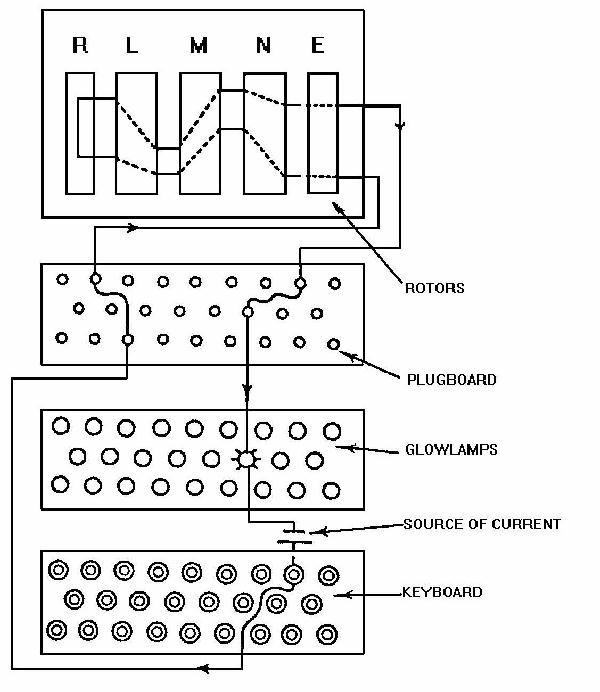
There are numerous books on the subject cryptology, which also contains descriptions of the various types and configurations of Enigma machines.[[1]](#footnote-1) All Enigma machines perform a substitution encryption in which a given letter of plaintext is replaced with different letter of the Roman alphabet in the ciphertext.[[2]](#footnote-2) If each letter of the plaintext was replaced with single letter in the ciphertext, then we would have what is known as a mono-alphabetic substitution cipher, e.g.,

Plaintext Letters : ABCDEFGHIJKLMNOPQRSTUVWXYZ

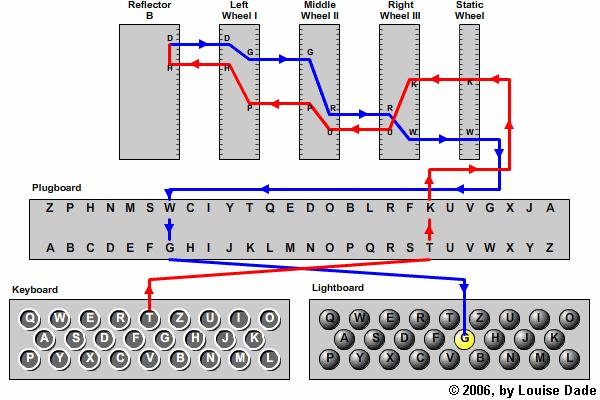
Ciphertext Letters: KSGJTDAYOBXHEPWMIQCVNRFZUL

In the above example, the plaintext letter C is always replaced with the letter G in the ciphertext. For example, the plaintext word ‘FOOD’ would be rendered as ‘DWWJ’ in the ciphertext. In contrast, the Enigma machines employed a poly-alphabetic cipher, wherein a given letter in the plaintext was substituted by more than one letter in the ciphertext.[[3]](#footnote-3) For example, in the case of a certain setting of the rotors of the Enigma machine, the plaintext word ‘FOOD’ would be rendered as HYWI in the ciphertext. In other words, each time the same keyboard key of a particular plaintext letter was pressed on the Enigma machine a different letter would appear as the ciphertext letter in the lighted display.

Physically, the various Enigma machines all had the following subsystems: 1) a 26 letter keyboard for input, 2) a 26 letter, individually lighted display for output, 3) a plug board consisting of 26 pairs of jacks, 4) three (3) or more stepping rotors, and 5) a non-stepping, half-rotor known as a reflector. The following figure gives a very crude schematic representation of a Engima machine with a single static rotor, (E), three (3) stepping rotors (N, M & L) and a reflector (R).

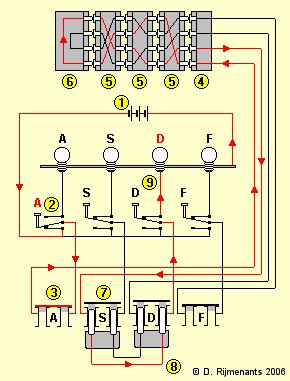


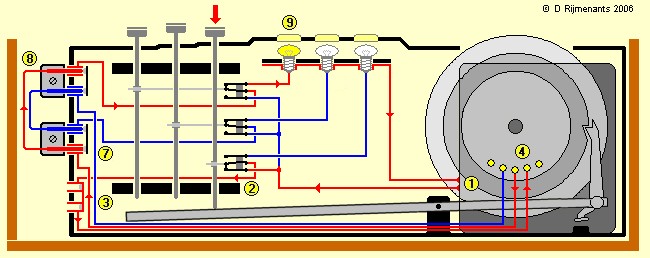
**FIGURE 1-1a – Schematic Representation of an Enigma Machine. Note, this is a very crude representation of the internal circuitry, and it does not covey the true complexity of the actual wiring. (Source:** [**http://faculty.gvsu.edu/aboufade/web/enigma/images/diagram.jpg**](http://faculty.gvsu.edu/aboufade/web/enigma/images/diagram.jpg) **.)**



**FIGURE 1-1b – A Slightly More Detailed Schematic Diagram of a Three (3) Rotor Enigma Machine. Note, this is also a very crude representation of the internal circuitry, and it does not covey the true complexity of the actual wiring. (Source:** [**http://enigma.louisedade.co.uk/howitworks.html**](http://enigma.louisedade.co.uk/howitworks.html) **.)**

Figs.1a & 1b do not, however, convey the true complexity of the wiring of the Enigma machine. For example, in Fig. 1b it is shown how the ‘T’ key is mapped to the ‘G’ light. However, what is not shown is how the ‘G’ key is mapped to the ‘T’ light. In other words, since the Enigma machine is self-inverting, i.e., T → G and G →T, pressing the ‘G’ key should light the ‘T’ bulb, but this means that the ‘G’ key must connect to the G terminal on the plug board, which is already connected, as depicted in Fig. 1b, to the ‘G’ light. In other words, pressing the ‘G’ key should light both the ‘G’ bulb and the ‘T’ bulb. How does the Enigma machine prevent this from happening? The answer to this question can be found in Fig. 1c, which shows the actual wiring of a keyboard key in an Enigma machine.





**FIGURE 1-1c - Two Detailed Views of the Actual Wiring of an Enigma Machine. [TOP]: Detailed view showing the wiring of the keys of the keyboard and the plug board. [BOTTOM]: Detailed view showing the wiring of the keys of the keyboard and plug board together with the mechanical connection to the keyboard keys. (Source:** [**http://users.telenet.be/d.rijmenants/en/enigmatech.htm**](http://users.telenet.be/d.rijmenants/en/enigmatech.htm) **.)**

Fig. 1c resolves the question about how pressing key does not light two of the bulbs simultaneously. Each key is connected to a SPDT (Single Pole Double Throw) switch that disconnects the light corresponding to the key being pressed, and then completes the common path to all the bulbs. …

Let us now consider each of the five (5) subsystems, in turn. The input subsystem, the keyboard, consists of only 26 keys, one for each capital (uppercase) letter of the alphabet. It is interesting what is not included in keyboard: 1) lowercase letters, 2) decimal digits (0-9), 3) a space bar or key, and 4) punctuation marks.[[4]](#footnote-4) Likewise with respect to the output system, the 26 individually lit letters, there are only capital letters represented. The following figures depicts an Enigma machine with its keyboard, lighted display, three (3) stepping rotors, and plug board clearly visible; the reflector rotor is under the top cover and is not visible. The cover or lid of the Engima housing contains ten (10) spare incandescent light bulbs (a.k.a., glow bulbs), two (2) spare patch cords for the plug board, and written instructions in German on how to use the machine.



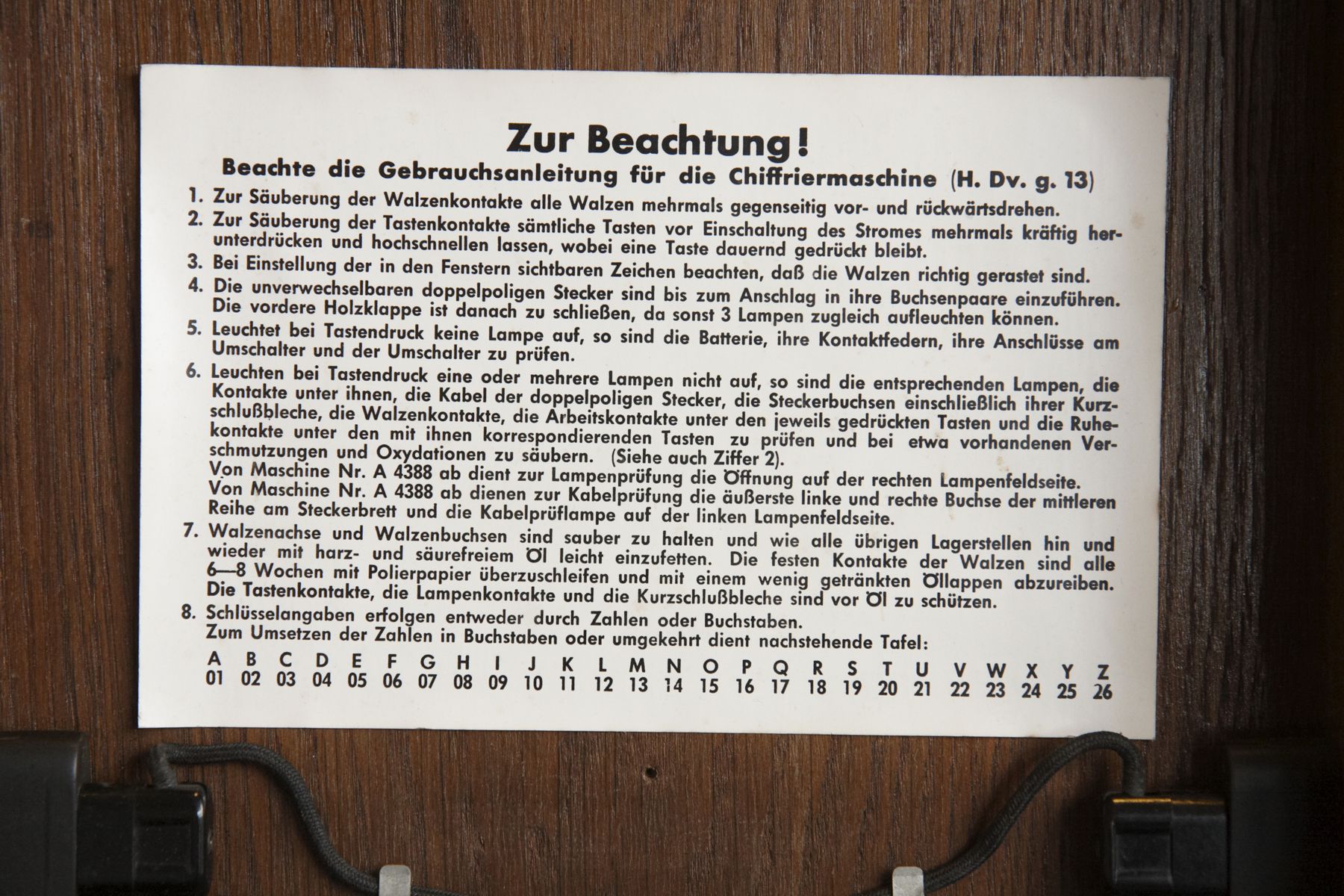
**FIGURE 1-2a- A Three (3) Rotor Enigma Machine. Note the power selector switch to the right of the three (3) rotors and to the immediate left of the pair of battery terminals. (Source: Technical University of Denmark,** [**http://www.matematiksider.dk/enigma\_dtu\_eng.html**](http://www.matematiksider.dk/enigma_dtu_eng.html) **.)**

The absence of number, space, and punctuation keys on the Enigma keyboard give a hint as to how messages were sent to and from it. The letters of the plaintext and ciphertext were run together in groups of five (5). Various sources have corroborated this idea that the ciphertext was sent in fixed-size groups of letters.[[5]](#footnote-5) In other words, suppose you wished to transmit the following plaintext message: “The quick brown fox jumped over the lazy dog”. This sentence would be written out as follows: [[6]](#footnote-6)

THEQU ICKBR OWNFO XJUMP EDOVE RTHEL AZYDO GXYZW

Note the use of nonsense letters, XYZW, in the last group of five letters to fill out the otherwise empty spaces. By employing fixed-size letter groups, no hint was given to people trying to break the code about the presence of one, two or three letter words, which would have made the task breaking the code somewhat easier. As we shall later in this introduction when we discuss the US Patent for the commercial version of the Enigma machine, there is an even more fundamental reason for employing a fixed grouping of the letters.

These five letter groups would then be submitted to the Enigma operator for encryption into, again, five letter groups. The encrypted five letter groups would then be sent via radio using Morse code. The use of radio to communicate messages to people in the field was the reason why encryption was necessary since anyone with a radio receiver could intercept the transmission. At the receiving end, the radio operator would transcribe the five letters groups representing the encrypted message and then give them to the Enigma operator to be decrypted into five letter groups, which, finally, would be read by someone who would organized the letters into variable length groups representing words in the language in use.



**FIGURE 1-2b- The Operating Instructions of an Enigma Machine (Source: Technical University of Denmark,** [**http://www.matematiksider.dk/enigma\_dtu\_eng.html**](http://www.matematiksider.dk/enigma_dtu_eng.html) **.).**

The following is my rough and ready translation of the German language operating instruction into English:

For Your Consideration!

Operating Instructions for the Cipher Machine (.. [H. Dv. g. 13])

1. To cleanse the rotor contacts of all rotors, repeatedly rotate them against one another in the forward and reverse direction.
2. To cleanse the keyboard contacts of all the keys before switching on the current repeatedly, forcefully press down and release upwards, whereby a key continues to remain pressed.
3. At the beginning observe that the number is visible in the window, so that the rotors are properly notched/registered.
4. The distinctive double pronged plug is be pushed into its electrical socket until it stops.
5. If during the pressing of a key no lamp lights, then the batteries, its spring contacts, its cable at the switch and the switch are to be tested.
6. If one or more of the lights do not light at a key press, then the aforementioned lamps, the contacts under them, the cable of the double pronged plugs, the plug sockets …, rotor contacts, the working contacts under the respective pressed key, and the resting contacts … are to be tested, and there about observed dirt and oxidation is to be removed. (See also Step #2, above).
7. Rotor studs and rotor sockets are to be cleaned and as … The solid contacts of the rotors are …
8. The key code follows either through numbers or letters.

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

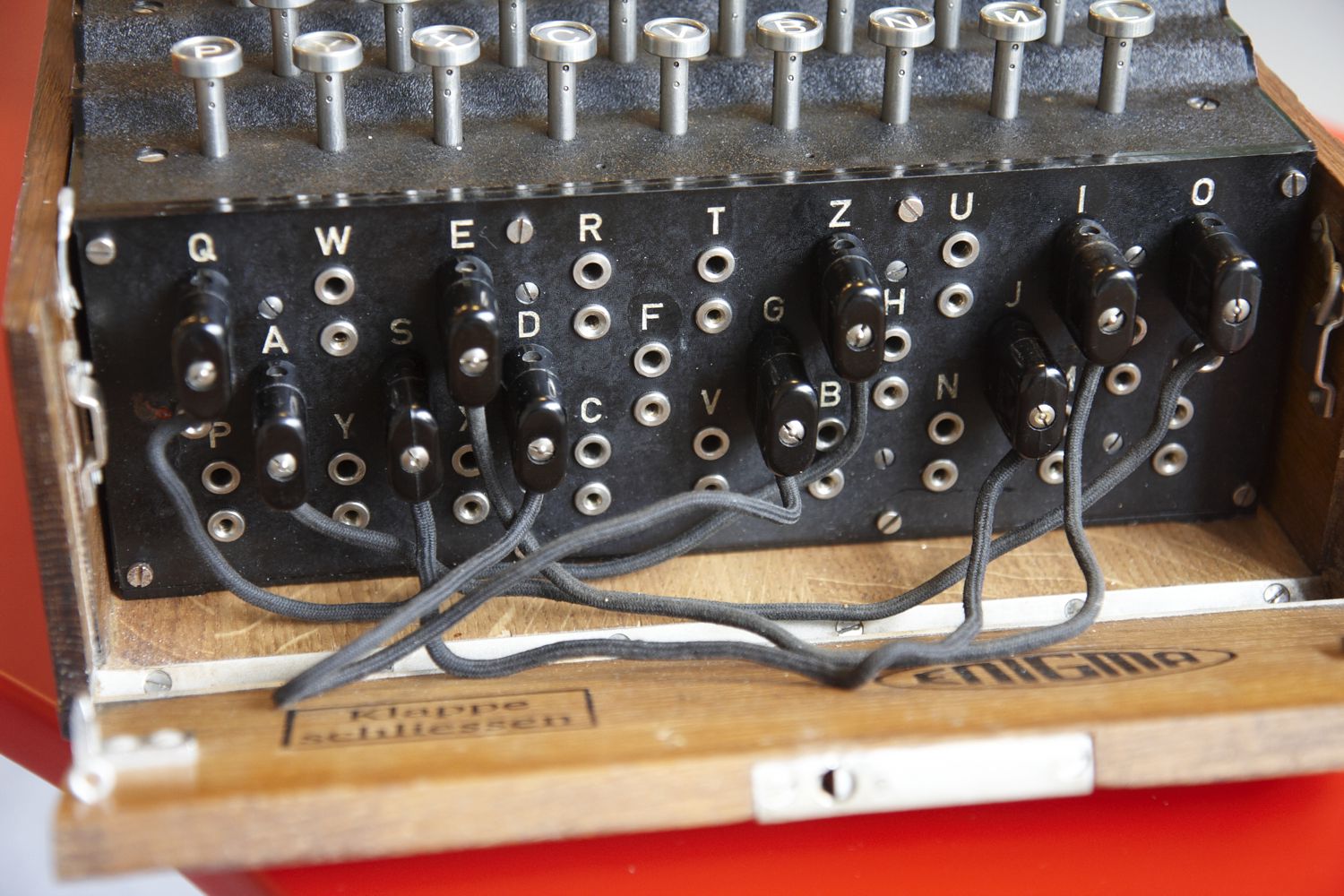
01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26

The plug board, situated on the front of the Enigma machine, is used to further complicate the encryption.

The more important complicating feature, however, was the attachment of a plugboard. It was this that distinguished the military from the commercial Enigma, and made it something that had unnerved the British analysts. It had the effect of performing automatically an extra swapping of the letters, both before entering the rotators, and after emerging from them. Technically, this was achieved by attaching wires, with plugs at each end, into a plugboard with 26 holes – rather like making connections on a telephone switchboard. It required ingenious electrical connections, and the use of double wires, to have the required effect. Until late 1938, it was usual in the German use of the machine to have only six or seven pairs of letters connected in this way. [Hodges 1983, p. 169]

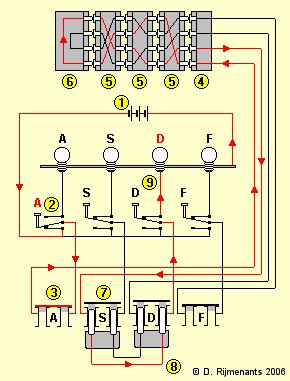
The above block quotation is incomplete in one vital respect, the plug board did not consist of “… 26 hole …”, rather, it consisted of 26 pairs of holes. The reason why there had to be a pairs of holes for each letter is that, like the switchboard in a telephone exchange, the cable connecting various jacks has two wires in it, one going to the tip of the plug and one going to the ring of the plug with the tip and ring insulated from one another..

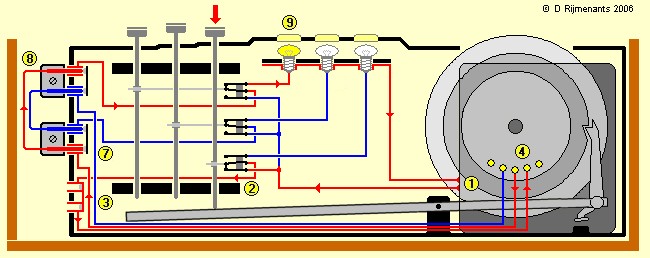
The patch cords for the plug board employed a cable with two, insulated wires in it: one wire going to upper portion and one to the lower portion of the jack pair. As the following figure shows quite clearly, each pair of jacks consists of large jack above a small jack. This size asymmetry was needed in order to prevent the operator from inserting the plug upside down effectively crossing the wires of the patch cord cable, i.e., the size asymmetry served as a ‘key’ for the connector thus ensuring proper mating.



**FIGURE 1-2c – The Plug Board of an Enigma Machine (Source: Technical University of Denmark,** [**http://www.matematiksider.dk/enigma\_dtu\_eng.html**](http://www.matematiksider.dk/enigma_dtu_eng.html) **.).**

Figs. 1a & 1b purport to explain how the plug board is wired into the Enigma machine, but they are of little value in this respect. For example, if no patch cords are plugged into the plug board, how can keyboard and the light panel still be connected to the rotors? The answer to this question can be found in Fig. 1c, which for convenience sake has been duplicated below. Fig. 1c reveals that the dual jacks in the plug board are normally internally short-circuited, i.e., when the corresponding plug-pair is not inserted, the jack-pair are connected together. Operationally, we know that only a few patch cords were used with the plug board. There is a second difficulty with the simple-minded wiring diagrams found in Figs. 1a & 1b, and this relates how the light in the light panel are connected as was mentioned earlier.

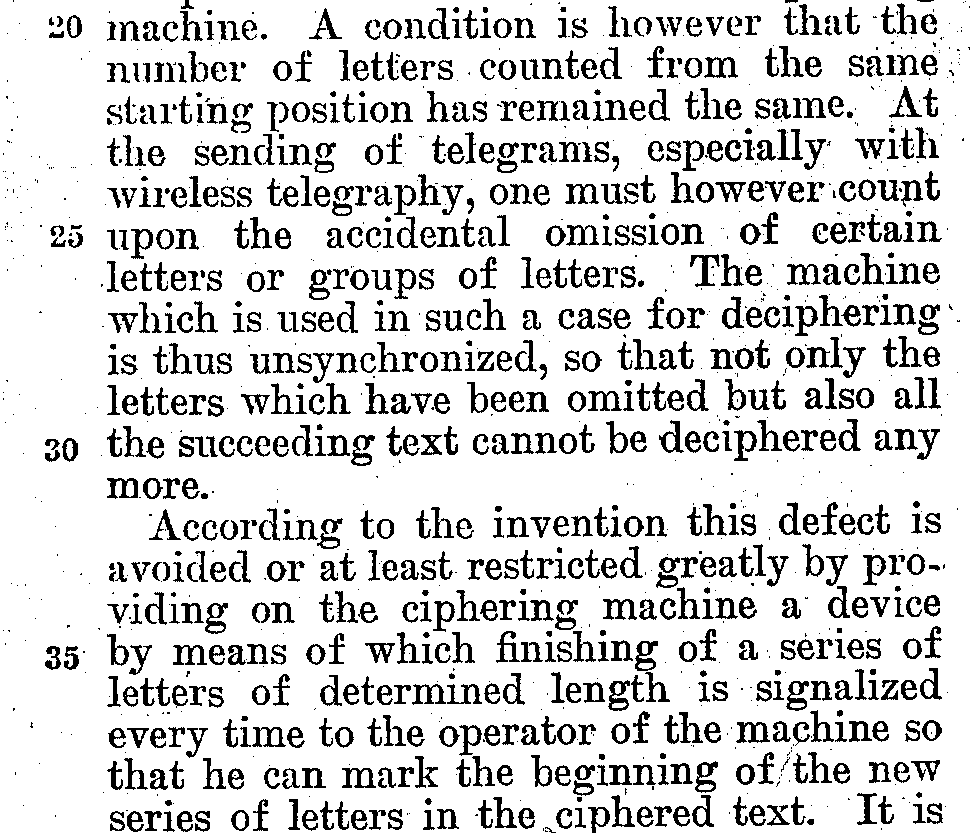




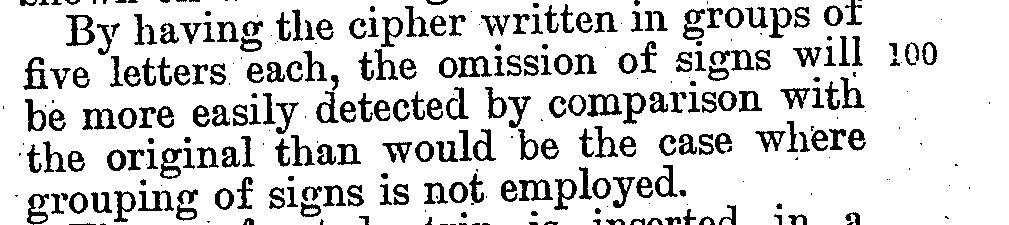
**FIGURE 1-1c [Duplicated from earlier in this document.] - Two Detailed Views of the Actual Wiring of an Enigma Machine. [TOP]: Detailed view showing the wiring of the keys of the keyboard and the plug board. [BOTTOM]: Detailed view showing the wiring of the keys of the keyboard and plug board together with the mechanical connection to the keyboard keys. (Source: http://users.telenet.be/d.rijmenants/en/enigmatech.htm.)**

According to the book by Konheim, the original, commercial version of the Engima was patented both in Germany and in the United States. Konheim indicates that the US Patent had the number 1,657,411 with the US Patent filed on February 6, 1923 and granted on January 24, 1928.[[7]](#footnote-7) Fig. 2e contains the drawings from the first sheet of US Patent No. 1,657,411. [[8]](#footnote-8) The commercial version of the Enigma machine revealed in the US Patent differs from the military version employed by the German military during WWII in the following ways: 1) the commercial machine had 4 stepping rotors instead of the 3 found in its military counterpart; 2) the commercial machine did not have a half-rotor (a.k.a., reflector) like the military version, and 3) the commercial machine did not employ the plug board used in the military machine. There are other differences between the commercial and military versions of the Enigma machine, which will be discussed later.

In the text of the US Patent, the inventor Arthur Scherbius mentions that these types of encryption machines had a very serious defect, see the excerpt below from page 2( sheet 1) of the US Patent,



The inventor indicates in the US Patent a counter could be added to the cipher machine so that, for example, a bell might ding to indicate that a predetermined number of characters had by enciphered. As an aid to detecting missing characters the inventor makes the following suggestion on page 4 (sheet 3) of the US Patent,

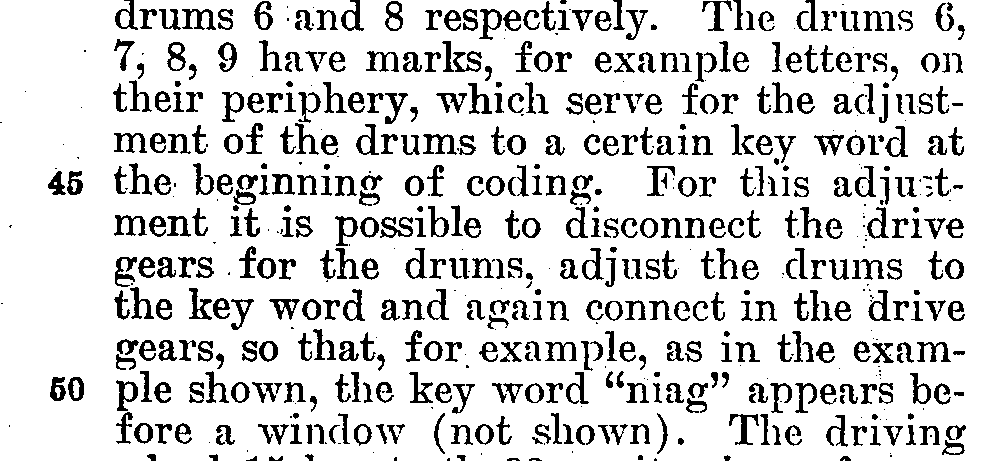


As was mentioned earlier, there are other differences between the commercial and military versions of the Enigma machines, and it would helpful to review before moving on. …

The ‘heart’ of both the commercial and military Enigma machines is the group of stepping routers. For the purposes of illustration, we shall only consider the military Enigma machine with its three (3) stepping rotors. These rotors, which have 26 positions corresponding to the letters of the Roman alphabet, are geared so that they rotate as do the wheels of a odometer, i.e., each full rotation of the slow rotor requires 26 full rotations of the medium rotor, while each full rotation of the medium rotor requires 26 full rotations of the fast rotor. From Fig. 2e, it is obvious that the fast rotor is the rightmost rotor since it is driven by the smallest gear (17) compared to gear (16). This arrangement is exactly what is found in an odometer. The miles wheel is the rightmost wheel, the tens of miles wheel is the middle wheel, and the hundreds of miles wheel is at the far left. The miles wheel rotates the fastest, followed by the tens of miles wheel, and the hundreds of miles wheel rotates slowest. Let us suppose that the three (3) odometer wheels are initially all set to 0: 100’s of miles wheel = 0, the 10’s of miles wheel = 0, and the miles wheel = 0. As the miles wheel rotates to higher miles, eventually it gets to mile 9, and then in the transition from mile 9 to mile 0, the 10’s of miles wheel rotates from the 0 position to the 1 position. …

In class on Tuesday, 2/12/13, one of the students mentioned that, in response to the pressing of a key on the keyboard, the fast stepping rotor moves first and then the light lights on the display, i.e., this is similar to the difference between the pre-increment, ++i, and post-increment, i++, operations in Java and C++. This is an important fact about the operation of the Enigma machine, which is rarely talked about. One can easily confirm that the military version of the Enigma machine does indeed employ the pre-increment operation by looking at the DVD of the movie Enigma, Scene 20 – *Decoding the intercepts*. In this scene the protagonists have obtained copies of some intercepts, which they try to decode using an Enigma machine. With the front panel raised, the lead character, Tom Jericho, demonstrates what happens when a key on the keyboard is pressed. One can see the key being pressed, the fast stepping (right most) rotor turning, and then the light bulb of the light panel illuminating. If one hits the PAUSE button of the DVD player remote control, then one can step through the scene one frame-at-a-time by repeatedly pressing the STEP button. The pre-incrementing is quite clear in the movie. It should be noted that the military version of the Enigma machine used in the movie is, in fact, an actual Enigma machine loaned to the movie company by one of the movie’s producers, the singer Mick Jagger, who also appears in the movie for a few seconds in Scene 10 – *Happiest moment of his life*.[[9]](#footnote-9)

As mentioned earlier, if the same letter key on the keyboard is pressed multiple times, it is mapped (encrypted) to a different letter. Of course, there are only 25 letters to which it can be mapped since it cannot be mapped to itself, but in an Enigma machine the key can be quite long, though still finite. The key code/word corresponds to the three (3) letters in the windows above the fast, medium and slow rotors. For example, in a three (3) stepping rotor machine a possible key code/word is RHV, corresponding to the slow (leftmost), medium (middle) and fast (rightmost) stepping rotors. The inventor makes the same statement, with regard to the four (4) stepping rotor machine, described on page 3 (sheet 2) in the US Patent,



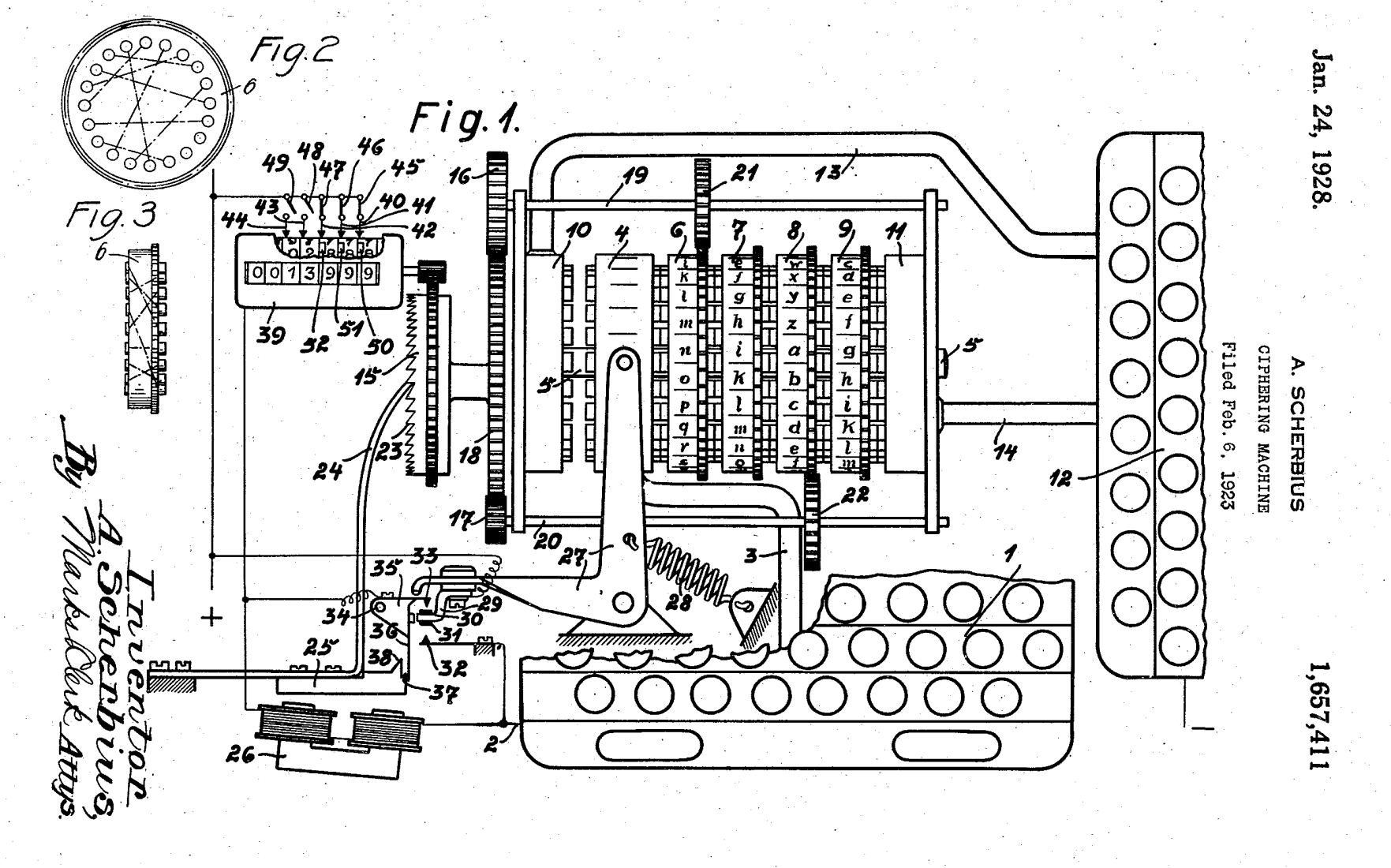
The key code/word determines the encryption key that is applied to a given letter of plaintext. The length of the encryption key is determined by the number of rotations for the key code/word to reappear in the windows of the stepping rotors. For example, supposing the same letter key on the keyboard was continually pressed, what would you observe? Starting with the initial key code/word, RHV in a three (3) stepping rotor machine, each time the same letter key is pressed the key code/word showing the windows would change. Eventually, though, as one kept pressing the same letter key, the initial key code/word would reappear in the windows of the three (3) stepping rotors. However, it would take 262 = 676 rotations of the fast stepping rotor, 26 rotations of the medium stepping rotor, and 1 rotation of the slow stepping rotor for the initial key code/word to reappear. In other words, the key is of length 263 = 17,576. However, as mentioned in Hodges’ book,

But there was much more to the machine actually in military use. For one thing, the three rotors were not fixed in place, but could be removed and replaced in any order. Until late 1938 there was a stock of just three rotors, which therefore allowed a total of six arrangements. In this way, the machine offered different alphabetic substitutions. [Hodges 1983, p. 168]

vvv

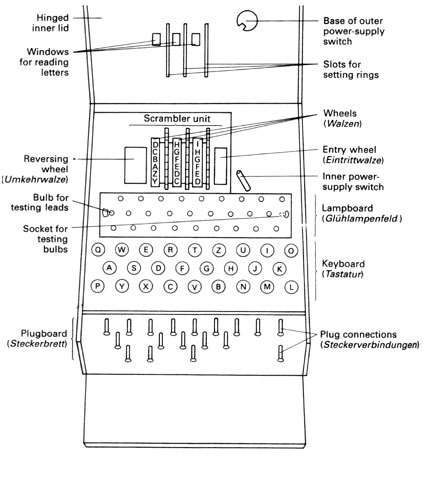


**FIGURE 1-2d – The Three (3) Stepping Rotors of an Enigma Machine. Note, the important questions with respect to this picture are as follows. First, which of the three (3) stepping rotors is the fast, medium, and slow one? Second, where is the reflecting rotor (a.k.a., the reflector)? The answers to these questions can be found, as explained in the text, at the following website:** [**http://www.pbs.org/wgbh/nova/military/how-enigma-works.html**](http://www.pbs.org/wgbh/nova/military/how-enigma-works.html) **. From left to right, the arrangement of the rotors is as follows: 1) the far left rotor embossed with the letter ‘B’ is the reflector [Umkehrwalze]; the next three (3) rotors with serrated edges are, respectively, the slow, medium and fast stepping rotors [Walzen]; and the black rotor at the far right is the fixed or entry rotor [Eintrittwalze]. (Source: Technical University of Denmark,** [**http://www.matematiksider.dk/enigma\_dtu\_eng.html**](http://www.matematiksider.dk/enigma_dtu_eng.html) **.)**



**FIGURE 1-2e - Diagram of a Commercial Enigma Machine taken from Sheet 1 of the US Patent 1,657,411. Note, for the sake of simplicity only drive gears (21 & 22) for rotors 6 & 8, respectively, are shown. (Source: Arthur Scherbius (Assignors/Inventors), Chiffriermaschinen Aktiengesellschaft [Cipher Machines, Joint-Stock Company], Berlin (Assignee); Ciphering Machine; US Patent 1,657,411; January 24, 1928; 7 pp.; URL:** [**http://www.uspto.gov/old-index.html**](http://www.uspto.gov/old-index.html) **.)**

The arrangement of the stepping rotors, the location of the reflector, and the stationary rotor can be found on various websites. For example, the PBS (Public Broadcasting System) Nova website, <http://www.pbs.org/wgbh/nova/military/how-enigma-works.html> , provides the following annotated schematic drawing of a three (3) stepping rotor military Enigma machine.



**FIGURE 1-3 – Annotated Schematic Drawing of a Military Enigma Machine (Source:** [**http://www.pbs.org/wgbh/nova/military/how-enigma-works.html**](http://www.pbs.org/wgbh/nova/military/how-enigma-works.html) **.).**

**TABLE 1-1 – Enigma Machine Versions According to Wrixon’s Book (Source: Fred B. Wrixon; Codes, Ciphers & Other Cryptic & Clandestine Communication; Barnes & Noble Books; 1998; pp. 260-263.)**

|  |  |  |  |
| --- | --- | --- | --- |
| Enigma Machine Version | Number of Stepping + Non-Stepping Rotors | Reflecting Rotor |  |
|  |  |  |  |
| A | 4 | No ¥ |  |
| B | - | - |  |
| C | 4 | Yes £ |  |
| D | 4 | Yes € |  |
|  |  |  |  |

¥ According to Wrixon, “Model A began with four rotors driven by four geared wheels [Wrixon 1998, p. 262].”

£

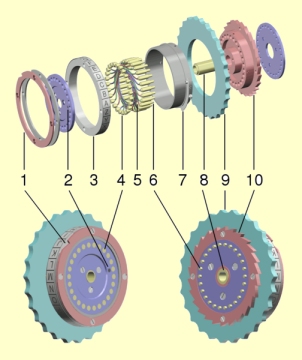
€

There is one last confounding feature of the Enigma machine, which was mentioned in Hodges’ biography of AlanTuring,

Obviously, the rotors had to be marked in some way on the outside so that the different positions could be identified. However, here entered yet another element of complexity. Each rotor was encircled by a ring bearing the 26 letters, so that with the ring fixed in position, each letter would label a rotor position.\* (In fact, the letter would show through a window at the top of the machine.) However, the position of the ring, relative to the wirings, would be changed each day. The wirings might be thought of as labeled by numbers from 1 to 26, and the position of the ring by the letters A to Z appearing in the window. So a ring-setting would determine where the ring was sit on the rotor, with perhaps the letter G on position 1, H on position 2, and so forth.

\* The rather tiresome complication introduced by the ring-setting is, unfortunately, required to make sense of what the Poles achieved. It will play no part thereafter. [Hodges 1983, pp. 168-169]

The following figure depicts the construction, in exploded view, of an Enigma rotor, and it shows the relationship between the ring and the notch used to advance the next slowest rotor, if there is one.

The rotors (Walzen in German) are the most important elements of the machine. These round disks [9], approximately 4 inch in diameter, are made from metal or bakelite and have a core with 26 spring-loaded contacts [6] on the right side, scramble wired [5] to 26 flat contacts [4] on the left side, with a hollow axle in the center [8]. On the outside of the wiring core there's a movable ring [3] with 26 numbers or letters and a notch [1]. This ring is rotatable and is locked with a sprinloaded pin (Wehrmacht) [7] or two springloaded arcs (Kriegmarine) into any of the 26 positions. Changing the position of the ring will change the position of the notch and alphabet, relative to the internal wiring. This setting is called the ring setting or Ringstellung and its position is visible by a dot marking [2]. Each rotor has on its left a notch [1] and on its right a ratchet [10]. These are used by the stepping mechanism to advance the rotors. The internal wiring is different for each rotor. This wiring represents a substitution encryption. The combination of several rotors, in ever-changing positions relative to each other, is what makes the encryption so complex.

The machine was introduced with three rotors. In 1939 the set was extended to five rotors, marked with Roman numerals I, II, III, IV and V, all with a single notch. The Kriegmarine extended this set of rotors with another three rotors called VI, VII and VIII, each with two notches. In 1942, the Kriegsmarine M4 introduced a fourth rotor. To achieve this, the wide B and C reflectors from the three rotor version were replaced by thin B and C reflectors, leaving room for the special fourth rotor. The fourth rotors were of two configurations, named Beta and Gamma, with spring-loaded contacts on both sides, making them incompatible with the other eight rotors.

FIGURE 1-4 – Explanation of the Construction of an Enigma Rotor (Source: <http://users.telenet.be/d.rijmenants/en/enigmatech.htm> .).

# 2 - Functional Description of the Encryption and Decryption Process

Konheim’s book gives a very clear a succinct description of the …

Hhh

# 3 - Top-Level Description of the Software Used to Simulate an Enigma Machine

There is no complicated mathematics needed to code the Enigma simulator. A brute force approach would simply use a set of integer arrays for the various mappings with each array element being pointed to via cyclic indices. Cyclic indices are indices that simply wrap around. For example, suppose the Enigma simulator only had four (4) letters from the Roman alphabet (A, B, C & D). Such a machine could be simulated using integer arrays containing four (4) elements with the following indices 0, 1, 2 & 3. In order to simulate the circular stepping rotors with a linear array, the array indices will have to be cyclic in the sense that as you increment or decrement them, they never fall off the front or back end of the array. In order to make the indices cyclic, after each increment, decrement or both, modulo arithmetic must be applied to them to keep them in the valid range: 0, 1, 2 & 3. There are two (2) rules that will be required to make the coding of the cyclic indices function properly:

1. always calculate the modulo of the sum of the indices, never the sum of the individual modulo for each index; and
2. one must have a way of dealing with a negative modulo.

Rule #1: Anytime you perform addition of two integers to form a new integer to be used as an index to an array, a modulo division of the sum must be performed before using the sum as an index. In other words, suppose you have to add together two indices, and use the result as an index to the array of four letters. In C++, the correct way to perform the addition is as follows, [[10]](#footnote-10)

intOutput = Array[(intIndex1 + intIndex2)%N];

In C++, the incorrect way to perform the addition would be as follows,

intOutput = Array[(intIndex1)%N + (intIndex2)%N];

The reason the second approach is incorrect is that the sum of modulo additions [the second approach] does not necessarily equal the modulo of the sum [the first approach]. For example, in the case where N = 4, the correct approach yields an index that is in the valid range (0, 1, 2 or 3)

(3-1)

while the incorrect approach yields an index value outside the valid range,

(3-2)

Rule #2: the other problem with using the C++ language is what happens when, for example, we perform a modulo division of a difference between indices. For example, consider the following C++ statement,

intDiff = (0 – 1)%N;

where, say, N =4 . The *lvalue* (left-value), intDiff, will equal -1, which will cause a runtime error if it is used as an index value since index values are only allowed to be non-negative and, further, must not fall off the front or rear end of the array.[[11]](#footnote-11) The simplest solution to this problem is anytime a difference is formed between indices, the resulting *lvalue* must be passed through the following ‘if’ statement to convert it into a positive integer value in the range 0-3,

if(lvalue < 0)

{

lvalue = N + lvalue;

}

else

{

// Do nothing.

}

One other point needs to be made about the use of modulo arithmetic. If the Enigma software simulator uses pointers instead of array indices, a problem arises. While pointers to contiguous memory data structures, such as arrays of primitive types, objects or structures, can be incremented and decremented like integer variables, there is no built-in modulo operator for pointers, i.e., trying to use the ‘%’ operator with pointers will lead to a compile-time error. In other words, if you decide to use pointers, you will have to write your own modulo operator – perhaps by overloading the normal modulo operator ‘%’.

# 4 - How to Test the Software Used to Simulate an Enigma Machine

After the software that purports to simulate the Enigma machine has been written, the question naturally arises as to how “prove” that the software is indeed functioning properly. There are basically four (4) tests that must be performed and successfully executed to assure proper operation of the Enigma software:

1. Single Letter Entry Test - Choose a single letter, say ‘A’, and repeatedly enter the letter through the keyboard, while, at the same time, observing the key code [of the three stepping rotors] and the display output. If the software is functioning properly, the letter ‘A’ should never map to itself since it is the job of the reflector to assure that this does not happen. The number of times the letter ‘A’ must be entered to complete this test can be determined by looking at the key code. If the initial key code is say XYZ, then each time the ‘A’ key is pressed the key code will change – increment. When the initial key code is reappears, that is when the testing can stop. If the letter ‘A’ has not mapped to itself by then, then the test is a partial success in that it implies that the reflector is functioning properly. However, in order for this test to be a complete success, during one complete cycle of the key code from the initial value of the key code to its reappearance, the letter ‘A’ must map to each other letter of the alphabet at least once. The proof of this last statement will be provided later in this section of this document.
2. Repeated Entry Test - In this test, each time a key on the keyboard is pressed, the key code must be carefully observed to make sure that it increments in an orderly manner. For example, one very common problem with the software used to simulate the Enigma machine is that it causes a race condition in the slow stepping rotor, i.e., the slow stepping rotor advances (rotates) faster than the medium stepping rotor. If racing is detected, then the software is not functioning properly. Note, the odometer arrangement of the stepping rotors, wherein one rotor drives its left hand counterpart, is analogous to a digital electronics shift register, binary counter, etc. and such serial arrangements of flip-flop gates can, if improperly designed and applied, exhibit race conditions.
3. Encryption Test - The penultimate test is simply verification that the mappings are what are expected for the combination of a) stepping rotors, b) location (Slow, Medium and Fast) of the stepping rotors, c) ring settings of the stepping rotors, d) reflector configuration, and e) plug board configuration. Of course, the issue here is how does one construct all the possible mapping sequences? If one had a “known good’ Enigma simulator, then that could be used to construct all possible mapping sequences. But that is just the point of the matter, we normally don’t have such a device.
4. Decryption Test - The final test consists of submitting encrypted text to the software to be converted into plaintext (clear text). One might think that this is a redundant test given that the software has passed the Encryption test, but the caveat in this final test is that the encrypted text must come from an Enigma simulator designed and coded by another group, but having the same configuration of stepping rotors, ring settings, and plug board wiring. This test can be thought of as a reproducibility test, i.e., two Enigma software simulators designed and coded by two different groups are the “same” if they can code and decode each other’s messages. Of course, this “sameness” does not imply that the two simulators were coded properly, just that they function the same, i.e., they could have both been designed and coded the same wrong way. If the two Enigma simulators disagree, then the question arises as to which one is correct, admitting, of course, the possibility that both could be incorrect, just in different ways.

The first two tests (#1 - Single Letter Entry Test & #2 - Repeated Entry Test) provide necessary but not sufficient conditions for the proper functioning of the Enigma simulator. In other words, the Enigma simulator must pass these two tests to be functioning properly, but passing these two tests does not guarantee proper functioning. Some further explanation will be needed in order to understand the previous statement. Consider the following truth-functional relation, [[12]](#footnote-12)

(4-1a)

(4-1b)

which can be recast as the following English language sentence,

If the Enigma simulator is functioning properly, then it passes tests #1 & #2.

In Eq. 1, the antecedent (*p*), “Enigma simulator functioning properly” implies “→” the consequence (*q*) “Passes Tests #1 & #2.” In other words, assuming that Eq. 1 is TRUE, the consequence “Passes Tests #1 & #2.” being TRUE is a necessary condition for the antecedent “Enigma simulator functioning properly” to also be TRUE. However, Eq. 1 can be TRUE and the consequence can be TRUE without the antecedent being TRUE, i.e., the antecedent could be FALSE. The following is the truth table for the truth-functional implication in Eq. 1,

**TABLE 4-1 – Truth Table for the Truth-Functional Relation (Source: Irving M. Copi, Carl Cohen, Kenneth McMahon; Introduction to Logic, 14th Ed.; Prentice Hall; 2011; p. 303. )**

|  |  |  |
| --- | --- | --- |
| *p* | *q* | *p → q* |
|  |  |  |
| T | T | T |
| T | F | F |
| F | T | T |
| F | F | T |
|  |  |  |

Unfortunately, Eq.1 does not imply that the converse of Eq. 1 ,

(4-2)

is TRUE.

One would think that the Internet would be full of test strategies for Enigma simulators, but, so far, I have not had much success locating such procedures. I was able, though, to locate a Master’s Thesis in which the author ran into the reproducibility problem amongst different software Enigma simulators,

It was only after a few weeks into designing an Enigma simulator that I discovered several simulators already existed, two I shall refer to belong to Andy Carlson(Fig.1) and Russell Schwarzger(Fig.2). Testing both Enigmas, I was quickly aware that the two simulators gave different answers despite looking similar in design. It was at this point that I felt that my design plan needed to be re-evaluated and altered to prevent myself ‘reinventing the wheel’. After some time spent investigating the two Enigma simulators and trying to figure out why they gave different outputs I became aware of the lack of information conveyed by the two machines. The two machines looked the same because they looked like an Enigma, but their respective inner workings must have been different for the results to be so (with all other things being equal, i.e. the Enigma settings). This was the only deduction I could make from each of the two machines. However, if I could visually demonstrate the paths the electrical signal took passing through the machine I could at least justify any answer my simulator gave rather than simply churning out possibly meaningless (and even random) characters. This would serve the purpose of improving understanding of the machine and increase the scope beyond the two existing machines. [Wong 20xy, p. 5][[13]](#footnote-13)

His solution to this conundrum was as follows,

Going back to the question I asked about which of the two previous Enigmas were accurate, it occurred that this question would be asked of mine. The definitive answer of course would be to compare messages encrypted by those simulators with an actual Enigma machine. For lack of available real life Enigmas, this is just not feasible.

But, given that the action motion of an Enigma was relatively simple and consistently reported, the most likely source of inaccuracy was in the mappings of each wheel, for authenticity I wanted an accurate source for which to base my wheel mappings on, but in theoretically, so long as there existed a mapping for each of my wheel it should not affect the enciphering ability of my simulator.

1. To prove that any machine is consistent with itself a message must be able to encrypt and decrypt itself using the same base settings. This would be the most fundamental requirement of any Enigma machine and will be the first test I shall perform.

2. As previously mentioned, the only reason why two simulators should produce different results would most probably lie in the order of the mappings for each wheel. If I used the same mappings as any other Enigma, my results should be identical. For this reason I have chosen to use the same wheel mappings as Andy Carlson’s Enigma so I could compare my results with an existing machine. [Wong 20xy, pp. 6-7]

# 5 - Breaking the Enigma Machine Cipher – If it can be done

One of the great myths about the efforts of the British coder breakers at Bletchley Park during WWII is that they brute force “broke” the Enigma codes. Nothing could be further from the truth.

There are, in fact, so many contradictory stories about the Enigma machines and how their code was broken, that it is hard to know where to start. For example, the first electro-mechanical device used to help break the Enigma code was the Polish *Bombe*. According to Andrew Hodges,

By November 1938 they [the Poles] had actually built such machines – six in fact, one for each possible rotor order. They produced a loud ticking sound, and were accordingly called the *Bombes* [bombs]. [Hodges 1983, p.175]

Hodges’ explanation for the origin of the name *Bombes* is quite reasonable, except that it is contradicted by other sources,[[14]](#footnote-14)



**FIGURE 5-1 – Description of the Origin of the *Bomba* or *Bombe* (Source: Chris Christensen; The Evolution of the Cryptographic Bombe; Department of Mathematics, Northern Kentucky University; 2/25/2010; p. 42).**

For those of us who are unfamiliar with the *Bomba* or *Bombe* as a desert, the following picture will be of some help.



**FIGURE 5-2 - A *Bombe* Desert (Source: Shona Crawford, Jasper Partington; The Ice Cream Book; Octopus Books Limited; 1980; p. 73.).**

Xxx



**FIGURE 5-3 - Description of the Origin of the Bomba or Bombe (Source: Chris Christensen; The Evolution of the Cryptographic Bombe; Department of Mathematics, Northern Kentucky University; 2/25/2010; p. 35).**

Xxx

1. Andrew Hodges; Alan Turing: The Enigma; Simon and Schuster, Inc.; 1983; pp. 160-241, Chapter 4 – Relay Race.

   Fred B. Wrixon; Codes, Ciphers & Other Cryptic & Clandestine Communication; Barnes & Noble Books; 1998; pp. 82-85, 91, 253, 260-263.

   Alan G. Konheim; Cryptography: A Primer; John Wiley & Sons, Inc.; 1981; pp. 212-227. [↑](#footnote-ref-1)
2. The term ‘Roman or Latin alphabet’ is a bit of a misnomer since, even though the letters in the Romans alphabet had the same form as we use today, there were only 23 letters (the letters J, V & W were missing), as opposed to the 26 letters employed in what is called the ‘Modern European alphabet’.

   Anonymous; Webster’s Seventh New Collegiate Dictionary; G. & C. Merriam Company; 1965; p. 26.

   B. F. C. A., J. Wh.; Alphabet; in Encyclopædia Britannica, Vol. 1 - Antarah; William Benton; 1964; pp. 667-668. [↑](#footnote-ref-2)
3. Andrew Hodges; Alan Turing: The Enigma; Simon and Schuster, Inc.; 1983; p. 163. [↑](#footnote-ref-3)
4. In the case of the German Enigma machines, there is also no β, , Ä, Ö, or Ü which are standard characters in the German language. Since the β normally represents the double S, i.e., β = SS, as in the German word for street, *straβe*, one simply uses the double SS’s instead of β. In the case of the letters A, O and U with the umlaut, the traditional way to represent letters carrying this diacritical mark is to replace the single letter with a two letter combination that sounds the same phonetically. For example, one can replace Ö with OE, which has the same sound.

   Margaret Keidel Bluske, Elizabeth Keidel Walther; Das Erste Jahr, 2nd Ed.; Charles Scribner’s Sons; 1970; pp. … [↑](#footnote-ref-4)
5. The US Patent for the commercial version of the Enigma machine makes the same assertion about the fixed-size letter groups of ciphertext. This patent will be discussed later in this document.

   Arthur Scherbius (Assignors/Inventors), Chiffriermaschinen Aktiengesellschaft [Cipher Machines, Joint-Stock Company], Berlin (Assignee); Ciphering Machine; US Patent 1,657,411; January 24, 1928; p. 4 (sheet 3).

   The main character in the 2002 movie ‘Enigma’ mentions that four (4) letter groupings of ciphertext were used by the German Navy, while five (5) letter grouping were employed by the German Army and Air Force (Luftwaffe).

   Starring: Dougray Scott, Kate Winslet, Jeremy Northam, Safron Burrows; Directed: Michael Apted; Screenplay: Tom Stoppard; Based on the book by Robert Harris; Enigma; Manhattan Pictures International, LLC; 2002; 119 minutes; DVD Scene 8 – *Missing German Intercepts*. [↑](#footnote-ref-5)
6. Alan G. Konheim; Cryptography: A Primer; John Wiley & Sons, Inc.; 1981; p. 217. [↑](#footnote-ref-6)
7. Alan G. Konheim; Cryptography: A Primer; John Wiley & Sons, Inc.; 1981; p. 212. [↑](#footnote-ref-7)
8. One can obtain US Patents over the Internet by logging onto the following URL: <http://www.uspto.gov/old-index.html> . From the home page, one selects SEARCH from PATENTS on the LHS (Left Hand Side) of the page. On the next page, one selects QUICK SEARCH. On the next page, one selects PATENT NUMBER for FIELD #1 and enters the actual patent number in the TERM #1 field. A text only version of the desired patent will appear. To obtain the actual patent with all its drawing, references, etc. in a .tif file format (TIFF = Tagged Image File Format), one must install a special reader plug-in in your Internet browser that is compatible with .tif files with the ITU (International Telecommunications Union) T.6 or CCITT (Comité Consultatif International Télégraphique et Téléphonique) Group 4 (G4) compression, e.g., the AlternaTIFF plug-in available, free, from <http://www.alternatiff.com/> .

   Note, due to bandwidth constraints, the US Patent & Trademark office forces you to download the patent one (1) page at a time, i.e., if the patent consists of N pages, then you will have N TIFF files. In order to convert the N TIFF files into a single multipage TIFF file, in Microsoft Windows XP OS (Operating System) you can use the Microsoft Office Document Imaging software located in your Microsoft Office folder. Run the Microsoft Office Document Imaging software and in the FILE pulldown menu select OPEN, and open the TIFF file containing the last page of the US Patent. Next, from the FILE pulldown menu select INSERT FILE, and select the TIFF file containing the next-to-last page of the US Patent. After you click the INSERT radio button, the INSERT FILE popup window will appear, just click the OK radio button. Continue clicking INSERT FILE in the FILE pulldown menu and inserting the next TIFF file. Keeping this up until you have reached and inserted every page of the US Patent. At this point, one can go to the FILE pulldown menu and select SAVE AS and create the multipage TIFF containing all the pages of the US Patent. Note, if the patent has many pages, the above manual procedure for generating a multipage TIFF file will become error prone and very time consuming. An alternative is to automate the process by using Microsoft Windows Script to essentially batch process the concatenation of the TIFF files, see <http://msdn.microsoft.com/en-us/library/ms950396.aspx> .

   Because the Microsoft Office Document Imaging software is not included with Microsoft Office 2010 and it is not bundled with Microsoft Windows 7 OS, one must use a workaround to install it. Follow the instructions contained in the following Microsoft URL: http://support.microsoft.com/kb/982760 in order to load the Microsoft SharePoint Designer 2007 software suite, which contains the Microsoft Office Document Imaging software. [↑](#footnote-ref-8)
9. Starring: Dougray Scott, Kate Winslet, Jeremy Northam, Safron Burrows; Directed: Michael Apted; Screenplay: Tom Stoppard; Based on the book by Robert Harris; Enigma; Manhattan Pictures International, LLC; 2002; 119 minutes; [↑](#footnote-ref-9)
10. Ivan Niven, Herbert S. Zuckerman; An Introduction to the Theory of Numbers; John Wiley & Sons, Inc.; 1972; pp. 20-25.

    Allan M. Kirch; Elementary Number Theory: A Computer Approach; Intext Educational Publishers; 1974; pp. 55-61. [↑](#footnote-ref-10)
11. Paul Deitel, Harvey Deitel; C++, How to Program, 8th Ed.; Prentice Hall; 2012; pp. 180.

    According to Deitel & Deitel,

    Variable names are said to be *lvalues* (for “left values”) because they can be used on the left side of the assignment operator [=]. Constants are said to be *rvalues* (for “right values”) because they can be used on only the right side of an assignment operator. *Lvalues* can also be used as *rvalues*, but not vice versa. [Deitel & Deitel 2012, p. 180] [↑](#footnote-ref-11)
12. Irving M. Copi, Carl Cohen, Kenneth McMahon; Introduction to Logic, 14th Ed.; Prentice Hall; 2011; pp. 300-308.

    Note, this textbook is used at FCC for the course PH206 – Logic. [↑](#footnote-ref-12)
13. Jonathan Wong; Enigma Simulator; Master’s Thesis, Department of Computing, City University, London EC1V OHB; 20xy; 52 pp; URL: <http://fortunegreen.co.uk/Final/Project.pdf> . [↑](#footnote-ref-13)
14. Chris Christensen; The Evolution of the Cryptographic Bombe; Department of Mathematics, Northern Kentucky University; 2/25/2010; 70 pp.; URL: <http://www.nku.edu/~christensen/091hnr304%20Enigma%20and%20the%20bombe.pdf> . [↑](#footnote-ref-14)