BUILDING A BOMBE

The Mathematics and Circuitry that Bested German Encryption

Jonah Weinbaum

Contents

Chapter 1

The UK Bombe

Section 1.1

Motivating Example

Section 1.2

Changes to Enigma

Starting in 1940, the German's enhanced the security of their key distribution. As discussed in CITE the *Grundstellung* rotor position was sent along with the daily key and an operator chose a *Spruchschlusse* to encode twice at the start of a message. Later iterations of this protocol removed the *Grundstellung* from key sheets.

These new key sheets contained the following columns columns *Tag/Datum*, *Walzenlage*, *Ringstellung*, *Steckerverbindungen*, and *Kenngruppen*

Notice the removal of the *Grundstellung* as well as the addition of the *Kenngruppen.* The *Kenngruppen* were a set of four trigrams used to identify which setting was being used to encode a message, this is particularly useful if trying to decode a message using a prior day's key. The operator would choose a trigram from the the *Kenngruppen*, append two letters to the front of the trigram, and this five letter combination (known as the *Buchstabenkenngruppe*) would preceed the message being sent. If a message was sent in multiple segments, multiple *Buchstabenkenngruppe* were used to start each segment.

When sending a message the operator was to use the following protocol

I. The time at which the message was sent is listed

II. The number of parts which the message contained is listed

III. Which message part is being sent is listed

IV. The length of the message part (not including *Buchstabenkenngruppe*) is listed

V. A *Grundstellung* rotor position is chosen and listed

VI. A *Spruchschlüssel* rotor position is chosen and encoded using the *Grundstellung*, this is listed

VII. The *Buchstabenkenngruppe* is listed

VIII. The message part encoded using the daily key and the *Spruchschlüssel* position is listed

It is clear that with this protocol, the Polish Bomba could no longer deduce the necessary details to decrypt Enigma messages. All of the permutation information contained in the original key distribution protocol was removed and a new method needed to be derived for infering information about the daily key.

Motivating Example

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

$$\begin{tikzcd}
D \arrow[d, "1"'] & Y \arrow[d, "2"'] & Y \arrow[d, "3"'] & Y \arrow[d, "4"'] & Y \arrow[d, "5"'] & Y \arrow[d, "6"'] & X \arrow[d, "7"'] & Y \arrow[d, "8"'] & Y \arrow[d, "9"'] & Y \arrow[d, "10"'] & X \arrow[d, "11"'] \\
A & B & R & A & C & A & D & A & B & R & A
\end{tikzcd}$$

where the top row indicates our enciphered message, the bottom row indicates the plaintext, and then indices on the arrows indicate how many steps forward our Enigma machine has moved while enciphering this message. Our goal is to determine which Enigma settings were used to encipher the message. In order to achieve this, we will examine which settings maintain the relationships between the enciphered and plaintext letters.

For example, any setting which maintains the above pairing must encipher A to D from the first position of the machine, then at the seventh position, it must encipher D to X, and at the eleventh position X must be enciphered back to A. It follows that if we had three Enigma machines connected in series, with an offset of 1, 7, and 11, from our initial position, then inputting A on the first machine would result in an ouput of A on the third machine. We visualize this loop as follows

$$\begin{tikzcd}
 & A \arrow[dl, <->, bend right, "1"'] \arrow[dr, <->, bend left, "11"] & \\
D \arrow[rr, <->, bend right, "7"'] & & X
\end{tikzcd}$$

To express this mathematically we denote the permutation represented by the Enigma at position i as σ_i. Since these each use the same plugboard we will also note the Enigma at position i not using the plugboard as $\overline{\sigma_i}$, that is $\sigma_i = P\overline{\sigma_i}P$ (conversely, $\overline{\sigma}_i = P\sigma_i P$). Then our loop is expressed by the fact that $\sigma_{11} \circ \sigma_7 \circ \sigma_1$ has a fixed point at A. We also note that the intermediate plugboard settings cancel out, that is

$$\begin{aligned} \sigma_1 \circ \sigma_7 \circ \sigma_{11} &= P\overline{\sigma_1}P \circ P\overline{\sigma_7}P \circ P\overline{\sigma_{11}}P \\ &= P \circ \overline{\sigma_1} \circ \overline{\sigma_7} \circ \overline{\sigma_{11}} \circ P \end{aligned}$$

We will condense this notation by defining

$$\sigma \coloneqq \sigma_1 \circ \sigma_7 \circ \sigma_{11}$$

and

$$\overline{\sigma} \coloneqq \overline{\sigma_1} \circ \overline{\sigma_7} \circ \overline{\sigma_{11}}$$

And thus we have shown $\sigma = P\overline{\sigma}P$ (conversely, $\overline{\sigma} = P\sigma P$).

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

$$\begin{aligned} \overline{\sigma}^i(\alpha) &= P \circ \sigma^i \circ P(\alpha) \\ &= P \circ \sigma^i(A) \\ &= P(A) \end{aligned}$$

and so we derive

$$P(A) = \alpha \Rightarrow P(A) = \overline{\sigma}^i(\alpha) \ \forall \ i \in \mathbb{N}$$

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

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

$$|\langle\overline{\sigma}\rangle \cdot \alpha| > 1 \Rightarrow P(A) \notin \langle\overline{\sigma}\rangle \cdot \alpha$$

thus eliminating several elements that A could be steckered to.

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

$$\overline{\sigma} = (ABCDEF)(GHIJK)(L)(MNOPQRSTUVWXYZ)$$

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

1.2.1. Scanning Methods

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

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

(b) If we perform single line scanning in sequence, that is, for each steckering hypothesis, we can rule out rotor positions which have all steckering hypotheses invalid. Turing called this method **serial scanning**.

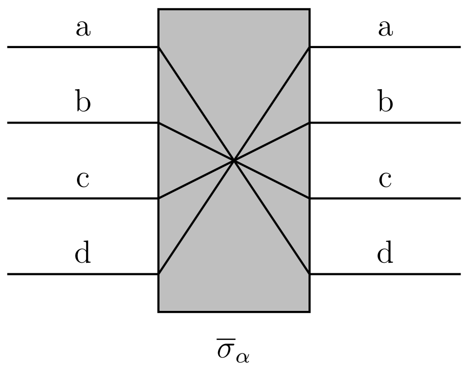
(c) Serial scanning requires a seperate examination of each steckering. Turing proposed a machine which could concurrently examine all steckering possibilities and eliminate rotor positions which had no valid steckerings. Turing called this method **simultaneous scanning**.

(d) If we find $\overline{\sigma}$ has a 26-cycle, then we must have that there are no 1-cycles and thus no valid steckerings. It then follows that the rotor position is incorrect. If we mechanism this process we can eliminate *some* rotor positions which do not have valid steckerings. We will call this method **spider scanning**. Note, however, that this method would not, for example, detect that a $13, 13$-cycle contains no valid steckerings. As Turing explained, "The ideal machine that Welchman was aiming at was to reject any position in which a certain fixed-for-the-time Stecker hypothesis led to any direct contradiction... The spider does more than this in one way and less in another. It is not restricted to dealing with one Stecker hypothesis at a time, and it does not find all direct contradictions." Effectively, spider-scanning is like a form of simultaneous scanning which is restricted to examining only one cycle at a time.

Iterations of each scanning methods were proposed or designed, but in the end we find that the spider scanning method was used in the implementation of the Bombe.At a high level, we encode the structure of the ciphertext-plaintext pairings, we input a hypothesis that a particular letter (e.g A) is steckered to another (e.g. α), and we electrically produce the elements that must also be steckered to our test letter (e.g. $\langle\overline{\sigma}\rangle \cdot \alpha$) if our hypothesis were to be true. If we find that this production generates a set of all letters and thus disqualifies this rotor position from producing the plaintext we observed, then we can continue on to the next rotor position until we have no contradictory results from our hypothesis.

Section 1.3
The Bombe

In the section, we outline the construction of a rudementary Bombe. Our goal is to construct a machine using the above insight to quickly elimate a rotor position based on contradictory hypotheses. For clarity sake, we will imagine that our Engima machine operates on an alphabet of only four letters $\{A, B, C, D\}$. We must shift from the above mathematical construction to an electrical one. We first abstract the idea of the Engima machine and do away with the input output system of keys and lamp lights. Rather, we imagine the Enigma machine as black-box that takes in 4 cables encoding, via current, each of the 4 letters, and outputs currents on the corresponding letter after applying the Enigma permutation. Imagine the following set of wires encoding a possible Enigma permutation we will denoted $\overline{\sigma}_\alpha$ (the bar indicates we are not yet considering the plugboard).



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

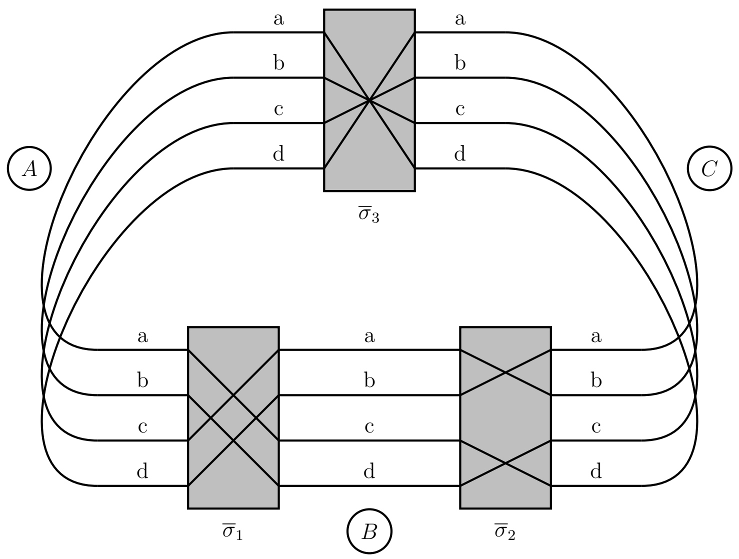
As described in our motivating example, we may want to connect Engima permutations in series to capture an underlying relationship between a ciphertext-plaintext pairing. Suppose we had plaintext ciphertext pairing

$$\begin{tikzcd}
\text{B} \arrow[d, "1"'] & \text{C} \arrow[d, "2"'] & \text{A} \arrow[d, "3"'] \\
\text{A} & \text{B} & \text{C}
\end{tikzcd}$$

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

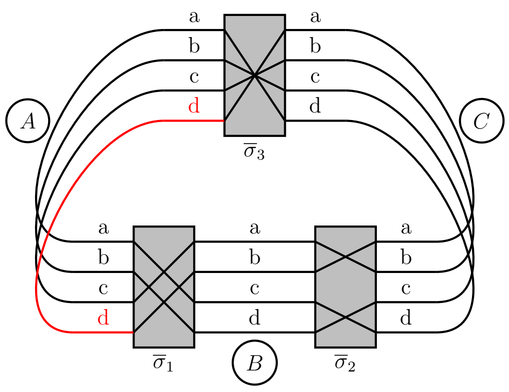
$$\begin{tikzcd}
& A \arrow[dl, leftrightarrow, bend right, "1"'] \arrow[dr, leftrightarrow, bend left, "3"] & \\
B \arrow[rr, leftrightarrow, bend right, "2"'] & & C
\end{tikzcd}$$

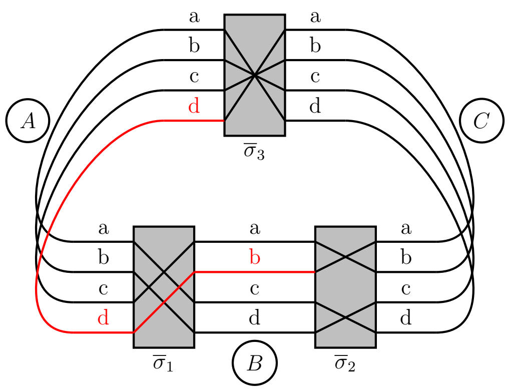
We can then think of this loop as a series of Enigma permutations

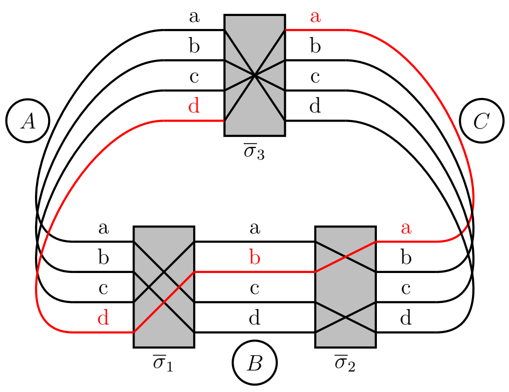


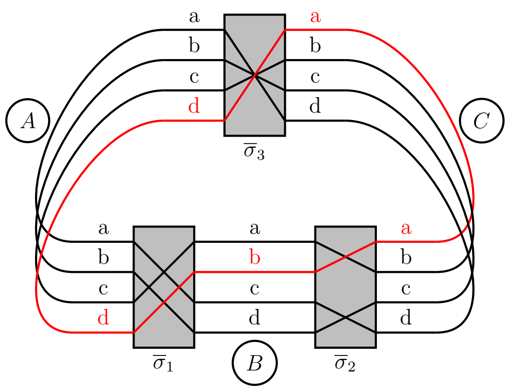
As before, let us make a hypothesis regarding steckering. Suppose $P(A) = D$. We know that A must map to A via $P\overline{\sigma}_3\overline{\sigma}_2\overline{\sigma}_1 P$. In our diagram, however we do not see a plugboard so how can we represent the plugboard's involvement? In order to achieve this *we* will function as the plugboard by applying voltage to the d line on the A thus implicitly performing the plugboard mapping in which $P(A) = D$. Then this will go through the electrical mapping $\overline{\sigma}_1$ arriving on the B cable, followed by $\overline{\sigma}_2$ arriving on the C cable, followed by $\overline{\sigma}_3$ arriving back on the A cable, where *we* as the implicit plugboard know that the output of this electrical mapping must go through the pluboard again to get a final output of A.

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

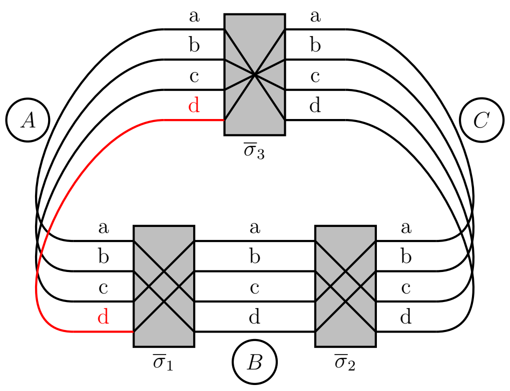

a b c d
a b c d
A
C
$\overline{\sigma}_3$
a b c d
a b c d
a b c d
$\overline{\sigma}_1$
B
$\overline{\sigma}_2$

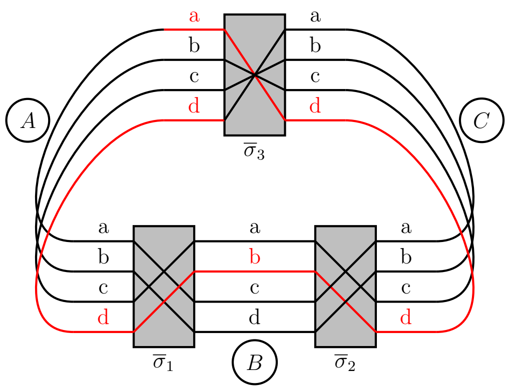

a b c d
a b c d
A
C
$\overline{\sigma}_3$
a b c d
a b c d
a b c d
$\overline{\sigma}_1$
B
$\overline{\sigma}_2$


a b c d
a b c d
A
C
$\overline{\sigma}_3$
a b c d
a b c d
a b c d
$\overline{\sigma}_1$
B
$\overline{\sigma}_2$

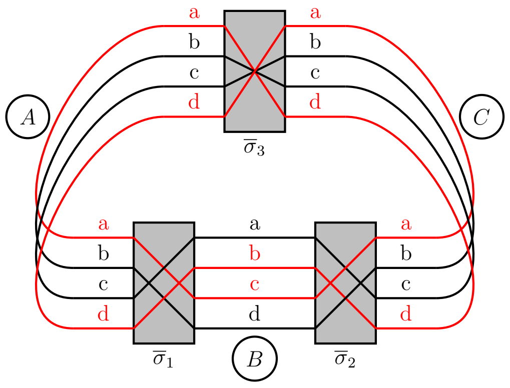

a b c d
a b c d
A
C
$\overline{\sigma}_3$
a b c d
a b c d
a b c d
$\overline{\sigma}_1$
B
$\overline{\sigma}_2$

Following our hypothesis that $P(A) = D$ we find that after sending current through Ad we arrive after mapping through $\overline{\sigma}$ back at Ad and this makes sense since we know that D will map back to A thus preserving the loop in our ciphertext-plaintext pairing. On the other hand, if we change $\overline{\sigma}$ and repreat this process which may end up in the following sitatuation





In this example, following our steckering hypothesis $P(A) = D$, results in Aa becoming live after mapping through $\overline{\sigma}$ and thus we must have $P(A) = A$ as well in order for this to ultimately map back to A and preserve our ciphertext-plaintext loop above. Thus we can eliminate the steckering possibilities that $P(A) \in \{A, D\}$. Effectively, this process is a mechanization of finding the cycle containing a particular letter. In the above diagram sending current from Ad until we arrive back at Ad is akin to repeatedly applying $\overline{\sigma}$ to D until we return to D. This can be visualized in our diagram as



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

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

We can also near instantaneously determine which arrangements to eliminate with spider-scanning. When an arrangement is such that it cannot be eliminated and further examination is needed, we will cause the machine to stop.

To see how this is done, we will illustrate how we can detect when a line say Ab is no longer live. We can form a differential relay such that it only engages when a current difference is present between *Ab* and some constant power supply line. Then, when current stops flowing through *Ab*, the relay will trigger. We can then wire the stopping mechanism to the contact terminal of the relay such that the stopping mechanism will only trigger when line *Ab* is not live (i.e. the relay is closed).

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

This begs the question of where to place this detecting circuit. If we are in the situation described above, a loop of Enigma permutations, then this choice does not matter. This is because if a 26-cycle is present in $\overline{\sigma}_3\overline{\sigma}_2\overline{\sigma}_1$, it must also be present in $\overline{\sigma}_2\overline{\sigma}1\overline{\sigma}_3$ and $\overline{\sigma}_1\overline{\sigma}_2\overline{\sigma}_1$. This is because all of these permutations are conjugates of one another, for example we have

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

Since permutations in the same conjugacy class must have the same cycle type, it follows that no matter where in the loop we place our detector, if all wires become live they will become live anywhere else we could place the detector. While this may seem a trivial observation we have actually stated something much stronger here which is that the cycle types of the permutation will not change no matter where we begin our observation from. This result will become relevant later in our analysis of the probability of triggering the stopping mechanism. (MAY NOT BE RELEVANT IF I DONT INCLUDE DIAGONAL ANALYSIS)

1.3.1. Diagonal Board

Chapter 2

Simulating the Bombe

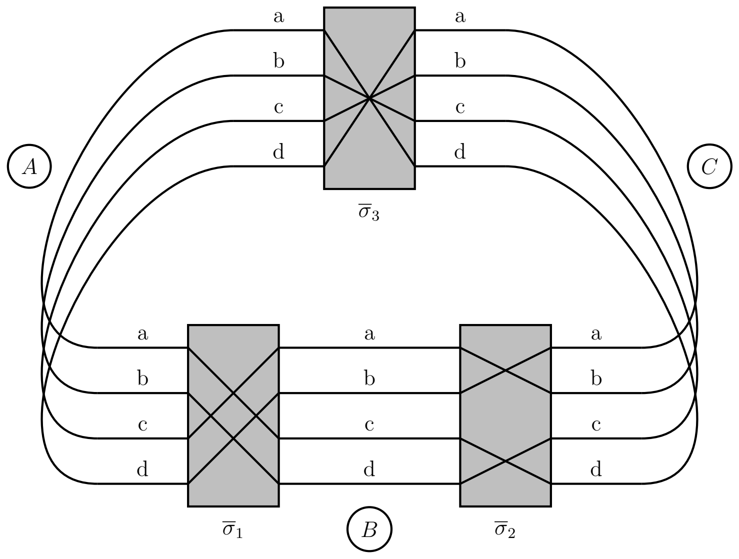
Chapter 3

Number of Stops

In this chapter we will derive the expected number of stops for particular arrangments of the machine. The machine is wired to stop when $\overline{\sigma}$ has cycle type other than (26). Turing only considers what he calls **normal stops** during his calculation of the expected number of stops. This is a stop which has cycle-type $(25, 1)$. We will attempt to expand to a consideration of all possible stops.

3.0.1. Prior Work

3.0.2. Stops Without Diagonal Board



Consider our simple example of a loop of three Enigmas on four letters. We might expect that $\overline{\sigma} = \overline{\sigma}_3\overline{\sigma}_2\overline{\sigma}_1$ being generated from considerably random permutations, is itself a random permutation. If this is the case then we would expect that we would get a (4) cycle with a probability of $\frac{1}{4}$. Then we expect the machine to stop with probability $\frac{3}{4}$. With enough loops this probability decreases exponentially and the machine has a tractible number of stops. However, try as we may, we can never find a collection of Enigma permutations $\{\overline{\sigma}_1, \overline{\sigma}_2, \overline{\sigma}_3\}$ which generate a (4) cycle in $\overline{\sigma}$. This is to say, in our above arrangment, the machine will stop at *every* rotor position thus making the process of checking stops intractible.

To see why this is the case, note that each $\overline{\sigma}_i$ has cycle type $(2, 2)$ thus they are permutations of even parity. On the other hand, any (4) cycle will have odd parity. When we compose 3 even permutations (i.e. $\overline{\sigma}_3\overline{\sigma}_2\overline{\sigma}_1$) we will always get an even parity permutation, thus this resulting permutation can *never* be a (4) cycle.

In the case of the Bombe, a cycle of even length can never produce a permutation with a (26) cycle. We can emperically observe this by simulating the Bombe's operation on a cycle of length 8 and we find that every single rotor position produces a stop.

From the above it is clear that $\overline{\sigma}$ is certainly not a purely random permutation, and simulations of loops of Enigma permutations of various lengths show that the probability distribution of these permutations is highly dependent on the length of the loop. A table for estimated probabilities for Enigma machines on 4 letters is shown below.

NOTE THAT WE ARE ASSUMING AN ENIGMA MACHINE IS A RANDOM

2,2... CYCLE

Given how skewed the distribution of permutations are for each cycle length it follows naturally that to express the probability of a stop with a particular machine arrangement should not just be a function of the number of closures or letters in a menu, but rather the particular length of each closure.

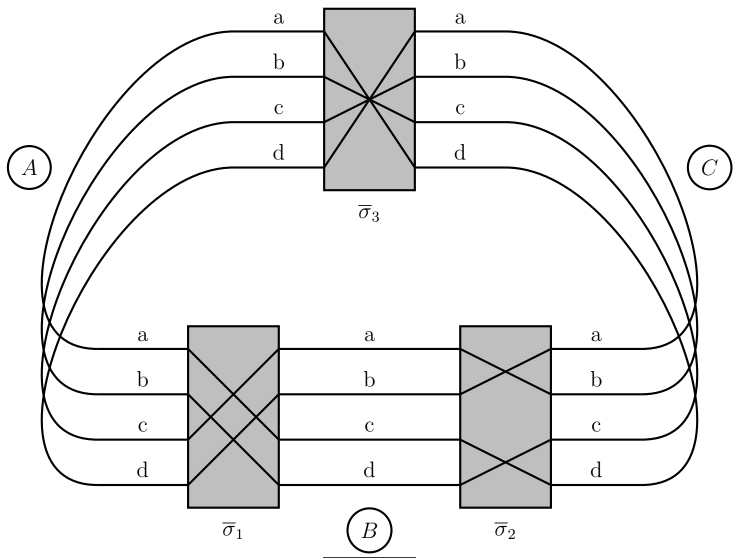
Singular Loop.

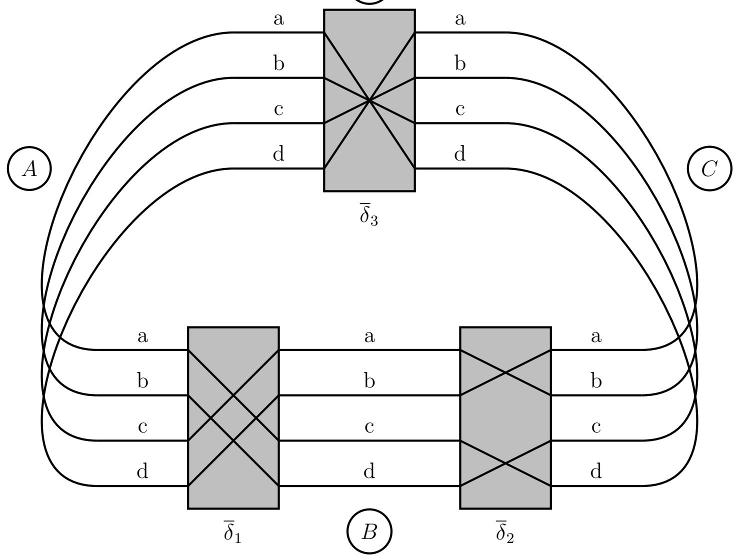
For the case of a single loop we can run a simulation to estimate the probability of a stop given the length of a loop. This is akin to the table above though we combine the total probabilities of any cycle type other than (26) to get a probability of a stop for each length of loop.

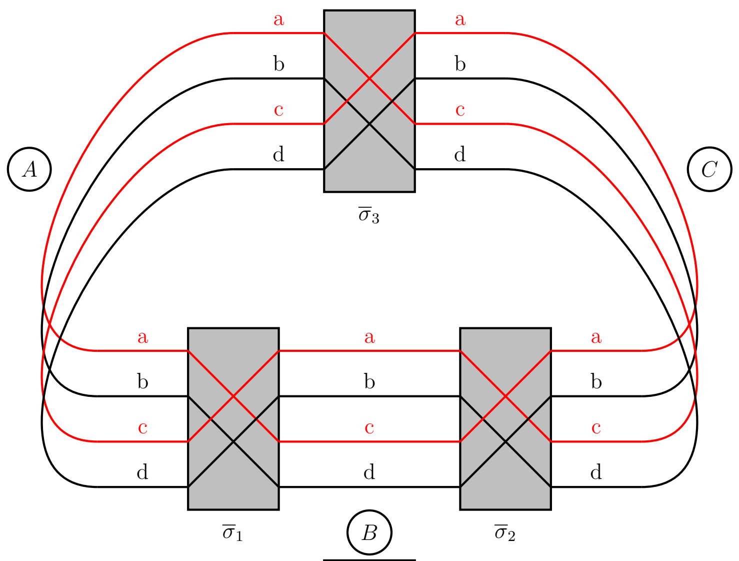
Multiple Loops.

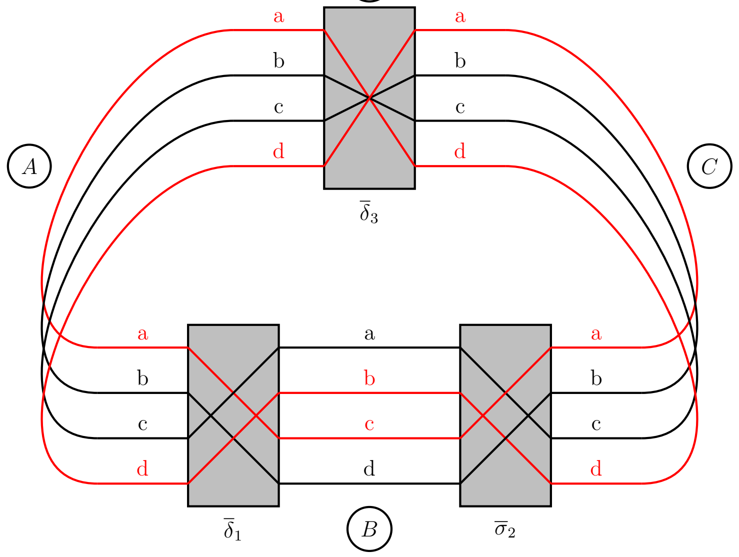
Multiple loops presents some complexities. Initally one might suspect that these loops are independent, and while the cycle type probabilities of each loop may be independent, due to the electrical interconnections between the loops we need to take more into account. For example, we have noted that an even loop length of Enigma machines will always stop. However, two loops of odd length may result in a configuration that will not cause a stop. To see this consider our example on 4 letters. In this case, an odd length loop can never generate a (4) cycle. Consider now the following loops of Enigma permutations representing permutations $\overline{\sigma}$ and $\overline{\delta}$ respectively.

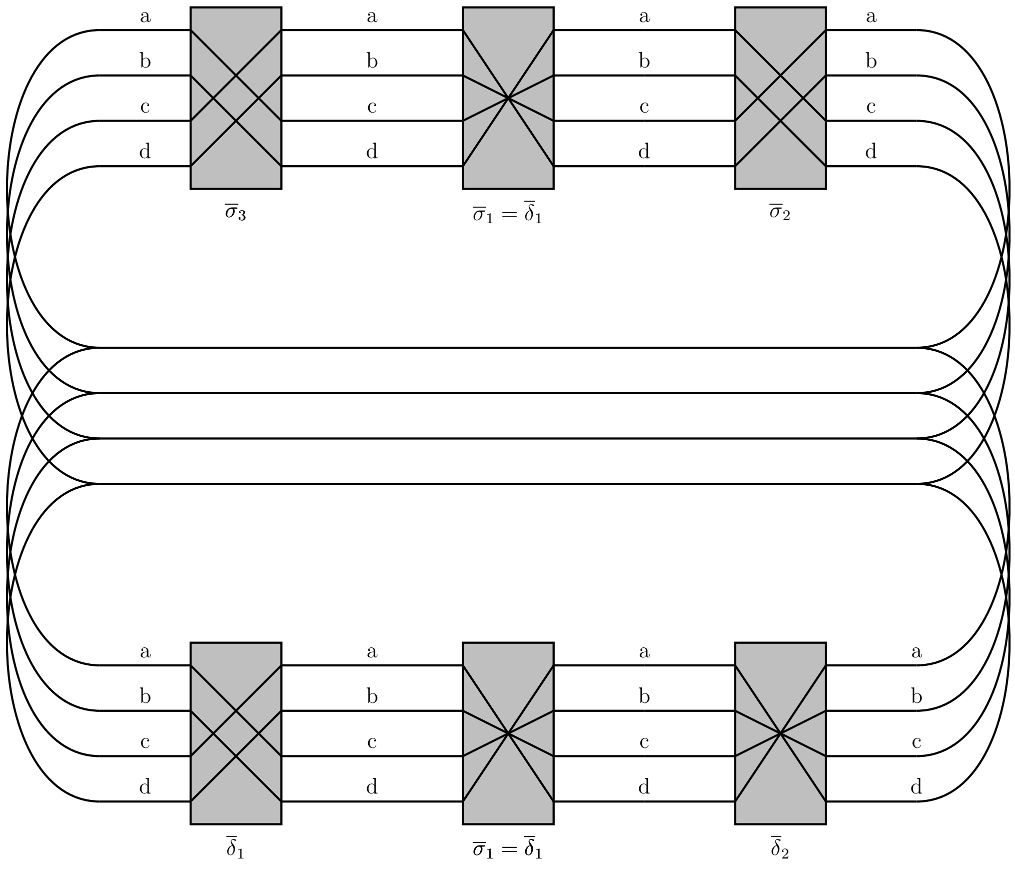
3.0.3. Introducing the Diagonal Board






a
b
c
d
a
b
c
d
A
C
$\overline{\sigma}_3$
a
b
c
d
a
b
c
d
a
b
c
d
$\overline{\sigma}_1$
B
$\overline{\sigma}_2$


a
b
c
d
a
b
c
d
A
C
$\overline{\delta}_3$
a
b
c
d
a
b
c
d
a
b
c
d
$\overline{\delta}_1$
B
$\overline{\sigma}_2$


a
b
c
d
$\overline{\sigma}_3$
$\overline{\sigma}_1 = \overline{\delta}_1$
$\overline{\sigma}_2$
$\overline{\delta}_1$
$\overline{\sigma}_1 = \overline{\delta}_1$
$\overline{\delta}_2$