CMSC 28400 Introduction to Cryptography **Autumn 2021**

Notes #2: Permutations and Enigma

Instructor: David Cash

Contents

These notes cover the permutation-based attack of Rejewski against the early Enigma machines, along with some mathematical background used in the attack.

2.1 The Simplified Enigma Machine

We will study a somewhat simplified version of the Enigma machine used in early 1930s. In Figure 2.1 two versions of the Enigma are pictured. The relevant parts of the machine are a keyboard, a light-board, a plug-board, and a rotor assembly consisting of three rotors along with a special *reflector* (not visible). The rotors are electrical implementations of a substitution cipher on $\Sigma = \{\mathtt{A}, \ldots, \mathtt{Z}\}$: They have 26 connections (with labels) on each side, and in between is a mess of wires that physically permutes how they are connected. The rotors are designed to be chained together, and also so that they can rotate to change how their 26 connections line up. The rotors are also designed so their their rotational position is externally visible after they are placed in the machine, and they can be manually rotated in discrete $1/26^{\text{th}}$ steps.

An Enigma user would actually have several rotors stored in a box, but only three would be used at a time (in this version; other rotor-based ciphers and later Enigmas had more than three rotors).

To operate these machines, one would start by configuring the machine according to the *day key* which were distributed to everyone in order for them to communicate. These settings included which rotors to use, how to align them with respect to the rest of the machine, which connections to make on the plugboard, as well as something called the *ring setting* that we will ignore.

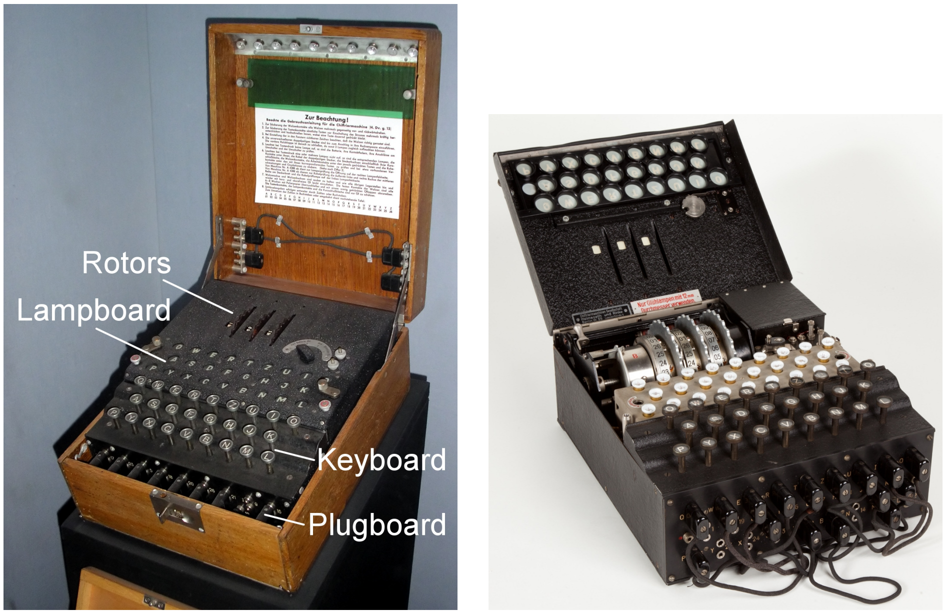


Figure 2.1: Two images of Enigma. Credit: Left image by Karsten Sperling (public domain, downloaded at `https://commons.wikimedia.org/wiki/File:EnigmaMachineLabeled.jpg`, Right image by Alessandro Nassiri (`https://creativecommons.org/licenses/by-sa/4.0/deed.en`, no changes.)

Once configured, the operation is simple: To encipher a letter one presses a key on the keyboard. A electric signal passes from the keyboard through the plugboard, three rotors, the reflector, and then through the rotors again and the plugboard again before lighting up a letter on the lightboard, which is the enciphered letter. This is partially diagrammed in top part of Figure 2.2, which is drawn as if the Enigma does not have a plugboard. We note that the reflector is always wired to avoid ever mapping a letter to itself.

The force of pressing the key would also rotate the rotors. The right-most "first" rotor will always rotate, and the other two rotors will rotate on a schedule depending on the positioning of a physical catch between the rotors. In these notes, we'll work under the assumption that only the first rotor moves. The effect of a rotating on the electrical connections is diagramed in the bottom of Figure 2.2.

2.1.1 Message Keys and Rejewski's Attack

It was recognized that if everyone were to set their Enigma to same day key and then encipher messages, than basic frequency analysis would defeat the encryption (since the same permutation would be applied to i^{th} letter by everyone).

In the early usage of the Enigma, this was mitigated by using *message keys*. The sender would choose a random three-letter message key xyz. The sender would then begin the transmission

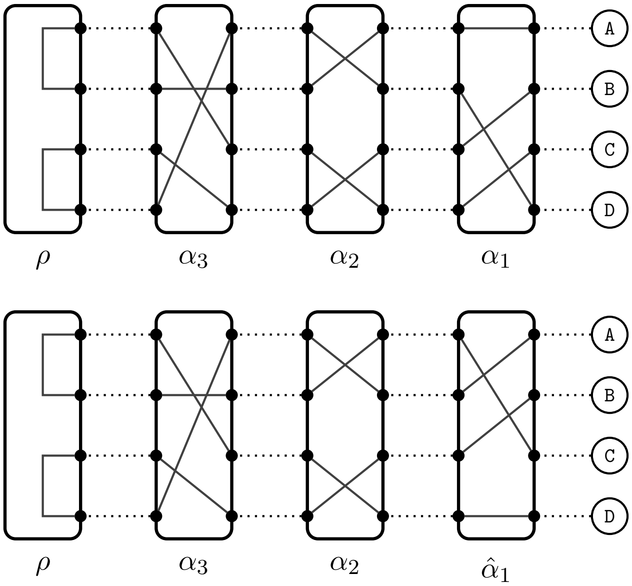


Figure 2.2: A diagram of the electrical connections in an Enigma without a plugboard, for the alphabet $\Sigma = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}\}$. The top diagram shows an initial configuration, while the bottom shows the configuration after the right rotor rotates by one step (upwards in the diagram). The labeling will be discussed later.

by enciphering the letters xyz *twice*. After this, the sender would manually rotate the rotors to positions x, y, z, and then encipher the payload of the message as before.

Example 2.1. *Suppose the sender chooses message key* `TAQ`*. The sender enciphers the messages twice using the day key; Suppose this produces the six letters* `ICPWLV`*. The sender then resets the three rotors to positions* `T`,`A`*, and* `Q`*, and then enciphers the rest of the message.*

The idea was that message keys would provide enough randomization to defeat frequency analysis. However, it did not work against more clever attacks. Specifically, we will show that given many ciphertexts (about fifty in practice), a clever attack easily recovers all of the message keys! Thus if one know the day keys (say by stealing them), then all messages can be decrypted.

2.2 Permutations and Cycle Decompositions

Our first order of business will be to give a clear and useful mathematical description of an Enigma machine. The most useful language for doing so turns out to be *permutations*, which we turn to in the next section before returning to the attack later.

Definition 2.1. *A* permutation on a set Σ *is a one-to-one and onto function from* Σ *to itself.*

Recall that *one-to-one* means that $x \neq y$ implies $\pi(x) \neq \pi(y)$, and *onto* means that for every $y \in \Sigma$ there exists $x \in \Sigma$ such that $\pi(x) = y$.

Here are two examples of permutations on the set $\Sigma = \{1, 2, 3, 4, 5, 6\}$. They are presented in "table form", so given x we can simply look up $\pi(x)$.

x	1	2	3	4	5	6
$\pi(x)$	4	1	6	5	2	3

and

x	1	2	3	4	5	6
$\sigma(x)$	2	1	5	4	6	3

Exercise 2.1. *Suppose Σ is finite. Show that any one-to-one function $\pi : \Sigma \to \Sigma$ is a permutation. Do the same for any onto function.*

Show that neither of these hold in general when Σ is infinite by finding a counterexample. (We won't need permutations on infinite sets again in this class.)

We will write $\pi\sigma$ for the composition of π and σ (where we mean: "first apply to σ, then apply π"). So in table notation $\pi\sigma$ is

x	1	2	3	4	5	6
$\pi\sigma(x)$	1	4	2	5	3	6

If $\pi\sigma = \sigma\pi$ then we say that π *and* σ *commute.*

We write π^{-1} for the inverse of π; You can prove that π^{-1} is also a permutation. The inverse π^{-1} for our example is given in table form as:

x	1	2	3	4	5	6
$\pi^{-1}(x)$	2	5	6	1	4	3

For any permutation π, the product $\pi\pi^{-1}$ is always the *identity permutation* $e : \Sigma \to \Sigma$ defined by $e(x) = x$.

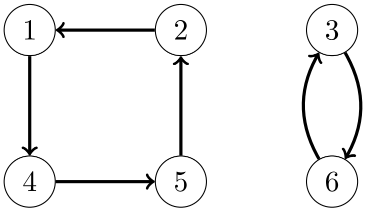
Exercise 2.2. *Show by induction on t*

$$(\pi_1 \cdots \pi_t)^{-1} = \pi_t^{-1} \cdots \pi_1^{-1}$$

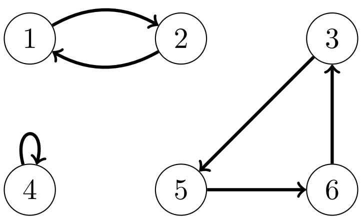
for any permutations $\pi_1, \ldots, \pi_t$.

2.2.1 Cycles in Permutations

The table setting is intuitive, but the real structure of permutations is revealed if we use them to create a graph, as follows. Let π be a permutation on Σ, and define a directed graph $G_\pi = (V, E)$ with vertex set $V = \Sigma$ and an edge from each $x \in \Sigma$ to $\pi(x)$. For example, here is the graph G_π created from the table above:



And here is the graph for G_σ:



What these graphs have in common is that *every node is a member of exactly one directed cycle* (which may be a self loop). Indeed this will be true for *any* permutation on a finite set: Start from any node and iteratively apply the permutation to walk on nodes. Since the set is finite, you must eventually repeat a node. Since the permutation is one-to-one, the first repetition must bring you back to where you started. Starting at another node not on this cycle will generate yet another cycle, and so on. The upshot is: *There is a one-to-one correspondence between permutations on Σ and graphs that consist of a covering of all of Σ with disjoint directed cycles.*

This structure motivates the following notation:

Definition 2.2. *We write $(a_1\ a_2\ \cdots\ a_t)$, where all of the $a_i \in \Sigma$, for the permutation on Σ that maps a_1 to a_2, a_2 to a_3, etc, a_t to a_1, and for the rest of the $x \in \Sigma$ maps x to itself.*

We use this notation combined with our product notation. So for example $\pi = (1\ 4\ 5\ 2)(3\ 6)$ is another way of writing π from above, which can be checked by hand (and of course, the notation still means that we apply the permutation $(3\ 6)$ first, and then permutation $(1\ 4\ 5\ 2)$).

Exercise 2.3. *Call two cycles $(a_1\ a_2\ \ldots)$ and $(b_1\ b_2\ \ldots)$* disjoint *if the sets $\{a_1, a_2, \ldots\}$ and $\{b_1, b_2, \ldots\}$ are disjoint. Show that disjoint cycles commute.*

Permutations can be written as a product of cycles in several ways. For example, we can check that $(1\ 4)(1\ 4\ 5\ 2) = (2\ 4\ 5)$. However, if we require that cycles be *disjoint*, then there is essentially only one way to express the permutation.

Theorem 1. *Every permutation on a finite set Σ can be written as a product of disjoint cycles. Moreover, this product is unique up to ordering the cycles and rotating elements within cycles.*

We will not prove this theorem formally, but if one accepts the equivalence between graphs of cycles and permutations then it is very intuitive; The cycles in the product expression of π correspond to the cycles of the graph, and the order of the cycles or how they are rotated does not affect the corresponding graph.

As an example of the theorem in action, $\pi = (1\ 4\ 5\ 2)(3\ 6)$ is a product of disjoint cycles, and

$$\pi = (1\ 4\ 5\ 2)(3\ 6) = (3\ 6)(1\ 4\ 5\ 2) = (6\ 3)(1\ 4\ 5\ 2) = (6\ 3)(5\ 2\ 1\ 4) = \ldots$$

and so on all are all equal π; These are the *only* ways to write π as a product of disjoint cycles. We shall refer to these ways (collectively) as *the* cycle decomposition of π, with the understanding that it is not exactly unique (only sort-of-unique).

It is important to remember that every element not appearing in a cycle is assumed to be fixed by the permutation; So in effect there are often extra unwritten cycles of length 1 hanging around.

Definition 2.3. *Let π be a permutation on Σ. We say that π has* type $[1^{z_1} 2^{z_2} \cdots n^{z_n}]$ *if for all $i = 1, \ldots, n$, the cycle decomposition of π has exactly z_i cycles of length i.*

When some $z_i = 0$ we omit i^{z_i} from the notation.

It takes a moment to verify that this definition is "well defined", which means that we if take *any* way of writing π as a disjoint product of cycles, and check their lengths, we will get the same type. For example, our π has type $[2^1 4^1]$, and it does not matter which ordering or rotations of the cycles we take.

Example 2.2. *Let $\Sigma = \{1, 2, 3, 4, 5, 6\}$.*

- *The permutation* $(1\ 2)(3\ 4)(5\ 6)$ *has type* $[2^3]$.
- *The permutation* $(1\ 2\ 3\ 4)$ *has type* $[1^2 4^1]$.
- *The permutation* $(1\ 4\ 3)(5\ 6)$ *has type* $[1^1 2^1 3^1]$.

2.2.2 Conjugates of Permutations

The structure of the Enigma machine can be simplified by introducing the concept of *conjugates* of permutations.

Definition 2.4. *Let π, σ be permutations on Σ. The* conjugate of σ by π *is defined to be the permutation $\pi\sigma\pi^{-1}$.*

If τ is the conjugate of σ by π for some π, then we say that τ is "a conjugate of σ."

Theorem 2. *Let σ have cycle decomposition $(a_1\ a_2\ \ldots)(b_1\ b_2\ \ldots)\cdots$. Then the conjugate $\pi\sigma\pi^{-1}$ has cycle decomposition*

$$(\pi(a_1)\ \pi(a_2)\ \ldots)(\pi(b_1)\ \pi(b_2)\ \ldots)\cdots.$$

In other words, conjugation by π acts on the cycle decomposition of σ by applying π the elements of the cycles.

Proof. We show that if $\sigma(x) = y$, then the conjugate $\pi\sigma\pi^{-1}$ maps the element $\pi(x)$ to $\pi(y)$. This is easy to check:

$$\pi\sigma\pi^{-1}(\pi(x)) = \pi(\sigma(x)) = \pi(y).$$

We can finish the proof by constructing the cycles of the conjugate step-by-step from the cycles of σ. □

Corollary 1. *A permutation σ has the same type as any of its conjugates.*

This relationship is actually if-and-only-if: If two permutations have the same type, then they are conjugates.

Exercise 2.4. *Prove the last sentence.*

2.3 Application to Enigma: Rejewski's Theorem

Now let's apply all of this to analyze an Enigma machine.

2.3.1 Permutation Structure of the Initial Setting

Looking at the top of Figure 2.2, we can see that the permutation computed by an Enigma machine in that initial configuration is

$$\Delta_1 = \alpha_1^{-1}\alpha_2^{-1}\alpha_3^{-1}\rho\alpha_3\alpha_2\alpha_1, \tag{2.1}$$

where α_i $(i = 1, 2, 3)$ and ρ refer to the labeled rotors in the diagram.

This looks complicated, but it becomes simpler if we let $\tau = \alpha_3\alpha_2\alpha_1$ be the effect of the three rotors going right-to-left. Then

$$\Delta_1 = \tau^{-1}\rho\tau,$$

so Δ_1 is the conjugate of ρ by τ^{-1} (it's by τ^{-1}. and not τ because the inverse appears on the left side instead of the right).

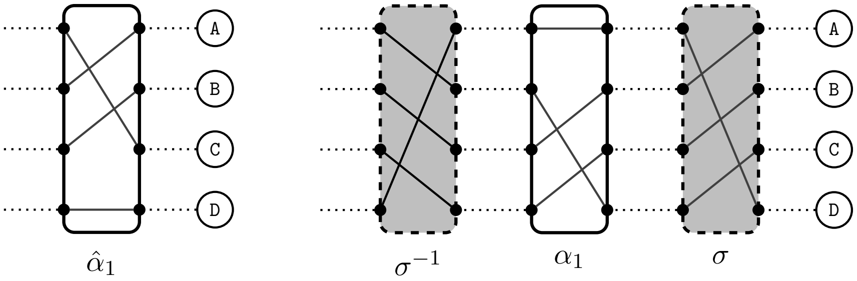


Figure 2.3: Writing the effect of rotation of a rotor as a conjugation by the "shift forward" permutation σ. The permutations diagrammed on the left and right are equal.

2.3.2 The Effect of a Rotor Rotation

You can visually see the effect of rotating the left rotor by looking at difference between the top and bottom of Figure 2.2. We now describe this in terms of permutations. Let Δ_1 be as above, and let Δ_2 be the permutation computed after rotating the left rotor (only). If we let $\hat{\alpha}_1$ be the permutation computed by the α_1 rotor after a rotation, then

$$\Delta_2 = \hat{\alpha}_1^{-1}\alpha_2^{-1}\alpha_3^{-1}\rho\alpha_3\alpha_2\hat{\alpha}_1, \tag{2.2}$$

that is, the same as Δ_1 but with $\hat{\alpha}_1$ instead of α_1. Again Δ_2 is a conjugate of ρ, and we only need to understand $\hat{\alpha}_1$.

Let σ be the shift-forward permutation, defined by $\sigma(\mathtt{A}) = \mathtt{B}, \dots, \sigma(\mathtt{Z}) = \mathtt{A}$. It is intuitive that the expression for $\hat{\alpha}_1$ should involve σ, but it might not be immediately clear exactly how. In Figure 2.3, and example $\hat{\alpha}_1$ is diagrammed along with how to produce α_1 from σ. The key is that rotation is equivalent to keeping α_1 fixed but *conjugating* it by σ^{-1}. You can check that the two sides of Figure 2.3 compute the same permutation. Intuitively, rotation is conjugating because the action on one side of the rotor (moving forward in the alphabet) induces the opposite action (moving backward in the alphabet) on the other side of the rotor. But if that doesn't make sense, perhaps the diagram is convincing.

We can continue with this thinking to Δ_i, the permutation computed on the i^{th} step. We get

$$\Delta_i = \sigma^{-i}\alpha_1^{-1}\sigma^{i}\alpha_2^{-1}\alpha_3^{-1}\rho\alpha_3\alpha_2\sigma^{-i}\alpha_1\sigma^{i}, \tag{2.3}$$

which is just a nasty-looking conjugate of ρ, but a conjugate nonetheless. Note that we have cut through a very complicated-looking machine and at least reduced it to a few equations.

2.3.3 The Types of Δ_i

The next key observation in understanding the behavior of an Enigma machine is that, at any given time, the machine will compute a permutation on the letters that has a specific type, namely $[2^{13}]$. In other words, the permutation computed will always be the product of 13 disjoint cycles of length 2. This will follow easily from the results of the previous section, plus the observation that the reflector ρ is wired to never map a letter to itself.

We generalize the type of permutation under consideration to any even size alphabet.

Definition 2.5. *Let $n = |\Sigma|$ be even. A permutation on Σ is called a* reflector *if it has type $[2^{n/2}]$.*

These permutations are more commonly called *proper involutions.* (A *(non-proper) involution* is a permutation with cycle decomposition consisting of cycles of length of 1 or 2, i.e. type $[1^{z_1}2^{z_2}]$ for some z_1, z_2.)

Lemma 1. *Let $|\Sigma|$ be even and ρ be a reflector on Σ. Then the following hold:*

- $\rho^{-1} = \rho$,
- *For all $x \in \Sigma$, $\rho(x) \neq x$.*

Exercise 2.5. *Prove this lemma. The reverse direction also holds; Prove that as well.*

Lemma 2. *Let Δ_i be the permutation computed by the simplified Engima machine at step i, as defined in Equation (2.1). Then Δ_i is a reflector.*

Proof. We showed above that Δ_i is a conjugate of ρ, and by Corollary 1, Δ_i has the same type as the reflector ρ, i.e. type $[2^{n/2}]$. Thus Δ_i is a reflector. □

2.4 Rejewski's Theorem and Attack

Rejewski's attack assumes it is given several ciphertexts generated with message keys chosen by senders; It will recover those message keys with only a little effort. In fact, the attack is efficient enough to do by hand!

The attack proceeds in three steps:

1. Use the ciphertexts to compute the entire tables (or cycle decomposition) of the permutations $\Delta_4\Delta_1, \Delta_5\Delta_2$, and $\Delta_6\Delta_3$.
2. Find all possible factorizations of $\Delta_4\Delta_1$, $\Delta_5\Delta_2$, and $\Delta_6\Delta_3$ into products of two reflectors.
3. From amongst the possible factorizations, figure out which ones are the correct ones, and hence learn $\Delta_1, \Delta_2, \Delta_3$. Now message keys can be recovered.

2.4.1 Step One: Learning $\Delta_4\Delta_1, \Delta_5\Delta_2$, and $\Delta_6\Delta_3$

This step is based on the following lemma:

Lemma 3. *Let ρ_1 and ρ_2 be reflectors on Σ. Then for $x \in \Sigma$, if $\rho_1(x) = y_1$, and $\rho_2(x) = y_2$, then $\rho_2\rho_1(y_1) = y_2$.*

Proof. Since ρ_1 is a reflector and $\rho_1(x) = y_1$, we have that $\rho_1(y_1) = x$. This can been seen in either of two ways: First, since ρ_1 is a reflector, $\rho_1^{-1} = \rho_1$, and

$$\rho_1(x) = y_1 \iff \rho_1^{-1}(\rho_1(x)) = \rho_1^{-1}(y_1) \iff x = \rho_1^{-1}(y_1) \iff x = \rho_1(y_1).$$

Alternatively, we could just observe that since ρ_1 is a reflector then it must have $(x\ y_1)$ in its cycle decomposition.

Finally, using $\rho_1(y_1) = x$ we get

$$\rho_2\rho_1(y_1) = \rho_2(x) = y_2,$$

as desired. □

Here is an example of how it is used against a single ciphertext:

Example 2.3. *Suppose we observe a ciphertext starting with* `ICPWLV`*. We know that there exist $x, y, z \in \Sigma$ such that*

$$\texttt{ICPWLV} = \Delta_1(x)\Delta_2(y)\Delta_3(z)\Delta_4(x)\Delta_5(y)\Delta_3(z),$$

where the Δ_i are unknown permutations computed by the Enigma machine. We do however know that they are all reflectors, by Lemma 2. Applying Lemma 3, we can infer that

$$\Delta_4\Delta_1(\texttt{I}) = \texttt{W}, \qquad \Delta_5\Delta_2(\texttt{C}) = \texttt{L}, \quad \textit{and} \quad \Delta_6\Delta_3(\texttt{P}) = \texttt{V}.$$

To learn all of $\Delta_4\Delta_1, \Delta_5\Delta_2$, and $\Delta_6\Delta_3$, the Polish team would look over many ciphertexts. Each ciphertext would tell them how one letter is mapped in each of the products; After enough time, they would typically see how all of the letters are mapped by each product. This completes step one.

2.4.2 Step Two: Factoring $\Delta_4\Delta_1, \Delta_5\Delta_2$, and $\Delta_6\Delta_3$

Now we assume we know the products, and want to recover the individual Δ_i. One approach is try all possible combinations of reflectors until we find a combination that composes to each one. Rejewski found a much faster way, based on the following theorem:

Theorem 3 (Rejewski)**.** *Let ρ_1 and ρ_2 be reflectors on Σ. Then if $(a_1\ a_2\ \cdots\ a_t)$ appears in the cycle decomposition of their product $\rho_2\rho_1$, the cycle*

$$(\rho_1(a_t)\ \rho_1(a_{t-1})\ \cdots\ \rho_1(a_1))$$

also appears in the cycle decomposition of $\rho_2\rho_1$, and moreover this cycle is distinct from $(a_1\ a_2\ \cdots\ a_t)$.

We will prove this theorem in the last section. For now we state an interesting corollary and then show how to apply the theorem.

Corollary 2. *Let ρ_1 and ρ_2 be reflectors, and let $[1^{z_1}2^{z_2}\cdots n^{z_n}]$ be the type of their product $\rho_2\rho_1$. Then all of the z_i are even.*

Exercise 2.6. *Prove the corollary.*

This lemma is applied once one has the cycle decomposition of $\Delta_4\Delta_1$ from the previous step. By the lemma, we know that the cycles must appear in pairs, where one cycle in the pair is the reverse of the other with Δ_1 applied to each element. If we knew exactly how the cycles were paired up, we could just read off the values of Δ_1. Unfortunately we *don't* know exactly how they pair up, but it will turn out that there aren't usually too many ways they could pair up. In the next step we'll figure out which is the correct one.

Example 2.4. *Suppose we have determined that*

$$\Delta_4\Delta_1 = (\texttt{OGKRYSD})(\texttt{ZUQWFIB})(\texttt{MJXCP})(\texttt{HLNVE})(\texttt{A})(\texttt{T}).$$

Here, the cycles of size one are written explicitly to emphasize their presence.

This is the type of cycle decomposition predicted by the Theorem: We have a pair of cycles of length 7*, a pair of size* 5*, and a pair of size* 1*. Since there are only two cycles of each length, we* *know how they must pair up. But we don't know how to cyclically align them as the theorem says. For instance, we may pair up*

$$(\texttt{OGKRYSD})(\texttt{ZUQWFIB}) = (\texttt{OGKRYSD})(\Delta_1(\texttt{D})\cdots\Delta_1(\texttt{G})\Delta_1(\texttt{O}))$$

and guess that $\Delta_1(\texttt{O}) = \texttt{B}, \Delta_1(\texttt{G}) = \texttt{I}$, etc. Alternatively, the cycles may also be written

$$(\texttt{OGKRYSD})(\texttt{UQWFIBZ}) = (\texttt{OGKRYSD})(\Delta_1(\texttt{D})\cdots\Delta_1(\texttt{G})\Delta_1(\texttt{O}))$$

in which case we would guess that $\Delta_1(\texttt{O}) = \texttt{Z}, \Delta_1(\texttt{G}) = \texttt{B}$, etc. Each of the 7 possible rotations generates a different guess for that part of Δ_1. The same issue happens with the cycles of size 5. With the 1-cycles there is thankfully no ambiguity.

Example 2.5. *A further complication comes up when $\Delta_4\Delta_1$ has more than two cycles of a given length. For instance, if*

$$\Delta_4\Delta_1 = (\texttt{AILNMC})(\texttt{RGYBOF})(\texttt{WZESQU})(\texttt{DHVXJP})(\texttt{K})(\texttt{T}),$$

then we have to decide how to pair up the cycles of size 6, and there are 3 different ways doing so; After pairing them up, we again of 6 ways of rotating each pair independently.

Exercise 2.7. *Suppose $\Delta_4\Delta_1$ has type $[1^{z_1}2^{z_2}\cdots n^{z_n}]$. Give a formula (in terms of the z_i) for the number of factorizations that the process above will give. Can you find the type that maximizes that number?*

2.4.3 Step Three: Identifying Δ_1, Δ_2, and Δ_3

The previous step generates several guesses for the Δ_i. To determine which is most likely the correct one, the team would exploit the fact that senders sometimes chose non-random message keys. Message keys like `AAA`, `BBB`, `ABC`, etc were chosen by operators, and these characteristics enabled a limited form of frequency analysis. An easy property is that the first character of the message key was most frequently `A`; This yields a guess for $\Delta_1(\texttt{A})$, and then the above technique will extend it to guess the rest of cycle containing `A`.

Exercise 2.8. *Let $\Delta_4\Delta_1$ be as in Example 2.5, and suppose you observe that the most frequent first letter of message keys is `Y`. What values of Δ_1 does this lead you to guess?*

This process takes some guesswork, but is reported to have worked well. It helped that a guess could be checked, because even less frequent message keys would often have some structure (like `HIJ`), and these would appear once the correct permutations had been recovered.

2.4.4 Proof of Rejewski's Theorem (Optional)

Let ρ_1 and ρ_2 be reflectors, and suppose $(a_1\ a_2\ \cdots\ a_t)$ appears in the cycle decomposition of their product $\rho_2\rho_1$. We need to show that $(\rho_1(a_t)\ \rho_1(a_{t-1})\ \cdots\ \rho_1(a_1))$ also appears and is distinct.

Start by writing $\rho_1(a_i) = b_i$ for $i = 1, \ldots, t$. Since ρ_1 is a reflector we also have that $\rho_1(b_i) = a_i$. We claim that none of the b_i will equal any a_j (more formally, that $\{a_1, \ldots, a_t\}$ and $\{b_1, \ldots, b_t\}$ are disjoint sets). We'll assume that's true for now and finish proving the theorem, and then come back to verify that fact at the end.

Since $\rho_2\rho_1(a_i) = a_{i+1 \bmod t}$, we must have $\rho_2(b_i) = a_{i+1 \bmod t}$; Again since ρ_2 is a reflector, we have $\rho_2(a_i) = b_{i-1 \bmod t}$ for all $i = 1, \ldots, t$ (note that we swapped up the subscript to subtract on one side instead of add to the other side, but this still true for general i). Then

$$\rho_2\rho_1(b_i) = \rho_2(a_i) = b_{i-1 \bmod t}.$$

In other words, we have shown that $\rho_2\rho_1$ maps b_i to $b_{i-1 \bmod t}$, i.e. $\rho_1(a_i)$ to $\rho_1(a_{i-1 \bmod t})$. This is exactly what we wanted to show. □

We now return to the claim that $\{a_1, \ldots, a_t\}$ and $\{\rho_1(a_1), \ldots, \rho_1(a_t)\}$ are disjoint sets. This will follow from the following lemma.

Lemma 4. *Suppose ρ_1 and ρ_2 are reflectors and that $\rho_1(x) = y$. Then x and y are in different cycles in the decomposition of $\rho_2\rho_1$.*

Proof. To show that x and y are in different cycles of $\rho_2\rho_1$, it suffices to show that $(\rho_2\rho_1)^k(x) \neq y$ for every $k \geq 1$. For any $k \geq 1$ we have

$$(\rho_2\rho_1)^k(x) = (\rho_2\rho_1)^{k-1}\rho_2\rho_1(x) = (\rho_2\rho_1)^{k-1}\rho_2(y).$$

We need to show that $(\rho_2\rho_1)^{k-1}\rho_2(y) \neq y$. We'll do this by showing that $(\rho_2\rho_1)^{k-1}\rho_2$ is a reflector. This follows by checking that $(\rho_2\rho_1)^{k-1}\rho_2$ is actually a conjugate of either ρ_1 or ρ_2 for every $k \geq 1$. For example, if $k = 1$ this is just ρ_2. If $k = 2$ then

$$(\rho_2\rho_1)^{2-1}\rho_2 = \rho_2\rho_1\rho_2 = \rho_2\rho_1\rho_2^{-1},$$

which is a conjugate of ρ_1 and hence a reflector. Working out the exact formula is left as an exercise. □