La bomba de Rejewski y su uso para descifrar mensajes de Enigma

Diego Viver Rodrigo

Trabajo de fin de grado de Matemáticas
Universidad de Zaragoza

Directores del trabajo: Paz Jiménez Seral y
Manuel Vázquez Lapuente
27 de junio de 2022

Abstract

During World War II and some time before, the German military started using complex cypher machines called the Enigmas. It was used to communicate through radio without the enemy being able to eavesdrop. Of course, there was a race to break the code, and the first people able to do so were Polish mathematicians at the Polish Cipher Bureau (*Biuro Szyfrów*).

In this article we discuss the methods devised by Jerzy Różycki and Marian Rejewski, that were used before the start of the War. First, we need to understand, in very broad strokes how the machine worked:

The machine consists of: a keyboard, 26 lightbulbs that each light up a letter, three rotating wheels set up one beside the others, a fixed wheel on the far left end of the rotating wheels called the reflector and a plug-board:

- The keyboard is used to type in the text we want to encipher, every time we push down a key, the leftmost wheel rotates one position.
- Every one of the rotating wheels has a ring with letters written on it. This ring can be detached and turned independently. We will call the letter that goes in the place of `A` the *ring-setting* of a specific wheel. Each wheel also has a notch affixed to the ring that determines when the next rotor will turn.

  The rotors can be inserted into the machine in any order and, in the period of time that interests us, there were three possible rotors. So there are 6 possible rotor orderings.

  The rotors can also be manually. This is often done before enciphering a message and the position, designated using the letter pointing up, is called the *ground-setting*.

  Every one of these wheels, even the fixed one, have a wiring from one side to the other, that sends a letter coming in to a different letter. This, coupled with the fact that the machine "re-shuffles" the letters on every key press, is the main mechanism of encryption of the Enigma.
- The plug-board allows the operator to manually plug a letter to another, both in the beginning an the end of the encryption process.
- The lightbulbs will, for every letter inputted, give us the corresponding enciphered letter for the message.

When we press down a key, we close an electric circuit that makes current go, in this order, through the plug-board, then the rightmost rotating wheel, then the middle rotor, then the rightmost rotor, then the reflector and then the reverse path to the lightbulbs. This can be shown, using a theorem of permutation theory, to give us a permutation that is its own inverse. This means that the key used to encrypt is the same as the one used to decrypt.

The key of a message is formed by four things:

- The order of the rotors,
- The *ring-setting* for each rotor,
- The *ground-setting* for each rotor,

- And the plug-board connections.

At different points of the war, certain steps of the enciphering method were changed to make the process more secure, at the time of the application of our methods we had that the rotor order, *ring-setting* and plug-board connections were changed unfrequently (every three months at the beginning, and right before the War, daily). And then, all messages began with a ground setting in the clear, followed by the operator typing a random key selected by them at that moment, twice. Then they would set the machine to that random setting, known as the *message key*, to type the message.

The first method discussed in this document, called *the Clock method* concerns finding the rightmost rotor, which might seem insignificant, but it cuts down the work of finding the rest of the information by a lot. This method is the only method used by the Poles that required a statistical analysis of the German language, as it is based on the fact that, given two phrases in German, if we were to write them one on top of the other, we would find that two times out of 26, we have the same letter in both messages in the same position. With this, and the fact that if we encipher these two messages with the same key, the coincidences will remain, they could find points where the middle rotor was turning over and, as each rotor has a different point of turnover, this allows for the deduction of the rotor in the rightmost position.

The second method is by far the most interesting, as it is, in a way, the predecessor to Alan Turing's *Bombe*. This method was called *Rejewski's Bomba*. It was an electro-mechanical contraption that allowed the poles to simultaneously find the rotor order and the *ring-setting*, which left them only with the plug-board to deduce. This machine is based on certain properties of the permutation performed by the Enigma machine on the letters of the message. This early attempt at a mechanized solution could be considered somewhat crude, as it essentially tries all 26^3 possible settings until it finds the ones that are likely, but it is surprisingly effective and fast. Even back in the day, the six *Bombi* that were used by the Cipher Bureau (one for each rotor ordering), where done in under two hours. Programmed on a modern computer, the whole process lasts under five seconds.

Índice general

Capítulo 1

Introducción

1.1 La máquina Enigma

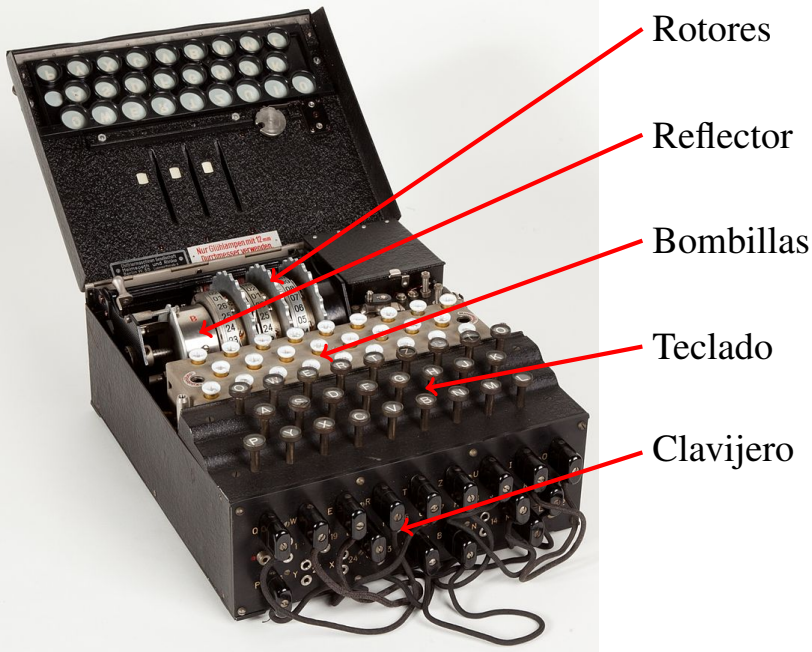

Figura 1.1: Máquina Enigma (Fuente de la imagen: [7])

La máquina Enigma es un dispositivo electromecánico que fue usado en la Segunda Guerra Mundial por el ejército alemán para cifrar mensajes enviados en morse por radio.

El mensaje se introducía sin cifrar mediante un teclado, similar al de una máquina de escribir. Al pulsar una tecla se cierra un circuito eléctrico y se enciende una bombilla. La bombilla que se enciende corresponde a la letra cifrada. Si tecleamos un texto cifrado, las bombillas iluminadas se corresponderán al texto sin cifrar si la clave (ver seccion 1.1.1) es la correcta.

Justo encima del teclado había una tapa con ventanas para las bombillas y tres huecos más, esta tapa en la figura 1.1 viene abierta. Debajo de la tapa se ven tres rotores móviles que se pueden extraer de la máquina, reordenándolos de cualquier manera. Los rotores tienen una serie de contactos a ambos lados. Por dentro, los contactos de un lado están conectados a los del otro mediante cables como se ve en el diagrama de la figura 1.2.

Además, cada uno de los rotores tienen una muesca que indica el punto de *turnover*, es decir, el punto en que un rotor hará girar el inmediatamente adyacente. Los rotores se pueden rotar manualmente y, además, el anillo alfabético que indica la posición como se aprecia en la figura 1.2 también se puede

girar independiente al propio rotor.

Figura 1.2: Rotor junto a diagrama simplificado del cableado similar al de [5] (Fuente de la imagen de la izquierda [6])

A la izquierda de los rotores se encuentra el reflector (*Umkehrwalze* en alemán). Marian Rejewski en [5] describe que en la máquina hay tres rotores móviles y uno inmóvil, que es el reflector. Esta descripción se debe principalmente a que funcionan igual, solo que el reflector solo tiene contactos en un lado, y éstos están conectados los unos a los otros por parejas. Esto se puede ver como una limitación técnica, porque la única manera de cablearlo es así. Esto tiene la consecuencia de que la clave para cifrar y para descifrar sea la misma.

Debajo del teclado de la máquina se encuentra el clavijero, que se trata de una serie de contactos que se pueden cablear manualmente mediante conectores. Estos contactos van en parejas, y cada pareja representa una letra.

Al pulsar una tecla, el rotor de la izquierda gira una posición. Esta acción también cierra el circuito eléctrico, haciendo que la corriente pase por el clavijero, el rotor derecho, central e izquierdo, después por el reflector, y hace el camino inverso por los rotores izquierdo, central y derecho y por último vuelve a pasar por el clavijero a una bombilla. Esta bombilla se corresponde a la letra que se enviará. Por tanto, tecleamos el mensaje a cifrar (el *plaintext*) y vemos letra a letra en las bombillas el mensaje que enviamos (el *ciphertext*).

Si al girar el rotor derecho su muesca de turnover pasa por un punto determinado, hará girar el rotor central y lo mismo pasa con el central.

1.1.1 Configuración de la máquina

La gran complejidad de descifrar mensajes de Enigma viene de la gran variedad de configuraciones posibles:

- El **orden de rotores** es simplemente en qué orden se colocan los rotores dentro de la máquina. En la época que nos interesa había solo tres rotores, por tanto el número de ordenes posibles es de $3! = 6$. Esto se indicaba en los libros de claves como listas de tres números romanos separados por guiones, por ejemplo `I-II-III`.

- Cada rotor tiene dos configuraciones, el *Ringstellung* y el *Grundstellung*, también llamados comúnmente **ajuste interno** y **ajuste externo**.

  El ajuste interno se refiere a la posición del anillo indicador del rotor con respecto a su posición neutra y se indica con tres letras, una para cada rotor, donde cada una representa la letra que se pone en la posición de la `A`, en el orden que se encuentran los rotores.

  El ajuste externo simplemente indica la letra que se ve desde la ventana de la máquina, de nuevo para cada uno de los rotores en el orden que están.

- La configuración del clavijero nos indica que parejas se conectan en éste escribiendo las propias parejas separadas por guiones, por ejemplo `AL-DE-FG`.

En algún momento más adelante se hablará de **ajuste conjunto**. Esto no hace más que resumir los dos ajustes relativos a los rotores en un solo ajuste, lo cual a veces es útil, sobre todo al hacer simulaciones computacionales. Si consideramos que cada letra es un elemento de $\mathbb{Z}_{26}$, donde la $\mathtt{A} \equiv 0$ y la $\mathtt{Z} \equiv 25$, si tomamos por ejemplo un ajuste interno de $\mathtt{J} \equiv 9$ y uno externo de $\mathtt{G} \equiv 6$, el ajuste conjunto sería:

$$\mathtt{G} - \mathtt{J} \equiv 6 - 9 = -3 \equiv 23 \equiv \mathtt{X}.$$

Esto lo que expresa es que, es lo mismo poner un rotor con el ajuste interno y externo del ejemplo, que ponerlo con el ajuste interno `A`, y el externo a `D`.

Esta equivalencia entre letras y números en $\mathbb{Z}_{26}$ se usará mucho.

1.2 Herramientas matemáticas

1.2.1 Permutaciones

Vamos a considerar como actúa la máquina sobre las distintas letras que conforman un mensaje usando permutaciones. La máquina, como se ha descrito antes, tiene varias partes que intercambian una letra por otra. Consideramos $\mathscr{A}$ el alfabeto de 26 letras, ya que no estamos considerando la "ñ". Llamaremos **permutación** a una aplicación biyectiva $\gamma : \mathscr{A} \to \mathscr{A}$.

Sabemos que el conjunto de todas las biyecciones γ forman un grupo no abeliano, tomando como operación la composición de las aplicaciones, la cual denotaremos como si fuese un producto.

Denotamos la acción de una permutación sobre una letra como un exponente, es decir, si p es una letra en A, la acción de una permutación γ sobre esta letra se denotará p^γ o $p\gamma$. Cuando aplicamos un producto de permutaciones a una letra, se aplican de izquierda a derecha, es decir en $p^{\gamma\lambda\rho}$, se aplica primero γ, luego λ, y por último ρ.

En concreto, la estructura que tienen las permutaciones como las hemos definido es la siguiente:

$$\{\gamma : \mathscr{A} \to \mathscr{A} \mid \gamma \text{ es biyectiva}\} \simeq \Sigma_{26}$$

El grupo que obtenemos es el **simétrico** Σ_{26}. El hecho de que tenga estructura de grupo se ve en que:

1. Dadas tres biyecciones $\gamma, \lambda, \mu : \mathscr{A} \to \mathscr{A}$, tenemos que $(\gamma\lambda)\mu = \gamma(\lambda\mu)$ al ser asociativa la composición de funciones.

2. Existe una identidad $\iota : \mathscr{A} \to \mathscr{A}$ que será la aplicación identidad que nos lleva cualquier $p \in A$ a la propia p. Esta aplicación cumple que, para cualquier $\pi : \mathscr{A} \to \mathscr{A}$ biyectiva, $\iota\gamma = \gamma\iota = \gamma$.

3. Para cualquier $\gamma : \mathscr{A} \to \mathscr{A}$ biyectiva, tiene una inversa γ^{-1} que cumple que $\gamma\gamma^{-1} = \gamma^{-1}\gamma = \iota$.

También es importante notar que, si dos permutaciones "no mueven" las mismas letras, conmutan. Es decir, sean $\gamma, \lambda : \mathscr{A} \to \mathscr{A}$ tales que, existen $\mathscr{B}, \mathscr{C} \subseteq \mathscr{A}$ disjuntos y tales que $\mathscr{B} \cup \mathscr{C} = \mathscr{A}$ y $\forall b \in \mathscr{B}$, $b^\gamma = b$ y $\forall c \in \mathscr{C}$, $c^\lambda = c$, entonces $\gamma\lambda = \lambda\gamma$. En este caso las llamaremos **permutaciones disjuntas**.

Tenemos dos maneras de denotar una permutación. La primera, es indicar la imagen de cada elemento de A en una matriz de la siguiente manera (tomamos como ejemplo el rotor `I` de la máquina):

$$\gamma = \begin{pmatrix} a & b & c & d & e & f & g & h & i & j & k & l & m & n & o & p & q & r & s & t & u & v & w & x & y & z \\ e & k & m & f & l & g & d & q & v & z & n & t & o & w & y & h & x & u & s & p & a & i & b & r & c & j \end{pmatrix},$$

lo que quiere decir, por ejemplo, que $a^\gamma = e$, $k^\gamma = n$, etc.

La otra manera es la que se llama **notación de ciclos disjuntos** y es la más sencilla para operar con permutaciones. Un **ciclo** es una permutación γ tal que existe una cadena de letras distintas (a_i) con $a_i \in A$, $i = 1, \ldots, n$, $n \leq 26$, que cumple que, con $j < n$, $a_{j+1} = a_j^\gamma$ y $a_1 = a_n^\gamma$, y el resto de letras de A que no están en esta cadena se quedan fijas. Esto lo denotamos:

$$\gamma = \left(a_1\ a_2 \cdots a_n\right).$$

Y, si las letras fijas son a_j con $j = n+1, \ldots, 26$, a veces las indicaremos como:

$$\gamma = \left(a_1\, a_2 \cdots a_n\right)(a_{n+1}) \cdots (a_{26}).$$

Cualquier permutación se puede denotar como un producto de ciclos disjuntos de una manera única salvo el orden de los ciclos ya que, al ser estos ciclos permutaciones disjuntas, conmutan.

Para pasar de una notación a la otra, comenzamos con una letra y vamos escribiendo las imágenes sucesivas hasta que volvemos a la primera, luego tomamos otra que no salga en el primer ciclo y así, sucesivamente hasta tener todas las letras. Tomando como ejemplo la permutación del primer rotor de nuevo:

- comenzando en la *a*:
$$\left(a\, e\, l\, t\, p\, h\, q\, x\, r\, u\right).$$
- Tomamos la *b* ahora, ya que no está en el ciclo anterior:
$$\left(b\, k\, n\, w\right).$$
- Seguimos hasta obtener las que faltan:
$$\left(c\, m\, o\, y\right),\ \left(d\, g\, f\right),\ \left(i\, v\right),\ \left(j\, z\right),$$
quedándose la *s* suelta.

Y con esto nos queda la expresión usada en la ecuación 1.2 para el rotor `I`, es decir:

$$(aeltphqxru)(bknw)(cmoy)(dgf)(iv)(jz)(s).$$

Lo último que debemos notar es que, como hemos comentado antes, estos ciclos, al ser disjuntos, conmutan. Esto quiere decir que los podemos escribir en cualquier orden, preferimos escribirlos de mayor a menor longitud. La lista de longitudes de los ciclos disjuntos que conforman una permutación se llama su **estructura de ciclos** y tiene una propiedad interesante en lo que a la máquina Enigma respecta.

Por ejemplo, la permutación del rotor `I` tiene estructura $(10,4,4,3,2,2,1)$.

Dadas dos permutaciones γ y σ, la permutación γ **conjugada con** σ es:

$$\sigma^{-1}\gamma\sigma = \gamma^{\sigma}.$$

Esta transformación conserva la estructura de ciclos de la permutación γ. Tenemos además un resultado aún más fuerte, si tenemos dos permutaciones γ y λ con la misma estructura de ciclos, entonces son conjugadas, es decir: existe una permutación σ tal que

$$\gamma = \lambda^{\sigma}.$$

Este teorema es de especial interés en lo que respecta a la máquina y permitió a Rejewski averiguar los cableados de los rotores, así como construir las bombas.

1.2.2 Enigma con permutaciones

Podemos ver la máquina Enigma como una serie de permutaciones del grupo Σ_{26} actuando sobre las letras del alfabeto. En la figura 1.3 podemos ver la serie de permutaciones que consideramos.

La letra *i*-ésima del *ciphertext* es la imágen de la letra *i*-ésima del *plaintext* por la permutación:

$$\alpha_i = \sigma\pi^{i+c_3}\nu\pi^{-i-c_3}\pi^{j(i)+c_2}\mu\pi^{-j(i)-c_2}\pi^{k(i)+c_1}\lambda\pi^{-k(i)-c_1}\rho\pi^{k(i)+c_1}\lambda^{-1}\pi^{-k(i)-c_1}\pi^{j(i)+c_2}\mu^{-1}\pi^{-j(i)-c_2}\pi^{i+c_3}\nu\pi^{-i-c_3}\sigma, \tag{1.1}$$

donde:

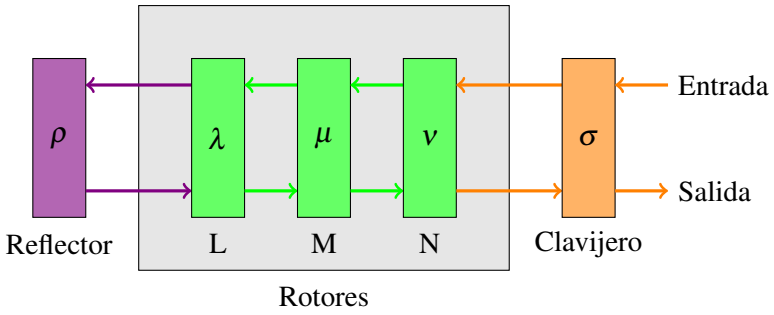

Figura 1.3: Diagrama simplificado de la máquina Enigma

- La permutación π es la permutación que manda una letra del alfabeto a la siguiente, es decir:

$$\pi = (abcdefghijklmnopqrstuvwxyz),$$

y al elevarla a la potencia i-ésima, simulamos el hecho de que el rotor en la posición N (derecha) haya girado i lugares.

- Las permutaciones ν, μ y λ simulan las permutaciones que hacen los rotores en posiciones N, M y L respectivamente, y pueden ser una de las siguientes dependiendo del orden de rotores. Estas pueden ser una de las siguientes (como se puede ver en [1]):

$$\begin{aligned} \texttt{Rotor I} &= (aeltphqxru)(bknw)(cmoy)(dgf)(iv)(jz)(s) \\ \texttt{Rotor II} &= (fixvyomw)(cdklhup)(esz)(bj)(gr)(nt)(a)(q) \\ \texttt{Rotor III} &= (cflvmzoyqirwukxsg)(abdhpejt)(n). \end{aligned} \tag{1.2}$$

- La permutación ρ es la permutación correspondiente al reflector, es una permutación de clase $2,\cdots,2$ (o sea, es un producto de trece trasposiciones). En esta época (como se puede ver en [2]), era el llamado reflector `B`, que se corresponde a la permutación (ver [1]):

$$\texttt{UKW-B} = (ay)(br)(cu)(dh)(eq)(fs)(gl)(ip)(jx)(kn)(mo)(tz)(vw) \tag{1.3}$$

- La permutación σ corresponde al clavijero . Esta siempre es un producto de trasposiciones (ver sección 1.1). Por ejemplo, el clavijero `AL XV CB` se correspondería a la permutación:

$$\sigma = (al)(xv)(cb)$$

Al ser un producto de trasposiciones, su inversa es ella misma, por eso no tenemos su inversa en la ecuación 1.1.

- La terna (c_1, c_2, c_3) se corresponde a la configuración conjunta de la máquina como se describe en la sección 1.1.1.

- $j(i)$ y $k(i)$ son números enteros que se corresponden a las rotaciones que se dan en los rotores L y M debido al turnover del rotor anterior :

  - Si el rotor anterior es el rotor `I`, el turnover sucede cuando la letra de la ventanilla cambia entre la `Q` y la `R`
  - Si el rotor anterior es el `II`, sucede cuando la letra cambia entre la `E` y la `F`
  - Si el rotor anterior es el `III`, sucede cuando la letra cambia entre la `V` y la `W`

Durante una parte de los desarrollos posteriores, se asumirá que no se da turnover en los rotores M y L, entonces, podremos escribir la permutación de una manera resumida:

$$\alpha_i = \sigma\pi^{i+c_3}\nu\pi^{-i-c_3}\chi_{c_2,c_3}\pi^{i+c_3}\nu\pi^{-i-c_3}\sigma \tag{1.4}$$

Donde χ_{c_2,c_3} es la parte central de la permutación de la ecuación 1.1 poniendo $j(i) = k(i) = 0$.
En algún momento omitiremos el clavijero, esto lo denotaremos con:

$$\overline{\alpha}_i = \pi^{i+c_3}\nu\pi^{-i-c_3}\chi_{c_2,c_3}\pi^{i+c_3}\nu\pi^{-i-c_3} \tag{1.5}$$

Es evidente que $\alpha_i = \sigma\overline{\alpha}_i\sigma = (\overline{\alpha_i})^{\sigma}$.

Antes se ha mencionado que la conjugación de permutaciones conserva la estructura de ciclos de la permutación conjugada. Podemos observar que la permutación α es el resultado de conjugar ρ por otra permutación. Como ρ tiene estructura de ciclo $(2,{}^{13}\!.\!.,2)$, entonces todas las α_i tendrán esa misma estructura. Esto es lo que hace que la configuración utilizada para cifrar y descifrar sea la misma.

1.2.3 Simulador

El simulador incluido en el apéndice A está programado usando *Python 3*. Modelamos la máquina de una manera mecánica. La simulación de la máquina consta de dos clases:

- La clase `Rotor` representa cada uno de los rotores de la máquina. Se inicializa indicando mediante una `string` de longitud uno la letra que se muestra en la ventana de la máquina cuando se ha efectuado el *turnover* (parámetro `to`), así como la permutación base que realiza el rotor, indicada como una `string` de longitud 26 que nos da la imagen del alfabeto bajo la permutación.

  Esta clase tiene varios métodos:

  - Los métodos `set_rs` y `set_gs` toman ambos un parámetro entero, y establecen respectivamente el ajuste interno y externo para el rotor dado.
  - Los métodos `permuta_ida` y `permuta_vuelta` hacen actuar la permutación correspondiente al estado actual del rotor, o su inversa, respectivamente, sobre la letra que toman como parámetro.
  - `turn` es el método que simula el hecho de que el rotor gire una posición, también comprueba si ha habido turnover en esa pulsación.
  - `turnover` se encarga de decirle a la clase `Maschine` si tiene que girar el siguiente rotor y de resetear la bandera `isTurnover` en caso afirmativo.
  - `ver_gss` sirve para ver lo que se vería por la ventanilla de la máquina correspondiente a ese rotor.

- La clase `Maschine` representa la máquina completa. Se inicializa mediante tres objetos de la clase `Rotor`, `R1, R2` y `R3` (van de derecha a izquierda, por tanto, `R1` es el rotor derecho), el reflector, indicado como una `string` de longitud 26 que representa la permutación realizada como imagen del alfabeto y el clavijero, en formato diccionario de *Python* (`None` en caso de que no haya conexiones).

  Esta clase tiene los siguientes métodos:

  - `settings` realiza la configuración inicial de la máquina, acepta dos parámetros que serán dos listas de enteros de longitud 3. La primera, `gss` representará los ajustes externos, mientras que la segunda, `rss`, representará los ajustes internos.
  - `cifrar_letra` acepta una `string` de longitud uno y devuelve el resultado de cifrar esa letra. También muta el estado de la máquina del mismo modo que si hubiésemos pulsado una tecla en la versión mecánica.

- `cifrar_mensaje` ejecuta el método anterior a lo largo de una cadena entera.
- El método mágico `__repr__`, que se ejecuta automáticamente al hacer una conversión implícita o explícita del tipo `Maschine` al tipo `string` nos muestra lo que se mostraría en las ventanas de la máquina.

Ejemplo 1.1. Ciframos el mensaje *Máquina Enigma en Python* con orden de rotores `I-II-III`, ajuste externo `UNI`, ajuste interno `ZAR` y clavijero `ME PC FR`.

Para empezar, el mensaje se quedaría de la forma:

`MAQUINAXENIGMAXENXPYTHON`

Sustituyendo los espacios por la letra `X`, como se hacía con la máquina original.
Tendríamos que expresar las claves con números:

- `UNI` será $(20, 13, 8)$
- `ZAR` será $(25, 0, 17)$

Creamos el objeto que representa la máquina, apoyándonos de los objetos de tipo `Rotor` que ya están creados en el apéndice A.4:

```
M = Maschine(R3, R2, R1, UKW_B, {"M":"E", "P":"C", "F":"R"})
M.settings([20, 13, 8], [25, 0, 17])
```

Y ciframos nuestro ejemplo:

```
M.cifrar_mensaje("MAQUINAXENIGMAXENXPYTHON")
```

Obteniendo como resultado:

`FVUEJRNVFILTRQNYSMNIMCBD`

Y para descifrarlo:

```
M.settings([20, 13, 8], [25, 0, 17])
M.cifrar_mensaje("FVUEJRNVFILTRQNYSMNIMCBD")
```

Nos dará como resultado:

`MAQUINAXENIGMAXENXPYTHON`

Capítulo 2

Nuestro problema y sus soluciones

2.1 Enmarcando el problema en el contexto historico en el que se daba

En [2, Apéndice C], Marian Rejewski da una cronología de los distintos métodos usados para poder romper los cifrados en los que los alemanes usaron la máquina Enigma y discute brevemente cómo se adaptaron desde los servicios de inteligencia polacos a los distintos cambios.

El primer método que se discutirá aquí es el método del reloj, que es un método muy temprano, desarrollado por Różycki antes del año 35, de acuerdo con Rejewski en [4]. En esta época ya se conocía el cableado de los rotores y ya se disponía de algún método para obtener información sobre los mensajes recibidos. Este método nos permite obtener el rotor que está en la posición derecha, pero en este periodo, el orden de rotores cambiaba trimestralmente, por tanto esto servía para varios días.

Las *bombas* aparecen en noviembre del año 1938, para mitigar el hecho de que el método anterior que tenían, el catálogo generado usando el *Ciclómetro*, se había vuelto inútil al cambiar los alemanes el método de transmitir las claves de mensaje. Las bombas se usaron hasta el principio de la guerra puesto que Rejewski y su equipo tuvieron que abandonar estas máquinas, las bombas, en las oficinas de Varsovia. Ya antes, perdieron efectividad al aumentar el número de rotores de tres a cinco. Esto quiere decir que las bombas tuvieron efectividad plena durante unos dos meses.

Es importante saber que, en la época cercana al año 38:

- El orden de rotores y el ajuste interno (*Ringstellung*) cambia a diario
- Se usa el reflector `B`
- El número de conexiones del clavijero varía entre 5 y 8, aunque a partir de enero de 1939, aumentaron el numero de conexiones a entre 7 y 10. Esto también hizo más complejo el proceso de descifrado.

Es importante saber cómo se transmiten los mensajes en esta etapa de la historia, para ello vemos un ejemplo:

```
1 Ferschreiben H.F.M.No. 563
2 +HRKM 13617  1807 =
3 AN HEERESGRUPPENKOMMANDO 2=
4 2109 - 1750 - 3 TLE - FRX FRX - 1TL - 172=
5
6 HCALN UQKRQ AXPWT WUQTZ KFXZO .....
```

Listing 2.1: Mensaje ejemplo de [3] enviado desde Frankfurt el día 21 de septiembre de 1938

Lo primero que se encuentra resaltado es la clave externa (Grundstellung) para la clave de mensaje y se transmite sin cifrar. Lo segundo resaltado es la clave de mensaje cifrada dos veces. Es decir, si la clave de mensaje fuese `XYZ`, esos seis caracteres serán el resultado de teclear `XYZXYZ` con la configuración indicada para ese día.

Esto se hacía para detectar posibles errores en la recepción de un mensaje. Sabiendo que el resultado de descifrar esas seis primeras letras tiene que ser una pareja de claves de la manera indicada, si al descifrar con la clave correcta no se obtenía eso, se transmitía un mensaje para pedir que se volviese a mandar la comunicación inicial.

Los autores, tanto de la época como posteriores, que han escrito sobre la máquina Enigma, para facilitar la comprensión, escribirían esto de la siguiente manera:

FRX, HCA LNU.

2.2 El método del reloj

Es el único método para descifrar la máquina Enigma que se basa en un ataque estadístico del mensaje, además de en características propias del idioma (el alemán en este caso) y permitía, en muchas situaciones, averiguar cuál era el rotor en la posición N (la de más a la derecha como se ve en 1.3).

El principio en el que se basa es que, dados dos textos en alemán, $V = (v_1, v_2, \ldots, v_n)$ y $W = (w_1, w_2, \ldots, w_m)$, en un rango de 26 letras se puede esperar que dos coincidirán en la misma posición, es decir, si tenemos un tramo de 26 letras seguidas de cada mensaje:

$$(v_k, v_{k+1}, \ldots, v_{k+26}), \quad (w_k, w_{k+1}, \ldots, w_{k+26});$$

con $1 \leq k \leq n-26$, cabe esperar encontrar dos índices p, q en ese rango tales que las letras correspondientes a esas posiciones coinciden. Es decir: $v_p = w_p$ y $v_q = w_q$.

Por ejemplo, podemos considerar las dos frases siguientes en alemán:

```
Niemand kann sich mehr eine Welt ohne Internet vorstellen
          =        =   =            =               =
Das Kind unseres Freundes ist seit dem Unfall behindert
```

Listing 2.2: Ejemplo del índice de coincidencia

Donde tenemos 5 coincidencias en un texto de 58 caracteres, lo que es coherente con lo dicho arriba. En castellano, este índice también es parecido. Hemos comparado el primer capítulo de *El Quijote* y el cuento *La Biblioteca de Babel* de Jorge Luis Borges, tenemos 827 coincidencias lo cual, siendo el texto más corto el extraido de *El Quijote* con 10600 caracteres, nos da un índice de coincidencia de 2.03 por cada 26 letras, similar a lo que pasaba con los textos en alemán.

Definimos ahora los textos cifrados como:

$$\begin{aligned} V_\alpha &= (v_1\alpha_1, v_2\alpha_2, \ldots, v_i\alpha_i, \ldots, v_n\alpha_n) = (v_i\alpha_i) \\ W_\alpha &= (w_1\alpha_1, w_2\alpha_2, \ldots, w_i\alpha_i, \ldots, w_n\alpha_n) = (w_i\alpha_i). \end{aligned}$$

Si asumimos que ciframos los dos textos V y W con la máquina en la misma configuración, como sabemos que la permutación que actúa en cada letra será la misma, de la fórmula de arriba se deduce que:

$$v_p\alpha_p = w_p\alpha_p \quad \text{y} \quad w_q\alpha_q = w_q\alpha_q,$$

es decir, vemos que las coincidencias están en las mismas posiciones, o, que si dos letras coinciden en los textos sin cifrar, coincidirán en los textos cifrados.

Volviendo al ejemplo anterior, si cifrásemos ambos mensajes con la misma configuración (orden de rotores `I-II-III`, *Grundstellung* `ABC`, *Ringstellung* `DEF` y sin clavijero), vemos que aparecen letras iguales en exactamente la misma posición que antes:

```
wlmntimhumbewbumbibwvddfysqmemyidvmalysfyqvawcarcymbcugii
          =         =   =             =                 =
adhcxnabeebviuhomtwwfuxfaioqdhakpkoklztmiwfxzfdbzjvjcdk
```

Listing 2.3: Ejemplo anterior cifrado

Arriba se han cifrado los espacios en blanco usando la letra `X`, que era lo usual.

Esto confirma que para cada letra, la permutación que la máquina hace en ese punto, es la misma y nos permitiría deducir, a falta de más información, que los dos mensajes se han cifrado con la misma clave. Sin embargo, si las claves fuesen distintas, como la máquina tiene la misma probabilidad de mostrar cada letra en el texto cifrado, veríamos una coincidencia solamente cada 26 caracteres, ya que estaríamos esencialmente comparando dos cadenas de caracteres aleatorias, donde cada letra tiene una posibilidad de aparecer de $1/26$.

Para aplicar este método, necesitamos tener dos mensajes cifrados con los mismos ajustes excepto el ajuste externo correspondiente al rotor de la derecha.

Dados suficientes mensajes interceptados el mismo día y, por tanto, cifrados con el mismo ajuste interno, orden de rotores y clavijero; se asume que se pueden encontrar al menos dos en los que las dos primeras letras de la clave de mensaje coincidan y solo difieran en la tercera, gracias a esto, conocemos la posición relativa de los rotores derechos de un mensaje respecto del otro. Por ejemplo, dadas las claves de mensaje `FNF` y `FNI`, sabemos que en el segundo mensaje, el rotor derecho está tres posiciones por delante de la posición en la que estaría en el primer mensaje.

Para comenzar el procedimiento del reloj, colocamos los mensajes, uno encima del otro, de la siguiente manera: consideramos las claves de mensaje (*Grundstellung*) (g_1, g_2, g_3) y (g_1, g_2, g_3'). Esto quiere decir que las permutaciones que actúan sobre cada una de las letras de los mensajes no es la misma. Denotamos por α_i a la permutación que actúa sobre V y β_i a la que actúa sobre W, por tanto, los mensajes cifrados serán de la siguiente manera:

$$\begin{aligned} V_\alpha &= (v_1\alpha_1, v_2\alpha_2, \ldots, v_i\alpha_i, \ldots, v_n\alpha_n) = (v_i\alpha_i) \\ W_\beta &= (w_1\beta_1, w_2\beta_2, \ldots, w_i\beta_i, \ldots, w_n\beta_n) = (w_i\beta_i). \end{aligned}$$

Ahora, examinamos los dos mensajes escribiéndolos de dos modos:

1. Escribimos el alfabeto empezando en g_3 (y ponemos después de la `Z` la `A` si fuese necesario) y escribimos el primer mensaje cifrado V_α, empezando justo después de g_3. Después, donde aparezca g_3', escribimos W_β. En la tabla de abajo se entiende que $g_3 + 1$ es la letra siguiente a g_3, considerando la siguiente a la `Z` la `A`:

g_3	g_3+1	$\cdots$	$\cdots$	g_3'	$g_3'+1$	$\cdots$	$\cdots$
	$v_1\alpha_1$	$v_2\alpha_2$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
					$w_1\beta_1$	$w_2\beta_2$	$\cdots$

   Ahora, si suponemos que la máquina está en la misma configuración a partir de g_3', es decir, que el ajuste externo sea (g_1, g_2, g_3') en ambos mensajes; deberíamos observar un índice de coincidencia coherente con lo dicho antes, es decir, de dos letras comunes por cada 26.

   Si no observamos esta coincidencia, esto querrá decir que se ha movido el rotor de el medio, es decir, como sabemos que, en ese punto, el ajuste correspondiente al rotor derecho es g_3' para ambos mensajes a partir de el punto que estamos observando; esto quiere decir que alguno de los otros dos ajustes han cambiado, es decir, que ha habido turnover en el rotor derecho entre la letra g_3 y la letra g_3'.

2. Repetimos esta forma de colocar los mensajes, pero ahora al revés, comenzando el alfabeto en g_3':

g_3'	$g_3'+1$	$\cdots$	$\cdots$	g_3	g_3+1	$\cdots$	$\cdots$
					$v_1\alpha_2$	$v_2\alpha_2$	$\cdots$
	$w_1\beta_1$	$w_2\beta_2$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

Miramos otra vez lo mismo, si los mensajes tienen el índice de coincidencia indicado anteriormente a partir de la g_3, esto querrá decir que la máquina está en la misma configuración ahí. Si no tenemos esa coincidencia, entonces habremos tenido *turnover* entre la letra g'_3 y g_3.

3. Mediante los pasos 1 y 2, hemos localizado un intervalo de letras entre las cuales está el *turnover* del rotor derecho. Como sabemos donde está el *turnover* para cada rotor, basta con comprobar cual de ellos está en este intervalo. Con suerte, será solo uno y entonces, habremos determinado con éxito cuál es el rotor que ocupa la posición derecha.

Ejemplo 2.1. Un ejemplo sencillo para ilustrar este método es tomando tres mensajes idénticos:

$$W_1, W_2, W_3 = \texttt{aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa}.$$

Al ser mensajes idénticos, el índice de coincidencia que buscamos es del 100 %. Ciframos estos mensajes con el orden de rotores `III-I-II`, Ringstellung configurado como `AAA` y el clavijero sin conexiones. Tomamos tres claves, en nuestro caso van a ser `AAA`, `AAL` y `AAT`, respectivamente. Esto queda así:

- Mensaje cifrado con la clave `AAA`: `cnbegnhejzleknscfkbknqhspbklvjrcfp`
- Mensaje cifrado con la clave `AAL`: `qolsqetfeydwopjcnbegnhejzleknscfkb`
- Mensaje cifrado con la clave `AAT`: `eydwopjcnbegnhejzleknscfkbknqhspbk`

Ahora, suponiendo que conocemos solo los mensajes cifrados y las claves de los mensajes, aplicamos el procedimiento descrito arriba:

```
Alfabeto  ->  abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz
Clave AAA ->   cnbegnhejzleknscfkbknqhspbklvjrcfp
Clave AAL ->              qolsqetfeydwopjcnbegnhejzleknscfkb
Iguales   ->              xxxxxxxxxxxxxxxxxxxxxx

```

Listing 2.4: Primera clave contra segunda

Esto indica que ha habido turnover entre la `A` y la `L`, seguimos:

```
Alfabeto  ->  lmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijk
Clave AAA ->                  cnbegnhejzleknscfkbknqhspbklvjrcfp
Clave AAL ->   qolsqetfeydwopjcnbegnhejzleknscfkb
Iguales   ->                  ====================

```

Listing 2.5: Primera clave contra segunda al revés

Esto nos indica que no tenemos turnover entre la `L` y la `A`. Comparamos ahora el mensaje cifrado con la primera clave y con la tercera:

```
Alfabeto  ->  abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz
Clave AAA ->   cnbegnhejzleknscfkbknqhspbklvjrcfp
Clave AAT ->                      eydwopjcnbegnhejzleknscfkbknqhspbk
Iguales   ->                      xxxxxxxxxxxxxxx

```

Listing 2.6: Primera clave contra tercera

```
Alfabeto  ->  tuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrs
Clave AAA ->          cnbegnhejzleknscfkbknqhspbklvjrcfp
Clave AAT ->   eydwopjcnbegnhejzleknscfkbknqhspbk
Iguales   ->          ===========================

```

Listing 2.7: Primera clave contra tercera al revés

Lo que nos indica que el turnover está entre la `A` y la `T`. Viendo cómo son los rotores (vemos cómo son los rotores en la sección 1.2.2) sabemos entonces que el turnover pasará en la `F` y por tanto, el rotor de la derecha es el `II`.

2.3 La bomba de Rejewski

2.3.1 Principio operativo

Las bombas fueron un método mecánico ideado por Rejewski utilizado para averiguar posibles ajustes internos de la máquina, así como el orden de rotores.

La herramienta básica para el uso de las bombas son los indicadores de los mensajes, tal y como se ven en el listing 2.1, es decir, de la forma:

$$g_1g_2g_3,\, m_1m_2m_3\ \overline{m}_1\overline{m}_2\overline{m}_3$$

Donde $g_1g_2g_3$ es el ajuste externo con el que se ha cifrado la clave de mensaje xyz, y $m_1m_2m_3\ \overline{m}_1\overline{m}_2\overline{m}_3$ es la clave de mensaje cifrada dos veces, la cual es única y esencialmente aleatoria para cada mensaje. Al descifrar correctamente esta clave de mensaje, obtendremos algo de la forma $xyz\ xyz$, donde x, y y z son letras cualesquiera, lo que quiere decir que los pares $m_i, \overline{m}_i$ vienen de la misma letra.

Con suficientes mensajes recibidos en un día (lo cual, como se ha comentado en la sección 2.1, quiere decir que tenemos el mismo orden de rotores, ajuste interno y clavijero), Rejewski indica en [4] que no es difícil encontrar tres mensajes de la siguiente manera:

$$\begin{array}{lll} g_1^1g_2^1g_3^1, & am_2^1m_3^1 & a\overline{m}_2^1\overline{m}_3^1 \\ g_1^2g_2^2g_3^2, & m_1^2am_3^2 & \overline{m}_1^2a\overline{m}_3^2 \\ g_1^3g_2^3g_3^3, & m_1^3m_2^3a & \overline{m}_1^3\overline{m}_2^3a \end{array}$$

En los documentos de Rejewski, a cada una de estas coincidencias se les llama *samiczka*, en plural, *samiczki*.

Sabemos que la permutación de la máquina es un producto de trasposiciones disjuntas por lo visto al final de la sección 1.2.1, por esto, si tenemos que $m_i = \overline{m}_i = a$ como arriba, esto quiere decir que $x^{\alpha_i} = x^{\alpha_{i+3}} = a$. Por tanto:

$$\left.\begin{array}{rcl} \alpha_i & = & (x\ a)\,\gamma \\ \alpha_{i+3} & = & (x\ a)\,\gamma' \end{array}\right\} \Rightarrow \alpha_i\alpha_{i+3} = (x)(a)\gamma\gamma',$$

donde γ y γ' son permutaciones cualesquiera, las cuales no nos son de particular interés. La permutación $\alpha_i\alpha_{i+3}$ se puede interpretar como la permutación que transforma las letras en la posición i y a las de la posición $i+3$.

Lo segundo es lo que observamos, es decir, que la letra en la posición i y en la posición $i+3$ coinciden, y lo primero quiere decir que, si tenemos la máquina en la configuración correcta, la letra a nos tiene que llevar a x en las dos posiciones. Notamos que de momento no hemos hablado de clavijero, de momento, podemos asumir que $\sigma = 1$.

Parte de el uso de la bomba es que suponemos que x no está en el clavijero, esto quiere decir que $\overline{\alpha_i}$ y $\overline{\alpha_{i+3}}$ llevan a x a la imagen de a por el clavijero.

Lo que hacemos es lo siguiente, para cada uno de los mensajes tomamos un par de máquinas enigma configuradas en una posición que se corresponderá a cada uno de las a. Estas máquinas no tienen clavijero y el ajuste interno de los rotores es el $(0,0,0)$ en todas. El orden de rotores será uno concreto para ambas máquinas. Para los ajustes externos, tenemos en cuenta la diferencia de ajuste conjunto que hay en cada uno de los casos en los que ha aparecido la a. Concretamente:

- Como el *turnover* depende solamente del ajuste externo, miramos el ajuste del rotor de la derecha y determinamos si va a haber *turnover* desde la posición inicial hasta la segunda a de acuerdo a lo explicado al final de la sección 1.2.2. También miraremos el ajuste del rotor central por si nos da *turnover* en el izquierdo.

- Si no hay *turnover*, la pareja de máquinas se configurará de la siguiente manera, dependiendo en la posición de la a:

$$g_1^ig_2^i\left(g_3^i + i - 1\right) \quad \text{y} \quad g_1^ig_2^i\left(g_3^i + i + 2\right)$$

Sumando usando la equivalencia entre letras y $\mathbb{Z}^{26}$.

- Si hay *turnover* entre las dos a en el rotor derecho, configuraremos la máquina teniéndolo en cuenta:

$$g_1^i g_2^i \left(g_3^i + i - 1\right) \quad \text{y} \quad g_1^i \left(g_2^i + 1\right) \left(g_3^i + i + 2\right).$$

  Si se da *turnover* en el rotor central, lo configuraremos la máquina de la siguiente manera:

$$g_1^i g_2^i \left(g_3^i + i - 1\right) \quad \text{y} \quad \left(g_1^i + 1\right) \left(g_2^i + 1\right) \left(g_3^i + i + 2\right).$$

  También podría darse *turnover* antes de la primera a para $i = 2$ o 3, en este caso habría que considerarlo en la configuración de la primera máquina del par.

  En [3], parece que no se considera el turnover, en la sección 2.3.3 veremos que esto lleva a errores.

- Giramos los rotores de ambas máquinas de las tres parejas de manera que siempre están en la misma posición relativa a la inicial, por ejemplo, giramos el rotor izquierdo una posición hacia delante y el derecho cinco posiciones hacia atrás en ambas en los tres pares. Lo que estamos haciendo es, esencialmente, a cada uno de los indicadores, le estamos sumando todas las posibles ternas de elementos de $\mathbb{Z}_{26}$.

  Para cada una de las 26^3 posiciones posibles, comprobamos si al teclear la x obtenemos la misma salida en ambas máquinas, en caso afirmativo, tenemos una posible configuración conjunta que nos da las *samiczki* correspondientes.

  Si denotamos esta configuración conjunta $c_1c_2c_3$ y recordando que todos los rotores tienen configuración interna 0, como $c_j = g_j - r_j$, podemos despejar r_j en $\mathbb{Z}_{26}$.

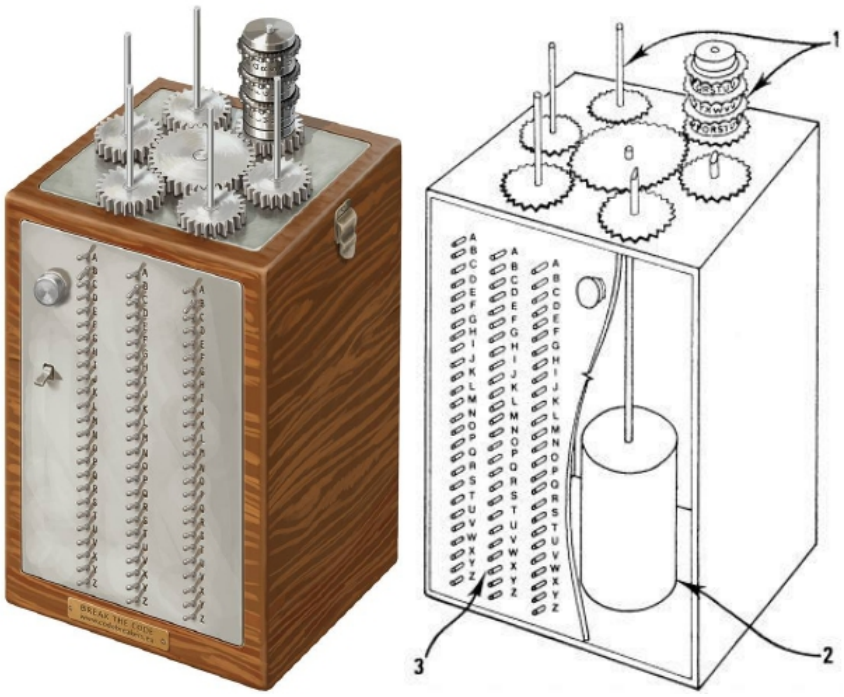

Figura 2.1: Diagrama de la bomba polaca, a la izquierda la recreación cortesía de [1] y a la derecha, el esquema original de Rejewski encontrado en [4]. La leyenda lee: *1: Rotores, 2: motor eléctrico, 3: interruptores*

Podemos ver en la figura 2.1 que la bomba tiene tres pares como los descritos antes, los cuales se configuran como ya se ha discutido, esta máquina se encarga de automatizar el proceso discutido

anteriormente de manera que solamente se para si encuentra un candidato válido en las tres parejas simultáneamente. Cada vez que la bomba se para, se anota la configuración correspondiente a la parada y se hace que continúe en ejecucuión. Como indica Rejewski en [4], como en esta época se usaban solamente tres rotores, se pueden comprobar a la vez todos los ordenes de rotores usando $3! = 6$ bombas. Este proceso completo, según [4], duraba un promedio de dos horas.

Después de obtener las posibles configuraciones internas, configuramos máquinas Enigma normales, ahora con *turnover* incluido, para cada uno de los indicadores y "desciframos" la clave de mensaje. Como hemos ignorado el clavijero, no obtendremos claves de mensaje válidas de la forma *xyz xyz*, pero algunas de las posibles configuraciones internas nos llevarán más cerca que otras, comenzaremos comprobando estas.

Rejewski no detalla como obtenían el clavijero después, pero en [3] se detalla un posible método donde se deducen manualmente posibles conexiones que nos llevan a obtener claves de mensaje plausibles. Si hemos encontrado el ajuste interno y orden de rotores adecuados, y se ha cumplido la suposición de que la *a* queda fija en el clavijero, las letras que se obtienen al descifrar el indicador son las imágenes de las letras de la clave por el clavijero.

Tanto los interruptores visto en la figura 2.1 como el diagrama de circuito que Link muestra en la figura 14 de [3] como una posible manera de fabricar físicamente la bomba, parecen indicar que podríamos hacer funcionar la bomba tomando una letra diferente en cada posición en vez de la misma en tres posiciones distintas, o incluso, cambiando la manera de introducir la configuración inicial, podríamos probar letras en las mismas posiciones.

El hecho de elegir la misma letra siempre es para mitigar el efecto del clavijero, teniendo en cuenta lo comentado en la sección 2.1, el clavijero cambia entre cinco y ocho pares de letras, lo que quiere decir que, de media, afecta a $\frac{10+16}{2} = 13$ letras, es decir, la mitad. Por tanto, si elegimos una letra, tendremos un 50 % de probabilidad de que no esté en el clavijero. Una sencilla cuenta nos permite ver que si usamos dos letras distintas, la probabilidad baja al 24 % y si usamos tres distintas, 11 %.

El hecho de no comprobar la misma posición es que, salvo que comprobemos con letras distintas, lo que, como ya se ha explicado, no es recomendable, estaríamos comprobando lo mismo en ordenes distintos.

Matemáticamente, podemos considerar que, para cada uno de los pares, partiendo de la configuración inicial discutida arriba, se consideran los conjuntos de configuraciones internas posibles de la siguiente manera:

$$\Pi_i(G,a) = \left\{R \in \mathbb{Z}_{26}^3 \mid a^{\tilde{\alpha}_i} = a^{\tilde{\alpha}_{i+3}}\right\}$$

Donde $G = (g_1, g_2, g_3) \in \mathbb{Z}_{26}^3$ es el ajuste interno de la cabecera y *x* la *samiczka*. Entonces, la máquina nos dará:

$$\Pi_i\left((g_1^1, g_2^1, g_3^1), a\right) \cap \Pi_i\left((g_1^2, g_2^2, g_3^2), a\right) \cap \Pi_i\left((g_1^3, g_2^3, g_3^3), a\right)$$

A partir de enero del año 1939, el método de la bomba perdió casi por completo su efectividad, puesto que los alemanes dispusieron de dos rotores más, para elegir tres de ellos para introducir en la máquina. De esta manera, el número de posibles ordenes de rotores aumenta a $5!/2! = 60$. Es decir, para poder aplicar el método de la misma manera, harían falta 60 bombas, lo que estaba fuera del alcance de los servicios secretos polacos. Esto les llevó a buscar métodos alternativos.

Esta bomba fue lo que inspiró a Alan Turing a construir su bomba que tanto éxito tuvo rompiendo Enigma durante la Segunda Guerra Mundial.

2.3.2 El simulador de la Bomba

El código correspondiente a la simulación de la Bomba se halla en el apéndice B y consta de dos clases:

- La clase `ParSincrono` modela cada uno de los pares descritos en la sección anterior. Se inicializa indicando el reflector a usar, representado por una `string` y tres objetos `Rotor`, que representan

respectivamente el rotor izquierdo, central y derecho del par correspondiente. Esta clase tiene varios métodos:

  - `setRotores` toma una lista de enteros, que representan el indicador del ajuste externo discutido en la sección anterior, así como un entero, `isTurnover` que sirve para indicar si tenemos que considerar que hay turnover de la siguiente manera:
    * 0 indica que no hay turnover.
    * 1 indica que el rotor derecho tiene turnover entre las dos samiczki.
    * 2 indica que el rotor central tiene turnover.
  - `rotar` simplemente cambia la posición de los rotores a la siguiente configuración que comprobar.
  - `ver_letra` es similar al método `cifrar_letra` de la clase `Maschine`. Cifra la letra indicada en el parámetro `letra` dos veces y lo devuelve en una tupla.
  - El método mágico `__repr__` nos muestra la posición en la que se hallan cada uno de los rotores.

- La clase `Bomba` modela la máquina tal y como la describe Link en [3], con la modificación discutida del turnover. Se inicializa indicando el reflector como `string`, los rotores como lista de enteros, los indicadores de ajuste externo como lista de cadenas de longitud tres y una lista de enteros que indican en cual de los pares hay que considerar el turnover y en que rotores de la manera especificada antes. De la lista de indicadores, el primero será el que viene de una *samiczka* en las posiciones 1 y 4, el segundo en las posiciones 2 y 5, y el tercero en 3 y 6. Esta clase solo tiene un método explícito y uno mágico:

  - El método `descifra` toma una letra, que será la que coincide en las posiciones discutidas, y hace funcionar la máquina para esa letra. Cada vez que hay una parada, se muestra en pantalla el estado de la bomba en esa parada. El método también devuelve una lista con todos estos estados.
  - El método mágico `__repr__` muestra la configuración externa de cada uno de los pares.

También tenemos una última función, `getPi`, que nos da los conjuntos Π_i discutidos en la sección anterior. Toma como argumentos lo mismo que el constructor de la Bomba con algún parámetro añadido:

- `ref` es el reflector en el mismo formato que antes.
- `ri, rc, rd` son objetos de la clase `Rotor` que representan, respectivamente, el rotor izquierdo, central y derecho del par a computar.
- `GS` es el indicador de ajuste externo.
- `samiczka` es la letra repetida.
- `isTurnover` es un entero que indica si tenemos que considerar que haya habido turnover de la misma manera que en el método `setRotores` de la clase `ParSincrono`.
- `i` es un entero entre 1 y 3 y nos indica cual de los Π_i estamos computando.

Esta función nos devuelve un objeto de tipo `set`, es decir, un conjunto, que contiene los posibles ajustes internos que nos dan las coincidencias. Esto quiere decir que se puede obtener el mismo resultado que usando la clase `Bomba` y luego computando los ajustes internos, que haciendo:

```
getPi(...,1).intersection(getPi(...,2)).intersection(getPi(...,3))
```

2.3.3 Ejemplos con y sin turnover

Para ver la necesidad del turnover, consideramos los siguientes casos. Usando el simulador descrito en A, obtenemos los siguientes casos:

- Con orden de rotores `I-II-III`, ajuste interno `BBC` y externo `DDL`, cifrando la clave de mensaje `DDD`[1] obtendríamos el siguiente indicador:

  `DDL, MGM MNY`

  Computando la $\Pi_1(\texttt{DDL}, \texttt{M})$ utilizando la función definida en el apéndice B.3 de la siguiente manera:

  ```
  "BBC" in getPi(UKW_B, R1, R2, R3, [3,3,11], "M", 0, 1)
  ```

  Obtenemos de salida `True`, lo que quiere decir que nuestra configuración correcta está en el conjunto. En este primer ejemplo no hace falta considerar el turnover, ya que el rotor derecho pasa de la `L` a la `P` y el rotor `III` realiza el turnover entre la `V` y la `W`, vemos ahora un ejemplo en el que sí tenemos turnover.

- Con orden de rotores `I-III-II`, ajuste interno `BBC` y externo `DDB`, cifrando la clave de mensaje `PPP`[2] obtendríamos el siguiente indicador:

  `DDB, TTZ XKZ`

  Como tenemos que el rotor derecho se mueve desde la `B` hasta la `H` hasta la segunda `Z`, y el turnover del rotor `II` está entre la `E` y la `F`, tenemos que considerarlo. Si calculásemos el conjunto $\Pi_3(\texttt{DDB}, \texttt{Z})$ sin tenerlo en cuenta, es decir, ejecutamos el siguiente código para comprobarlo:

  ```
  "BBC" in getPi(UKW_B, R1, R3, R2, [3,3,1], "Z", 0, 3)
  ```

  Nos devolverá `False`. Sin embargo:

  ```
  "BBC" in getPi(UKW_B, R1, R3, R2, [3,3,1], "Z", 1, 3)
  ```

  Sí que nos devuelve `True`, dándonos por tanto la respuesta correcta.

[1] O cualquier clave de la forma `DXY`
[2] O cualquier clave de la forma `XYP`

Bibliografía

[1] *Crypto Museum (Página Web)*. URL: https://www.cryptomuseum.com.

[2] Władysław Kozaczuk. *Enigma: How the German Machine Cipher Was Broken, and How It Was Read by the Allies in World War Two*. Ed. por Christopher Kasparek. Frederick, Md: University Publications of America, 1984. ISBN: 0-89093-547-5. URL: https://archive.org/details/enigmahowgermanm0000koza.

[3] D. Link. "Resurrecting Bomba Kryptologiczna: Archaeology of Algorithmic Artefacts, I". En: *CRYPTOLOGIA* 33.WOS:000265292400008 (2009), págs. 166-182. ISSN: 0161-1194. DOI: 10.1080/01611190802562809.

[4] Marian Rejewski. "How Polish Mathematicians Broke the Enigma Cipher". En: *Annals of the History of Computing* 3.3 (1981), págs. 213-234. DOI: 10.1109/MAHC.1981.10033.

[5] Marian Rejewski. *Memories of my work at the Cipher Bureau of the General Staff Second Department (1930-1945)*. Poznan: Wydawnictwo Naukowe Uniwersytetu im. Adama Mickiewicza, 2011. ISBN: 978-83-232-2237-8.

[6] *Wikimedia (Página web)*. URL: https://commons.wikimedia.org/wiki/File:Enigma_rotors_and_spindle_showing_contacts_rachet_and_notch.jpg.

[7] *Wikipedia (Página web)*. URL: https://en.wikipedia.org/wiki/en:Museo_Nazionale_Scienza_e_Tecnologia_Leonardo_da_Vinci.

Índice alfabético

Apéndice A

Simulador de Enigma

A.1 Funciones auxiliares

```
def letra_a_num(a):
    return ord(a.upper())-ord("A")

def num_a_letra(a):
    return "ABCDEFGHIJKLMNOPQRSTUVWXYZ"[a]

def cicla_permuta(perm, n=1):
    for _ in range(0,n):
        perm = perm[1:]+str(perm[0])
        perm = "".join([num_a_letra((letra_a_num(i)-1)%26) for i in perm])
    return perm

def permuta(perm, direc, a):
    if direc == 1:
        return perm[letra_a_num(a)]
    else:
        return num_a_letra(perm.index(a))

def pasa_clavijero(pares, a):
    if pares == None:
        return a
    elif not (a in pares.keys() or a in pares.values()):
        return a
    else:
        for k, v in pares.items():
            if k == a:
                return v
            elif v == a:
                return k
```

A.2 Rotores

```
class Rotor:
    def __init__(self, to, perm):
        self.rs = 0 # Ringstellung
        self.gs = 0 # Grundstellung
        self.perm_init = perm
        self.to = to # Donde se hace el turnover
        self.perm = perm # Permutacion del rotor
        self.isTurnover = False # Si se tiene o no que hacer turnover
    def set_rs(self, rs):
        self.rs = rs
```

```
        self.perm = cicla_permuta(self.perm, -rs%26)
    def set_gs(self, gs):
        self.gs = gs
        self.perm = cicla_permuta(self.perm_init, gs)
    def permuta_ida(self, a):
        return permuta(self.perm, 1, a)
    def permuta_vuelta(self, a):
        return permuta(self.perm, 0, a)
    def turn(self):
        self.gs = (self.gs + 1) % 26
        self.perm = cicla_permuta(self.perm)
        if (self.gs)%26 == self.to:
            self.isTurnover = True
    def turnover(self):
        if self.isTurnover:
            self.isTurnover = False
            return True
        else:
            return False
    def ver_gss(self,i=0):
        return num_a_letra((self.gs-i)%26)

```

A.3 La máquina

```
class Maschine:
    def __init__(self, r1, r2, r3, ukw, kl):
        self.r1, self.r2, self.r3 = r1, r2, r3
        self.ukw = ukw
        self.kl = kl

    def settings(self, gss, rss):
        self.r1.set_gs(gss[2])
        self.r2.set_gs(gss[1])
        self.r3.set_gs(gss[0])

        self.r1.set_rs(rss[2])
        self.r2.set_rs(rss[1])
        self.r3.set_rs(rss[0])
    def cifrar_letra(self, a):
        self.r1.turn()
        if self.r1.turnover():
            self.r2.turn()
            if self.r2.turnover():
                self.r3.turn()

        elif self.r2.turnover():
            self.r2.turn()
            self.r3.turn()
        ret = pasa_clavijero(self.kl, a)
        ret = self.r1.permuta_ida(ret)
        ret = self.r2.permuta_ida(ret)
        ret = self.r3.permuta_ida(ret)
        ret = permuta(self.ukw,1,ret)
        ret = self.r3.permuta_vuelta(ret)
        ret = self.r2.permuta_vuelta(ret)
        ret = self.r1.permuta_vuelta(ret)
        ret = pasa_clavijero(self.kl, ret)
        return ret
    def cifrar_mensaje(self,X):
        return "".join([self.cifrar_letra(i) for i in X])
```

```
37      def __repr__(self):
38          return "("+num_a_letra(M.r3.gs) + ", "+ num_a_letra(M.r2.gs)+", "+
        num_a_letra(M.r1.gs)+")"
39
```

A.4 Rotores I, II y III

```
1 permR1 = "EKMFLGDQVZNTOWYHXUSPAIBRCJ"
2 permR2 = "AJDKSIRUXBLHWTMCQGZNPYFVOE"
3 permR3 = "BDFHJLCPRTXVZNYEIWGAKMUSQO"
4 R1 = Rotor(letra_a_num("Q")+1, permR1)
5 R2 = Rotor(letra_a_num("E")+1, permR2)
6 R3 = Rotor(letra_a_num("V")+1, permR3)
7 juegoRotores = [R1,R2,R3]
```

Apéndice B

Simulador de la Bomba de Rejewski

B.1 Cada uno de los pares

```
import copy

class ParSincrono:
    def __init__(self, r, i, m, d):
        self.r = r
        self.i1 = copy.deepcopy(i)
        self.m1 = copy.deepcopy(m)
        self.d1 = copy.deepcopy(d)
        self.tod1 = 0
        self.tom1 = 0

        self.i2 = copy.deepcopy(i)
        self.m2 = copy.deepcopy(m)
        self.d2 = copy.deepcopy(d)
        self.tod2 = 0
        self.tom2 = 0

    def setRotores(self, setting, isTurnover):
        self.i1.set_gs(setting[0])
        self.m1.set_gs(setting[1])
        self.d1.set_gs(setting[2])

        if isTurnover == 1:
            self.m2.set_gs((setting[1]+1)%26)
        elif isTurnover == 2:
            self.i2.set_gs((setting[2]+1)%26)
            self.m2.set_gs((setting[1]+1)%26)
        else:
            self.i2.set_gs(setting[0])
            self.m2.set_gs(setting[1])
        self.d2.set_gs((setting[2]+3)%26)

    def rotar(self):
        self.d1.turn()
        self.tod1 += 1

        if self.tod1 >= 26:
            self.tod1 = 0
            self.m1.turn()
            self.tom1 += 1
            if self.tom1 >= 26:
                self.tom1 = 0
                self.i1.turn()

```

```
        self.d2.turn()
        self.tod2 += 1

        if self.tod2 >= 26:
            self.tod2 = 0
            self.m2.turn()
            self.tom2 += 1
            if self.tom2 >= 26:
                self.tom2 = 0
                self.i2.turn()

    def ver_letra(self, letra):
        l1 = self.d1.permuta_ida(letra)
        l1 = self.m1.permuta_ida(l1)
        l1 = self.i1.permuta_ida(l1)
        l1 = permuta(self.r, 1, l1)
        l1 = self.i1.permuta_vuelta(l1)
        l1 = self.m1.permuta_vuelta(l1)
        l1 = self.d1.permuta_vuelta(l1)

        l2 = self.d2.permuta_ida(letra)
        l2 = self.m2.permuta_ida(l2)
        l2 = self.i2.permuta_ida(l2)
        l2 = permuta(self.r, 1, l2)
        l2 = self.i2.permuta_vuelta(l2)
        l2 = self.m2.permuta_vuelta(l2)
        l2 = self.d2.permuta_vuelta(l2)

        return (l1,l2)

    def __repr__(self):
        c1 = self.i1.ver_gss()
        c2 = self.m1.ver_gss()
        c3 = self.d1.ver_gss(1)

        c4 = self.i2.ver_gss()
        c5 = self.m2.ver_gss()
        c6 = self.d2.ver_gss(1)

        return c1+c2+c3+" "+c4+c5+c6

```

B.2 La bomba

```
class Bomba:
    def __init__(self, ref, rotores, gss, to):
        rotores_ = [juegoRotores[i-1] for i in rotores]
        self.enigma1 = ParSincrono(
            ref, rotores_[0], rotores_[1], rotores_[2])
        self.enigma2 = ParSincrono(
            ref, rotores_[0], rotores_[1], rotores_[2])
        self.enigma3 = ParSincrono(
            ref, rotores_[0], rotores_[1], rotores_[2])

        gssnum = []

        for i in gss:
            gsi = []
            for j in i:
                gsi.append(letra_a_num(j))
            gssnum.append(gsi)
```

```
18
19          gssnum[1][2] = (gssnum[1][2]+1)%26
20          gssnum[2][2] = (gssnum[2][2]+2)%26
21
22          self.enigma1.setRotores(gssnum[0], to[0])
23          self.enigma2.setRotores(gssnum[1], to[1])
24          self.enigma3.setRotores(gssnum[2], to[2])
25
26      def descifra(self, samiczka):
27          ret = []
28          for i in range(26**3):
29              a1,a2 = self.enigma1.ver_letra(samiczka)
30              b1,b2 = self.enigma2.ver_letra(samiczka)
31              c1,c2 = self.enigma3.ver_letra(samiczka)
32
33              a = a1 == a2
34              b = b1 == b2
35              c = c1 == c2
36
37              if a and b and c:
38                  print("BREAK " + str(i))
39                  print(str(self) + " --> " + a1 + b1 + c1)
40                  ret.append(str(self))
41
42              self.enigma1.rotar()
43              self.enigma2.rotar()
44              self.enigma3.rotar()
45          return ret
46                  #break
47
48      def __repr__(self):
49          return str(self.enigma1) + ", " + str(self.enigma2) + ", " + str(self.
        enigma3)
50
```

B.3 Obtención de los conjuntos Π_i

```
1  def getPi(ref, ri, rc, rd, GS, samiczka, isTurnover, i):
2      GS[2] = GS[2] + i-1
3      P1 = ParSincrono(ref, ri, rc, rd)
4      P1.setRotores(GS, isTurnover)
5
6      Pip = set({})
7
8      for _ in range(26**3):
9          v = P1.ver_letra(samiczka)
10         if v[0] == v[1]:
11             Pip.add(str(P1))
12         P1.rotar()
13
14     Ret = set({})
15     for v in Pip:
16         vv = v[0:3]
17         ret = ""
18         for y, z in zip(GS, vv):
19             ret = ret + num_a_letra((y-letra_a_num(z))%26)
20         Ret.add(ret)
21     return Ret
```