Universidad de Málaga
Facultad de ciencias

Trabajo Fin de Grado

Trenzas y Criptografía

Braids and Cryptography

Tema: Criptografía

Trabajo de Fin de Grado en Matemáticas

Autor: **Rafael Fernández Ortiz**

Tutor: **Antonio Díaz Ramos**

Área de conocimiento y/o departamento: **Álgebra, Geometría y Topología**

Junio 2019

DECLARACIÓN DE ORIGINALIDAD DEL TFG

D. *Rafael Fernández Ortiz*, con DNI *77184478C*, estudiante del Grado de *Matemáticas* de la Facultad de Ciencias de la Universidad de Málaga,

DECLARO:

Que he realizado el Trabajo Fin de Grado titulado "*Trenzas y Criptografía*" y que lo presento para su evaluación. Dicho trabajo es original y todas las fuentes bibliográficas utilizadas para su realización han sido debidamente citadas en el mismo.

Para que así conste, firmo la presente en Málaga, el *19 de Junio de 2019*

Índice general

Resumen I

Abstract I

Introducción I

1. Preliminares **1**
1. Grupos libres 1
1.1. Palabras. Reducción de palabras 1
1.2. Grupos Libres 2
1.3. Presentación de grupos 3
2. Grupo Simétrico 5
3. Grupos de trenzas de Artin 10

2. Trenzas y Grupos de trenzas **12**
1. Trenzas y diagrama de trenzas 12
1.1. Diagrama de trenzas 13
1.2. Grupos de trenzas 20
1.3. Trenzas puras. Grupos de trenzas puras 23
2. La palabra fundamental Δ. El problema de la palabra 25
2.1. La solución al problema de las palabras 26
2.2. Algoritmo de Artin 32

3. Criptografía de trenzas **35**
1. El problema de la conjugación 35
2. Criptografía. Definiciones 36
3. Criptosistemas basados en grupos de trenzas 39
3.1. Criptosistema Anshel-Anshel-Fisher-Goldfeld 39
3.2. Criptosistema de intercambio Ko et al 42
3.3. Comunicación mediante sistema AAFG y Block Cipher. 44

4. Ataques a la criptografía de trenzas **48**
1. Solución del problema de la conjugación 48
2. Ataque heurístico al problema de la conjugación 50
3. Ataque al problema de la conjugación múltiple simultánea 54
4. Propuesta de ataque 57

Bibliografía **59**

Resumen

Comenzaremos dando una introducción a la teoría de trenzas, tanto desde un punto de vista geométrico como algebraico. Para ello, previamente se repasarán definiciones y resultados en la teoría de grupos libres y grupos simétricos. Presentaremos el grupo de trenzas y sus generadores como presentación de grupo y sus elementos, respectivamente.

Tras esto, desarrollaremos, desde un enfoque geométrico, la idea de trenza, sus diagramas, como interactúan entre sí y algunos resultados sumamente importantes que establecerán una relación esencial entre el los grupos de trenzas, vistos como presentación de grupo y como la descripción geométrica que acabamos de mencionar. De esta forma, podremos trabajar con las trenzas desde una visión geométrica o una visión algebraica según convenga.

Trataremos en detalle el *problema de la palabra* y el *problema de la conjugación*, problemas fundamentales que los grupos de trenzas presentan y que han derivado, como aplicación en la criptografía, en sistemas de intercambio de claves originados por variaciones del esquema propuesto por *Diffie-Helman*.

Por ultimo, trataremos de definir una solución o establecer un método para la resolución del problema de la conjugación como posible ataque frente al sistema criptográfico basado en trenzas. Así, pondremos en juicio la seguridad que dicho sistema establece.

Para el desarrollo de los cálculos entre trenzas utilizaremos un software de computación algebraico, *SageMath*, que nos resultará esencial para poner a prueba los métodos propuestos y realizar distintos *benchmarks*, es decir, pruebas de rendimientos.

Todos los recursos utilizados para el desarrollo de este estudio se pueden encontrar en `https://github.com/rafafrdz/braids-and-cryptography`

Palabras clave:

trenza, grupo de trenzas, grupo de Artin, permutación, trenza pura, palabra fundamental, problema de la palabra, problema de la conjugación, trenza geométrica, criptografía, clave pública, clave privada, simétrico, asimétrico, ataque, algoritmo, sage

Abstract

We will begin by giving an introduction to the theory of braids, both from a geometric and algebraic point of view. To do this, we will previously review definitions and results in the theory of free groups and symmetrical groups. We will present the group of braids and their generators as group presentation and their elements, respectively.

Afterwards, wwe will develop, from a geometric approach, the idea of braid, its diagrams, how they interact with each other and some extremely important results that will establish an essential relationship between the groups of braids, seen as a group presentation and as the geometric description just mentioned. In this way, we will be able to work with the braids from a geometric or algebraic view as appropriate.

We will deal in detail with the word problem and the conjugation problem, fundamental problems that braid groups present and that have derived, as an application in cryptography, in key exchange systems originated by variations of the scheme proposed by *Diffie-Helman*.

Finally, we will try to define a solution or establish a method for solving the problem of conjugation as a possible attack against the cryptographic system based on braids. Thus, we will judge the security that such a system establishes.

For the development of calculations between braids we will use algebraic computing software, *SageMath*, which will be essential for us to test the proposed methods and perform different *benchmarks*.

All the resources used for the development of this study can be found in `https://github.com/rafafrdz/braids-and-cryptography`

key words:

BRAID, BRAID GROUPS, ARTIN GROUPS, PERMUTATION, PURE BRAID, FUNDAMENTAL WORD, WORD PROBLEM, CONJUGATION PROBLEM, GEOMETRIC BRAID, CRYPTOGRAPHY, PUBLIC KEY, PRIVATE KEY, SYMMETRIC, ASYMMETRIC, ATTACK, ALGORITHM, SAGE

Introducción

Desde el comienzo de la existencia del ser humano como especie, la capacidad de relacionarse ha estado muy presente. Una parte fundamental de esta relación ha sido y es la comunicación. La comunicación no es más que la capacidad para transferir cierta información, al que denominamos mensaje, mediante una vía, es decir, hablando, por escrito, por pulsos, etc; y esencialmente a través de un lenguaje. Este componente, el lenguaje, es elemento que hace dar sentido a la comunicación entre los miembros de dicha comunicación, por eso es primordial dar un repaso y contextualizar a través de la historia acerca del lenguaje, es ahí de dónde partiremos y a dónde llegaremos.

Podríamos hacer referencia a los primeros mensajes que se dieron en la historia del ser humano, las pinturas rupestres. Estas pinturas eran la mínima forma de representación que tenían las tribus de aquella época sobre su vida, cultura, etc. Eran mensajes que transcendían a través del tiempo ya sea para transmitir el conocimiento a los descendientes de esa misma tribu, de otras tribus, o a nosotros mismos para luego tener conocimiento de sus formas de vida.

Fig 1. Pinturas rupestres en las cuevas de Altamira, España

Si nos adelantamos unos años más, nos encontramos con los egipcios y sus conocidos y misteriosos jeroglíficos. Se estima que la escritura jeroglífica se comenzó a utilizar hacia 3300 a. C. Durante muchos años se consideró que la muestra más antigua de escritura jeroglífica egipcia era la Paleta de Narmer, encontrada durante unas excavaciones en Hierakonpolis en la década de 1890, que fueron datados ca. 3200 a. C. Sin embargo, en 1998 nuevas excavaciones en Abydos fueron halladas muestras de proto-jeroglíficos, datados del período Naqada III del siglo XXXIII aC.

Fig 2. Jeroglíficos inscritos en las columnas del templo de Karnak, Egipto

Los símbolos eran también figurativos: representaban algo tangible, a menudo fácil de reconocer, incluso para alguien que no conociese el significado del mismo. Y es que, para diseñar la escritura jeroglífica, los egipcios se inspiraron en su entorno: objetos de la vida cotidiana, animales, plantas, partes del cuerpo, etc. Durante el Antiguo, Medio y Nuevo Imperio se calcula que existían alrededor de 700 símbolos jeroglíficos, mientras que en la época greco-latina, su número aumentó a más de 6.000.

Los jeroglíficos se grababan en piedra y madera, o bien, en el caso de la escritura hierática y demótica, con cálamo y tinta sobre papiros, ostraca, o soportes menos perdurables. El empleo de los jeroglíficos grabados se limitaba a los dominios en los que la estética o el valor mágico de las palabras adquirían relevancia: fórmulas de ofrendas, frescos funerarios, textos religiosos, inscripciones oficiales, etc. Desde la época del Imperio Antiguo, la escritura jeroglífica egipcia fue un sistema en el que se mezclaban logogramas, signos consonánticos (simples, dobles, triples e incluso de cuatro o más consonantes) y determinantes.

Claro está, y como era de esperar, este lenguaje se perdió y por tanto era imposible de entender el significado de los jeroglíficos, o eso era lo que se pensaba hasta el hallazgo de la piedra de Rosetta.

La piedra de Rosetta es un fragmento de una antigua estela egipcia inscrita sobre una piedra de granodiorita cuyo contenido es un decreto publicado en Menfis (Egipto) en el año 196 a. C. en nombre del faraón Ptolomeo V. El decreto aparece en tres escrituras distintas: el texto superior en jeroglíficos egipcios, la parte intermedia en escritura demótica y la inferior en griego antiguo. Gracias a que presenta esencialmente el mismo contenido en las tres inscripciones, con diferencias menores entre ellas, esta piedra facilitó la clave para el desciframiento moderno de los jeroglíficos egipcios.

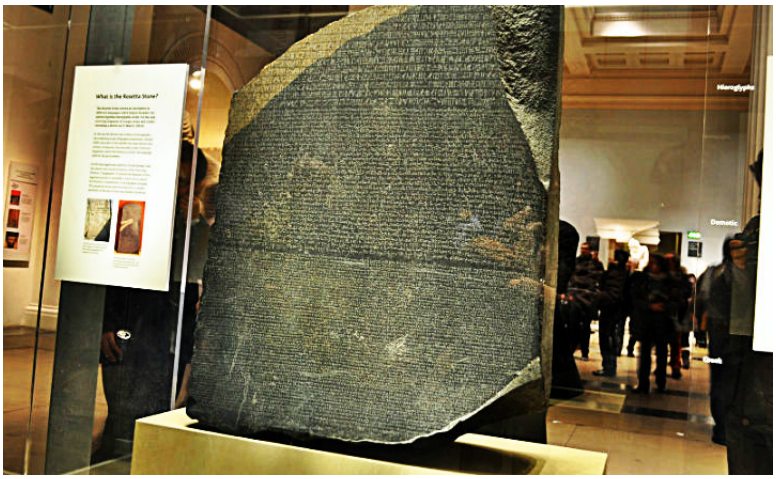


Fig 3. Piedra Rosetta, expuesta actualmente en el Museo Británico, Londres

Como el lector se habrá percatado, acabamos de introducir en nuestro texto la palabra descifrar. Por el momento lo dejaremos estar para darle su correspondiente importancia más adelante.

Son muchos los lenguajes o lenguas que hoy en día conocemos, ya sean muertas (como el latín, el griego clásico, el anglosajon, etc) o vivas (español, inglés, japonés, etc.), lenguas que bien podemos entender o bien no dependiendo de nuestro conocimiento, es aquí dónde nos encontramos con el factor de la interpretación del lenguaje.

Bien es cierto, que no se trata de un problema tan arduo como el de los jeroglíficos. La interpretación de las distintas lenguas en cada punto histórico no ha asumido ningún tipo de sobre esfuerzo. Tanto es así, que a dia de hoy podemos enviar mensajes mediante las tecnologías a otras personas de distintos puntos del planeta y un traductor digital lo interpreta por nosotros, antiguamente y con fines más políticos en su gran medida, ese factor de interprete o traductor estaba formado por una persona o grupo de personas.

Por tanto, vemos de manera clara la sencillez con la que los humanos podemos comunicarnos, la simplicidad de enviarnos mensajes e información, la de transmitir conocimientos, etc. pero, ¿Qué ocurre cuando ese mensaje, esa información o conocimiento, no se desea transmitir de manera tan clara? ¿Qué ocurre cuando ese mensaje se desea enviar a una persona en concreta y que únicamente sepa de su contenido esa persona? Entramos en la fase del ocultismo, es decir, hablamos de técnicas de esteganografías y de criptografía.

Por un lado la **esteganografía** se trata del estudio y aplicación de técnicas que permiten ocultar mensajes u objetos, dentro de otros; por ejemplo, un mensaje dentro de una imagen o fotografía, una grabación de voz a una escala de graves mas bajos dentro de un clip de música, etc. Esta técnica se usaba mucho durante la Segunda Guerra Mundial, se creaban botones para los trajes o zapatos que contenían microfilmes o grabadoras de sonidos.

Fig 4. Viñeta cómica de Mortadelo con el zapatófono (Izq)
Agente 86 con el zapatófono (Der)

Por otro, la **criptografía**, es el campo que estudia las distintas técnicas mediante algoritmos de cifrados, para transformar un mensaje sin tener en cuenta la estructura lingüística ni alterar el mensaje

Remontemonos de nuevo a la antigüedad, nos situamos en la época del Imperio Romano. Entre las innumerosas batallas, la mensajería con información acerca de las respectivas estrategias militares era un frecuente hábito entre generales y altos cargos de la guerra. Sin embargo, las posibilidades de que el enemigo se apoderara de cierta información y por tanto sumase ventaja en la batalla eran muy altas. Se idearon diferentes técnicas de esteganografía y criptografía de mensajes, pero la que podemos destacar como la más eficaz de la época fue el cifrado César.

Esta técnica consistía en asignar a cada letra del abecedario y desplazarla tres posiciones, veamos un caso de uso:

Si quisieramos usar el cifrado César al siguiente mensaje

Hola Mundo

Deberíamos ahora desplazar cada letra tres posiciones, es decir, cambiar cada letra por la letra del abecedario que está situado tres posiciones a su derecha. El resultado sería el siguiente:

Krñd Oxpgr

Así, si se quisiera deshacer el cifrado, es decir, descifrar el mensaje, tan sólo deberíamos revertir el desplazamiento, es decir, cambiar las letras del mensaje cifrado por las letras cuya posición en el abecedario están ubicadas tres posiciones a la izquierda.

Era una técnica bien sencilla, y muy eficaz. Sin embargo esto no duraría para siempre. Luego se fue jugando con los desplazamientos, esta vez de ser tres posiciones podian ser siete, once, catorce, etc. Toda esta técnica tiene una base en la aritmética modular.

Así, dado $\mathfrak{A}$ el abecedario asociado a la lengua del mensaje, $m = len(\mathfrak{A})$, y sea $x \in \mathbb{N}$ tal que $x < m$, podemos definir la letra cifrada correspondiente con desplazamiento k a

$$C_k(x) \equiv x + k \; mod(m)$$

Dato Curioso. En el clásico de la ciencia ficción cinematográfica *2001: Una odisea del espacio* de Stanley Kubrik en 1968. El super ordenador, elemento principal de la trama, es bautizado como HAL 9000, que acaba por volverse loco e intenta asesinar a toda la tripulación de la nave espacial. Pues bien, el nombre HAL no es más que IBM con el cifrado César con desplazamiento uno. Para aquellos que no conozcan IBM se trata de la mayor empresa, al menos en la época, de fabricación de ordenadores. ¿Sería acaso un guiño de Kubrik?

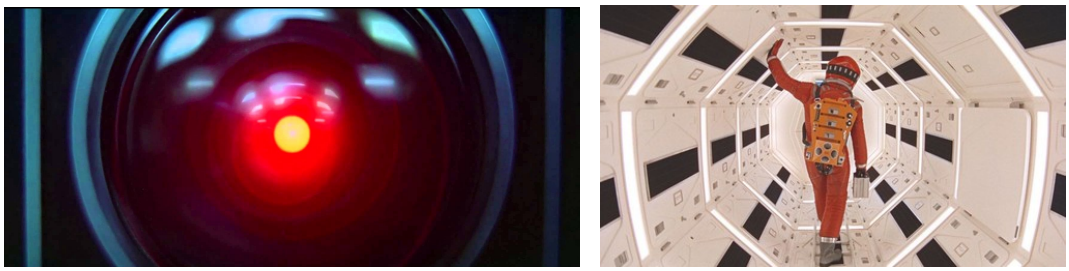


Fig 5. Escenas de *2001: Una odisea en el espacio*

Sin embargo esto no bastaba. Al final, la seguridad de los métodos de cifrados tradicionales eran relativamente sencillos de romper. No fue hasta el año 1976, cuando *Whitfield Diffie* y *Martin Hellman* publicaron lo que sería la mayor innovación en la criptografía, un protocolo basado en el intercambio de claves. Desde entonces, ya no se habla de algoritmos de cifrados sino de sistemas de criptografías, y es por esto, que la seguridad no recaía en el algoritmo en sí, sino en la propia clave.

A raíz de esto, los siguientes campos de investigación en la criptografía ya no se basaba en encontrar un método de cifrado más resistente, sino en ser capaz de proporcionar un sistema que proporcionara un método incapaz de resolverse mediante ingeniería inversa. Pero, ¿qué tiene que ver las trenzas en todo esto?.

Los sistemas criptográficos se han construido debido a los complejos problemas que las matemáticas han originado. En 1947, *Emil Artin* matemático austriaco, publicó un artículo dónde construiría y definiría lo que se conoce como los grupos de Artin, con sus respectivos resultados. Esta teoría, fuertemente relacionada con el problema de la conjugación que expusieron *W. Diffie* y *M. Hellman*, y contribuida con las investigaciones de *Garside*, hicieron surgir aplicaciones en sistemas criptográficos basandose en trenzas.

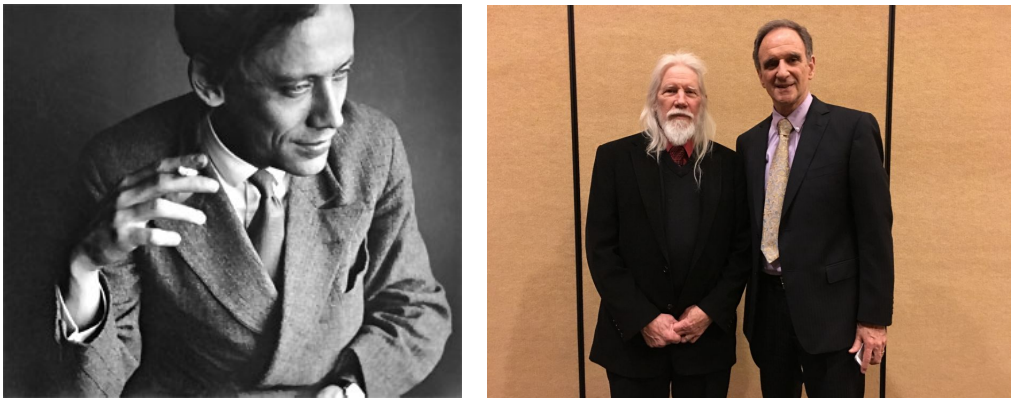


Fig 6. Emil Artin (izq). Diffie y Hellman (der)

Capítulo 1

Preliminares

Es esencial asentar las bases para la teoría de trenzas que desarrollaremos en los capítulos siguientes. Por ello, se hará un repaso a toda la teoría que esta conlleva, es decir, haremos repaso de los grupos libres, la presentación de grupos y los grupos simétricos o de permutación. Cerraremos el capítulo presentando los grupos de trenzas de Artin e introduciendo ciertos resultados claves que tendrán su transcendencia *a posteriori*.

1. Grupos libres

1.1. Palabras. Reducción de palabras

Definición 1.1 (Palabra)**.** *Sea A un conjunto no vacio (no necesariamente finito) de elementos $a_i \in A$ con $i \in \{1, \dots n\}$. Consideramos A como un* ***alfabeto*** *y los elementos a_i como las letras de dicho alfabeto. Definimos una* ***palabra*** *w como la yuxtaposición de las a_i de A.*

$$w = a_1^{\epsilon_1} a_2^{\epsilon_2} \cdots a_n^{\epsilon_n} \qquad a_i \in A,\ \epsilon_i \in \mathbb{Z}$$

De este modo, denotamos $\mathcal{L}(w)$ como la longitud de la palabra w y al elemento unidad $w_e := e$, representado como la palabra vacía, que tiene longitud $\mathcal{L}(w_e) = 0$

Ejemplo 1.1. Sea $A = \{a_1, a_2, a_3\}$. Entonces

$$a_1 a_3^{-4} a_2^2 a_3, \qquad a_2^4 a_3^{-1} a_1 a_3, \qquad a_1^7$$

son ejemplos de palabras de A.

■

Sobre estas palabras se realizan modificaciones que son naturales a ellas mismas, estas son las *contracciones elementales*. Estas contracciones consisten en remplazar $a_i^n a_i^m$ por a_i^{n+m} en una palabra y el elemento a_i^0 de una palabra, en la palabra vacia w_e o directamente quitarla.

Al proceso de hacer un número finito de contracciones elementales se denomina *reducción de palabra* y, por consiguiente, la palabra resultante de dicha reducción se denomina *palabra reducida*, cuya característica relevante es la imposibilidad de efectuar más contracciones elementales.

Nótese que estas contracciones elementales equivalen formalmente a las manipulaciones usuales de exponentes enteros.

Ejemplo 1.2. La forma reducida de la palabra

$$a_2^3 a_2^{-1} a_3 a_1^2 a_1^{-7}$$

es $a_2^2 a_3 a_1^{-5}$

■

Observación 1.1. Es necesario advertir que, aunque parezca obvio la forma en la que se reduce las palabras, se ha de ser consciente de la existencia de una demostración que lo respalda. Son pocos los autores que la exponen y su desarrollo es bastante tediosa[1], motivo por el cual presento tales transformaciones como *evidentes*.

Definición 1.2 (Reverso)**.** *Sea* $w = a_1^{\epsilon_1} \cdots a_i^{\epsilon_i}$ *con* $\epsilon_i = \pm$ *para un cierto* $i \in \{1, \dots n\}$*, definimos el* reverso *de* w *y lo denotamos como* $rev(w)$ *a la palabra*

$$rev(w) = a_i^{\epsilon_i} \cdots a_1^{\epsilon_1}$$

Observamos que dadas dos palabras w y w', se tiene que $rev(ww') = rev(w')rev(w)$.

1.2. Grupos Libres

Sea A un conjunto no vacío. Sea $F[A]$ el conjunto de todas las palabras reducidas tomadas de A. Se tiene que $F[A]$ con la operación $\cdot$ (producto), definida de forma tal que para w_1 y w_2 palabras en $F[A]$, $w_1 \cdot w_2$ es la forma reducida de la palabra obtenida por la yuxtaposición $w_1 w_2$ de dichas dos palabras; se denomina como el **grupo libre** $F[A]$ o el **grupo libre generado** por A.

Ejemplo 1.3. Sea $w_1 = a_2^3 a_1^{-5} a_3^2$ y $w_2 = a_3^{-2} a_1^2 a_3 a_2^{-2}$ entonces

$$w_1 w_2 = a_2^3 a_1^{-3} a_3 a_2^{-2}$$

■

Es trivial que tal operación producto en $F[A]$ está bien definida y es asociativa. La palabra unidad o vacía w_e actúa como elemento identidad o neutro y, dada una palabra reducida $w \in F[A]$ de la forma $w = a_1^{\epsilon_1} a_2^{\epsilon_2} \cdots a_n^{\epsilon_n}$, el elemento inverso w^{-1} es de la forma $w^{-1} = a_n^{-\epsilon_n} a_{n-1}^{-\epsilon_{n-1}} \cdots a_2^{-\epsilon_2} a_1^{-\epsilon_1}$, así:

$$ww^{-1} = e = w^{-1}w$$

Así una definición más precisa:

Definición 1.3 (Grupo libre)**.** *Si* G *es un grupo con un conjunto* $A = \{a_i\}$ *de generadores y además* G *es isomorfo a* $F[A]$ *de la siguiente forma* $\phi : G \longrightarrow F[A]$ *tal que* $\phi(a_i) = a_i$*, entonces* G *es libre en* $\{a_i\}$ *y las* a_i *son los generadores libres de* G*. Un grupo es libre si es libre en algún conjunto* $\{a_i\}$ *no vacío.*

Ejemplo 1.4. Un ejemplo de grupo libre es $(\mathbb{Z}, \cdot)$ el cual es libre en $\{1\}$ y en $\{-1\}$.

■

[1] Podemos encontrar su demostración en *Some combinatorial aspects of reduced words in finite coxeter groups* de J.R Stembridge, o en *Reduce words and plane partitions* de S. Fomin y A.N. Kirillov

Observación 1.2. Es trivial que todo grupo libre es infinito

Teorema 1.1. *Sea A y B dos conjuntos no vacios. Si un grupo G es libre en A y también lo es en B, entonces los conjuntos A y B tienen el mismo número de elementos.*

Esto es, cualesquiera dos conjuntos de generadores libres de un grupo libre tiene la misma cardinalidad.

Definición 1.4 (Rango del grupo libre)**.** *Si G es libre en A, el número de elementos en A es el rango del grupo libre G.*

Teorema 1.2. *Dos grupos libres son isomorfos sí y sólo sí tienen el mismo rango.*

Teorema 1.3. *Sea $H \leq G$ un subgrupo propio no trivial de un grupo libre G, entonces H es grupo libre.*

1.3. Presentación de grupos

La idea de *presentación de grupo* es formar un grupo dando un conjunto de generadores y ciertas ecuaciones o relaciones que deseamos que satisfagan los generadores. Se desea que el grupo sea tan libre como lo sean los generadores que estén sujetos a dichas relaciones.

Ejemplo 1.5. Supongamos que G tiene generadores x e y, y es libre excepto por la relación $xy = yx$, que podemos verlo o expresarlo de la forma $xyx^{-1}y^{-1} = 1$. Es claro que dicha relación es precisamente la que aporta la conmutatividad en G, mientras que $xyx^{-1}y^{-1}$ sea sólo uno de los muchos conmutadores posibles de $F[\{x, y\}]$. Así, G es abeliano libre en dos generadores y es isomorfo a $F[\{x, y\}]$ módulo el menor subgrupo normal de $F[\{x, y\}]$ que contiene a $xyx^{-1}y^{-1}$.

■

Este ejemplo, muestra una generalización. Sea $F[A]$ un grupo libre, supongamos que deseamos formar un nuevo grupo lo más parecido a $F[A]$, sujeto a ciertas relaciones que se deben satisfacer. Como cualquier relación se puede expresar de tal forma que el segundo miembro sea 1, podemos considerar las relaciones como $r_i = 1$, donde $r_i \in F[A]$. De forma inmediata se debe tener que

$$x^{-1} r_i^n x = 1$$

para cualquier $x \in F[A]$ y $n \in \mathbb{Z}$. Además, cualquier producto de elementos iguales a 1 será de nuevo igual a 1. De esta forma, cualquier producto finito de la forma

$$\prod_j x_j^{-1} r_{i_j}^{n_j} x_j$$

donde las r_{i_j} no necesariamente son distintas, tendrá que ser igual a 1 en el nuevo grupo.

Es sencillo ver que el conjunto de todos estas relaciones forman un subgrupo normal $\mathfrak{R}$ de $F[A]$. De esta forma, el grupo que buscamos es (al menos isomorfo a) $F[A]/\mathfrak{R}$. Teniendo más o menos claro la idea, formalicemos esto que queremos decir

Definición 1.5 (Presentación de grupo)**.** *Sea A un conjunto y sea $\{r_i\} \subseteq F[A]$. Sea $\mathfrak{R}$ el menor de los subgrupos normales de $F[A]$ que contiene a las r_i. Una **presentación** de G es un isomorfismo ϕ de $F[A]/\mathfrak{R}$ sobre el grupo G. Así, los conjuntos A y $\{r_i\}$ consituyen una **presentación de grupo**. Donde:*

1. *El conjunto A es el* ***conjunto de generadores*** *de la presentación.*

2. *Los r_i son los conectores.*

3. *Una ecuación de la forma $r_i = 1$ es una relación.*

Diremos que una presentación es **finita** o que es una **presentación finita** si tanto A como $\{r_i\}$ son conjuntos finitos.

Así, dado un conjunto $\{x_i\}$ de generadores y $\mathfrak{R}$ el conjunto de las relaciones. La expresión de la presentación de grupo la denotamos explicitamente de la siguiente forma $\langle\{x_i\} \mid \mathfrak{R}\rangle$. Por tanto, podemos referirnos a $F[A]/\mathfrak{R}$ como el grupo de presentación $\langle A \mid \mathfrak{R}\rangle$

Observación 1.3. Las relaciones tal que el segundo miembro de la igualdad sea el elemento neutro, suele omitirse la igualdad y dicho miembro, es decir, al grupo

$$\left\langle A \mid s_1 s_2 = s_2 s_1^{-1}, s_3 s_2 = 1 \right\rangle$$

lo podemos ver de la forma

$$\left\langle A \mid s_1 s_2 = s_2 s_1^{-1}, s_3 s_2 \right\rangle$$

■

En el Ejemplo 1.5, se trataba de una presentación finita donde el conjunto $\{x, y\}$ es el conjunto de generadores, mientras que $xyx^{-1}y^{-1}$ es el único conector. La ecuación $xyx^{-1}y^{-1} = 1$ o $xy = yx$ es una relación.

Ejemplo 1.6. Consideremos $A = \{a\}$ y $\mathfrak{R} = \{a^n\}$ con $n \in \mathbb{N}$. Entonces la presentación $\langle a \mid a^n\rangle$ es isomorfo a $\mathbb{Z}_n$

■

De este modo, podriamos ver que el grupo $\langle a \mid a^6\rangle$ es isomorfo a $\mathbb{Z}_6$

Ejemplo 1.7. Consideremos ahora el grupo definido por dos generadores a y b con las relaciones $a^2 = 1$, $b^3 = 1$ y $ab = ba$; esto es la presentación

$$\left\langle a, b \mid a^2, b^3, aba^{-1}b^{-1} \right\rangle$$

La relación $a^2 = 1$ da $a = a^{-1}$, $b^3 = 1$ da $b^{-1} = b^2$. Así, todo elemento en este grupo puede escribirse como un producto de potencias no negativas de a y b. Por otro lado, la relación $ab = ba$ nos permite escribir primero todos los factores con a y después con b. De aquí que todo elemento del grupo es igual a algún $a^n b^m$. Sin embargo, como $a^2 = 1$ y $b^3 = 1$, se tiene que en este grupo hay como mucho seis elementos distintos

$$1, b, b^2, a, ab, ab^2$$

Por otro lado, consideramos el morfismo $\Psi :\longrightarrow \mathbb{Z}_6$ tal que $\Psi(a) = \bar{3}$ y $\Psi(b) = \bar{2}$. Así que, como:

$$\begin{aligned} a^2 = 1 &\longrightarrow \bar{3}^2 = \bar{3} + \bar{3} = \bar{0} \\ b^3 = 1 &\longrightarrow \bar{2}^3 = \bar{2} + \bar{2} + \bar{2} = \bar{0} \\ aba^{-1}b^{-1} = 1 &\longrightarrow \bar{3} + \bar{2} + \bar{3} + \bar{2} + \bar{2} = \bar{0} \end{aligned}$$

Por tanto, se tiene que $\bar{n} = p \cdot \bar{3} + q \cdot \bar{2} = p\,\Psi(a) + q\,\Psi(b) = \Psi\,(pa + qb)$. Esto es, que Ψ es sobreyectiva, lo cual dice que tiene exáctamente 6 elementos. Por tanto, esta presentación también da un grupo de orden 6 que es abeliano. El grupo es isomorfo a $\mathbb{Z}_6$.

■

Con este ejemplo queremos dar a entender que diferentes presentaciones pueden ofrecer grupos isomorfos. Cuando esto ocurre decimos que son **presentaciones isomorfas**. En general, es difícil saber si dos presentaciones son isomorfas, se ha demostrado que en un buen número de dichos problemas relacionados con esta teoría no son, en general, solubles.

Dentro de estos problemas no solubles incluyen el problema de decidir, además del mencionado, cuando un grupo dado por una presentación es finito, libre, abeliano o trivial, y el famoso ***problema de la palabra***, que consiste en determinar en qué caso una palabra r dada es consecuencia de un conjunto dado de palabras $\{r_i\}$, que como veremos más adelante, este problema de la palabra nos dará juego en los próximos capítulos de este trabajo.

Definición 1.6. *Dados $g, g' \in G$, decimos que son conjugados sí existe $h \in G$ tal que $g = h^{-1}g'h$.*

2. Grupo Simétrico

Definición 1.7 (Permutación)**.** *Sea X un conjunto no vacío, llamamos* permutación *de X a una biyección $\sigma : X \longrightarrow X$. Denotamos al conjunto de todas las permutaciones de X como $\mathfrak{S}_X$*

Dotaremos a la representación de las permutaciones $\sigma \in \mathfrak{S}_n$ de la siguiente forma

$$\sigma = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ \sigma(x_1) & \sigma(x_2) & \dots & \sigma(x_n) \end{pmatrix}$$

En general, las permutaciones de un conjunto X forman un grupo junto a la composición de aplicaciones como operación binaria.

Si X es un conjunto finito o infinito numerable, podemos asumir $X = \{1, 2, \dots n\}$, se entiende esto pues, como podemos establecer una biyección entre X y el conjunto $\{1, 2, \dots n\}$ donde $n = \#X$, una permutación en X la podemos tratar como una permutación en el conjunto $\{1, 2, \dots n\}$ de esta forma, la expresión anterior queda resumida de la forma

$$\sigma = \begin{pmatrix} 1 & 2 & \dots & n \\ \sigma(1) & \sigma(2) & \dots & \sigma(n) \end{pmatrix}$$

Así, consideraremos desde ahora el conjunto $X = \{1, 2, \dots n\}$ y por tanto a $\mathfrak{S}_X$ lo denotaremos como $\mathfrak{S}_n$. El siguiente teorema dice que $\mathfrak{S}_n$ es un grupo, el llamado *grupo simétrico* o *grupo de permutaciones* de n letras.

Teorema 1.4. *Sea X un conjunto con n elementos. Sea $\mathfrak{S}_n$ el conjunto de todas las permutaciones de X. Se tiene que $\mathfrak{S}_n$ junto a la composición tiene estructura de grupo.*

Demostración. La asociatividad de la composición de aplicaciones dota a la operación binaria la propiedad asociativa. El elemento neutro en $\mathfrak{S}_n$ es justo la aplicación identidad, que envía cada elemento consigo mismo. Mientras que, al tratarse de aplicaciones biyectivas, para cada $\sigma \in \mathfrak{S}_n$, la aplicación inversa σ^{-1} es el elemento inverso en $\mathfrak{S}_n$. ■

Aunque es natural la multiplicacion de izquierda a derecha de elementos en un grupo, cuando hablamos de la composición, las funciones se componen de derecha a izquierda, es decir, sea σ y σ' permutaciones en un conjunto X, para calcular $\sigma\sigma'$:

$$\sigma\sigma' = (\sigma \circ \sigma')(x) = \sigma(\sigma'(x))$$

Esto es, primero hacemos σ' y después σ.

Definición 1.8. *Sea X un conjunto no vacío, sea $\mathfrak{S}_X$ el conjunto de las permutaciones de X. Dado $\sigma \in \mathfrak{S}_X$ y $x_i \in X$, decimos que σ **fija** a x_i si $\sigma(x_i) = x_i$. Decimos que lo **mueve** si $\sigma(x_i) \neq x_i$.*

Definición 1.9 (r-ciclo)**.** *Sean $i_1, i_2, \cdots i_r$ enteros distintos entre 1 y n. Dado $\sigma \in \mathfrak{S}_n$ tal que*

$$\sigma(i_1) = i_2,\ \sigma(i_2) = i_3,\ \ldots\ \sigma(i_{r-1}) = i_r,\ \sigma(i_r) = i_1$$

*y, además, fija los $n - r$ enteros restantes. Entonces decimos que σ es un **r-ciclo**, y por tanto, se puede decir que tiene longitud r o que es de orden r. Se denota el ciclo σ por $(i_1, i_2, \ldots i_r)$*

Observación 1.4. Podemos ver que un 1-ciclo fija un elemento de X, así todos los 1-ciclos son iguales a la identidad. Usualmente se omite escribir los 1-ciclos ya que representan los puntos no movidos por la permutación.

Por otro lado, un 2-ciclo, el cual lo único que hace es intercambiar un par de elementos, se denomina **trasposición**. Esto es, consideraremos las transposiciones $\tau_1, \tau_2, \ldots \tau_n \in \mathfrak{S}_n$ donde τ_i permuta i con $i + 1$ dejando fijos los demás elementos.

Ejemplo 1.8. La permutación

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 6 & 3 & 5 & 1 & 4 & 2 & 7 \end{pmatrix} = (162354)$$

es un 6-ciclo. Por otro lado,

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 4 & 2 & 3 & 5 & 6 \end{pmatrix} = (243)$$

es un 3-ciclo.

■

Definición 1.10 (Permutaciones disjuntas)**.** *Se dice que dos permutaciones $\sigma, \sigma' \in \mathfrak{S}$ son* disjuntas *si para todo $x \in X$, $\{x \mid \sigma(x) \neq x\} \cap \{x \mid \sigma'(x) \neq x\} = \emptyset$.*

Esto es, que cada x movida por una, es fijado por la otra. Una familia de permutaciones $\{\sigma_i\}_{i\in}$ es disjunta si cada par de ellas son disjuntas.

Ejemplo 1.9. Los ciclos (135) y (27) son disjuntos; sin embargo, los ciclos (135) y (347) no lo son. Podemos ver esto calculando sus productos, que como sabemos, no es mas que hacer la composición:

$$(135)(27) = (135)(27)$$

$$(135)(347) = (13475)$$

El producto de dos ciclos no disjuntos pueden reducirse a algo mas sencillo, tal y como acabamos de ver. Sin embargo, el producto de ciclos disjuntos es irreducible.

■

Los ciclos son unas permutaciones a partir de las cuales se pueden construir todas las demas.

Teorema 1.5 (Factorización)**.** *Sea α una permutación en $\mathfrak{S}_n$, se tiene que, o bien α es un ciclo o un producto de ciclos.*

La demostración se desarrollará por inducción sobre el número k de puntos que son movidos.

Demostración. Para $k = 0$ es trivial, pues es la propia identidad y por definición es un 1-ciclo. Supongamos $k > 0$, sean i_1 un punto movido por α, es decir, definimos $\alpha(i_1) = i_2$, $\alpha(i_2) = i_3$, $\dots$ $\alpha(i_r) = i_{r+1}$, donde r es el menor entero para el cual $i_{r+1} \in \{i_1, i_2, \dots i_r\}$, la lista de los k puntos que α mueve. Esta lista de puntos debe ser finita pues solo hay n puntos posibles que puedan ser movidos. Así, supongamos que $\alpha(i_{r+1}) = i_1$; en caso contrario, podríamos suponer $\alpha(i_{r+1}) = i_j$ para algún $j \geq 2$, sin embargo se tiene que $\alpha(i_{j-1}) = i_j$, lo que llevaría a una contradicción pues α es inyectiva.
Sea σ el ciclo $(i_1 i_2 \cdots i_r)$.

- Si $r = n$, entonces α es el ciclo σ.
- Si $r < n$, y A consta de los de los $n - r$ puntos restantes, entonces se tiene que $\alpha(A) = A$ y $\sigma(A)$ fija los puntos de A. Así:

  $$\sigma|_{\{i_1, i_2, \dots i_r\}} = \alpha|_{\{i_1, i_2, \dots i_r\}}$$

  Es decir, la permutación α restringido al conjunto $\{i_1, i_2, \dots i_r\}$ es exactamente el ciclo σ. Sea α' la permutación tal que $\alpha'(A) = \alpha(A)$ y la cual fija a los puntos $\{i_1, i_2, \dots i_r\}$, entonces α' y σ son disjuntos y se tiene que $\alpha = \sigma\alpha'$. Puesto que α' fija menos puntos que α por inducción se tiene que α' es un ciclo o un producto de ciclos. Y por tanto, se tiene que α es un producto de ciclos.

■

Si todos los ciclos son 1-ciclos, entonces estamos hablando de que esa permutación es la identidad. La multiplicación de permutaciones no necesariamente es conmutativa. Es fácil ver que $\mathfrak{S}_n$ con $n \geq 3$ es no abeliano.

Teorema 1.6. *El grupo de permutaciones $\mathfrak{S}_n$ con $n \geq 3$ es no abeliano.*

Para la demostración basta verlo para el grupo $\mathfrak{S}_3$

Demostración. Sea $(12), (23) \in \mathfrak{S}_3$ se tiene que

$$(12)(23) = (123)$$

mientras que

$$(23)(12) = (132)$$

■

Esto va a ser clave para el desarrollo, más adelante, del grupo de trenzas. Así, un resultado que nos puede facilitar la conmutatividad en los grupos de permutación es el siguiente

Lema 1.1. *Sea σ, σ' dos ciclos disjuntos, entonces conmutan, es decir, $\sigma\sigma' = \sigma'\sigma$.*

Demostración. Sea $x \in X$, se tiene que como σ es disjunto a σ', esto es que si σ permuta a x, σ' lo deja fijo, entonces se tiene que $(\sigma \circ \sigma')(x) = \sigma(x) = (\sigma' \circ \sigma)(x)$. Por otro lado, si tanto σ como σ' deja invariante a x, se tiene que $(\sigma \circ \sigma')(x) = x = (\sigma' \circ \sigma)(x)$.

■

Definición 1.11 (Factorización completa)**.** *Una factorización completa de una permutación α es una factorización de α como producto de ciclos disjuntos los cuales contienen a los 1-ciclos para cada i que fija la permutación.*

Teorema 1.7. *Todo elemento de $\mathfrak{S}_n$ se puede expresar como una única factorización completa de ciclos disjuntos salvo reordenación de dichos ciclos.*

Así, podemos decir que el orden de una permutación α de $\mathfrak{S}_n$ es igual al mínimo común múltiplo de las distintas longitudes de los distintos ciclos que forma la factorización de α.

Ejemplo 1.10. Consideramos el ciclo $\sigma = (5231)$, entonces $\sigma^2 = (53)(21)$, $\sigma^3 = (5132)$ y $\sigma^4 = 1$. Entonces, se tiene que el orden de σ es igual a 4, es decir, la longitud del ciclo.

Sea

$$\alpha = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 6 & 9 & 4 & 1 & 8 & 3 & 5 & 7 & 2 \end{pmatrix}$$

Tenemos que $\alpha = (1634)(29)(587)$ es la descomposición en productos de ciclos disjuntos para α. Así, el orden de α es $mcm\{4, 3, 2\} = 12$

Consideremos ahora α^{1000}, donde α es el mismo del caso anterior. Puesto que el orden de α era 12, se tiene que

$$\alpha^{1000} = \alpha^{(12 \cdot 83 + 4)} = (\alpha^{12})^{83} + \alpha^4 = \alpha^4$$

Como además sabemos que $\alpha = (1634)(29)(587)$ y los ciclos disjuntos conmutan entre sí, obtenemos que $\alpha^{1000} = \alpha^4 = ((1634)(29)(587))^4 = (1634)^4(29)^4(587)^4 = (587)^4 = (587)^3(587) = (587)$. Así, se tiene finalmente que

$$\alpha^{1000} = (587)$$

y este tiene orden 3.

■

Recordamos que cuando hablamos del concepto de transposición, hablamos de un 2-ciclo, es decir una permutación que intercambia dos elementos y fija los demás. De esto, podemos dar lugar a otro tipo de factorización de permutaciones.

Teorema 1.8. *Todo ciclo puede descomponerse como producto de transposiciones.*

Demostración. Basta comprobar la igualdad siguiente

$$(1, 2, \dots r) = (1\ r)(1\ (r-1)) \dots (1\ 2)$$

recordando que la composición de aplicaciones se evalúa de derecha a izquierda. ■

Corolario 1.1. *Toda permutación se puede escribir como un producto de transposiciones.*

Notamos que la descomposición a la que nos referimos en el corolario anterior ya no es única. Basta observar que $1 = (12)(12)$ con lo cual si definimos $\sigma = (36)(13)(25)$ entonces podemos verlo también de la forma $\sigma = (36)(13)(25)(12)(12)$ o bien $\sigma = (12)(12)(36)(13)(25)$, entre otras posibilidades. Además, las trasposiciones que ocurren no son necesariamente conmutativas: $(13)(12) = (123)$ y $(12)(13) = (132)$

Sin embargo, existe una propiedad que se mantiene en todas las descomposiciones posibles como producto de transposiciones para una permutación dada. Probaremos que la *paridad* del número de factores es igual para todas las factorizaciones posibles de permutaciones.

Definición 1.12 (Paridad)**.** *Decimos que una permutación $\alpha \in \mathfrak{S}_n$ es **par** si se puede escribir como producto de un número par de transposiciones. En cualquier otro caso, decimos que es **impar**.*

Definición 1.13 (Signo o Signatura)**.** *Sea $\alpha \in \mathfrak{S}_n$ y $\alpha = \beta_1\beta_2 \dots \beta_t$ es una factorización completa en ciclos disjuntos. Definimos el **signo** de α como*

$$sgn(\alpha) = (-1)^{n-t}$$

Consideremos τ una transposición, esto es que mueve dos elementos y fija los $(n-2)$ elementos restantes, por lo tanto $t = (n-2) + 1 = n - 1$, así:

$$sgn(\tau) = (-1)^{n-(n-1))} = -1$$

Lema 1.2. *Si $\alpha \in \mathfrak{S}_n$ y τ una transposición, entonces*

$$sgn(\tau\alpha) = -sgn(\alpha)$$

Demostración. Sea $\tau = (ab)$ una transposición y sea $\beta = \gamma_1 \cdots \gamma_t$ una factorización completa de β en ciclos disjuntos. Si a y b se intercambian en el mismo ciclo γ_i, supongase γ_1, entonces $\gamma_1 = (ac_1c_2 \cdots c_k bd_1d_2 \cdots d_l)$ con $k, l \geq 0$. Entonces se tiene que

$$\tau\gamma_1 = (ac_1c_2 \cdots c_k)(bd_1d_2 \cdots d_l)$$

Así, $\tau\beta = (\tau\gamma_1)\gamma_2 \cdots \gamma_t$ es una factorización completa con un ciclo extra. Por lo tanto

$$sgn(\tau\beta) = (-1)^{n-(t+1)} = -sgn(\beta)$$

La otra posibilidad es que a y b aparezcan en distintos ciclos, supongamos que $\gamma_1 = (ac_1 \cdots c_k)$ y $\gamma_2 = (bd_1 \cdots d_l)$ con $k, l \geq 0$. Entonces, se tendría que $\tau\beta = (\tau\gamma_1\gamma_2)\gamma_3 \cdots \gamma_t$. Así

$$(\tau\gamma_1\gamma_2) = (ac_1c_2 \cdots c_k bd_1d_2 \cdots d_l)$$

y en este caso, la factorización completa de $\tau\beta$ tendría un ciclo menos de los que tiene β, y entonces

$$sgn(\tau\beta) = (-1)^{n-(t-1)} = -sgn(\beta)$$

■

Teorema 1.9. *Sea α,$\beta \in \mathfrak{S}_n$, entonces*

$$sgn(\alpha\beta) = sgn(\alpha)sgn(\beta)$$

Demostración. Desarrollaremos la demostración por inducción. Consideremos $\alpha = \tau_1 \cdots \tau_m$ una factorización de α en transposiciones.

- Supongamos $m = 1$.

$$\begin{array}{rclcl} sgn(\alpha\beta) & = & sgn(\tau_1\beta) & = & -sgn(\beta) \\ & = & sgn(\tau_1)sgn(\beta) & = & sgn(\alpha)sgn(\beta) \end{array}$$

- Supongamos cierto para m. Veamos para $m+1$

$$\begin{array}{rclcl} sgn(\alpha\beta) & = & sgn(\tau_1 \cdots \tau_m\tau_{m+1}\beta) & = & sgn(\tau_1)sgn(\tau_2 \cdots \tau_m\tau_{m+1}\beta) \\ & = & sgn(\tau_1)sgn(\tau_2 \cdots \tau_m\tau_{m+1})sgn(\beta) & = & sgn(\tau_1 \cdots \tau_m\tau_{m+1})sgn(\beta) \\ & = & sgn(\alpha)sgn(\beta) & & \end{array}$$

■

3. Grupos de trenzas de Artin

Tras la introducción a las presentaciones de grupos y grupos simétricos, vamos a definir los grupos de Artin como tales, es decir, mediante generadores y determinadas relaciones.

Definición 1.14. *Llamamos* grupo de Artin *o* grupo de trenzas de Artin *y lo denotamos como B_n, al grupo generado por los $n-1$ generadores $\sigma_1 \ldots \sigma_{n-1}$ tales que verifican las siguientes relaciones:*

$$\begin{array}{rl} \sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1} & i = 1, 2, \ldots n-2 \\ \sigma_i\sigma_j = \sigma_j\sigma_i & i, j = 1, 2, \ldots n-1 \text{ con } |i-j| \geq 2 \end{array}$$

Estas relaciones se denominan *relaciones de Artin.*

Observación 1.5. Por definición, se tiene que $B_1 = \{1\}$, es el grupo trivial, B_2 es el grupo generado por un sólo elemento σ_1 con ausencia de las relaciones de Artin; así B_2 se puede ver como grupo cíclico.

Ahora como sigue, podemos ver el grupo de trenzas B_n como un subgrupo de $\mathfrak{S}_n$.

Lema 1.3. *Sea G un grupo y sean $s_1, s_2, \ldots, s_{n-1}$ sus elementos tales que satisfacen las relaciones de Artin, entonces hay un único homomorfismo f de grupos tal que*

$$f : B_n \longrightarrow G$$

con $s_i = f(\sigma_i)$, $\forall i = 1, 2, \ldots, n-1$.

Ahora bien, podemos aplicar el lema 1.3 considerando al grupo simétrico $G = \mathfrak{S}_n$, dónde un elemento de $\mathfrak{S}_n$ es una permutación del conjunto $\{1, 2, \ldots, n\}$. Consideramos las **trasposiciones simples** $s_1, \ldots s_{n-1} \in \mathfrak{S}_n$, donde s_i permuta i con $i+1$ y deja fijo los otros elementos de $\{1, 2, \ldots n\}$. Es trivial ver como las trasposiciones verifican las relaciones de Artin. Así, por el Lema 1.3, existe un único homomorfismo de grupos

$$\pi : B_n \longrightarrow \mathfrak{S}_n$$

tal que $s_i = \pi(\sigma_i)$, $\forall i = 1, 2, \ldots n-1$ y que además es sobreyectivo. A este morfismo lo llamaremos *proyección canónica.*

Lema 1.4. *El grupo B_n con $n \geq 3$ es un grupo no abeliano*

Demostración. El grupo $\mathfrak{S}_n$ con $n \geq 3$ es no abeliano[2] puesto que $s_1 s_2 \neq s_2 s_1$. Como la proyección $B_n \longrightarrow \mathfrak{S}_n$ es sobreyectiva, se tiene que B_n es no abeliano para $n \geq 3$. ■

Observación 1.6. Es claro que el homomorfismo de grupos $\iota : B_n \longrightarrow B_{n+1}$ tal que $\iota(\sigma_i) = \sigma_i$, es inyectivo. A tal aplicación la denominaremos *inclusión natural.*

Esto nos beneficia en tanto que podemos ver B_n como subgrupo de B_{n+1} por ι. De esta forma podemos considerar la cadena de grupos $B_1 \subset B_2 \subset \ldots$ Además, componiendo ι con la proyección $\pi : B_{n+1} \longrightarrow \mathfrak{S}_{n+1}$ obtenemos la composición de $\pi : B_n \longrightarrow \mathfrak{S}_n$ con la inclusión canónica $\mathfrak{S}_n \longrightarrow \mathfrak{S}_{n+1}$ proporcionando conmutatividad al siguiente diagrama:

$$\begin{array}{ccc} B_n & \xrightarrow{\pi} & \mathfrak{S}_n \\ \downarrow{\scriptstyle \iota} & & \downarrow \\ B_{n+1} & \xrightarrow{\pi} & \mathfrak{S}_{n+1} \end{array}$$

[2]Teorema 1.6, página 7

Capítulo 2

Trenzas y Grupos de trenzas

El objeto principal de este capítulo es dar una introducción a la teoría de trenzas, definiendo los conceptos de trenza y grupo de trenzas, desarrollar sus estructuras, es decir, estudiar cómo son sus elementos y cómo se comportan entre ellos; y por último dar una serie de propiedades algebraicas y resultados que nos facilitará posteriormente en la implementación de las trenzas en el ambiente de la criptografía.

Para el desarrollo del capítulo, veremos como desde el punto de vista geométrico podremos construir el grupo de trenzas, al que posteriormente podremos darle un sentido algebraico estableciendo una relación de isomorfía con los grupos de trenzas de Artin, de esta forma podremos tratar las trenzas desde ambos sentidos.

El capítulo desarrollará toda la teoría relacionada a las trenzas y grupos de trenzas teniendo en cuenta, como referencias bibliográficas, los artículos [1], [2] y [7].

1. Trenzas y diagrama de trenzas

En esta sección vamos a construir e interpretar geométricamente lo que se conoce como *trenzas*. Además de presentar los diagramas que lo representan, una forma sencilla y visual para manejar estos objetos matemáticos.

Definiremos una operación producto entre dichos diagramas que será esencial para la construcción de los grupos de trenzas.

De ahora en adelante, denotaremos por I el intervalo unidad cerrado $[0,1]$ en el conjunto de los números reales $\mathbb{R}$. De este modo, entendemos por un *intervalo topológico* a un espacio topológico homeomorfo a I.

Definición 2.1. *Una* trenza *de $n \geq 1$ hebras es un conjunto $b \subset \mathbb{R}^2 \times I$ formada por n intervalos topológicos disjuntos llamados hebras de b tal que la proyección $\mathbb{R}^2 \times I \to I$ envía a cada hebra a I homeomorficamente y*

$$b \cap \Big(\mathbb{R}^2 \times \{0\}\Big) = \{(1,0,0),(2,0,0),\ldots,(n,0,0)\}$$

$$b \cap \Big(\mathbb{R}^2 \times \{1\}\Big) = \{(1,0,1),(2,0,1),\ldots,(n,0,1)\}$$

Es claro que caada hebra de b corta cada plano $\mathbb{R}^2 \times \{t\}$ con $t \in I$ en exactamente un punto y conecta un punto $(i,0,0)$ a un punto $(\sigma(i),0,1)$, donde $i, \sigma(i) \in \{1,2,\ldots,n\}$.

La secuencia $(\sigma(1), \sigma(2), \ldots, \sigma(n))$ es una permutación del conjunto $\{1, 2, \ldots, n\}$ llamada *permutación asociada.*

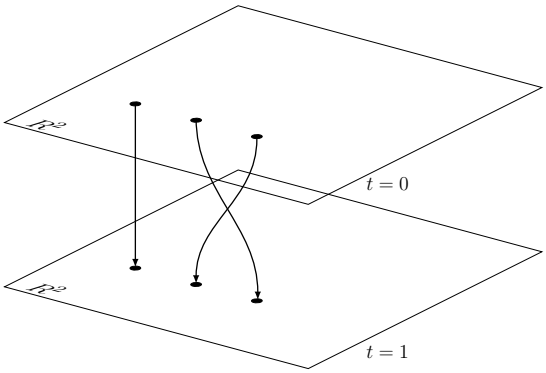


Fig 2.1 Ejemplo de trenza geométrica.

Dos trenzas b y b' con n hebras son *isótopas* si b puede ser deformada de forma continua a b' en la clase de trenzas. Es decir, b y b' son isótopas si hay una aplicación continua $F : b \times I \to \mathbb{R}^2 \times I$ tal que para cada $s \in I$, la aplicación $F_s : b \to \mathbb{R}^2 \times I$, que envía $x \in b$ a $F(x, s)$, es una inmersión cuya imagen es una trenza de n hebras, tal que $F_0 = id_b : b \to b$, y $F_1(b) = b'$.

Es obvio que la relación de isotopía es una relación de equivalencia sobre las clases de trenzas geométricas de n hebras. La correspondiente clase de equivalencia se denomina *trenzas de n hebras.*

Dados dos trenzas de n hebras $b_1, b_2 \subset \mathbb{R}^2 \times I$, definimos su producto, y lo damos mediante yuxtaposición, $b_1 b_2$, al conjunto de puntos $(x, y, t) \in \mathbb{R}^2 \times I$ tal que $(x, y, 2t) \in b_1$ si $0 \leq t \leq 1/2$ y $(x, y, 2t - 1) \in b_2$ si $1/2 \leq t \leq 1$.

Es evidente que tanto b_1 como b_2 son trenzas de n hebras. Además, si b_1 y b_2 son isótopas a b_1' y b_2', respectivamente; entonces $b_1 b_2$ es isótopa a $b_1' b_2'$. Esta operación producto es asociativa y tiene elemento neutro que no es más que la *trenza trivial* 1_n que deja inalterable todos los nodos, es decir, no permutan.

$$1_n = \{1, 2, \ldots, n\} \times \{0\} \times I$$

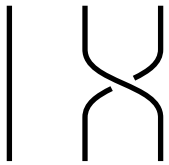
Aunque se desarrollará más en adelante, se puede intuir que el conjunto de las trenzas de n cadenas junto a dicha operación producto forman un grupo, en efecto, el *grupo de trenzas.*

1.1. Diagrama de trenzas

Cualquier trenza geométrica es isotópa a una trenza geométrica $b \subset \mathbb{R}^2 \times I$ tal que b es una subvariedad unidimensional plana de $\mathbb{R}^2 \times I$ ortogonal tanto a $\mathbb{R}^2 \times 0$ como a $\mathbb{R}^2 \times 1$. Por tanto, trabajando con trenzas, es a veces conveniente restringirse a dicha representación plana.

Así, la anterior representación[1] quedaría de la forma:

[1]Representación de trenza geométrica, página 13



Llamamos *hebra* a las conexiones de un punto a otro, entonces

Definición 2.2. *Un* diagrama de trenzas *sobre n hebras es un conjunto $D \subset \mathbb{R} \times I$ separados como unión de n intervalos topológicos llamados hebras de D tal que siguen las siguientes tres condiciones:*

1. *La proyeccion $\mathbb{R} \times I \longrightarrow I$ envía cada hebra homeomorficamente a I.*
2. *Los puntos de $\{1, 2, \ldots n\} \times \{0, 1\}$ son los puntos finales para las hebra.*
3. *Los puntos de $\mathbb{R} \times I$ pertenecen a, como mucho, a dos hebras. Estas hebras se pueden cruzar, es decir, que dos puntos de ellas interseccionan, se debe representar como una hebra por encima y la otra por debajo.*

Nótese que tres hebras de un diagrama de trenzas D nunca coinciden en un mismo punto. Sin embargo, cuando coinciden dos hebras en un punto se dice que es un **cruce**, un **cruzamiento** o un **punto doble**.

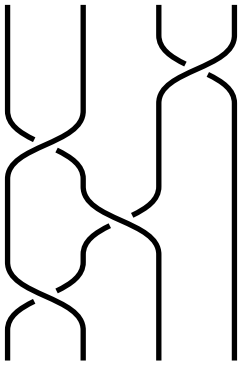


Fig 2.2 Ejemplo de diagrama de cuatro hebras

Vamos ahora a describir la relación que existe entre las trenzas y los diagramas de trenzas.

Cada diagrama de trenzas D presenta una clase de isotopía de trenzas geométricas seguidas. Dos diagramas de trenzas D y D' de n hebras se dicen que son isotópas si hay una función continua $F : D \times I \to \mathbb{R} \times I$ tal que para cada $s \in I$ el conjunto $D_s = F(D \times s) \subset \mathbb{R} \times I$ es un diagrama de n hebras, donde $D_0 = D$ y $D_1 = D'$.

Se entiende que F envía los cruzamientos de D a los cruzamientos de D_s para todo $s \in I$ preservando sub/supercruzado. A la familia de diagramas de trenzas $\{D_s\}_{s \in I}$ es llamado una isotopía de D a D'.

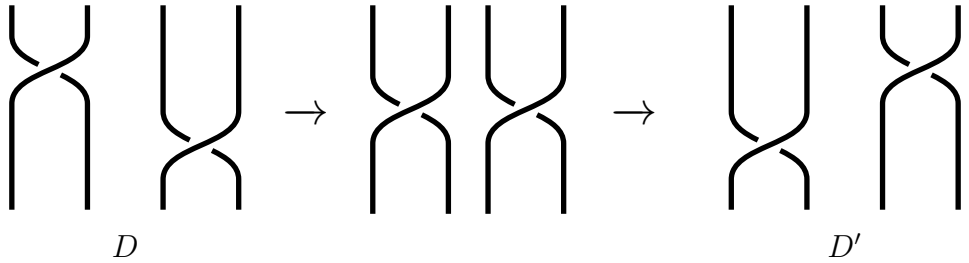


Fig 2.3 Ejemplo de isotopía de diagramas de trenzas

Dado dos diagramas de trenzas D_1, D_2 de n hebras, su producto D_1D_2 se obtiene conectando D_1 por encima de D_2 y encajando el diagrama resultante en $\mathbb{R} \times I$.

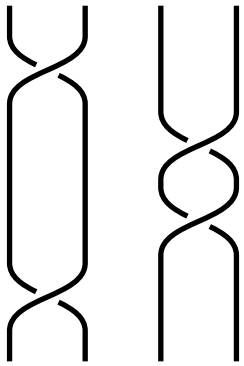


Fig 2.4 Producto de D con D'

Movimientos de Reidemeister sobre diagramas de trenzas

Las transformaciones de diagramas de trenzas Ω_2, Ω_3 siguientes se denominan *movimientos de Reidemeister*.

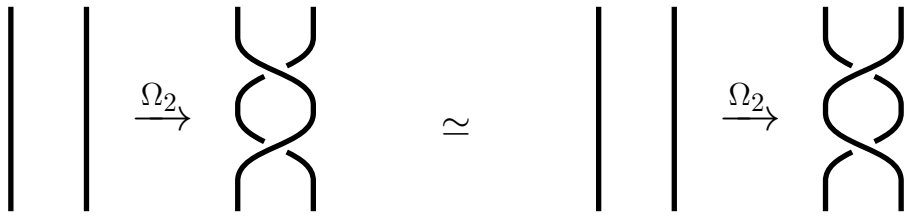


Fig 2.5a El movimiento de Reidemeister Ω_2

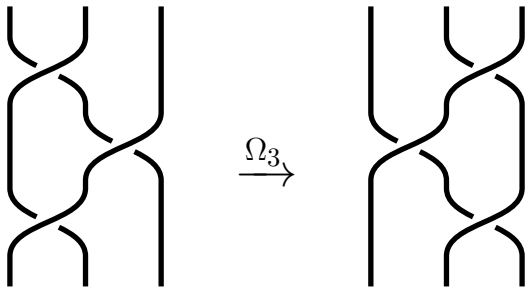


Fig 2.5b El movimiento de Reidemeister Ω_3

El origen de estos movimientos se encuentra en la teoría de nudos y diagramas de nudos.

Definición 2.3 (R-equivalencia)**.** *Dados dos diagramas D y D', son R-equivalentes si D puede transformarse en D' mediante una secuencia finita de isotopías y movimientos de Reidemeister $\Omega_2^{\pm 1}$, $\Omega_3^{\pm 1}$.*

El movimiento Ω_2 involucra dos hebras de una trenza y crea dos cruces adicionales mientras que el movimiento Ω_3 involucra tres hebras de una trenza y preserva el número de cruces. Además, estas transformaciones de diagramas de trenzas preserva las correspondientes isotopías de trenzas.

Teorema 2.1. *Dos diagramas de trenzas presentan una isotopía de trenzas geométricas si y sólo si, sus diagramas son R-equivalentes.*

La demostración consta de cuatro fases.

Demostración.

Paso 1. Introduciremos algunas notaciones que usaremos en el siguiente paso.

Consideremos una trenza geométrica $b \subset \mathbb{R}^2 \times I$ de n hebras. Para $i = 1, 2, \dots n$ denotamos la i-ésima hebra de b, esto es, la hebra adyacente al punto $(i, 0, 0)$ por b_i. Cada plano $\mathbb{R}^2 \times \{t\}$ con $t \in I$ interseca la hebra b_i en un único punto denotado por $b_i(t)$. En particular, se tiene que $b_i(0) = (i, 0, 0)$.

Sea ρ la métrica Euclidea sobre $\mathbb{R}^3$. Dado un número real $\epsilon > 0$, el ϵ-entorno cilíndrico (cylinder ϵ-neighborhood) de b_i contiene todos los puntos $(x, t) \in \mathbb{R}^2 \times I$ tal que $\rho((x, t), b_i(t)) < \epsilon$. Este entorno interseca cada plano $\mathbb{R}^2 \times \{t\} \subset \mathbb{R}^2 \times I$ sobre un disco abierto de radio ϵ centrado en $b_i(t)$.

Para distintos $i, j \in \{1, \dots n\}$, la función $t \longrightarrow \rho(b_i(t), b_j(t))$ es una función continua en I con valores positivos. Como I es compacto, se tiene que esta función tiene un mínimo. Sea

$$|b| = \frac{1}{2} \min_{1 \le i < j \le n} \min_{t \in I} \rho(b_i(t), b_j(t)) > 0$$

Es claro que los entornos cilindricos de radio $|b|$ de las hebras de b son disjuntas dos a dos. Para cualquier par de trenzas geométricas b, b' de n hebras y para cualquier $i = 1, \dots n$, la función $t \longrightarrow \rho(b_i(t), b'_i(t))$ es una función continua en I con valores no negativos. Como I es compacto, esta función tiene un valor máximo. Sea

$$\tilde{\rho}(b, b') = \max_{1 \le i < j \le n} \max_{t \in I} \rho(b_i(t), b'_i(t)) \ge 0$$

Dado b, b' y b'' trenzas geométricas de n hebras, la función $\tilde{\rho}$ satisface los axiomas de una métrica

$$\begin{array}{ll} (1.) & \tilde{\rho}(b, b') = \tilde{\rho}(b', b) \\ (2.) & \tilde{\rho}(b, b') = 0 \iff b = b' \\ (3.) & \tilde{\rho}(b, b'') \le \tilde{\rho}(b, b') + \tilde{\rho}(b', b'') \end{array}$$

Nótese también que

$$|b| \le |b'| + \tilde{\rho}(b, b')$$

De esta forma, para cualquier $t \in I$ y unos ciertos $i, j = 1, \dots n$ distintos, se tiene que

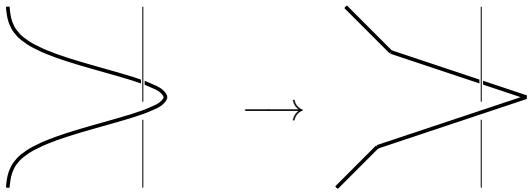
$$\begin{array}{ll} |b| & = \frac{1}{2} \rho\left(b_i(t), b_j(t)\right) \\ & \le \frac{1}{2} \left(\rho\left(b_i(t), b'_i(t)\right) + \rho\left(b'_i(t), b'_j(t)\right) + \rho\left(b'_j(t), b_j(t)\right) \right. \\ & \le \frac{1}{2} \left(\rho\left(b(t), b'(t)\right) + 2|b'| + \rho\left(b'(t), b(t)\right) \right. \\ & = |b'| + \rho\left(b(t), b'(t)\right) \end{array}$$

Paso 2. Demos previamente un concepto fundamental para el desarrollo de la demostración, el de trenza poligonal.

Definición 2.4 (Trenza poligonal)**.** *Se dice que una trenza es* poligonal *si todas sus hebras están formadas por segmentos consecutivos.*

Cualquier trenza geométrica se puede aproximar a una trenza poligonal de la siguiente forma. Tomado un entero $N \leq 2$ y un índice $i = 1, \dots N$, consideramos el segmento en $\mathbb{R}^2 \times I$ con puntos finales $b_i(\frac{k-1}{N})$ y $b_i(\frac{k}{N})$. La unión de esos N segmentos es una linea fragmentada, b_i^N, cuyos puntos finales son $b_i^N(0) = b_i(0) = (i, 0, 0)$ y $b_i^N(1) = b_i(1)$.

Para un N suficientemente grande, esta linea fragmentada está en el $|b|$-entorno cilíndrico de b_i. Por tanto, para un N suficientemente grande, las lineas fragmentadas $b_1^N, b_2^N, \dots b_n^N$ son disjuntas y forman una trenza poligonal, b^N que se aproxima a b. Además, para cualquier número $\epsilon > 0$ y un N suficientemente grande, tenemos que $\tilde{\rho}(b, b^N) < \epsilon$. Por ejemplo, la siguiente figura muestra una aproximación poligonal de una trenza.



Vamos a reformular la noción de isotopía de trenzas desde el punto de vista poligonal. Para ello, introduciremos los llamados Δ-movimientos sobre trenzas poligonales.

Sea A,B, y C tres puntos en $\mathbb{R}^2 \times I$ tal que la tercera coordenada de A es estrictamente más pequeña que la tercera coordenada de B y a su vez, esta más pequeña que la tercera coordenada de C. El movimiento $\Delta(ABC)$ se aplica a una trenza poligonal $b \subset \mathbb{R}^2 \times I$ haciendo que el segmento AC se transforme y coincide con el triangulo ABC (cuando hablamos del triángulo ABC nos referimos al 2-simplex con vertices ABC.

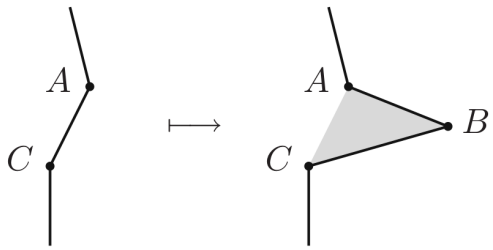


Fig 2.6 Ejemplo de Δ-movimiento

Bajo esta suposición, el movimiento $\Delta(ABC)$ sobre b remplaza $AC \subset b$ por $AB \cup BC$, manteniendo el resto de b intacto. El movimiento inverso $(\Delta(ABC))^{-1}$ realiza el cambio inverso, es decir, transforma $AB \cup BC$ por AC.

Estos movimientos $\Delta(ABC)$ y $(\Delta(ABC))^{-1}$ son llamados **Δ-movimientos**. Es fácil ver que estas relaciones de trenzas poligonales mediante Δ-movimientos se tratan de isotopías. Vamos a establecer esta afirmación.

Corolario 2.1. *Si las trenzas poligonales b, b' son isótopas, entonces b puede transformarse en b' mediante una secuencia finita de Δ-movimientos.*

Definición 2.5 (Trenza poligonal genérica)**.** *Una trenza poligonal es genérica si su proyección a* $\mathbb{R} \times I = \mathbb{R} \times \{0\} \times I$ *a lo largo de la segunda coordenada tiene solo un único cruce transversal.*

Paso 3. Ligeramente deformando los vértices de una trenza poligonal y manteniendo la curvatura, podemos aproximar esta trenza a una trenza poligonal genérica. Más aún, si b, b' son trenzas poligonales genéricas que se relacionan por una secuencia de Δ-movimientos, entonces la ligera deformación de los vértices de las trenzas poligonales intermedias, podemos asegurar que esas trenzas poligonales también son genéricas.

Corolario 2.2. *Si dos trenzas poligonales genéricas* b*,* b' *son isótopas, entonces* b *puede transformarse en* b' *mediante una secuencia finita de* Δ*-movimientos tal que todas las trenzas poligonales intermedias son genéricas.*

Del mismo modo que existe de las trenzas a sus diagramas de trenzas, presentamos los **diagramas de las trenzas poligonales genéricas**. Estos diagramas, son diagramas de trenzas, cuyas hebras están formadas por segmentos consecutivos con la diferencia con respecto a las anteriores, que podemos asumir siempre que un vértice de estos segmentos no coinciden con los cruces del diagrama.

— Sin embargo, por comodidad a la hora de representar gráficamente las siguientes demostraciones o transformaciones, se utilizará la representación habitual que hemos visto hasta entonces y no la definida para las trenzas poligonales genéricas. En el artículo *Braid Groups* [1] se pueden ver tales diagramas.

Corolario 2.3. *Los diagramas de dos trenzas poligonales genéricas que están relacionadas por un* Δ*-movimiento son* R*-equivalentes.*

Demostración. Consideremos un Δ-movimiento $\Delta(ABC)$ sobre una trenza poligonal genérica b que lo envía a otra trenza poligonal genérica b'.

Sea A', C' dos puntos del segmento AB y BC, respectivamente. Sea D un punto en el segmento AC tal que la tercera coordenada de D está estrictamente entre las terceras coordenadas de A' y C'.

Aplicando a b los movimientos $\Delta(AA'D)$ y $\Delta(DC'C)$, podemos transformar el segmento AC al segmento fraccionado $AA'DC'C$. Si ahora, aplicamos los movimientos $\Delta(A'DC')^{-1}$ y $\Delta(A'BC')$, obtenemos b'.

Esto muestra que el movimiento $\Delta(ABC)$ puede ser realizado por una secuencias de cuatro Δ-movimientos sobre triángulos pequeños. Esta expansión del movimiento $\Delta(ABC)$ puede iterarse.

De esta forma, subdividiendo el triangulo ABC a triangulos más pequeños y expandiendo el Δ-movimiento como composición de Δ-movimientos más pequeños, podemos reducirnos al caso donde la proyección de ABC a $\mathbb{R} \times I$ coincide con el resto del diagrama de b ya sea a lo largo del segmento, o a lo largo de dos segmentos con un punto donde cruza.

Caso 1. Si ambos puntos finales del segmento están en $AB \cup BC$, entonces el diagrama es transformado bajo $\Delta(ABC)$ por Ω_2. Si uno de los puntos finales del segmento está en

AC y el otro en $AB \cup BC$ entonces el diagrama es transformado por una isotopía.

Caso 2. Si la proyección de ABC a $\mathbb{R} \times I$ coincide con el diagrama a lo largo de los segmentos con un punto donde cruza, entonces podemos distinguir dos subcasos. Subdivir si fuera necesario el triangulo ABC a triangulos más pequeños y expandiendo nuestro Δ-movimiento como una composición de Δ-movimientos a lo largo de los pequeños triangulos, podemos reducirnos al caso en el cual el movimiento preserva la parte del diagrama.

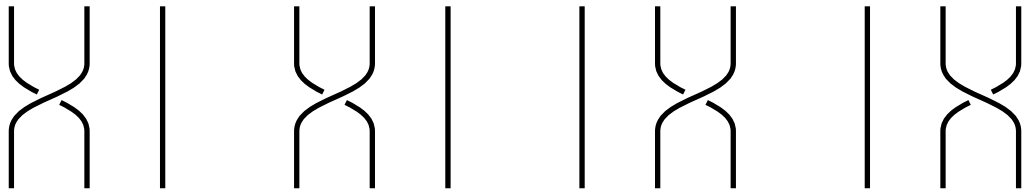
Si la proyección de ABC a $\mathbb{R} \times I$ coincide con el resto del diagrama a lo largo de dos segmentos que tienen un cruce, entonces podemos distinguir de manera similar varias subcasos.

Deberíamos entonces, subdividir si es necesario el triángulo ABC en triángulos más pequeños y expandiendo nuestro Δ-movimiento como una composición de Δ-movimientos sobre esos triángulos más pequeños.

De esta forma, podemos reducirnos al caso en el que el movimiento preserva la parte del diagrama que se encuentra fuera de un disco pequeño en R × I y realizar los cambios dentro de este disco mediante las siguientes seis fórmulas:

$$\begin{array}{lll} \mathrm{d}_1^+ d_2^+ d_1^+ \longleftrightarrow d_2^+ d_1^+ d_2^+ & \mathrm{d}_1^+ d_2^+ d_1^- \longleftrightarrow d_2^- d_1^+ d_2^+ & \mathrm{d}_1^- d_2^- d_1^+ \longleftrightarrow d_2^+ d_1^- d_2^- \\ \mathrm{d}_1^- d_2^- d_1^- \longleftrightarrow d_2^- d_1^- d_2^- & \mathrm{d}_1^+ d_2^- d_1^- \longleftrightarrow d_2^- d_1^- d_2^+ & \mathrm{d}_1^- d_2^+ d_1^+ \longleftrightarrow d_2^+ d_1^+ d_2^- \end{array}$$

Aquí, $d_1^{\pm}$ y $d_2^{\pm}$ son diagramas de trenzas de tres hebras.



d_1^+ d_1^- d_2^+ d_2^-

Fig 2.7 Los diagramas d_1^+, d_1^-, d_2^+, d_2^-

Falta probar que para cada uno de ellos, los diagramas respectivos sean R-equivalentes. Vemos que la transformación $d_1^+ d_2^+ d_1^+ \longrightarrow d_2^+ d_1^+ d_2^+$ es justo Ω_3. Para las otras cinco transformaciones, la R-equivalencia se establece siguiendo la secuencia de movimientos:

$$\begin{aligned} \omega &= \mathrm{d}_1^+ d_2^+ d_1^- \xrightarrow{\Omega_2} d_2^- d_2^+ d_1^+ d_2^+ d_1^- \xrightarrow{\Omega_3^{-1}} d_2^- d_1^+ d_2^+ d_1^+ d_1^- \xrightarrow{\Omega_2^{-1}} d_2^- d_1^+ d_2^+ \\ \gamma &= \mathrm{d}_1^- d_2^- d_1^+ \xrightarrow{\Omega_2} d_1^- d_2^- d_1^+ d_2^+ d_2^- \xrightarrow{\omega^{-1}} d_1^- d_1^+ d_2^+ d_1^- d_2^- \xrightarrow{\Omega_2^{-1}} d_2^- d_1^- d_2^- \\ \mu &= \mathrm{d}_1^- d_2^- d_1^- \xrightarrow{\Omega_2} d_2^- d_2^+ d_1^- d_2^- d_1^- \xrightarrow{\gamma^{-1}} d_2^- d_1^- d_2^- d_1^+ d_1^- \xrightarrow{\Omega_2^{-1}} d_2^- d_1^- d_2^+ \end{aligned}$$

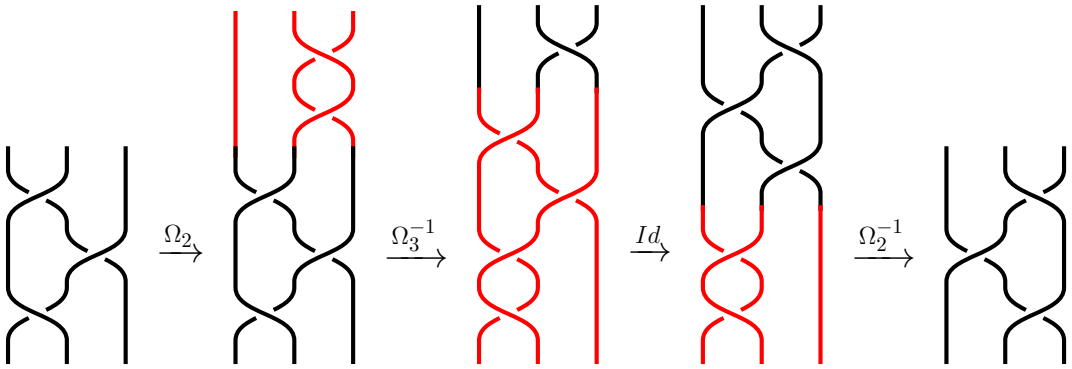


Fig 2.8a Construcción de ω

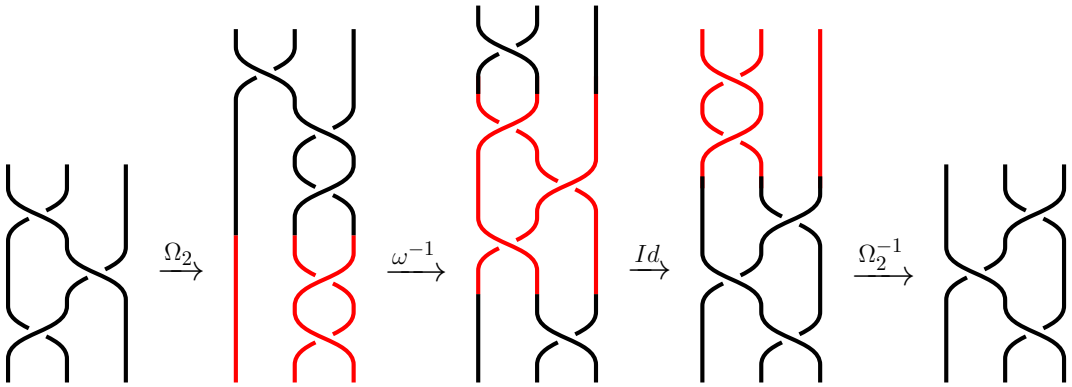


Fig 2.8b Construcción de γ

■

Paso 4. Ahora sí, podemos completar la demostración del *Teorema* 2.1.

Es obvio que la R-equivalencia de diagramas de trenzas presenta isotopías de trenzas. Consideremos los diagramas D_1 y D_2, se tiene que para $i = 1, 2$, enderezando D_i cerca de los cruces y aproximando el resto de D_i mediante líneas discontinuas como en el paso 2, obtenemos un diagrama D_i de una trenza poligonal genérica b^i. Si la aproximación es suficientemente cerrada, entonces D_i' es isotópica a D_i y por tanto b^1 y b^2 son isotópicas.

Por el ***Corolario 1.3*** implica que b^1 puede transformarse en b^2 por una secuencia finita de Δ-movimientos en la clase de trenzas poligonales genéricas.

Por el ***Corolario 1.4*** se tiene que los diagramas D_1' y D_2' son R-equivalentes. Por tanto los diagramas D_1 y D_2 son también R-equivalentes.

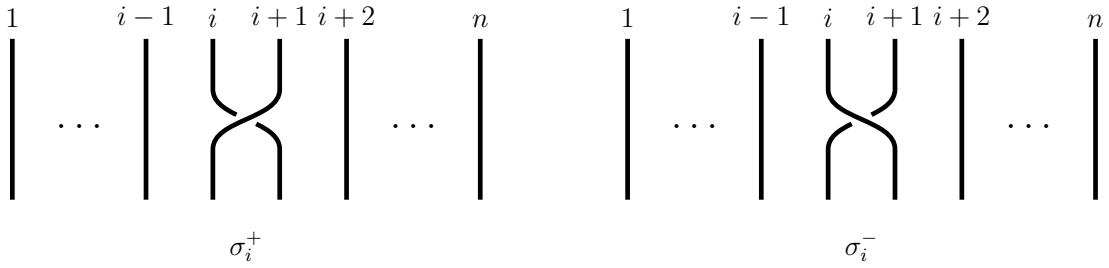
□

1.2. Grupos de trenzas

Tras ver, en un cierto término geométrico, qué entendemos o como podemos contemplar las trenzas; consideramos $\mathcal{B}_n$ el conjunto de trenzas de n hebras con el producto ya definido. El siguiente lema muestra que $\mathcal{B}_n$ es un grupo.

Lema 2.1. *Para cada $\beta \in \mathcal{B}_n$ existe al menos un inverso $\beta^{-1} \in \mathcal{B}_n$*

Demostración. Sea $i = 1, 2, \ldots, n-1$. Definimos dos trenzas elementales, σ_i^+ y σ_i^- representadas de la siguiente forma.



σ_i^+ σ_i^-

Fig 2.9 Trenzas elementales σ_i^+ y σ_i^-

Veamos que las trenzas $\sigma_i^+, \ldots \sigma_{n-1}^+, \sigma_1^-, \ldots, \sigma_{n-1}^- \in \mathcal{B}_n$ generan a $\mathcal{B}_n$ como un monoide. Para ver esto, consideramos una trenza β con n hebras respresentada por un diagrama D. Mediante una deformación suave de $D \subset \mathbb{R} \times I$ en un entorno de sus puntos de cruces, podemos distinguir y aislar dichos cruces del diagrama como sigue:

Sea $k \in \mathbb{N}$ tal que $0 = t_0 < t_1, \ldots t_k = 1$. La intersección de D con cada $\mathbb{R} \times [t_j, t_{j+1}]$ con $j = 1, \ldots k$ define un trozo de la trenza β con exactamente un cruce, esto es, tal subdiagrama de la trenza no es más que el diagrama de las trenzas elementales σ_i^+ o σ_i^- para algun $i = 1, \ldots n-1$.

Así, podemos ver D como una descomposición de k diagramas de trenzas elementales

$$\beta = \beta(D) = \sigma_{i_1}^{\epsilon_1} \sigma_{i_2}^{\epsilon_2} \ldots \sigma_{i_k}^{\epsilon_k} \tag{2.1}$$

con $\epsilon_j \in \{+, -\}$.

Claramente, como para cualquier $i = 1, \ldots n-1$, $\sigma_i^+ \sigma_i^- = \sigma_i^- \sigma_i^+ = 1$, se tiene que

$$\beta^{-1} = \sigma_{i_k}^{-\epsilon_k} \sigma_{i_{k-1}}^{-\epsilon_{k-1}} \ldots \sigma_{i_1}^{-\epsilon_1}$$

es un inverso de β en $\mathcal{B}_n$ ■

Como el lector habrá podido comprobar, llegados a este punto tenemos casi la equivalencia entre los grupos de trenzas desde una vista geométrica y los grupos de Artin. Efectivamente, queda probar que las relaciones que se cumplían en la definición de estos últimos grupos se dan también en los grupos de trenzas.

Lema 2.2. *Los elementos $\sigma_1^+, \ldots \sigma_{n-1}^+ \in \mathcal{B}_n$ satisface las siguientes relaciones de trenzas*

$$\begin{array}{ll} \sigma_i^+ \sigma_j^+ = \sigma_j^+ \sigma_i^+ & i, j = 1, 2, \ldots n-1 \text{ con } |i-j| \geq 2 \\ \sigma_i^+ \sigma_{i+1}^+ \sigma_i^+ = \sigma_{i+1}^+ \sigma_i^+ \sigma_{i+1}^+ & i = 1, 2, \ldots n-2 \end{array} \tag{2.2}$$

Demostración. La primera relación es trivial pues es simplemente una isotopía de diagramas. Mientras que la segunda relación no es más que aplicar el movimiento de Reidemeister Ω_3 ■

Así, ya estamos en las condiciones necesarias para ver la relación que existe los grupos de trenzas $\mathcal{B}_n$ y los grupos de Artin B_n. Efectivamente, cuando hablemos de grupos de trenzas, lo podemos ver tanto como trenzas en términos geométricos como hemos estado viendo hasta ahora, o bien como un grupo de Artin, es decir, en un aspecto más algebráico.

Teorema 2.2. *Para cada $\epsilon = \pm$, existe un único isomorfismo $\varphi_\epsilon : B_n \longrightarrow \mathcal{B}_n$ tal que $\varphi_\epsilon(\sigma_i) = \sigma_i^\epsilon$ para todo $i = 1, 2, \ldots n-1$.*

Demostración. Sin perdida de generalidad, consideremos $\epsilon = +$. La existencia y unicidad de φ_+ siguen directamente del Lema 1.4 y el Lema 1.7. Mientras que por el Lema 1.6 se tiene que como $\sigma_1^+, \ldots \sigma_{n-1}^+$ generan $\mathcal{B}_n$ como un grupo, entonces φ_+ es sobreyectiva.

Vamos a construir una aplicación de conjuntos $\psi : \mathcal{B}_n \longrightarrow B_n$ tal que $\psi \circ \varphi_+ = Id$. Esto hará que φ_+ sea inyectiva.

Como en la demostración del Lema 1.6, podemos descomponer cualquier trenza $\beta \in \mathcal{B}_n$ con diagrama de trenzas D como productos de trenzas elementales. Por tanto, se tiene que

$$\psi(D) = \psi\left(\sigma_{i_1}^{\epsilon_1}\sigma_{i_2}^{\epsilon_2}\ldots\sigma_{i_k}^{\epsilon_k}\right) = \sigma_{i_1}^{\epsilon_1}\sigma_{i_2}^{\epsilon_2}\ldots\sigma_{i_k}^{\epsilon_k} \in B_n$$

donde podemos entender $\sigma_i^+ = \sigma_i$ y $\sigma_i^- = \sigma_i^{-1}$.

Así, podemos ver que $\psi(D)$ depende solo de β. Por el Teorema 1.10 es necesario ver que $\psi(D)$ es invariante bajo isotopías de D y movimientos de Reidemeister.

Las isotopías de D que invarian el orden de los cruces de D mantienen la descomposición en trenzas elementales y por tanto preservan $\psi(D)$. Por otro lado, una isotopía que cambia el orden de dos cruces de D reemplazando el termino $\sigma_i^{\epsilon_i}\sigma_j^{\epsilon_j}$ en la descomposición por $\sigma_j^{\epsilon_j}\sigma_i^{\epsilon_i}$ para cierto $i, j \in \{1, 2, \ldots n-1\}$ con $|i-j| \geq 2$, también se mantiene bajo ψ al ser enviado a B_n y cumplirse la primera relación de trenzas en la Definición 1.14 de grupos de Artin.

El movimiento Ω_2 (resp. Ω_2^{-1}) en D añade (resp. elimina) en la descomposición una secuencia $\sigma_i^+\sigma_i^-$ o $\sigma_i^-\sigma_i^+$. Claramente esto preserva $\psi(D)$.

El movimiento Ω_3 sobre D remplaza una secuencia $\sigma_i^+\sigma_{i+1}^+\sigma_i^+$ en la descomposición por $\sigma_{i+1}^+\sigma_i^+\sigma_{i+1}^+$, que, bajo ψ, es enviada al mismo elemento en B_n por la segunda relación de trenzas en la Definición 1.14 de grupos de Artin. De forma análoga para el movimiento Ω_3^{-1}.

Esto hace que la aplicación ψ esté bien definida. Por construcción, $\psi \circ \varphi_+ = Id$. Así, ψ es injectiva y por tanto biyectiva.

■

Desde ahora en adelante, por convenio, identificaremos al grupo de trenzas $\mathcal{B}_n$ y al grupo de Artin B_n mediante φ_+. A los elementos de B_n los llamaremos trenzas de n hebras y escribiremos σ_i para la trenza σ_i^+. Así, con esta notación, $\sigma_i^- = (\sigma_i^+)^{-1} = \sigma_i^{-1}$.

La proyección al grupo simetrico $\pi : B_n \longrightarrow \mathfrak{S}_n$ puede describirse en términos geométricos. Para una trenza geométrica b de n hebras, la permutación $\pi(b) \in \mathfrak{S}_n$ envía cada $i \in \{1, 2, \ldots n\}$ al único $j \in \{1, 2, \ldots n\}$ tal que la hebra de b unida al $(i, 0, 0)$ tiene el segundo punto final en $(j, 0, 1)$.

Corolario 2.4. *La inclusión natural $\iota : B_n \longrightarrow B_{n+1}$ es inyectiva para todo n.*

Demostración. En términos geométricos, la aplicación $\iota : B_n \longrightarrow B_{n+1}$ añade, a una trenza geométrica b de n hebras, una hebra vertical más a la derecha de dicha trenza b. Denotamos a esta trenza de $n+1$ hebras por $\iota(b)$.

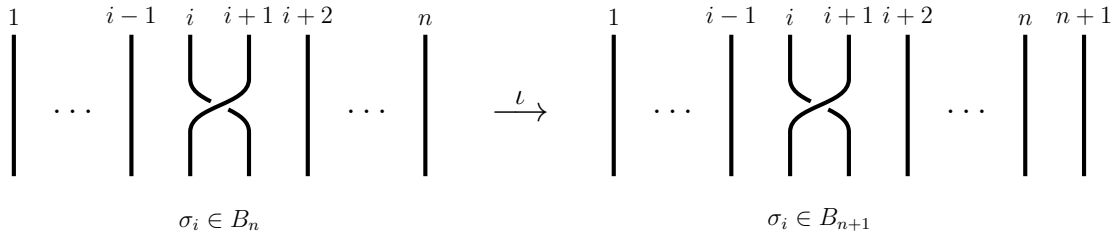


Fig 2.10 Representación del homomorfismo $\iota : B_n \longrightarrow B_{n+1}$ aplicado a σ_i

Si b_1, b_2 son dos trenzas geométricas de n hebras tal que $\iota(b_1)$ es isotópica a $\iota(b_2)$, entonces la restricción de la isotopía a un nivel menos, es decir, las n primeras hebras por la izquierda de $\iota(b_1)$ y $\iota(b_2)$ forman también una isotopía, y por tanto obtenemos una isotopía de b_1 a b_2. Por tanto, ι es inyectiva.

■

1.3. Trenzas puras. Grupos de trenzas puras

En este apartado, introduciremos las llamadas **trenzas puras** y las usaremos para establecer ciertas propiedades algebráicas importantes del grupo de trenzas.

Definición 2.6 (Grupo de trenzas puras)**.** *Llamamos el* grupo de trenzas puras *al kernel ó núcleo de la proyección natural* $\pi : B_n \longrightarrow \mathfrak{S}_n$ *y lo denotamos por* P_n*.*

$$P_n = Ker\,(\pi : B_n \longrightarrow \mathfrak{S}_n)$$

Los elementos de P_n se llaman **trenzas puras de** n **hebras**. Una trenza geométrica de n hebras representa un elemento de P_n si y sólo si para todo $i = 1, 2 \ldots n$ la hebra de esta trenza enlazada al punto $(i, 0, 0)$ tiene el segundo punto final en $(i, 0, 1)$. A estas trenzas geométricas se les dice que son **puras**.

Es claro que dado p, $q \in P_n$ se tiene que $pq \in P_n$. Además, el lector puede ver con claridad que el grupo de las trenzas puras es **subgrupo normal**.

Ejemplo 2.1. Demos una serie de ejemplos de trenzas puras para que el lector no se sienta desorientado:

$$\sigma_3\sigma_2\sigma_1\sigma_1\sigma_2^{-1}\sigma_3^{-1} \qquad \sigma_2\sigma_1\sigma_2^{-1}\sigma_3\sigma_3\sigma_2\sigma_3^{-1}\sigma_1^{-1}\sigma_2\sigma_1^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_3\sigma_2^{-1} \qquad \sigma_1^{-1}\sigma_1^{-1}\sigma_2\sigma_1\sigma_1\sigma_2^{-1}\sigma_1\sigma_1$$

Es necesario ver el papel importante que juega la trenza pura de n hebras $A_{i,j}$ con $1 \leq i < j \leq n$ representada en la siguiente figura

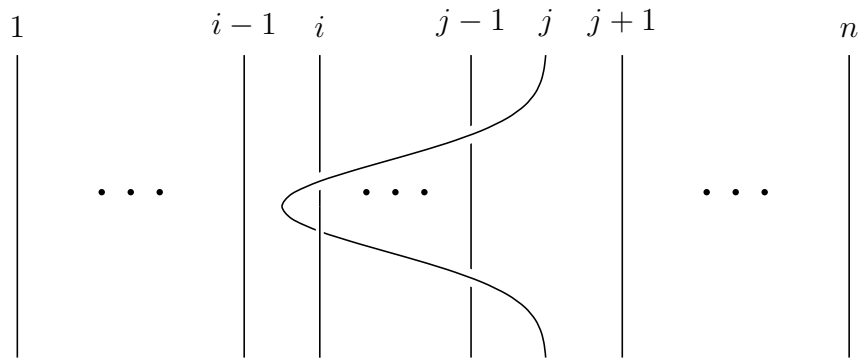


Fig 2.11 La n-trenza pura $A_{i,j}$, con $1 \leq i < j \leq n$

Esta trenza puede expresarse por los generadores $\sigma_1, \ldots \sigma_{n-1}$ de la siguiente forma

$$A_{i,j} = \sigma_{j-1}\sigma_{j-2} \ldots \sigma_{i+1}\sigma_i^2\sigma_{i+1}^{-1}\sigma_i^{-1} \ldots \sigma_{j-2}^{-1}\sigma_{j-1}^{-1}$$

Podemos observar en el Ejemplo 2.1, la trenza $\sigma_3\sigma_2\sigma_1\sigma_1\sigma_2^{-1}\sigma_3^{-1}$ es precisamente $A_{1,4}$.

Las trenzas $\{A_{i,j}\}_{i,j}$ son conjugables entre sí en B_n. Donde $\alpha_{i,j} \in A_{i,j}$ son de la forma

$$\alpha_{i,j} = \sigma_{j-1}\sigma_{j-2} \ldots \sigma_i$$

para cualquier $1 \leq i < j \leq n$.

Por otro lado, es sencillo ver que para cualquier $1 \leq i < j < k \leq n$,

$$\alpha_{j,k}A_{i,j}\alpha_{j,k}^{-1} = A_{i,k} \quad y \quad \alpha_{i,k}A_{i,j}\alpha_{i,k}^{-1} = A_{j,k}$$

Sin embargo, veríamos rápidamente que las trenzas $\{A_{i,j}\}_{i,j}$ no son conjugables entre sí en P_n. La conmutatividad que se presenta en el diagrama que vimos cuando definimos los grupos de Artin,

$$\begin{array}{ccccc} P_n & \hookrightarrow & B_n & \xrightarrow{\pi} & \mathfrak{S}_n \\ \downarrow{\scriptstyle \iota} & & \downarrow{\scriptstyle \iota} & & \downarrow \\ P_{n+1} & \hookrightarrow & B_{n+1} & \xrightarrow{\pi} & \mathfrak{S}_{n+1} \end{array}$$

implica que el homomorfismo de inclusión $\iota : B_n \longrightarrow B_{n+1}$ envía P_n a P_{n+1}. A este homomorfismo $P_n \longrightarrow P_{n+1}$ inducido por ι lo denotaremos de la misma forma. En términos geométricos, $\iota : P_n \longrightarrow P_{n+1}$ añade a una trenza pura geométrica b de n hebras una hebra vertical más a su derecha.

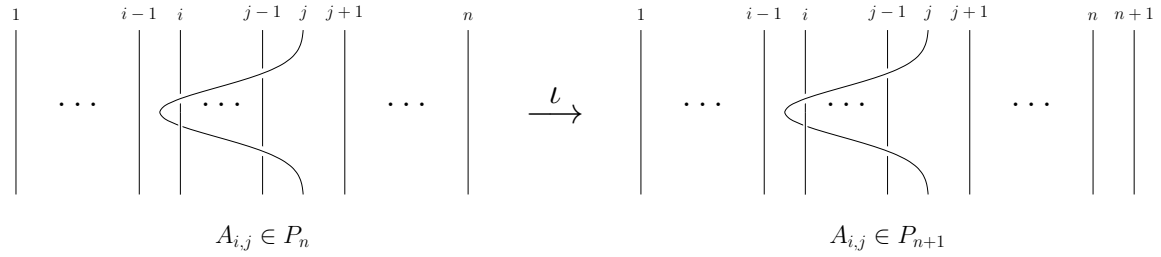


Fig 2.12 Representación del homomorfismo $\iota : P_n \longrightarrow P_{n+1}$ aplicado a la trenza pura $A_{i,j}$

Por el Corolario 1.5, $\iota : P_n \longrightarrow P_{n+1}$ es inyectiva. Es conveniente entonces ver P_n como un subgrupo de P_{n+1} mediante ι. De esta forma, obtenemos una cadena ascendente de grupos $P_1 \subset P_2 \subset P_3 \subset \ldots$. Es claro que $P_1 = \{1\}$ y P_2 es un grupo cíclico infinito generado por $A_{1,2} = \sigma_1^2$.

2. La palabra fundamental $\triangle$. El problema de la palabra

Como hemos podido comprobar, existen diferentes formas para representar una trenza, y también distintas trenzas que tienen, mediante una isotopía, a la misma representación de trenzas.

Dados dos trenzas equivalentes, es relativamente sencillo probar, mediante una serie de relaciones que transforman una en la otra, que efectivamente lo son. Sin embargo, probar para dos trenzas cualesquieras que no son equivalentes podría tomar algo más de tiempo. A este problema, se le llama **problema de isotopía** o, si estamos tratando las trenzas en términos de palabras, se le conoce como el **problema de la palabra**.

Hay numerosas pruebas para saber si dos trenzas no son la misma. Por ejemplo, considerando la *proyección natural* $\pi : B_n \longrightarrow \mathfrak{S}_n$, cuyo kernel es el grupo de trenza puras P_n. Se tiene que dado $b \in B_n$, $\pi(b)$ es la permutación que realiza el propio b.

Este homomorfismo puede ser una forma fácil de ver que dos trenzas no sean la misma. Sin embargo, el problema está que la condición de que las imágenes sean la misma no implica que las trenzas sean equivalente.

Y es así pues, dado φ una aplicación con $\varphi : B_n \longrightarrow X$ donde X es algún conjunto tal que si $\varphi(b) \neq \varphi(b')$ entonces $b \not\equiv b'$. A esta aplicación la llamamos **isotopía invariante**, o decimos que es una invariante. Decimos que es una **isotopía completamente invariante** si, además de ser una isotopía invariante, se tiene que si $\varphi(b) = \varphi(b')$ entonces $b \equiv b'$.

Así, encontrar una isotopía completamente invariante es dar una solución para el problema de isotopías de trenzas, y por tanto, en términos de palabras, dar una solución al problema de las palabras.

Comenzamos al inicio de este texto introduciendonos en las nociones de presentación de grupos y cerramos con el problema de la palabra.

Pues bien, tras haber sido capaz de relacionar mediante una isomorfía, teorema 2.2, los grupos de trenzas de Artin con los grupos de trenzas, y por tanto entender que se sus elementos se pueden ver desde ambos puntos de vista, le resultará fácil al lector ver los elementos de un grupo de Artin como elementos de un grupo simétrico, y por tanto, ver estos como palabras.

Dado una palabra $w \in B_n$, efectivamente, $w = \sigma_r \cdots \sigma_s$ con $1 \leq r \leq s \leq n$. Denotaremos como Π_s a la palabra $\sigma_1 \cdots \sigma_s$, y por tanto $\Pi_n = \sigma_1 \cdots \sigma_n \in B_n$.

Ahora sí, tras un largo recorrido, podemos introducir la esencia de este trabajo. Demos paso a la palabra fundamental y el problema de las palabras.

Definición 2.7 (Palabra fundamental de orden n)**.** *Consideramos B_n, llamamos palabra fundamental de orden n y lo denotamos como Δ_n a la siguiente palabra*

$$\Delta_n \equiv \Pi_{n-1}\Pi_{n-2}\cdots\Pi_1 = (\sigma_1\cdots\sigma_{n-1})(\sigma_1\cdots\sigma_{n-2})\cdots(\sigma_1\sigma_2)\sigma_1$$

Por convenio, si nos encontramos en B_n, a Δ_n lo denotaremos como Δ sin más.

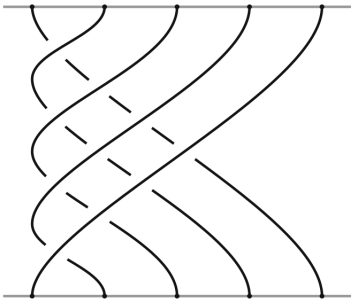


Fig 2.13 La trenza Δ_5

Lema 2.3. *Sea B_n, dados s,t tal que $1 < s \leq t \leq n-1$ se tiene que $\sigma_s\Pi_t = \Pi_t\sigma_{s-1}$*

Demostración. La demostración es realmente sencilla, nos bastará usar las relaciones de trenzas como sigue:

$$\begin{aligned}
\sigma_s\Pi_t &\equiv \sigma_s(\sigma_1\cdots\sigma_{t-1}) \\
&= \sigma_s(\sigma_1\cdots\sigma_{s-2})\sigma_{s-1}\sigma_s(\sigma_{s+1}\cdots\sigma_{t-1}) \\
(2.2a) &= (\sigma_1\cdots\sigma_{s-2})\sigma_s\sigma_{s-1}\sigma_s(\sigma_{s+1}\cdots\sigma_{t-1}) \\
(2.2b) &= (\sigma_1\cdots\sigma_{s-2})\sigma_{s-1}\sigma_s\sigma_{s-1}(\sigma_{s+1}\cdots\sigma_{t-1}) \\
(2.2a) &= (\sigma_1\cdots\sigma_{s-2})\sigma_{s-1}\sigma_s(\sigma_{s+1}\cdots\sigma_{t-1})\sigma_{s-1} \\
&= \Pi_t\sigma_{s-1}
\end{aligned}$$

■

2.1. La solución al problema de las palabras

En 1969, Garside describió en su articulo *The braid group and other groups* [2], la solución al problema de las palabras. Esta consistía en dar una expresión a una palabra con una determinada forma, no única, pero sí equivalente entre ellas. Se le conoce como la forma normal de Garside.

Para presentar la forma normal de Garside, empezaremos definiendo lo que se llama *trenza positiva*, esta es una trenza la cual puede ser escrita como productos de generadores de Artin con potencias positivas. Al conjunto de las trenzas positivas se le denota como B_n^+. Este conjunto no es más que un subconjunto de B_n y que por tanto tiene estructura de monoide bajo la operación producto o composición de trenzas. Un claro ejemplo de

trenza positiva es la *palabra fundamental*[2] $\Delta \in B_n$, que presenta un papel esencial para la forma normal de Garside.

Así es pues, la palabra fundamental, presenta importantes propiedades tales como que para cualquier generador σ_i, podemos escribir $\Delta_n = \sigma_i A = B\sigma_i$ donde A y B son trenzas positivas. Además, para cualquier σ_i, se tiene que $\Delta^{-1}\sigma_i\Delta = \sigma_{n-i}$.

Entonces, la solución para el problema de la palabra es el siguiente

Teorema 2.3 (Forma normal de Garside)**.** *Sea $w \in B_n$, w se puede expresar de manera única, salvo equivalencias, de la forma $w = \Delta^m A$, donde A es una trenza positiva.*

Esta solución, a pesar de establecerse como única, hay que tener cuidado puesto que dicha unicidad se refiere salvo equivalencia de trenzas.

Si consideramos $w = \Delta^m A$ la forma normal de Garside con A trenza positiva, es posible dar $w = \Delta^m A'$ otra forma normal de Garside con A' positiva, sin embargo $A \equiv A'$.

Claro está, estamos ante la primera solución que se estableció. En posteriores estudios se ha refinado dando una solución con una expresión realmente única en el propio sentido de la palabra, la *forma normal greedy*.

Sin embargo, antes de continuar, debemos profundizar aún más y establecer un cierto orden parcial entre elementos de B_n, así como algunas nociones y conceptos relevantes previos a la exposición de la forma normal greedy:

Dados $v, w \in B_n$, decimos que v es un *prefijo* de w y lo denotamos como $v \preceq w$ si y sólo si, existe $\beta \in B_n^+$ tal que $w = v\beta$. Es simple probar que, $w \in B_n^+$ si y sólo si $e \preceq w$ dónde e es elemento unidad tal y como definimos en la página 1, que corresponde a la trenza identidad, trenza que fija cada nodo; y que $v \preceq w$ si y sólo si $w^{-1} \preceq v^{-1}$.

Dado $p \in B_n$, decimos que p es una *trenza de permutación*, una *trenza simple* o un *factor canónico*, si satisface $e \preceq p \preceq \Delta$.

Su nombre viene del hecho de que existe una biyección entre el conjunto de trenzas de permutación en B_n y el conjunto de permutaciones $\mathfrak{S}_n$.

Geométricamente, una trenza de permutación es una trenza donde cada par de hebras se cruza a lo sumo una vez.

Dado una trenza de permutación p, definimos el *starting set* $S(p)$ y el *finishing set* $F(p)$ como los conjuntos:

$$S(p) = \{\ i \mid p = \sigma_i p' \text{ para algún } p' \in B_n^+\}$$

$$F(p) = \{\ i \mid p = p'\sigma_i \text{ para algún } p' \in B_n^+\}$$

El starting set es el conjunto de los indices de los generadores de B_n^+ tales que pueden empezar la expresión de p. El conjunto finishing set es similar con respecto los generadores que acaben la expresión de p.

[2]Sección 2, página 25

Ejemplo 2.2. Consideremos la trenza $p = \sigma_2\sigma_3\sigma_2\sigma_1$ en B_4. Se tiene que, mediante las relaciones de Artin:

$$\sigma_2\sigma_3\sigma_2\sigma_1 = \sigma_3\sigma_2\sigma_3\sigma_1 = \sigma_3\sigma_2\sigma_1\sigma_3$$

donde estas son todas las posibles expresiones de la forma $\sigma_i p'$ o $p'\sigma_i$ que se puedan dar tal que $p' \in B_4^+$. Así, $S(p) = \{2, 3\}$, mientras que $F(p) = \{1, 3\}$.

Otro ejemplo muy interesante es que para Δ se tiene que $S(\Delta) = F(\Delta) = \{1, \ldots, n-1\}$.

■

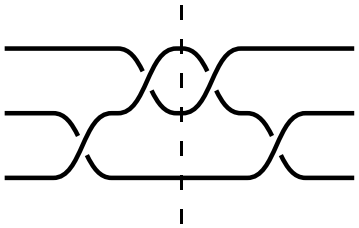
Dado una trenza $w \in B_n^+$, denominamos la factorización **cargada a la izquierda** de w como la descomposición de w en una secuencia de trenzas de permutación, esto es:

$$w = p_1 p_2 \cdots p_k$$

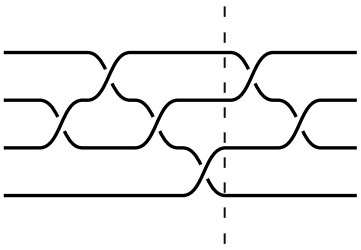
donde cada p_i es una trenza de permutación tal que $S(p_{i+1}) \subseteq F(p_i)$, es decir, que si añadimos cualquier generador de p_{i+1} a p_i, esta dejaría de ser una trenza de permutación. Veamos un par de ejemplos.

Ejemplo 2.3. Consideremos la trenza $a = \sigma_1\sigma_2\sigma_2\sigma_1$ en B_3. Se tiene que la trenza a esta en su factorización cargada a la izquierda pues $a = p_1p_2$, donde $p_1 = \sigma_1\sigma_2$ y $p_2 = \sigma_2\sigma_1$.

Se puede ver en su representación como para cada p_i sus hebras se cruzan a lo sumo una vez tal y como definimos antes:

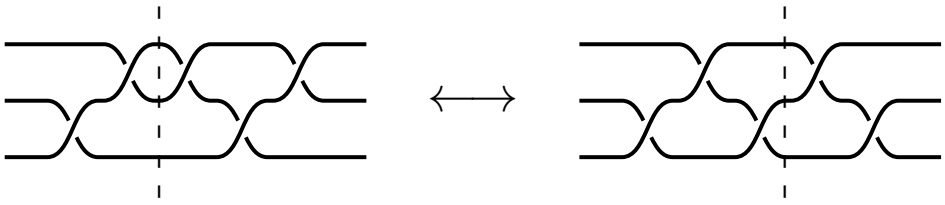


donde se ve claramente que $S(p_2) = \{1, 2\} = F(p_1)$. Si consideramos ahora la trenza $b = \sigma_2\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2$ en B_3, podemos ver en su representación:



donde $p_1 = \sigma_2\sigma_3\sigma_2\sigma_1$, $p_2 = \sigma_3\sigma_2$ y además $S(p_2) = \{3\} \subset \{1, 3\} = F(p_1)$. Por tanto, la trenza b está en su factorización cargada a la izquierda.

Por el contrario, si consideramos $c = \sigma_1\sigma_2\sigma_2\sigma_1\sigma_2$ en B_3, se tiene que no es una factorización cargada a la izquierda, puesto que si nos fijamos en su representación, el último cruce lo podemos adelantar hacia la izquierda, es decir, se tendría que $\sigma_1\sigma_2\sigma_2\sigma_1\sigma_2 = \sigma_1\sigma_2\sigma_1\sigma_2\sigma_1$



Y por lo tanto podríamos pasar una permutación de p_2 a p_1.

■

Teorema 2.4 (Forma normal greedy)**.** *Toda palabra $w \in B_n$ puede ser expresada de forma única como*

$$w = \Delta^r w_1 \cdots w_s,$$

donde $r \in \mathbb{Z}$ es maximal, w_i son trenzas de permutación y $w_1 w_2 \cdots w_s$ es una factorización cargada a la izquierda.

Esta expresión es lo que se conoce como la *forma normal greedy* de w. En algunos libros podemos encontrar la expresión $(r; w_1, \ldots w_s)$ para referirse a dicha forma de w. Al parámetro s se le llama *complejidad* o *longitud canónica* de w. Además, se define *inf* $w = r$, y *sup* $w = s + inf\ w$.

Dado $w \in B_n$ una trenza cualquiera, el hecho de encontrar la forma normal greedy de w puede parecer, a priori, altamente desconcertante. Por lo que prestaremos un método, con una serie de indicaciones, para su ejercicio.

1. Para cualquier generador con potencia negativa σ_i^{-1}, remplazar esta por $\Delta^{-1} B_i$ donde B_i es una trenza positiva.

2. Mover cada aparición del elemento Δ todo lo posible a la izquierda mediante la relación: $\sigma_i \Delta = \Delta \sigma_{n-i}$. De esta forma podremos asegurar la **forma normal de Garside**

   $$w = \Delta_m^r A$$

   donde A es una trenza positiva.

3. Escribir la trenza positiva A como una descomposición cargada a la izquierda de trenzas de permutación.

   Para ello, tomamos A y la separamos en trenzas de permutación, tal y como hemos visto en el ejemplo 2.3. Así tenemos $A = w_1 w_2 \cdots w_k$, donde cada w_i es una trenza de permutación. Se debe cumplir que $S(w_{i+1}) \subseteq F(w_i)$ para todo i. En caso contrario, debemos usar las relaciones del grupo de Artin para forzar que esto se cumpla.

Realmente es esencial dar una serie de ejemplos para que el lector se familiarice con esta práctica

Ejemplo 2.4. Sea $w = \sigma_1 \sigma_3^{-1} \sigma_2 \in B_4$. Lo primero que haremos será reemplazar σ_3^{-1} por $\Delta^{-1} B$ con $B \in B_4^+$, por lo que entonces, debemos determinar quien es B. Se tiene que,

como $\sigma_3^{-1} = \Delta^{-1}B$ entonces $B = \Delta\sigma_3^{-1}$.

$$\begin{aligned} B &= \Delta\sigma_3^{-1} \\ &= \sigma_1\sigma_2\sigma_3\sigma_1\sigma_2\sigma_1\ \sigma_3^{-1} \\ &= \sigma_1\sigma_2\sigma_3\sigma_1\sigma_2\sigma_3^{-1}\sigma_1 \\ &= \sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\sigma_3^{-1}\sigma_1 \\ &= \sigma_2\sigma_1\sigma_2\sigma_3\sigma_2\sigma_3^{-1}\sigma_1 \\ &= \sigma_2\sigma_1\sigma_3\sigma_2\sigma_3\sigma_3^{-1}\sigma_1 \\ &= \sigma_2\sigma_1\sigma_3\sigma_2\sigma_1 \end{aligned}$$

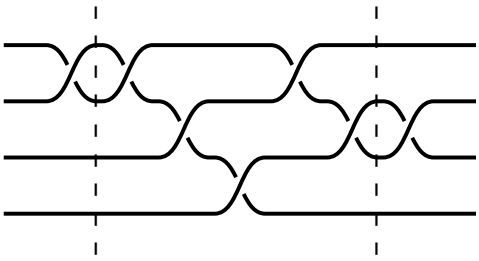
De aquí, bien se puede tomar B como cualquiera de las representaciones equivalentes a la trenza $\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1$, es decir:

$$B = \sigma_2\sigma_1\sigma_3\sigma_2\sigma_1 = \sigma_2\sigma_3\sigma_1\sigma_2\sigma_1 = \sigma_2\sigma_3\sigma_2\sigma_1\sigma_2 = \sigma_3\sigma_2\sigma_3\sigma_1\sigma_2 = \sigma_3\sigma_2\sigma_1\sigma_3\sigma_2$$

Siguiendo donde nos quedamos, se tiene entonces que $w = \sigma_1\Delta^{-1}\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_2$. Pasamos Δ a la izquierda mediante la relación $\sigma_i\Delta = \Delta\sigma_{n-i}$, con lo que obtenemos la forma normal de Garside

$$w = \Delta^{-1}\sigma_3\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_2,$$

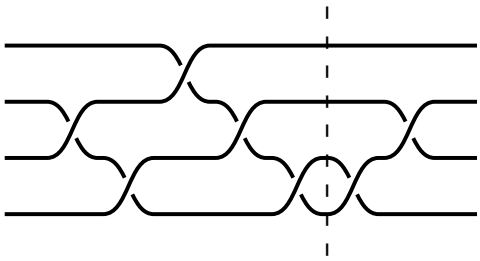
Ahora viene siendo lo complejo, debemos descomponer la parte positiva de la forma normal de Garside, es decir, descomponer $\sigma_3\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_2$ en una factorización cargada a la izquierda. Consideramos $B = \sigma_3\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_2$, la representación de B es la siguiente:



podemos ver $B = w_1'\ w_2'\ w_3'$ donde $w_1' = \sigma_3$, $w_2' = \sigma_3\sigma_2\sigma_1\sigma_3\sigma_2$ y $w_3' = \sigma_2$. Es claro que no es una factorización cargada a la izquierda, sin embargo podemos observar, que mediante las relaciones de Artin, es sencillo pasar algunos generadores de w_3' a w_2' y a su vez de w_2' a w_1'.

$$\begin{aligned} B &= \sigma_3\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_2 \\ &= \sigma_3\sigma_3\sigma_2\sigma_3\sigma_1\sigma_2\sigma_2 \\ &= \sigma_3\sigma_2\sigma_3\sigma_2\sigma_1\sigma_2\sigma_2 \\ &= \sigma_3\sigma_2\sigma_3\sigma_1\sigma_2\sigma_1\sigma_2 \\ &= \sigma_2\sigma_3\sigma_2\sigma_1\sigma_2\sigma_1\sigma_2 \\ &= \sigma_2\sigma_3\sigma_1\sigma_2\sigma_1\sigma_1\sigma_2 \\ &= \sigma_2\sigma_1\sigma_3\sigma_2\sigma_1\sigma_1\sigma_2 \end{aligned}$$

cuya representación es



así tenemos $B = w_1 w_2 = \sigma_2\sigma_1\sigma_3\sigma_2\sigma_1 \ \cdot \ \sigma_1\sigma_2$ y que además se tienen que

$$\begin{aligned} w_1 &= \sigma_2\sigma_1\sigma_3\sigma_2\sigma_1 \\ &= \sigma_2\sigma_3\sigma_1\sigma_2\sigma_1 \\ &= \sigma_2\sigma_3\sigma_2\sigma_1\sigma_2 \\ &= \sigma_3\sigma_2\sigma_3\sigma_1\sigma_2 \end{aligned}$$

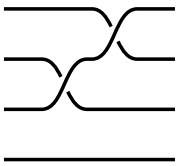
se tiene que $S(w_2) = \{2\} \subseteq \{2,3\} = F(w_1)$ y por tanto ahora si, B es una factorización cargada a la izquierda y, por tanto

$$w = \Delta^{-1} \cdot \sigma_2\sigma_1\sigma_3\sigma_2\sigma_1 \ \cdot \ \sigma_1\sigma_2$$

es su forma normal greedy.

■

Ejemplo 2.5. Sea $w = \sigma_2\sigma_3 \in B_4$. Procedamos como el ejemplo anterior. Como w no presenta ningún generador con potencia negativa, obviamos ese paso. Veamos su representación:



Podemos darnos cuenta que w es ella misma en sí la única trenza de permutación. Por tanto $w = \Delta^0 B = \sigma_2\sigma_3$, es su propia forma normal greedy.

■

Ejemplo 2.6. Consideremos la trenza $w = \sigma_1\sigma_2^{-1}\sigma_3\sigma_1^{-1}$.

Vemos que en la expresión nos encontramos los generadores σ_2^{-1} y σ_1^{-1} ambos con potencia negativa. Así, mediante los mismos cambios que hemos estado haciendo, debemos reemplazar σ_2^{-1} y σ_1^{-1} por $\Delta^{-1}C$ y $\Delta^{-1}B$, respectivamente, con C y B trenzas positivas.

$$\begin{aligned} C &= \Delta\sigma_2^{-1} \\ &= \sigma_1\sigma_2\sigma_3\sigma_1\sigma_2\sigma_1 \ \sigma_2^{-1} \\ &= \sigma_1\sigma_2\sigma_3\sigma_2\sigma_1\sigma_2\sigma_2^{-1} \\ &= \sigma_1\sigma_2\sigma_3\sigma_2\sigma_1 \end{aligned}$$

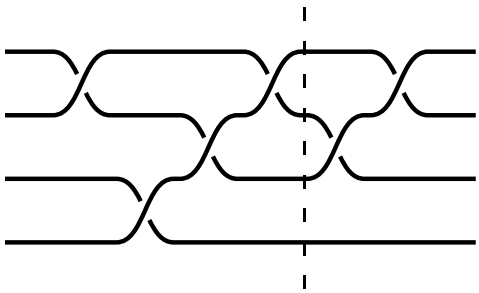
por lo que tendríamos $w = \sigma_1\Delta^{-1}\sigma_1\sigma_2\sigma_3\sigma_2\sigma_1\sigma_3\sigma_1^{-1}$. Ahora bien, podríamos actuar de forma análoga con σ_1^{-1}, sin embargo si prestamos un poco de atención:

$$\begin{aligned} w &= \sigma_1\Delta^{-1}\sigma_1\sigma_2\sigma_3\sigma_2\sigma_1\sigma_3\sigma_1^{-1} \\ &= \Delta^{-1}\sigma_3\sigma_1\sigma_2\sigma_3\sigma_2\sigma_1\sigma_3\sigma_1^{-1} \\ &= \Delta^{-1}\sigma_3\sigma_1\sigma_2\sigma_3\sigma_2\sigma_3\sigma_1\sigma_1^{-1} \\ &= \Delta^{-1}\sigma_3\sigma_1\sigma_2\sigma_3\sigma_2\sigma_3 \end{aligned}$$

Y de esta manera, la forma normal de Garside de w es

$$w = \Delta^{-1}\sigma_3\sigma_1\sigma_2\sigma_3\sigma_2\sigma_3$$

Ahora, para dar la forma normal de greedy recordamos que debemos descomponer en trenzas de permutación dando una factorización cargada a la izquierda. Consideremos $A = \sigma_3\sigma_1\sigma_2\sigma_3\sigma_2\sigma_3$, su representación es:



Se tiene que $A = \sigma_3\sigma_1\sigma_2\sigma_3\sigma_2\sigma_3 = (\sigma_3\sigma_1\sigma_2\sigma_3)\ (\sigma_2\sigma_3) = w_1w_2$ y además $S(w_2) = \{2\} \subseteq \{2,3\} = F(w_1)$. Por tanto $w = \Delta^{-1} \cdot (\sigma_3\sigma_1\sigma_2\sigma_3) \cdot (\sigma_2\sigma_3)$ está en su forma normal greedy.

■

Teorema 2.5 (Caracterización de la igualdad entre palabras)**.** *La condición necesaria y suficiente para que dos palabras en B_n sean iguales es que sus formas normales greedy sean idénticas, asumiendo que previamente se fija la palabra que representa a cada trenza de permutación.*

La complejidad[3] de transformar una palabra w, desde el punto de vista del grupo de presentación de Artin, a su forma normal greedy es $O(|w|^2 n\ log\, n)$, donde $|w|$ es la longitud de la palabra w en B_n.

2.2. Algoritmo de Artin

A efectos prácticos, para resolver el problema de la palabra, no se requiere esencialmente una isotopía completamente invariante como se establece teóricamente en la página 25. Existen varios algoritmos, suficientemente buenos, que pueden dar solución a la equivalencia entre dos trenzas sin necesidad explicita de encontrar tal isotopía. Entre los más destacados, podemos encontrar el algoritmo de Naïve, el método de revertir subpalabras o el algoritmo de Artin, el cual trataremos en las siguientes lineas.

Artin ofrece una solución al problema de la isotopía de trenzas que, *«siendo no muy*

[3]Página 189 del artículo *A practical attack on some braid group based cryptographic primitives* [5]

eficiente ni relativamente claro de computar» como él mismo mencionó, fue el primero de la historia que se desarrolló y se implementó. Estamos hablando en términos de ineficiencia y computo relativos al momento histórico.

Consideremos una trenza pura $b \in P_n$, el cual conecta los puntos A_i con B_i mediante la curva C_i para $i = 1, \dots n$. Sea b_1 la trenza obtenida de remplazar la curva C_1 de b por la curva D_1 que conecta A_1 y B_1 mediante una línea recta que no interfiere con las demás.

Definimos $c_1 = bb_1^{-1}$, entonces c_1 tiene la propiedad especial que si tu remplazas la curva C_1 en c_1 por D_1 se tiene que $b_1 b_1^{-1} = e$. A este tipo de trenzas, a la trenza c_1, se le llama trenza 1-*pura*. Ésto se extiende de forma similar a la definición para una trenza *i-pura*.

Sin tener en cuenta el hecho de que C_1 está aún en la trenza c_1, realizando la misma isotopía que coge $b_1 b_1^{-1}$ a e y *estirando* lo necesario. Encontraremos que la curva C_1 está enredada en la trenza identidad e. Esto es, si a esta trenza c_1, llamada trenza 1-*pura*, remplazamos C_1 por D_1 la hebra recta, obtendremos ahora la trenza identidad.

Nótese ahora que como $bb_1^{-1} \equiv c_1$, se tiene que $b \equiv c_1 b_1$. Esto quiere decir, que cualquier trenza es isotópica al producto de una trenza 1-*pura* c_1, y una trenza b_1 cuya primera hebra no interfiere con las demás. En otras palabras, podriamos ver b_1 como una trenza en B_{n-1} con la diferencia de se le añade a la izquierda una hebra que hace que esté en B_n. Esto hace que cualquier trenza pura de n hebras tenga una única descomposición $b = c_1 c_2 \cdots c_n$, donde cada c_i es una trenza *i-pura*. Por otro lado, es claro que en la descomposición de e, se tiene que cada $c_i = e$.

Ejemplo 2.7. Consideremos $\gamma = \sigma_2\sigma_1\sigma_2^{-1}\sigma_3\sigma_3\sigma_2\sigma_3^{-1}\sigma_1^{-1}\sigma_2\sigma_1^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_3\sigma_2^{-1}$ la trenza pura que mencionamos en el Ejemplo 2.1 de la página 23. Se tiene que $\varphi(\gamma) = c_1 b_1$ donde

$$c_1 = \sigma_1\sigma_2^{-1}\sigma_3\sigma_3\sigma_2\sigma_1^{-1}\sigma_1^{-1}\sigma_1^{-1} \qquad b_1 = \sigma_2^{-1}\sigma_3\sigma_2^{-1}\sigma_2\sigma_3^{-1}\sigma_2$$

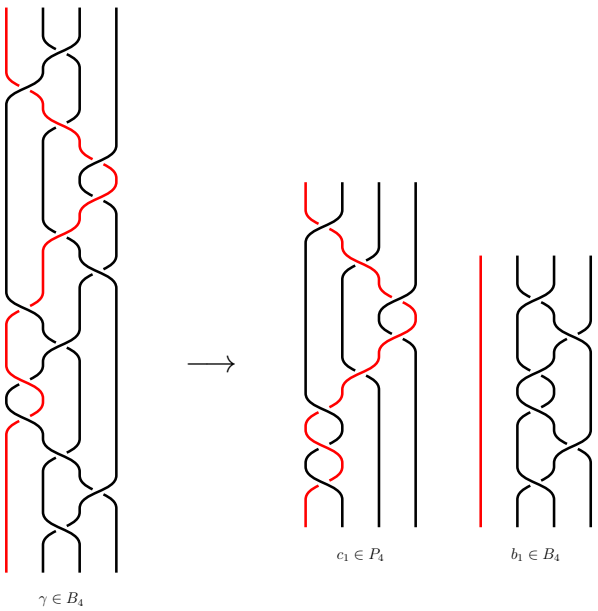


Fig 2.14 Descomposición de γ en c_1 y b_1.

Sea φ la función que toma una trenza $b \in B_n$ y devuelve la trenza 1-*pura* c_1 y la trenza $b_1 \in B_{n-1}$. Consideramos el siguiente algoritmo más como una prueba de concepto que una implementación en sí misma.

Algorithm 1: Algoritmo de Artin

Input : b_1, b_2
Output: Si b_1 y b_2 son equivalentes

$b = b_1 b_2^{-1}$
if b *no es pura* **then**
  return "$b_1 \not\equiv b_2$"
end
for $i = 1, \dots n$ **do**
  $\varphi(b) = (b_i, c_i)$
  if $c_i \neq e$ **then**
    return "$b_1 \not\equiv b_2$"
  end
  $b := b_i$
end
return "$b_1 \equiv b_2$"

Capítulo 3

Criptografía de trenzas

1. El problema de la conjugación

En los últimos años se han realizado numerosos intentos de formas y aplicaciones en la criptografía originados por los problemas en la teoría combinatoria de grupos. Concretamente se han dado esquemas criptográficos derivados de los grupos de trenzas y algunos problemas que ocurren en estos grupos. Nosotros nos centraremos concretamente en uno muy significativo que ha acabado aplicándose como sistema criptográfico, el *problema de la conjugación.*

Problema de la posibilidad de la conjugación. Dado dos trenzas w, v, nos preguntaremos, ¿Existe una tercera trenza a tal que $w = ava^{-1}$?, es decir, ¿Se puede determinar la existencia de una trenza a tal que w y v son conjugados[1]?

La primera aparición del problema de la conjugación fue en el paper de Artin [12]. Artin dió un algoritmo[2] para resolver el problema de la isotopía de trenzas e introduce un nuevo tipo de problema, que a su vez, no fue resuelto. Imagínese que una trenza se enrolla alrededor de un eje para que los extremos se toquen formando una trenza cerrada.

Dados dos trenzas b_1, b_2, sus cierres son isotópicos si y sólo si $b_1 \equiv cb_2c$, para alguna trenza c.

Entendió además, que una solución para este problema podría aplicarse al problema de identificar nudos y enlaces. Desde entonces, han aparecido variaciones del problema de la conjugación, los cuales se usan como problema base para diferentes criptosistemas, tales como:

Problema de la conjugación. Dado dos trenzas conjugadas b_1 y b_2, buscar una trenza conjugante c, es decir, buscar una trenza c tal que $b_1 \equiv c^{-1}b_2c$

Problema de la conjugación múltiple simultánea. Dados m pares de elementos $(b_1, k_1), \ldots, (b_m, k_m)$ en $B_n \times B_n$, donde cada par es conjugada por una misma trenza. Encontrar una trenza conjugante $c \in B_n$ tal que $b_i \equiv c^{-1}b_ic$ para todo $i = 1, \ldots m$.

[1] Definición 1.6 de palabra conjugada, página 5
[2] Algoritmo 1, página 34

2. Criptografía. Definiciones

Para comenzar el capítulo y empezar a relacionar toda la teoría de trenzas que hemos desarrollado y la criptografía, será esencial definir los conceptos y las nociones básicas de esta última. Debemos entender que lo interesante de la criptografía, que es a su vez su principal ventaja y su principal problema, es la dificultad computacional de los problemas matemáticos en los que están basado. Es sencillo entender, que a mayor nivel de dificultad, mayor es la seguridad, menor la posibilidad de un posible ataque, pero a su vez mayor necesidad de recursos por su elevado nivel computacional.

Ejemplo de ello es la factorización de números de la forma $n = pq$, donde p y q son primos exageradamente grandes. Este problema lo podemos ver aplicado en el sistema de criptografía RSA, uno de los sistemas más utilizado y que, a día de hoy, es prácticamente imposible de *crackear*. O el problema del logaritmo discreto, que dada una ecuación $g^x = h$ en un grupo donde g es un generador, determina x.

Debemos dejar claro antes de continuar, que existe una diferencia sustancial entre *sistemas de intercambio de clave pública* o *criptografía asimétrica* y *sistema de clave privada* o *criptografía simétrica*.

La siguiente definición fue tomada del libro *Introduction to Cryptography. Principles and Applications* [9].

Definición 3.1 (Criptosistema de clave privada)**.** *Sea (K, M, C) una terna donde K, M y C son tres conjuntos. Un sistema de clave privada, o criptosistema de clave privada, o criptosistema simétrico es una aplicación*

$$E : K \times M \longrightarrow C$$

tal que para cada $k \in K$, la aplicación

$$\begin{array}{rccc} E_k : & M & \longrightarrow & C \\ & m & \longmapsto & E(k, m) \end{array}$$

es biyectiva.

A los elementos $m \in M$ los denominaremos *plaintext*, *texto plano* o *mensaje*; los elementos $c \in C$, los *cyphertext*, *texto cifrado* o *mensaje cifrado*; y los elementos $k \in K$, *key* o *clave*. Por otro lado, a la aplicación E_k la llamaremos *encriptación* o *algoritmo de cifrado*, y a su inversa, $D_k := E_k^{-1}$, la *desencriptación* o *algoritmo de descifrado*.

En este sistema de criptografía simétrica, emisor y receptor son conocedores de la aplicación de encriptación (y por tanto de su inversa). La *clave* k, es compartida por ambos y mediante E_k y D_k mantienen el intercambio de información m. Es por esto, que la seguridad del sistema reside en como de compleja es la función E_k.

Ejemplos importantes de criptografía simétrica son *data encryption standard* (DES) o *advanced encryption standard* (AES)[3].

En esta clase de criptografía, se debe destacar dos formas o categorias de encriptación: Los *cifrados por bloques* o *block ciphers* y los *cifrados por flujo* o *stream ciphers*. Los *block ciphers*, a diferencia de los *stream ciphers*, cifran los textos planos por bloques de longitud fija. Mientras que los *steam ciphers*, lo hace carácter a carácter.

Definición 3.2 (Stream Ciphers)**.** *Sea K un conjunto de claves y M un conjunto de textos planos. En este contexto, llamaremos a los elementos de M carácteres. Entonces, dado $m = m_1 m_2 \ldots$, un cifrado por flujo o stream cipher es una aplicación*

$$\begin{array}{rccc} E^* : & K^* \times M^* & \longrightarrow & C^* \\ & (k, m) & \longmapsto & E^*(k, m) := c \end{array}$$

con $c = c_1 c_2 \ldots$, $k := k_1 k_2 \ldots$ tal que $c_i = E(k_i, m_i)$, donde $m_i \in M$ y $k_i \in K$ para cada $i \in \mathbb{N}$.

Esto es, dado un texto plano $m = \prod_i m_i$, se cifra carácter a carácter con su clave correspondiente $k_i \in K$. Y por tanto, la desencriptación ser procederá de igual modo, carácter a carácter.

Definición 3.3 (Block Ciphers)**.** *Un cifrado por bloques o block cipher es un esquema simétrico de clave privada con $M = C = \{0,1\}^n$ un espacio de claves $K = \{0,1\}^r$:*

$$\begin{array}{rccc} E : & \{0,1\}^r \times \{0,1\}^n & \longrightarrow & \{0,1\}^n \\ & (k, m) & \longmapsto & E(k, m) \end{array}$$

Usando una clave secreta k de longitud binaria r, el algoritmo E particiona el texto plano en bloques m de longitud binaria n, y encripta bloque a bloque. El resultado es un bloque encriptado $c = E(k, m)$ con longitud también n.

Las longitudes de bloques más usadas son de 64 (DES) y 128 (AES), y claves de longitudes binarias de 56 (DES) o 128, 192 o 256 (AES).

Si dos individuos desean intercambiar información, previamente deben generar una clave privada e intercambiarla. Debido a esto, surgieron los sistemas de intercambio de claves públicas, o como también se conocen, la criptograía asimétrica; como protocolo para hacer más segura la comunicación.

La siguiente definición fue tomada literalmente del artículo *Publick-key cryptosystems provably secure against chosen ciphertext attacks* [11].

Definición 3.4 (Criptosistema de clave pública)**.** *Un criptosistema de clave pública, Public-Key cryptosystem ó PK-Cryptosystem, consiste en:*

1. *Una generador de claves G que dado un elemento de entrada n devuelve un par (e, d), donde e es la clave pública escrita en un archivo público, esto es que cualquiera pueda conocerlo, y d la clave privada.*

2. *Un mecanismo de encriptación que dado un mensaje m de texto plano y la clave pública e, genera un mensaje de texto cifrado c.*

[3]Para mayor detalle y en profundidad, [9] página 19

3. *Un mecanismo de desencriptación que dado la clave pública e, la clave privada d y el mensaje cifrado c, devuelve el mensaje en texto plano m.*

La definición obedece que si el mecanismo de desencriptación toma e, d, c dónde c fue generado por el mecanismo de encriptación tomando d y m, y dónde e, d fueron creados por el generador de claves G, entonces devuelve el mensaje m.

Este sistema criptográfico no se trata de un algoritmo para cifrar mensajes, sino más bien para hacer segura el canal de comunicación. Todos los sistemas de intercambios de claves han ido surgiendo como variaciones del protocolo propuesto por *Whitfield Diffie* y *Martin Hellman* en 1976.

Lo interesante de esta clase de esquemas criptográficos es, que no importa como de bueno sea el sistema de clave privada o algoritmo de cifrado que lo acompaña, la seguridad reside en la clave generada. No es la clave privada la que se comparte, como en los sistemas criptográficos simétricos, sino que esta se genera mediante un intercambio de claves públicas y se reserva a uno mismo sin la posibilidad de que ningun intruso la intercepte. Este cambio de mentalidad frente a los problemas de la criptografía fue crucial en las nuevas lineas de investigación que surgieron a posteriori.

Para establecer un orden de seguridad entre los sistemas de criptografía, se debe considerar el tipo de ataque que se puede asumir, es decir, tener en cuenta la capacidad del posible intruso, y el tipo de *crackeo* que deseamos prevenir. Dadas estas características, explicaremos cuánto de dificil es, tanto en sentido de eficacia como de nivel computacional requerido, romper un criptosistema con ataques específicos.

Presentaremos los tipos de ataques en orden creciente de severidad.

1. *Ciphertext-only attack.*
   El atacante sólo capta el ciphertext o texto cifrado.

2. *Known-plaintext attack.*
   El atacante conoce tanto el texto plano como el cifrado.

3. *Chosen-plaintext attack.*
   El atacante puede elegir el texto plano a su elección y explotarlo mediante mecanismos de encriptación para ver los valores del sistema.

4. *Chosen-ciphertext attack.*
   Donde además del acceso al mecanismo de cifrado, el atacante puede elegir, según convenga, el texto cifrado a su elección y mediante el mecanismo de descifrado obtener el texto plano correspondiente.

Los sistemas de criptografía basadas en trenzas que propondremos y desarrollaremos en las próximas páginas consisten en criptosistemas de clave pública sensibles a ataques *chosen-plaintext*, que siguen el modelo propuesto por *Diffie-Hellman* y originados por el *problema de la conjugación.*

Escenario

Consideremos dos personas Alicia y Bob. Consideremos la red Internet como canal de comunicación entre ordenadores, en el cual se entiende que están conectados los or-

denadores de Alicia, Bob y los posibles intrusos respectivamente. Supóngase que Alicia quiere comunicarse con Bob pero no quiere que nadie más intercepte la información que intercambian. Entre ellos se establece un criptosistema de clave pública.

3. Criptosistemas basados en grupos de trenzas

A la hora de establecer un criptosistema basado en trenzas, debemos tener cuidado al confiar en el problema de la conjugación de las trenzas.

Es decir, si por ejemplo la trenza pública es $x = \sigma_2^{-1}\sigma_3$ y la trenza conjugante $c = \sigma_1$, deberemos tener cuidado para que posibles interceptores no capten el conjugador c. Por la forma de la trenza en el problema de la conjugación de la palabra, si enviásemos $cxc^{-1} = \sigma_1\sigma_2^{-1}\sigma_3\sigma_1^{-1}$, sería sencillo darse cuenta quién es el conjugador y por tanto quien es x. Sin embargo, si enviásemos la forma normal de greedy[4] de cxc^{-1}:

$$\Delta^{-1}\sigma_1\sigma_2\sigma_3\sigma_2\sigma_2\sigma_3$$

sería mucho mas difícil conocer c.

De ahora en adelante, toda trenza que consideremos enviar en los distintos criptosistemas que expongamos se considerarán en su forma normal. El siguientes esquema está basado en el problema de la conjugación o el problema de la conjugación múltiple simultánea.

3.1. Criptosistema Anshel-Anshel-Fisher-Goldfeld

El criptosistema Anshel-Anshel-Fisher-Goldfeld [AAFG], fue un criptosistema teórico de clave pública ideado por estos mismos en 1999. En 2001, se reunieron para implementar tal criptosistema en base a los grupos de trenzas.

Algorithm 2: Criptosistema Anshel-Anshel-Fisher-Goldfeld

Información Pública:
Indice del grupo de trenzas $n \in \mathbb{N}$
Subconjunto de trenzas $X_A = \{a_1, a_2, \cdots a_r\} \subset B_n$
Subconjunto de trenzas $X_B = \{b_1, b_2, \cdots b_s\} \subset B_n$
Clave Privada :
Alicia: $a \in \langle a_1, \ldots, a_r \rangle$
Bob: $b \in \langle b_1, \ldots, b_s \rangle$
Clave Pública :
Alicia envía $(ab_1a^{-1}, \ldots, ab_sa^{-1})$
Bob envía $(ba_1b^{-1}, \ldots, ba_rb^{-1})$
Clave compartida : $aba^{-1}b^{-1}$

[4]Ejemplo 2.4, página 29

Teorema 3.1. *En el intercambio de claves del sistema Anshel-Anshel-Fisher-Goldfeld, Alicia y Bob pueden obtener $aba^{-1}b^{-1}$ de manera eficiente.*

Demostración. Alicia conoce a, toda la *información pública* y recibe $(ba_1b^{-1}, \ldots, ba_rb^{-1})$ de Bob. Sea $a = x_1x_2\cdots x_t$ con $x_i \in \{a_1, a_2, \ldots a_r\}$, se tiene que como $bab^{-1} \equiv bx_1b^{-1}\cdots bx_tb^{-1}$ entonces

$$\left((bx_1b^{-1}\cdots bx_tb^{-1})\ a^{-1}\right)^{-1} = (bab^{-1}a^{-1})^{-1} = aba^{-1}b^{-1}$$

De forma análoga. Bob conoce b, toda la *información pública* y recibe $(ab_1a^{-1}, \ldots, ab_sa^{-1})$ de Alicia. Sea $b = y_1\cdots y_l$ con $y_i \in \{b_1, b_2, \cdots b_s\}$, se tiene que como $aba^{-1} = ay_1a^{-1}\cdots ay_la^{-1}$, entonces

$$(ay_1a^{-1}\cdots ay_la^{-1})\ b^{-1} = aba^{-1}b^{-1}$$

Y por tanto, ambos comparten la misma clave. ■

El siguiente ejemplo pone en práctica el criptosistema. Debido a la complejidad que pueda resultar la explicación si se consideran subconjuntos moderadamente grandes y trenzas sutilmente complejas, se muestra el caso de una situación sencilla:

Ejemplo 3.1. Nos situamos en el escenario antes definido. Consideremos $n = 4$, sea $X_A = X_B = \{\sigma_1, \sigma_3\} \subset B_4$. Sea $a = \sigma_1$ la clave privada de Alicia, y sea $b = \sigma_3\sigma_1^{-1}$ la clave privada de Bob.

Bob le envía a Alicia $[(\Delta^{-1}\sigma_1\sigma_3\sigma_2\sigma_3\sigma_1\sigma_2),\ (\sigma_3)]$, donde

$$\begin{aligned} ba_1b^{-1} &= \sigma_3\sigma_1^{-1}\ \cdot\ \sigma_1\ \cdot\ \sigma_1\sigma_3^{-1} \\ &= \Delta^{-1}\sigma_1\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1 \end{aligned}$$

$$\begin{aligned} ba_2b^{-1} &= \sigma_3\sigma_1^{-1}\ \cdot\ \sigma_3\ \cdot\ \sigma_1\sigma_3^{-1} \\ &= \sigma_3\sigma_1^{-1}\sigma_1\sigma_3\sigma_3^{-1} \\ &= \sigma_3 \end{aligned}$$

De esta forma, Alicia sabe que como $bab^{-1} = b\sigma_1b^{-1} = \Delta^{-1}\sigma_1\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1$, se tiene que

$$\begin{aligned} \left(bab^{-1}a^{-1}\right)^{-1} &= \left(\Delta^{-1}\sigma_1\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1\sigma_1^{-1}\right)^{-1} \\ &= \left(\Delta^{-1}\sigma_1\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\right)^{-1} \\ &= \sigma_2^{-1}\sigma_3^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_3^{-1}\sigma_1^{-1}\Delta \\ &= \sigma_2^{-1}\sigma_3^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_3^{-1}\sigma_1^{-1}\sigma_1\sigma_2\sigma_3\sigma_1\sigma_2\sigma_1 \\ &= \sigma_2^{-1}\sigma_3^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_3^{-1}\sigma_2\sigma_3\sigma_1\sigma_2\sigma_1 \\ &= \sigma_2^{-1}\sigma_3^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_3^{-1}\sigma_2\sigma_3\sigma_2\sigma_1\sigma_2 \\ &= \sigma_2^{-1}\sigma_3^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_3^{-1}\sigma_3\sigma_2\sigma_3\sigma_1\sigma_2 \\ &= \sigma_2^{-1}\sigma_3^{-1}\sigma_1^{-1}\sigma_3\sigma_1\sigma_2 \\ &= \sigma_2^{-1}\sigma_1^{-1}\sigma_3^{-1}\sigma_3\sigma_1\sigma_2 \\ &= e \end{aligned}$$

Mientras que, por otro lado

$$\begin{aligned} aba^{-1}b^{-1} &= \sigma_1\sigma_3\sigma_1^{-1}\sigma_1^{-1}\sigma_1\sigma_3^{-1} \\ &= \sigma_1\sigma_3\sigma_1^{-1}\sigma_3^{-1} \\ &= \sigma_1\sigma_3\sigma_3^{-1}\sigma_1^{-1} \\ &= e \end{aligned}$$

De forma análoga se resuelve para Bob.

■

En el ejemplo anterior se ha dado el caso de que la propia *clave compartida* es justamente la trenza identidad e. Esto es debido a que en la elección de a y b en B_n, se han tomado dos trenzas que conmutan entre sí.

Sea n el orden del grupo de trenzas B_n, y sea $G_A, G_B \leq B_n$ con $r = s$ generadores cada uno. Consideramos que, para cada generador a_i (resp. b_i) del subgrupo G_A (resp. G_B), está formado por $5 \leq l \leq 10$ generadores de Artin. Y sea a (resp. b) la clave privada de Alicia (resp. Bob), formada por k generadores de G_A (resp G_B). Se contempla realizar un estudio de tiempos en pruebas de generación de claves.

Para ello se realiza 100 pruebas con distintos parámetros en un ordenador con procesador de 8 núcleos a 1.60GHz cada uno.

n	r = s	k	tiempo medio por clave (seg)
4	3	5	7.2504
8	3	5	59.002
20	3	5	1549.8618
4	10	20	5.0701
8	10	20	47.522
20	10	20	1714.3092

Describimos a continuación el script diseñado y utilizado, escrito en *SageMath*, el cual contiene el algoritmo del criptosistema AAFG.

Script del algoritmo AAFG en SageMath

```
from random import *
from functools import *
from time import *

orden = 5  # fijamos el orden de Bn, por defecto
def CrypSysAAFG(n = orden, numGeneradores = 5, r = 3):
    B = BraidGroup(n)
    def Alicia():
        ga = list(map(lambda x: list(map(lambda x: randint(1, n-1), range(randint(5, 10)))), range(r)))
        xt = list(map(lambda x: choice(ga), range(numGeneradores)))
        a = B(list(reduce((lambda a,b: a+b),xt)))
        indxa = [ i for x in xt for i in range(len(ga))  if ga[i]==x]
        return ga, xt, a, indxa
    def Bob():
        gb = list(map(lambda x: list(map(lambda x: randint(1, n-1), range(randint(5, 10)))), range(r)))
        yk = list(map(lambda x: choice(gb), range(numGeneradores)))
        b = B(list(reduce((lambda a,b: a+b),yk)))
```

```
        indxb = [ i for y in yk for i in range(len(gb))  if gb[i]==y]
        return gb, yk, b, indxb

    ga, xt, a, indxa = Alicia()
    gb, yk, b, indxb = Bob()

    # Bob le envia a Alicia
    clavePublica = list(map(lambda x: (b*B(x)*b.inverse()).left_normal_form(), ga))

    def generaClaveCompartida(k=None):
        bab = list(reduce((lambda a,b: a+b), map(lambda i : clavePublica[i], indxa)))
        tmp = B([1])*B([1]).inverse()
        for i in bab:
            tmp*=i
        bab = tmp
        ## Alicia genera la clave compartida
        ababAlicia = (bab*a.inverse()).inverse()
        abab = a * b * a.inverse() * b.inverse()
        ## Check de clave compartida
        if ababAlicia.left_normal_form() == abab.left_normal_form():
            k = abab
        return k

    claveCompartida = generaClaveCompartida()
    return claveCompartida

def benchmark(m=orden, r = 3, k = 5):
    t0 = time()
    CrypSysAAFG(m,k,r)
    t1=time()
    return(t1-t0)
```

3.2. Criptosistema de intercambio Ko et al

Como sabemos, los grupos de trenzas no son en general conmutativo[5], sin embargo existen elementos que conmutan entre sí. Es más, existen grandes subgrupos de B_n los cuales no comparten ninguna trenza y por tanto conmutan entre ellos. Aprovechando esto, Ko et al. en el artículo *New public-key cryptosystem using braid groups* [15], construyó un nuevo sistema de intercambio de claves basado en el protocolo de intercambio de *Deffie-Hellman*. El esquema utiliza una funcion f unidireccional con la peculiaridad de que sus imágenes son facilmente computables pero sus preimagenes no.

Sea n un entero positivo con $n > 4$. Sea l, r con $1 < l$ y $r < n$. Consideramos dos subgrupos de B_n, LB_l como el subgrupo generado por $\{\sigma_1, \ldots, \sigma_{l-1}\}$ y RB_r como el subgrupo generado por $\{\sigma_{n-r+1}, \ldots, \sigma_{n-1}\}$. Es decir, LB_l es el subgrupo de B_n el cual posee las trenzas que mueven las $l-1$ primeras de hebras de la izquierda, mientras que RB_r contiene las trenzas que mueven las $r+1$ hebras desde la derecha. Esto hace que se permita la conmutatividad entre las trenzas de LB_l con las de RB_r. Así, la funcion f unidireccional que Ko et al. propone es la siguiente:

$$f_L : LB_l \times B_n :\longrightarrow B_n \times B_n$$

donde $f(b, c) = (bcb^{-1}, c)$. De forma similar, se puede considerar la funcion f_R con dominio $RB_r \times B_n$. Estas funciones son unidireccionales ya que es sencillo computar bcb^{-1}, pero dado bcb^{-1} y c, la computación de b implica resolver el *problema de la conjugación*.

Problema. *Diffie-Hellman* como problema de conjugación. Dado una trenza $x \in B_n$ y axa^{-1} y bxb^{-1}, con $a \in LB_n$ y $b \in RB_n$ desconocidos, encontrar la trenza $abx(ab)^{-1}$ o $bax(ba)^{-1}$.

[5]Demostración en página 11, *Lema* 1.4.

Algorithm 3: Sistema de intercambio de claves Ko et al.

Información Pública:
Indice del grupo de trenzas $n \in \mathbb{N}$

$l, r \in \mathbb{N}$ tal que $l + r = n$

Una trenza $x \in B_n$

Clave Privada : Alicia: $a \in LB_l$, Bob: $b \in RB_r$

Clave Pública : Alicia envía axa^{-1}, Bob envía bxb^{-1}

Clave compartida : $(ab)x(ab)^{-1}$

Esquema de encriptación Ko et al.

Además del sistema de intercambio de claves, Ko et al propone, en *New public-key cryptosystem using braid groups* [15], un esquema de encriptación usando una funcion *hash* H. Esta función es unidireccional y resitente a colisiones, es decir: Dados $a \neq b$ es totalmente imposible que $H(a) = H(b)$. Además de esto, es completamente inviable que conociendo $H(a) = H(b)$ se puedan determinar a y b. Así, definimos la función $H : B_n \longrightarrow \{0,1\}^k$ como el hash que envía el grupo de trenzas al espacio de mensaje $\{0,1\}^k$. Y definiremos el operador *exclusive-or*, también conocido como XOR, que denotaremos como $\oplus$, de la siguiente forma:

$$\begin{aligned} \oplus : \ \{0,1\}^k \times \{0,1\}^k &\longrightarrow \{0,1\}^k \\ \left((a_i)_{i=1}^k, (b_i)_{i=1}^k\right) &\longmapsto (a_i)_{i=1}^k \oplus (b_i)_{i=1}^k := (a_i + b_i (mod 2))_{i=1}^k \end{aligned}$$

Este operador $\oplus$, es un operador logico Booleano, usado frecuentemente en criptografía tanto en la generación de *bits pareados*, como en la comprobación de errores y tolerancia a fallos. XOR compara dos bits de entrada y genera un bit de salida. La lógica es la siguiente:

Si dos bits son los mismos, entonces devuelve 0. Si no, es decir, si son diferentes, devuelve 1.

Algorithm 4: Protocolo de encriptación Ko et al.

Información Pública:
Indice del grupo de trenzas $n \in \mathbb{N}$

$l, r \in \mathbb{N}$ tal que $l + r = n$

Clave Pública : Alicia elige $a \in LB_l$

Clave Privada : Alicia envía (x, y), con $x \in B_n$ e $y = axa^{-1}$

Encriptación :

Para que Bob transmita $m \in \{0,1\}^k$ a Alicia

1. Bob elige $b \in RB_r$ aleatorio

2. Bob envía $(c, d) := (bxb^{-1},\ H(bxb^{-1}) \oplus m)$ a Alicia

Desencriptación : Alicia computa $m = H(aca^{-1}) \oplus d$

De esta forma, cuando Alicia computa $H(aca^{-1}) \oplus d$, realmente está calculando m:

$$\begin{aligned} H(aca^{-1}) \oplus d &= H(abxb^{-1}a^{-1}) \oplus H(byb^{-1}) \oplus m \\ &= H(abxb^{-1}a^{-1}) \oplus H(baxa^{-1}b^{-1}) \oplus m \\ &= m \end{aligned}$$

puesto que $a \in LB_l$ y $b \in RB_r$ implica $ab = ba$, y por tanto $H(abxb^{-1}a^{-1}) = H(baxa^{-1}b^{-1})$.

3.3. Comunicación mediante sistema AAFG y Block Cipher.

La idea de generar la clave compartida es, para que a través de esta, se pueda realizar la comunicación de forma segura. Vamos a establecer un caso práctico, y sencillo, de una comunicación partiendo del escenario descrito en la página 38, en la que se propone utilizar el sistema de intercambio de claves AAFG para generar una clave privada, y tras esto, un método de encriptación elemental que he diseñado, para el intercambio de información.

Para proceder con este ejemplo vamos a utilizar *SageMath* que es un sistema de computación algebraica escrito en Python, C, entre otros; y que nos ayudará a realizar las operaciones con trenzas, ayuda que realmente nos será de mucha utilidad pues vamos a considerar parámetros relativamente altos y nos expondremos a una situación compleja a nivel práctico. La documentación de SageMath orientada a la computación de trenzas la podemos encontrar en [6].

Contexto.

Toda información que se transmita por la red se puede descomponer en bloques de 1 byte, esto es, 8 bits; o potencias de esta, 64, 128 o 256 bits. Cada carácter en un mensaje de texto plano, por ejemplo, es equivale a un byte. Por tanto, un mensaje que contenga el texto *hola mundo* equivaldría a 10 bytes (recordamos que el espacio cuenta también como un byte), es decir, diez bloques de 8 bits.

Consideraremos el grupo de trenzas B_8 de forma que, para cada representación binaria de cada carácter, cada elemento 1 equivale a la trenza elemental σ_i donde i es la posición de dicho 1 en tal representación. A este tipo de trenzas la denominaremos como *trenza binaria*. Por ejemplo: el carácter h tiene como representación binaria 01101000, que equivale a la trenza $h = \sigma_2\sigma_3\sigma_5$.

Con esto, el proceso de comunicación consistirá en:

1. Sea k la clave compartida generada de la misma forma que el ejemplo 3.1, página 40.
2. Sea $r \geq 1$, consideraremos $\Omega = \{a_1, \ldots, a_r\}$ el subgrupo de B_8 donde cada a_i es una trenza binaria.
3. Sea $\Phi : B_8 \longrightarrow B_8$ la conjugación por k, el homomorfismo algoritmo de cifrado.
4. Dado un mensaje $m \in \langle a_1, \ldots a_r \rangle$, y sea k nuestra clave compartida. Consideraremos el mensaje cifrado c asociado a m como $c = \Phi(m)$. De esta forma, si $m = m_1 \cdots m_t$

con $m_i \in \Omega$, se tiene que

$$c = \Phi(m) = \Phi(m_1) \cdots \Phi(m_t)$$

transfiriendo byte a byte, es decir, cada $\Phi(m_i)$ en su forma normal greedy.

5. Cuando el receptor reciba el mensaje cifrado c, le resultará una tarea ardua la desencriptación mediante la clave compartida k, pues cada $\Phi(m_i)$ viene dado en su forma normal greedy.

   Sin embargo, al ser conocedor de la clave compartida k, puede disponer del conjunto $\Omega' = \{\Phi(m_1), \ldots, \Phi(m_r)\}$, donde esta vez, cada elemento no está en su forma normal greedy, y ver la equivalencia de trenzas entre los elementos de Ω' y los del mensaje.

   Para ver la equivalencia puede bien seguir el algoritmo propuesto por Artin [Alg.1, pág. 34], o bien comprobar que las formas normales greedy coinciden.

Efectivamente, el método de cifrado que proponemos se trata de un cifrado por bloques o block cipher, donde, para exponer un ejemplo y considerar cálculos sencillos, hemos fijado la longitud del bloque en $n = 8$.

Este cifrado es sensible a ataques *known-plaintext*, puesto que conoce tanto los carácteres cifrados como el mismo conjunto M de los carácteres del texto plano, y mediante un análisis de frecuencia, por ejemplo, sería sencillo conocer la información transmitida.

Pongamoslo en práctica con el siguiente ejemplo. Los siguientes resultados han sido realizados mediante *SageMath*.

Ejemplo 3.2. Supongamos que Alicia y Bob quieren establecer una comunicación via Internet para intercambiarse mensajes. Para ello, mediante el criptosistema AAFG, establecen una clave compartida de la siguiente forma:

Sea $X_A = X_B = \{\sigma_1, \ldots \sigma_7\} \subset B_8$. Sea $a = \sigma_3\sigma_1\sigma_2$ la clave privada de Alicia, y sea $b = \sigma_5^{-1}\sigma_1$ la clave privada de Bob. Este le envía a Alicia $\Omega = \{b\sigma_1 b^{-1}, \ldots, b\sigma_7 b^{-1}\}$, donde:

$$\begin{aligned}
b\sigma_1 b^{-1} &= \sigma_1 \\
b\sigma_2 b^{-1} &= \Delta^{-1}\sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1\sigma_4\sigma_3\sigma_2\sigma_1\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1\sigma_6\sigma_5\sigma_4\sigma_3\sigma_2\sigma_7\sigma_6\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1\sigma_1\sigma_2 \\
b\sigma_3 b^{-1} &= \sigma_3 \\
b\sigma_4 b^{-1} &= \Delta^{-1}\sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\sigma_4\sigma_3\sigma_2\sigma_1\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1\sigma_6\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1\sigma_7\sigma_6\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1\sigma_4\sigma_5 \\
b\sigma_5 b^{-1} &= \sigma_5 \\
b\sigma_6 b^{-1} &= \Delta^{-1}\sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\sigma_4\sigma_3\sigma_2\sigma_1\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1\sigma_6\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1\sigma_7\sigma_6\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1\sigma_4\sigma_5 \\
b\sigma_7 b^{-1} &= \sigma_7
\end{aligned}$$

De esta forma Alicia sabe que como

$$\begin{aligned}
bab^{-1} &= b\ (\sigma_3\sigma_1\sigma_2)\ b^{-1} \\
&= (b\sigma_3 b^{-1})\ (b\sigma_1 b^{-1})\ (b\sigma_2 b^{-1}) \\
&= \sigma_3\sigma_1\Delta^{-1}\sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1\sigma_4\sigma_3\sigma_2\sigma_1\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1\sigma_6\sigma_5\sigma_4\sigma_3\sigma_2\sigma_7\sigma_6\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1\sigma_1\sigma_2 \\
&= \sigma_3\sigma_2^{-1}\sigma_3^{-1}\sigma_4^{-1}\sigma_5^{-1}\sigma_6^{-1}\sigma_7^{-1}\sigma_1^{-1}\sigma_7\sigma_6\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1\sigma_1\sigma_2 \\
&= \sigma_3\sigma_2^{-1}\sigma_1^{-1}\sigma_2\sigma_1\sigma_1\sigma_2
\end{aligned}$$

Y por tanto, se tiene que

$$\begin{aligned}\left(bab^{-1}a^{-1}\right)^{-1} &= \left(\left(\sigma_3\sigma_2^{-1}\sigma_1^{-1}\sigma_2\sigma_1\sigma_1\sigma_2\right)\cdot\left(\sigma_3\sigma_1\sigma_2\right)^{-1}\right)^{-1}\\ &= \left(\sigma_3\sigma_2^{-1}\sigma_1^{-1}\sigma_2\sigma_1\sigma_1\sigma_2\sigma_2^{-1}\sigma_1^{-1}\sigma_3^{-1}\right)^{-1}\\ &= \sigma_3\sigma_1^{-1}\sigma_2^{-1}\sigma_1\sigma_2\sigma_3^{-1}\end{aligned}$$

Mientras que, por otro lado:

$$\begin{aligned}aba^{-1}b^{-1} &= \left(\sigma_3\sigma_1\sigma_2\right)\left(\sigma_5^{-1}\sigma_1\right)\left(\sigma_2^{-1}\sigma_1^{-1}\sigma_3^{-1}\right)\left(\sigma_1^{-1}\sigma_5\right)\\ &= \sigma_3\sigma_1\sigma_2\sigma_5^{-1}\sigma_1\sigma_2^{-1}\sigma_1^{-1}\sigma_3^{-1}\sigma_1^{-1}\sigma_5\end{aligned}$$

Podemos verificar que ambas trenzas son la misma mediante una sencilla comprobación en sage:

```
1
2  n = 8
3  B = BraidGroup(n)
4  a = B([3,1,2])
5  b = B([-5,1])
6
7  def compact(w):
8      h = B([1])/B([1])
9      for i in w:
10         h *= i
11     return h
12
13 elem = list(map(lambda x: B([x+1]), range(n-1)))
14 elemCiph = list(map(lambda x: (b*x/b).left_normal_form(), elem))
15
16 h = ((compact(elemCiph[2]) * compact(elemCiph[0]) * compact(elemCiph[1])/a).inverse()).left_normal_form()
17 abab = (a*b/a/b).left_normal_form()
18
19 # Comprobacion
20 print(h == abab)
```

Por tanto, la clave compartida que estableceremos para la comunicación es $k = \sigma_3\sigma_1^{-1}\sigma_2^{-1}\sigma_1\sigma_2\sigma_3^{-1}$.

Ahora bien, consideremos el algoritmo de cifrado $\Phi : B_8 \longrightarrow B_8$ tal que $\Phi(b) = kbk^{-1}$, el abecedario $\Lambda = \{a, b, \ldots, z\}$ cuya representación binaria asociada viene dada por:

$$\begin{aligned}a &\equiv (01100001)_2\\ b &\equiv (01100010)_2\\ c &\equiv (01100011)_2\\ &\vdots\\ y &\equiv (01111001)_2\\ z &\equiv (01111010)_2\end{aligned}$$

De esta forma, las trenzas binarias asociadas a cada carácter, respectivamente, son las siguientes:

$$\begin{aligned}a &= \sigma_1\sigma_2\sigma_7\\ b &= \sigma_1\sigma_2\sigma_6\\ c &= \sigma_1\sigma_2\sigma_6\sigma_7\\ &\vdots\\ y &= \sigma_1\sigma_2\sigma_3\sigma_4\sigma_7\\ z &= \sigma_1\sigma_2\sigma_3\sigma_4\sigma_6\end{aligned}$$

Sea $m = hola\ mundo$ el mensaje que Alicia quiere enviar a Bob. Esta le envia, byte a byte, el mensaje; de forma que Bob recibe $\{c_i\}_i$ donde cada $c_i = \Phi(m_i) = km_ik^{-1}$.

Bob, con cada byte cifrado que recibe, como conoce k, comprueba quien es c_i con respecto los elementos de $\Omega = \{kak^{-1}, \ldots, kxk^{-1}\}$ mediante la equivalencia entre trenzas. Esto es:

Para la primera transferencia, Alicia envía

$$\begin{aligned}\Phi(h) &= \Phi(\sigma_1\sigma_2\sigma_4)\\ &= \Delta^{-2}\sigma_1\sigma_2\sigma_3\sigma_2\sigma_1\sigma_4\sigma_3\sigma_2\sigma_1\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1\sigma_6\sigma_5\sigma_4\\ &\quad \sigma_3\sigma_2\sigma_1\sigma_7\sigma_6\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1\sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1\sigma_4\sigma_3\\ &\quad \sigma_2\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1\sigma_6\sigma_5\sigma_4\sigma_3\sigma_7\sigma_6\sigma_5\sigma_1\sigma_2\sigma_3\sigma_2\sigma_1\\ &\quad \sigma_4\sigma_3\sigma_1\sigma_3\sigma_2\sigma_1\end{aligned}$$

Bob comprueba quién es $\Phi(h)$ en Ω. Se tiene que, como

$$\begin{aligned}khk^{-1} &= \left(\sigma_3\sigma_1^{-1}\sigma_2^{-1}\sigma_1\sigma_2\sigma_3^{-1}\right)\ (\sigma_1\sigma_2\sigma_4)\ \left(\sigma_3\sigma_1^{-1}\sigma_2^{-1}\sigma_1\sigma_2\sigma_3^{-1}\right)\\ &\equiv \Phi(h)\end{aligned}$$

el primer byte corresponde al carácter h. De forma sucesiva y análoga, Bob será capaz de recibir de manera integra el mensaje m de Alicia.

■

Como ejercicio práctico ideal, tras generar la clave mediante el protocolo de intercambio AAFG, se debería considerar un método de cifrado por bloques de longitud fija $n = 128$. La idea es similar: Considerar B_{128}, particionar el texto plano en su representación binaria en bloques de 128 bits, considerar las respectivas trenzas binarias y realizar el cifrado.

Capítulo 4

Ataques a la criptografía de trenzas

Severos ataques contra los esquemas criptográficos basadas en trenzas fueron propuestos en los ultimos años. Como la seguridad de tales esquemas dependen de la dificultad de conocer la clave privada, no es de sorprender que los ataques consistan principalmente en dar una solución al problema de la conjugación y sus variaciones.

En este capítulo, presentaremos una solución con un aspecto más teórico y un método heurístico en el que incluye ciertos algoritmos que han sido propuestos para la resolución del *problema de la conjugación* y el *problema de la conjugación múltiple simultánea.*

1. Solución del problema de la conjugación

El camino más obvio para atacar a los esquemas criptográficos basados en grupos de trenzas es resolver el propio problema de la conjugación. Solución la cual propuso Garside en *The braid group and other groups* [2], donde explica que el problema de la conjugación para una trenza es soluble mediante un subconjunto finito de la clase de las infinitas conjugaciones de dicha trenza, a este conjunto lo denominadas *Summit Sets*. La idea básica es como sigue:

El método de Garside consiste en asociar a cada $b \in B_n$ un conjunto finito distinguido de conjugadores de b, al cual denomina como *Summit Set*. Más aún, E.A. Elrifai y H.R. Morton proponen, en *Algorithms for positive braids* [14], remplazar, y de qué manera remplazar, dicho conjunto por un subconjunto suyo al que denominan *Super Summit Set*, el cual es más pequeño y más fácil de determinar.

Definición 4.1 (Summit Set)**.** *Sea $b \in B_n$, sea $C(b)$ la clase infinita de conjugados de b, esto es, $C(b) = \{cbc^{-1} : c \in B_n\}$. Garside nos muestra que existe un conjunto $\mathcal{SS}(b) \subset C(b)$ el cual el llama summit set de b.*

Definición 4.2 (Super Summit Set)**.** *Sea $b \in B_n$, llamamos super summit set de la trenza b y lo denotamos como $\mathcal{SSS}(b)$, al conjunto de todos los conjugados de b con la complejidad* [1] *mínima posible.*

Teorema 4.1. *Para toda trenza b, el conjunto $\mathcal{SSS}(b)$ es finito y algebraicamente computable.*

[1] La definición de complejidad se estableció tras exponer el teorema 2.4, pág. 29

Su demostración la podemos encontrar en [14].

Por construcción, dos trenzas w y v, son conjugadas si y sólo si sus $\mathcal{SSS}$ coinciden. Es decir, $\mathcal{SSS}(w)$ y $\mathcal{SSS}(v)$ o son iguales, o son disjuntos. El super summit set es calculado mediante una serie de conjugaciones especiales llamadas *cycling* y *decycling*, las cuales son fáciles de computar.

Definición 4.3. *Sea w una trenza en B_n, y sea $(r, w_1, \ldots, w_s)$ su forma normal greedy. Se definen las trenzas $\partial_+(b)$ y $\partial_-(b)$ como*

$$\partial_+(b) = \Delta_n^k w_2 \cdots w_s \phi_n^k(w_1) \qquad \partial_-(b) = \Delta_n^k \phi_n^k(w_s) w_1 \cdots w_{s-1}$$

donde ϕ_n es el automorfismo que envía σ_i a σ_{n-i}, para cada i. Decimos que $\partial_+(b)$ (resp. $\partial_-(b)$) es obtenida haciendo cycling (resp. decycling) a b. Se tiene que, por construcción, las trenzas $\partial_+(b)$ y $\partial_-(b)$ son conjugadas de b.

El punto está en que, si b es una trenza en B_n que no está en su $\mathcal{SSS}(b)$, es decir, que no tiene la minima complejidad en su clase de conjugación; entonces haciendo cycling o decycling a lo sumo $n(n-1)/2$ veces, uno puede buscar un conjugado de b con una complejidad estrictamente más pequeña. Así, repitiendo la operación, obtenemos tras un número finito de pasos un conjugado b^* de b que está en $\mathcal{SSS}(b)$.

El siguiente resultado es que, si b está en su $\mathcal{SSS}(b)$, entonces todo el conjunto $\mathcal{SSS}(b)$ puede ser obtenido saturando $\{b\}$ mediante conjugaciones de trenzas simples:

Empezariamos con $b \in \mathcal{SSS}(b)$, de forma sucesiva consideraremos todas los conjugados sbs^{-1} donde s es una trenza simple. Guardaremos esos conjugados b' que tienen la misma complejidad que b y desecharemos el resto, hasta no tener mas conjugados que añadir.

Por tanto, un procedimiento completo para decidir si dos trenzas b, b' son conjugadas es:

1. Usando cycling y decycling, encontrar un conjugado b^* de b estando en $\mathcal{SSS}(b)$.
2. Usando cycling y decycling, encontrar un conjugado b'^* de b' estando en $\mathcal{SSS}(b')$.
3. Determinar $\mathcal{SSS}(b)$ saturando $\{b^*\}$ bajo conjugaciones simples.
4. Entonces, b y b' son conjugados si y sólo si b'^* está en $\mathcal{SSS}(b)$

Al realizar un seguimiento de las trenzas conjugadas usadas en cada paso, uno no sólo es capaz de decidir si b y b' son conjugadas, sino que además se obtiene un conjugador, en el caso de existir, es decir en el caso de que b y b' sean conjugadas. Entonces, mediante este proceso, uno puede resolver tanto el *problema de la posibilidad de la conjugación* y el *problema de la conjugación*.

En *Braid-based cryptography* [8], Patrick Dehornoy define un refinamiento del super summit set al cual denomina *ultra summit set* $\mathcal{USS}$. Y procede dando un resultado más eficiente para realizar un ataque al problema de la conjugación similar al recien expuesto.

2. Ataque heurístico al problema de la conjugación

Dado dos trenzas conjugadas $v, w \in B_n$, nos gustaria encontrar una trenza conjugada $\alpha \in B_n$ tal que $w = \alpha^{-1}v\alpha$.

Para ello, y siguiendo el artículo *A practical attack on some braid group based cryptographic primitives* [5], expondremos un algoritmo experimental que, de forma heurística, será capaz de encontrar ese α, que en el contexto de la criptografía y más concretamente del criptosistema AAFG[2], no es más que la clave compartida, que al fin y al cabo es el objeto de nuestro estudio como posible intruso o interceptor.

Por conveniencia, introduciremos la noción de cola. Sea $\gamma \in B_n^+$ se dice que es una cola de alguna trenza $b \in B_n^+$ si y sólo si, existe ua factorización $b = \beta\gamma$ con $\beta \in B_n^+$. Además, definimos el automorfismo τ donde dado una trenza $w \in B_n$, se tiene que $\tau(w) = \Delta^{-1}w\Delta$. Con esto, podemos establecer el siguiente lema[3]:

Lema 4.1. *Sea $v, w \in B_n$ dos trenzas positivas conjugadas entre si, esto es, que existe $\alpha \in B_n^+$ tal que $w = \alpha^{-1}v\alpha$. Sea $\Delta^r w_1 \cdots w_s$ la forma normal greedy de w. Entonces se mantienen las siguientes relaciones:*

- *Si $\inf w < \inf v$, entonces el factor canónico $\Delta\tau^r(w_1^{-1})$ es una cola de α.*
- *Si $\sup w > \sup v$, entonces w_s es una cola de α.*

Demostración. La demostración la podemos encontrar en el *Lema* 4.3 del artículo de E.A. Elrifai y H.R. Morton, *Algorithms for positive braids* [14]. ■

Así, mediante el *Lema* 4.1, consideremos el algoritmo 5, el cual nos ayudará a abordar el problema de la conjugación en B_n. Este algoritmo pone en juicio la seguridad del sistema de criptografía basado en trenzas.

Comportamiento del algoritmo 5

Veamos como se comporta el algoritmo. La idea es buscar, o más bien adivinar, el conjugador de v.

En los pasos 1 y 2, consideramos α que será el elemento conjugador que inicie el algoritmo, y consideraremos $v, w \in B_n$ en sus formas normales. En el paso 3 y 4 conjugaremos w de una forma que, tras acabar el paso, la longitud canónica de w no distará tanto con la longitud canónica de v como antes. Usando el Lema 4.1, al comenzar el paso 5 tendremos de esta forma $w = x'^{-1}vx'$ con $x = x'\alpha$ y $x' \in B_n^+$.

Asumimos llegados a esta situación, la parte no descubierta x' del conjugador x está esencialmente determinada por su permutación inducida $\pi(x')$. Por lo que, hacemos una llamada a la función *gpermutation*, la cual nos permitirá *adivinar* la permutación $\pi(x')$ y devolvernos la trenza simple o factor canónico μ asociada a ella, tal que $\pi(\mu) = \phi(x')$.

[2]Criptosistema Anshel-Anshel-Fisher-Goldfeld, página 39
[3]Definición de $inf\, w$ y $sup\, w$ en la página 29

Algorithm 5: Algoritmo de ataque heurístico al problema de la conjugación

Input :
$v, w \in B_n$ con $w = x^{-1}vx$ para alguna trenza $x \in B_n^+$ desconocida con $inf\, x = 0$.
Output:
$\alpha \in B_n^+$ con $w = \alpha^{-1}v\alpha$, en caso de éxito. *Fracaso*, en caso de fracasar.

$\alpha := e$
$v := \Delta^{r_v} v_1 \cdots v_{s_v}$
$w := \Delta^{r_w} w_1 \cdots w_{s_w}$
while $inf\, w < inf\, v$ **do**
  $\gamma := \Delta\tau^{r_w}(w_1^{-1})$
  $\alpha := \gamma\alpha$
  $w := \gamma w \gamma^{-1}$ y lo normalizamos.
end
while $sup\, w > sup\, v$ **do**
  $\gamma := w_{s_w}$
  $\alpha := \gamma\alpha$
  $w := \gamma w \gamma^{-1}$ y lo normalizamos.
end
$\mu := gpermutation(w, v)$
$\alpha := \mu\alpha$
$w := \mu w \mu^{-1}$
if $v = w$ **then**
  return α
end
else
  return *Fracaso*
end

Esta función la describiremos a continuación[4], sin embargo antes de eso, debemos disponer de los siguientes resultados:

Definición 4.4 (Trenza de permutación)**.** *Sea f una permutación en $\mathfrak{S}_n$. Podemos definir de forma recursiva una trenza positiva cuya permutación es f como sigue. Consideramos $br(1) = e$ y*

$$br(f) = \beta_{f(k),k}\, br(g)$$

donde $k = \max\{s \mid s \in \{1, \ldots, n\} \text{ y } s \neq f(s)\}$ y $g \in \mathfrak{S}_n$ es definido como:

$$g(i) = \begin{cases} f(i) & \text{cuando } i < k \text{ y } f(i) < f(k) \\ f(i) - 1 & \text{cuando } i < k \text{ y } f(i) > f(k) \\ f(i) & \text{cuando } i \leq k \end{cases}$$

A esta trenza la denominamos *trenza de permutación.*

Teorema 4.2. *Una trenza es simple si y sólo si es una trenza de permutación.*

Demostración. La demostración la podemos encontrar en *Combinatorial aspects of braids with applications to cryptography* [4] de M. Bennett, *Teorema* 4.21. ■

De esta forma podemos, dado la proyección canónica $\pi : B_n \longrightarrow \mathfrak{S}_n$, que envía cada trenza $b \in B_n$ a la permutación correspondiente asociada $s \in \mathfrak{S}_n$, considerar un método basandonos en la *definición* 4.4 que nos devuelva la imagen inversa $\pi^{-1}(f) \in B_n^+$, donde $f \in \mathfrak{S}_n$.

Función. braidPermutation()

Sea $p \in \mathfrak{S}_n$ una permutación expresada como una lista de enteros. Por ejemplo, a la permutación $(1743)(26)(5)$ le corresponde la expresión $[7, 6, 1, 3, 5, 2, 4]$.

```
from functools import *

def braidPermutation(p):
    B = BraidGroup(max(p))
    xs,n = list(map(lambda x: x-1, p)), len(p)
    trenza = []
    while xs != list(range(n)):
        k= max(list(filter(lambda i: xs[i]!= i, range(n))))
        fk , ys= xs[k], xs
        trenza += [[fk+1,k+1]]
        for i in range(n):
            if i < k and xs[i] < xs[k]:
                ys[i] = xs[i]
            elif i < k and xs[i] > xs[k]:
                ys[i] = xs[i] -1
            else:
                ys[i] = i
        xs = ys
    if trenza != []:
        braidPermutation = list(reduce((lambda a,b: a+b), list(map(lambda x: list(range(x[0],x[1])), trenza
    ))))
        braid = B(braidPermutation)
        return braid
    else:
        return B([1])/B([1])
```

De este modo, mediante el método construido, a la permutación $(1743)(26)(5)$ le corresponde la trenza

$$b = \sigma_4\sigma_5\sigma_6\sigma_2\sigma_3\sigma_4\sigma_5\sigma_3\sigma_4\sigma_2\sigma_3\sigma_1\sigma_2\sigma_1$$

[4]Función *gpermutation*(), página 53

Y por tanto, el algoritmo de la función *gpermutation* es el siguiente:

Función. gpermutation()

Dado $v, w \in B_n$, con $w = x^{-1}vx$ para algun $x \in B_n^+$ con $inf\ x = 0$. Se desea encontrar un factor canónico $\mu \in B_n^+$ tal que $\pi(\mu) = \pi(x)$.

Algorithm 6: Algoritmo de gpermutation

Input :
$v, w \in B_n$ con $w = x^{-1}vx$ para alguna trenza $x \in B_n^+$ desconocida con $inf x = 0$.
Output:

Un factor canónico $\mu \in B_n^+$, tal que $\pi(\mu) = \pi(x)$.

Sea $\tau \in \mathfrak{S}_n$ la permutación identidad
$(\chi_i, \ldots, \chi_n) := (False, \ldots, False)$
for i desde n hacia 1 **do**
  $r := i$, $s := i$
  while $\chi_r = False$ **do**
    $_r := True$
    $r := \pi(v)(r)$, $s := \pi(w)(s)$
    if $r \neq s$ **then**
      $\tau(r) := s$
    end
  end
end
$(\chi_i, \ldots, \chi_n) := (False, \ldots, False)$
for i desde n hacia 1 **do**
  if $\chi_i = False$ **then**
    $\chi_i := True$, $r := i$
    while $\chi_{\tau(r)} = False$ y $\tau(r) \neq r$ **do**
      $r := \tau(r)$, $\chi(r) := True$
    end
    $tau(r) := i$
  end
end
return $\pi^{-1}(\tau)$

Donde π^{-1} es la función antes definida *braidPermutation*().

Volviendo al algoritmo 5, llegados al paso 6 devolvemos α en caso de que sea el conjugador que estabamos buscando. Téngase en cuenta que ni el algoritmo 5 ni la función *gpermutación*() son probabilísticos, por lo que no hay ninguna posibilidad de que, una vez que uno de ellos falle, las aplicaciones sucesivas puedan mejorar este resultado.

Ahora si, el script de *SageMath* correspondiente a la implementación del algoritmo 5

propuesto en [5] es el siguiente:

```
from functools import *
load("gpermutation.sage")
n = #Parametro a determinar
B = BraidGroup(n)

def tau(w): #Funcin tau
    return B.delta()*w*B.delta().inverse()

## Dado que Sage devuelve la left_normal_form()
## en una tupla, esta funcion hace que
## la compacte en una unica expresion como trenza
def compact(w):
    h = B([1])/B([1])
    for i in w:
        h *= i
    return h

# Algoritmo de ataque heursicto
def heuristicAttack(w,v):
    alpha, delta = B([1])/B([1]), B.delta().inverse()
    v, w = v.left_normal_form(), w.left_normal_form()
    rv = -list(filter(lambda x: v[0]==delta**x, range(n)))[0]
    rw = -list(filter(lambda x: w[0]==delta**x, range(n)))[0]
    while rw < rv:
        gamma = tau(w[1].inverse())
        for i in range(rw-1):
            gamma = tau(gamma)
        gamma = B.delta()* gamma
        alpha = gamma * alpha
        w = (gamma * compact(w) / gamma).left_normal_form()
        rw = -list(filter(lambda x: w[0]==delta**x, range(n)))[0]
    supw, supv = len(w[1:]) + rw, len(v[1:]) + rv
    while supw > supv:
        gamma = w[-1]
        alpha = gamma * alpha
        w = (gamma * compact(w) / gamma).left_normal_form()
        rw = -list(filter(lambda x: w[0]==delta**x, range(n)))[0]
        supw = len(w[1:]) + rw
    mu = gpermutation(compact(w),compact(v))
    alpha = mu * alpha
    w = (mu * compact(w) / mu).left_normal_form()
    if v == w:
        return alpha
    else:
        return "Fracaso"
```

3. Ataque al problema de la conjugación múltiple simultánea

El algoritmo 5 puede modificarse para que funcione en caso de que se de el *problema de la conjugación múltiple simultánea*.

Dado $m \geq 1$ palabras $v_i, w_i \in B_n$ con $w_i = x^{-1}v_i x$ para cada $1 \leq i \leq m$ y para algún $x \in B_n$ desconocido, buscamos algun $\alpha \in B_n$ tal que satisfaga $w_i = \alpha^{-1}v_i\alpha$ para todo i. Consideramos el siguiente algoritmo, que no es más que una modificación del algoritmo 5 visto en la página 51.

En este caso, la función *gsimultpermutation*(), es una modificación de la *gpermutation*(), del cual no se detalla explicitamente en ningún momento a lo largo del artículo [5], y que toma parejas de trenzas conjugadas $(v_i, w_i) \in B_n^2$ tales que $w_i = x^{-1}v_i x$ para todo

Algorithm 7: Algoritmo de ataque heurístico al problema de la conjugación múltiple simultánea

Input :
Sean $m \geq 1$ pares de trenzas, $(v_i, w_i) \in B_n^2$ con $w_i = x^{-1}v_i x$ para alguna trenza $x \in B_n^+$ desconocida con $infx = 0$.

Output:

$\alpha \in B_n^+$ con $w_i = \alpha^{-1}v_i\alpha$ para todo i, en caso de éxito. *Fracaso*, en caso de fracasar.

Consideramos $\alpha := e$
$v_i := \Delta^{r_{v_i}} v_1^{(i)} \cdots v_{s_v}^{(i)}$
$w^{(i)} := \Delta^{r_{w_i}} w_1^{(i)} \cdots w_{s_w}^{(i)}$

while $inf\, w_j < inf\, v_j$ para algún $j \in \{1, \ldots, m\}$ **do**
  $\gamma := \Delta\tau^{r_{w_j}} \left(\left(w_1^{(j)} \right)^{-1} \right)$
  $\alpha := \gamma\alpha$
  $w_i := \gamma w_i \gamma^{-1}$ para todo $i \in \{1, \ldots, m\}$
  Normalizamos todas las w_i
end

while $sup\ w_j > sup\ v_j$ *para algún* $j \in \{1, \ldots, m\}$ **do**
  $\gamma := w_{s_w}^{(j)}$
  $\alpha := \gamma\alpha$
  $w_i := \gamma w_i \gamma^{-1}$ para todo $i \in \{1, \ldots, m\}$
  Normalizamos todas las w_i
end
$\mu := gsimultpermutation\,((w_1, v_1), \ldots, (w_m, v_m))$
$\alpha := \mu\alpha$
$w_i := \mu w_i \mu^{-1}$ para todo $i \in \{1, \ldots, m\}$
if $v_i = w_i$ *para todo* i **then**
  return α
end
else
  return *Fracaso*
end

i y da como resultado un supuesto factor canónico β tal que $\pi(\beta) = \pi(x)$.

Pruebas de ataques

A continuación se expondrán ciertos resultados, publicados en [5], que han sido realizados en pruebas de ataque siguiendo el algoritmo 7 en sistema de intercambio de claves AAFG y al sistema de Ko et al.

Ataque contra AAFG

Sea n el orden del grupo de trenzas B_n, y sea $G_A, G_B \leq B_n$ con $r = s$ generadores cada uno. Consideramos que, para cada generador a_i (resp. b_i) del subgrupo G_A (resp. G_B), está formado por $5 \leq l \leq 10$ generadores de Artin. Y sea a (resp. b) la clave privada de Alicia (resp. Bob), compuesta por k generadores de G_A (resp G_B). Se contempla realizar un estudio de tiempos en pruebas de generación de claves.

Los resultados mostrados, expuestos en el artículo [5], siguiendo el algoritmo 7 para atacar el *problema de la conjugación múltiple simultánea* para los parámetros usados a continuación, son:

n	r = s	k	Número de pruebas	Ratio de éxito
80	20	5	1000	99.0 %
80	20	10	1000	98.9 %

Ataque contra Ko et al.

Para las pruebas, se ha considerdo $n \geq 45$ y elegido $l = n/2$ y $r = n - l$. Para x y a (resp. b) se han usado trenzas con longitud canónica $p \geq 3$. La trenza que se requiere x debe ser *suficientemente complicada*. Sin embargo, no entra en detalle cómo se considera una trenza *suficientemente complicada*, simplemente se considera una trenza $x \in B_n$ generada de manera aleatoria.

En este experimento, como intruso, conseguimos interceptar los elementos de la forma $y = axa^{-1}$. A continuación se presentan los resultados de los experimentos realizados para algunos de los parámetros propuestos n y p, donde p denota la longitud canónica tanto de x como de a (resp. b).

n	p	Número de pruebas	Ratio de éxito
45	3	1000	78.1 %
50	5	1000	79.1 %
70	7	1000	79.0 %
90	12	1000	80.0 %

Tanto los resultados de las pruebas de ataques contra AAFG como estas últimas, aun siendo publicados, no han sido probados en este estudio.

4. Propuesta de ataque

Si bien es cierto, mencionamos que dicho algoritmo, el expuesto en [5], es experimental y, además, faltan algunos detalles tanto en su construcción y como en su utilización con respecto los resultados publicados. Ante esto, he considerado diseñar y proponer el siguiente método, el cual ha dado buenos resultados:

```
from functools import *

#######################################################
## Algoritmo:
## Ataque contra el problema de la conjugacin
## ==========================================
## Autor: Rafael Fernandez Ortiz
## Fecha: 16/Junio/2019
#######################################################

## Dado que Sage devuelve la left_normal_form()
## en una tupla, esta funcion hace que
## la compacte en una unica expresion como trenza
def compact(w):
    h = B([1])/B([1])
    for i in w:
        h *= i
    return h
#######################################################
## Parametros iniciales
## ====================
# n = # A determinar
# B = BraidGroup(n)
# v = # Trenza a determinar
# x = # Trena a determinar
# w = compact((x*v/x).left_normal_form())
#######################################################

## Dada una permutacion, te devuelve su trenza asociada
def braidPermutation(p):
    B = BraidGroup(max(p))
    xs,n = list(map(lambda x: x-1, p)), len(p)
    trenza = []
    while xs != list(range(n)):
        k= max(list(filter(lambda i: xs[i]!= i, range(n))))
        fk , ys= xs[k], xs
        trenza += [[fk+1,k+1]]
        for i in range(n):
            if i < k and xs[i] < xs[k]:
                ys[i] = xs[i]
            elif i < k and xs[i] > xs[k]:
                ys[i] = xs[i] -1
            else:
                ys[i] = i
        xs = ys
    if trenza != []:
        braidPermutation = list(reduce((lambda a,b: a+b), list(map(lambda x: list(range(x[0],x[1])), trenza
    ))))
        braid = B(braidPermutation)
        return braid
    else:
        return B([1])/B([1])

def tau(c1,c2):
    t = [None]*n
    c1, c2 = [x-1 for x in c1], [x-1 for x in c2]
    for i in range(len(c1)):
        t[c1[i]] = c2[i]+1
    return t

## Algoritmo para estimar las posibles trenzas conjugadoras
def guessBraid(w,v):
    t = [None]*n
```

```
    pv = v.permutation().cycle_tuples()
    pw = w.permutation().cycle_tuples()
    xs = []
    for i in pv:
        xs += [(i,map(lambda a: tau(i,a),filter(lambda x: len(i)==len(x), pw)))]

    ## Metodo para estimar todas las permutaciones posibles
    def guessPermutation(ciclos=xs):
        m = max([len(x[0]) for x in ciclos])

        def permJoin(ls, ds):
            ks = ls[:]
            for i in range(len(ks)):
                if ks[i] is None:
                    ks[i] = ds[i]
            return (ks)

        def shuffle(m, desc=ciclos):
            ind = [x[0] for x in (filter(lambda x: len(x[0]) == m, desc))]
            ps = [x[1] for x in (filter(lambda x: len(x[0]) == m, desc))]
            t, b = [], len(ind)
            for i in range(b):
                d = [None] * n
                for x in ps:
                    d = permJoin(d, x[i])
                    i = (i + 1) % b
                t += [d]
            return t

        partesPerm = list(filter(lambda x: x != [], [shuffle(k) for k in range(m, 0, -1)]))
        permutaciones = partesPerm[0]
        for t in partesPerm[1:]:
            permutaciones = [permJoin(x, y) for x in permutaciones for y in t]
        return permutaciones

    return guessPermutation()

## Realiza el ataque devolviendote la trenza conjugadora en caso exito
## Devuelve None en caso de fracaso
def attack(a,b):
    trenzasPosibles = [braidPermutation(x) for x in guessBraid(a,b)]
    for x in trenzasPosibles:
        if a.left_normal_form()==(x*b/x).left_normal_form():
            return x
```

De nuevo, si consideramos el siguiente escenario:

Sea n el orden del grupo de trenzas B_n, sean $m \geq 1$ pares de trenzas, $(v_i, w_i) \in B_n^2$ con $w_i = x^{-1}v_i x$ para alguna trenza $x \in B_n^+$ desconocida. Donde v_i está formado por un número aleatorio comprendido entre $1 \leq k \leq p$, con $p \in \mathbb{N}$, de trenzas elementales, es decir, $v_i = \sigma_{\alpha(1)} \cdots \sigma_{\alpha(k)}$ con $\alpha : \{1, \dots k\} \longrightarrow \{1, \dots, n-1\}$ la función que devuelve un numero aleatorio.

Se pretende, mediante el siguiente experimento, encontrar el conjugante x. Para algún parámetro concreto, se ha debido reducir el número de pruebas debido al exceso de tiempo de computación.

Los resultados de nuestro experimento de ataque frente al *problema de la conjugación*, es decir, cuando $m = 1$, en un ordenador con SO Linux y procesador de 8 núcleos a 1.60 GHz cada uno, son los siguientes:

n	p	Número de pruebas	Ratio de éxito	Tiempo de experimento (seg)	Tiempo medio de ataque (seg)
5	5	1000	48.9 %	17.550032	0.01755
5	10	1000	24.2 %	37.570034	0.03757
10	5	1000	73.1 %	275.9162659	0.2759162659
10	10	100	39.0 %	61.19964	0.6119964
30	5	20	95.0 %	655.522619	32.77613

Mientras que, de igual forma y para los mismos parámetros, en nuestro experimento de ataque frente al *problema de la conjugación múltiple simultánea*, es decir para $m > 1$, se pretende encontrar el conjugante x tal que $w_i = xv_ix^{-1}$ para todo $i \in \{1, \ldots, m\}$.

Los resultados de los experimentos, fijando $m = 10$, en un ordenador con SO Linux y procesador de 8 núcleos a 1.60 GHz cada uno, son los siguientes:

n	p	Número de pruebas	Ratio de éxito	Tiempo de experimento (seg)	Tiempo medio de ataque (seg)
5	5	1000	91.8 %	160.674895	0.16067
5	10	1000	62.5 %	357.01656	0.35701
10	5	100	100 %	320.810438	3.2081
10	10	100	79.0 %	665.586119	6.65586
30	5	10	100 %	43788.459	437.88459

El script utilizado para los distintos experimentos es *pruebas_attack.sage*, dónde m es el parámetro *numvi* y p es el parámetro *numGeneradores*.

Conclusión

Los resultados mostrados en ataques al *problema de la conjugación múltiple simultánea*, muestran porcentajes de éxitos mayores que los realizados al *problema de la conjugación*.

Esto es debido a que estimar quién es el conjugante para una sóla trenza tiene su porcentaje de éxito como hemos podido comprobar, sin embargo si se diera el caso de fracaso, el ataque finalizaría justo ahí.

Para el caso de la *conjugación múltiple simultánea* se tiene un número $m > 1$ de oportunidades para seguir atacando, y por tanto, seguir estimando el conjugante. Así es, el *problema de la conjución múltiple simultánea* da mayor información a mayor número de trenzas que se conjugue.

Por otro lado, podemos ver reflejado en los resultados como el ratio de éxito es mayor cuando se fija p el número de generadores que forman la trenza y varía el orden n que, por el contrario, si fijases el orden n y se variase el número de generadores que forman la trenza. Efectivamente, tal y como mencionamos en la página 32, esto es debido a que el costo computacional que presenta la generación de la palabra fundamental sigue la función $O(|w|^2 n \, log \, n)$.

Así que, a pesar de todo, se debería ser consciente de la *fragilidad* que presenta estos sistemas criptográficos basados en trenzas pues, como hemos podido comprobar, sólo requiere tiempo para destruir la aparente seguridad que proporciona.

Bibliografía

[1] C. Kassel y V. Tuarev, *Braid Groups*. Editorial Board, 2008.

[2] F.A. Garside, *The braid group and other groups*. 1969.

[3] J.B. Fraleigh, *Álgebra abstracta*. Editorial Addison-Wesley. 1982.

[4] M. Bennett, *Combinatorial aspects of braids with applications to cryptography*. Universidad de matemáticas de Waterloo, Ontario, Canada. 2015.

[5] D. Hofheinz y R. Steinwandt, *A practical attack on some braid group based cryptographic primitives*. EA Practical Attack on some Braid Groups Cryptographic Primitivesditorial Desmedt. 2003.

[6] M.A. Marco Buzunariz, V. Braun, S.F. Jørgensen, R.Lipshitz, T. Monteil y S.Oehms, *Documentación de SageMath, Braid Groups*. `http://doc.sagemath.org/html/en/reference/groups/sage/groups/braid.html`

[7] D. Garber, *Braid group cryptography*. Departamento de matemáticas aplicadas del instituto de tecnologías de Holon, Israel. 2008.

[8] P. Dehornoy, *Braid-based cryptography*. 2000.

[9] H.Delfs y H. Knebl *Introduction to Cryptography. Principles and Applications*. Editorial Springer. 2007

[10] Moni Naor y Moti Yung, *Publick-key cryptosystems provably secure against chosen ciphertext attacks*. 1990.

[11] M.I. Gónzalez y R. Steinwandt, *Group Theoretic Cryptography*. Editorial Chapman & Hall. 2015.

[12] E. Artin, *Theory of braids*. 1947.

[13] J. Stillwell, *The word problem and the isomorphism for groups*. 1982.

[14] E.A. Elrifai y H.R. Morton, *Algorithms for positive braids*. Departamento de matemáticas del King Khalid univerity, Abha, Arabia Saudi. Departamento de matemáticas de la universidad de Liverpool, Peach St, Liverpool, Inglaterra.

[15] Ki Hyoung Ko, Sang Jin Lee, Jung Hee Cheon, Jae Woo Han, Ju-sung Kang y Choonsik Park, *New public-key cryptosystem using braid groups*. Departamento de matemáticas del instituto de ciencias y tecnologías, Taejon, Korea. Departamento de matemáticas del Brown university, Providence, EEUU. 2000.

[16] Jae Choon Cha,Ki Hyoung Ko, Sang Jin Lee, Jae Woo Han, Jung Hee Cheon, *An efficient implementation of braid groups*. Departamento de matemáticas del instituto de ciencias y tecnologías, Taejon, Korea. 2001.

[17] J.L. Chacón, *Permutaciones y grupo simétrico*. Universidad de los Andes, Venezuela.