UNIVERSITÉ DU QUÉBEC À MONTRÉAL

CRYPTOGRAPHIE ET GROUPES DE TRESSES

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN MATHÉMATIQUES

PAR

DANIEL GAGNON

FÉVRIER 2007

REMERCIEMENTS

La réalisation de ce mémoire n'aurait pas été possible sans la précieuse collaboration de mon directeur, Mr. François Bergeron. Je dois également souligner l'aide apportée par Mme Manon Gauthier ainsi que Mme Lise Tourigny.

J'aimerais aussi remercier mes collègues étudiants et enseignants, entre autres Geneviève Paquin, Marie-Ève Provost-Larose, Francois Hotte et Cédric Lamathe, pour leur soutien, les bons conseils et la belle ambiance de travail. Pour terminer merci au LACIM et au département de mathématiques de l'UQÀM pour l'ensemble de leur apport durant mon séjour à la maîtrise.

TABLE DES MATIÈRES

LISTE DES FIGURES

RÉSUMÉ

Nous abordons dans ce travail, l'utilisation de groupes algébriques dans le domaine de la cryptographie. Nous étudions un protocole d'échange de clés (I. Anshel, 2001) qui utilise le groupe de tresses B_n et plus particulièrement le problème de conjugaison dans ce groupe. Nous voyons également comment il est possible de construire une attaque sur ce cryptosystème en tentant de résoudre le problème de conjugaison dans B_n à l'aide d'une représentation du groupe de tresses, la représentation de Burau.

INTRODUCTION

La cryptographie permet de sécuriser de l'information circulant entre deux personnes et qui est susceptible d'être interceptée. La sécurité d'un protocole cryptographique est généralement basée sur un probléme mathématique. Il existe plusieurs protocoles cryptographiques, mais le développement rapide de nouvelles technologies pousse les chercheurs à utiliser de nouveaux outils mathématiques pour construire des protocoles toujours plus sécuritaires.

Depuis quelques annés, on s'interesse à l'utilisation de groupes algébriques en cryptographie. Des protocoles ont étés proposés avec entre autres les groupes de tresses B_n. Pour l'instant, ces groupes ne sont pas encore utilisés puisque leur efficacité n'est pas certaine. Nous étudierons dans ce travail, l'utilisation du groupe de tresses B_n dans le cadre d'un protocole cryptographique. Il existe plusieurs problèmes difficiles à résoudre dans le groupe B_n, quelques-uns d'entre eux impliquent la conjugaison. Nous nous concentrerons sur un problème en particulier, qui est celui de recherche du conjugueur. Le problème de recherche du conjugueur est à la base de la sécurité d'un protocole d'échange de clés, souvent appellé protocole AAFG1, introduit en 2001. Nous verrons comment, en tentant de résoudre le problème de recherche du conjugueur à l'aide de la théorie de la représentation des groupes, nous pouvons mettre en jeu la sécurité de ce système cryptographique.

CHAPITRE I

PROBLÉMATIQUE DE LA CRYPTOGRAPHIE

Introduction

L'envoi de messages entre correspondants s'effectue nécessairement par le biais d'un canal pour y faire circuler le dit message. Or, très souvent, le canal utilisé est non-sécuritaire, c'est-à-dire qu'un tiers peut lire les messages y circulant. On désire donc coder les messages pour en assurer la confidentialité. La *cryptographie* est l'étude des techniques permettant de coder (*chiffrer*) des messages, c'est-à-dire de les rendre inintelligibles sans une action spécifique.

La cryptographie est une discipline très ancienne, dont les traces dans l'histoire remontent jusqu'à l'antiquité. On sait que les Grecs utilisaient des méthodes pour chiffrer des messages dès le VI ième siècle avant Jésus-Christ. Pendant la plus grande partie de son histoire, la cryptographie a surtout été utilisée à des fins militaires et diplomatiques, et par des groupes assez restreints. Ce n'est environ que depuis trente ans que son utilisation s'est répandue de façon plus générale. La plus grande partie de la littérature portant sur ce sujet est donc assez récente et la recherche y est très active puisque la problématique soulevée est loin d'être simplement résolue.

Cryptosystème

Une méthode cryptographique, et l'ensemble des éléments permettant de l'appliquer, est appellé un *cryptosystème*. Un cryptosystème doit assurer trois propriétés des messages :

1. L'*intégrité* : Le message clair envoyé par l'émetteur doit être identique au message décrypté par le récepteur.
2. La *confidentialité* : On veut qu'une personne interceptant les cryptogrammes ne puisse pas les déchiffrer.
3. L'*authentification* : Le récepteur doit pouvoir déterminer avec certitude que le message provient bien de l'émetteur voulu.

Bien qu'un cryptosystème comprenne, entre autres, deux alphabets : l'alphabet utilisé pour construire les *messages non chiffrés* (*messages clairs*) que nous notons $\mathcal{A}$, et l'alphabet utilisé pour construire les *messages chiffrés* (*cryptogrammes*), que nous notons $\mathcal{B}$. Ces deux alphabets sont souvent égaux.

Le *chiffrement* et le *déchiffrement* se font à l'aide d'une clé, qui prend la forme d'une fonction mathématique. Pour le chiffrement, la fonction d'encodage E envoie un mot sur l'alphabet $\mathcal{A}$ vers un mot sur l'alphabet $\mathcal{B}$:

$$E : \mathcal{A}^* \to \mathcal{B}^*$$

Le déchiffrement se fait à l'aide d'une autre fonction D qui envoie un mot sur l'alphabet $\mathcal{B}$ vers un mot sur l'alphabet $\mathcal{A}$, cette fonction est en fait l'inverse de la fonction de chiffrement E.

Pour l'utilisation d'un cryptosystème, on suppose qu'il y a deux correspondants désirant communiquer l'un avec l'autre : l'*émetteur*, qui désire envoyer le message, et le *récepteur* qui reçoit le message. Pour faciliter l'explication des exemples et des protocoles, dans le texte, nous utiliserons les termes fréquemment utilisés dans la littérature. Ainsi, pour désigner deux personnes communiquant entre elles sur un canal (sécuritaire ou non) on utilise les noms d'*Alice* et de *Bob*.

Il est naturel de distinguer deux types de chiffrements : les *chiffrements à clés publiques* et les *chiffrements à clés privées*. Ces deux types de chiffrement ont chacun leurs avantages et leurs inconvénients qui les rendent utiles et complémentaires. Nous verrons dans les prochaines sections les propriétés de ces deux approches.

1.1 Cryptographie à clé privée

Dans la *cryptographie à clé privée*, aussi appellée *cryptographie symétrique*, on utilise une même clé pour chiffrer et déchiffrer. La clé utilisée est secrète et connue seulement des deux parties qui veulent échanger de l'information. Cette façon de fonctionner suppose que les deux parties s'entendent initialement sur cette clé secrète, il est alors primordial d'échanger cette clé par un canal sécuritaire ou, comme nous le verrons ultérieurement par un canal non-sécuritaire en utilisant un protocole d'échange de clé qui en conserve le secret.

1.2 Cryptographie à clé publique

La *cryptographie à clé publique*, ou *cryptographie asymétrique*, utilise des paires de clés : une privée et une publique. Dans ce modèle, chaque utilisateur du cryptosystème détient une paire distincte. L'idée générale est que, pour chaque paire de clés, seule la clé privée permet de déchiffrer un message qui a été chiffré avec la clé publique.

Fonctions à sens unique

La réalisation de systèmes à clé publique passe par l'utilisation de fonctions dites *à sens unique*. On entend par là que ce sont des fonctions facilement calculables, mais dont l'inverse est difficilement calculable. Cette notion de calcul difficile ou facile est pour l'instant informelle et sera clarifiée ultérieurement. Pour certaines fonctions à sens unique, il existe souvent une information, conservée secrète, qui permet de facilement calculer l'inverse. Ces fonctions sont appellées *fonctions à sens unique et à brèche secrète.* À titre d'exemple, on peut citer le produit de deux nombres premiers comme étant une fonction à sens unique. Le produit de deux nombres premiers se calcule facilement, mais le calcul inverse, qui consiste à factoriser le nombre en produit de deux nombres premiers, est difficilement calculable.

C'est une fonction à sens unique et à brèche secrète qui est à la base de la sécurité d'un

système à clé publique. Par exemple, si Alice veut coder un message avant de l'envoyer à Bob, elle utilise la clé publique f_B de Bob, qui est une telle fonction. La clé secrète de Bob est la fonction f_B^{-1}, qui lui permettra de déchiffrer le message d'Alice.

Pour clarifier la notion de *fonction facilement calculable*, on considère que les calculs sont effectués à l'aide d'ordinateurs. Concrètement la facilité de calcul se traduit par le temps nécessaire au calcul. L'utilisation de fonctions à sens unique en cryptographie relève de l'intention de vouloir rapidement crypter un message tout en empêchant un décryptage rapide de la version codée du message.

Plus techniquement, la facilité de calcul se mesure par la complexité algorithmique de ce calcul. Ainsi, une fonction facilement calculable aurait typiquement une complexité en $O(n)$ tandis qu'une fonction difficilement calculable aurait par exemple une complexité en $O(a^n)$, avec $a > 1$.

Fonctions de hachage

Une *fonction de hachage* est une fonction à sens unique qui a comme domaine des nombres de longueur variable mais dont les nombres de l'image de la fonction sont de longueur fixe et inférieure à ceux du domaine. Une fonction de hachage idéale contient le moins possible de collisions (deux nombres ayant la même image). Le but d'une fonction de hachage est donc de conserver de l'information sur un nombre en réduisant sa taille et de ne pas pouvoir retracer ce nombre une fois qu'il a été traité par la fonction. Une utilisation fréquente des fonctions de hachage en cryptographie est pour produire une *empreinte cryptographique*, il s'agit du résultat obtenu lorsqu'on applique une fonction de hachage à un message.

Signature numérique

Lorsqu'Alice envoie un message à Bob, ce dernier est le seul à pouvoir le déchiffrer, puisqu'Alice utilise la clé publique de Bob pour chiffrer le message et Bob utilise sa clé privée pour le déchiffrer. De cette façon, l'intégrité du message est assurée. Toutefois, Bob

ne peut être certain que l'émetteur du message est véritablement Alice. N'importe qui pourrait utiliser la clé publique de Bob pour lui envoyer un message en se faisant passer pour Alice. Pour cette raison, on doit inclure une signature numérique pour assurer au destinataire que le message provient bien de la personne voulue.

La figure 1.1 illustre l'envoi d'un message d'Alice vers Bob dans le contexte d'un cryptosystème à clé publique. La signature numérique est utilisée dans cet exemple.

Afin de signer un message , on utilise son *empreinte cryptographique* (E dans la figure 1.1). On chiffre l'empreinte cryptographique avec notre clé privée puis on concatène l'empreinte chiffrée à la suite du message. On chiffre enfin le tout avec la clé publique du destinataire, pour enfin envoyer le message.

Lorsqu'il reçoit le message, le destinataire doit tout d'abord le déchiffrer avec sa clé privée, déchiffrer l'empreinte (reçue avec le message) avec la clé publique de l'émetteur, créer une empreinte du message qu'il a déchiffré et comparer celle-ci avec celle qui lui a été envoyée. Si les deux empreintes sont identiques, on a aucun doute sur l'origine du message.

1.2.1 Cryptosystème RSA

Pour mieux comprendre le fonctionnement des cryptosystèmes à clé publique, on présente le système RSA. Il s'agit d'un système efficace et très utilisé. La fonction à sens unique utilisée dans le cryptosystème RSA exploite la multiplication de deux grands nombres premiers. Le calcul de la fonction inverse, nécessite la factorisation d'un grand nombre, ce qui est reputé difficile à calculer. L'information secrète, permettant de facilement inverser la fonction, consiste en la factorisation du nombre en question.

Création des clés

Voici les étapes à suivre pour créer une paire de clés privée/publique :

1. On choisit deux nombres premiers, p et q assez grands, par exemple de l'ordre de

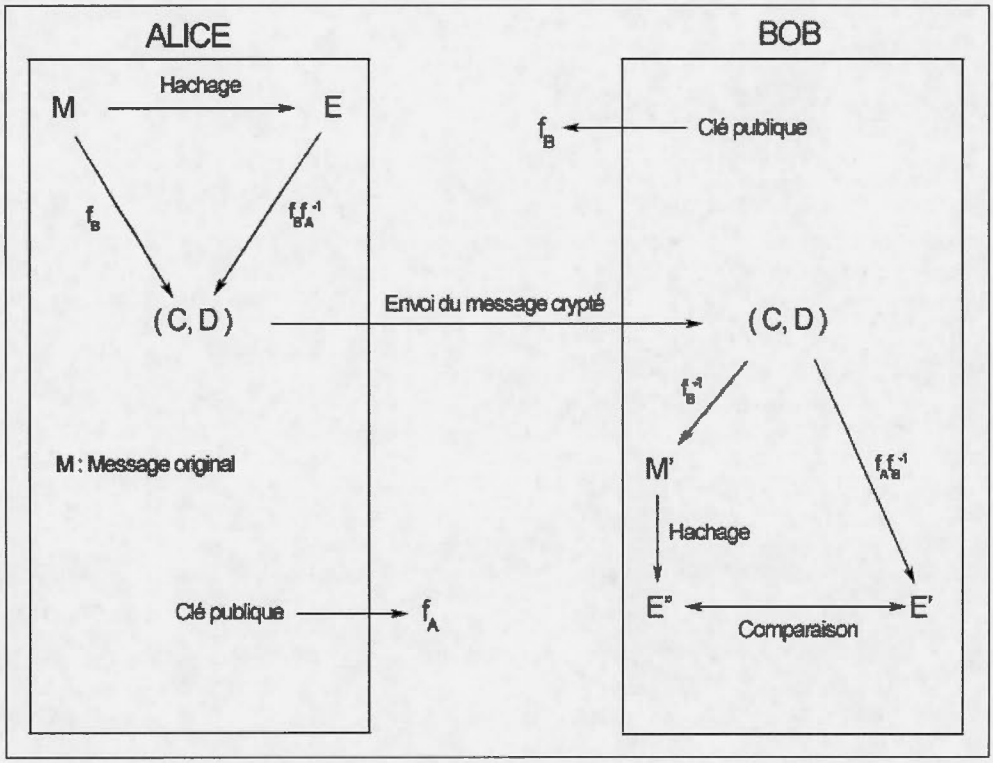


FIG. 1.1 Envoi d'un message crypté dans le cadre d'un cryptosystème à clé publique

10^{100}.

2. On calcule $n = pq$.
3. On choisit un nombre e tel que e et $(p-1)(q-1)$ soient relativement premiers.
4. On trouve un nombre d tel que $(p-1)(q-1)$ soit un facteur de ed, c'est-à-dire que $ed = 1 \bmod (p-1)(q-1)$.

La clé publique est constituée des nombres (n, e), et la clé privée est constituée des nombres (n, d).

Chiffrement

Pour chiffrer les messages, on procède comme suit :

1. Pour chiffrer un message $m \in \mathbb{N}$, l'expéditeur calcule avec la clé publique du destinataire $((n, e))$:

$$c := m^e \bmod (n).$$

   On suppose que $m < p$ et $m < q$ de sorte que m et n soient relativement premiers.

2. Sur réception du message chiffré, il suffit au destinataire de calculer $(c^d \bmod (n))$ pour récupérer m.

Pour comprendre pourquoi

$$m = c^d \bmod (n) \tag{1.1}$$

on passe par le théorème d'Euler-Fermat :

Théorème 1 *Soient a et n deux nombres premiers entre eux, alors $a^{\varphi(n)} \equiv 1 \bmod (n)$.*

Ici, la fonction φ est la fonction indicatrice d'Euler qui, pour tout entier n, est définie comme

$$\varphi(n) := \#\{m | 1 \leq m \leq n, (n, m) = 1\} \tag{1.2}$$

Dans le cas où $n = pq$, (premiers distincts) il est facile de vérifier que $\varphi(n) = (p-1)(q-1)$. Pour voir que l'égalité 1.1 est satisfaite, on calcule comme suit :

$$\begin{aligned} c^d \bmod (n) &= (m^e \bmod (n))^d \bmod (n) \\ c^d \bmod (n) &= (m^e)^d \bmod (n) \\ c^d \bmod (n) &= m^{ed} \bmod (n) \end{aligned}$$

Mais

$$ed = 1 \bmod (\varphi(n))$$

donc $ed = i\varphi(n) + 1$ pour un $i \in \mathbb{N}$.

Ainsi,

$$\begin{aligned} m^{ed} \bmod (n) &= m^{i(p-1)(q-1)+1} \bmod (n) \\ m^{ed} \bmod (n) &= m^{i(p-1)(q-1)} m \bmod (n) \\ m^{ed} \bmod (n) &= m \bmod (n) \end{aligned}$$

1.2.2 Protocole d'échange de clés

Nous allons maintenant décrire comment il est possible d'échanger une clé commune secrète via un canal non-sécuritaire. Plus particulièrement, nous allons présenter le protocole bien connu de *Diffie-Hellman*. Celui-ci est basé sur le calcul du logarithme discret.

Problème du logarithme discret :

Soit $\mathbb{F}_p^* = \mathbb{Z}/\mathbb{Z}_p = \{1, 2, ..., p-1\}$ le groupe multiplicatif des entiers modulo p, avec p premier, et soit $g \in \mathbb{F}_p^*$ fixé. Le problème du logarithme discret est, étant donné $y \in \mathbb{F}_p^*$, de déterminer s'il existe un élément x tel que $y = g^x$ existe, et alors d'en calculer la valeur.

Le protocole d'échange de clés qui suit a été introduit en 1976 par W. Diffie et M.E. Hellman. Dans un contexte de cryptographie à clé privée, les deux parties, Alice et Bob, cherchent ici à s'entendre sur un clé commune secrète que eux seul connaîtront.

Protocole de Diffie-Hellman

1. Alice et Bob s'entendent publiquement, c'est-à-dire en utilisant un canal de communication non sécuritaire, sur le choix d'un entier p et d'un élément $g \in \mathbb{F}_p^*$.
2. Alice choisit un entier positif $K_A < p$ (du même ordre de grandeur que p).
3. Alice calcule le résidu modulo p de g^{K_A} et l'envoie à Bob.
4. Bob choisit un entier positif $K_{B<p}$ du même ordre de grandeur que p.
5. Bob calcule le résidu modulo p de g^{K_B} et l'envoie à Alice.

Chacune des deux parties pourra donc calculer la clé commune $g^{K_A K_B}$ puisque

$(g^{K_A})^{K_B} = (g^{K_B})^{K_A} = g^{K_A K_B}$ est vrai dans $\mathbb{F}_p^*$.

La sécurité de cette démarche dépend de la difficulté de calculer le logarithme discret.

Il est donc clair que la résolution du problème du logarithme discret mène à la résolution du problème de Diffie-Hellman, mais on n'a pas montré que l'inverse est vrai. Il n'existe, à ce jour, pas de preuve que le problème de Diffie-Hellman ne peut se résoudre que via la résolution du problème du logarithme discret.

La cryptographie à clé publique, bien qu'ayant plusieurs avantages, est, de facon générale, plus coûteuse au point de vue algorithmique que la cryptographie à clé privée. Un juste équilibre entre ces deux techniques est donc chose commune, On utilise ainsi souvent un cryptosystème asymétrique pour construire une clé secrète entre deux parties, puis on utilise cette clé dans le cadre d'un cryptosystème symétrique.

1.3 Cryptanalyse

La cryptanalyse est l'étude de la sécurité des méthodes cryptographiques. Lorsqu'on réussit à déchiffrer un cryptogramme sans en connaître la clé, on dit que le système est *cassé*. Une tentative de casser un système cryptographique est appellé une *attaque*. Une attaque a pour but de trouver la clé servant à déchiffrer et non de déterminer l'algorithme

utilisé puisque dans la plupart des cryptosystèmes l'algorithme de chiffrement est public. Dans le cas d'un cryptosystème à clés privées, il existe au moins trois types d'attaques possibles :

1. L'***attaque à texte chiffré seul*** qui correspond à la situation où l'attaquant ne dispose que d'un ou plusieurs textes chiffrés.
2. L'***attaque à texte clair connu*** pour laquelle l'attaquant possède une ou plusieurs paires de textes chiffrés et clairs correspondants.
3. L'***attaque à texte clair choisi*** où l'attaquant peut choisir un message clair.

Dans le cas d'un cryptosystème à clés publiques, l'attaque consiste à tenter de déterminer la fonction inverse de la fonction permettant de chiffrer l'information.

CHAPITRE II

GROUPE DE TRESSES

Introduction

Nous allons voir plus loin comment il est possible de mettre en place un système cryptographique, basé sur les calculs dans les groupes de tresses. À cette fin, on rappelle d'abord dans ce chapitre, les définitions de base concernant les groupes de tresses, introduits par Artin en 1925. On en donne ici une description géométrique, algébrique, puis on présente quelques propriétés.

2.1 Définition du groupe de tresses

2.1.1 Approche géométrique

Tresse géométrique

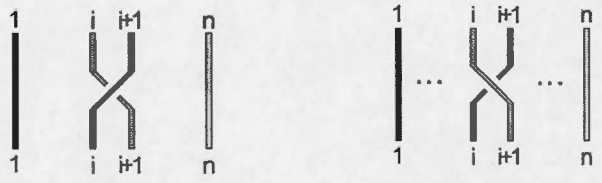
Informellement, une tresse est un ensemble de *brins* placés à la verticale, qui se croisent les uns les autres. On peut modéliser une tresse dans $\mathbb{R}^3$, un brin est alors une courbe reliant le point $(x, 0, 0)$ au point $(y, 0, 1)$. Une tresse sur n brins est un ensemble de n courbes reliant les points $(1, 0, 0), ..., (n, 0, 0)$ aux points $(1, 0, 1), ..., (n, 0, 1)$. Un tel modèle est appellé *tresse géométrique* à n brins.

Soient deux tresses géométriques β_1 et β_2. Le produit $\beta_1\beta_2$ est définit comme l'empilement de la tresse β_1 sur β_2 en attachant chaque brin de β_1 à son brin respectif dans β_2,

puis en comprimant le tout pour que les nouvelles courbes ainsi formées soient incluses entre les plans $z = 0$ et $z = 1$.

Diagramme de tresse

Pour faciliter l'étude des tresses, on peut également utiliser un modèle qu'on appelle diagramme de tresse. On l'obtient en projetant une tresse géométrique dans le plan $y = 0$, en réarrangeant les courbes de la tresse géométrique de façon à obtenir des brins rectilignes et en distinguant dans quel sens les brins s'intersectent. Chacune des intersections dans le diagramme de tresse est appellé *un croisement.* Les *croisements* dans une tresse sont aussi appelés *tresses élémentaires* et sont notés par la lettre grecque σ_i, ou son inverse, σ_i^{-1}, l'indice i indique que le croisement est situé entre les brins i et $i+1$. Par convention, la tresse élémentaire σ_i est un croisement entre les brins i et $i+1$ où le brin i passe sous le brin $i+1$ dans le diagramme de tresse. Dans la tresse élémentaire inverse σ_i^{-1}, le brin $i+1$ passe sous le brin i. Ainsi, la figure 2.2 illustre une tresse sur cinq brins qui présente trois croisements.



(a) Diagramme du générateur σ_i

(b) Diagramme du générateur σ_i^{-1}

FIG. 2.1 Tresses élémentaires

En fixant les extrémités d'une tresse géométrique et en déplacant les brins, on change la forme de la tresse, mais pas sa structure topologique. Si une tresse ω' est obtenue en déplacant de la sorte les brins d'une tresse ω, on dit que les tresses ω et ω' sont *équivalentes.* Cette notion d'équivalence permet de contruire un ensemble de classes d'équivalence qui est muni d'une structure de groupe, c'est le *groupe de tresses* à n brins noté B_n. Voyons maintenant comment les propriétés d'une structure de groupe apparaissent dans cet

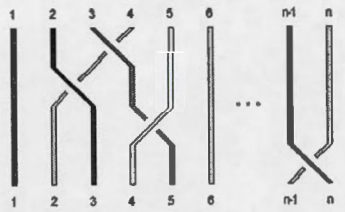


FIG. 2.2 Exemple d'un diagramme de tresse

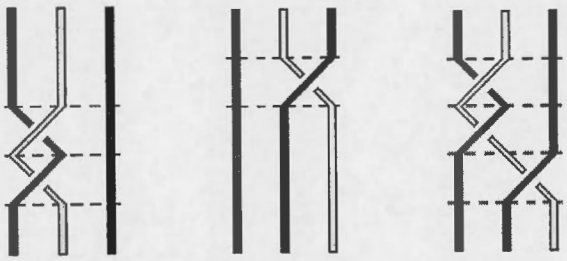


FIG. 2.3 Exemple d'un produit de tresses

ensemble particulier. L'élément neutre du groupe de tresses géométriques est appellé la *tresse triviale*. Il s'agit d'une tresse ne contenant aucun croisement, donc constituée uniquement de brins parallèles, comme le montre la figure 2.4. La tresse inverse β^{-1}

FIG. 2.4 Tresse triviale

d'une tresse β est définie comme étant l'image mirroir de β par rapport à l'axe horizontal. La figure 2.5 montre le diagramme de la tresse produit de $\sigma_2\sigma_1\sigma_3$ et de son inverse $(\sigma_2\sigma_1\sigma_3)^{-1}$. On peut bien voir qu'il est possible d'effectuer un déplacement des brins sur cette tresse, en laissant les extrémités fixes qui résulterait en la tresse triviale. Remarquons que dans la tresse inverse de la figure 2.5, les croisements sont inversés, c'est-à-dire que le brin i passe par-dessus le brin $i+1$ plutôt qu'en dessous. Si, dans

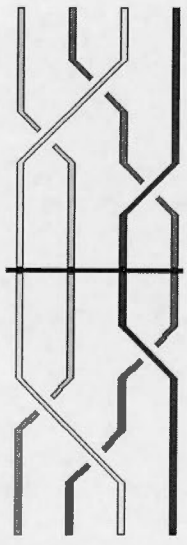


FIG. 2.5 Exemple du produit d'une tresse et de son inverse

une tresse, on réunit, pour chaque brin, ses deux extrémités, on obtient un *noeud*. La

théorie des noeuds est tout aussi riche que celle des tresses, sinon plus, mais nous n'en traiterons ici que très peu.

2.1.2 Approche algébrique

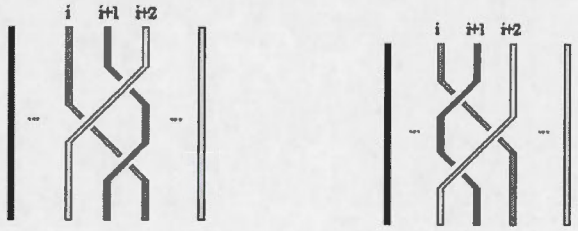
Plus formellement, une présentation classique du groupe de tresses, donnée par Artin en 1925, est la suivante : Sur les générateurs $\sigma_1, \sigma_2, ..., \sigma_{n-1}$, on considère les relations

$$\sigma_i \sigma_j = \sigma_j \sigma_i \tag{2.1}$$

lorsque $|i - j| > 1$, et

$$\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1} \,. \tag{2.2}$$

La relation 2.1 implique deux croisements consécutifs séparés par au moins un brin dans la tresse, les croisements peuvent donc être déplacés verticalement dans le diagramme de tresse sans difficulté. La figure 2.7 illustre la relation 2.2. On peut voir, en utilisant la notion d'équivalence définie à la section 2.1.1, que les deux diagrammes décrivent la même tresse. Cette relation est un fait un des trois *mouvements de Reidemeister*, qui sont des déplacements de brins permettant de passer d'un noeud à un autre noeud équivalent.



(a) $\sigma_{i+1}\sigma_i\sigma_{i+1}$ (b) $\sigma_i\sigma_{i+1}\sigma_i$

FIG. 2.6 Exemple de la relation 2.2

2.2 Tresse fondamentale

Un morphisme naturel allant de B_n au groupe S_n, des permutations de n éléments est défini comme suit : À chaque croisement σ_i ou σ_i^{-1}, on associe tout simplement la transposition $(i \;\; i+1)$.

$$\varphi : B_n \longrightarrow S_n \tag{2.3}$$

Le noyau de ce morphisme est appellé le *groupe des tresses pures*, c'est donc l'ensemble des tresses pour lesquelles chaque brin relie i à i.

Définition 2 *Soit l'application Ψ, qui associe à une permutation $\pi = (t_1, t_1+1)(t_2, t_2+1)...(t_k, t_k+1)$ dans S_n, la tresse $\sigma_{t_1}\sigma_{t_2}...\sigma_{t_k}$. La tresse obtenue est dite* tresse de permutation *ou* tresse simple.

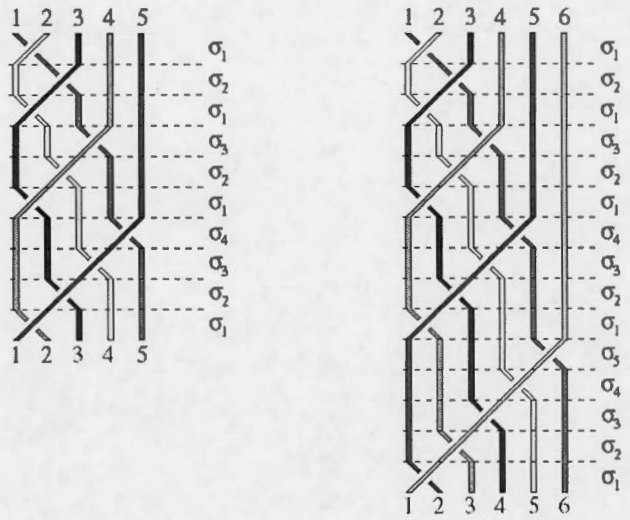
On note $\tilde{S_n}$ l'ensemble des tresses de permutation. Une de ces tresses de permutation se démarque , il s'agit de la tresse dite *fondamentale*, dénotée Δ_n. Dans la littérature, on retrouve aussi l'expression *élément de Garside* pour la désigner. Elle est l'image de la permutation $(n\; n-1\; ...\; 1)$ par l'application Ψ. Pour simplifier, on utilisera seulement le symbole Δ en omettant l'indice pour désigner Δ_n dans le groupe B_n. Géométriquement, Δ_n peut être produite à partir de la tresse triviale en tournant d'un demi-tour les points d'attache du bas de la tresse. En fait, un brin partant de la position i en haut du diagramme de tresse, se retrouve à la position $n+1-i$ en bas du diagramme. La tresse fondamentale peut être définie par récurrence comme suit :

$$\Delta_{n+1} = \Delta_n \sigma_n \sigma_{n-1} ... \sigma_1 \tag{2.4}$$

avec

$$\Delta_2 = \sigma_1 \text{ et } \Delta_1 = Id$$

La figure 2.2 illustre la différence entre les deux tresses fondamentales Δ_5 et Δ_6, on peut ainsi mieux comprendre l'équation de récurrence 2.4.



(a) $\Delta 5$ (b) $\Delta 6$

FIG. 2.7 Deux diagrammes de tresses fondamentales (Δ_5 et Δ_6)

Dans la tresse fondamentale, chaque paire de brins crée un croisement et donc, le nombre de générateurs nécessaires pour écrire le mot de la tresse fondamentale est $\frac{n(n-1)}{2}$.

Voici deux autres façons de décrire la tresse Δ_n, pour $1 \leq i \leq n-1$:

$$\begin{aligned} \Delta_n &= (\sigma_i)(\sigma_{i+1}\sigma_i)...(\sigma_{n-1}...\sigma_i)(\sigma_{i-1}\sigma_i...\sigma_{n-1})...(\sigma_1...\sigma_{n-1}) \\ &= (\sigma_1...\sigma_{n-1})...(\sigma_1...\sigma_{i+1})(\sigma_i...\sigma_1)...(\sigma_i\sigma_{i-1})\sigma_i \end{aligned} \tag{2.5}$$

Ceci nous permet de trouver (Mahlburgh, 2004) la propriété suivante :

$$\Delta_n\sigma_i = \phi(\sigma_i)\Delta_n \text{ où } \phi(\sigma_i) = \sigma_{n-i} \tag{2.6}$$

L'équation 2.5 nous montre que tout générateur de B_n peut diviser Δ_n à gauche et à droite. Cette propriété sera exploitée ultérieurement.

2.3 Forme normale

On a vu qu'on pouvait exprimer une tresse par un mot. Différents mots peuvent exprimer la même tresse, c'est à dire qu'un élément du groupe de tresses est une classe d'équivalence de mots. Nous définissions ici une forme normale désignant un mot dans chaque

classe d'équivalence, ce mot sera le représentant de sa classe d'équivalence. La forme normale d'un mot nous permet de résoudre le *problème de mot*, c'est-à-dire, déterminer si deux mots sont équivalents. Avant d'introduire la forme normale, nous aurons besoin de quelques notions.

Définition 3 *Soient $a, b \in B_n$, on dit que a* divise *b à gauche (resp. à droite) s'il existe une tresse $c \in B_n$ telle que $b = ac$ (resp. $b = ca$). On note par $a \preceq b$ (resp. $b \succeq a$) pour signifier que a divise b à gauche (resp. à droite). Lorsque a divise b à gauche (resp. à droite), on dit également que b est un multiple à droite (resp. à gauche) de a.*

Définition 4 *On définit un ordre ($\leq$) sur les éléments de B_n :*

$$v \leq w \Leftrightarrow \exists\, \alpha, \beta \in B_n^+ \text{ avec } w = \alpha v \beta\ .$$

B_n^+ est l'ensemble des tresses positives, c'est-à-dire des tresses formées uniquement de générateurs positifs. On dit que toute tresse $a \in B_n$ telle que $Id \leq a \leq \Delta_n$ est un *facteur canonique* ou une *tresse simple*. Si on restreint le morphisme $\varphi : B_n \to S_n$, à l'ensemble des facteurs canoniques dans B_n, on obtient une bijection.

Soit une tresse $y \in B_n^+$ et sa factorisation $y = ab$ où a est un facteur canonique et b est une tresse positive. Si a est de longueur maximale, alors la factorisation est "left-weighted". On appelle le *minorant* (resp. *majorant*) d'une tresse $w \in B_n$, notée $inf(w)$ (resp. $sup(w)$), le plus grand (resp. petit) nombre $i \in \mathbb{Z}$ tel que $\Delta_n^i \leq w$ (resp. $w \leq \Delta_n^i$).

Définition 5 *On dit qu'une suite de tresses positives $(b_1, ..., b_p)$ est* normale à gauche *si, pour tout i, b_i est une tresse simple et non triviale et si b_i est la plus grande tresse simple divisant $b_i b_{i+1}$.*

Toute tresse $w \in B_n$ peut s'écrire de manière unique $w = \Delta_n^r W_1 W_2 ... W_s$ avec $r = inf(w)$, $s = sup(w) - inf(w)$ et les facteurs canoniques W_i tels que $Id \leq W_i \leq \Delta_n$ et que $W_i W_{i+1}$ est "left-weighted" pour $1 \leq i \leq s$. Il s'agit de la *forme normale à gauche*. L'entier s est appellé la longueur canonique de w ($l_c(w)$).

Éliminer les termes négatifs

Les facteurs canoniques dans la forme normale d'une tresse $w \in B_n$ sont composés de générateurs uniquement positifs. Évidemment, si la tresse w contient un générateur négatif σ_i^{-1}, on doit pouvoir trouver sa forme normale. Pour trouver une tresse composée uniquement de générateurs positifs qui est équivalente à un générateur négatif, on multiplie le générateur négatif à gauche par $\Delta_n^{-1}\Delta_n$. Avec la propriété 2.5, on peut annuler le générateur négatif σ_i^{-1} et on se retrouve avec une tresse de la forme $\Delta_n^{-1}v$ où v est une tresse positive.

Exemple 6 *Trouvons une tresse équivalente au générateur $\sigma_3^{-1} \in B_4$ de la forme $\Delta_n^{-1}v$ où v est une tresse positive.*

$$\begin{aligned}
\sigma_3^{-1} &= \Delta_4^{-1}\Delta_4\sigma_3^{-1} \\
&= \Delta_4^{-1}\sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1\sigma_3^{-1} \\
&= \Delta_4^{-1}\sigma_2\sigma_1\sigma_2\sigma_3\sigma_2\sigma_1\sigma_3^{-1} \\
&= \Delta_4^{-1}\sigma_2\sigma_1\sigma_3\sigma_2\sigma_3\sigma_1\sigma_3^{-1} \\
&= \Delta_4^{-1}\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1\sigma_3\sigma_3^{-1} \\
&= \Delta_4^{-1}\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1
\end{aligned}$$

Trouver la forme normale

Lorsqu'on veut trouver la forme normale d'une tresse, on change d'abord tous les générateurs négatifs par une tresse équivalente de la forme $\Delta_n^{-1}v$, où v est positive, puis on déplace à l'extrême gauche du mot le terme Δ_n^{-1}. Par la propriété 2.6, chaque terme σ_i à gauche de Δ_n^{-1} est remplacé par σ_{n-i} lorsqu'on déplace le terme Δ_n^{-1}. On doit ensuite placer les générateurs positifs restants en tresses simples.

On a déjà dit que chaque facteur canonique de B_n peut être associé à une unique permutation dans S_n, donc c'est vrai pour les W_i, on verra plus loin qu'on implémente souvent la forme normale d'une tresse en informatique en utilisant des permutations.

2.4 Réduction d'un mot

La réduction d'un mot est une application ayant la propriété suivante :

Propriété 7 *Un mot de tresse a représente la tresse identité si et seulement si* $\mathrm{red}(a)$ *est un mot vide.*

La réduction d'un mot associe un mot de tresse a, à un mot $\mathrm{red}(a)$ sous *forme réduite.* La forme réduite d'un mot n'est pas une forme normale donc, par exemple, même si deux mots de tresses sont équivalents, leur réduction peut donner deux mots différents.

À ce jour, on connait des algorithmes de réduction plus efficaces que les algorithmes permettant de trouver la forme normale d'un mot de tresse. La réduction est donc à privilégier pour résoudre le problème de mot puisqu'on n'a qu'à vérifier, pour deux mots a et b, que $\mathrm{red}(ab^{-1})$ est un mot vide.

Problèmes difficiles dans le groupe de tresses

Voici quelques problèmes réputés difficiles à résoudre dans le groupe de tresses. Ces problèmes pourraient être, à un certain point, pertinents à utiliser dans l'élaboration d'un cryptosystème.

1. Décision de conjugaison

   On connaît : $x \in B_n$, $y \in B_n$

   But : Trouver si x et y sont conjugués ou non.

2. Problème de conjugaison

   On connaît : $x \in B_n$, $y \in B_n$ tels que x et y sont conjugués.

   But : Trouver $a \in B_n$ tel que $x = aya^{-1}$.

3. Problème de conjugaison généralisé

   On connaît : $x \in B_n$ et $y \in B_n$ tels que $y = bxb^{-1}$, pour un $b \in B_m$, avec $m \leq n$ et B_m un sous-groupe de B_n.

   But : Trouver $a \in B_n$ tel que $x = aya^{-1}$.

4. Décomposition du conjugué

   On connaît : $x \in B_n$ et $y \in B_n$ tels que $y = bxb^{-1}$, pour un $b \in B_m$, avec $m \leq n$.

   But : Trouver $a \in B_m$ et $a^* \in B_m$ tels que $x = aya^*$

5. Racine $p^{ième}$

   On connaît : $y \in B_n$, $p \in \mathbb{Z}$ tels que $y = x^p$ avec $x \in B_n$.

   But : Trouver $z \in B_n$ tel que $y = z^p$

6. Problème de Markov

   On connaît : $y \in B_n$ tel que y est conjugué à une tresse de la forme $w\sigma_{n-1}^{\pm 1}$ avec $w \in B_{n-1}$.

   But : Trouver $z \in B_n$ et $x \in B_{n-1}$ tels que $zyz^{-1} = x\sigma_{n-1}^{\pm 1}$.

2.5 Conjugaison dans B_n

Le problème de recherche du conjugueur dans le groupe B_n est réputé difficile. Nous nous attarderons plus loin sur ce problème dans le cadre de protocoles cryptographiques. Pour l'instant, voyons quelques définitions et propriétés relatives à la conjugaison dans B_n.

Définition 8 *Soit une tresse $w \in B_n$ dont la forme normale est $w = \Delta_n^{-r} s_1 s_2 ... s_p$. L'opération δ_+ définie par $\delta_+(w) = \Delta_n^{-r} s_2 ... s_p \phi(s_1)$ est appellée* cyclage *et l'opération δ_- définie par $\delta_-(w) = \Delta_n^{-r} \phi(s_p) s_1 ... s_{p-1}$ est appellée* décyclage

Lemme 9 *Pour toute tresse $w \in B_n$, $\delta_+(w)$ et $\delta_-(w)$ sont des conjugués de w.*

Preuve: Soit $w = \Delta_n^{-r} s_1 s_2 ... s_p$ la forme normale à gauche d'une tresse du groupe B_n. On peut montrer que $\delta_+(w)$ et w sont conjugués par l'élément $\phi(s_1)$ et que $\delta_-(w)$ et w sont conjugués par l'élément s_p.

On a

$$\begin{aligned} \phi(s_1)\delta_+(w) &= \phi(s_1)\Delta_n^{-r} s_2 ... s_p \phi(s_1) \\ &= \Delta_n^{-r} s_1 s_2 ... s_p \phi(s_1) \qquad \text{(avec la propriété 2.6)} \\ &= w\phi(s_1) \end{aligned}$$

donc $\delta_+(w) = (\phi(s_1))^{-1} w \phi(s_1)$.

Et

$$\begin{aligned} \delta_-(w)s_p &= \Delta_n^{-r}\phi(s_p)s_1...s_{p-1}s_p \\ &= s_p\Delta_n^{-r}s_1s_2...s_p \qquad \text{(avec la propriété 2.6)} \\ &= s_p w \end{aligned}$$

donc $\delta_-(w) = s_p^{-1} w s_p$. ■

Super Summit Set

On utilise souvent un sous-ensemble fini d'une classe de conjugaison dans B_n appellé le *Super Summit Set* d'un élément $w \in B_n$ ($SSS(w)$). Il s'agit de l'ensemble des éléments conjugués de w tels que la longueur canonique est minimale. Si deux éléments sont conjugués, ils ont le même Super summit set. Deux tresses faisant partie du même Super Summit Set sont conjuguées, puisqu'elles font partie de la même classe de conjugaison, et l'élément les conjuguant est une tresse simple, puisqu'elles ont la même longueur canonique.

Nous avons vu que l'opération de cyclage δ_+ n'ajoute pas de générateurs à une tresse, et qu'elle constitue une conjugaison par un élément simple. Ainsi la tresse résultante d'une suite d'opérations de cyclage sur une tresse originale ω aura une complexité inférieure ou égale à celle de ω. Pour une tresse $\omega = \Delta_n^{-r}s_1s_2...s_p$, en appliquant p fois l'opération de cyclage, on peut déterminer un conjugué de w de complexité minimale, donc un élément de $SSS(\omega)$.

2.6 Autres groupes intéressants

Nous présentons ici quelques groupes qui sont des généralisations du groupe B_n. La figure 2.8 illustre le raffinement des groupes gaussiens vers les groupes de tresses dont nous traitons dans cette section.

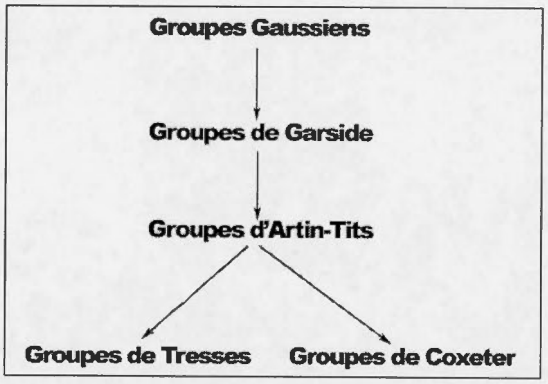


FIG. 2.8 Généralisations de B_n

2.6.1 Groupes d'Artin

Les groupes de tresses sont un cas particulier d'une famille plus générale : les *groupes d'Artin*, ou groupes d'Artin-Tits. Les groupes d'Artin ont comme générateurs $x_1, x_2, ..., x_n$ avec les relations :

Pour chaque couple (i, j), $1 \leq i, j \leq n$:

$$(x_i x_j)_{m_{i,j}} = (x_j x_i)_{m_{i,j}} \tag{2.7}$$

où $(x_i x_j)_{m_{i,j}}$ est le produit de $m_{i,j}$ générateurs x_i et x_j, en les alternant et en débutant par x_i :

$$(x_i x_j)_{m_{i,j}} = \underbrace{x_i x_j x_i x_j ...}_{m_{i,j} \text{ termes}}$$

La donnée d'un groupe d'Artin revient donc au choix des entiers $m_{i,j}$. Par convention, on pose $m_{i,i} = 1$.

Clairement, le groupe de tresses B_n correspond au choix

$$m_{i,j} = 2, \; si \; |i - j| > 1$$

et

$$m_{i,j} = 3, \; si \; |i - j| = 1 \; .$$

2.6.2 Groupes de Coxeter

L'etude des *groupes de Coxeter* est intimement liée à celle des groupes d'Artin. Ils ont une présentation qui s'obtient en ajoutant simplement les relations $x_i^2 = 1$, avec $1 \leq i \leq n$, à celle des groupes d'Artin. Ainsi, chaque groupe d'Artin est associé à un groupe de Coxeter qui joue un rôle analogue au groupe symétrique. B_n étant un cas particulier des groupes d'Artin, on peut associer, à B_n un groupe de Coxeter en ajoutant les relations $x_i^2 = 1$.

2.6.3 Groupes de Garside

Dans la section 2.2, nous avons vu que la tresse fondamentale, qu'on appelle aussi *élément de Garside*, dans le groupe B_n était divisible à gauche et à droite par tout générateur de B_n. En fait, dans un monoïde quelconque $\mathcal{M}$, un élément de Garside est un élément dont l'ensemble des diviseurs à gauche coincide avec l'ensemble des diviseurs à droite, et dont les diviseurs engendrent $\mathcal{M}$. Ces diviseurs doivent également être en nombre fini.

Les éléments qui divisent l'élément de Garside d'un monoïde sont appellés les *simples* relativement à l'élément de Garside. La notion d'élément de Garside est fondamentale dans la définition d'un groupe de Garside. Un groupe de Garside est le groupe de fractions d'un monoïde de Garside, que nous définissons un peu plus loin. Auparavant voyons quelques définitions.

Un élément x dans un monoïde $\mathcal{M}$ est appellé un *atome* si x est différent de 1 et si $x = yz$ implique $y = 1$ ou $z = 1$. En d'autres termes, x doit être indécomposable. On note par $A_{\mathcal{M}}$ l'ensemble des atomes du monoïde $\mathcal{M}$. On peut donc exprimer un élément comme un produit d'atomes. La borne supérieure de la longueur de la décomposition d'un élément $x \in \mathcal{M}$ en produit d'atomes est appellée la *norme* de x, on la note $\| x \|$. Si $\mathcal{M}$ est engendré par $A_{\mathcal{M}}$ et que tous les éléments de $\mathcal{M}$ ont une norme finie, alors on dit que $\mathcal{M}$ est un *monoïde atomique.*

On dit qu'un monoïde $\mathcal{M}$ est un *monoïde gaussien* si $\mathcal{M}$ est atomique, simplifiable et

si, pour chaque couple $x, y \in \mathcal{M}$ il existe un ppcm à gauche et à droite de x et y ainsi qu'un pgcd à gauche et à droite de x et y. Enfin, on dit qu'un monoïde gaussien $\mathcal{M}$ est un *monoïde de Garside* s'il contient un élément de Garside.

Un monoïde de Garside peut contenir plus d'un élément de Garside. En fait toute puissance d'un élément de Garside est aussi un élément de Garside. On peut aussi vérifier que le pgcd de deux éléments de Garside en est également un, en effet, tous ses diviseurs à gauche et à droite coïncident.

CHAPITRE III

ASPECTS DE LA THÉORIE DES REPRÉSENTATIONS D'UN GROUPE

Introduction

La théorie de la représentation des groupes consiste à dégager des propriétés de groupes en étudiant leur action sur des espaces vectoriels. Autrement dit, cette façon de procéder permet de « transporter »l'étude de groupes abstraits au contexte plus concret de l'algèbre linéaire. On peut ainsi ramener des problèmes difficiles de la théorie des groupes à un contexte dans lequel on a plus d'outils pour résoudre ces problèmes. Nous donnons dans ce chapitre un très bref aperçu de la théorie de la représentation de groupes, plus particulièrement pour le groupe de tresses B_n. Dans les chapitres ultérieurs nous verrons que ces représentations contribuent à résoudre partiellement certains problèmes difficiles qui ont étés mentionnés au chapitre 2.

3.1 Représentation linéaire d'un groupe

Définition 10 *Une représentation linéaire d'un groupe G est un homomorphisme de G dans le groupe $GL(V)$ des transformations linéaires inversibles de V dans V :*

$$T : G \to GL(V) .$$

Ici V est un espace vectoriel, de dimension finie n, sur un corps $\mathbb{K}$.

L'espace vectoriel V est l'*espace sous-jacent à la représentation* et la dimension de l'espace vectoriel V est appellé le *degré* de la représentation. En d'autres termes, une représentation associe, à chaque élément du groupe une matrice dans le groupe $GL_n(\mathbb{K})$, des matrices $n \times n$ sur le corps $\mathbb{K}$. Bien entendu, l'opération interne dans le *groupe linéaire* $GL_n(V)$ est la multiplication matricielle.

On dit d'une représentation T qu'elle est *fidèle* si T est injective. La *représentation triviale* est celle pour laquelle $T(g)$ est la transformation triviale I_n, pour tout $g \in G$.

Exemple 11 *Considérons le groupe G, défini par les générateurs s et t, avec les relations suivantes :*

$$s^4 = e, t^2 = e, st = ts^{-1} \tag{3.1}$$

Il s'agit en fait du groupe dihédral D_4, constitué des isométries du carré. Le groupe G contient 8 éléments, et on en obtient une représentation en posant :

$$s = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad t = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

puisque les matrices en question satisfont les relations de 3.1.

Définition 12 *Pour deux représentations : $T_1 : G \to GL(V_1)$ et $T_2 : G \to GL(V_2)$, on dit que T_1 et T_2 sont* équivalentes *s'il existe un isomorphisme d'espaces vectoriels $\phi : V_1 \to V_2$ tel que $\phi T_1(g)\phi^{-1} = T_2(g)$ pour tout $g \in G$.*

Pour une représentation $T : G \to GL(V)$ on dit qu'un sous-espace vectoriel U de V est *stable* (relativement à T) si $T(g)(u) \in U$ pour tout $g \in G$ et tout $u \in U$. La notion de sous-espaces stables permet d'introduire naturellement la notion de représentation irréductible.

Définition 13 *Soit V un espace vectoriel dont les deux seuls sous-espaces stables (relativement à $R : G \to GL(V)$) sont 0 et V, alors on dit que le représentation R est* irréductible.

3.2 Représentation du groupe symétrique

Nous verrons ultérieurement la relation entre le groupe symétrique et le groupe de tresses ; interessons nous pour l'instant à une représentation particulière du groupe symétrique. La *représentation standard* du groupe symétrique associe, pour chaque transposition $(i\ i+1)$, la matrice

$$\left(\begin{array}{c|cc|c} I_{i-1} & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & I_{n-i-1} \end{array}\right).$$

3.3 Représentation de Burau

La représentation de Burau $\varphi : B_n \to GL_n(\mathbb{Z}[t,t^{-1}])$ associe, au générateur d'Artin σ_i la matrice

$$\varphi(\sigma_i) = \left(\begin{array}{c|cc|c} I_{i-1} & 0 & 0 & 0 \\ \hline 0 & 1-t & t & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & I_{n-i-1} \end{array}\right).$$

Observons que l'inverse de $\varphi(\sigma_i)$ est

$$\varphi(\sigma_i)^{-1} = \left(\begin{array}{c|cc|c} I_{i-1} & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ 0 & \frac{1}{t} & \frac{t-1}{t} & 0 \\ \hline 0 & 0 & 0 & I_{n-i-1} \end{array}\right).$$

On vérifie facilement que ces matrices satisfont les relations de tresses. La matrice associée à ω une tresse quelconque peut donc se calculer en multipliant les matrices correspondant aux générateurs qui apparaissent dans n'importe quelle expression. Pour ω, on désigne par $\varphi(\omega)$ le résultat ainsi obtenu.

Exemple 14 *Pour $w = \sigma_1\sigma_3\sigma_2 \in B_4$, calculons $\varphi(w)$:*

$$
\begin{aligned}
\varphi(w) &= \varphi(\sigma_1\sigma_3\sigma_2) \\
&= \varphi(\sigma_1)\cdot\varphi(\sigma_3)\cdot\varphi(\sigma_2) \\
&= \begin{pmatrix} 1-t & t & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1-t & t \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1-t & t & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 1-t & t & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1-t & t & 0 \\ 0 & 1-t & 0 & t \\ 0 & 1 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 1-t & -t(-1+t) & t^2 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1-t & 0 & t \\ 0 & 1 & 0 & 0 \end{pmatrix}
\end{aligned}
$$

3.3.1 Propriétés de la représentation de Burau

Pour une tresse $\omega = x\sigma_i$ avec $x \in B_n^+$ on peut vérifier, pour la k ième colonne $\varphi(\omega)_k$, de la matrice $\varphi(\omega)$, l'équation

$$
\varphi(\omega)_k = \begin{cases} (1-t)\varphi(x)_i + \varphi(x)_{i+1} & \text{si } k = i \\ t\varphi(x)_i & \text{si } k = i+1 \\ \varphi(x)_k & \text{sinon} \end{cases} \tag{3.2}
$$

Lemme 15 *Soit une matrice $A = [a_{ij}] \in GL_n$ appartenant à l'image de la représentation de Burau. Alors, on a*

$$
A \cdot \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T = \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T \tag{3.3}
$$

Preuve: On sait que toute matrice de Burau peut être obtenue par un produit d'images de générateurs ($\varphi(\sigma_i)$). La matrice désignant l'image d'un générateur répond à la propriété que la somme de toutes ses rangées soit égale à 1 :

$$\varphi(\sigma_i) = \left(\begin{array}{c|ccc} I_{i-1} & 0 & 0 & 0 \\ \hline 0 & 1-t & t & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & I_{n-i-1} \end{array}\right).$$

Il suffit alors de vérifier que, pour la multiplication matricielle, la propriété est préservée. Soient deux matrices de dimension $n \times n$ A et B, alors

$$A \cdot \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T = \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T$$

et

$$B \cdot \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T = \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T$$

entraîne évidemment que

$$A \cdot B \cdot \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T = \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T.$$

■

L'équation 3.3 montre que 1 est l'une des valeurs propres de toutes les matrices de Burau.

Lemme 16 *Soit une matrice $A = [a_{ij}] \in GL_n$ appartenant à l'image de la représentation de Burau. Alors, on peut dégager l'équation suivante :*

$$\begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix} \cdot A = \begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix} \tag{3.4}$$

Preuve: Comme pour la preuve précédente, on remarque que la propriété est vraie pour chaque matrice image d'un générateur.

$$\begin{pmatrix} t \\ t^2 \\ \dots \\ t^n \end{pmatrix} \cdot \left(\begin{array}{c|ccc} I_{i-1} & 0 & 0 & 0 \\ \hline 0 & 1-t & t & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & I_{n-i-1} \end{array}\right) = \begin{pmatrix} t \\ t^2 \\ \dots \\ t^n \end{pmatrix}$$

Nous allons donc vérifier que la multiplication matricielle préserve la propriété.

Soient deux matrices de dimension $n \times n$ A et B, alors

$$\begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix} \cdot A = \begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix}$$

et

$$\begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix} \cdot B = \begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix}$$

entraîne évidemment que

$$\begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix} \cdot A \cdot B = \begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix}.$$

■

La représentation de Burau est fortement liée à la représentation standard du groupe symétrique présentée à la section 3.2. On remarque qu'en remplacant la valeur t par 1, dans la représentation de Burau, on retrouve la représentation standard de S_n. Cette caractéristique de la représentation de Burau est notamment due au morphisme 2.3 envoyant un générateur σ_i du groupe B_n sur la transposition $(i\ i+1)$ dans S_n.

3.3.2 Inverser la représentation

Puisque la fonction φ n'est pas injective, la correspondance φ^{-1} peut admettre plus d'une image pour une matrice donnée. Soit une matrice $A \in \mathrm{Im}(\varphi)$, pour trouver une tresse $a \in B_n$ telle que $\varphi(a) = A$, on utilise un algorithme heuristique. La tresse qu'on cherche est sous la forme d'Artin, c'est-à-dire, écrite avec les générateurs σ_i.

Supposons qu'on veuille trouver un mot de tresse ω, tel que $\mathrm{l}(\omega) \geq 2$, à partir de la matrice $\varphi(\omega)$. On trouve, un à un, les générateurs σ_i composant $\sigma_{i_1} \dots \sigma_{i_l}$, un mot de tresse pour ω, en partant de σ_{i_l} à σ_{i_1}. Pour pouvoir trouver une valeur de i telle que $\omega = x\sigma_i$, on doit calculer

$$\varphi(x) = \varphi(\omega)\varphi(\sigma_i)^{-1}$$

avec $\mathrm{l}(x) < \mathrm{l}(\omega)$.

On fait ensuite la même chose pour le mot de tresse x, jusqu'à ce qu'on obtienne la matrice identité.

Pour déterminer, à partir d'une matrice $\varphi(\omega)$, un candidat pour le dernier générateur du mot ω, on trouve la colonne de la matrice qui contient la plus grande puissance (positive) de t. Le numéro de cette colonne correspond "généralement" au dernier générateur du mot ω augmenté de 1. Plus précisément, Lee et Park (EonKyung Lee, 2003) ont montré que cette technique indique le dernier générateur du mot ω avec une probabilité supérieure à $1 - \frac{1}{m-1}$ où m est le nombre de brins de la tresse.

Exemple 17 *Soit la matrice*

$$\varphi(\omega) = \begin{pmatrix} 1-t & 1-t^2 & t^2 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1-t & 0 & t \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

La première colonne contenant la plus grande puissance de t est la troisième, donc $\omega = \omega' \cdot \sigma_2$. Trouvons maintenant la valeur de $\varphi(\omega')$:

$$\begin{aligned} \varphi(\omega') &= \varphi(\omega) \cdot \varphi(\sigma_2)^{-1} \\ &= \begin{pmatrix} 1-t & t & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1-t & t \\ 0 & 0 & 1 & 0 \end{pmatrix}, \end{aligned}$$

Cette fois ci, la première colonne contenant la plus grande puissance de t est la deuxième, donc $\omega' = \omega'' \cdot \sigma_1$. Trouvons maintenant la valeur de $\varphi(\omega'')$:

$$\varphi(\omega'') = \varphi(\omega') \cdot \varphi(\sigma_1)^{-1}$$
$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1-t & t \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

Maintenant, la première colonne contenant la plus grande puissance de t est la quatrième, donc $\omega'' = \omega''' \cdot \sigma_3$. La valeur de $\varphi(\omega''')$ est la matrice identité :

$$\varphi(\omega''') = \varphi(\omega'') \cdot \varphi(\sigma_3)^{-1}$$
$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Ainsi, $\omega = \sigma_3\sigma_1\sigma_2$.

On peut donc trouver un mot de tresse w à partir de la matrice $\varphi(w)$. Rappellons toutefois que la représentation de Burau n'est pas fidèle. Si deux mots de tresse, disons v et w sont envoyés sur la même matrice par la représentation de Burau ($\varphi(v) = \varphi(w) = A$ avec $v \neq w$), l'algorithme proposé ne nous donne aucune information sur le résultat (v ou w) qui sera privilégié.

3.3.3 Représentation de Burau réduite

La représentation de Burau peut être décomposée en une somme directe de deux représentations irréductibles, l'une de dimension 1, et l'autre, qu'on appelle représentation de Burau réduite, qui est de dimension $n-1$. La représentation de Burau réduite $\varphi_{red} : B_n \to GL_{n-1}(\mathbb{Z}[t, t^{-1}])$ est simplement :

$$\varphi_{red}(\sigma_i) = \left(\begin{array}{c|ccc|c} I_{i-2} & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 0 & t & -t & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & I_{n-i-2} \end{array}\right)$$

Notons que pour les générateurs σ_1 et σ_{n-1}, on obtient :

$$\varphi_{red}(\sigma_1) = \left(\begin{array}{cc|c} -t & 1 & 0 \\ 0 & 1 & 0 \\ \hline 0 & 0 & I_{n-i-2} \end{array}\right), \varphi_{red}(\sigma_{n-1}) = \left(\begin{array}{c|cc} I_{n-i-2} & 0 & 0 \\ \hline 0 & 1 & 0 \\ 0 & t & -t \end{array}\right)$$

Propriétés de la représentation de Burau réduite

Nous avons vu que la représentation de Burau est décomposable en une somme directe de deux représentations. Nous identifierons la matrice d'une tresse $\omega \in B_n$ associée à cette décomposition par $\varphi'(\omega)$, elle est de la forme :

$$\varphi'(\omega) = \left(\begin{array}{c|c} \varphi_{red}(\omega) & 0 \\ \hline 0 & 1 \end{array}\right)$$

La matrice $\varphi'(\omega)$, avec $\omega \in B_n$ est un conjugué de la matrice $\varphi(\omega)$:

$$\varphi(\omega) = c^{-1} \cdot \varphi'(\omega) c \tag{3.5}$$

où

$$c = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & t & 0 & \dots & 0 \\ 1 & t & t^2 & \dots & 0 \\ 1 & t & t^2 & \dots & 0 \\ 1 & t & t^2 & \dots & t^{n-1} \end{pmatrix}.$$

Nous utilisons les propriétés des matrices de Burau présentées à la section 3.3.1 ainsi que l'équation 3.5 pour trouver des propriétés des matrices de Burau réduites.

Lemme 18 *Soit une matrice $A = [a_{ij}] \in GL_n$ appartenant à l'image de la représentation de Burau φ'. Alors, on a*

$$A \cdot \begin{pmatrix} 1 & 1+t & \dots & 1+t+\dots+t^{n-1} \end{pmatrix}^T = \begin{pmatrix} 1 & 1+t & \dots & 1+t+\dots+t^{n-1} \end{pmatrix}^T \quad (3.6)$$

Preuve: Nous avons vu que, pour toute matrice de Burau $\varphi(\omega)$ d'une tresse $\omega \in B_n$, $\varphi(\omega) \cdot \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T = \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T$.

Ainsi :

$$\begin{aligned} \varphi(\omega) \cdot \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T &= \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T \\ c^{-1} \cdot \varphi'(\omega) \cdot c \cdot \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T &= \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T \\ \varphi'(\omega) \cdot c \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T &= c \cdot \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T \\ \varphi'(\omega) \cdot \begin{pmatrix} 1 & 1+t & \dots & 1+t+\dots+t^{n-1} \end{pmatrix}^T &= \begin{pmatrix} 1 & 1+t & \dots & 1+t+\dots+t^{n-1} \end{pmatrix}^T \end{aligned}$$

■

Lemme 19 *Soit une matrice $A = [a_{ij}] \in GL_n$ appartenant à l'image de la représentation de Burau φ'. Alors, on a*

$$\begin{pmatrix} 0 & \dots & 0 & t^n \end{pmatrix} \cdot A = \begin{pmatrix} 0 & \dots & 0 & t^n \end{pmatrix} \quad (3.7)$$

Preuve: Nous avons vu que, pour toute matrice de Burau $\varphi(\omega)$ d'une tresse $\omega \in B_n$, $\begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix} \cdot \varphi(\omega) = \begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix}$.

Ainsi :

$$\begin{aligned} \begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix} \cdot \varphi(\omega) &= \begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix} \\ \begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix} \cdot c^{-1} \cdot \varphi'(\omega) \cdot c &= \begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix} \\ \begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix} \cdot c^{-1} \cdot \varphi'(\omega) &= \begin{pmatrix} t & t^2 & \dots & t^n \end{pmatrix} \cdot c^{-1} \\ \begin{pmatrix} 0 & \dots 0 & t^n \end{pmatrix} \cdot \varphi'(\omega) &= \begin{pmatrix} 0 & \dots 0 & t^n \end{pmatrix} \end{aligned}$$

■

3.3.4 Polynômes caractéristiques

Le polynôme d'Alexander d'une tresse $a \in B_n$, noté $P_a(t)$ est simplement

$$P_a(x) = \det(x \cdot Id - \varphi(a)),$$

il s'agit du polynôme caractéristique de la matrice de Burau de la tresse a.

Exemple 20 *Toujours pour $w = \sigma_1\sigma_3\sigma_2$, la tresse de l'exemple 14, calculons le polynôme d'Alexander associé.*

On a la représentation de Burau suivante :

$$\varphi(w) = \begin{pmatrix} 1-t & -t(-1+t) & t^2 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1-t & 0 & t \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Et donc

$$\begin{aligned} P_w(x) &= \det(x \cdot Id - \varphi(w)) \\ &= x^4 - x^3 + x^3t - tx^2 + x^2t^2 - t^2x + t^3x - t^3 \\ &= x^4 + (t-1)x^3 + (t^2-t)x^2 + (t^3-t^2)x - t^3 \end{aligned}$$

.

Pour toute tresse $\omega \in B_n$, on a la relation suivante entre le polynôme caractéristique de la représentation $\varphi(\omega)$ et le polynôme caractéristique de la représentation $\varphi_{red}(\omega)$:

$$\det(x \cdot Id - \varphi(w)) = (x-1) \cdot \det(x \cdot Id - \varphi_{red}(w)) \tag{3.8}$$

On a déjà vu à la section 3.3 que toute matrice de Burau a 1 comme valeur propre. L'équation 3.8 illustre bien le fait que la représentation de Burau est équivalente à la somme directe d'une représentation de dimension 1 et de la représentation de Burau réduite.

3.4 Représentation de Lawrence-Krammer

Avec la représentation de Burau, celle de Lawrence-Krammer est l'une des plus interessantes. En 1990, Ruth Lawrence a introduit une famille de représentations de B_n en utilisant des constructions topologiques. Quelques années plus tard, Dan Krammer en a donné une formulation algébrique et a montré que cette dernière était fidèle pour $n = 4$. En 2001, Bigelow (Bigelow, 2001) a étendu ce dernier résultat à tout n. Ainsi, on a prouvé que les groupes de tresses sont linéaires, c'est-à-dire qu'il existe une représentation fidèle de B_n pour toute valeur de n. On appelle aujourd'hui cette représentation la *représentation de Lawrence-Krammer*. Plus précisement, on a un morphisme de groupe

$$\mathcal{K} : B_n \to GL_m(\mathbb{Z}[t^{\pm 1}, q^{\pm 1}])$$

avec $m = \frac{n(n-1)}{2}$. Nous appellerons la matrice $\mathcal{K}(\omega)$, pour une tresse $\omega \in B_n$, *matrice de Krammer*. Pour définir la matrice de Krammer d'un générateur σ_i, nous définissons l'action de la représentation $\mathcal{K}$ sur un module V de rang m dont la base est $\{x_{ij}\}_{i \leq i < j \leq n}$. La base est en fait l'ensemble des transpositions de S_n. Une *transposition* est une permutation qui échange seulement deux éléments, non nécessairement consécutifs, il y en a bien $\binom{n}{2}$. La transposition qui échange les éléments i et j est dénotée $s_{i,j}$.

Définition de la représentation de Lawrence-Krammer :

$$\mathcal{K}(\sigma_k)(x_{ij}) = \begin{cases} tq^2 x_{k,k+1} & \text{si } i = k \text{ et } j = k+1 \\ (1-q)x_{i,k} + q x_{i,k+1} & \text{si } j = k \text{ et } i < k \\ x_{ik} + tq^{k-i+1}(q-1)x_{k,k+1} & \text{si } j = k+1 \text{ et } i < k \\ tq(q-1)x_{k,k+1} + q x_{k+1,j} & \text{si } i = k \text{ et } k+1 < j \\ x_{kj} + (1-q)x_{k+1,j} & \text{si } i = k+1 \text{ et } k+1 < j \\ x_{ij} & \text{si } i < j < k \text{ ou } k+1 < i < j \\ x_{ij} + tq^{k-i}(q-1)^2 x_{k,k+1} & \text{si } i < k < k+1 < j \end{cases}$$

Exemple 21 *Voyons deux exemples pour la représentation* $\mathcal{K} : B_3 \to GL_3(\mathbb{Z}[t^{\pm 1}, q^{\pm 1}])$.

$$\mathcal{K}(\sigma_1) = \begin{pmatrix} tq^2 & 0 & 0 \\ tq(q-1) & 0 & q \\ 0 & 1 & 1-q \end{pmatrix}$$

$$\mathcal{K}(\sigma_2) = \begin{pmatrix} 1-q & q & 0 \\ 1 & 0 & tq - 1q^2 \\ 0 & 0 & tq^2 \end{pmatrix}$$

3.4.1 Propriétés de la représentation

Lemme 22 *Soit une tresse* $\omega \in B_n$ *dont la forme normale à gauche est* $\Delta_n^k x_1 x_2 ... x_l$, *où les* x_i *sont des tresses simples et soit* δ *le nombre minimal de générateurs d'Artin dans le mot* ω, *on a les bornes suivantes sur les valeurs de la matrice* $\mathcal{K}(\omega)$:

1. *Le degré en* t *est minoré par* k *et majoré par* $k + l$.
2. *Le degré en* q *est minoré par* $2(n-1) \cdot min(0, k) + n - 2$ *et est majoré par* $2(n-1) \cdot \max(k + l, k) + n - 2$.
3. *Les coefficients de chaque valeur (polynôme en* q *et* t *) dans la matrice de Krammer sont majorés par* 2^δ.

3.4.2 Inverser la représentation

Cheon et Jun (Jung Hee Cheon, 2003) ont introduit en 2003 un algorithme pour inverser la représentation de Lawrence-Krammer, on présente ici l'idée générale de cet algorithme.

Soit une tresse $\omega \in B_n$ sous forme normale, et soit la matrice de Krammer $\mathcal{K}(\omega)$ qui lui est associée. Pour trouver la tresse originale ω sous sa forme normale à gauche, c'est-à-dire $\omega = \Delta_n^k x_1 x_2 ... x_l$, où les x_i sont des tresses simples, on trouve d'abord la valeur de k avec le lemme 22. On peut ensuite calculer $\mathcal{K}(\omega') = \mathcal{K}(\Delta_n)^{-k}\mathcal{K}(\omega)$ où $\omega' = x_1 x_2 ... x_l$. Nous allons ensuite trouver chacune des tresses simples x_i en partant de la gauche.

Tout d'abord, on remplace les variables t dans la matrice $\mathcal{K}(\omega')$ par 0. Puis on choisit, les éléments de la base $\{x_{ij}\}_{i\leq i<j\leq n}$, pour lesquels la ligne associée, dans la matrice $\mathcal{K}(\omega')$ n'est pas nulle. Ces éléments correspondent chacun à une reflexion. Le produit de ces reflexions constitue une permutation à laquelle est associée une tresse simple qui correspond à x_1. On reprend en remplacant la variable t par 0 dans la matrice $\mathcal{K}(x_1)^{-1}\mathcal{K}(\omega')$, jusqu'à ce que $\mathcal{K}(\omega')$ soit égale à la matrice identité.

CHAPITRE IV

CRYPTOSYSTÈMES BASÉS SUR LE GROUPE DE TRESSES

Introduction

Bien que l'utilisation du groupe de tresses B_n en cryptographie apparaisse déjà en 2000 sous la forme d'un protocole d'échange de clés inspiré du protocole de Diffie-Hellman (EonKyung Lee, 2003), nous allons plutôt nous interesser à un autre protocole introduit en 2001 par Anshel, Anshel, Fisher et Goldfeld (I. Anshel, 2001). Ce protocole, appellé AAFG1, fait intervenir le problème de la conjugaison dans B_n. Les auteurs prétendent que le protocole est sécuritaire dans la mesure où le problème de conjugaison est difficilement résoluble. En fait, ce protocole est un raffinement d'un protocole déjà suggéré en 1999 (I. Anshel, 1999). Dans ce dernier article, les auteurs ne font pas allusion au groupe B_n mais utilisent plutôt des monoïdes quelconques.

D'autres protocoles ont été proposés depuis, mais on ne présentera que les deux protocoles ci-haut mentionnés pour leur importance historique du point de vue de la cryptographie avec les groupes de tresses.

4.1 Adaptation du protocole de Diffie-Hellman au contexte des groupes de tresses

On a vu à la section 1.2.2 que le protocole de Diffie-Hellman, basé sur l'exponentiation modulo un entier, permet un échange de clés sécuritaire. Notre premier protocole en est inspiré. L'idée générale est d'utiliser le groupe de tresses B_n et deux sous-groupes

de ce dernier qui commutent entre eux. En supposant n pair et $m = \frac{n}{2}$, on utilise le sous-groupe de B_n dont les éléments ne sont formés que par des croisements sur les m premiers brins, appellons-le GB_n. L'autre sous-groupe de B_n utilisé est généré par des croisements sur les m derniers brins, on l'appelle DB_n. Les deux sous-groupe GB_n et DB_n commutent, on s'en convainc facilement avec la relation 2.1.

Construction de la clé commune

On suppose que deux personnes, que nous nommons, comme il est coutume, Alice et Bob, désirent s'échanger une clé secrète. On associe à chacune un des deux sous-groupes de B_n qu'on vient de définir : Alice utilise le sous-groupe GB_n et Bob utilise le sous-groupe DB_n. Les deux personnes se choisissent, chacun de leur côté, une clé secrète dans leur sous-groupe respectif. Ainsi, Alice choisit une clé $s \in GB_n$ et Bob choisit une clé $r \in DB_n$. Par la suite, Alice et Bob choisissent ensemble une clé publique $p \in B_n$.

Les deux parties conjuguent ensuite la clé publique avec leur clé secrète respective : Alice calcule $p' = sps^{-1}$ et Bob calcule $p'' = rpr^{-1}$. Ces deux derniers conjugués sont chacun échangés à l'autre partie, et chacune des deux personnes utilise le conjugué reçu pour calculer un nouveau conjugué avec sa clé secrète respective. C'est-à-dire que Alice calcule $t_A = sp''s^{-1}$ et Bob calcule $t_B = rp'r^{-1}$.

On a ainsi construit une clé commune secrète puisque

$$\begin{aligned} t_A &= sp''s^{-1} = srpr^{-1}s^{-1} \\ &= rsps^{-1}r^{-1} = rp'r^{-1} \\ &= t_B \end{aligned}$$

La sécurité du protocole est basée sur le problème de recherche du conjugueur. La connaissance de la clé publique p et des deux conjugués p' et p'' ne suffit pas pour pouvoir construire la clé commune t_A, on doit connaître les clés privées r et s.

4.2 Protocole algébrique d'échange de clés

On présente maintenant un protocole algébrique d'échange de clés basé sur les calculs dans des monoïdes quelconques. On insiste plus sur les propriétés essentielles aux calculs que sur la nature particulière des éléments du monoïde. Le protocole est développé dans un contexte général abstrait. Nous verrons plus loin un raffinement de ce protocole en suggérant l'utilisation de groupes et de fonctions précis.

4.2.1 Protocole

Le protocole permet à deux personnes de construire une clé commune secrète à distance, les deux personnes sont décrites par les prénoms Alice et Bob. On utilise deux monoïdes U et V et trois fonctions sur ces monoïdes :

$$\gamma : U \times U \to V$$

$$\alpha : U \times V \to V$$

$$\beta : U \times V \to V$$

Ces fonctions doivent posséder les propriétés suivantes :

1. Pour tout élément x, y_1 et $y_2 \in U$,
   $\gamma(x, y_1 \cdot y_2) = \gamma(x, y_1) \cdot \gamma(x, y_2)$
2. Pour tout élément $x, y \in U$,
   $\alpha(x, \gamma(y, x)) = \beta(y, \gamma(x, y))$
3. Soient $y_1, y_2, ..., y_k \in U$ et $\gamma(x, y_1), \gamma(x, y_2), ..., \gamma(x, y_k)$ des éléments rendus publics alors que l'élément x est gardé secret. Alors, il est impossible de déterminer x.

Construction de la clé commune

Pour débuter, on attribue à chacune des deux personnes, Alice et Bob, des sous-monoïdes de U respectifs S_A et S_B en supposant que S_A est généré par les éléments $s_1, s_2, ..., s_m$ et S_B par les éléments $t_1, t_2, ..., t_n$.

Chacune des deux personnes choisit ensuite un élément secret dans son sous-monoïde. Ainsi Alice choisit un élément secret $a \in S_A$ et Bob choisit un élément secret $b \in S_B$. Chaque personne transmet à son correspondant, la suite des résultats de la fonction γ, avec la clé secrète comme première variable et chacun des générateurs de son sous-monoïde respectif comme deuxième variable :

$$\gamma(a, t_1), ..., \gamma(a, t_n) \text{ pour Alice,}$$

$$\gamma(b, s_1), ..., \gamma(b, s_m) \text{ pour Bob}$$

On remarque que, par la propriété 3, même si les éléments $\gamma(a, t_i)$ et $\gamma(b, s_i)$ sont transmis par un canal non sécuritaire, les éléments a et b sont gardés secrets.

Alice peut calculer $\gamma(b, a)$ avec les éléments $\gamma(b, s_i)$ via la propriété 1, et puisque a est généré par les éléments s_i, et donc calculer ensuite

$$\alpha(a, \gamma(b, a)).$$

Bob peut calculer $\gamma(a, b)$ avec les éléments $\gamma(a, t_i)$ par la propriété 1 et puisque b est généré par les éléments t_i, et donc calculer ensuite

$$\beta(b, \gamma(a, b))$$

Par la propriété 2, on a que Alice et Bob ont construit une clé commune K puisque

$$K = \alpha(a, \gamma(b, a)) = \beta(b, \gamma(a, b))$$

4.2.2 Utilisation du protocole avec la conjugaison

On propose d'utiliser l'algorithme de la section 4.2 avec, comme fonction γ, la conjugaison, pour répondre à la propriété 3. En effet, on sait que le problème de conjugaison est pour plusieurs groupes réputé difficile. Nous avons vu, dans le chapitre 2, quelques problèmes difficiles dans certains groupes, dont les groupes de tresses. Le problème de racine p ième est le seul qui n'implique pas la conjugaison, mais il ne pourrait être utilisé

ici puisque la fonction a comme domaine $(G \times \mathbb{Z})$, où G est un groupe et que le protocole nécessite un domaine de la forme $(U \times U)$ où U est un monoïde.

Voyons donc comment la fonction de conjugaison peut être appliquée ici. Un même groupe G, remplace à la fois les deux monoïdes U et V dans le protocole initial ($U = V = G$), et on définit la fonction γ comme :

$$\gamma(x, y) = x^{-1}yx \, .$$

On doit ainsi, pour respecter la propriété 2, trouver des fonctions α et β telles que $\alpha(x, y^{-1}xy) = \beta(y, x^{-1}yx)$. Dans notre cas, ce sera les fonctions :

$$\alpha(u, v) = u^{-1}v$$

et

$$\beta(u, v) = v^{-1}u \, .$$

Problème de l'appartenance

La sécurité du cryptosystème dépend du problème de conjugaison, mais également du problème d'appartenance. En effet, même si on réussit à trouver un élément conjugueur, celui-ci peut être différent de la clé secrète recherchée. Par exemple, avec la clé secrète a et les données $a^{-1}t_i a$ rendues publiques, une tierce personne peut, en tentant de trouver la valeur de a, trouver un autre conjugueur a' pour lequel $a^{-1}t_i a = a'^{-1}t_i a'$ pour tout i. Ce conjugueur, différent de la clé secrète a, doit faire partie du sous-groupe ayant servi à générer la clé secrète (S_A ou S_B). Dans le cas contraire, il serait impossible de trouver la clé commune, nous allons montrer pourquoi.

On sait que les éléments a et b sont respectivement inclus dans les groupes S_A et S_B, deux sous-groupes de G, mais l'élément a' (ou b') trouvé par l'algorithme peut ne pas faire partie de ces sous-groupes.

Nous allons voir que, si a' ne fait pas partie du sous-groupe S_A, duquel l'élément secret a fait partie, alors la clé commune construite avec a' est différente de celle construite

avec a, et cela même si a' est un conjugueur de x et y. On supppose qu'on résolve le problème de conjugaison avec les données du cryptosystème. On a alors trouvé deux éléments secrets a' et b'. Ces deux éléments ne sont peut-être pas les mêmes que les clés secrètes a et b, mais on sait qu'ils permettent de conjuguer les mêmes éléments que ces clés secrètes c'est-à-dire :

$$a^{-1}t_i a = (a')^{-1}t_i a' \text{ avec } i = 1, ..., n$$

et

$$b^{-1}s_i b = (b')^{-1}s_i b' \text{ avec } i = 1, ..., m\ .$$

Supposons $a' = c_b a$ et $b' = c_a b$ et analysons un peu ces deux facteurs c_a et c_b, qu'on suppose être différents de 1.

Pour $i = 1, ..., n$, on a

$$\begin{aligned} a^{-1}t_i a &= (a')^{-1}t_i a' \\ &= (c_b a)^{-1}t_i c_b a \\ &= a^{-1}c_b^{-1}t_i c_b a \end{aligned}$$

Ce qui veut dire que l'élément c_b et l'ensemble des t_i commutent. De la même façon, pour $i = 1, ..., m$ l'élément c_a commute avec chacun des s_i. Simulons maintenant la construction d'une clé commune K' avec ces deux éléments a' et b' : On a $K = a^{-1}b^{-1}ab$, et donc

$$\begin{aligned} K' &= (a')^{-1}(b')^{-1}a'b' \\ &= (c_b a)^{-1}(c_a b)^{-1}c_b a c_a b \\ &= a^{-1}c_b^{-1}b^{-1}c_a^{-1}c_b a c_a b \\ &= a^{-1}b^{-1}c_b^{-1}c_a^{-1}c_b c_a a b \end{aligned}$$

Pour que $K' = K$, les deux éléments c_a et c_b doivent commuter. La façon la plus simple de s'en assurer, est de montrer que c_a est généré par les générateurs t_i et c_b est généré par les générateurs s_i, de cette façon, puisque c_a commute avec chaque élément s_i et que

c_b commute avec chaque élément t_i, c_a commuterait avec c_b. On a donc les conditions suivantes sur les éléments a' et b' :

$$a' = c_b a \text{ où } a \in S_A \text{ et } c_b \in S_A \text{ donc } a' \in S_A$$

$$b' = c_a b \text{ où } b \in S_B \text{ et } c_a \in S_B \text{ donc } b' \in S_B$$

Donc, après avoir trouvé un élément conjugueur une personne qui tente d'attaquer le cryptosystème devrait également résoudre le problème de l'appartenance.

Groupes à utiliser

Une prochaine étape consiste à choisir un groupe pour lequel la mise en place de cet algorithme est simple et efficace. Pour circonscrire le choix d'un tel groupe, on impose les contraintes supplémentaires suivantes :

1. On aimerait que le groupe soit bien connu, et qu'en particulier, le problème de recherche du conjugueur, soit réputé difficile. Idéalement, on aimerait avoir une preuve que le problème de recherche du conjugueur dans le groupe considéré a été montré irrésoluble par un algorithme déterministe en temps moindre qu'exponentiel.
2. Le problème du mot dans le groupe recherché devrait être résoluble en temps polynomial par un algorithme déterministe.
3. On doit pouvoir « maquiller »les éléments du groupe recherché de façon à ce qu'il soit impossible de retrouver un élément x en ne connaissant que les éléments a et $x^{-1}ax$. Ainsi, lorsque le groupe recherché est défini en termes de générateurs et relations, certaines de ces relations devraient être courtes pour faciliter ce maquillage des éléments. (On utilise parfois le terme *diffusion*, pour décrire ce phénomène.)

En général, la condition 1 est très difficile à satisfaire. Dans un contexte assez général, prouver qu'il n'existe aucun algorithme déterministe polynomial permettant de résoudre

le problème de conjugaison est souvent équivalent à prouver que prouver $P \neq NP$. À l'instar du problème de factorisation de grands nombres, utilisé dans le cryptosystème RSA, ce type de problème est étudié depuis plusieurs décennies. Un algorithme déterministe permettant de le résoudre en temps polynomial n'a pas encore été trouvé, pas plus qu'une preuve de non-existence d'un tel algorithme.

La condition 2 est nécessaire pour que les calculs puissent s'effectuer dans un temps relativement court.

La condition 3 permet d'éviter quelques attaques triviales. Mentionnons aussi qu'autant que le maquillage, la longueur du conjugué (longueur du mot) est importante et doit donner le moins d'indications possibles sur la longueur de l'élément conjugueur.

Pour l'instant, le groupe proposé, qui répond le mieux à ces conditions semble être le groupe de tresses B_n. Il a été beaucoup étudié depuis plusieurs années dans différentes branches des mathématiques ainsi qu'en physique. Cette grande variété d'approches pour étudier le groupe de tresses renforce l'impression que la difficulté du problème de conjugaison est bien réelle. Cependant, on a toujours pas de preuve qu'il n'existe pas d'algorithme déterministe résolvant le dit problème en temps polynomial. Le protocole est illustré par la figure 4.1, mais nous verrons également, dans la prochaine section, un exemple de la réalisation de ce protocole à l'aide du groupe B_n.

4.3 Exemple d'utilisation du protocole avec le groupe de tresses

On propose ici un exemple pour illustrer le cryptosystème AAFG1 avec le groupe de tresses B_n. Tout d'abord, Alice et Bob ont chacun un sous-groupe de B_n qui leur est associé :

$$S_A = < \sigma_1\sigma_2, \sigma_3\sigma_1^{-1}, \sigma_2 >$$

$$S_B = < \sigma_2\sigma_1^{-1}, \sigma_3^{-1}, \sigma_1\sigma_3 >$$

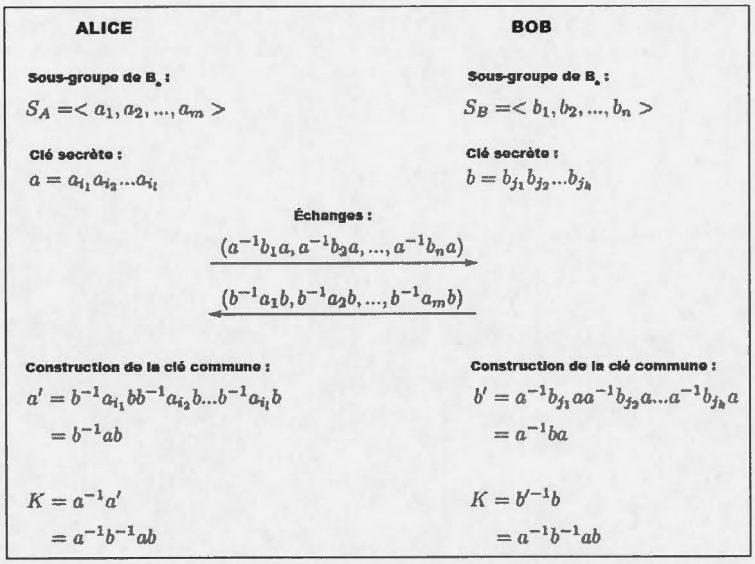


FIG. 4.1 Utilisation du protocole avec B_n

Alice choisit une clé secrète $a \in S_A$:

$$\begin{aligned} a &= a_3 a_2 \\ &= \sigma_2 \sigma_3 \sigma_1^{-1} \\ &= \Delta^{-1} \cdot \sigma_2 \sigma_3 \sigma_2 \cdot \sigma_2 \sigma_1 \end{aligned}$$

Bob choisit une clé secrète $b \in S_B$:

$$\begin{aligned} b &= b_1 b_2 b_3 \\ &= \sigma_2 \sigma_1^{-1} \sigma_3^{-1} \sigma_1 \sigma_3 \\ &= \sigma_2 \end{aligned}$$

Alice transmet à Bob le conjugué de chacun des générateurs de S_B avec sa clé secrète a :

$$\begin{aligned} b'_1 &= a^{-1}b_1a \\ &= (\sigma_2\sigma_3\sigma_1^{-1})^{-1}(\sigma_2\sigma_1^{-1})(\sigma_2\sigma_3\sigma_1^{-1}) \\ &= \Delta^{-1}\cdot\sigma_2\sigma_1\sigma_3\cdot\sigma_1\sigma_2\sigma_3 \end{aligned}$$

$$\begin{aligned} b'_2 &= a^{-1}b_2a \\ &= (\sigma_2\sigma_3\sigma_1^{-1})^{-1}(\sigma_3^{-1})(\sigma_2\sigma_3\sigma_1^{-1}) \\ &= \Delta^{-1}\cdot\sigma_1\sigma_2\sigma_3\sigma_2\cdot\sigma_2 \end{aligned}$$

$$\begin{aligned} b'_3 &= a^{-1}b_3a \\ &= (\sigma_2\sigma_3\sigma_1^{-1})^{-1}(\sigma_1\sigma_3)(\sigma_2\sigma_3\sigma_1^{-1}) \\ &= \Delta^{-2}\cdot\sigma_1\sigma_2\sigma_1\cdot\sigma_2\sigma_1\sigma_3\cdot\sigma_1\sigma_2\sigma_3\sigma_2\sigma_1\cdot\sigma_1\sigma_2\sigma_3 \end{aligned}$$

Bob transmet à Alice le conjugué de chacun des générateurs de S_A avec sa clé secrète b :

$$\begin{aligned} a'_1 &= b^{-1}a_1b \\ &= (\sigma_2)^{-1}(\sigma_1\sigma_2)(\sigma_2) \\ &= \Delta^{-1}\cdot\sigma_1\sigma_2\sigma_3\sigma_1\cdot\sigma_1\sigma_2\cdot\sigma_2 \end{aligned}$$

$$\begin{aligned} a'_2 &= b^{-1}a_2b \\ &= (\sigma_2)^{-1}(\sigma_3\sigma_1^{-1})(\sigma_2) \\ &= \Delta^{-1}\cdot\sigma_1\sigma_2\sigma_3\sigma_2\cdot\sigma_3\sigma_2 \end{aligned}$$

$$\begin{aligned} a'_3 &= b^{-1}a_3b \\ &= (\sigma_2)^{-1}(\sigma_2)(\sigma_2) \\ &= \sigma_2 \end{aligned}$$

Alice reconstruit sa clé secrète en utilisant les générateurs a'_i envoyés par Bob :

$$\begin{aligned} a' &= a'_3a'_2 \\ &= \Delta^{-1}\cdot\sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\cdot\sigma_3\sigma_2 \end{aligned}$$

Bob recontruit sa clé secrète en utilisant les générateurs b'_i envoyés par Alice :

$$\begin{aligned} b' &= b'_1 b'_2 b'_3 \\ &= \Delta^{-1} \cdot \sigma_2\sigma_3\sigma_2\sigma_1 \cdot \sigma_1\sigma_2\sigma_3 \end{aligned}$$

Alice peut maintenant construire la clé K_A en calculant $a^{-1}a'$

$$\begin{aligned} K_A &= a^{-1}a' \\ &= (\Delta^{-1} \cdot \sigma_2\sigma_3\sigma_2 \cdot \sigma_2\sigma_1)^{-1}(\Delta^{-1}\sigma_1\sigma_2\sigma_1\sigma_3\sigma_2 \cdot \sigma_3\sigma_2) \\ &= \Delta^{-1} \cdot \sigma_2\sigma_3\sigma_2 \cdot \sigma_2\sigma_3\sigma_2 \end{aligned}$$

Bob peut maintenant construire la clé K_B en calculant $(b')^{-1}b$

$$\begin{aligned} K_B &= (b')^{-1}b \\ &= (\Delta^{-1} \cdot \sigma_2\sigma_3\sigma_2\sigma_1 \cdot \sigma_1\sigma_2\sigma_3)^{-1}(\sigma_2) \\ &= \Delta^{-1} \cdot \sigma_2\sigma_3\sigma_2 \cdot \sigma_2\sigma_3\sigma_2 \end{aligned}$$

Alice et Bob ont construit une clé commune puisque $K_A = K_B$, en effet :

$$\begin{aligned} K_A &= a^{-1}a' \\ &= a^{-1}b^{-1}a_2bb^{-1}a_1bb^{-1}a_1b \\ &= a^{-1}b^{-1}a_2a_1a_1b \\ &= a^{-1}b^{-1}ab \end{aligned}$$

et

$$\begin{aligned} K_B &= (b')^{-1}b \\ &= (b'_1b'_2b'_3)^{-1}b \\ &= (a^{-1}b_1aa^{-1}b_2aa^{-1}b_3a)^{-1}b \\ &= (a^{-1}b_1b_2b_3a)^{-1}b \\ &= (a^{-1}ba)^{-1}b \\ &= a^{-1}b^{-1}ab \end{aligned}$$

L'exemple précédent nous permet de voir, comment la forme normale permet de cacher l'information, par exemple, dans le cas d'un conjugué sous forme normale, l'élément conjugueur est indistinguable. Bien sûr, notre exemple utilise des éléments de petite taille dans B_n, nous verrons plus loin que la forme normale d'un élément conjugué peut révéler de l'information sur l'élément conjugueur lorsque les éléments utilisés contiennent un plus grand nombre de générateurs.

Faiblesses du protocole avec le groupe de tresses

Génération des clés

Dehornoy (Dehornoy, 2000) fait mention de l'importance de la méthode employée pour générer les clés. Il remarque que, en générant les clés de façon aléatoire, dans le cas d'une tresse conjuguée, disons $x^{-1}ax$, la forme normale de la tresse x contient un nombre assez grand de préfixes commun à la forme normale de la tresse $x^{-1}ax$. Par exemple, pour une clé de longueur canonique 200 dans le groupe B_{100}, le nombre moyen de facteurs canoniques identiques entre la clé et son conjugué est de 197. La génération des clés ne devrait donc pas être totalement aléatoire.

Longueur des générateurs

Une attaque possible sur ce cryptosystème serait une attaque sur la longueur (*length attack*). Il s'agit de tenter de trouver l'élément conjugueur à partir du conjugué en le conjuguant avec des générateurs (de S_A ou S_B selon le cas) qui permettent de diminuer sa longueur. Pour contrer cette attaque, on doit tout d'abord avoir un paramètre N (nombre de brins) suffisamment grand, et, les générateurs de S_A et S_B devraient être relativement courts.

Paramètres suggérés

L'article dans lequel est introduit le cryptosystème propose certains paramètres pour assurer une sécurité satisfaisante. On suggère de travailler avec un groupe de tresses ayant un minimum de 80 brins (B_{80}). Les sous-groupes S_A et S_B devraient être générés par 20 éléments de B_{80}, et, chacun de ces générateurs devrait être composé de 5 à 10 générateurs d'Artin. Les générateurs des deux sous-groupes (a_i ou b_i) devraient être choisis de façon à ce que chaque générateur d'Artin du groupe B_{80} se retrouve dans au moins un des générateurs de S_A et de S_B. Finalement, les deux clés privées a et b devraient être formées de 100 générateurs (a_i ou b_i).

CHAPITRE V

ATTAQUE DU CRYPTOSYSTÈME AAFG1

Introduction

Quelques attaques ont été proposées contre le cryptosystème présenté à la section 4.2, elles tentent de résoudre le problème de conjugaison dans le groupe de tresses. Ces attaques permettent de remettre sérieusement en question l'efficacité du cryptosystème et nous poussent à fixer de nouveaux paramètres si ce cryptosystème devait un jour être utilisé. Nous nous concentrons ici sur une attaque en particulier qui utilise la théorie de la représentation des groupes, mais avant nous donnons un bref apercu de deux autres attaques.

5.1 Attaque basée sur le *Super Summit Set*

Nous avons vu, à la section 2.5, que le Super Summit Set d'une tresse $\omega \in B_n$ était un sous-ensemble de la classe de conjugaison de ω pour lequel la complexité des tresses est minimale. Afin de trouver un élément conjugueur pour deux tresses conjuguées, disons $x, y \in B_n$, on trouve d'abord, pour chacune des deux tresses, un conjugué contenu dans le Super Summit Set en effectuant l'opération de cyclage. Notons ces deux conjugués x' et y'. Rappellons que $SSS(x) = SSS(y)$ car x et y sont conjuguées. Puisque x' et y' font partie de $SSS(x)$, on sait que ces deux tresses sont conjuguées par une tresse simple. Les tresses simples constituent un sous-ensemble fini de B_n et donc, on peut, en conjuguant x' par chacune des tresses simples, déterminer laquelle est le conjugueur de

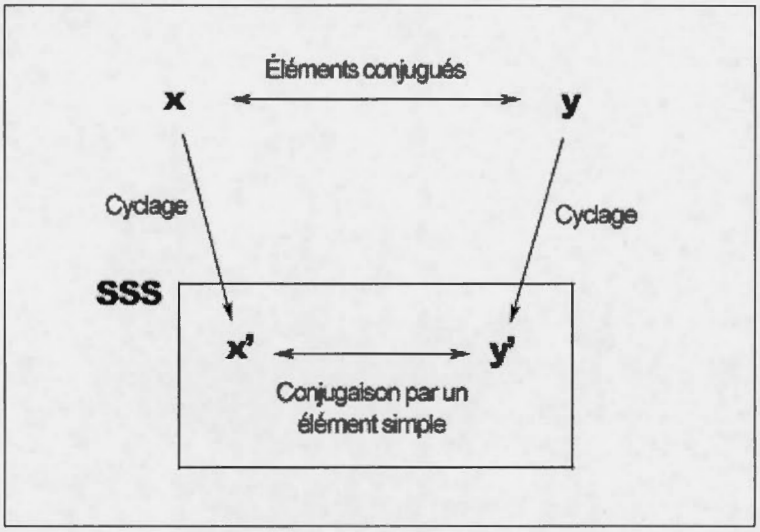


FIG. 5.1 Résolution du problème de conjugaison en utilisant le Super Summit Set

x' et y'. Avec ce conjugueur et les éléments conjugueurs utilisés lors de l'opération de cyclage pour passer de x à x' et de y à y', (Rappellons que x et $\delta_+(x)$ sont conjuguées.) on peut déterminer le conjugueur de x et y.

Même si l'ensemble des tresses simples est fini, sa cardinalité ($n!$ pour B_n) constitue un problème si on veut utiliser cette attaque avec les paramètres suggérés dans (I. Anshel, 2001). Néanmoins, l'idée d'utiliser le Super Summit Set est interessante et peut constituer une base pour une attaque plus prometteuse.

5.2 Attaque basée sur la longueur des clés

Une attaque basée sur la longueur des conjugués est une approche heuristique au problème de conjugaison. Étant donné deux tresses conjuguées x et y, pour lesquelles la longueur de x est plus grande que celle de y, on tente de conjuguer la tresse x avec chacun des générateurs de B_n jusqu'à ce que la tresse résultante ait une longueur plus petite que celle de x. On recommence ensuite l'opération avec cette tresse résultante, puis ainsi de suite jusqu'à obtenir une tresse de longueur égale à la longueur de la tresse

y. Si la tresse finale est égale à y, on construit le conjugué de x et y avec les générateurs utilisés, sinon, on recommence l'opération avec différents générateurs.

Les auteurs du cryptosystème AAFG1 suggèrent certains paramètres sur la longueur des clés pour contrer ce genre d'attaque mais l'idée d'utiliser la longueur des conjugués pourrait être combinée à d'autres techniques pour constituer une attaque plus efficace.

5.3 Attaque basée sur la théorie de la représentation des groupes

Nous avons vu au chapitre 3, deux représentations du groupe B_n, la représentation de Burau, et celle de Lawrence-Krammer. La représentation de Burau est utilisée dans une attaque contre le cryptosystème AAFG1 dans un article de James Hughes (Hughes, 2002). Nous voyons cette attaque en détail dans la présente section.

Nous avons vu que pour porter atteinte à la sécurité du cryptosystème AAFG1, on doit trouver un moyen de résoudre de façon efficace le problème de conjugaison dans le groupe de tresses. James Hughes propose d'utiliser la théorie des représentations pour résoudre le problème de conjugaison en un temps polynomial. Bien qu'ayant une complexité interessante, l'algorithme ne permet pas de trouver à coup sûr la solution, les résultats des essais effectués sont toutefois suffisants pour mettre en doute la sécurité du cryptosystème.

L'idée de l'attaque est de transporter le problème de recherche du conjugueur du groupe de tresses vers le groupe linéaire en utilisant la représentation de Burau. Cette représentation n'est pas fidèle et c'est en partie pour cette raison que l'algorithme proposé est de nature probabiliste.

Algorithme

Considérons les données de l'exemple 4.3. Soit un ensemble de paires de conjugués dans le groupe B_n ayant tous le même conjugueur, et soit la représentation de Burau définie

à la section 3.3 :

$$\varphi : B_n \to GL_n$$

L'algorithme est divisé en trois principales étapes, illustrées par la figure 5.2

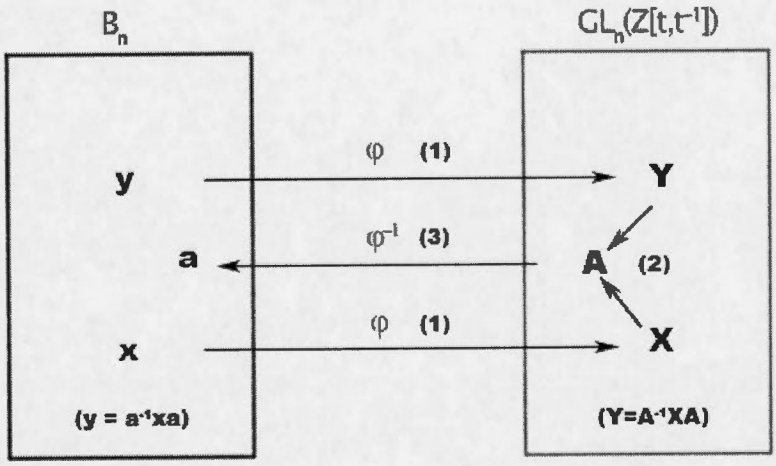


FIG. 5.2 Résolution du problème de conjugaison dans le groupe linéaire

1. Pour chaque paire de conjugués x et y trouver les matrices $\varphi(x)$ et $\varphi(y)$.
2. Construire un système d'équations permettant de trouver une matrice $A \in GL_n$ telle que $\varphi(y) = A\varphi(x)A^{-1}$, pour chaque paire de conjugués x et y.
3. Trouver l'élément $a' \in B_n$ tel que $\varphi(a') = A$.

La première étape se fait facilement en utilisant la définition de la représentation de Burau. Pour la deuxième étape de l'algorithme, on doit construire un système d'équations et le résoudre, c'est ce que nous voyons dans la section suivante.

5.3.1 Problème de conjugaison dans GL_n

Construction d'un système d'éqations (Hughes, 2002)

Pour chaque paire de matrices $\varphi(x) = X, \varphi(y) = Y \in GL_n$, telles que $X = AYA^{-1}$, on veut trouver la valeur de la matrice A. On connaît les valeurs contenues dans X et Y, on peut donc construire un système d'équations de la façon suivante :

Soient les matrices

$$R = XA = \left(r_{ij}\right)$$

et

$$S = AY = \left(s_{ij}\right)$$

Pour tout $i = 1, ..., n$ et $j = 1, ..., n$.

On a que

$$R = S$$

$$\Rightarrow \left(r_{ij}\right) = \left(s_{ij}\right)$$

$$\Rightarrow \forall i = 1, ..., n \text{ et } j = 1, ..., n\ ,\ \ r_{ij} - s_{ij} = 0$$

On ajoute au système d'équations les n^2 équations tirées de cette paire de conjugués.

Les équations précédentes ne suffisent pas à résoudre le système, et, même si c'était le cas, la matrice trouvée pourrait ne pas faire partie de l'image de la représentation de Burau. Pour avoir de plus fortes chances de trouver une matrice de Burau comme matrice solution, on ajoute au système des équations qu'on tire des propriétés des matrices de Burau, vues au chapitre 3. Ainsi, pour la matrice R, on a les n équations

$$\sum_{j=1}^{n} r_{ij} = 1 \text{ pour } i = 1, ..., n$$

et les n équations

$$\sum_{i=1}^{n} r_{ij} t^i = t^j \text{ pour } j = 1, ..., n\ .$$

On utilise les mêmes équations pour la matrice S et on obtient $4n$ nouvelles équations à ajouter au système. On répète ceci pour chaque paire de matrices conjuguées, avant de résoudre le sytème d'équations.

Amélioration du système d'équations

Dans son article, Hughes (Hughes, 2002) utilise seulement les équations des lemmes 15 et 16 pour completer le système d'équations. Nous proposons d'y ajouter quelques équations basées sur les propriétés des matrices de Burau réduites vues à la section 3.3.3.

Tout d'abord, pour la matrice R, on a les n équations

$$\sum_{j=1}^{n} r_{ij} \cdot (1 + t + ... + t^{j-1}) = 1 + t + ... + t^{i-1} \text{ pour } i = 1, ..., n \ ,$$

et l'équation

$$r_{nn} = 1 \ .$$

Ensuite, on peut trouver de nouvelles équations en utilisant le polynôme caractéristique des matrices. En effet, on a que $R = XA = \varphi(xa)$, ainsi, avec l'équation 3.5, on trouve $\varphi'(xa) = c \cdot \varphi(xa) \cdot c^{-1}$. Le polynôme caractéristique de la matrice $\varphi(xa)$ est égal à celui de la matrice $\varphi'(xa)$, on a donc, en comparant chacun des coefficients, n nouvelles équations.

On utilise évidemment les mêmes équations pour la matrice S. L'utilisation d'une procédure, écrite en langage MAPLE, qui permet de générer des équations servant à déterminer la matrice conjuguante de deux matrices de Burau, montre que ces nouvelles équations réduisent le nombre de degrés de liberté. La procédure a été testée sur quelques tresses de courte longueur et l'ajout de ces nouvelles équations diminuait le nombre de degrés de liberté.

5.3.2 Retrouver la tresse à partir de la matrice

Une fois le conjugueur déterminé dans le groupe GL_n, on doit trouver la tresse qui lui correspond, c'est-à-dire, l'élément $a \in B_n$ pour lequel $\varphi(a) = A$. On utilise la méthode présentée à la section 3.3.2. On a vu que la probabilité de trouver l'élément initial avec cet algorithme est assez élevée, dans notre cas, comme le groupe est B_{80}, la probabilité est de $1 - \frac{1}{79}$.

On trouve donc une tresse a telle que $\varphi(a) = A$. Il faut d'abord vérifier que a permet de conjuguer x et y. On calcule d'abord la tresse $a^{-1}ya$ qu'on met sous forme normale et on la compare avec x. Si les deux tresses sont équivalentes, on a trouvé un élément conjugueur. L'élément conjugueur trouvé n'est peut être pas nécessairement le même que la clé secrète, On sait toutefois qu'il permettra de reconstruire la clé commune, à condition de résoudre le problème d'appartenance (section 4.2.2).

5.3.3 Efficacité de l'attaque basée sur la représentation de Burau

Fidélité de la représentation

On a vu que la représentation de Burau n'est pas fidèle. Dans son article, Hughes a tenté de déterminer quel pourcentage des clés utilisées dans un cryptosystème de type AAFG1 seraient affectées par les éléments non triviaux du noyau de la représentation de Burau. Il a déterminé empiriquement que dans environ 96% des cas, la représentation de Burau est fidèle. L'expérience consistait simplement à choisir un certain nombre de tresses selon les paramètres suggérés dans l'article (I. Anshel, 2001), et, pour chacune de ces tresses, disons $a_i \in B_{80}$, vérifier si $\varphi^{-1}(\varphi(a_i)) = a_i$. L'idée de contourner l'attaque en utilisant la non-injectivité de la représentation semble donc peu prometteuse.

Résolution du système d'équations

La résolution du système d'équations (section 3.3.2) permettant de trouver une matrice conjuguée est certainement l'étape la plus problématique de cette attaque. Le système

d'équations qu'on veut résoudre peut ne produire aucune solution. Hughes, pour tester sa méthode a généré, pour chaque attaque simulée, cinq clés privées, selon les paramètres suggérés. Parmi ces cinq clés, il choisit celle dont la matrice du système d'équations semble la plus facile à résoudre, en observant la répartition des variables autour de la diagonale. Il tente ensuite de résoudre le système pour cette clé spécifique. Pour 44 essais, il réussit à trouver la solution 40 fois. À trois reprises, la solution est incorrecte et, une fois, la solution ne peut être déterminée.

L'auteur ne spécifie pas, par contre, si les solutions incorrectesle sont parce qu'elles ne sont pas égales à la clé privée initiale ou parce qu'elles ne sont pas un élément conjugueur. Il aurait été pertinent de le spécifier puisque dans le premier cas, une solution incorrecte pourrait tout simplement signifier que plusieurs éléments conjugueurs sont possibles. Dans le cas où la solution ne peut être trouvée, on aurait pu ajouter des équations au système pour tenter de raffiner la solution, par exemple en utilisant l'équation 3.2.

40 solutions trouvées sur 44 peut paraître être un résultat satisfaisant, mais il ne s'agit que des résultats obtenus pour la clé privée offrant le système d'équations le plus facile à résoudre parmi les cinq clés générées aléatoirement. Pour les autres clés, on peut imaginer des résultats moins concluants.

Les équations que nous proposons d'ajouter au système permettent de réduire le nombre de degrés de liberté, mais rien n'indique que cette amélioration permettrait de résoudre plus facilement le problème de conjugaison. La difficulté de la résolution du système d'équations, paraît donc être l'élément le plus interessant à utiliser pour la génération de clés qui rendraient cette attaque vaine.

5.3.4 Attaque utilisant la représentation de Lawrence-Krammer

On peut envisager l'utilisation d'une autre représentation pour effectuer cette attaque. La représentation de Krammer, étant donné sa fidélité, pourrait être interessante. De plus, on a vu à la section 3.4.2 qu'il était possible d'inverser cette représentation. Dans l'article (Jung Hee Cheon, 2003), Cheon et Jun proposent une attaque sur le crypto-

système basé sur Diffie-Hellman, (section 4.1) qui utilise la représentation de Krammer. Dans ce cryptosystème, les éléments conjugueurs qu'on tente de trouver sont dans deux sous groupes de B_n qui commutent entre eux, cela permet d'ajouter des contraintes au système d'équations lors de la résolution du problème du conjugueur dans le groupe de matrices. Par contre, si on voulait utiliser la représentation de Krammer dans une attaque contre le cryptosystème AAFG1, on devrait trouver d'autres contraintes pour augmenter le nombre d'équations dans le système. De plus, même en résolvant le problème de recherche du conjugueur dans $GL_m(\mathbb{Z}[q^{\pm 1}, t^{\pm 1}])$, la solution n'est pas nécessairement une matrice de Krammer (Budney, 2005).

5.4 Améliorations possibles du cryptosystème AAFG1

L'attaque que nous avons vue dans le chapitre 5, nous montre que la sécurité du cryptosystème AAFG1 peut être mise en jeu si le groupe utilisé admet des représentations. Si une représentation de ce groupe est linéaire et si, en plus, le conjugaison dans l'image de ce groupe est facilement résoluble, le cryptosystème ne peut être considéré sécuritaire. Le groupe de tresses B_n admet une représentation, celle de Burau, cette dernière n'est pas linéaire mais elle permet toutefois de mettre en jeu la sécurité du cryptosystème AAFG1.Voyons quelles seraient les améliorations possibles.

Génération des clés

Les deux attaques que nous avons brièvement présentées au début de ce chapitre ont poussé certains chercheurs à proposer certains paramètres sur les clés utilisées. Par exemple, pour éviter une attaque utilisant le Super Summit Set, on peut tenter de créer des conjugués dont le Super Summit Set est de grande taille. Il est également possible de rendre plus difficile une attaque par la longueur en suggérant certaines longueurs pour les générateurs des sous-groupes de B_n. La réussite de l'attaque utilisant la représentation de Burau dépend de la résolution du système d'équations. On pourrait tenter d'utiliser cette information pour contourner cette attaque. Par exemple la longueur canonique des conjugués, ou bien l'utilisation de générateurs formés sur les premiers et les derniers

brins dans la clé secrète pourraient avoir une influence sur la facilité de résolution du système d'équations. Ces idées sont à étudier, mais on doit garder en tête que chaque attaque apporte un lot de contraintes pour une utilisation sécuritaire du cryptosystème AAFG1 et amenuise ses chances d'être utilisé dans le futur.

Utilisation de groupes différents

Le groupe B_n est étudié depuis relativement longtemps et l'utilisation en cryptographie de certains problèmes dans ce groupe semble pertinente. Différentes attaques, dont celle que nous avons étudiée ici nous permettent de remettre en question l'utilisation du groupe de tresses. Nous avons vu, dans le chapitre 2 des groupes semblables aux groupes de tresses dont certains en sont des généralisations. On peut croire que ces groupes pourraient être utilisés en cryptographie, mais, pour l'instant, nos connaissances sur ces groupes sont moindres que celles sur B_n.

CONCLUSION

Ce travail tente de donner une idée d'une utilisation d'un groupe dans l'élaboration d'un cryptosystème. Nous avons vu comment, à l'aide du groupe de tresses, on pouvait construire un protocole d'échange de clés, plus précisément en utilisant la difficulté de résolution du problème de conjugaison dans ce groupe. Quelques attaques ont été proposées pour tenter de déceler des failles dans la sécurité de ce protocole. L'attaque utilisant la représentation de Burau, sur laquelle nous avons mis l'emphase, s'avère efficace. De plus nous avons pu donner quelques pistes pour l'améliorer. À la lumière de ces attaques, l'utilisation de groupes de tresses dans l'élaboration d'un protocole d'échange de clés, telle que proposée dans (I. Anshel, 2001), est tout à fait douteuse. À défaut d'améliorer le protocole, ou d'utiliser un problème différent du problème de conjugaison, la cryptographie devra utiliser d'autres groupes que B_n pour assurer une sécurité convaincante.

RÉFÉRENCES

Bigelow, S. 2001. « Braid groups are linear », *J. Amer. Math. Soc.*, vol. 14, p. 471–486.

Budney, R. 2005. « On the image of the lawrence-krammer representation », *Journal of Knot Theory and Its Ramifications*, vol. 14, p. 773.

Dehornoy, P. 2000. « Braid based cryptography », *Contemporary Mathematics*, vol. 360, p. 5–33.

EonKyung Lee, J. H. P. 2003. « Cryptoanalysis of the public-key encryption based on braid groups », *Lecture Notes in Computer Science*, vol. 2656, p. 477–490.

Hughes, J. 2002. « A linear algebraic attack on the aafg1 braid group cryptosystem », *Lecture Notes in Computer Science*, vol. 2384, p. 176–189.

I. Anshel, M. Anshel, B. F. 2001. « New key agreement protocol in braid group cryptography », *Lecture Notes in Computer Science*, vol. 2020, p. 13–27.

I. Anshel, M. Anshel, D. G. 1999. « An algebraic method for public-key cryptography », *Mathematical Research Letters*, vol. 6, p. 1–5.

Jung Hee Cheon, B. J. 2003. « A polynomial time algorithm for the braid diffie-hellman conjugacy problem », *Lecture Notes in Computer Science*, vol. 2729, p. 212–225.

Mahlburgh, K. 2004. An overview of braid group cryptography.

Index