REPUBLIQUE DU CAMEROUN

PAIX-TRAVAIL-PATRIE

****

UNIVERSITE DE YAOUNDE 1

****

ECOLE DOCTORALE

REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

****

UNIVERSITY OF YAOUNDE 1

****

POST GRADUATE SCHOOL

Laboratoire d'Algèbre, Géométrie et Applications (LAGA)

THEME :

CALCUL DES QUASI-GROUPES DANS GAP VERS LE CODAGE ALGÉBRIQUE ET LA CRYPTOGRAPHIE

Master en Mathématiques

Option : Algèbre

par

NZETCHUEN MANGAPTCHE Eunisse Lopez

Matricule : 17A2481

Sous l'encadrement de :

Dr. OGADOA AMASSAYOGA

Chargée de Cours

Université de Yaoundé 1

Année Académique 2022/2023

CALCULS DES QUASI-GROUPES DANS GAP VERS LE CODAGE ALGÉBRIQUE ET LA CRYPTOGRAPHIE

Mémoire rédigé et soutenu publiquement en vue de l'obtention du diplôme de MASTER

Par :

NZETCHUEN MANGAPTCHE EUNISSE LOPEZ

Licencée de Mathématiques

Matricule : 17A2481

Sous la direction de :

Dr OGADOA AMASSAYOGA

Chargée de cours

Université de Yaoundé I

Année académique 2022-2023

✠ Dédicaces ✠

Je dédie ce mémoire à M TCHUINTE Clovis.

✠ Remerciements ✠

Ce mémoire est l'aboutissement de plusieurs mois de dure labeur et d'énormes sacrifices. Je profite de cette occasion pour dire un grand merci à toutes ces personnes qui, de près ou de loin, ont participé à cette formation académique, ainsi qu'à la réalisation de ce travail. Je dis merci :

* A **Dieu** pour son amour, sa grâce et sa bonté. Merci Seigneur d'avoir veillé sur moi en tout temps et de m'avoir guidé tout au long de cette formation qui à durée cinq ans.
* A mon encadreur **Dr OGADOA AMASSAYOGA** : je vous remercie de m'avoir donné l'opportunité de travailler avec vous, je vous remercie encore plus pour m'avoir toujours soutenu, conseillé et encouragé, malgré votre emploi de temps très chargé vous avez toujours trouvé un moment à me consacrer. Veuillez recevoir ma profonde gratitude, ainsi que mes sincères remerciements.
* Au **Dr FOBASSO Arnaud** pour sa disponibilité, à m'aider dans la compréhension du domaine de la cryptographie et du codage ; aussi, de m'avoir donné l'opportunité de travailler avec vous, je vous remercie infiniment pour m'avoir toujours soutenu, conseillé et encouragé, malgré vos multiples occupations vous avez toujours trouvé un moment à me consacrer. Veuillez recevoir ma profonde gratitude, ainsi que mes sincères remerciements..
* A tous les membres du jury qui m'ont fait un immense honneur en acceptant d'évaluer nos travaux.
* A tous les enseignants du département de Mathématiques l'université Yaoundé 1 qui

ont activement participé à ma formation depuis septembre 2017.

* A mes parents **M NZETCHUEN philippe et Mme TCHUENKAM sylvie** pour leur soutient affection et prières qui m'ont toujours accompagné dans les moments difficiles.
* A mon oncle **M TCHUINTE clovis** pour ses multiples aides tout au long de ma formation.
* A mes amis **WEYEPE Ulrich, KENGNI Borel, MOTSEBO Arnold, NGASSA Lorna, NZEKA Aimée** pour leur soutient.
* A mes camarades pour leurs soutient lors de la formation.
* A tous les parents, amis et connaissances, que nous n'avons pas mentionnés, mais qui de près ou de loin, ont contribué à notre formation jusqu'ici.

✠ Déclaration sur l' honneur ✠

Le présent travail est une oeuvre originale du candidat et n'a été soumise nulle part ailleurs en partie ou en totalité, pour une autre évaluation académique. Les contributions externes ont été dûment mentionnées et recensées en bibliographie.

Signature du candidat

NZETCHUEN MANGAPTCHE EUNISSE LOPEZ

✠ Résumé ✠

Les quasi-groupes sont des structures algébriques anciennes et largement étudiées. Un quasi-groupe est un magma non vide dont la loi de composition interne satisfait la propriété du carré latin. Introduits en 1935 par Ruth Moufang, les quasi-groupes sont devenus récemment des candidats prometteurs pour des applications en cryptographie et en codage algébrique. Ces structures se distinguent par leur capacité à résoudre des problèmes réputés difficiles et par leur arithmétique efficace. Ce mémoire propose une présentation générale des quasi-groupes : définitions, propriétés essentielles, structures dérivées et exemples. Ensuite, il explore leurs applications en cryptographie, notamment à travers des systèmes cryptographiques basés sur les quasi-groupes, ainsi que dans le domaine du codage algébrique, en se concentrant sur les codes détecteurs d'erreurs construits à partir de ces structures.

Mots-clés : quasi-groupes, carré latin, systèmes cryptographiques, codes détecteurs d'erreurs, GAP.

✠ Abstract ✠

Quasigroups are very old and long-studied algebraic structures. In fact, a quasigroup is a non-empty magma whose internal composition law verifies the Latin square property. Introduced in 1935 by Ruth Moufang. In recent years they have emerged as a potential candidate platform for cryptography and encryption [5]. They have problems problems and offer arithmetic [5]. Formal computation systems such as SAGE and GAP compute quasigroups. The present work is a general presentation of quasigroups : definitions, essential properties, derived structures and examples ; then we will present some applications of quasigroups in cryptography, in particular some cryptosystems based on quasigroups and on algebraic encoding, in particular some error detection codes based on quasigroups.

Keywords : Quasigroups, latin square, cryptosystems, error detecting codes, GAP. .

✠ Table des matières ✠

Chapitre Un

Introduction

Les mathématiciens Galois, Cayley et Dedeking se sont intéressés à la notion de groupes au 19^e siècle [5]. Ils ont découvert que le tableau définissant la loi d'un groupe vérifie un lemme dit de réarrangement : " Chaque élément du groupe apparait une fois et une seule dans chaque ligne et chaque colonne de la table"[5]. Mais ils ont aussi découvert que tout tableau vérifiant cette propriété ne définit pas obligatoirement la loi d'un groupe. La loi obtenue est cependant "quasiment" celle d'un groupe ; d'où le nom de quasi-groupe donné aux structures correspondantes. Les quasi-groupes sont des structures algébriques très anciennes et longtemps étudiées. Introduites par Ruth MOUFANG en 1935 [5], ensuite Bruck a contribué par divers résultats dans la théorie des quasi-groupes en 1944 ; le concept de théorie des nombres non associatifs a été étudié en profondeur par Evans en 1957 [5]. Au cours de ces dernières années, des articles se multiplient en mathématiques appliquées et en ingénierie informatique sur l'importance des quasi-groupes dans la théorie de codage [2] qui est un ensemble de méthodes permettant le transfert d'information de façon efficace et précise ; et de la cryptographie qui est un ensemble des procédés visant à chiffrer des informations pour assurer la confidentialité entre l'émetteur et le destinataire. Le 31 mars 2023, le professeur Raoul M Falcon(Université de Venise) publiait sur les carrés latins et les quasi-groupes d'Hadamard ; Précisément sur l'importance dans la randomisation de séquences et la genèse des fonctions d'hachage pour le chiffrement séquentiel [2]. Ils se sont donc relevés être un candidat potentiel de plateformes pour la cryptographie et le codage algébrique. Ils possèdent des problèmes réputés difficiles et offrent une arithmétiques [2]. Les quasi-groupes sont importants pour la cryptographie pour deux raisons. Tout d'abord, les propriétés de fermeture et d'inversion des quasi-groupes constituent des outils pratiques pour créer des systèmes cryp-

tographiques [5]. D'autre part, l'absence d'associativité permet de créer plus facilement des fonctions à sens unique [2]. Dans notre mémoire les questions suivantes seront évoquées : Qu'est ce qu'un quasi-groupe ? quelles sont les propriétés spécifiques ? quelle est son utilité en cryptographie et codage algébrique ? Notre mémoire s'articule autour de trois chapitres : dans le premier chapitre nous rappelons quelques résultats mathématiques utiles de la théorie des quasi-groupes, de la cryptographie et des codes détecteurs d'erreurs ; ensuite dans le second chapitre, nous présenterons quelques cryptosystèmes basés sur les quasi-groupes et quelques codes détecteurs d'erreurs basés sur les quasi-groupes et dans le troisième chapitre nous présenterons le langage de programmation GAP enfin la conclusion.

Chapitre Deux

Préliminaires

L'objectif principal de ce chapitre est de présenter globalement les quasi-groupes , ainsi que le concept cryptologie nécessaire à la compréhension des cryptosystèmes basés sur ces derniers que nous décrivons dans les chapitres suivants. Les principaux résultats de ce chapitre sont extraits de [3,4,5,8]

2.1 Généralités sur les quasi-groupes

2.1.1. Quelques définitions et notions mathématiques

Définition 2.1.1. *Une relation binaire R sur un ensemble E est une propriété portant sur les couples d'éléments de E.*

Définition 2.1.2. *Une loi de composition interne sur un ensemble E est une application qui à deux éléments d'un ensemble E associe un élément de E.*

Définition 2.1.3. *Un magma ou Groupoïde est un ensemble muni d'une loi de composition interne.*

Définition 2.1.4. *Un Groupe est un ensemble E muni d'une loi de composition interne notée $*$ et définie par :* $\begin{array}{rl} * : E \times E & \to E \\ (x, y) & \mapsto x * y \end{array}$ *telle que*

- La loi $*$ possède un élément neutre $e \in E$ c'est-à-dire pour tout élément $x \in E$, on a $x * e = e * x = x$.

- La loi $*$ soit associative c'est-à-dire

pour tout $x, y, z \in E$ $(x * y) * z = x * (y * z)$.

- Tout élément x de E possède un symétrique (ou inverse) pour $*$ noté x^{-1} satisfaisant :

$x * x^{-1} = x^{-1} * x = e$.

Si de plus, pour tout $x, y \in E$, $x * y = y * x$ alors, G est dit commutatif (ou abélien).

Définition 2.1.5. *Un sémi-groupe est un magma $(E, *)$ dont la loi de composition interne est :*

- Unifère : Elle possède un élément neutre bilatère(à droite et à gauche) ;

- Associative ;

- Régulière : Si $x * y = x * z$ alors $y = z$ (régulière à gauche) et si $y * x = z * x$, alors $y = z$ (régulière à droite)

Définition 2.1.6. *La translation à gauche est définie par l'application ci-dessous :*

$$\begin{array}{rccc} La : & E & \to & E \\ & x & \mapsto & a * x \end{array} \quad \text{avec } a \in (E, *).$$

Définition 2.1.7. *La translation à droite est définie par l'application ci-dessous :*

$$\begin{array}{rccc} Ra : & E & \to & E \\ & x & \mapsto & x * a \end{array} \quad a \in (E, *).$$

Définition 2.1.8. *Une opération d'arité n sur un ensemble E est une application de E^n dans E. Une telle application est dite n-aire.*

Exemple 2.1.1.

1. *$e \mapsto e + 2$ est une opération unaire sur les entiers naturels.*
2. *La multiplication est une opération binaire sur les réels.*
3. *Un groupoïde binaire $(E, *)$ est un ensemble non vide E avec une opération binaire $*$.*
4. *Un groupoïde n-aire $(E, *)$ est un ensemble non vide E accompagné d'une opération n-aire $*$.*

Définition 2.1.9. *La table de Cayley est un tableau à double entrée. Lorsqu'un ensemble fini E est muni d'une loi de composition interne $\bullet$, il est possible de créer un tableau qui présente, pour tous les éléments a et b de E, les résultats obtenus par cette loi $\bullet$ à l'intersection de la ligne représentant a et de la colonne b se trouve $a \bullet b$. Le tableau ainsi constitué est appelé table de Cayley du magma $(E, \bullet)$.*

Définition 2.1.10. *Soit $n \in N$ avec $n \geq 2$. Un carré latin est un tableau à n lignes et n colonnes contenant n éléments distincts et dont chaque ligne et colonne contient un exemplaire de chacun des n éléments.*

Exemple 2.1.2.

1	2	3	4	5
2	3	4	5	1
3	4	5	1	2
4	5	1	2	3
5	1	2	3	4

2.1.2. Quelques définitions de Quasi-groupes

Définition 2.1.11. *Une loi $*$ est dite symogène s'il existe pour chaque couple (a, b) une solution (a, b) unique aux équations $a * x = b \quad et \quad y * a = b$ (Il s'agit de la propriété du carré latin).*

Définition 2.1.12. *Un quasi-groupe est un magma $(E, *)$ non vide et symogène ; c'est à dire un ensemble non vide muni d'une loi de composition interne $*$ symogène.*

Remarque 2.1.1. *Un magma $(E, *)$ est appelé quasi-groupe si les translations à gauches et à droites sont des bijections pour tout $a \in E$*

Définition 2.1.13. *Un groupoïde $(E, *)$ est appelé quasi-groupe si sur l'ensemble E , il existe des opérations "$\backslash$" et "$/$" telles que dans l'algèbre $(E, *, \backslash, /)$ les identités suivantes sont satisfaites : $x * (x\backslash y) = y, (y/x) * x = y, x\backslash(x * y) = y, (y * x)/x = y$*

Définition 2.1.14. *Un ensemble non vide Q muni d'une opération n-aire f définie par : $f : Q^n \mapsto Q$ est appelé quasi-groupe n-aire ou n-quasi-groupe (d'ordre $|Q|$) si dans l'égalité*

$z_0 = f(z_1, ..., z_n)$, *la connaissance de n-éléments de l'ensemble* $\{z_0, z_1, ..., z_n\}$ *définit de manière unique l'élément restant.*

Définition 2.1.15. *Un sous ensemble non vide* H *d'un ensemble* E *est un sous quasi-groupe (sous-boucle, sous-groupe) d'un quasi-groupe* $(E, *)$ *signifie que* $(H, *)$ *est un quasi-groupe(boucle, groupe).*

2.1.3. Exemple de quasi-groupe

1. $(\mathbb{R}, *)$ avec $*$ définie par $x * y = x + \frac{y}{2}$ est un quasi-groupe.
   En effet, Soit $a,\ x,\ y,\ k, m\ \in \mathbb{R}$ montrons que les équations $a * x = k\ et\ y * a = m$ admettent chacune une unique solution dans $\mathbb{R}$
   On a : $a * x = k \Longleftrightarrow a + \frac{y}{2} = k \Longleftrightarrow y = 2(k - a)$;
   $de\ plus, y * a = m \Longleftrightarrow y = \frac{(2m+a)}{2}$;
   ainsi $(\mathbb{R}, *)$ est un quasi-groupe.
   $(R, *)$ n'est pas associatif. En effet, soient $x, y, z \in \mathbb{R}$;
   $x * (y * z) = x + \frac{(y*z)}{2} = x + \frac{y}{2} + \frac{z}{4}$
   tandis que $(x * y) * z = x + \frac{y}{2} + \frac{z}{2}$;
   Il vient donc que pour $x = y = z = 1$; $x * (y * z) = \frac{7}{4}$ et $x * (y * z) = 2 \neq \frac{7}{4}$
   d'où $(R, *)$ n'est pas associatif.

2. Tout groupe est un quasi-groupe. En effet, Soit $(E, +)$ un groupe étant donné que $(-a+c)$ et $(a-b)$ sont des solutions uniques des équations $a+x = c$ et $y+b = a$ respectivement par conséquent, $(E, +)$ est un quasi-groupe ; par conséquent tout groupe est un quasi-groupe.

3. Tout système triple de Steiner est un quasi-groupe. En effet, Un système triple de Steiner en abrégé STS d'ordre n se compose d'un ensemble X de n points et d'une collection B de sous-ensembles de X appelés blocs ou triplets, de sorte que chaque bloc contient exactement 3 points et que deux points se trouvent ensemble dans exactement un bloc. En d'autres termes, c'est une pair (S, T) où S est un ensemble à

n éléments et T est un ensemble constitué des triplets de S (appelé blocs) tels que chaque paire d'éléments de S apparait ensemble dans un unique triplet de T.

Exemples :

$*n = 7$, $S = \{0, 1, 2 \ldots\}$ $T = \{013, 124, 235, 346, 450, 561, 602\}$.

$*n = 3$ (l'exemple trivial) $S = \{0, 1, 2\}$ $T = \{012\}$.

$*n = 1$ (un exemple encore plus trivial) $= \{1\}$ $T = \emptyset$

$*n = 9$ (exemple un peu sérieux) $S = \{1, 2, \ldots 9\}$

$T = \{123, 147, 159, 168, 456, 267, 249, 789, 369, 348, 357\}$.

2.1.4. Différence entre quasi-groupe et groupe

En mathématiques, plus particulièrement en algèbre abstraite, un quasi-groupe est une structure algébrique ressemblant à un groupe dans le sens où la " division" est toujours possible. Les quasi-groupes se distinguent des groupes principalement par le fait qu'ils ne sont pas nécessairement associatifs et qu'ils n'ont pas nécessairement d'élément d'identité.

2.1.5. Différence entre quasi-groupe et semi-groupe

La loi de composition interne dans un quasi-groupe n'est pas nécessairement associative contrairement à celle des semi-groupes. Par contre les quasi-groupes associatifs sont toujours des semi-groupes inverses, mais les semi-groupes inverses ne sont pas nécessairement des quasi-groupes associatifs cependant, les quasi-groupes associatifs sont de manière équivalente, des semi-groupes inverses.

2.1.6. Principales propriétés

1. La loi d'un quasi-groupe est régulière. En effet, si $x * y = x * z$, alors il existe c tel que $c = x * y$ et $c = x * z$.

   Mais d'après le lemme de réarrangement qui stipule que chaque élément de la table apparait une et une seule fois sur chaque ligne et chaque colonne, l'équation $c = x * y$ a une et une seule solution en y. Donc $y = z$, et la loi est régulière à gauche.

On montre de manière analogue que la loi est régulière à droite.

2. La table de multiplication d'un quasi-groupe fini est appelée un carré latin : une matrice $n \times n$ remplie avec n symboles différents d'une façon telle que chaque symbole apparaisse exactement une fois par ligne et une fois par colonne.

3. La "division" est toujours possible dans un quasi-groupe. En effet,

   Soit "." la correspondante de $E \times E$ dans E définie par :

   Pour tout $x, y, z \in E, [(x, y).z] \Leftrightarrow [z * y = x]$

   Cette correspondance est intuitivement "l'opération" $*$, autrement dit une "division" ;

   C'est une application car $*$ est régulière. C'est donc une loi de composition interne donc $(E, .)$ est un magma, et la "division" "." s'applique donc à tous les couples de $E \times E$.

2.1.7. Notions d'isotopisme , isomorphisme et conjugué

2.1.7.1. Notion d'isotopisme

Définition 2.1.16. *Un isotopisme est une application qui permute les lignes, les colonnes et les symboles d'un carré latin. Deux carrés latins sont dits isotopiques si l'un peut être cartographié dans l'autre à l'aide d'un isotopisme.*

Définition 2.1.17. *Soit L un carré latin et soient α , β, λ des permutations des lignes, des colonnes et $N(l'ensemble\ des\ symboles\ du\ carr\ latin)$, respectivement. Alors $\theta = (\alpha, \beta, \lambda)$ est un isotopisme, et L est isotope à $\theta(L)$.*

Exemple 2.1.3. *Ces deux carrés latins sont isotopes puisque θ avec $\alpha, \beta, \lambda = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$ associe le carré (a) au carré (b)*

1	2	3
2	3	1
3	1	2

(a)

3	1	2
1	2	3
2	3	1

(b)

2.1.7.2. Notion d'isomorphisme

Définition 2.1.18. *Un isomorphisme est un cas particulier des isotopismes lorsque* $\alpha = \beta = \lambda$ *. Un isomorphisme est un isotopisme où* $\theta = (\alpha, \alpha, \alpha)$

Exemple 2.1.4. *L'exemple précédent est également celui d'un isomorphisme.*

2.1.7.3. Notion de conjugué ou de paratrophe

Soit π une permutation de l'ensemble $\{r, c, s\}$ où les éléments correspondent respectivement à la ligne, la colonne et le symbole d'un carré latin. Alors π donne lieu à une fonction qui, appliquée à un carré latin , génère un nouveau carré latin. Le nouveau carré latin L est appelé un conjugué ou paratrophe de L.
Puisse que l'ensemble $\{r, c, s\}$ donne lieu à six permutations, il y a six conjugués (dont le carré lui-même) de chaque carré latin.

Définition 2.1.19. *Soit* $(G, \bullet)$ *un quasi-groupe ; considérons sur* G *les lois de composition internes suivantes :*

1. *La loi "* $*$ *" définie comme ci-contre* $\begin{array}{rl} * : G \times G & \to G \\ (a, b) & \mapsto a * b = b \bullet a \end{array}$
2. *La suivante est appelée division à droite par rapport à la loi "* $\bullet$ *" et est notée* $/$ *et est définie par :* $\begin{array}{rl} / : G \times G & \to G \\ (a, b) & \mapsto b/a \end{array}$ *avec* $b/a = c$ *si et seulement si* $b = c \bullet a$.
3. *Cette application notée* $\backslash$ *est appelée division à gauche par rapport à* $\bullet$ *et est définie par :* $\begin{array}{rl} \backslash : G \times G & \to G \\ (a, b) & \mapsto a\backslash b \end{array}$ *avec* $a\backslash b = c$ *si et seulement si* $b = a \bullet c$.
4. *La loi "*$/$*" définie comme ci-dessous :* $\begin{array}{rl} / : G \times G & \to G \\ (a, b) & \mapsto a\backslash b \end{array}$
5. *La loi "*$\backslash\backslash$*" définie par :* $\begin{array}{rl} \backslash\backslash : G \times G & \to G \\ (a, b) & \mapsto a\backslash\backslash b = b/a \end{array}$

G muni de ces 5 lois est un quasi-groupe.

Preuve :

1. $(G;*)$ est un quasi-groupe car la translation à droite suivant $*$ coïncide avec la translation à gauche suivant la loi $\bullet$ et la translation à gauche suivant $*$ coïncide avec la translation à droite suivant $\bullet$ et $(G;\bullet)$ est un quasi-groupe.

2. Montrons que $(G;/)$ est un quasi-groupe

   Soit $x;a \in G$; considérons les translations à gauche et à droite suivant a pour la loi $/$.

   Soit $x \in G$, on a :

   ${}_{/a}T(x) = a/x; {}_{/a}T(x) = k \Longleftrightarrow a/x = k \Longleftrightarrow x = k\backslash a$ donc ${}_{/a}T$ est bijective.

   $/T_a(x) = x/a = T_a^{-1}(x) \forall x \in G$. Donc $/T_a = T_a^{-1}$; de plus T_a^{-1} est bijective d'où $/T_a$ est une bijection. Puisque ${}_{/a}T et /T_a$ sont bijectives, il vient que $(G;/)$ est un quasi-groupe.

3. Montrons que $(G;\backslash)$ est un quasi-groupe.

   Soit $x,k,a \in G$

   $\bullet {}_{\backslash a}T =_a T^{-1}$; puisque ${}_aT^{-1}$ est bijective, alors ${}_{\backslash a}T$ est bijective.

   $\bullet \backslash T_a(x) = k \Longleftrightarrow x\backslash a = k \Longleftrightarrow x = a/k = T_k^{-1}(a)$.

   Puisque T_k^{-1} est bijective, alors l'équation $x\backslash a = k$ admet une solution unique. Il vient donc que $\backslash T_a$ est bijective.

   ${}_{\backslash a}T$ et $\backslash T_a$ étant bijectives, $(G;\backslash)$ est un quasi-groupe.

4. Montrons que $(G;\backslash\backslash)$ est un quasi-groupe.

   Soit $a \in G$: $\bullet \backslash\backslash {}_aT = /T_a$ et d'après la démonstration (3), on a $/T_a$ est une bijection donc $\backslash\backslash {}_aT$ est bijective.

   $\bullet \backslash\backslash T_a = /T_a$ et d'après la preuve $(3) /T_a$ est bijective ; donc $\backslash\backslash T_a$ est bijective.

   $\backslash\backslash T_a$ et $\backslash\backslash {}_aT$ étant bijectives, $(G;\backslash\backslash)$ est un quasi-groupe.

5. Montrons que : $(G;//)$ est un quasi-groupe.

   On a :${}_{//a}T = \backslash T_a$ et $//T_a = {}_{\backslash a}T$ d'après ce qui précède , on a ${}_{//a}T et //T_a$ sont bijectives.

   D'où $(G;//)$ est un quasi-groupe.

   Ainsi G muni de chacune de ces cinq lois est un quasi-groupe ; ce sont les quasi-

groupe conjugués du quasi-groupe $(G; \bullet)$.

2.1.7.4. Exemple d'un quasi-groupe et de ses 5 quasi-groupes conjugués

Considérons le quasi-groupe fini $(Q; \bullet)$ dont le carré latin est donné comme ci-dessous :

$\bullet$	0	1	2	3	4
0	0	3	1	4	2
1	4	1	0	2	3
2	3	4	2	0	1
3	1	2	4	3	0
4	2	0	3	1	4

Ceux de ses cinq quasi-groupes conjugués sont :

1. Pour la loi $*$ on a la carré latin ci-dessous :

$*$	0	1	2	3	4
0	0	4	3	1	2
1	3	1	4	2	0
2	1	0	2	4	3
3	4	2	0	3	1
4	2	3	1	0	4

2. Pour la loi $\backslash$:

$\backslash$	0	1	2	3	4
0	0	2	4	1	3
1	2	1	3	4	0
2	3	4	2	0	1
3	4	0	1	3	2
4	1	3	0	2	4

3. Pour la loi $\backslash\backslash$:

$\backslash\backslash$	0	1	2	3	4
0	0	3	4	2	1
1	4	1	3	0	2
2	1	0	2	4	3
3	2	4	1	3	0
4	3	2	0	1	4

4. Pour la loi $/$:

$/$	0	1	2	3	4
0	0	4	1	2	3
1	3	1	0	4	2
2	4	3	2	1	0
3	2	0	4	3	1
4	1	2	3	0	4

5. Pour la loi $//$

$//$	0	1	2	3	4
0	0	2	3	4	1
1	2	1	4	0	3
2	4	3	2	1	0
3	1	4	0	3	2
4	3	0	1	2	4

Proposition 2.1.1 (3)**.** *Un ensemble formé d'un quasi-groupe et de ses cinq quasi-groupes conjugués ne représente pas nécessairement six quasi-groupes distincts ; il peut se réduire en un, deux, trois ou six quasi-groupes distincts.*

Remarque 2.1.2.

1. *Si l'on prend l'union des six conjugués d'un carré latin et leur classe d'isotopie correspondantes on obtient ce qu'on appelle la classe principale du carré latin.*
2. *Un isotopisme d'un carré latin à lui-même est dit autotopisme*
3. *Un isomorphisme d'un carré latin à lui-même s'appelle automorphisme*

4. *La relation d'être isotopique définie sur l'ensemble de tous les carrés latins est une relation d'équivalence. Les classes d'équivalence ainsi formées sont appelées classes d'isotopie.*

2.2 Structures dérivées

Ici on présente l'ensemble des structures qui dérivent des quasi-groupes

1. Un quasi-groupe avec un élément neutre est appelé boucle.

   Exemple 2.2.1. *Posons* $G = \{1; 2; 3; 4\}$, $(G, \circ)$ *où* $\circ$ *est une loi de composition interne dont la table est donnée ci-dessous :*

$*$	0	1	2	3	4
0	0	1	2	3	4
1	1	0	3	4	2
2	2	4	0	1	3
3	3	2	4	0	1
4	4	3	1	2	0

   Cette table est une boucle avec pour élément neutre 0.

2. Un moufang ou boucle de Moufang est un quasi-groupe neutroactif , c'est à dire un quasi-groupe $(E, *)$ dans lequel, pour tous a, b et c :

$$(a * b) * (c * a) = (a * (b * c)) * a$$

   comme son nom le suggère, une boucle de Moufang est une boucle.
   En effet, pour tout a de E, si e est un élément de E tel que $a*e = a$, alors pour tout x dans E, $(x*a)*x = (x*(a*e))*x = (x*a)*(e*x)$; donc $x = e*x$ et e est un élément neutre à gauche. Pour tout b appartenant à E tel que $b*e = e$, $y*b = e*(y*b)$, car e est neutre à gauche ; ainsi $(y*b)*e = (e*(y*b))*e = (e*y)*(b*e) = (e*y)*e = y*e$, d'où $y * b = y$, et b est donc neutre à droite. Enfin, $e = e * b = b$, et b est donc un élément neutre.

3. Tout groupe est un quasigroupe : en effet, la loi d'un groupe est symogène : $a * x = b$ si et seulement si $x = a^{-1} * b$, et $y * a = b$ si et seulement si $y = b * a^{-1}$.

   Inversement, tout quasi-groupe associatif est un moufang, donc une boucle, et toute boucle associative est un groupe.

   Ceci montre que l'ensemble des groupes est exactement l'ensemble des quasi-groupes associatifs.

2.3 Généralités sur la cryptographie

Définition 2.3.1. *La cryptographie est l'ensemble des méthodes qui permet à deux parties souvent appelées Alice et Bob de communiquer sur un canal non sécurisé sans permettre à un tiers, appelé adversaire , de savoir ce qui se dit.*

Définition 2.3.2. *Une clé est une partie cachée (généralement petite) ou un paramètre de chiffrement.*
Il existe deux types de cryptographie : la cryptographie symétrique (à clé sécrète) ici les algorithmes de chiffrement symétrique se fondent sur une même clé pour chiffrer et déchiffrer un message et la cryptographie asymétrique qui est un procédé qui intègre deux clés de chiffrement, une clé publique et une clé privée.

Définition 2.3.3. *Un chiffrement est un moyen (une méthode, un algorithme) de transformation d'une information en vue de sa défense.*

Définition 2.3.4. *On entendra par théorie du codage (théorie du code) une science de la défense des informations contre les erreurs accidentelles causées par la transformation et l'envoi (transmission) de ces informations.*

Presque toutes les constructions connues de codes de détection et de corrections d'erreurs d'algorithmes cryptographiques et de systèmes de chiffrement ont utilisé des structures algébriques associatives telles que des groupes et des champs, Il existe une possibilité d'utiliser des structures non associatives telles que les quasi-groupes et les neo-champs(def) dans presque toutes les branches de la théorie du codage, et en particulier en cryptologie. Souvent

, les codes et chiffrements basés sur les systèmes non associatifs présentent de meilleures possibilités que les codes et chiffrements connus basés sur les systèmes associatifs[28,78]. L'efficacité des applications des quasi-groupes en cryptologie est basée sur le fait que les quasi-groupes sont des "permutations généralisées" d'un certain type et que le nombre de quasi-groupe d'ordre n est supérieur à $n!(n-1)....2!1!$ [25].

Dans le cas le plus général, où Alice a l'intention d'envoyer un message à Bob , elle choisit un message appelé texte clair, et le chiffre en utilisant la méthode cryptographique de son choix. Le résultat s'appelle le texte chiffré et est envoyé à Bob sur un canal approprié.

Lorsque Bob reçoit le message, il le déchiffre à l'aide de la méthode cryptographique spécifiée et acquiert le texte clair d'origine. Une paire d'algorithmes utilisée pour le chiffrement et le déchiffrement est communément appelée chiffrement.

Pour qu'une méthode cryptographique soit efficace, il doit être facile de chiffrer le message, mais très difficile ou impossible de le déchiffrer sans certaines informations, qui ne devraient, de préférence être connues que de Bob et peut être Alice. Ces informations cachées nécessaires au décryptage sont communément appelées une clé. Ainsi, si l'adversaire intercepte le texte chiffré lorsqu'il est envoyé à Bob, il ne devrait pas pouvoir acquérir le texte en clair original sans avoir à la clé.

2.3.1. Fonction de hachage

2.3.1.1. Sur la fonction unidirectionnelle

Une fonction $F : X \mapsto Y$ est dite fonction à sens uique, si les conditions suivantes sont remplies :

• il existe un algorithme polynomial de calcul de $F(x)$ pour tout $x \in X$;

• il n'existe pas d'algorithme polynomial d'inversion de la fonction F , c'est à dire qu'il n'existe pas d'algorithme polynomial en temps pour résoudre l'équation $F(x) = y$ relativement en la variable x.

Définition 2.3.5. *Une fonction de hachage $H()$ qui associe un message de longueur arbitraire M à une valeur de hachage de longueur fixe $H(M)$ est une fonction de hachage*

unidirectionnelle, si elle satisfait les propriétés suivantes :

1. *La description de $H()$ est publiquement connue et ne devrait nécessiter aucune informations secrètes pour son fonctionnement.*

2. *Étant donné M, il est facile de calculer $H(M)$.*

3. *Étant donné $H(M)$ dans la plage de $H()$, il est difficile de trouver un message M pour $H(M)$ donné, et étant donné M et $H(M)$, il est difficile de trouver un message $M_0 \neq M$ qui hachent le même résultat $(H(M) = H(M_0))$*

Définition 2.3.6. *Une fonction de hachage à sens unique $H()$ est appelée fonction de hachage sans collision(CFHF), s'il est difficile de trouver deux messages distincts M et M_0 qui hachent le même résultat $(H(M) = H(M_0))$.*

2.4 Généralités sur les codes détecteurs d'erreurs

2.4.1. Quelques définitions

Définition 2.4.1. *L'alphabet est un ensemble fini de symboles avec lequel sont écrits les mots du code. On prendra pour alphabet $\mathbb{F}_q$, sur un corps fini à q éléments muni des opérations somme et produits.*

Définition 2.4.2. *Le message A est une liste de k-symboles à transmettre , on l'assimile à un élément de $\mathbb{F}_q^k$*

Définition 2.4.3. *L'encodeur est un algorithme qui transmette le message A en un mot du code C de longueur n comportant des redondances on l'assimile à une fonction injective : $enc : \mathbb{F}_q^k \mapsto \mathbb{F}_q^n$*

Définition 2.4.4. *Le code est l'ensemble des mots possibles : $C = enc(B), B \in \mathbb{F}_q^k$*

Définition 2.4.5. *Le mot reçu $D \in \mathbb{F}_q^n$ est une version potentiellement corrompue du message émis $D = C + E$ avec E les erreurs introduites.*

Lorsque deux personnes communiquent entre elles, il arrive souvent que le message reçu diffère de celui émis ; lorsque les données sont transmises ou stockées, elles peuvent être sujettes à des perturbations telles que des bruits, des interférences électromagnétiques ou des erreurs de lecture/ écriture. Les codes détecteurs d'erreurs sont conçus pour détecter ces erreurs et fournir un mécanisme pour les corriger si possible.

Les codes détecteurs d'erreur sont des méthodes utilisées pour détecter les erreurs qui se produisent lors de la transmission des données. Ils sont largement utilisés dans les systèmes de communication, les réseaux informatiques et le stockage de données.

error type		relative frequency in %	
		Verhoeff	Beckley
single error	$a \to b$	79.0 (60-95)	86
adjacent transposition	$ab \to ba$	10.2	8
jump transposition	$abc \to cba$	0.8	
twin error	$aa \to bb$	0.6	6
phonetic error ($a \geqslant 2$)	$a0 \to 1a$	0.5	
jump twin error	$aca \to bcb$	0.3	
other error		8.6	

2.4.2. Quelques caractéristiques

Nous allons par la suite présenter quelques caractéristiques des codes détecteurs d'erreurs. Les notions de cette sous section ont principalement été extraites de [4,5]

1. **Redondance** : Les codes correcteurs d'erreurs ajoutent une certaine redondance aux données transmises. Cette redondance est utilisée pour vérifier des données lorsqu'elles sont reçues.
2. **Bits de contrôle** : Les codes détecteurs d'erreur ajoutent des bits supplémentaires, appelés bits de contrôle, aux données transmises. Ces bits sont calculés à partir des bits de données et permettent de vérifier si des erreurs se sont produites pendant la transmission.
3. **Capacité de détection** : La capacité d'un code détecteur d'erreur à identifier une erreur dépend du nombre de bits supplémentaires ajoutés et du type d'algorithme utilisé. Certains codes peuvent détecter toutes les erreurs simples, tandis que d'autres peuvent également détecter certaines combinaisons d'erreurs multiples.

4. **Utilisation** : Les codes détecteurs d'erreur sont utilisés dans de nombreux domaines, tels que les communications sans fil, les systèmes de stockage de données, les réseaux informatiques les systèmes embarqués, etc. Ils sont essentiels pour garantir l'intégrité des données lors de leur transmission.

2.4.3. Méthodes de détection

Il existe différentes méthodes pour détecter les erreurs à l'aide des codes détecteurs d'erreur, telles que :

1. **Code de parité** : Ce code ajoute un bit supplémentaire à chaque groupe de bits pour s'assurer que le nombre total de bits à 1 est pair(parité paire) ou impaire (parité impaire). Si le nombre de bits à 1 diffère du type de parité spécifié, une erreur est détectée.
2. **Code CRC(Cyclic Redundancy Check)** : Ce code utilise un polynôme générateur pour calculer un reste qui est ajouté aux données. Le récepteur effectue également le calcul et compare le reste reçu avec celui calculé localement. Si les restes diffèrent, une erreur est détectée.
3. **Code Hamming** : Ce code ajoute des bits de contrôle supplémentaires aux données pour former des mots codés qui on,t une distance minimale garantie entre eux. Cela permet la détection et la correction d'erreurs simples.
4. **Code BCH (Bose-Chaudhuri-Hocquenghem)** : Ce code utilise des polynômes cycliques pour encoder les données avec des symboles supplémentaires, offrant une détection et une correction d'erreurs efficaces.

Il était question pour nous de présenter de façon générale la théorie des quasi-groupes, de la cryptographie et des codes détecteurs d'erreurs.

Chapitre Trois

Calculs des quasi-groupes pour la cryptographie et le codage algébrique

Dans ce chapitre, nous présenterons quelques systèmes cryptographiques et les codes correcteurs d'erreurs basés sur les quasi-groupes. Les principaux résultats de ce chapitre ont été extraits de [4,6,8]

3.1 Calculs des quasi-groupes pour la cryptographie

Nous présenterons dans cette partie les cryptosystèmes basés sur les quasi-groupes.

3.1.1. Chiffrement avec un quasi-groupe comme clé

Eliska Ochodkova et vaclav Snavel ont proposé d'utiliser des quasi-groupes pour l'encodage sécurisé du système de fichiers.
Un quasi-groupe $(Q, \bullet)$ et son (23)-parastrophe $(Q, \backslash)$ vérifient les identités suivantes :
$x \backslash (x \bullet y) = y$, $x \bullet (x \backslash y) = y$. Les auteurs proposent d'utiliser cette propriété des quasi-groupes pour construire un chiffrement de flux.

Définition 3.1.1. *Soit A un alphabet non vide, k un entier naturel, $u_i,\ v_i \in A, i \in \{1, ...k\}$. Un élément fixe l $(l \in A)$ est appelé leader.*

Dans la suite, nous allons présenter un algorithme de chiffrement et un algorithme de déchiffrement.

3.1.1.1.]

Algorithme de chiffrement [5] **Entrée :** $(u_1, ...u_k), l$;

Sortie : $(v_1....v_k)$;

1- $v_1 = l \bullet u_1$;

2- **pour** i **allant de** 1 **à** $k-1$ **faire** :

3- $v_{i+1} = v_i \bullet u_{i+1}$;

4- **fin pour**

5- **Retourner** $(v_1....v_k)$;

3.1.1.2.]

Algorithme de déchiffrement [5] **Entrée :** $(v_1, ...v_k), l$;

Sortie : $(u_1....u_k)$;

1- $u_1 = l \setminus v_1$;

2- **pour** i **allant de 1 à** $k-1$ **faire**

3- $u_{i+1} = v_i \setminus v_{i+1}$;

4- **fin pour**

5-**Retourner** $(u_1....u_k)$;

Exemple 3.1.1. *Soit l'alphabet A composé des lettres $a,\ b,\ c$. Soient les quasi-groupes $(A, \bullet)$ et $(A, \setminus)$ définie par les tables de Cayley suivantes :*

$\bullet$	a	b	c
a	b	c	a
b	c	a	b
c	a	b	c

$\setminus$	a	b	c
a	c	a	b
b	b	c	a
c	a	b	c

Soit $l = a$ et $u = bbcaacba$; alors le texte chiffré est $v = cbbcaaca$

De même en appliquant la fonction de déchiffrement à v on a : $u = bbcaacba$

3.1.2. Cryptosystème basé sur les quasi-groupes n-aire

Soit Q un alphabet fini non vide k un entier naturel, $u_i\ v_i \in Q, i \in 1, ...k$.

Définissons un quasi-groupe n-aire (Q, f). Il est clair que tout quasi-groupe $(Q,^{(i,n+1)} f)$,

tout élément fixe i est défini de façon unique. Ci dessous, pour plus de simplicité, nous prenons $i = n$. Prenons les éléments fixes $l_a^{(n-1)(n-1)}, (l_i \in Q)$ que nous appelerons leaders. Soit $(U_1, U_2, ..., U_k)$ un K-uplet de Q, nous présenterons la procédure de chiffrement et de déchiffrement :

3.1.2.1.]

Procédure de chiffrement [6] 1- $v_1 = f(l_1^{n-1}, u_1)$;

2- $v_2 = f(l_n^{2n-2}, u_2)$;

.....,

3- $v_{n-1} = f(l_{n^2-3n+3}^{(n-1)(n-1)}, u_{n-1})$;

4- $v_n = f(v_1^{n-1}, u_n)$;

5- $v_{n+1} = f(v_2^{n-1}, u_{n+1})$

; 6- $v_{n+2} = f(v_3^{n+1}, u_{n+2)}$

; ....

On obtient ainsi le texte chiffré $v_1v_2.....v_{n-1}v_nv_{n+1}$.

3.1.2.2.]

Procédure de chiffrement [6] Cet algorithme est construite de façon similaire comme dans le cas binaire

1- $u_1 =^{(n,n+1)} f(l_1^{n-1}, v_1)$;

2- $u_2 =^{(n,n+1)} f(l_n^{2n-2}, v_2)$;

.....,

3- $u_{n-1} =^{(n,n+1)} f(l_{n^2-3n+3}^{(n-1)(n-1)}, v_{n-1})$;

4- $u_n =^{(n,n+1)} f(v_1^{n-1}, v_n)$;

5- $u_{n+1} =^{(n,n+1)} f(v_2^{n-1}, v_{n+1})$;

6- $u_{n+2} =^{(n,n+1)} f(v_3^{n+1}, v_{n+2})$;

....

3.1.3. Deux nouveaux chiffrements basés sur des carrés latins isotopiques

La notion d'isotopismes donne lieu à deux chiffrements simples et étroitement liés de la manière suivante :

3.1.3.1.]

Chiffrement avec un isotopisme comme clé [5,6] Soit le message qui consiste en un carré latin P d'ordre n avec des symboles pris comme les éléments de l'ensemble $\mathbb{N}$. Soit la clé sécrète un isotopisme θ . On défini la procédure de chiffrement à $\theta(P) = C$. Où C est le texte chiffré qui doit être envoyé sur un canal éventuellement non sécurisé. Puisque θ se compose de trois permutations , il est inversible, et donc la procédure de déchiffrement devient $\theta^{-1}(C) = P$.

Le langage de programmation des groupes GAP que nous verrons au chapitre 3 nous montrerons comment quitter d'un message clair pour un carré latin correspondant.

Exemple 3.1.2. *Supposons qu'Alice veuille envoyer un message à Bob. Alice et Bob ont une clé secrète partagée sous la forme* $\theta = (\alpha, \beta, \lambda)$ *avec* $\alpha = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \end{pmatrix}$ β $\lambda = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix}$

Soit M *le carré latin défini comme suit*

3	4	1	2
2	3	4	1
1	2	3	4
4	1	2	3

Alice chiffre M *en prenant* $C = \theta(M)$ *, le carré latin résultant est :*

1	2	3	4
3	4	1	2
2	3	4	1
4	1	2	3

Elle l'envoie à Bob. Quand Bob veut déchiffrer C il effectue $M = \theta^{-1}(C)$

3.1.3.2.]

Chiffrement avec un carré latin comme clé [6,7] Une autre approche consiste à prendre un carré latin comme clé sécrète et à envoyer un isotopisme à appliquer au carré comme texte chiffré. Dans ce cas, il serait judicieux de numéroter arbitrairement les lignes et les colonnes (voir figure) car les permutations correspondantes aux lignes et aux colonnes n'offrent aucune sécurité supplémentaire si un adversaire connait leur ordre d'origine.

Trouver un isotopisme tel que $L' = \theta(L)$ pour L et L' donnés peut être difficile [7] donne un algorithme avec la complexité du pire cas $o(nlog(2n))$ pour le problème étroitement lié de déterminer si deux carrés latins sont isotopiques. Par conséquent, cette méthode fonctionne mieux lorsqu'une liste ordonnée de positions est attachée au texte chiffré. Cela permet de prendre un isotopisme aléatoire θ comme texte chiffré, puis d'attacher une liste de positions dans $\theta(L)$ où L est la clé secrète. Ces positions peuvent soit faire référence à l'ordre naturel des lignes et des colonnes, soit à l'ordre arbitraire qui leur est appliqué. Bien sur, les parties communicantes doivent décider à l'avance de ce à quoi les positions se réfèrent.

Dans ce chiffrement, le texte chiffré θ et la liste des positions seraient de notoriété publique. Le carré latin et l'ordre arbitraire des lignes et des colonnes doivent être tenues secrets.

	2	1	3
2	b	a	c
3	c	b	a
1	a	c	b

Figure1 : Un carré latin avec un ordre arbitraire de lignes et de colonnes.

Les nombres de la rangée supérieure correspondent à l'ordre des colonnes et les nombres de la colonne la plus à gauche correspondent à l'ordre des lignes.

Exemple 3.1.3. *Disons qu'Alice et Bob ont comme clé partagée le carré latin L, dans la Figure 2 avec l'ordre des lignes et des colonnes comme spécifié dans la figure. Si Bob veut envoyer le message $(1, 3, 4, 3, 2)$ à Alice, il doit d'abord décider d'un isotopisme. Supposons que Bob génère aléatoirement θ à partir de l'Exemple 2.1.2. Il applique θ à L et obtient le carré latin L' qui peut être vu sur la Figure 3. il envoie θ avec une liste ordonnée de positions dans L comme l'indique les tableaux ci-dessous :*

	4	1	2	3
3	1	2	3	4
1	2	3	4	1
4	4	1	2	3
2	3	4	1	2

;

2	3	4	1
1	2	3	4
3	4	1	2
4	1	2	3

Figure 2 : *Un carré latin d'ordre 4, pris comme clé dans cet exemple.*

Les nombres de la rangée supérieure correspondent à l'ordre des colonnes et les nombres de la colonne la plus à gauche correspondent à l'ordre des lignes.

Figure 3 : Un carré latin d'ordre 4, $\theta(L)$ où L est le carré de la Figure 2.
$((1, 2)(1, 4)(4, 3)(3, 3)(2, 4))$ *(ces positions font référence à l'ordre naturel des lignes et des colonnes) à Alice.*
Alice peut facilement récupérer $\theta(L) = L'$ et trouver les positions données dans ce carré qui lui donne le message déchiffré.

3.2 Calculs des quasi-groupes pour le codage algébrique [6]

Les investigateurs statistiques de J.Verhoff et D.F Beckley ont montré que les erreurs les plus fréquentes commises par des opérateurs humains lors de la transmission de données sont des erreurs uniques(c'est à dire les erreurs dans un composant exactement), des transpositions adjacentes(c'est à dire des erreurs commises en inter changeant des chiffres adjacents

, c'est à dire des erreurs de la forme $ab \mapsto ba$) et insertion ou erreurs de suppression. On note que, si tous les mots de code sont de longueur égale, des erreurs d'insertion et de suppression peut être détecté facilement.

Il est bien connue qu'il existe une possibilité de détecter certains types d'erreurs commises par des opérateurs humains lors de la transmission de données à l'aide d'un code vérification de symbole. Dans cette section, nous construisons des codes de détection d'erreurs systématiques qui nous permettent de détecter pratiquement toutes les erreurs du tableau1 dont l'apparition se produit avec une certaine régularité : à savoir, les erreurs simples, les erreurs de transposition adjacentes, les erreurs de transposition par saut, les erreurs jumelles , les erreurs de jumeau de saut et les erreurs phonétiques, c'est à dire toutes les erreurs du tableau1 à l'exception de " autres erreurs".
pour détecter les erreurs simples et les transpositions adjacentes, on utilise souvent des systèmes de chiffres de contrôle ; ceux ci consistent généralement en mots de code $a_1....a_{n+1}$ contenant, outre les chiffres d'informations $a_1....a_n$, un caractère de contrôle(vérification a_{n+1}).

Définition 3.2.1. *Un système de chiffre de contrôle avec un caractère de contrôle est un système systématique de code de détection d'erreur sur un alphabet Q qui survient en ajoutant un chiffre de contrôle a_{n+1} à chaque mot $a_1a_2.....a_n \in Q^n$:d'une loi de composition interne notée $*$:*

$$\mathcal{C} : Q^n \to Q^{n+1}$$
$$a_1a_2...a_n \mapsto a_1a_2...a_na_{n+1}$$

Ici le mot "systématique" signifie que le caractère de contrôle est le dernier symbole de tout mot de code du code C.

Nous pouvons voir le code C comme une application sur un alphabet Q telle que le symbole de contrôle a_{n+1} est obtenue à partir des symboles d'information $a_1, a_2, ...a_n$ de la manière suivante : $g(a_1, a_2, ...a_n) = a_{n+1}$ où g est une opération n-aire sur l'ensemble Q.

Définition 3.2.2. *Nous appellerons le code C avec un caractère de contrôle a_{n+1} sur un alphabet Q un code n-aire (Q, g). Si dans un code n-aire (Q, g) l'opération g est l'opération d'un quasi-groupe n-aire alors ce code sera appelé un code n-quasi-groupes (Q, g).*

Nous dirons que les mots de code n $a_1a_2...a_na_{n+1}$ et $b_1b_2...b_nb_{n+1}$ sont égaux si et seulement si $a_i = b_i pour tout i \in 1, ...n+1$. Parfois un mot de code $a_1a_2...a_n$ sera noté a_1^{n+1}. Par une erreur dans un mot de code a_1^{n+1} d'un code C sur un alphabet Q, on entend tout mot $b_1^{n+1} \in Q^{n+1}$ tel qu'il existe au moins un indice $j \in [|1, n+1|]$ tel que $a_j \neq b_j$.
Un code n-aire (Q, g) détecte une erreur dans un mot de transmission reçu $a_1a_2...a_{n+1}$ si et seulement si $g(a_1^n) \neq a_{n+1}$.

Theorème 3.2.1 (6)**.** *Tout code n-aire (Q, g) détecte toutes les erreurs simples si et seulement s'il s'agit d'un code n-quasi-groupe, c'est à dire que l'opération n-aire g est une opération n-aire de quasi-groupe.*

3.2.1. Sur les possibilités des codes de quasi-groupes

Maintenant, nous voudrions montrer que tous les codes de quasi-groupes n-aires (Q, g) sur le même alphabet Q et avec différentes opérations de quasi-groupes g (d'arité n fixé) ont des possibilités égales de détecter les erreurs.
Comme d'habitude, $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ désigne un coefficient binomial. Nous appellerons une erreur sur k place $(k > 1)$ dans un mot de code une $k - erreur$

Theorème 3.2.2 (6)**.** *Tout code de n-quasi-groupe (Q, g) sur un alphabet fixe Q, $(|Q| = q)$, avec un nombre fini fixe n de symboles, d'informations et avec un chiffre de contrôle peut détecter :*

1. *Tous $q^{n+1}q^n$ éventuels mots erronés qui pourraient être reçu par erreur pour un mot de code particulier(a_1^{n+1}), où l'erreur (ou les erreurs) se produisent soit parmi les n informations chiffrés/dans le chiffre de contrôle unique ;*
2. *Tous $q^n - q^{n-1}$ éventuels mots erronés qui pourraient être reçu par erreur pour un mot de code particulier (a_1^{n+1}) où l'erreur (ou les erreurs) ne se produit que parmi les n informations chiffres.*
3. *Tous les $\binom{n+1}{k}(q-1)^{k-1}(q-2)$ mots parmi les mots erronés qui diffèrent du mot de code de quasi-groupe transmis donné à exactement k emplacement lorsque*

l'erreur (ou les erreurs) se produits uniquement parmi les chiffres d'informations

Preuve :

1. si on fixe un mot de code (a_1^{n+1}) d'un code de quasi-groupe n-aire (Q, g) (c'est à dire pour le mot pour l'égalité $a_{n+1} = g(a_1, ...a_n)$) puis tous les autres mots possibles (x_1^{n+1}) où $x_1, ...x_{n+1} \in Q$ seront erronées. Cependant, $q^n - 1$ d'entre eux sont des mots de code car ils satisfont l'équation de vérification $x_{n+1} = g(x_1, ..., x_n)$ et ainsi, ces mots ne seront pas reconnus comme erronés. Par conséquent, un code quasi-groupe n-aire détecte $(q^{n+1} - 1) - (q^n - 1) = q^{n=1} - q^n$ sue les $q^{n+1} - 1$ mots erronés qui peuvent être reçu.

2. Dans ce cas, nous supposons qu'un symbole de contrôle a_{n+1} a été transmis sans erreur. Le cas (b) se prouve par analogie avec le cas (a). Il existe $q^n - 1$ erreurs possibles et là il existe $q^{n-1} - 1$ mots de code quasi-groupe pour lesquels l'équation de contrôle $a_{n+1} = g(x_1^n)$ ne détecte pas toute l'erreur.
   Par conséquent , un code de quasi-groupe n-aire (Q, g) détecte $q^n - q^{n-1}$ erreurs dans les n premiers symboles d'information de tout mot de code de quasi-groupe (a_1^{n+1})

3. Il y'a $(q-1)^k$ mots qui diffèrent du mot de code transmis donné par (a_1^{n+1}) dans k places fixes et il y'a tous les $\binom{n+1}{k}$ choix pour les k places fixes, donc il y'a $\binom{n+1}{k}(q-1)^k$ mots qui diffèrent du mot de code transmis donné par (a_1^{n+1}) à exactement k endroits. Cependant, parmi les $(q-1)^k$ mots qui diffèrent du mot de code transmis donné par (a_1^{n+1}) avec k fixe, il y'a $(q-1)^{k-1}$ qui satisfont l'équation de contrôle $x_{n+1} = d(x_1, ..., x_n)$ et ainsi, ces mots ne seront pas reconnus comme erronés. (Pour voir cela, observez qu'il y'a $(q-1)^{k-1}$ manières erronés de remplir $k-1$ des k places fixes et que, pour chacune d'elles, il y'a exactement une façon de remplir la place fixe restante de telle sorte que le mot entier satisfasse l'équation de vérification.) Il s'ensuit que, $\binom{n+1}{k}(q-1)^k$- $\binom{n+1}{k}(q-1)^{k-1}$= $\binom{n+1}{k}(q-1)^{k-1}(q-2)$. des éventuels mots erronés seront détectes.

4. Dans ce cas, nous supposons que le symbole de contrôle de a_{n+1} a été transmis sans erreur. Le cas (d) se démontre par analogie avec le cas (c).

Définition 3.2.3. *On dit que les codes de quasi-groupes n-aires (Q, g) et (Q, f) sont isotopes si leurs opérations de quasi-groupe n-aire (Q, g) et (Q, f) sont isotopes.*

Remarque 3.2.1. *Du théorème 2.2.2, il s'ensuit que les codes de quasi-groupes isotopiques n-aires détectent un nombre égal de k-erreurs pour tout k approprié.*

Si nous supposons que la probabilité de recevoir un mot avec plusieurs erreurs à la place d'un mot de code transmis par a_1^{n+1} est la même que celle de recevoir un mot erroné qui ne contient qu'une seule erreur, alors on obtient :

Corollaire 3.2.1 (6)**.**

1. *La fréquence relative des erreurs détectés par un code de quasi-groupe n-aire (Q, g) en n chiffre d'information et en un symbole de contrôle de tout mot de code de quasi-groupe fixe a_1^{n+1} est égale à : $\frac{q^n}{q^n+q^{n-1}+\ldots+1} > \frac{q-1}{q}$.*
2. *La fréquence relative des erreurs détectées par le code de quasi-groupe n-aire (Q, g) de tous les types dans les n premiers symboles d'informations de tout mot de code de quasi-groupe fixe par $\frac{q^{n-1}}{q^{n-1}+q^{n-2}+\ldots+1} > \frac{q-1}{q}$.*
3. *La fréquence relative des k-erreurs détectées par le code de quasi-groupe n-aire (Q, g) dans n chiffres d'informations et dans un symbole de contrôle de tout mot de code de quasi-groupe fixe a_1^{n+1} est égale à $\frac{q-2}{q-1}$.*

Preuve :

1. Par le théorème 2.2.2 on a $\frac{q^{n+1}-q^n}{q^{n+1}-1} = \frac{q^n(q-1)}{(q-1)(q^n+q^{n-1}+\ldots+1)} = \frac{q^n}{(q^n+q^{n-1}+\ldots+1)}$. Plus loin, $\frac{q^{n+1}-q^n}{q^{n+1}-1} \geq \frac{q^{n+1}-q^n}{q^{n+1}} = \frac{q-1}{q}$.
2. Ce cas ce prouve par analogy au cas (a).

3.

$$\frac{\binom{n+1}{k}(q-1)^{k-1}(q-2)}{\binom{n+1}{k}(q-1)^{k}} = \frac{q-2}{q-1}$$

4. ce cas se prouve par analogy au cas (c).

3.2.2. Codes TAC-quasi-groupes et n-quasi-groupe

Définition 3.2.4. *Un quasi-groupe binaire $(Q, \bullet)$ est dit anticommutatif (parfois un tel quasi-groupe est appelé quasi-groupe antisymétrique) si et seulement si l'implication suivante est vraie :*

$x \bullet y = y \bullet x \Longrightarrow x = y$ *pour tout* $x, y \in Q$

Définition 3.2.5. *Un quasi-groupe binaire anti-commutatif $(Q, \bullet)$ est dit totalement anti-commutatif (parfois TAC-quasi-groupe en abrégé) si l'implication suivante est vraie :*

$x \bullet x = y \bullet y \Longrightarrow x = y$ *pour tout* $x, y \in Q$

Définition 3.2.6. *Un retrait de la forme $f(a_1^{i-1}, x_i, a_{i+1}^{i+k+1}, .., x_{i+k}, a_{i+k+1}^{n})$ d'un quasi-groupe n-aire (Q, f) où $(a_1^{i-1}, a_{i+1}^{i+k+1}, a_{i+k+1}^{n})$ sont des éléments fixes de l'ensemble $Q, i \in [|1, n-k|], k \in [|1, n-k|]$ est appelé $(i, i+k) - retrait\ binaire\ d'un\ quasi - groupe(Q, f)$*

Theorème 3.2.3 (6)**.** *Un code de quasi-groupe n-aire (Q, g) détecte toute erreur de transposition et de jumeau sur les places de la forme $(i, i+k)(i \in [|1, n-k|], k \in [|1, n-1|], i+k \leq n)$ si et seulement si tous les $(i, i+k)-$retrait-quasi-groupe binaire (Q, f) sont des quasi-groupes totalement anti-commutatifs.*

Preuve :

Supposons que tous les retraits binaires $(i, i+k)$ d'un quasi-groupe n-aire (Q, d) sont des quasi-groupes totalement anti-commutatifs, de la définition de quasi-groupe binaire totalement anti-commutatifs il s'ensuit que le code (Q, d) détecte toute erreur de transposition et jumeau à la place $(i, i+k)$.

Inversement, si l'on suppose qu'il existe un lieu $(i, i+k)$ et qu'il existe des éléments $a_1^{i-1}, b, a_{i+1}^{i+k-1}, c, a_{i+k+1}^n$ $(b \neq c)$ tel que $d(a_1^{i-1}, b, a_{i+1}^{i+k-1}, c, a_{i+k+1}^n) = d(a_1^{i-1}, c, a_{i+1}^{i+k-1}, b, a_{i+k+1}^n)$, ainsi le retrait binaire $d(a_1^{i-1}, x, a_{i+1}^{i+k-1}, y, a_{i+k+1}^n)$ est un quasi-groupe non commutatif et nous avons une contradiction.

Si nous supposons qu'il existe un lieu $(i, i+k)$ et qu'il existe des éléments $a_1^{i-1}, b, a_{i+1}^{i+k-1}, c, a_{i+k+1}^n$ $(b \neq c)$ tel que $d(a_1^{i-1}, b, a_{i+1}^{i+k-1}, b, a_{i+k+1}^n) = d(a_1^{i-1}, c, a_{i+1}^{i+k-1}, c, a_{i+k+1}^n)$ alors le retrait binaire $d(a_1^{i-1}, x, a_{i+1}^{i+k-1}, y, a_{i+k+1}^n)$ est un quasi-groupe non commutatif et nous avons encore une contradiction.

3.2.3. Les erreurs phonétiques

Dans cette partie, nous supposons que tous les quasi-groupes sont définis sur un ensemble Q tel que $Q = \{0, 1, ..., m\}$. On peut voir une erreur phonétique $a0 \longrightarrow 1a, a \neq 0, a \neq 1$, comme un type particulier de double erreur dans un mot de code à la place de la forme $(i, i+1)$

Theorème 3.2.4 (6). *Un T-quasi-groupe binaire $(Q, \bullet)$ de la forme $x \bullet y = \alpha x + \beta y + b$ détecte toutes les erreurs phonétiques si et seulement si, $(\alpha - \beta)a \neq \alpha 1$ pour tout $a \in Q$ tel que $a \neq 0, a \neq 1$.*

Preuve :

On trouve les conditions quand $a.0 = a.1$. Nous avons $a.0 = \alpha a = b, 1.a = \alpha 1 + \beta a + b$. Alors, le quasi-groupe $(Q, \bullet)$ ne peut pas détecter les erreurs phonétiques si et seulement si $\alpha a + b = \alpha 1 + \beta A + b$, c'est à dire si et seulement si $(\alpha - \beta)a = \alpha 1$. Ainsi, le T-quasi-groupe $(Q, \bullet)$ détecte tous les erreurs phonétiques si et seulement si $(\alpha - \beta)a \neq \alpha 1$ pour tout $a \in Q$ tel que $a \neq 0, a \neq 1$.

Theorème 3.2.5 (6). *Il n'existe pas de code $(n-1) - T - quasi - groupe(Q, g)$ avec pour équation de vérification $d(x_1^n) = \alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_n = 0$ qui détecte simultanément toutes les erreurs de transpositions et toutes les erreurs phonétiques en tous lieux de la forme $(i, i+1)$.*

Preuve :

Par le théorème 2.2.2, il résulte que pour détecter toutes les erreurs de transposition sur un

endroit de la forme $(i, i+1)$ la condition suivante doit être remplie : l'application $\alpha_i - \alpha i + 1$ est une permutation sur l'ensemble Q.

Du théorème précédent, il résulte que le code (Q, g) détecte toutes les erreurs phonétiques sur un endroit $(i, i+1)$ si et seulement si $(\alpha_i - \alpha i + 1)a \neq \alpha_i 1$. Mais, si l,application $(\alpha_i - \alpha_{i+1})$ est une permutation sur l'ensemble Q , alors, $a \neq (\alpha_i - \alpha_{i+1})^{-1}\alpha_i 1$.

Nous prouvons que $a \neq 0 et 1$. Si $a = 0$, alors $\alpha_i^{-1}(\alpha_i - \alpha_{i+1})0 = 0$, puisque $\alpha_i, (\alpha_i - \alpha_{i+1})$ sont des automorphismes du groupe abélien $(Q, +)$. Donc $0 = 1$ et nous avons une contradiction. Si $a = 1$, alors $(\alpha_i - \alpha_{i+1})1 = \alpha_i 1$, $\alpha_i 1 - \alpha_{i+1}1 = \alpha_i 1, \alpha_{i+1}1 = 0, 0 = 1$ et on a encore une contradiction.

Donc si l,application $(\alpha_i - \alpha_{i+1})$ est une permutation sur l'ensemble Q, alors il existe un élément $a \in Q, a \neq 0, a \neq 1$ tel que $(\alpha_i - \alpha_{i+1}) = \alpha_i 1$. Dans ce cas, le code (Q, g) ne peut pas exactement détecter une erreur phonétique) l'endroit $(i, i+1)$.

3.2.4. Quelques exemples de codes détecteurs d'erreurs

1. On peut proposer un code C sur $_{10}$ avec l'équation de contrôle suivante : $x_1 + \alpha x_2 + \beta x_3 + x_4 + \alpha x_5 + \beta x_6 + ... \equiv 0 mod(10)$. Où $\alpha = (087639125)(4)$, $\beta = (045781632)(9)$. Ce code ne detecte pas deux erreurs de transposition, deux erreurs jumelles de transposition, pas plus de huit erreurs jumelles sautées à n'importe quel endroit de la forme $(i, i+1), (i, i+2)$. Preuve : Les tables de cayley des quasi-groupes $x.y = x + \alpha y$, $x \circ y = \alpha x + \beta y$ et $x * y = \beta x + y$ sont suivantes :

.	0	1	2	3	4	5	6	7	8	9
0	8	2	5	9	4	0	3	6	7	1
1	9	3	6	0	5	1	4	7	8	2
2	0	4	7	1	6	2	5	8	9	3
3	1	5	8	2	7	3	6	9	0	4
4	2	6	9	3	8	4	7	0	1	5
5	3	7	0	4	9	5	8	1	2	6
6	4	8	1	5	0	6	9	2	3	7
7	5	9	2	6	1	7	0	3	4	8
8	6	0	3	7	2	8	1	4	5	9
9	7	1	4	8	3	9	2	5	6	0

∘	0	1	2	3	4	5	6	7	8	9
0	2	4	8	0	3	5	1	6	9	7
1	6	8	2	4	7	9	5	0	3	1
2	9	1	5	7	0	2	8	3	6	4
3	3	5	9	1	4	6	2	7	0	8
4	8	0	4	6	9	1	7	2	5	3
5	4	6	0	2	5	7	3	8	1	9
6	7	9	3	5	8	0	6	1	4	2
7	0	2	6	8	1	3	9	4	7	5
8	1	3	7	9	2	4	0	5	8	6
9	5	7	1	3	6	8	4	9	2	0

$*$	0	1	2	3	4	5	6	7	8	9
0	4	5	6	7	8	9	0	1	2	3
1	6	7	8	9	0	1	2	3	4	5
2	0	1	2	3	4	5	6	7	8	9
3	2	3	4	5	6	7	8	9	0	1
4	5	6	7	8	9	0	1	2	3	4
5	7	8	9	0	1	2	3	4	5	6
6	3	4	5	6	7	8	9	0	1	2
7	8	9	0	1	2	3	4	5	6	7
8	1	2	3	4	5	6	7	8	9	0
9	9	0	1	2	3	4	5	6	7	8

Dans le quasi-groupe $(Q, .)$ seuls les éléments 7et 8 sont permutables : $8.7 = 7.8 = 4$. Donc ce code ne détecte pas seulement les erreurs de transpositions $78 \longrightarrow 87$ et $87 \longrightarrow 78$ aux endroits de forme $(1 + 3j; 2 + 3j)$ pour tout j approprié. Dans ces endroits, ce code ne detecte pas les éléments suivants erreurs jumelles :

(a) $00 \longleftrightarrow 44$ (c'est à dire $00 \longrightarrow 44, 44 \longrightarrow 00$)

(b) $11 \longleftrightarrow 77$

(c) $55 \longleftrightarrow 88$.

Dans le le quasi-groupe $(Q, \circ)$ seuls les éléments 1 et 8 sont permutables. Donc le code C ne détecte pas les erreurs de transposition $18 \longleftrightarrow 81$ aux endroits de la forme $(2+3j; 3+3j)$ pour tout j convenable. Dans ces endroits, le code ne détecte pas non plus les erreurs jumelles $11 \longleftrightarrow 88$.

Seuls les éléments 4 et 7 commutent dans le quasi-groupe $(Q, *)$. Donc le code C ne détecte pas les erreurs de transposition $47 \longleftrightarrow 74$ aux endroits de la forme $(3 + 3j; 4 + 3j)$ pour tout j convenable. Dans ces endroits, le code ne détecte pas les huit erreurs jumelles suivantes : $33 \longleftrightarrow 77, 44 \longleftrightarrow 66 \longleftrightarrow 88$.

Aux endroits de la forme $(1 + 3j; 3 + 3j)$, le code C ne peut pas détecter le même ensemble d'erreurs qu'aux endroits de la forme $(3 + 3j; 4 + 3j)$, aux endroits de la forme $(2+3j; 4+3j)$ le code C ne peut pas détecter les même ensembles d,erreurs que

dans les lieux de la forme $(1+3j; 4+3j)$, et aux endroits de la forme $(3+3j; 5+3j)$ le code C ne peut pas détecter les même ensembles d'erreurs que dans les lieux de la forme $(2+3j; 3+3j)$.

2. "European Article Number code(EAN) et (après avoir ajouté 0 comme le premier chiffre) le Universal Product Code(UPC) avec $G = (\mathbb{Z}_{10}, +), n = 13, e = 0, \delta_{2i+1}(a) = a = L_1(a) et \delta_{2i}(a) = 3a = L_3(a)...$"

   Eb d'autres termes , le code EAN est le code avec l'équation de contrôle $x_1 + 3x_2 + x_3 + 3x_4 + x_5 + 3x_6 + x_7 + 3x_8 + x_9 + 3x_{10} + x_{11} + 3x_{12} + x_{13} = 0$ où $x_i \in \mathbb{Z}_{10}, i \in [|1, 13|]$.

   Notons que ce système ne detecte pas les erreurs de transpositions adjacentes $...ab... \longrightarrow ...ba...$, pour $|a - b| = 5$.

   Preuve :

   Ce système ne detecte pas les erreurs de le forme $...acb... \longrightarrow ...bca...$, c'est à dire les soi-disant erreurs de transposition de saut pour toute paire d'éléments $a, b \in \mathbb{Z}_{10}$. En effet, nous avons la transposition par saut $...acb... \longrightarrow ...bca...$ Par passage au fonctionnement en groupe nous avons les expressions suivantes : $...3a+c+3b..., ...3b+c+3a...ou...a+3c+b..., ...3b+c+3a...oua+3c+b... = ...b+3c+a...$ Par conséquant le code EAN ne détecte pas les "les erreurs de transposition de saut."

   Le code EAN ne détecte pas les erreurs jumelles $...aa... \longrightarrow ...bb...$ pour $|a - b| = 5$. en effet, en passant à l'opération de groupe, on obtient les expressions suivantes : $... + 3a + a + ... = ... + 4a..., ... + 3b + b + ... = ... + 4b + ...ou... + a + 3a + ... = ... + 4a + ..., ... + b + 3b + ... = ... + 4b + ...$

   Dans le cas où $4a = 4b$, le code EAN ne détectera pas les erreurs jumelles . On peut réécrire la dernière égalité sous cette forme : $4(a - b) \equiv 0 mod(10)$. Donc $(a - b) \equiv 0 mod(5), |a - b| = 5$ puisque $a, b \in 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$

   Par analogie, il est possible de prouver que le code EAN ne peut paas également détecter les erreurs jumelles de saut $(...aca... \longrightarrow ...bcb...)$ pour toute paire d'élément $a, b \in \mathbb{Z}_{10}$ tel que $|a - b| = 5$.

   Donc le code EAN ne détecte pas les erreurs de transpositions adjacentes, les erreurs

jumelles ou les erreurs jumelles de saut pour toute paire d'éléments $a, b \in \mathbb{Z}_{10}$ tel que $|a - b| = 5$ et il ne détecte pas les erruers de transpositions de saut pour toute paire d'éléments $a, b \in \mathbb{Z}_{10}$.

Chapitre Quatre

Présentation de GAP et implémentation de quelques cryptosystèmes basés sur les quasi-groupes

4.1 PRÉSENTATION DE GAP

4.1.1. Historique

La première version publiée, GAP 2.4, remonte à 1998 [8]. Toutefois, Joachim Neubuser et ses étudiants de l'université d'AIX-La-Chapelle ont commencé à travailler dessus dès 1986 [8]. Le projet a débuté comme projet de fin d'études pour quatre étudiants : Johannes Meier, Alice Niemeyer, Werner Nickel et Martin Schonert. L'interpréteur est sous licence GPL , les paquets de la bibliothèque peuvent être sous d'autres licences. GAP a d'abord été développé à l'université d'AIX-La-Chapelle (Allemagne) de 1986 à 1997 puis jusqu'en 2005 le développement a été réalisé à l'université de ST Andrews (Royaume Uni)[8]. Depuis 2005, il est réalisé en partenariat par quatre centres : les deux précédents, ainsi que l'université du colorado à Fort Collins (Etats Unis) et l'université de Brunswick (Allemagne)[8]. En 2008, GAP a recu de l'ACM|SIGSAM le Richard Dimick Jenks Memorial Price for Excelence in Software Engineering applied to comuter Algebra[8]. La dernière version de GAP publiée 4.12.2 remonte à décembre 2022 [8].

4.1.2. Motivation et Contexte

La mise en œuvre de nouveaux algorithmes et la puissance toujours croissante des ordinateurs modernes ont un impact révolutionnaire sur la science. Les mathématiques, dont la nature artistique et créative aurait pu être considérée comme une barrière à l'influence des ordinateurs n'échappent pas à cette révolution. Une grande partie des mathématiques publiée de nos jours fait largement appel aux calculs automatiques. Fait remarquable, une partie substantielle des résultats ne pourrait pas être obtenue sans elle ou prendrait un temps déraisonnable. Cela comprend de nombreux résultats théoriques exceptionnels qui dépendent des ordinateurs pour vérifier une hypothèse ou effectuer des calculs particuliers tout en prouvant un théorème. L'utilisation des ordinateurs se développe également dans l'enseignement des mathématiques à tous les niveaux. En ce qui concerne l'algèbre abstraite, un système d'algèbre informatique difficile à ignorer est GAP. Il est largement utilisé , lors de l'enseignement de l'algèbre élémentaire et dans la littérature scientifique avancée, testant et réfutant les conjectures.

4.1.3. Définition

GAP (abréviation de groups, algorithms, programming) est un logiciel libre, pour l'algorithmique algébrique. Il possède une bibliothèque très riche, notamment pour la théorie des nombres, la combinatoire, le calcul sur les groupes finis, les polynômes, mots, automates, graphes et codes.

4.1.4. Comment y accéder

GAP est un programme gratuit et est disponible en téléchargement sur
https ://www.gap.system.org/. Sur cette page web, les utilisateurs peuvent accéder à un référentiel de développement hébergé sur Github, contribuer à l'amélioration du logiciel et poser des questions sur ses fonctionnalités. Il est disponible sous windows et les systèmes de type UNIX (dont Linux et Mac Os X). La version pour Mac Os 9 n'est plus fournie depuis GAP 4.5. Le langage a une syntaxe assez proche du logiciel Maple. GAP est également intégré aux logiciels SageMath et Wins.

4.1.5. Commande de base de GAP

4.2 Loops : Calcul avec des quasi-groupes dans GAP

4.2.1. Brève description

Loops est un package pour GAP4 dont le but est de fournir aux chercheurs en algèbre non-associative un outil de calcul puissant concernant les boucles finies et les quasi-groupes ; et d'étendre GAP vers le domaine des structures non-associative. Le package se compose de trois parties complémentaires : des méthodes de base pour les quasi-groupes et les boucles, les méthodes spécifiques pour les boucles (principalement les boucles de Moufang), des bibliothèques de petites boucles.

4.2.2. Création des quasi-groupes et de boucles

Les quasi-groupes et les boucles sont représentés dans GAP par des tables de cayley. Lorsque Q est un quasi-groupe d'ordre n, la table de cayley associée est un tableau $n \times n$ de symboles $1, ..., n$ tel que si et seulement si l'entrée de la ligne i et de la colonne j est k. De même pour les boules.
La table de cayley peut être saisie manuellement ou lu à partir d'un fichier.

4.2.2.1. Tester les tables de cayley

Les opérations synonymes suivantes testent si une table de multiplication est une table de multiplication d'un quasi-groupe ; c'est à dire un carré latin de symbole $1, ...$:

• ISQuasigroupTable(L)A

• ISQuasigroupCayleyTable(L)A

Un carré latin sur l'alphabet $1, ...n$ est dit normalisé si la première ligne et la première colonne lisent $1, ...n$. La table de cayley d'une boule n'est donc qu'un autre nom pour un carré latin normalisé.

Les opérations suivantes testent si L est un carré latin normalisé :

• ISTableLoop(L)A

• ISLoopCayleyTable(L)A

Un carré latin peut être normalisé en permutant ses colonnes de sorte que la première ligne indique $1, ...n$ puis en permutant ses lignes de sortent que la première colonne indique $1, ...n$.

NB : Il est important de noter que les lignes ou les colonnes soient permutées en premier.

4.2.2.2. Création manuelle de quasi-groupe et de boucles

Lorsque L est un carré latin sur $1, ...n$. le quasi-groupe correspondant est obtenu avec :

• QuasigroupByCayleyTable(L)F

• QuasigroupByCayleyTable(L,<nom>)F

Si <nom> est donné, alors la sortie des éléments du quasi-groupe aura la forme <nom>1,<nom>2.

Lorsque L est normalisé, la boucle correspondante est renvoyée par :

• BoucleParCayleyTable(L)F.

4.2.2.3. Créer des quasi-groupes et des boucles à partir d'un fichier

Taper manuellement une grande table de multiplication est fastidieux et source d'erreurs. Nous avons donc inclus un algorithme universel dans Loops qui lit les tables de multiplication de quasi-groupes à partir d'un fichier. Au lieu d'ecrire un algorithme séparé pour chaque format commun, notre algorithme s'appuie sur l'utilisateur pour fournir un peu d'informations sur le fichier d'entrée. Voici un aperçu de l'algorithme :

4.2.2.4. Algorithme

Entrée : nom de fichier F, chaine D

Étape 1 : lit tout le contenu de F dans une chaine S

Étape 2 : remplacez tous les caractères de fin de ligne dans S par des espaces

Étape 3 : Remplacez par des espaces tous les caractères de S qui apparaissent dans D.

Étape 4 : Divisez S en sous chaines maximales sans espaces, appelées morceaux.

Étape 5 : reconnaitre les morceaux distincts .Soit n le nombre de morceaux distincts.

Étape 6 : si le nombre de morceaux n'est pas n^2 signaler une erreur.

Étape 7 : Construire la table de multiplication en attribuant les valeurs numériques $1, ...n$ en morceaux, selon leur position parmi les morceaux distincts.

Exemple 4.2.1.

input file	string D	resulting mult. table	comments
0 1 2 1 2 0 2 0 1	""	1 2 3 2 3 1 3 1 2	Data does not have to be arranged into an array of any kind.
red green green red	""	1 2 2 1	Chunks can be any strings.
[[0, 1], [1, 0]]	"[,]"	1 2 2 1	A typical table produced by GAP is easily parsed by deleting brackets and commas.
x&y&z\hline y&z&x\hline z&x&y	"&\\einlh"	1 2 3 2 3 1 3 1 2	A typical TEX table with rows separated by lines. We must use "\\" in D because "\\" represents the string "\" in GAP.
I am as mad as I say I am	"day"	1 2 3 2 3 1 3 1 2	Just for fun.

4.2.3. Attributs de base

La liste des éléments d'un quasi-groupe Q est obtenue par la commande usuelle :

1. Elements(Q) A

   La table de cayley d'un quasi-groupe est renvoyée avec

2. CayleyTable(Q) A

   L'élément neutre d'une boucle L est obtenu via

3. One(L) A

   Si vous voulez savoir si un quasi-groupe Q a un élément neutre, vous pouvez le savoir avec la fonction standard pour les magmas

4. MultiplicativeNeutralElement(Q) A

   La taille d'un quasi-groupe Q est calculée par

5. Size(Q) A

   lorsque L est une boucle puissance associative (c'est à dire que les ordres des éléments spnt bien définis dans L), l'exposant de L est de plus petit entier positif divisé par les ordres de tous les éléments de L. L'attribut suivant calcule l'exposant sans texter la puissance associativité :

6. Exponent(L) A

Exemple 4.2.2. *gap> Q := QuasigroupByCayleyTable([[1,2], [2,1]]);*
<quasigroup of order 2>
gap> [IsQuasigroup(Q); IsLoop(Q); Size(Q); Elements(Q);]
$[true, false, 2, [q1; q2]]$
gap> IsQuasigroupElement(Element(Q)[2]);
true
gap> CayleyTable(Q);
$[[1, 2], [2, 1]]$

4.2.4. Les opérations arithmétiques de base

Chaque élément de quasi-groupe dans GAP sait à quel quasi-groupe il appartient. Il est donc possible d'effectuer des opérations arithmétiques avec des éléments de quasi-groupe sans se référer au quasi-groupe. Tous les éléments impliqués dans le calcul doivent appartenir au même quasi-groupe.

4.2.4.1. Multiplication

Deux éléments d'un même quasi-groupe sont multipliés par $x * y$ dans GAP. Comme la multiplication des éléments est ambiguë dans le cas non associatif, nous multiplions toujours l'élément de gauche à droite (c'est à dire $x * y * z$ signifie $(x * y) * z$). Bien sur, on peut spécifier l'association par des parenthèses.

4.2.4.2. La Division

Les algébristes universels introduisent deux opérations supplémentaires pour les quasi-groupes. Asavoir la division gauche $x\backslash y$ satisfaisant $x \bullet (x\backslash y) = y$, et la division droite satisfaisant $(x/y) \bullet x = yx$. Ces deux opérations peuvent être trouvées dans LOOPS comme suit :

1. LeftDivision(x, y) 0
2. RightDivision(x, y) 0

   Lorsque Q est un quasi-groupe, $x \in Q$ et S est une liste d'éléments de Q alors,
3. LeftDivision (S, x) 0
4. LeftDivision (x, S) 0
5. RightDivision(S, y) 0
6. m RightDivision(y, S) 0

Renvoie la liste des éléments obtenus en effectuent la division respective de S par x, ou de x par S, en utilisant un élément de S à la fois.

4.2.4.3. Puissances et inverses

Les puissances des éléments ne sont pas bien définies dans les quasi-groupes, car les parenthèses peuvent avoir de l'importance meme pour un seul élément. On dit que le quasi-groupe Q est mono-associatif, si pour tout $x \in Q$, le sous-magma engendré par x est associatif. De meme, la bouble L est dite de puissance associative, si pour tout $x \in L$, la sous boucle engendrée par x est associative.
Pour les magmas et les exposants entiers positifs, GAP définit les puissances de la manière suivantes : $x^1 = x, x^{2k} = (x^k)(x^k)$ et $x^{2k+1} = (x^{2k}).x$ pour un entier positif k.
Nous avons les commandes suivantes :

1. LeftInverse(x) 0
2. RightIverse(X) 0
3. Inverse(x) 0

Exemple 4.2.3. *gap> M :=MoufangLoop(12,1);; x :=M.2;*

12

gap> [x ∗ M.3,x^ 2, x^(-1), Inverse(x)];

$[14, 11, 12, 12]$

gap> One (M) = LeftDivision(x,x);

true

4.2.5. Générateurs

Les deux attributs suivants sont des synonymes de GeneratorsofMagma.

1. GeneratorsOfQuasigroup(Q) A
2. GeneratorsOfLoop(L) A

Comme d'habitude dans GAP, on peut se référer au ième générateur d'un quasi-groupe Q par Qi. Notons que ce n'est pas nécessairement le cas que $Qi = Elements(Q)[i]$, puisque l'ensemble des générateurs peut être un sous-ensemble propre des éléments.

Chapitre Cinq

Conclusion

Dans ce mémoire, nous avons présenté, des notions de base sur la théorie des quasi-groupes, de la cryptographie , des codes détecteurs d'erreurs ; nous avons également présenter quelques cryptosystèmes , et codes détecteurs d'erreurs basés et les quasi-groupes enfin le langage de programtion GAP.

Nous envisagerons par la suite construire nos propres crytosystèmes sécurisés basés sur les quasi-groupes.

✠ Bibliographie ✠

[1] D. F. Beckley, "Phonetic error detection in data transmission systems," *IEEE Transactions on Communications*, vol. 23, no. 2, pp. 129–135, 1975.

[2] R. H. Bruck, "Some results in the theory of quasigroups," *Transactions of the American Mathematical Society*, vol. 55, pp. 19–52, 1944.

[3] T. Evans, "Non-associative algebraic structures and their applications," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 53, no. 4, pp. 759–770, 1957.

[4] R. M. Falcon, "Latin squares and Hadamard quasigroups in cryptographic randomization," *Journal of Cryptographic Engineering*, vol. 13, no. 1, pp. 45–61, 2023.

[5] R. Moufang, "Uber die Komposition der quadratischen Formen," *Mathematische Annalen*, vol. 110, pp. 416–430, 1935.

[6] E. Ochodkova and V. Snavel, "Secure encoding using quasigroup transformations," *Advances in Mathematics of Communications*, vol. 1, no. 1, pp. 27–36, 2007.

[7] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, 1948.

[8] J. Verhoeff, "Error detecting decimal codes for arithmetic operations," *IEEE Transactions on Computers*, vol. C-18, no. 12, pp. 1056–1063, 1969.