UNIVERSITE CHEIKH ANTA DIOP DE DAKAR

FACULTE DES SCIENCES ET TECHNIQUE

ECOLE DOCTORALE

MATHEMATIQUES, INFORMATIQUE ET SCIENCES CONNEXES

THÈSE DE DOCTORAT UNIQUE ÈS SCIENCES MATHÉMATIQUES

pour obtenir le grade de

DOCTEUR de l'**Université Cheikh Anta Diop De Dakar**

Option : **Codage, Cryptologie, Algèbre et Applications**

Titre :

ENTIERS DUAUX ET CRYPTOGRAPHIE À CLE PUBLIQUE

Présentée et soutenue publiquement le Samedi 23 Mars 2013 à 10h à l'amphi 7

par : Mamadou Ghouraïssiou CAMARA

Sous la direction de **Dr. Djiby SOW**

devant le jury composé de :

Président	Mamadou Sangharé	Professeur	Univ. Cheikh A. Diop de Dakar
Rapporteurs	Patick Solé	Directeur de Recherche	CNRS Labo LTCI (France)
	André Leroy	Professeur	Univ. Artrois Lens (France)
Examinateurs	Cheikh Thiécomba Gueye	Professeur	Univ. Cheikh A. Diop de Dakar
	Oumar Diankha	Maître de conférences	Univ. Cheikh A. Diop de Dakar
	Elhadj Momo Bangoura	Directeur Nationale	Enseignement Superieur (Guinée)
Directeur de Thèse	Djiby Sow	Maître de conférences	Univ. Cheikh A. Diop de Dakar

ANNEE ACCADEMIQUE : 2012-2013

DEDICACES

Qu'ALLAH soit Loué

Je rends grâce à DIEU

PAIX et SALUT sur le PROPHETE MOHAMED

je dédie cette thèse :

- A mes parents : ma mère Mouminatou BALDE et mon père Mamadouba CAMARA ;

- A ma femme Aïssatou Bella BALDE ;

- A mes soeurs et frères ;

- A mes homonymes, mes cousines et mes cousins ;

- A mes nièces, mes neveux, mes tantes et mes oncles ;

- A mes amis (es) et camarades de tous horizons ;

- A tous les copins et amies de Baguiré-Thiagna-Megnéré

- A mes collègues ;

- A mes maîtres et professeurs ;

- A tout ceux qui m'ont soutenu de près ou de loin ;

- A toutes ces personnes qui m'ont tout donné sans rien demander.

REMERCIEMENTS

Je remercie ALLAH Le Tout Puissant

Le Clément et Le Miséricordieux

Paix et Salue sur le Prophète

Je tiens à remercier vivement Monsieur le Professeur **Mamadou SANGHARE, Directeur de l'école Doctorale de Mathématiques et Informatique**, pour l'honneur qu'il m'a fait en acceptant de présider mon jury de thèse.

Je tiens à remercier très sincèrement le Professeur **Djiby SOW**qui a dirigé mes travaux de recherche. Je lui exprime ma profonde gratitude pour tout ce qu'il a fait pour moi et que je ne saurais entièrement lister ici. Particulièrement, je le remercie très sincèrement pour l'execellente formation que j'ai reçue auprès de lui sur tous les plans (didactique, recherche, humain,..). J'ai apprécié tout particulièrement sa patience, sa capacité d'écoute et sa grande disponibilté.

Je remercie très sincèrement les Professeurs **Patick Solé (France), André Leroy (France), Cheikh Thiecoumba Gueye (Senegal), Oumar Diankha (Senegal) et Elhadj Momo Bangoura (Guinée)** pour avoir accepté d'être rapporteurs de cette thèse ou examinateurs dans ce jury.

Je remercie :

- tous les membres du Laboratoire d'Algèbre, de Cryptologie, de Géométrie Algébrique et Applications (LACGAA),

- mes compagnons de tous les jours à la fac, je veux nommer Demba Sow, Ahmed Youssef Yoklil, André Mialébama, Ahmed Khalifa, Abdoul Aziz Ciss, Erness Bouzaboabo, Raoul Thissouba.

- tous mes enseignants de la Faculté des Sciences et Technique de l'université Cheikh Anta Diop de Dakar.

- tous mes enseignants de l'université Gamal Abdel Nasser de Conakry, très particulièrement à **Dr. Lanciné Fofana et Dr. Elhadj Momo Bangoura**.

Table des matières

INTRODUCTION

NTRU est un cryptosystème à clé publique proposé en 1996 par J. Hoffstein, J. Pipher et J. H. Silverman dans [11]. NTRU est basé sur la géométrie des nombres et l'arithmétique dans les anneaux. Concrètement c'est un cryptosystème très rapide basé sur l'anneau des polynômes $\frac{\mathbb{Z}[X]}{\left(X^N-1\right)}$. La géométrie des nombres a été inventée en 1896 par Minkowski. Une des préoccupations consiste à donner une minoration du nombre des points d'un réseau de $\mathbb{R}^n$ qui appartiennent à une partie convexe donnée, et ce, en fonction de son volume. Les problèmes difficiles liés à NTRU découlent des problèmes difficles liés au réseaux comme les suivants (voir [4], [10], [25], [23])

- Problème **SVP, Shortest Vector Problem** (problème du plus court vecteur) : Etant donnée une base B d'un réseau L, trouver un vecteur non nul de L le plus court possible pour la norme euclidienne ;
- Problème **CVP, Closest Vector Problem** (problème du plus proche vecteur) : Etant donnée une base B d'un réseau L et un vecteur $v \in L$, trouver un vecteur de L le plus proche possible de v pour la norme euclidienne.

NTRU est l'objet de quelques attaques dont les plus dangereuses sont "l'attaque du milieu" et "l'attaque reseau" sur la clé publique et sur le texte chiffré. Mais un choix convenable de la taille des paramètres permet d'éviter ces attaques. Un choix correct de padding pour le chiffrement NTRU appelé NEAP a été proposé dans [14] avec une preuve de sécurité en présence d'un échec de déchiffrement. En 2005, un algorithme a été conçu par Howgrave *et al.* [15], pour un bon choix de la taille des paramètres par rapport aux niveaux de sécurité désirés. Comme les échecs de déchiffrement sont une insuffisance pour NTRU et toutes ses variantes, Dwork *et al.* ont conçu en 2007 une méthode pour "Immuniser les processus de chiffrement contre les erreurs de déchiffrement" [6]. NTRU attire l'interêt des chercheurs parce que c'est un schéma post-quantique, c'est à dire, les attaques connues sur les réseaux concernant NTRU restent exponentielles sur un ordinateur quantique. Raison pour laquelle plusieurs auteurs ont déjà travaillé sur la généralisation du chiffrement de NTRU dans des sens differents sur des anneaux.

Nous avons les généralisations suivantes :

- En 2002, William D. Banks et Igor E. Shparlinski ont proposé dans [1] une nouvelle variante de NTRU sans inverse de polynômes dans le même anneau que NTRU classique. Cette généralisation résiste aux attaques connues sur le NTRU clasique, telle que l'attaque sur les reseaux. Mais cette généralisation est moins efficace que le schema classique de NTRU car la longueur de la clé publique et celle du texte chiffré sont approximativement doublées.
- En 2002, Gaborit, Ohler et Solé ont proposé CTRU où l'anneau $\mathbb{Z}$ dans NTRU est ramplacé par $\mathbb{A} = \mathbb{F}_2[T]$. CTRU a été cassé par Kouzmenko dans sa thèse unique [18] en 2006 et par Vats [32] en 2008.

matrices à coefficients dans le même anneau que NTRU classique. Le cryptosystème MaTRU utilise une transformation linéaire plus efficace et fournit un niveau de sécurité comparable à NTRU.

- En 2006, Kouzmenko dans sa thèse unique [18] a proposé une variante où l'anneau $\mathbb{Z}$ de NTRU est remplacé par l'anneau des entiers de Gauss $\mathbb{Z}+i\mathbb{Z} = \{a+ib \ / \ a,b \in \mathbb{Z}, \ \ i^2=-1\}$. Ce système est légèrement plus resistant pour l'attaque sur les reseaux que NTRU mais il est moins efficace que NTRU.
- En 2009, Nevins, Karimianpour et Miri [24] ont proposé une nouvelle variante où l'anneau $\mathbb{Z}$ de NTRU est remplacé par l'anneau des entiers de Einstein $\mathbb{Z}+w\mathbb{Z} = \{a+wb \ / \ a,b \in \mathbb{Z}, \ \ i^2=-1, \ w = e^{2i\frac{\pi}{3}}\}$. Ce système est similaire à NTRU sur les entiers de Gauss.
- En 2009, Malekian, Zakerolhosseini et Mashatan [21] ont proposé une nouvelle variante où l'anneau $\mathbb{Z}$ de NTRU est remplacé par l'anneau des quaternions $\mathbb{H} = \{a+ib+jc+kd \ / \ a,b,c,d \in \mathbb{Z}, \ \ i^2 = j^2 = k^2 = ijk = -1\}$. Ce système est aussi similaire à NTRU sur les entiers de Gauss ou entiers de Eisenstein.

Toutes ses généralisations de NTRU ont été réalisées sur des anneaux intègres tels que $\mathbb{Z}$, $\mathbb{Z}[i]$, $\mathbb{Z}[w]$ et $\mathbb{H}$.

Le but de ce travail est d'utiliser l'anneau (avec des diviseurs de zeros) $\mathbb{Z}+\epsilon\mathbb{Z}, \ \epsilon^2=0$ (appelé l'anneau des entiers Duaux) pour en faire une nouvelle version de NTRU.

Pour atteindre ce but, nous avons dans un premier temps étudié certaines propriétés arithmetiques élémentaires de l'anneau des entiers duaux [2], puis nous nous sommes servis de ces resultats pour proposer une généralisation de NTRU.

Ce travail est structuré comme suit : premièrement, nous présenterons une vue d'ensemble de la cryptographie, ensuite nous parlerons des réseaux et des techniques de réduction de réseaux sur lesquels sont basés la sécurité de notre système DTRU. Après, nous décrirons le cryptosystème NTRU classique et quelques attaques connues contre lui. Enfin nous présenterons nos differentes contributions regroupées sur les chapitres 3 et 4.

Dans le chapitre 3, le challenge était de proposer une arithmétique sur l'anneau $\mathbb{D} = \frac{\mathbb{Z}[\epsilon]}{(\epsilon^2)}$ non intègre (avec des diviseurs de zéros). Pour cela nous avons pu :

1. sur l'arithmétique dans $\mathbb{D} = \frac{\mathbb{Z}[\epsilon]}{(\epsilon^2)}$
   - Caractériser les éléments inversibles, les diviseurs de zéro et les éléments irréductibles de $\mathbb{D}$,
   - introduire les concepts de pseudo-factorisation, de pseudo-division euclidienne et de pseudo-norme,
   - proposer un algorithme de pseudo-division qui renvoie toujours le même reste car le reste de la Pseudo-division en génerale n'est pas unique.
   - proposer un algorithme pour inverser un élément(inversible) dans $\frac{\mathbb{D}}{(z)}$ où $z \in \mathbb{D}$.
2. sur l'arithmetique dans $\mathbb{D}[x]$ où $\mathbb{D} = \frac{\mathbb{Z}[\epsilon]}{(\epsilon^2)}$
   - proposer un algorithme pour inverser un polynôme $f(x)$ dans $\frac{\mathbb{D}[x]}{(g(x))}$ où $f(x)$ et $g(x)$ appartiennent à $\mathbb{D}[x]$ et dans $\frac{\frac{\mathbb{D}}{(z)}[x]}{(g(x))}$ où $z \in \mathbb{Z}$,
   - caractériser les polynômes irréductibles

Tous ces résultats d'arithmétique ont été implémenté en C/C++ avec NTL de V. Shoup [31].

La principale difficulté était d'être capable de construire un algorithme de division avec un reste unique et de pouvoir inverser un polynôme à coefficients sur les entiers Duaux. Cependant, nous avons réussis à reprendre NTRU avec succes sur les entiers Duaux (appelé DTRU) en particulier, l'anneau quotient de l'anneau des entiers Duaux. Notre schéma a le même niveau de sécurité que NTRU classique mais il n'est pas plus efficace. Ce travail montre que NTRU peut fonctionner même si l'anneau contient des diviseurs de zéro ! Ce cryptosystème DTRU été implémenté en C/C++ avec NTL de V. Shoup [31].
Nous avons aussi repris le cryptosystème "NTRU sans inverse de polynômes" proposé par Banks et Shparlinski [1] sur l'anneau des entiers Duaux. Cette version est plus sûre que NTRU mais moins efficace.

COMPLEXITE ET CRYPTOGRAPHIE

Ce chapitre est une courte introduction de la théorie de la complexité (qui évalue le temps de calcul et l'espace mémoire nécessaire pour résoudre un problème algorithmique) et de la cryptographie (qui construit des protocoles et des algorithmes pour protéger des données et des communications).

Complexité

Le but de la théorie de la compléxité est :

- pour les algorithmes d'évaluer le coût de leur résolution. Le coût peut être le nombre d'opérations élmentaires (complexité temporelle) ou l'espace mémoire nécessaire (complexité en space) ;
- pour les problèmes (de décision), de les classifier suivant leur niveau de difficulté.

Classe de complexité

Soit $f : \mathbb{N} \to \mathbb{R}^+$ une fonction à valeurs positives.

Définition 0.0.1. *Nous définissons l'ensemble $\mathcal{O}(f(n))$ comme suit :*

$$\mathcal{O}(f(n)) = \{g : \mathbb{N} \to \mathbb{R}^+ | \exists n_0 \in \mathbb{N}, c \in \mathbb{R}^+ \text{ tel que } g(n) \leq cf(n), \forall n > n_0\}$$

On dit que g est dans grand $\mathcal{O}$ de f

Définition 0.0.2. *Nous définissons l'ensemble $\Omega(f(n))$ comme suit :*

$$\Omega(f(n)) = \{g : \mathbb{N} \to \mathbb{R}^+ | \exists c > 0, \exists n_0 \in \mathbb{N} \text{ tel que } g(n) \geq cf(n), \forall n > n_0\}$$

Proposition 0.0.1. *Soit $g : \mathbb{N} \to \mathbb{R}^+$ une fonction à valeurs positives.*

$$g \in \mathcal{O}(f) \Leftrightarrow f \in \Omega(g)$$

Preuve

Soit $g \in \mathcal{O}(f) \Leftrightarrow \exists n_0 \in \mathbb{N}, c \in \mathbb{R}^+ \text{ tel que } \forall n > n_0, g(n) \leq c.f(n) \Leftrightarrow f(n) \geq \frac{1}{c}.g(n) \Leftrightarrow f \in \Omega(g)$

Définition 0.0.3.

$$\Theta(f) = \mathcal{O}(f) \cap \Omega(f)$$

Proposition 0.0.2. .

$\Theta(f) = \{g : \mathbb{N} \to \mathbb{R}^+ \text{ tel que } \exists c_1, c_2, \exists n_0 \in \mathbb{N}^+, \forall n > n_0 \ c_1.f(n) \leq g(n) \leq c_2.f(n)\}$

Soient $f, g : \mathbb{N} \to \mathbb{R}^+$ deux fonctions à termes positifs.

On définit l'ordre partiel sur l'ordre de grandeur des fonctions comme suit :

$$\Theta(f) \leq \Theta(g) \ si \ f \in \mathcal{O}(g)$$

On a alors :

$$\Theta(f) = \Theta(g) \ si \ f \in \mathcal{O}(g) \ et \ g \in \mathcal{O}(f)$$

$$\Theta(f) < \Theta(g) \ si \ f \in \mathcal{O}(g) \ et \ g \notin \mathcal{O}(f)$$

Proposition 0.0.3. *Soit $f, g : \mathbb{N} \to \mathbb{R}^+$ deux fonctions à termes positifs.*

- *Si $\lim_{n\to\infty} \frac{g}{f} = c > 0$ (c constante) alors $g \in \Theta(f)$*
- *Si $\lim_{n\to\infty} \frac{g}{f} = 0$ alors $g \in \mathcal{O}(f)$ et $f \notin \mathcal{O}(g)$*
- *Si $\lim_{n\to\infty} \frac{g}{f} = +\infty$ alors $f \in \mathcal{O}(g)$ et $g \notin \mathcal{O}(f)$*

Nous allons comparer les algorithmes selon l'ordre de grandeur de leur compléxité.

Types de complexité

Notons D_n l'ensemble des données de taille inférieure ou égale à n, $\mathcal{C}(d)$ le coût associé à l'éxécution de la donnée d par un algorithme et $\mathcal{C}(n)$ la complexité de l'algorithme.

Définition 0.0.4. *La complexité dans le **pire des cas** est $\mathcal{C}(n) = Max\{\mathcal{C}(d), d \in D_n\}$*

Cette complexité est assez facile à calculer.

Définition 0.0.5. *Notons par $p(d)$ la probabilité d'obtenir la donnée d.*
*La complexité en **moyenne** est $\mathcal{C}(n) = \sum_{d \in D_n} p(d).\mathcal{C}(d)$*

Il faut signaler que la complexité en moyenne est souvent difficile à calculer car il faut connaître le distribution des données

Définition 0.0.6. *La complexité dans le **meilleur des cas** est $\mathcal{C}(n) = Min\{\mathcal{C}(d), d \in D_n\}$*

Définition 0.0.7. ***Classes de complexité***

1. Un algorithme est polynômial si sa complexité dans le pire des cas est de la forme $\mathcal{O}(n^k)$ où n est la taille de l'entrée et k une constante.
2. Un algorithme dont la complexité ne peut pas être majorée par un polynôme est appelé algorithme exponentielle.

1. Un **problème de décision** est un problème dont la reponse est **Oui** ou **Non**
2. La **classe de complexité P** est l'ensemble de tous les problèmes de décision que l'on peut résoudre en temps polynômial.
3. La **classe de complexité NP** est l'ensemble des problèmes de décision pour lesquels la reponse **Oui** peut être verifiée en temps polynomial en se servant d'une information supplementaire appelée "certificat".
4. La **classe de complexité Co-NP** est l'ensemble des problèmes de décision pour les quels la réponse **Non** peut être verifiée en temps polynomial en se servant d'une information supplementaire appelée "certificat".

Ainsi, nous avons $\mathbf{P} \subseteq \mathbf{NP}$. On ne sait pas si $\mathbf{P} = \mathbf{NP}$. Il est considéré comme l'un des problèmes les plus ardus de la théorie de la complexité. Enfin, donnons une définition des "problèmes les plus difficiles dans **NP**".

Définition 0.0.8. *Un problème dans **NP** est dit NP-complet si tout problème dans **NP** peut être réduit à lui en temps polynomial.*

Ainsi, trouver une solution en temps polynomial pour un problème NP-complet nous donnerait une solution en temps polynomial pour tous les problèmes de NP, et cela signifierait que $\mathbf{P} = \mathbf{NP}$. Par conséquent, les problèmes **NP**-complets sont les problèmes les plus difficiles dans **NP**. On estime que $\mathbf{P} \neq \mathbf{NP}$

Cryptographie

Il est considéré que la cryptographie a commencé depuis l'apparition de la première langue écrite. Les gens ont toujours essayé de développer des moyens d'écrire leurs secrets pour que eux seuls puissent les lire. A titre d'exemple, les Egyptiens communiquaient à l'aide de messages écrits en hiéroglyphes connus seulement par une partie de la population que sont les scribes. Dans les temps modernes, il a fallu des efforts considérables pour être en mesure de comprendre leur langue (à savoir, la découverte de la pierre de Rosette par Champollion).

Définition 0.0.9. *La cryptographie est l'étude des techniques mathématiques relatives aux aspects de la sécurité de l'information tels que la confidentialité, l'intégrité des données, l'authentifications des correspondants, et la non-répudiation.*

- *La confidentialité* est un service utilisé pour protéger le contenu des informations. Seules les personnes autorisées doivent pourvoir accéder aux informations ainsi protégées. Le chiffrement de l'information permet de résoudre le problème de la confidentialité.
- *L'intégrité des données* est un service qui traite de la modification non autorisée des données. Pour garantir l'intégrité des données, il faut avoir la capacité de détecter la manipulation des données par des tiers non autorisés. On résout ce problème à l'aide des *fonctions de hachage*, qui à une donnée de longueur arbitraire associent une donnée de taille fixe appelée *empreinte*.

et à l'information elle-même.

– *La non-répudiation* est un service qui empêche une entité de nier d'avoir commis une action.

Fonction à sens unique avec trappe

Fonctions à sens unique

Il existe un certain type de fonctions qui jouent un rôle significatif en cryptographie. Il s'agit les fonctions à sens unique.

Définition 0.0.10. *Une fonction $f : A \longrightarrow B$ est appellé fonction à sens unique si pour tout $x \in A$, il est facile de calculer $f(x)$ (complexité polynomiale) mais pour tout $y \in Im(f)$ il est mathématiquement impossible (complexité exponentielle) de trouver un $x \in A$ tel que $f(x) = y$.*

Fonction à sens unique avec trappes

Une fonction à sens unique avec trappe est une fonction à sens unique qui peut être inversée facilement, si on connaît un secret appelé trappe.

Soit S un ensemble fini d'éléments, une permutation p sur S est une bijection de S sur lui-même (c'est-à-dire $p : S \to S$).

Terminologie et concepts de bases

Donnons quelques définitions et concepts de bases.

1. Soit $\mathcal{A}$ un ensemble fini, appelé alphabet de définition.
2. Soit $\mathcal{M}$ un ensemble appelé espace des messages clairs où chaque message est une chaîne finie d'éléments de $\mathcal{A}$.
3. Soit $\mathcal{C}$ un ensemble appelé espace des messages chiffrés où chaque message chiffré est une chaîne finie d'éléments de $\mathcal{A}$.
4. Soit $\mathcal{K}$ un ensemble fini appelé espace des clés.
5. Chaque $k \in \mathcal{K}$, détermine une injection E_k de $\mathcal{M}$ vers $\mathcal{C}$, appelée fonction de chiffrement, k est appelé clé de chiffrement.
6. A chaque k est associé un $k^{'} \in \mathcal{K}$, tel que $D_{k'}$ dénote une injection de $\mathcal{C}$ vers $\mathcal{M}$, appelée fonction de déchiffrement, $k^{'}$ est appelé clé de déchiffrement.
7. Le processus consistant à appliquer une transformation E paramétrisée par k et notée E_k à un message $m \in \mathcal{M}$ est appelé chiffrement de m.
8. Le processus consistant à appliquer une transformation D paramétrisée par $k^{'}$ et notée $D_{k'}$ à un message chiffré $c \in \mathcal{C}$ est appelé déchiffrement de c.
9. Un schéma de chiffrement consiste en un quintuplet $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ tel que $\forall k, k^{'} \in \mathcal{K}$, il existe E_k et $D_{k'} : \forall x \in \mathcal{M} : D_{k'}(E_k(x)) = x$. Notons que si $\mathcal{M} = \mathcal{C}$, alors notre chiffrement est une permutation.

et le chiffrement asymétrique (ou la cryptographie à clé publique).

Cryptographie à clé secrète

Traditionellement, seulement les cryptosystèmes à clé secrète ont été utilisés. Cela signifie que toute personne en mesure de chiffrer un message sait automatiquement la façon de le déchiffrer, c'est à dire pour chaque paire de clés k et $k^{'}$, il est facile de déterminer k connaissant $k^{'}$ et inversement.

Cette méthode repose sur la possibilité d'échanger le secret de la clé privée entre les parties qui communiquent, ce qui est assez gênant pour de nombreuses utilisations modernes dans les cas où toutes les communications passent par un canal public comme l'internet.

Comme exemple de chiffrement symétrique nous allons parler de l'AES (Advenced Encryption Standard).

L'AES : Advenced Encryption Standard

L'AES comme son nom indique est un standard de chiffrement symétrique destiné à remplacer le DES (Data Encryption Standard) qui est victime de plusieurs attaques. DES utilise des clés de longueur 56 bits (64 bits au total dont 8 bits pour le contrôle de parité)

Description de l'AES

L'AES est un algorithme de chiffrement par bloc qui opère sur des blocs (des textes clairs) qu'il transforme en blocs chiffrés (des textes chiffrés) par une séquence de N_r rounds ou opérations, à partir d'une clé de 128, 192 et 256 bits. Suivant la taille de celle ci, le nombre de rounds diffère : respectivement 10, 12 et 14 rounds.

Un état (State) d'AES est une structure qui contient les résultats intermédiares. Une telle structure contient toujours 4 lignes et le nombre de colonnes égale la longueur de la clé divisée par 32.

Dans le cas d'une clé de 128 bits, un état (State) d'AES est un bloc de 128 bits, soit 16 octets, qu'ont peut réprésenter par un tableau de quatre lignes et quatre colones (noté 4x4) :

x_0	x_4	x_8	x_{12}
x_1	x_5	x_9	x_{13}
x_2	x_6	x_{10}	x_{14}
x_3	x_7	x_{11}	x_{15}

Arithmétique dans le corps $GF(2^8)$

Pour pouvoir effectuer des opérations algèbriques sur des octets (=chaîne de 8 bits) on utilise le corps $F_{256} = GF(2^8)$.

Le corps $GF(2^8)$ est un corps composé de 256 éléments. Un élément de $GF(2^8)$ peut être représenté par un polynôme de degré inférieur ou égal à 7, à coefficients dans $GF(2)$, le corps à deux éléments ou par un nombre binaire que l'on notera aussi sous forme hexadécimale.

Si $p(x) \in GF(2^8)$, alors $p(x)$ peut s'écrire comme suit :

$$a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$$

ou les $b_i \in \{0, 1\}$. Par exemple, considérons la valeur hexadécimale 67 qui correspond à $(01100111)_2$ en nombre binaire, ou encore au polynôme

$$x^6 + x^5 + x^2 + x + 1$$

Addition dans $GF(2^8)$. L'addition de deux éléments de $GF(2^8)$ correspond à la somme des polynômes correspondants. Les coefficients seront réduits modulo 2.

Exemple

$$(x^6 + x^5 + x^2 + x + 1) + (x^7 + x^3 + x^2 + x + 1) = x^7 + x^6 + x^5 + x^3$$

ou en notation binaire : $(01100111)_2 + (10001111)_2 = (11101000)_2$

Multiplication dans $GF(2^8)$. La multiplication de deux éléments de $GF(2^8)$ correspond à la multiplication des polynômes correspondants, modulo un polynôme irréductible $m(x)$ qui est fixé et qui est de degré 8 sur $GF(2)$. Pour le cas de AES, ce polynôme est $m(x) = x^8 + x^4 + x^3 + x + 1$ ou encore $11B$ en notation hexadécimal. Pour la multiplication au niveau des octets, la meilleur méthode est de procéder par additions et décalages.

La multiplication par x *dans* $GF(2^8)$ est un cas particulier de la multiplication. Soit $p(x) = a_7x^7 + a_6x^6+a_5x^5+a_4x^4+a_3x^3+a_2x^2+a_1x+a_0$, alors $x.p(x) = a_7x^8+a_6x^7+a_5x^6+a_4x^5+a_3x^4+a_2x^3+a_1x^2+a_0x$. Soit a_7 est nul, et $xb(x)$ est déjà réduit modulo $m(x)$, soit $a_7 = 1$, et il suffit de retrancher ou d'ajouter puisqu'on est dans $GF(2)$.

Inverse dans $GF(2^8)$. Tout polynôme $p(x)$ non nul de $GF(2^8)$ admet un inverse. Pour déterminer cet inverse on utilise l'algorithme d'Euclide Etendu, qui détermine deux polynômes $u(x)$ et $v(x)$ tel que

$$u(x)p(x) + v(x)m(x) = 1$$

L'élément $u(x)$ est l'inverse de $p(x)$ modulo $m(x)$.

Chiffrement/Déchiffrement AES

Le chiffrement de l'AES est décrit par le schema suivant :

- une addition initiale de la clé notée **AddRoundKey**, suivie de $N_r - 1$ rounds, chacune constituée de quatres opérations :
  1. **SubBytes** : est une fonction de substitution non linéaire lors de laquelle chaque octet est remplacé par un autre octet choisi dans une table dite de substitution (*Sbox).*
  2. **ShiftRows** : est une étape de transposition ou chaque élément de la matrice est décalé cycliquement à gauche d'un certain nombre de colonnes.
  3. **MixColumns** : est une opération de mélange sur les colonnes.
  4. **AddRoundKey** : est une étape où chaque octet de l'état est combiné avec la clef de round.
- Enfin, un round final qui correspond à un round dans lequel l'étape MixColumns est omise est appliqué.

également dans l'ordre inverse.

Description des opérations de l'AES

Etape SubBytes : Dans cette étape, chaque élément de la matrice State est remplacé par un autre élément choisi dans une table dite de substitution inversible notée *SBox* et ceci en deux étapes :

1. Calcul de l'inverse de chaque octet dans le corps F_{256} ; l'octet 00 est son propre inverse.
2. Transformation affine f de cet octet calculé. Cette transformation est définie comme suit :

$$b = f(a) \Longleftrightarrow \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Etape Shiftrows : Dans cette étape chaque élément d'une ligne de la matrice State est décalé cycliquement à gauche de respectivement 0, 1, 2 et 3 éléments. La prémière ligne n'est donc pas décalée.

Etape MixColumns : Dans cette étape on opère sur les colonnes de la matrice State en les traitant comme un polynôme $a(x)$ de degré 3 à coefficients dans F_{256}.

Présentons l'addition et la multiplication dans l'anneau $\frac{F_{256}[x]}{(x^4+1)}$ qui représente l'ensemble des polynômes de degrés inférieurs à 4 et à coefficients dans F_{256}.

L'addition est le XOR. La multiplication est la multiplication modulo $x^4 + 1$. Il faut noter que $x^i \ modulo \ (x^4 + 1) = x^{i \ modulo \ 4}$

Soient $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ et $b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$ deux éléments de $\frac{F_{256}[x]}{(x^4+1)}$, nous aurons :

$a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$.

$c(x) = a(x)b(x) \ modulo \ (x^4 + 1) = c_3x^3 + c_2x^2 + c_1x + c_0$ avec

$c_0 = (a_0b_0) \oplus (a_3b_1) \oplus (a_2b_2) \oplus (a_1b_3)$

$c_1 = (a_1b_0) \oplus (a_0b_1) \oplus (a_3b_2) \oplus (a_2b_3)$

$c_2 = (a_2b_0) \oplus (a_1b_1) \oplus (a_0b_2) \oplus (a_3b_3)$

$c_3 = (a_3b_0) \oplus (a_2b_1) \oplus (a_1b_2) \oplus (a_0b_3)$

car $x^i \ modulo \ (x^4 + 1) = x^{i \ modulo \ 4}$

Matriciellement, cette opération s'écrit :

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Dans MixColomns, on réalise l'opération suivante sur chaque colonne :

$(03x^3 + x^2 + x + 02) \times a(x) \ modulo \ (x^4 + 1)$

Sous forme matricielle on obtient :

$$\begin{pmatrix} 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \times \begin{pmatrix} \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Cryptographie à clé publique

En 1976, un nouveau type de cryptographie a été proposé par W. Diffie et M. Hellman appelé chiffrement à clé publique. Le principe est assez simple : soientt $\{E_e : e \in \mathcal{K}\}$ l'ensemble des fonctions de chiffrement et $\{E_d : d \in \mathcal{K}\}$ l'ensemble des fonctions de déchiffrement, où $\mathcal{K}$ est l'espace des clés. Considerons toute paire de transformations (E_e, E_d) associée au chiffrement/déchiffrement et on suppose que chaque paire a la propriété que connaissant E_e il est calculatoirement impossible (complexité exponentielle), étant donné aléatoirement un texte chiffré $c \in \mathcal{C}$, de trouver le message $m \in \mathcal{M}$ tel que $E_e(m) = c$. Cette propriété implique que étant donné un e il est calculatoirement impossible de déterminer la clé de déchiffrement correspondante. e et d sont simplement les fonctions qui decrivent respectivement, le chiffrement et le déchiffrement. E_e est considéré ici comme une fonction à sens unique avec trappe, avec d étant l'information (trappe) nécessaire pour le calcul de la fonction inverse et ainsi être capable de déchiffrer.

Maintenant, considérons la communication entre deux parties Aïssa et Oumar. Oumar le destinataire, selectionne une paire de clés (e, d). Oumar envoie la clé de chiffrement e (appelée clé publique) à Aïssa mais il garde la clé de déchiffrement d (appelé clé privée). Aissa peut maintenant envoyer un message chiffrer m à Oumar en appliquant la fonction de chiffrement déterminée par la clé publique de Oumar pour obtenir $c = E_e(m)$. Oumar déchiffre le texte chiffré en appliquant la transformation inverse D_d uniquement déterminé par la clé privée d.

Parmis les chiffrements asymétriques, nous allons présenter le chiffrement RSA[30] et le chiffrement Elgamal [8].

Chiffrement RSA

RSA(Rivest Shamir Adleman) est un algorithme de chiffrement asymétrique, très utilisé dans le commerce électronique, et plus généralement pour échanger des données confidentielles sur Internet. **RSA** utilise l'anneau $\frac{\mathbb{Z}}{n\mathbb{Z}}$.

Génération des Clés

Pour la génération des clés nous allons effectuer les opérations suivantes :

1. Choisir deux grands nombres premiers distincts p et q de même taille,
2. Calculer $N = pq$ (appelé *module du chiffrement*)
3. Calculer l'indicatrice d'Euler de N : $\varphi(N) = (p-1)(q-1)$
4. Choisir e(appelé exposant de chiffrement) tel que, $0 < e < \varphi(N)$ et $pgcd(e, \varphi(N)) = 1$
5. Trouver d tel que $0 < d < \varphi(N)$ et $ed \equiv 1$ *modulo* $\varphi(N)$

Le couple (e, N) est appelé clé publique, alors que le couple (d, N) est appelé clé privée.

Chiffrement avec la clé publique (e, N)

$$C = M^e \ modulo \ N$$

Déchiffrement avec la clé privée (d, N)

On vérifie que $M = C^d \ modulo \ N$

Notons que cette version purement mathématique n'est pas sur.

Chiffrement ElGamal

L'algorithme El Gamal, a été développé par Taher ElGamal en 1984. Il n'est pas breveté et peut être utilisé librement.

Introduisons d'abord le problème du logarithme discret

Soit G un groupe noté multiplicativement, $g \in G$ un élément d'ordre q (assez grand) et $H =< g >$ le sous-groupe engendré par g

Problème de logarithme discret (DLP) : Etant donné $y \in H$ existe-t-il $a \in [1, q-1]$ tel que $y = g^a$. Si oui, comment le calculer ? a est appelé le logarithme discret de y à base g, on note $a = log_g(y)$.

Exemples 0.0.1. *Soit G le groupe multiplicatif du corps $\frac{\mathbb{Z}}{p\mathbb{Z}}$ avec p un grand nombre prémier et α une racine primitive modulo p. Etant donné un entier B non divisible par p, le problème se résume à trouver un entier a vérifiant : $B = \alpha^a \ mod \ p$.*

Problème Calculatoire de Diffie-Hellman(CDH) : Etant donnés g^a et g^b dans le sous-groupe H, peut-on reconstituer g^{ab} avec $a, b \in [1, q-1]$. Bien entendu, on peut résoudre se problème si on sait résoudre le problème du logarithme discret.

Problème Décisionnel de Diffie-Hellman(DDH) : Etant donnés $(g^a, g^b) \in H^2$ $(a, b \in [1, q-1])$ et un candidat $z = g^c \in H$ $(c \in [1, q-1])$, décider si $z = g^{ab}$ c'est-à-dire si $c = ab$.

Là aussi si on sait résoudre le logarithme discret, on résout le problème décisionnelle.

La sécurité de ElGamal est basée sur la difficulté du problème de logarithme discret dans un corps fini.

Le chiffrement d'ElGamal est non déterministe, car l'opération de chiffrement dépend du message et d'une valeur aléatoire choisie par l'émetteur. Il y a donc plusieurs textes chiffrés qui peuvent correspondre à un même texte clair.

Génération des clés

Soit p un nombre premier tel que le problème de logarithme discret dans $\mathbb{Z}_p$ soit difficile, et soit $\alpha \in \mathbb{Z}_p^*$ un élément primitif. On sélectionne un entier $s \in [1, p-2[$ et on calcul $t = \alpha^s \ mod \ p$.

La clé publique est (p, α, t) et la clé privée est s.

Chiffrement ElGamal

Si on veut chiffrer le message m pour Oumar on procède comme suit :

1. Obtenir la clé publique (p, α, α^s)
2. Sélectionner un entier $k \in [1, p-1[$
3. Calculer $K = \alpha^k \ mod \ p$
4. Calculer $C = m\alpha^{ks} \ mod \ p$

Déchiffrement ElGamal

Si Oumar reçoit le message (K, C) il utilise sa clé privée pour déchiffrer le message. Pour cela il calcule $m^{'} = C.K^{-s}$

En effet, $m^{'} = C.K^{-s} = m\alpha^{ks}.\alpha^{-ks}\ mod\ p = m$

NB : Le chiffrement ElGamal n'a pas un niveau de securité elevé.

Signature digitale

Parmi les problèmes auxquels s'attaque la cryptographie, on trouve l'authentification de l'origine de données et l'intégrité : lorsque l'on communique avec une personne sur un canal peut sûr, on aimerait bien que le destinateur puisse s'assurer que le message émane bien de l'auteur auquel il est attribué et qu'il n'a pas été altéré pendant le transfert. Pour ce, la signature d'un message a été inventé.

Terminologie

- Soit $\mathcal{M}$ est un ensemble appelé espace de messages clairs
- Soit $\mathcal{S}$ est un ensemble appelé espace des signatures
- Soit $\mathcal{S}_A : \mathcal{M} \to \mathcal{S}$ est une transformation appelée fonction de signature pour une entité A
- Soit $\mathcal{V}_A : \mathcal{M} \times \mathcal{S} \to \{vraie, faux\}$ est une transformation appelée fonction de vérification de la signature de A

Définition 0.0.11. *Les transformations $\mathcal{S}_A$ et $\mathcal{V}_A$ constituent un schéma de signature digitale.*

Procédure

Si une entité A veut créer une signature pour le message $m \in \mathcal{M}$, elle doit effectuer les opérations suivantes :

1. Calculer $s = \mathcal{S}_A(m)$
2. Transmettre la pair (m, s). s est appelé la signature de m.

Vérification de la Procedure

Pour vérifier que la signatire s a été créée par A, une entité O éxecute ce qui suit :

1. Obtenir la fonction de vérication $\mathcal{V}_A$ de A
2. Calculer $v = \mathcal{V}_A(m, s)$
3. Si $v = vraie$ on accepte la signature sinon on rejette la signature.

Remarque 0.0.1. *Les transformations $\mathcal{V}_A$ et $\mathcal{S}_A$ sont typiquement caractérisées par des clés ; c'est à dire il existe une classe des algorithmes de signature et de verification de signature connus publiquement, et chaque algorithme est identifié par une clé. Ainsi l'algorithme de signature $\mathcal{S}_A$ de A est déterminé par une clé secrète k_A et A est le seul à connaître cette clé privée. De la même manière l'algorithme $\mathcal{V}_A$ est déterminé par une clé v_A qui est publique correspondante à la clé privée k_A.*

signature

Il existe plusieurs propriétés que les fonctions de signature et de vérification de signature doivent satisfaire.

- Une signature s de A d'un message m est valide si et seulement si $\mathcal{V}_A(m, s) = vraie$.
- Il calculatoirement difficile (complexité exponentielle) pour toute entité autre que A de trouver pour tout $m \in \mathcal{M}$, un $s \in \mathcal{S}$ tel que $\mathcal{V}_A(m, s) = vraie$.

NB : Cette version de signature ne garantie pas l'intégrité car elle n'utilise pas de fonction de hachage que nous allons voir dans la suite.

Parmis les signatures numériques examinons la signature **RSA** et la signature **ElGamal**

Signature RSA

Le principe est le suivant : Le module N de RSA est le produit de deux grands nombres premiers p et q. On pose $\varphi(N) = (p-1)(q-1)$. Soit un entier e tel que, $0 < e < \varphi(N)$ et $pgcd(e, \varphi(N)) = 1$ et soit d un entier vérifiant : $0 < e < \varphi(N)$ et $ed \equiv 1 \ mod \ \varphi(N)$. La clé (e, N) est la clé publique et la clé (d, N) est la clé privée.

Signature

Pour signer un message $m \in \frac{\mathbb{Z}}{N\mathbb{Z}}$, l'entité A calcule $s = \mathcal{S}_A(m) = m^d \ mod \ N$ et on transmet (m, s) où s est la signature de m.

Vérification de la signature

On utilise la clé publique (e, N) pour vérifier la validité de la signature en calculant

$$s^e \ mod \ N.$$

On doit s'assurer que

$$m = s^e \ mod \ N$$

Signature ElGamal

Génération des clés

- choisir un grand nombre premier p
- choisir α un générateur de $\mathbb{Z}_p^*$
- choisir $s \in [1, p-2]$
- calculer $K = \alpha^s \ mod \ p$

La clé s est la clé secrète qu'on utilise pour la génereration des signatures, et la clé (p, α, K) est la clé publique qu'on utilise pour la vérification des signatures.

Génération de la signature

Pour générer la signature s d'un message m, on procède comme suit :

- on choisit aléatoirement $s \in [1, p-2]$ tel que $pgcd(s, p-1) = 1$
- on calcule $\gamma = \alpha^k \ mod \ p$
- on calcule $\delta = (m - s.\gamma).k^{-1} \ mod \ p-1$

Vérication de la signature

Supposons que nous avons reçu le message m et la signature s. Cette signature est valide si :

- $0 < \gamma < p$
- $K^{\gamma}.\gamma^{\delta} \equiv \alpha^{m} \ mod \ p$

NB : Pour garantir une meilleur securité, on modélise cette signature sur les courbes elliptiques.

Fonctions de Hachage

Une fonction de hachage est une fonction à sens unique sans trappe. Ce type de fonction est très utilisé en cryptographie, principalement dans le but de réduire la taille des données à traiter par la fonction de chiffrement. En effet, la caractéristique principale d'une fonction de hachage est de produire un condensé des données. Une fonction de hachage doit associer un et un seul condensé à un texte en clair(cela signifie que la moindre modification sur le texte clair entraîne une modification sur son condensé).

Propriétés des Fonctions de Hachage

Soit $H : \{0,1\}^{*} \rightarrow \{0,1\}^{n}$ une fonction de hachage.
$x \in \{0,1\}^{*}$ est une préimage de $y \in \{0,1\}^{n}$ si $y = H(x)$

1. **Compression** : la fonction de hachage H prend en entrée une chaîne binaire x de taille quelconque et renvoie une chaîne binaire $H(x)$ de taille fixe n.
2. **Fonction facile à calculer** : étant donnée une entrée $x \in \{0,1\}^{*}$, $H(x)$ doit être facile à calculer (complexité polynomiale).
3. **La fonction H doit resister au calcul du préimage** c'est-à-dire qu'il doit être très difficile de retrouver ou générer une préimage x (complexité exponentielle) telle que $H(x) = y$ pour tout y donné dont l'entrée correspondant est inconnue.
4. **la résistance de la seconde préimage** : étant donné x, il est calculatoirement infaisable de trouver une deuxième préimage $x^{'} \neq x$ telle que $H(x^{'}) = H(x)$.
5. **La resistance à la collision :** il est calculatoirement infaisable de trouver deux entrés x et $x^{'}$ telle que $H(x) = H(x^{'})$.

MAC(Message Authentification Code)

Un code d'authentification de message(**MAC)** est une fonction de hachage paramétrée par une clé secrète k, $H_k()$, qui respecte la facilité de calcul, la propriété de compression et qui est telle que :

pour tout k fixé et inconnu par un adversaire, il est calculatoirement difficile pour cet adversaire de calculer une paire $(x, H_k(x))$ à partir de sa connaissance d'un nombre quelconque des paires $(x_i, H_k(x_i))$ où x est différent de tous les x_i

Sécurité

Soit $H : \{0,1\}^{*} \rightarrow \{0,1\}^{n}$ une fonction de hachage. On dit que H atteint une sécurité idéale si :

- ▷ étant donné x, trouver une deuxième préimage $x^{'} \neq x$ telle que $H(x^{'}) = H(x)$ nécessite 2^n opérations.
- ▷ pour trouver un couple $(x, x^{'})$ tel que $x \neq x^{'}$ et $H(x) = H(x^{'})$ nécessite $2^{\frac{n}{2}}$ opérations.

RESEAUX ARITHMETIQUES, NTRU ET SES VARIANTES

Dans ce chapitre nous allons faire un petit rappel sur les réseaux arithmétiques, ensuite nous présenterons la version originale du cryptosystème NTRU, comme décrit dans([11]) et quelques variantes de NTRU.

Réseaux Arithmétiques

Le cryptosystème NTRU est "probablement basé" sur la difficulté de trouver un court vecteur dans un grand réseau. Il semble donc important de décrire ce qu'est un réseau et dire quelques mots sur les algorithmes de réduction de réseaux. Notre résumé est basé sur ([5]) et ([20]).

Notions de Bases

Définition 0.0.12. *Définissons le produit scalaire de deux vecteurs* $u = (u_1, u_2, ..., u_n)$, $v = (v_1, v_2, ..., v_n) \in \mathbb{R}^n$ *comme*

$$\langle u, v\rangle = \sum_{i=1}^{n} u_i v_i.$$

Définition 0.0.13. *Définissons la norme euclidienne d'un vecteur* $v = (v_1, v_2, ..., v_n) \in \mathbb{R}^n$ *comme*

$$\|v\| = \sqrt{\sum_{i=1}^{n} v_i^2}.$$

Soientt n un entier positif avec $n \leq m$, $(b_1, ..., b_n)$ des vecteurs linéairement indépendants de $\mathbb{R}^m$. Soit B la matrice dont les colonnes sont composées des coordonnées des vecteurs $b_1, ..., b_n$

Définition 0.0.14. *Le réseau engendré par la famille* $(b_1, ..., b_n)$*(ou par* B*) est l'ensemble*

$$L(b_1, ..., b_n) = L(B) = \sum_{i=1}^{n} \mathbb{Z} b_i = \{\sum_{i=1}^{n} \lambda_i b_i | \lambda_i \in \mathbb{Z}\}$$

L'entier n *est la dimension de* L*.*

Il faut noter que $L(B)$ est stable par les opérations élémentaires suivantes sur B : permutation de deux lignes, multiplication d'une ligne par -1, additionner une ligne multiple d'un entier avec une autre ligne. Pour une base matricielle B, considerons la valeur $|det(B)|$. Il est facile de montrer que si B et B' génèrent le même reseau alors $|det(B)| = |det(B')|$. Ainsi nous pouvons définir le déterminant d'un réseau et sa dimension.

matrice dont les colonnes sont composées des coordonnées des vecteurs $b_1, ..., b_n$ et soit $L(B)$ le réseau généré par B. Le déterminant ou le volume de $L(B)$ est

$$detL(B) = |det(B)|.$$

Définition 0.0.16. *Soit $(b_1, ..., b_n)$ un n-uplets de vecteurs linéairement indépendants de $\mathbb{R}^m$ et soit B la matrice dont les colonnes sont composées des coordonnées des vecteurs $b_1, ..., b_n$ et soit $L(B)$ le réseau généré par B. La dimension de $L(B)$ est n. Si $n = m$ on dira que $L(B)$ est de dimension pleine.*

Définition 0.0.17. *(**Parallélépipède fondamental**). Soit $(b_1, ..., b_n)$ des vecteurs linéairement indépendants de $\mathbb{R}^m$. Le* parallélépipède fondamental *des b_i est :*

$$\mathcal{P}(b_1, ..., b_n) = \{\sum_{i=1}^{n} a_i b_i |\ 0 \leq a_i < 1\}.$$

L'ensemble des points $v + \mathcal{P}(B),\ v \in L(B)$ forme une partition de $L(B)$. Nous pouvons montrer facilement que le volume du parallélépipède fondamental est le même pour toutes les bases B qui engendrent le réseau $L = L(B)$ et est égal au déterminant du réseau (voir[5]).

Définition 0.0.18. *(**Orthogonalisation de Gram-Schmidt**)*

Soit $(b_1, ..., b_n)$ une base d'un réseau $L \subset \mathbb{R}^m$. On considère la famille de vecteurs $(b_1^*, ..., b_n^*)$ définit par

$$b_1^* = b_1 \quad b_j^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$$

avec pour $j < i$

$$\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}$$

Alors $(b_1^*, ..., b_n^*)$ est une base orthogonale de $\mathbb{R}^m$.

L'algorithme de réduction du reseau

L'algorithme LLL de réduction est un composant crucial dans un grand nombre d'algorithmes de la théorie des nombres. Il est utilisé pour résoudre un certain nombre de problèmes, par exemple, il a été utilisé pour cryptanalyser des schémas de chiffrement à clé publique basés sur des problèmes utilisant des réseaux.

Définition 0.0.19. *Une base $(b_1, ..., b_n)$ est LLL-réduite si, la base $(b_1^*, ..., b_n^*)$ produite par la méthode d'orthogonalisation de Gram-Schmidt vérifie*

$$|\mu_{i,j}| \leq \tfrac{1}{2},\ \text{pour } 1 \leq j < i \leq n$$

(où $|\mu_{i,j}|$ est la valeur absolue de $\mu_{i,j}$) et

$$\|b_i^*\|^2 \geq (\tfrac{3}{4} - \mu_{i,i-1}^2)\|b_{i-1}^*\|^2 \text{ pour } 1 < i \leq n.$$

avec les propriétés du théorème suivant :

Théorème 0.0.1. *Soit $(b_1, ..., b_n)$ une base LLL-réduite et $(b_1^*, ..., b_n^*)$ la base orthogonale associée par la méthode de Gram-Schmidt. Alors*

1. $\|b_j^*\|^2 \geq 2^{i-j}\|b_i^*\|^2$ *pour* $1 \leq j \leq i \leq n$
2. $det(L) \leq \prod_{i=1}^{n} \|b_i\| \leq 2^{\frac{n(n-1)}{4}} det(L)$
3. $\|b_j\| \geq 2^{\frac{i-1}{2}}\|b_i^*\|$ *pour* $1 \leq j \leq i \leq n$
4. $\|b_1\| \geq 2^{\frac{n-1}{4}}(det(L))^{\frac{1}{n}}$
5. *pour tout vecteur non nul* $v \in L$, $\|b_1\| \leq 2^{\frac{n-1}{2}}\|v\|$

Il faut signaler que la complexité de l'algorithme LLL est polynômiale, ce qui signifie qu'il est très efficace. Les bases que cet algorithme nous fournit sont composées de vecteurs assez courts, ce qui peut apporter une réponse qui satisfait partiellement aux deux problèmes suivants

Problème SVP, Shortest Vector Problem (problème du plus court vecteur) : Etant donné une base B d'un réseau L, trouver un vecteur non nul de L le plus court possible pour la norme euclidienne.

Problème CVP, Closest Vector Problem (problème du plus proche vecteur) : Etant donné une base B d'un réseau L et un vecteur $v \in L$, trouver un vecteur de L le plus proche possible de v pour la norme euclidienne.

NTRU

Le cryptosystème NTRU, qui fait l'objet de ce travail, est basé sur le problème : trouver un vecteur court dans un réseau (**SVP**) ou le vecteur le plus proche d'un vecteur donné (**CVP**).

Le principal avantage du cryptosystème NTRU par rapport aux autres cryptosystèmes à clé publique est sa rapidité dans les processus de chiffrement et de déchiffrement. Il est comparable aux cryptosystèmes à clé secrète. Par exemple, considérons les chiffres tirés du site officiel du Cryptosystème NTRU([28]).

	NTRU 251	RSA 1024	ECC 163
clé publique (bits)	2008	1024	164
clé secrète (bits)	251	1024	163
blocks claires (bits)	160	702	163
blocks chiffrés (bits)	2008	1024	163
vitesse chiffrement(blocks/seconde)	22727	1280	458
vitesse déchiffrement(blocks/seconde)	10869	110	702

Ils fournissent une comparaison entre les vitesses de NTRU et deux autres cryptosystèmes à clé publique populaires, avec des niveaux plus ou moins équivalents pour la sécurité.

Un autre avantage concernant le futur de la cryptographie à clé publique est que le cryptosystème NTRU résiste encore contre les attaques des ordinateurs quantiques contrairement à RSA et ElGamal.

Le cryptosystème NTRU depend de plusieurs sortes de paramètres, en particulier les paramètres N, p *et* q. Soit N un entier premier. Le domaine des opérations de NTRU est l'anneau des polynômes à coefficients entiers $\mathcal{P} = \frac{\mathbb{Z}[X]}{(X^N-1)}$. On considère les deux anneaux quotients :

$$\mathcal{P}_p = \frac{\frac{\mathbb{Z}}{p\mathbb{Z}}[X]}{(X^n-1)}, \quad \mathcal{P}_q = \frac{\frac{\mathbb{Z}}{q\mathbb{Z}}[X]}{(X^n-1)}$$

Supposons que p et q sont premiers entre eux et q est plus grand que p. Supposons aussi que $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r, \mathcal{L}_m$ sont des sous-ensembles de $\mathcal{P}$. $\mathcal{L}_m$ sera l'espace des messages clairs, nous choisissons nos clés dans $\mathcal{L}_f$, et dans $\mathcal{L}_g$ et les polynômes dans $\mathcal{L}_r$ serviront comme des éléments aléatoires dans notre schéma de chiffrement. Ainsi, les éléments f de $\mathcal{P}$ peuvent être représentés sous la forme

$$f = \sum_{i=0}^{N-1} f_i x^i = (f_0, f_1, ..., f_{N-1})$$

et notons par $*$ la multiplication dans $\mathcal{P}$, $\mathcal{P}_p$ et $\mathcal{P}_q$, définie comme suit :

$h = f * g$ avec $h_k = \sum_{i+j \equiv k (mod N)} f_i g_j = \sum_{i=0}^{k} f_i g_{k-i} + \sum_{i=k+1}^{N-1} f_i g_{N+k-i}$

Nous définissons aussi la norme d'un élément $F \in \mathcal{P}$ comme suit

$$\|F\|_\infty = \max_{0 \le i \le n}(F_i) - \min_{0 \le i \le n}(F_i)$$

Génération de clés

Pour la création des clés nous choisissons deux polynômes aléatoires $f \in \mathcal{L}_f$, $g \in \mathcal{L}_g$. Le polynôme f doit être inversible dans $\mathcal{P}_p$ et dans $\mathcal{P}_q$. Soient f_p et f_q les inverses respectives de f dans $\mathcal{P}_p$ et $\mathcal{P}_q$. Nous calculons alors le polynôme

$$h = f_q * g \in \mathcal{P}_q$$

qui est notre clé publique. Notre clé privée est f et on doit aussi garder secrètement f_p pour le déchiffrement.

Chiffrement et Déchiffrement

Supposons que Aissa veut nous envoyer un message $m \in \mathcal{L}_m$. Elle choisit un polynôme $r \in \mathcal{L}_r$ aléatoirement et calcule :

$$e = pr * h + m \in \mathcal{P}_q,$$

Où h est notre clé publique. Si nous recevons e, premièrement nous multiplions e par notre clé privée pour obtenir

$$a = f * e = pr * f * f_q * g + f * m = pr * g + f * m \in \mathcal{P}_q$$

Sous certaines conditions(c'est-à-dire si $\|pr * g + f * m\|_\infty < q$), nous pouvons voir a comme un polynôme de $\mathcal{P}$, le reduire modulo p et le multiplier par f_p qui est l'inverse de f modulo p pour retrouver le message initial modulo p :

$$b = f_p * a = m \in \mathcal{P}_p$$

Succès du Déchiffrement : Nous devons choisir les paramètres de tels sorte que nous soyons capable de déchiffrer : la condition suivante doit être vérifier :

$$\|pr * g + f * m\|_\infty < q.$$

Les coefficients de f, r et g doivent être très petits. Les auteurs de NTRU définissent les ensembles suivants :

$$\mathcal{L}(d_1, d_2) := \{f | f \in \mathcal{P} \text{ a } d_1 \text{ coefficients égals à } 1,\ d_2 \text{ coefficients égals à } -1, \text{ reste } 0\}.$$

Ainsi, ils choisissent les valeurs entières pour d_f, d_g, et d_r et les ensembles

$$\mathcal{L}_f = \mathcal{L}(d_f, d_f - 1), \quad \mathcal{L}_g = \mathcal{L}(d_g, d_g 1), \quad \mathcal{L}_r = \mathcal{L}(d_r, d_r)$$

Il faut noter que $\mathcal{L}_f$ ne doit pas être $\mathcal{L}(d_f, d_f)$, si non les éléments $f \in \mathcal{L}_f$ ne serons certainement pas inversibles : nous voulons trouver $f^{'}$ tel que $f^{'} f = 1$, mais le choix de f implique que $f(1) = 0$ et alors $f^{'} f(1) = 0 = 1$, qui est une contradiction.

Soit

$$pr * g + f * m \in \mathcal{P}$$

L'objectif est de déterminer d_f, d_g, d_r, de sorte que l'inégalité

$$|a_i| < \tfrac{q-1}{2}$$

soit vérifiée pour chaque i, $0 \leq i < N$ avec une probabilité très grande.

Les nombres entiers d_f, d_g, d_r et d_m sont fixés pour chaque version. Actuellement, ces paramètres sont les suivants (voir[11]) :

Sécurité	N	p	q	d_f	d_g	d_r
Moyenne	251	2	128	72	71	72
Haute	347	2	128	64	173	64
Très Haute	503	2	256	420	251	170

Analyse de la sécurité

Attaque par force brute

Un attaquant peut récupérer la clé privée, en essayant de trouver $f \in \mathcal{L}_f$ et de vérifier si les coefficients de $f * h \ mod \ q$ sont petits. Dans la pratique, la taille de $\mathcal{L}_g$ est plus pétite que $|\mathcal{L}_f|$ et, par conséquent, il est logique de tester tous les $g \in \mathcal{L}_g$ et voir si les coefficients de $f * h^{-1} \ mod \ q$ sont petits. De la même manière, il peut essayer de récupérer un message individuel en essayant toutes les $r \in \mathcal{L}_r$ et tester la taille des coefficients de $e - r * h \ mod \ q$

En résumant ce qui est en haut, la sécurité de la clé est donnée par $|\mathcal{L}_g|$ et la sécurité du message par $|\mathcal{L}_r|$. Ce qui donne :

$$|\mathcal{L}_g(d_g, d_g)| = \begin{pmatrix} N \\ d_g \end{pmatrix} \begin{pmatrix} N - d_g \\ d_g \end{pmatrix} = \frac{N!}{(d_g!)^2 (N - 2d_g)!}$$

$$|\mathcal{L}_r(d_r, d_r)| = \begin{pmatrix} N \\ d_r \end{pmatrix} \begin{pmatrix} N - d_r \\ d_r \end{pmatrix} = \frac{N!}{(d_r!)^2(N-2d_r)!}$$

Il existe une autre attaque appelée attaque du milieu qui concerne la clé publique et le message ([16])

.

Attaques basées sur les réseaux

Dans NTRU, la clé publique $h \equiv f_q * g \ (mod\ q)$ vérifie $f * h \equiv g \ (mod\ q)$.
Alors, il existe un polynôme $u \in \mathcal{P} = \frac{\mathbb{Z}[x]}{(x^N-1)}$ tel que

$$f * h - q * u = g$$

On considère alors l'ensemble

$$\mathcal{L} = \{(f, g) \in \mathcal{P}^2 | \exists u \in \mathcal{P},\ f * h - q * u = g\}.$$

Avec les coordonnées de f, g et h, l'égalité ci-dessus prend la forme

$$\begin{pmatrix} f_0 \\ f_1 \\ . \\ . \\ . \\ f_{N-1} \\ \hline -u_1 \\ -u_2 \\ . \\ . \\ . \\ u_{N-1} \end{pmatrix} * \left(\begin{array}{cccccc||cccccc} 1 & 0 & . & . & . & 0 & h_0 & h_1 & . & . & . & h_{N-1} \\ 0 & 1 & . & . & . & 0 & h_{N-1} & h_0 & . & . & . & h_{N-2} \\ 0 & 0 & & & & & . & . & & & & . \\ . & . & & & & & . & . & & & & . \\ . & . & & & & & . & . & & & & . \\ 0 & 0 & . & . & . & 1 & h_1 & h_2 & . & . & . & h_0 \\ \hline 0 & 0 & . & . & . & 0 & q & 0 & . & . & . & 0 \\ 0 & 0 & . & . & . & 0 & 0 & q & . & . & . & 0 \\ . & . & & & & . & . & . & & & & 0 \\ . & . & & & & . & . & . & & & & 0 \\ . & . & & & & . & . & . & & & & 0 \\ 0 & 0 & & . & . & 0 & 0 & 0 & . & . & . & q \end{array}\right) = \begin{pmatrix} f_0 \\ f_1 \\ . \\ . \\ . \\ f_{N-1} \\ \hline g_0 \\ g_1 \\ . \\ . \\ . \\ g_{N-1} \end{pmatrix}$$

Dans la clé publique h, les polynômes f et g ont de petits coefficients et $(f, g) \in \mathcal{L}$. Alors en appliquant l'algorithme LLL au reseau $\mathcal{L}$, on obtient une base réduite dans laquelle les premiers vecteurs sont assez courts, et on peut ainsi déterminer f et g. Si (f, g) est le plus court vecteur du reseau $\mathcal{L}$, cela revient à résoudre le problème SVP. Ceci illustre que la sécurité de NTRU est basée sur ce problème difficile.

Quelques variantes de NTRU

A cause des multiples attaques sur les premières versions de **NTRU**, le cryptosystème **NTRU**, possède plusieurs versions. Ce qui fait l'évolution de **NTRU** vers des versions plus sûres. Ainsi on peut éviter les attaques basées sur la réduction des réseaux en remplaçant l'anneau $\mathbb{Z}$ par un autre anneau.

En 2002, Banks et Shparlinski[1] ont présenté une nouvelle variante de **NTRU**. La différence fondamentale entre **NTRU** et cette nouvelle variante se trouve au niveau de la génération des clés.

Génération des clés

Pour la génération des clés on procède comme suit :

1. Choisir aléatoirement un polynôme $f \in \mathcal{L}_f$
2. Calculer l'inverse f_p de f dans $\mathcal{P}_p$.
3. Choisir aléatoirement un polynôme $G \in \mathcal{P}$
4. Calculer l'inverse G_q de G dans $\mathcal{P}_q$.
5. Choisir aléatoirement un polynôme $g \in \mathcal{L}_g$
6. Calculer $h = (G_q * g)\ mod\ q$ dans $\mathcal{P}_q$
7. Calculer $H = (G_q * f)\ mod\ q$ dans $\mathcal{P}_q$

La clé publique est alors constituée par le couple (h, H) et (f, f_p, G) est la clé privée.

Maintenant si on veut chiffrer un message $m \in \mathcal{L}_m$ on procède comme suit :

Chiffrement

On choisit d'abord aléatoirement un polynôme $\varphi \in \mathcal{L}\varphi$ et on calcule $e = p\varphi * h + H * m\ (mod\ q)$.

On transmet le message chiffré e. Pour déchiffrer e, on effectue les opérations suivantes :

On calcule $a = G * e\ (mod\ q)$ avec des coefficients dans l'intervalle $]-\frac{q}{2}, \frac{q}{2}[$ et on retrouve m en calculant $m = f_p * a\ (mod\ p)$

Cette version resiste aux attaques classiques sur NTRU, mais son temps d'exécution est pratiquement double à cause de la présence de deux polynômes h et H.

MaTRU

Une nouvelle variante de NTRU a été proposée en 2005 par Coglianese et Goi[3]. Dans cette variante l'anneau $\mathbb{Z}$ a été remplacé par l'anneau des matrices $\mathbb{M}$ carrées $k \times k$ à coefficients dans $\mathcal{P} = \frac{\mathbb{Z}[X]}{(X^N-1)}$. En plus de N et k, MaTRU utilise aussi les paramètres $p,\ q \in \mathbb{N}$. Les nombres p et q peuvent être premiers ou non, mais ils doivent être premiers entre eux.

Les paramètres de MaTRU sont composés de quatres entiers (n, k, p, q) et les 5 ensembles de matrices $(\mathcal{L}_f,\ \mathcal{L}_\varphi,\ \mathcal{L}_A,\ \mathcal{L}_w,\ \mathcal{L}_m) \subset \mathbb{M}$

1. $\mathcal{L}_A$ est l'ensemble de toutes les matrices $C \in \mathbb{M}$ telles que $C^0,\ C^1,\ \ldots,\ C^{k-1}$ sont linéairement indépendants modulo q.
2. $\mathcal{L}_f$ et $\mathcal{L}_\varphi$ sont des ensembles de toutes les matrices $D \in \mathbb{M}$ vérifiant

$$\sum_{i=0}^{k-1} c_i C^i,\ C \in \mathcal{L}_A,\ c_0,\ c_1,\ \ldots\ c_{k-1} \in \mathcal{P}$$

3. $\mathcal{L}_m$ est l'ensemble des messages constitués des toutes les matrices dont les termes sont des polynômes ayant des coefficients modulo p.

   $\mathcal{L}_m = \{M \in \mathbb{M} \mid$ les coefficients des polynômes dans M sont entre $\lceil -\frac{p-1}{2} \rceil\ et\ \lceil \frac{p+1}{2} \rceil\}$

Pour créer une paire de clés publique et privée, on procède comme suit :

- Choisir deux matrices $A,\ B \in \mathcal{L}_A$.
- Selectionner aléatoirement k polynômes $\alpha_0, \alpha_1,\ \ldots,\ \alpha_{k-1} \in \mathcal{P}$
- Selectionner aléatoirement k polynômes $\beta_0, \beta_1,\ \ldots,\ \beta_{k-1} \in \mathcal{P}$
- Calculer $f = \sum_{i=0}^{k-1} \alpha_i A^i \in \mathcal{L}_f$
- Calculer $g = \sum_{i=0}^{k-1} \beta_i B^i \in \mathcal{L}_f$
- Calculer F_p tel que $F_p * f = I\ (mod\ p)$ et F_q tel que $F_q * f = I\ (mod\ q)$ où I est la matrice unité.
- Calculer G_p tel que $F_p * g = I\ (mod\ p)$ et G_q tel que $G_q * g = I\ (mod\ q)$ où I est la matrice unité.
- Choisir aléatoiment une matrice $w \in \mathcal{L}_w$
- Calculer $h = F_q * w * G_q\ (mod\ q)$ dans $\mathcal{P}_q$

La clé publique est le triplet $(h,\ A,\ B)$ et la clé privée est $(f,\ g,\ F_p,\ G_p)$

Chiffrement

Pour chiffrer un message $m \in \mathcal{L}_m$, on choisit d'abord aléatoirement k polynômes $\phi_0, \phi_1,\ \ldots,\ \phi_{k-1} \in \mathcal{P}$ et k polynômes $\psi_0, \psi_1,\ \ldots,\ \psi_{k-1} \in \mathcal{P}$ et on calcule successivement $\phi = \sum_{i=0}^{k-1} \phi_i A^i \in \mathcal{L}_\phi$, $\psi = \sum_{i=0}^{k-1} \psi_i B^i \in \mathcal{L}_\psi$ et $e = p\phi * h * \psi + m\ (mod\ q)$

e est le message chiffré que l'on va transmetre.

Déchiffrement

Si on reçoit le message chiffré e, on calcule d'abord $a = f * e * g\ (mod\ q)$ en mettant les coefficients des polynômes dans l'intervalle $[-\frac{q}{2},\ \frac{q}{2}[$. On retrouve le message en calculant $m' = F_p * a * G_p\ (mod\ p)$.

Remarquons que les niveaux de sécurité de MaTRU sont comparables à ceux de NTRU et de plus le chiffrement et le déchiffrement de MaTRU sont $\frac{k}{2}$ fois plus rapides que NTRU. Il faut signaler egalement que le déchiffrement de MaTRU peut rencontrer les mêmes problèmes que NTRU Classique.

CTRU

En 2002 Gaborit, Ohler et Solé ont proposé une nouvelle version de NTRU qu'ils ont appelé CTRU[7]. Dans cette version l'anneau des entiers est remplacé par l'anneau $\mathbb{A} = \mathbb{F}[T]$ des polynômes univariés à coefficients dans un corps fini. Ils évitent ainsi les attaques basées sur l'algorithme LLL ou le théorème des restes chinois. Un outil de cryptanalyse de CTRU est la forme normale de Popov des matrices à coefficients polynomiaux. Noter que CTRU a été cassé par Kouzmenko dans sa thèse unique [18] en 2006 et par Vats [32] en 2008.

La vitesse de chiffrement et de déchiffrement de CTRU est la même que celle de NTRU pour les mêmes valeurs de N.

Le domaine des opérations de CTRU est l'anneau

$$\mathcal{R} = \frac{\mathbb{A}[X]}{(X^N - 1)}$$

Soit F un polynôme de $\mathcal{R}$. Le degré de F qu'on note $deg_T(F)$ est le degré de F en tant que polynôme en T.

$2 \leq m \leq s$.

Soit d_f, d_g et d_φ des entiers vérifiants $d_f = s - m - 1$ et $d_g = d_\varphi = \lfloor \frac{1}{2}(s - m - 1) \rfloor$. Considérons les ensembles

$$\mathcal{L}_m = \{M \in \mathcal{R} | deg_T(M) < S\}$$

$$\mathcal{L}_f = \{f \in \mathcal{R} | deg_T(f) < 1 + d_f\}$$

$$\mathcal{L}_g = \{g \in \mathcal{R} | deg_T(g) < 1 + d_g\}$$

$$\mathcal{L}_f = \{\varphi \in \mathcal{R} | deg_T(\varphi) < 1 + d_\varphi\}$$

, où $deg_T(f)$ est le degré de f en tant que polynôme en T.

CTRU s'utilise de la même façon que NTRU.

Génération des clés

- choisir aléatoirement $f \in \mathcal{L}_f$
- Calculer l'inverse f_Q de f modulo $(X^N - 1, Q)$
- Calculer l'inverse f_P de f modulo $(X^N - 1, P)$
- Choisir aléatoirement $g \in \mathcal{L}_g$
- Calculer $h \equiv g * f_Q$ dans l'anneau quotient $\frac{\mathcal{R}}{(Q)}$.

La clé publique est h et la clé secrète est $(f,\ f_P)$. Pour chiffrer un message $M \in \mathcal{L}_m$, on procède comme suit :

Chiffrement

- Choisir aléatoirement un polynôme $\varphi \in \mathcal{L}_\varphi$
- Calculer $e = P * \varphi * h + M\ (mod\ Q)$ dans l'anneau quotient $\frac{\mathcal{R}}{(Q)}$.

Le message chiffré est e. Pour retrouver le message originale M on procède de la manière suivante :

Déchiffrement

- Calculer $a = e * f$ dans l'anneau quotient $\frac{\mathcal{R}}{(Q)}$
- On calcule $M = a * f_Q$ dans l'anneau quotient $\frac{\mathcal{R}}{(P)}$

Cryptanalyse de NTRU

La sécurité du cryptosystème NTRU([11] est liée à la difficulté de trouver le plus court vecteur (SVP) dans un réseau euclidien de grande dimension. Cette caractéristique est intéressante puisqu'elle s'avère résistante aux attaques basées sur des ordinateurs quantiques, ce qui n'est pas le cas de l'algorithme RSA basé sur la factorisation d'entiers de grande taille.

Plusieurs recherches ont été effectuées afin d'améliorer la sécurité de NTRU, notamment en augmentant la taille des paramètres utilisés. Cela a pour conséquence de complexifier la reduction de base à l'intérieur des réseaux, au détriment de la rapidité de l'algorithme puisque la taille des clés publiques devient plus grande et la durée du chiffrement/déchiffrement croît. Il faut juste signaler que NTRU reste cependant toujours avantageux en termes de performance lorsqu'on le compare à RSA.

Le cryptanalyse de NTRU est basé sur l'algorithme LLL([19] qui permet la réduction de réseau, c'est-à-dire extraire une base presque orthogonale à partir d'une base quelconque. En décomposant la clé publique h sous la forme d'un produit de deux polynômes f et g, on arrive à utiliser l'algorithme LLL pour trouver une base et obtenir le plus petit vecteur d'un réseau euclidien particulier. Cela nous permet

pratique que pour des paramètres de NTRU assez faibles.

Les versions actuelles de NTRU utilisant les paramètres suivants sont considérées sûres.

Sécurité	N	p	q
Modéré	167	3	128
Moyenne	251	3	128
Haute	347	3	128
Très Haute	503	3	256

ARITHMETIQUE SUR LES ENTIERS DUAUX

Ce chapitre est une adaptation française de notre premier article intitulé *Introduction to the arithmetic of the dual integers*, publié dans **Far East Journal of Mathematical Sciences (FJMS) Volume 57, Issue 1, Pages 13 - 39 (October 2011)**.

Dans ce chapitre, nous ferons une étude sur l'arithmétique de l'anneau des entiers duaux $\mathbb{D} = \mathbb{Z} + \epsilon\mathbb{Z}$ qui est un anneau non euclidien.

Nous introduisons les notions des anneaux pseudo-factoriel et pseudo-euclidien. Pour les entiers duaux, en général, la factorisation et le reste de la pseudo-division ne sont pas uniques à cause de l'existence de diviseurs de zero. Cependant nous avons pu proposer un algorithme qui donne toujours le même reste pour la pseudo-division.

Commençons tout d'abord par quelques définitions et propriétés de base des structures algébriques et ensuite nous passerons sur l'arithmétique des entiers duaux.

Définition 0.0.20. *1. Un élément a d'un anneau A est un diviseur de zéro s'il existe $b \neq 0$ telque $ab = 0$ (donc dans notre définition 0 est un diviseur de zero).*

2. *Un anneau principal est un anneau dans lequel tout idéal est principal.*

3. *Un anneau A intègre est Euclidien s'il existe une application $d : A \to \mathbb{N}$ avec les propriétés suivantes :*
   a) $d(a) \geq 0 \quad \forall a \in A$
   b) $\forall a \in A$ *et* $\forall b \in A\backslash\{0\}$ *Nous avons :* $d(a) \leq d(ab)$,
   c) Soit $a, \in A$ *et soit* $t \in A\backslash\{0\}$ *alors il existe un couple unique* $(q, r) \in A^2$ *tel que* $a = bq + r$ *avec* $r = 0$ *ou* $d(r) < d(b)$

Lemme 0.0.1. *Tout anneau Euclidien est principal.*

Définition 0.0.21. *Ici, nous rappelons quelques définitions au sujet de la divisibilité, primalité et irreductibilité dans un anneau.*

1. *Soit A un anneau. Soient a, b des éléments de A. On dira que b divise a avec $b \neq 0$, ou alternativement, que a est divisible par b, s'il existe $c \in A$ tel que $a = bc$. Si b divise a, alors b est appelé un diviseur de a.*

2. *Un élément z d'un anneau A est premier si z est différent de zéro ou unité, et z divise ab implique z divise a ou divise b.*

implique que a est inversible ou b est inversible.

Définition 0.0.22. *Un anneau factoriel A est un anneau dans lequel tout élément $x \in A$ non nul peut s'écrire $x = up_1^{k_1}p_2^{k_2}...p_n^{k_n}$ avec u un élément inversible dans A et p_i irréductible dans A pour $i = 1, 2, ..., n$. Et cette écriture est unique à un inversible près.*

Lemme 0.0.2. *Tout anneau principal est factoriel.*

Proposition 0.0.4. *Soit A un anneau principal, alors les conditions suivantes sur un élément $z \in A$ sont équivalentes :*

1. *z est irréductible ;*
2. *z est premier ;*
3. *zA est un idéal premier non nul ;*
4. *zA est un idéal maximal non nul.*
5. *$\frac{A}{zA}$ est un corps.*

Définition 0.0.23. *Un anneau commutatif est dit nœtherien, s'il satisfait la condition de maximalité pour les idéaux, c'est-à-dire si tout ensemble non vide des idéaux propres admet un élément maximal. Bien sur, cette condition est équivalente à la condition des chaînes ascendantes.*

Proposition 0.0.5. *Un anneau est nœtherien, si et seulement si tout idéal de A est de type fini.*

Proposition 0.0.6. *Rappelons que si un anneau commutatif A est nœtherien, alors :*

- *l'anneau des polynômes $A[x]$ est nœtherien,*
- *tout anneau quotient de A est nœtherien,*
- *tout anneau commutatif S contenant A qui est de type fini comme un A-module est nœtherien.*

Anneau Pseudo-factoriel

Soit A un anneau commutatif non nécessairement intègre. On pose A^* l'ensemble des éléments inversibles dans A et J_A l'ensemble des diviseurs de zero.

Posons $A^\circ = A\backslash(A^* \cup J_A)$ alors A° est un sous-ensemble multiplicatif de A. Un élément $a \in A^\circ$ divise $b \in A^\circ$ si $b = ac$ pour un $c \in A^\circ \cup A^*$. Dans ce cas nous disons aussi que a est un facteur ou un diviseur de b.

Un élément $a \in A^\circ$, est irréductible si toute factorisation $a = a_1a_2$ dans A a la propriété que l'un d'entre eux est inversible. Les éléments de A° qui ne sont pas irréductibles sont dits réductibles.

Un élément non nul $p \in A^\circ$ est dit premier si la condition que p divise un produit $ab \in A^\circ$ implique toujours que p divise a ou p divise b. Il faut noter que "premier" implique "irréductible" mais l'inverse est fausse.

Un anneau A est pseudo-factoriel si tout élément $b \in A^\circ$ peut être factorisé en un nombre fini d'éléments irréductibles : $b = up_1p_2...p_m$, où les p_i sont irréductibles et non nécessairement distincts et $u \in A^*$.

Preuve

Posons S l'ensemble des éléments de $A-(A^*\cup J_A)$ qui ne peuvent pas être factorisés comme ci-dessus et montrons que S est vide. Pour cela raisonnons par l'absurde c'est-à-dire supposons que $S\neq\emptyset$ et posons $\Gamma_S=\{<a>,a\in S\}$. Γ_S est une famille non vide d'idéaux de A, comme A est nœthérien Γ_S admet un élément maximal $<a_0>$ au moins d'où a_0 est réductible(car S ne contient que des éléments qu'on ne peut factoriser alors que tout élément irréductible est déjà sous forme factorisée).

a_0 réductible implique que $a_0=xy$ (x,y non inversibles) alors $a_0\in<x>\Rightarrow<a_0>\subset<x>$.

Est ce que $x\in<a_0>$?

$x\in<a_0>\Rightarrow x=\alpha a_0\Rightarrow a_0=\alpha a_0 y\Rightarrow a_0(1-\alpha y)=0$, a_0 non diviseur de zéro, alors $1-\alpha y=0$ donc y est inversible, ce qui est absurde. D'où x n'appartient pas à $<a_0>$ ceci équivaut à dire que $<a_0>$ est strictemnet contenu dans $<x>$ alors $<x>$ n'appartient pas à Γ_S ce qui équivaut à x n'appartient pas à S alors x peut être factorisé en éléments irréductibles.

$$x=u.p_1^{\alpha_1}...p_m^{\alpha_m}$$

où les p_i sont irréductibles et deux à deux distincts.

De même on montre que $y=v.q_1^{\beta_1}...q_l^{\beta_l}$

$a_0=xy=uv.p_1^{\alpha_1}...p_m^{\alpha_m}.q_1^{\beta_1}...q_l^{\beta_l}=\omega r_1^{\gamma_1}...r_m^{\gamma_m}...r_{m+l}^{\gamma_m+l}$

a_0 peut être factorisé ce qui est absurde car $a_0\in S$ d'où $S=\emptyset$ alors tout élément non inversible qui n'est pas un diviseur de zéro est factorisable.

□

Arithmétique sur les entiers duaux

Pseudo-factorisation

Définition 0.0.24. *Soit $\mathbb{D}=\frac{\mathbb{Z}[x]}{(x^2)}=\mathbb{Z}+\mathbb{Z}x$, avec $x^2=0$. Nous notons aussi $\mathbb{D}=\mathbb{Z}[\epsilon]$ avec $\epsilon^2=0$. L'anneau $\mathbb{D}$ est appelé l'anneau des entiers duaux. Si $z=a+\epsilon b\in\mathbb{D}$ alors a est appelée la partie réelle de z et est notée $Re(z)$; b est appelée la partie imaginaire de z et est notée $Im(z)$.*

Lemme 0.0.3.

1. *Un élément $z=a_1+a_2\epsilon\in\mathbb{D}$ est inversible si et seulement si la partie réelle a_1 de z est inversible c'est-à-dire $a_1=\pm1$. Ainsi les éléments inversibles sont de la forme $\pm1+b\epsilon$ avec $b\in\mathbb{Z}$. De plus l'inverse de $(\pm1+b\epsilon)$ est $(\pm1-b\epsilon)$.*
2. *Un entier dual $z=a_1+a_2\epsilon$ est un diviseur de zéro si et seulement si la partie réelle a_1 de z est égale à zéro. Donc les diviseurs de zéro sont de la forme $b\epsilon$ avec $b\in\mathbb{Z}$.*

Preuve Simple. □

Lemme 0.0.4. *Tout élément $z=n+\epsilon m\in\mathbb{D}$ avec $Re(z)=n\neq0$ peut être écrit de la forme $n+\epsilon m=(|n|+\epsilon r)(\pm1+\epsilon q)$ où $r=\pm m\ mod\ |n|$ et $q=\lfloor\frac{\pm m}{|n|}\rfloor$.*

Preuve - Supposons $n>0$ et posons $r=m\ mod\ |n|$ et $q=\lfloor\frac{m}{|n|}\rfloor$ alors $(n+\epsilon r)(1+\epsilon q)=n+\epsilon(nq+r)=n+\epsilon m$ comme souhaité.

$n + \epsilon m$ comme souhaité.

□

Lemme 0.0.5. *Un élément $z = n + \epsilon m = (|n| + \epsilon r)(\pm 1 + \epsilon q) \in \mathbb{D}$ où $r = \pm m \ mod \ |n|$ et $q = \lfloor \frac{\pm m}{|n|} \rfloor$ est irreductible si et seulement si $n = p$ ou $n = p^l (l > 1$ et $pgdc(r, p) = 1)$ où p est un entier premier.*

Preuve

Remarquons premièrement que par le lemme 0.0.4, il suffira d'étudier l'irréductibilité des éléments de la forme $z = n + \epsilon r$ avec $r < |n|$.

Soit $z = n + \epsilon r \in \mathbb{D}$ et écrire $z = n + \epsilon r = (\alpha + \epsilon\beta)(\alpha' + \epsilon\beta') = \alpha\alpha' + (\alpha\beta' + \beta\alpha')\epsilon$ alors :

$$\begin{cases} \alpha\alpha' = n & (1) \\ \alpha\beta' + \beta\alpha' = r & (2) \end{cases}$$

Nous allons étudier quatre cas :

<u>1^{er} *Cas*</u> : Supposons que n est premier, alors à partir des équations ci-dessus il est facile de voir que $n + \epsilon m$ est irréductible.

<u>2^{eme} *Cas*</u> : Supposons $n = p^l$ et $pgdc(p, r) = 1$ (où p est un élément premier et $l > 1$ un entier), alors

$$\begin{cases} \alpha\alpha' = p^l (l > 1) & (1) \\ \alpha\beta' + \beta\alpha' = r & (2) \end{cases}$$

Posons $\alpha = p^k$ et $\alpha' = p^{k'}$, avec $k + k' = l$ et $k < k'$.

Alors l'équation

$$(2) \Rightarrow p^k\beta' + \beta p^{k'} = r \Rightarrow p^k(\beta' + \beta p^{k'-k}) = r$$

Si $pgdc(r, p) = 1$ alors $p^k = 1$ qui implique que $k = 0$ ainsi $\alpha = 1 \Rightarrow (\alpha + \beta\epsilon)$ est inversible.

par conséquent, $n + r\epsilon$ est irréductible.

<u>3^{eme} *cas*</u> : Supposons $n = p^l$ et p divise r, alors $n + r\epsilon = gcd(n, r)(n' + r'\epsilon)$ mais $pgdc(n, r) \neq \pm 1$ et $n' \neq \pm 1$ parce que p divise r et n et $r < |n|$. Par conséquent $n + r\epsilon$ est réductible.

<u>4^{eme} *cas*</u> : Supposons $n = n' \times n''$, avec $n', n" > 2$ et $pgdc(n', n'') = 1$ alors à partir de l'algorithme Euclide etendu il existe u et v tel que $un' + vn'' = 1$. Ainsi nous avons $r = run' + rvn''$ d'où

$$(n' + \epsilon rv)(n'' + \epsilon ru) = n'n'' + \epsilon(n'ru + n''rv) = n'n'' + r\epsilon = n + r\epsilon$$

ce qui prouve que $n + r\epsilon$ est réductible car $n' + \epsilon rv$ et $n" + \epsilon ru$ sont non inversibles.

□

Proposition 0.0.8. *Il n'existe pas d'éléments premiers dans l'anneau des entiers duaux $\mathbb{D}$.*

Preuve

Il suffit de prouver qu'un entier dual irréductible n'est pas premier. Du lemme ci-dessus, les entiers duaux irréductibles sont de la forme $z = n+\epsilon m = (|n|+\epsilon r)(\pm 1+\epsilon q) \in \mathbb{D}$ où $r = \pm m \ mod \ |n|$ et $q = \lfloor \frac{\pm m}{|n|} \rfloor$ et $n = p$ ou ($n = p^l$, $l > 1$ et $pgdc(r, p) = 1$) où p est un entier premier.

$$\big(p+(1+\frac{p+1}{2})\epsilon\big)\big(p+(-1+\frac{p-1}{2})\epsilon\big)=(p^2+p^2\epsilon)=p^2(1+\epsilon),$$

d'où p (p est irréductible) divise

$$\big(p+(1+\frac{p+1}{2})\epsilon\big)\big(p+(-1+\frac{p-1}{2})\epsilon\big)$$

mais il ne divise pas $p+(1+\frac{p-1}{2})\epsilon$ ni $p+(-1+\frac{p+1}{2})\epsilon$. Par conséquent p n'est pas un entier dual premier comme désiré.

2. Soit $(p^l+\epsilon r)$ un entier dual irréductible avec $l>1$ et $gcd(r,p)=1$, prouvons que $(p^l+\epsilon r)$ n'est pas un entier dual premier.
   - Supposons que r est impair. Nous avons

$$\big(p^l+(1+\frac{r+1}{2})\epsilon\big)\big(p^l+(-1+\frac{r-1}{2})\epsilon\big)=(p^{2l}+p^lr\epsilon)=p^l(p^l+r\epsilon),$$

     d'où $(p^l+r\epsilon)$(qui est irréductible) divise

$$\big(p^l+(1+\frac{r+1}{2})\epsilon\big)(p^l+(-1+\frac{r-1}{2})\epsilon\big)$$

     mais il ne divise pas $p^l+(1+\frac{r-1}{2})\epsilon$ ni $p^l+(-1+\frac{r+1}{2})\epsilon$.
   - Supposons que r est pair : Nous avons

$$\big(p^l+(1+\frac{r}{2})\epsilon\big)\big(p^l+(-1+\frac{r}{2})\epsilon\big)=(p^{2l}+p^\alpha r\epsilon)=p^l(p^l+r\epsilon),$$

     ainsi $(p^l+r\epsilon)$ (qui est irréductible) divise

$$\big(p^l+(1+\frac{r}{2})\epsilon\big)\big(p^l+(-1+\frac{r}{2})\epsilon\big)$$

     mais il ne divise pas $p^l+(1+\frac{r}{2})\epsilon$ ni $p^l+(-1+\frac{r}{2})\epsilon$.

□

Remarque :

Il est facile de voir que l'ensemble des diviseurs de zéro $J_D=\epsilon\mathbb{Z}=\epsilon\mathbb{D}$ est un idéal principal et premier. Donc $\mathbb{D}/J_D$ est un anneau quotient intègre :

$\mathbb{D}/J_D=\frac{\mathbb{Z}+\epsilon\mathbb{Z}}{\epsilon\mathbb{Z}}\cong\mathbb{Z}$

Nous voyons que $\epsilon\mathbb{Z}=\epsilon\mathbb{D}$ est un idéal premier mais selon notre définition ϵ n'est pas un entier dual premier parce que il est un diviseur de zéro.

□

Théorème 0.0.2. *L'anneau des entiers duaux $\mathbb{D}$ est pseudo-factoriel.*

Preuve

En appliquant les propriétés ci-dessus sur un anneau nœtherien nous avons : $\mathbb{Z}$ est pseudo-factoriel alors $\mathbb{Z}[x]$ est pseudo-factoriel ainsi $\frac{\mathbb{Z}[x]}{(p(x))}$ est aussi pseudo-factoriel. En particulier, l'anneau des entiers duaux $\frac{\mathbb{Z}[x]}{(x^2)}=\mathbb{Z}[\epsilon]=\mathbb{D}$ est pseudo-factoriel.

Algorithme de pseudo-factorisation

Soit $z=n+m\epsilon\in\mathbb{D}$ avec $n\neq 0$.

2. posons $n' = n/pgdc(n, r_0)$ et $r'_0 = r_0/pgcd(n, r_0)$ (alors $pgdc(n', r'_0) = 1$ et $r'_0 < |n|$).
3. $n' \longleftarrow fact(n') = p_1^{\alpha_1}...p_k^{\alpha_k}$ (c'est la factorisation des entiers)
4. si $k = 1$ alors enregistrons $z_0 = (n' + r'_0\epsilon)$
5. si $k > 1$ alors $pgdc(p_1^{\alpha_1}, (p_2^{\alpha_2}...p_k^{\alpha_k})) = 1$, et il existe (u_1, v_2) tel que

$$u_1 p_1^{\alpha_1} + v_1 p_2^{\alpha_2}...p_k^{\alpha_k} = 1$$

en multipliant cette expression par r'_0 nous obtenons :

$$r'_0 u_1 p_1^{\alpha_1} + r'_0 v_1 p_2^{\alpha_2}...p_k^{\alpha_k} = r'_0$$

$$n' + r'_0\epsilon = (p_1^{\alpha_1} + r'_0 v_1\epsilon)(p_2^{\alpha_2}...p_k^{\alpha_k} + r'_0 u_1\epsilon)$$

$$p_1^{\alpha_1} + r'_0 v_1\epsilon = (p_1^{\alpha_1} + r_1\epsilon)(1 + q_1\epsilon)$$

$$n \longleftarrow n'/p_1^{\alpha_1} = p_2^{\alpha_2}...p_k^{\alpha_k}$$

$$r_0 \longleftarrow r'_0 u_1$$

répétons 1) 2) 3) 4) 5)

6. retournons $z = fact(pgcd(n, r_0)(\prod(p_j^{\alpha_j} + r_j\epsilon)(1 + q_j\epsilon)(1 + q_0\epsilon)$

Exemples

Pseudo-factorisation : $n' = 35 = 5 \times 7$.

$5u_1 + 7v_1 = 1$, (3,-2) est une solution de cette équation parce que $5 \times 3 + 7(-2) = 1$

$-6 = 5 \times 3(-6) + 7(-2)(-6)$

$z = 2^2(35-6\epsilon) = 2^2(5+12\epsilon)(7-18\epsilon) = 2^2(5+2\epsilon)(1+2\epsilon)(7+3\epsilon)(1-3\epsilon) = 2^2(5+2\epsilon)(7+3\epsilon)(1+2\epsilon)(1-3\epsilon)$

Non-unicité de la factorisation

$z = (5 + 2\epsilon)(5 + 3\epsilon) = (5^2 + 5^2\epsilon) = 5^2(1 + \epsilon)$.

Algorithme de Pseudo-division

Un anneau A est pseudo-euclidien s'il existe une application $\varphi : A \to \mathbb{N}$ qui vérifie les propriétés suivantes :

1) $\varphi(z) \geq 0 \quad \forall z \in A$

2) $\forall z \in A$ et pour tout $t \in A - J_A$ où J_A est l'ensemble des diviseurs de zéro nous avons :

$$\varphi(z) \leq \varphi(zt)$$

,

tel que $z = tq + r$ avec $r = 0$ ou $\varphi(r) < \varphi(t)$

L'application φ définie ci-dessus, est appelée pseudo-norme.

Théorème 0.0.3. *$(\mathbb{D}, \varphi)$ est un anneau pseudo-euclidien. En effet, si $z, t \in \mathbb{D}$ et t n'est pas diviseur de zéro, alors on peut trouver $(q, r) \in \mathbb{D}^2$ tel que $z = tq + r$ avec $r = 0$ ou $\varphi(r) < \frac{\varphi(t)}{4}$ (remarquons que (q, r) n'est pas nécessairement unique).*

Preuve

Soit $\varphi : \mathbb{D} \to \mathbb{N}$ par $\varphi(x) = x\overline{x} = Re(x)^2 = a^2$ où $x = a + b\epsilon$ et $\overline{x} = a - \epsilon b$

La condition 1) est satisfaite parce que φ est toujours positive.

Soit $t \in \mathbb{D}\backslash J_{\mathbb{D}}$ alors $\varphi(t) \geq 1$ et nous avons $\varphi(z) \leq \varphi(z)\varphi(t) = \varphi(zt)$. Ceci montre que la seconde condition est satisfaite.

Maintenant montrons que pour tout $z, t \in \mathbb{D}$ avec $t \in \mathbb{D}\backslash J_{\mathbb{D}}$, il existe $(q, r) \in \mathbb{D}$ tel que $z = tq + r$ avec $r = 0$ ou $\varphi(r) < \frac{\varphi(t)}{4}$.

Posons $z = a + b\epsilon$ et $t = a' + b'\epsilon$ et définissons $n = t\bar{t} = a'^2 = \varphi(t) = \varphi(\bar{t})$.

Calculons $z\bar{t} = aa' + (a'b - ab')\epsilon = a_1 + a_2\epsilon$ avec $a_1 = aa'$ et $a_2 = a'b - ab'$

Par application de la division euclidienne, il existe un unique couple (q_1, r_1) tel que $a_1 = nq_1 + r_1$ et $a_2 = nq_2 + r_2$ avec $|r_1| < \frac{|n|}{2}$ et $|r_2| < \frac{|n|}{2}$. Les deux dernières conditions sont équivalentes à $\varphi(r_1) < \frac{\varphi(n)}{4}$ et $\varphi(r_2) < \frac{\varphi(n)}{4}$.

Posons $q = q_1 + q_2\epsilon$ et $r = r_1 + r_2\epsilon$ alors nous avons $nq + r = (nq_1 + r_1) + (nq_2 + r_2)\epsilon = a_1 + a_2\epsilon$.

Ainsi, nous déduisons que $z\bar{t} = nq + r \quad$ avec $\varphi(r) = \varphi(r_1) < \frac{\varphi(n)}{4}$.

Maintenant $z\bar{t} = nq + r$ implique que $z\bar{t} - \bar{t}tq = r$ ainsi $\bar{t}(z - tq) = r$.

En utilisant φ, nous avons $\varphi(\bar{t})\varphi(z - tq) = \varphi(r) \Rightarrow n\varphi(z - tq) = \varphi(r) = \varphi(r_1) < \frac{\varphi(n)}{4} = \frac{n^2}{4}$

D'où nous avons $\varphi(z - tq) < \frac{n}{4} = \frac{\varphi(t)}{4}$

Posons $r' = z - tq \Rightarrow \varphi(r') < \frac{\varphi(t)}{4}$ par application des formules ci-dessus.

Par conséquent, $q = q_1 + q_2\epsilon$ et $r' = z - tq$ implique que $z = tq + r'$ avec $\varphi(r') < \frac{\varphi(t)}{4}$.

□

Algorithme de pseudo-division

<u>En entrée</u> : $z \in \mathbb{D}$ et $t \in \mathbb{D}\backslash J_{\mathbb{D}}$;

<u>En sortie</u> : $(q, \rho) \in \mathbb{D}$ tel que $z = qt + r$ avec $\varphi(\rho) < \frac{\varphi(t)}{4}$.

1. Posons $a_1 \leftarrow Re(z\bar{t})$, $a_2 \leftarrow Im(z\bar{t})$ et $n = t\bar{t}$
2. Appliquons l'algorithme de division pour trouver (q_i, r_i) avec $i = 1, 2$ tel que $a_i = nq_i + r_i$;
3. sortie $q = q_1 + q_2\epsilon$ et $\rho = z - tq$.

Exemple

Soit $z = 122 - 29\epsilon$ et $t = 3 - 50\epsilon$ alors notre algorithme donne $q = 40 - 673\epsilon$ et

$r = z - tq = 2 - 4048\epsilon$

Implémentation de l'algorithme de Pseudo-division en C++

La fonction suivante implemente l'algorithme de Pseudo-division. Un objet de la classe *duaux* représente un entier dual $a + b\epsilon$ où a et b sont des éléments de la classe ZZ(implémenté dans la librairie NTL([31])).

ZZn ;
$n = (d1 * d1.conj()).getX()$;
duaux $y1 = d2 * d1.conj()$;
$ZZ\ a\ =\ y1.getX()$;
$ZZ\ b\ =\ y1.getY()$;
$ZZ\ u,\ u1$;
$u1 = a$;
$u = to\_ZZ(0)$;
$DivRem(u, u1, a, n)$; $//a = un + u1$
if$(2 * u1 > n)$ {
$u1- = n;$
$u++$;
}
$ZZ\ v,\ v1$;
$v1 = b;$
$v = to_Z Z(0)$;
$DivRem(v, v1, b, n)$; $//b = vn + v1$
if$(2 * v1 > n)$ {
$v1- = n;$
$v++;$
}
$q = duaux(u, v);$
$r = d2 - q * d1;$
}

Calcul modulo

Rappelons que si $a \in \mathbb{N}\backslash\{0, 1\}$ et $z_1, z_2 \in \mathbb{Z}$, nous disons que z_1 et z_2 sont équivalents "modulo a" (noté $z_1 \equiv z_2\ mod\ a$) s'il existe $\alpha \in \mathbb{Z}$ tel que $z_1 - z_2 = \alpha a$.

Définition 0.0.25. *Soit $t \in \mathbb{D}\backslash J_{\mathbb{D}}$ et $z_1, z_2 \in \mathbb{D}$, nous disons que z_1 et z_2 sont équivalents " $\overleftrightarrow{modulo}\ t$" (noté $z_1 \equiv z_2\ \overleftrightarrow{mod}\ t$) s'il existe $\alpha \in \mathbb{D}$ tel que $z_1 - z_2 = \alpha t$.*

Noter que le reste de la pseudo-division n'est pas unique, mais notre algorithme donne toujours le même reste et il est plus petit possible par rapport à notre pseudo-norme. Nous allons démontrer ces deux propriétés dans les deux théorèmes suivants.

Théorème 0.0.4. *L'algorithme pseudo-division décrit dans la preuve du théorème 3.2.2 donne des représentants uniques $\overleftrightarrow{mod}\ t$. Plus précisément, si z_1, z_2, t sont trois entiers duaux tels que t n'est pas diviseur de zéro et*

$$z_1 \equiv z_2 \overleftrightarrow{mod}\ t,$$

Alors l'algorithme nous donne deux restes r'_1 et r'_2 tels que $r'_1 = r'_2$.

Soient $z_1, z_2 \in \mathbb{D}$, supposons que t est non diviseur de zéro et que $z_1 \equiv z_2 \overset{\longleftrightarrow}{mod}\ t$. L'algorithme de division nous donne $q_1, r_1, q_2, r_2 \in \mathbb{D}$ tel que $z_1 = q_1 t + r_1$ et $z_2 = q_2 t + r_2$.

Prouvons que $r_1 = r_2$.

<u>1^{er} *cas :*</u> Ici, nous étudions le cas particulier où $t = n$ $(n \in \mathbb{N}^*)$

Soit $z_1 = a_1 + b_1\epsilon$, l'agorithme nous donne u_1, u_2, v_1 et v_2 tel que $a_1 = u_1 n + u_2$ et $b_1 = v_1 n + v_2$ avec $-\frac{1}{2}n < u_2, v_2 < \frac{1}{2}n$

En substituant a_1 et b_1 nous avons : $z_1 = u_1 n + u_2 + (v_1 n + v_2)\epsilon = (u_1 + \epsilon v_1)n + u_2 + \epsilon v_2$

l'algorithme donne $q_1 = u_1 + \epsilon v_1$ et $r_1 = u_2 + \epsilon v_2$.

Il est clair que $\varphi(r_1) = u_2^2 < \frac{1}{4}\varphi(n)$. Ainsi $z_1 = q_1 n + r_1$ avec $\varphi(r_1) < \frac{1}{4}\varphi(n)$.

Puisque $z_2 \equiv z_1 \overset{\longleftrightarrow}{mod}\ n$ alors il existe $\alpha \in \mathbb{D}$ tel que $z_2 = z_1 + \alpha n$ donc $z_2 = (\alpha + q_1)n + r_1$. Posons $T = \alpha + q_1$. Nous avons $z_2 = (Re(T) + Im(T)\epsilon)n + u_2 + \epsilon v_2$

$z_2 = a_2 + \epsilon b_2$ avec $a_2 = Re(T)n + u_2$ et $b_2 = Im(T)n + v_2$ et en utilisant l'unicité de l'algorithme de division dans $\mathbb{Z}$ nous trouvons (u_1', u_2') et (v_1', v_2') tel que $a_2 = u_1' n + u_2'$ et $b_2 = v_1' n + v_2'$ et donc par identification nous avons $u_1' = Re(T)$, $u_2' = u_2$, $v_1' = Im(T)$, $v_2' = v_2$

Ainsi, le reste pour z_2 est $r_2 = u_2' + \epsilon v_2' = u_2 + \epsilon v_2 = r_1$, ce qui prouve le théorème dans ce cas.

<u>2^{eme} *cas :*</u> Maintenant supposons que $t \in \mathbb{D}\backslash J_{\mathbb{D}}$ est arbitraire.

Posons $n = t\bar{t}$, puisque $z_1 \equiv z_2 \overset{\longleftrightarrow}{mod}\ t$, nous obtenons que $z_1\bar{t} \equiv z_2\bar{t} \overset{\longleftrightarrow}{mod}\ n$ et la première partie de la preuve nous donne $z_1\bar{t} = q_1 n + r_1$ et $z_2\bar{t} = q_2 n + r_1$.

Des égalités précédentes(rappelons que $n = t\bar{t}$), nous obtenons $(z_1 - q_1 t)\bar{t} = (z_2 - q_1 t)\bar{t}$ parce que $r_1 = r_2$.

Comme l'entier dual t n'est pas diviseur de zéro, alors nous avons $z_1 - q_1 t = z_2 - q_2 t$ ainsi les restes que nous recevons de l'algorithme sont égaux$(r_1' = r_2')$ ce qui prouve le théorème.

□

Dans le théorème suivant, nous allons démontrer que notre algorithme renvoie le plus petit reste possible par rapport à notre pseudo-norme.

Théorème 0.0.5. *Soit y, t deux entiers duaux. Supposons que $t \in \mathbb{D}\backslash J_{\mathbb{D}}$. Si $\varphi(y) < \frac{1}{4}\varphi(t)$, alors tous les entiers duaux égaux à $y \overset{\longleftrightarrow}{mod}\ t$ sont réduits à y en utiliant l'algorithme décrit précédemment.*

Preuve. En utilisant le dernier théorème, il suffit de prouver que l'algorithme réduit y lui-même.

<u>1^{er} cas :</u> Supposons premièrement que $t = n$ est un entier et $y = a + b\epsilon$ avec $\varphi(y) < \frac{1}{4}\varphi(t)$ ainsi $a^2 < \frac{1}{4}n^2 \Rightarrow |a| < \frac{1}{2}n$.

Ainsi, avec l'algorithme de division nous obtenons u, u_0, v, v_0 tels que $a = un + u_0$ et $b = vn + v_0$ avec $u = v = 0$, $u_0 = a$, $v_0 = b$. La sortie est $r = u_0 + v_0\epsilon = y$ comme nous le souhaitons.

<u>2^{eme} cas</u> : Maintenant, supposons que t est un entier dual arbitraire. L'algorithme montre pour $n = t\bar{t}$. Nous avons $\varphi(y\bar{t}) = \varphi(y)\varphi(\bar{t}) < \frac{1}{4}\varphi(t)\varphi(\bar{t}) < \frac{n^2}{4}$.

De manière analogue à la première partie de cette preuve, la sortie de l'algorithme (appliqué à $y\bar{t}$ et $t\bar{t}$) est $q = 0, r = 0$. Ainsi, à la fin de l'algorithme nous avons $r_0 = y - tq = y$, ce qui achève la démonstration.

□

Proposition 0.0.9. *Soit y, t deux entiers duaux avec $t \in \mathbb{D}\backslash J_{\mathbb{D}}$. Si l'algorithme de pseudo-division renvoie (q, r) tel que $y = tq + r$ avec $r = 0$ ou $\varphi(r) < \frac{1}{4}\varphi(t)$ alors pour chaque entier m, les entiers duaux $q' = q + m\epsilon$ et $r' = r - t_1 m\epsilon$ (où $t_1 = Re(t)$) vérifient $y = tq' + r'$ avec $\varphi(r') = \varphi(r) < \frac{1}{4}\varphi(t)$*

Ecrire $y = q_1 + y_2\epsilon$, $t = t_1 + t_2\epsilon$, puisque $t \in \mathbb{D}\backslash J_{\mathbb{D}}$ alors $|t_1| > 1$. Soit (q, r) le couple renvoyé par l'algorithme de pseudo-division et écrire $q = q_1 + q_2\epsilon$ et $r = r_1 + r_2\epsilon$, alors nous avons :

$$y = tq + r \text{ avec } \varphi(r) < \frac{\varphi(t)}{4} \Rightarrow y_1 + y_2\epsilon = (t_1q_1 + r_1) + (t_1q_2 + t_2q_1 + r_2)\epsilon \quad (\alpha)$$

Soit (q', r') le couple tel que $y = tq' + r'$ avec $\varphi(r') < \frac{\varphi(t)}{4}$ et écrire $q' = q'_1 + q'_2\epsilon$ et $r' = r'_1 + r'_2\epsilon$ alors :

$$y_1 + y_2\epsilon = (t_1q'_1 + r'_1) + (t_1q'_2 + t_2q'_1 + r'_2)\epsilon \quad (\beta).$$

Par identification nous avons $t_1q_1 + r_1 = t_1q'_1 + r'_1$ mais $\varphi(r) < \frac{\varphi(t)}{4} \Leftrightarrow \varphi(r_1) < \frac{\varphi(t_1)}{4} \Leftrightarrow |r_1| < \frac{|t_1|}{2}$, de la même façon nous avons aussi $\varphi(r') < \frac{\varphi(t)}{4} \Leftrightarrow \varphi(r'_1) < \frac{\varphi(t_1)}{4} \Leftrightarrow |r'_1| < \frac{|t_1|}{2}$. Mais, $t_1q_1 + r_1 = t_1q'_1 + r'_1 \Rightarrow |r_1 - r'_1| = |t_1||q'_1 - q_1|$ mais $|r_1| < \frac{|t_1|}{2}$ et $|r'_1| < \frac{|t_1|}{2}$ donc nécessaire $r_1 = r'_1$ et $q_1 = q'_1$.

Ainsi (α) et (β) donnent $t_1q_2 + t_2q_1 + r_2 = t_1q'_2 + t_2q_1 + r'_2 \Rightarrow t_1(q_2 - q'_2) = r'_2 - r_2$. Maintenant, posons $q'_2 = q_2 + m$ alors $r'_2 = r_2 - t_1m$. Par conséquent $q' = q + m\epsilon$ et $r' = r - t_1m\epsilon$ comme attendu. Plus précisement il est facile de voir que $y = tq' + r'$ avec $\varphi(r') = \varphi(r)$.

□

Sur l'inversibilité dans $\frac{\mathbb{D}}{(z\mathbb{D})}$

Remarque : Soient $a, b \in \mathbb{D}$, nous disons que a et b sont co-premiers (noté $a \wedge b = 1$) s'ils existent $a', b' \in \mathbb{D}$ tel que $aa' + bb' = 1$.

Soit $z = z_1 + \epsilon z_2$ un entier dual non inversible.

Soit $a = a_1 + \epsilon a_2$ alors a est inversible dans $\frac{\mathbb{D}}{z\mathbb{D}}$ si et seulement si $\exists b \in \mathbb{D}$ tel que $\overline{a}\overline{b} = \overline{1} \Leftrightarrow \exists b = b_1 + b_2\epsilon, k = k_1 + k_2\epsilon \in \mathbb{D} : ab - 1 = kz$

$\Leftrightarrow$

$$\begin{cases} a_1b_1 - 1 = k_1z_1 & (\alpha) \\ a_1b_2 + a_2b_1 = k_1z_2 + k_2z_1 & (\beta) \end{cases}$$

$\Leftrightarrow$

$$\begin{cases} a_1b_1 + (-k_1)z_1 = 1 & (\alpha) \\ a_1b_2 + a_2b_1 = k_1z_2 + k_2z_1 & (\beta) \end{cases}$$

Maintenant, supposons que a est inversible, $(\alpha) \Leftrightarrow \overline{a_1}\overline{b_1} = \overline{1}$ dans $\frac{\mathbb{Z}}{z_1\mathbb{Z}}$

. Nous résolvons (α) par l'algorithme Euclide Etendu dans $\mathbb{Z}$ et trouvons le couple $(b_1, -k_1)$ tel que $a_1b_1 + (-k_1)z_1 = 1$ (α).

. Maintenant en utilisant ce résultat dans (β) on a : $(\beta) \Leftrightarrow a_1b_2 + (-k_2)z_1 = k_1z_2 + (-b_1)a_2$; $k_1z_2 + (-b_1)a_2$ est déjà connu et posons $w = k_1z_2 + (-b_1)a_2$.

En multipliant (α) par w on a $a_1(b_1w) + (-k_1w)z_1 = w$.

Nous pouvons choisir $b_2 = b_1w$ et $k_2 = k_1w$ comme une solution de (β).

Ainsi $b = b_1 + \epsilon b_2 = b_1 + b_1w\epsilon = b_1(1 + w\epsilon)$ et nous concluons que l'inverse de a dans $\frac{\mathbb{D}}{z\mathbb{D}}$ est $b_1(1 + w\epsilon)$ avec $b_1 = a_1^{-1}$ dans $\frac{\mathbb{Z}}{z_1\mathbb{Z}}$.

□

Cette remarque intéressante conduit au lemme suivant.

lentes :

1) a est inversible dans $\frac{\mathbb{D}}{z\mathbb{D}}$,

2) $a \wedge z = 1$ (a est co-premier avec z dans $\mathbb{D}$),

3) $Re(a) \wedge Re(z) = 1$ ($Re(a)$ est co-premier avec $Re(z)$ dans $\mathbb{Z}$).

Preuve

1) $\Leftrightarrow$ 2) et 2) $\Rightarrow$ 3) sont évidentes. Montrons que 3) $\Rightarrow$ 2).

Ecrivons $a = a_1 + a_2\epsilon$ et $z = z_1 + z_2\epsilon$ alors $Re(a) \wedge Re(z) = 1 \Leftrightarrow \exists (u_1, v_1) \in \mathbb{Z}^2$ tel que $a_1u_1 + z_1v_1 = 1$

Posons $w = -a_2u_1 - z_2v_1$ et $u_2 = u_1w$ et $v_2 = v_1w$, ainsi $u = u_1 + \epsilon u_2 = u_1 + \epsilon u_1 w$ et $v = v_1 + \epsilon v_2 = v_1 + \epsilon v_1 w$

Evaluons $au + zv$.

Nous avons $au+zv = a_1u_1+z_1v_1+\epsilon((a_1u_2+a_2u_1)+(z_1v_2+z_2v_1)) = 1+\epsilon(a_1u_1w+a_2u_1+z_1v_1w+z_2v_1) = 1$ parce qu'en substituant w par sa valeur $a_1u_1w + a_2u_1 + z_1v_1w + z_2v_1 = 0$. D'où le resultat cherché.

□

Algorithme Pseudo-Euclide Etendu

<u>En entrée</u> : $a = a_1 + a_2\epsilon,\ z = z_1 + z_2 \in \mathbb{D}\backslash J_{\mathbb{D}} \cup \mathbb{D}^*$

<u>En sortie</u> :

- "a n'est pas inversible dans $\frac{\mathbb{D}}{z\mathbb{D}}$", si $pgdc(a_1, z_1) \neq 1$
- $u = u_1 + u_2\epsilon$ et $v = v_1 + v_2\epsilon$ tel que $au + zv = 1$, si $pgdc(a_1, z_1) = 1$

1. si $pgdc(a_1, z_1) \neq 1$ renvoyer "a n'est pas inversible dans $\frac{\mathbb{D}}{z\mathbb{D}}$"
2. si $pgdc(a_1, z_1) = 1$
   (a) calculer $(u_1, v_1) \in \mathbb{Z}^2$ par l'Algorithme Euclide Etendu tel que $a_1u_1 + z_1v_1 = 1$
   (b) $w \leftarrow -a_2u_1 - z_2v_1$, $u_2 \leftarrow u_1w$ et $v_2 \leftarrow v_1w$
   (c) renvoyer $u \leftarrow u_1 + \epsilon u_2 = u_1 + \epsilon u_1 w$ et $v \leftarrow v_1 + \epsilon v_2 = v_1 + \epsilon v_1 w$

□

Remarquons que $au + zv = 1 \Leftrightarrow (au_1 + zv_1)(1 + w\epsilon) = 1 \Leftrightarrow au_1 + zv_1 = 1 - w\epsilon$.

Exemple Soit $a = 13 + 31\epsilon$ et $z = 6 - 22\epsilon$. Nous avons $13 \times (-5) + 6 \times 11 = 1$. Posons $w = -31 \times (-5) - (-22) \times 11 = 397$, $u_2 = -5 \times 397 = -1985$, $v_2 = 11 \times 397 = 4367$ alors $u = -5 - 1985\epsilon$ et $v = 11 + 4367\epsilon$. Ainsi l'inverse de $a = 13 + 31\epsilon$ dans $\frac{\mathbb{D}}{z\mathbb{D}}$ est $u \overset{\longleftrightarrow}{mod}\ z = (-5 - 1985\epsilon) \overset{\longleftrightarrow}{mod}\ (6 - 22\epsilon)$

Implémentation de l'algorithme Pseudo-Euclide Etendu en C++

La fonction suivante implémente l'algorithme Pseudo-Euclide Etendu qui nous permet d'inverser un entier dual modulo un autre entier dual. Un objet de la classe *duaux* représente un entier dual $a + b\epsilon$ où a et b sont des éléments de la classe ZZ (implémenté dans la librairie NTL([31])).

```
duaux InvDuaux(duaux & a, duaux & z){
  duaux b, k;
    ZZ dd, ds, dt, a1, z1, b1, b2, a2, z2, w, k1, k2;
    a1 = a.getX(); a2 = a.getY(); z1 = z.getX(); z2 = z.getY();
    XGCD(dd, ds, dt, a1, -z1);
```

```
        b1 = ds;
        k1 = dt;
        w = k1*z2-a2*b1;
        b2 = b1*w; k2 = k1*w;
        b = duaux(b1, b2);
        k = duaux(k1, k2);
      }
    else
        cout << "Non inversible" << endl;
  return b; //b est l'inverse de a modulo z
}
```

Proposition 0.0.10. *Soit $z = z_1 + z_2\epsilon$ un entier dual dans $\mathbb{D}\backslash J_{\mathbb{D}} \cup \mathbb{D}^*$ alors $|z_1| > 1$*

1. *et formellement $\frac{\mathbb{Z}}{|z_1|\mathbb{Z}} + \epsilon z_1\mathbb{Z} \subset \frac{\mathbb{D}}{z\mathbb{D}}$.*
2. *En outre $\left(\frac{\mathbb{Z}}{|z_1|\mathbb{Z}}\right)^* + \epsilon z_1\mathbb{Z} \subset \left(\frac{\mathbb{D}}{z\mathbb{D}}\right)^*$.*

La **preuve** est similaire aux preuves des théorèmes 0.0.4, 0.0.5, proposition 0.0.9 et lemme 0.0.6.

Arithmétique dans $\frac{\mathbb{D}}{p\mathbb{D}}[x]$ où p est un entier premier

Si p est un entier premier alors p est irréductible comme un entier premier mais p n'est pas nécessairement un entier dual premier. Ainsi si p est un entier dual alors $\frac{\mathbb{D}}{p\mathbb{D}}$ n'est pas nécessairement un anneau intègre.

Nous avons les isomorphismes suivants :
$\frac{\mathbb{D}}{p\mathbb{D}} = \frac{\frac{\mathbb{Z}[t]}{t^2\mathbb{Z}[t]}}{p\frac{\mathbb{Z}[t]}{t^2\mathbb{Z}[t]}} \cong \frac{\frac{\mathbb{Z}}{p\mathbb{Z}}[t]}{t^2\frac{\mathbb{Z}}{p\mathbb{Z}}[t]}$ si p est un entier premier.

Ainsi nous avons $\frac{\mathbb{D}}{p\mathbb{D}} = \frac{\mathbb{Z}[\epsilon]}{p\mathbb{Z}[\epsilon]} \cong \frac{\mathbb{Z}}{p\mathbb{Z}}[\epsilon]$ avec $\epsilon^2 = 0$, si p est entier premier.

Il est connu que l'on peut effectuer la division dans tout anneau polynomial si le coefficient dominant du diviseur est inversible. Dans la proposition suivante nous énonçons ce résultat dans le cas particulier où l'anneau est l'anneau quotient de l'anneau des entiers duaux.

Proposition 0.0.11. *Soit $P \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$ un polynomial non nul dont le coefficient dominant est inversible dans $\frac{\mathbb{D}}{p\mathbb{D}}$. Pour tout polynôme $F \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$ il existe des polynômes $Q, R \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$ tel que $P = FQ + R$ et $degR < degF$; En outre, le couple (Q, R) est unique.*

Pseudo-Factorisation dans $\frac{\mathbb{D}}{p\mathbb{D}}[x]$

.

Lemme 0.0.7. *1. Un élément $f = f_1 + f_2\epsilon \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$ avec $f_i \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x]$ est inversible si et seulement si la partie réelle f_1 de f est inversible c'est-à-dire que $f_1 = c \neq 0$ est une constante non nul dans $\frac{\mathbb{Z}}{p\mathbb{Z}}$. Par conséquent, les éléments inversibles sont de la forme $c + \epsilon h$ avec $h \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x]$ et $c \in \left(\frac{\mathbb{Z}}{p\mathbb{Z}}\right)^*$. Par ailleurs l'inverse de $(c + h\epsilon)$ est $c^{-1}(1 - hc^{-1}\epsilon)$ avec $c \in \left(\frac{\mathbb{Z}}{p\mathbb{Z}}\right)^*$.*

est égale à zéro. D'où les diviseurs de zéro sont de la forme ϵh avec $h \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x]$.

Preuve Simple. □

Lemme 0.0.8. *Tout élément $f = f_1 + f_2\epsilon \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$ avec $f_i \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x]$ et $deg(f_1) > 0$ peut être écrit sous la forme $f_1 + \epsilon f_2 = (f_1 + \epsilon r)(1 + \epsilon d)$ où $r = f_2 \ mod \ f_1$ et $d = f_2/f_1$ $(f_2 = f_1 d + r)$.*

Preuve Soit $f = f_1 + f_2\epsilon$ avec $deg(f_1) > 0$. Posons $r = f_2 \ mod \ f_1$ et $d = f_2/f_1$ alors $(f_1 + r\epsilon)(1 + d\epsilon) = f_1 + (f_1 d + r)\epsilon = f_1 + f_2\epsilon$

□

Lemme 0.0.9. *Un élément $f = f_1 + f_2\epsilon = (f_1 + \epsilon r)(1 + \epsilon d) \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$ où $r = f_2 \mod f_1 \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x]$ et $d = f_2/f_1 \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x]$, est irréductible si et seulement si $f_1 = q$ ou ($f_1 = q^\alpha$, $\alpha > 1$ et $r \wedge p = 1$) où q est un polynôme irréductible dans $\frac{\mathbb{Z}}{p\mathbb{Z}}[x]$.*

Preuve Soit $f = f_1 + f_2\epsilon \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$. Daprès le lemme 0.0.8 $f = (f_1 + \epsilon r)(1 + \epsilon d)$ avec $deg(r) < deg(f_1)$. Etudions l'irréductibilité des éléments de la forme $f_1 + \epsilon r$ avec $deg(r) < deg(f_1)$.

Soit $f = f_1 + \epsilon r \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$ avec $deg(r) < deg(f_1)$ et écrivons $f = f_1 + \epsilon r = (h + k\epsilon)(h' + k'\epsilon) = hh' + (hk' + kh')\epsilon$ alors :

$$\begin{cases} hh' = f_1 & (1) \\ hk' + kh' = r & (2) \end{cases}$$

Etudions les quatre cas suivants :

$1^{er} Cas$: Supposons que f_1 est irréductible, alors $hh' = f_1 \Rightarrow h = c$ ou $h' = c$ (avec $c \in (\frac{\mathbb{Z}}{p\mathbb{Z}})^*$). Dans les deux cas on voit que $f_1 + r\varepsilon$ est irréductible.

$2^{eme} Cas$: Supposons que $f_1 = q^\alpha$ ($\alpha > 1$ et $r \wedge q = 1$) où q est un polynôme irréductible. Posons $h = q^{\alpha_1}$ et $h' = q^{\alpha_2}$ avec $\alpha_1 + \alpha_2 = \alpha$ et $\alpha_1 < \alpha_2$

$(2) \Rightarrow q^{\alpha_1} k' + q^{\alpha_2} k = r \Rightarrow q^{\alpha_1}(k' + q^{\alpha_2 - \alpha_1} k) = r$

Si $r \wedge q = 1$ alors q^{α_1} est inversible dans $\frac{\mathbb{Z}}{p\mathbb{Z}}$, ce qui implique que $(h + \epsilon k)$ est inversible, d'où $f_1 + r\epsilon$ est irréductible.

$3^{eme} Cas$: Suppons que $f_1 = q^\alpha$ ($\alpha > 1$) et q divise r alors il existe un polynôme $r' \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x]$. tel que $r = qr'$. Donc $f_1 + r\epsilon = q^\alpha + qr'\epsilon = q(q^{\alpha-1} + r'\epsilon)$ or $\alpha > 1 \Rightarrow \alpha - 1 > 0 \Rightarrow q^{\alpha-1}$ est non inversible et en plus $deg(r) < deg(f_1)$. Par conséquent $f_1 + r\epsilon$ est réductible.

$4^{eme} Cas$: Supposons que $f_1 = f_1' * f_1^{"}$ avec $f_1' \wedge f_1^{"} = 1$ et $deg(f_1') \geq 1$ et $deg(f_1^{"}) \geq 1$ alors à partir de l'algorithme Euclide Etendu ils existent $u, v \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x]$ tel que $f_1' u + f^{"} v = 1$. Ainsi nous avons $f_1' ur + f^{"} vr = r$ d'où

$$(f_1' + rv\epsilon)(f_1^{"} + ru\epsilon) = f_1' f_1^{"} + (f_1' ur + f_1^{"} vr)\epsilon = f_1 + r\epsilon.$$

Ce qui prouve que $f_1 + r\epsilon$ est réductible car ni $f_1' + \epsilon rv$, ni $f_1^{"} + \epsilon ru$ n'est inversible.

□

Proposition 0.0.12. *Il n'y a pas de polynômes premiers dans $\frac{\mathbb{D}}{p\mathbb{D}}[x]$ (l'anneau quotient des entiers duaux où p est un entier premier).*

ci-dessus, les polynômes irréductibles à coefficients des entiers duaux sont de la forme $f = f_1 + f_2\epsilon = (f_1 + r\epsilon)(1 + \epsilon d) \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$ où $r = f_2$ *modulo* f_1 et $d = f_2/f_1$ et $f_1 = q$ ou $f_1 = q^\alpha$ ($\alpha > 1$ et $r \wedge q = 1$) où $q \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x]$ est un polynôme irréductible,

1. Soit $q \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x]$ un polynôme irréductible, prouvons que q n'est pas un polynôme premier dans $\frac{\mathbb{D}}{p\mathbb{D}}[x]$. Nous avons
$$(q + (1 + \frac{q+1}{2})\epsilon)(q + (-1 + \frac{q-1}{2})\epsilon) = (q^2 + q^2\epsilon) = q^2(1 + \epsilon),$$
d'où q (qui est irréductible) divise
$$(q + (1 + \frac{q+1}{2})\epsilon)(q + (-1 + \frac{q-1}{2})\epsilon)$$
mais ne divise pas $q + (1 + \frac{q+1}{2})\epsilon$ ni $q + (-1 + \frac{q-1}{2}\epsilon)$. Par conséquent q n'est pas un polynôme premier dans $\frac{\mathbb{D}}{p\mathbb{D}}[x]$.
2. Soit $(q^\alpha + r\epsilon)$ un polynôme irréductible dans $\frac{\mathbb{D}}{p\mathbb{D}}[x]$ avec $\alpha > 1$ et $r \wedge q = 1$, prouvons que $(q^\alpha + r\epsilon)$ n'est pas un polynôme premier. Nous avons :
$$(q^\alpha + (1 + \frac{r+1}{2})\epsilon)(q^\alpha + (-1 + \frac{r-1}{2})\epsilon) = (q^{2\alpha} + q^\alpha r\epsilon) = q^\alpha(q^\alpha + r\epsilon),$$
d'où $(q^\alpha + r\epsilon)$ (qui est irréductible) divise
$$(q^\alpha + (1 + \frac{r+1}{2})\epsilon)(q^\alpha + (-1 + \frac{r-1}{2})\epsilon)$$
mais ne divise ni $q^\alpha + (1 + \frac{r+1}{2})\epsilon$, ni $q^\alpha + (-1 + \frac{r-1}{2}\epsilon)$. Par conséquent $(q^\alpha + r\epsilon)$ n'est pas un polynôme premier dans $\frac{\mathbb{D}}{p\mathbb{D}}[x]$.

□

Théorème 0.0.6. *L'anneau $\frac{\mathbb{D}}{p\mathbb{D}}[x]$ (où $\mathbb{D}$ est l'anneau des entiers duaux et un entier premier), est pseudo-factoriel.*

Preuve

La preuve découle des propriétés des anneaux nœtheriens et de la définition d'un anneau pseudo-factoriel (voir proposition 0.0.7 et théorème 0.0.2).

□

Algorithme de pseudo-factorisation : on peut adapter étape par étape l'algorithme de pseudo-factorisation pour les entiers duaux du théorème 0.0.3.

□

Exemples 0.0.2. *Factorisons $f = f_1 + \epsilon f_2$ avec $f_1 = x^4 + x^3 + x^2 + x$ et $f_2 = x^2 - 1$ alors*

$f = (x + 1)(x^3 + x + \epsilon(x - 1))$

$f_1' = x^3 + x = x(x^2 + 1)$ *or* $x \wedge x^2 + 1 = 1$ *donc il existe* u *et* v *tel que* $xu + (x^2 + 1)v = 1$.

Le couple $(-x, 1)$ *est une solution de cette équation car* $x(-x) + x^2 + 1 = 1$

$x - 1 = (x - 1)x(-x) + (x - 1)(x^2 + 1)$

$f = (x - 1)(x^3 + x + \epsilon(x - 1)) = (x - 1)(x + \epsilon(x - 1))(x^2 + 1 + \epsilon(-x^2 + x))$

$x + \epsilon(x - 1) = (x - \epsilon)(1 + \epsilon)$ *et* $x^2 + 1 + \epsilon(-x^2 + x) = (x^2 + 1 + \epsilon(x + 1))(1 - \epsilon)$

$f = (x - 1)(x - \epsilon)(x^2 + 1 + \epsilon(x + 1))$

$\big(x+(1+\frac{x}{2})\epsilon\big)\big(x+(-1+\frac{x}{2})\epsilon\big) = (x^2+x^2\epsilon) = x^2(1+\epsilon)$, ainsi x qui est irréductible, divise $(x+(1+\frac{x}{2})\epsilon)(x+(-1+\frac{x}{2})\epsilon)$ mais il ne divise ni $x+(1+\frac{x}{2})\epsilon = \big(x+\epsilon\big)\big(1+\frac{1}{2}\epsilon\big)$ ni $x+(-1+\frac{x}{2})\epsilon = \big(x-\epsilon\big)\big(1+\frac{1}{2}\epsilon\big)$ qui sont non inversibles.

Sur l'inversibilité dans $\dfrac{\frac{\mathbb{D}}{p\mathbb{D}}[x]}{(g(x))}$

Soit $f, g \in \frac{\mathbb{D}}{z\mathbb{D}}[x]$, nous disons que f et g sont co-premiers(noté $f \wedge g = 1$) s'il existe $f', g' \in \frac{\mathbb{D}}{z\mathbb{D}}[x]$ tel que $ff' + gg' = 1 \overset{\longleftrightarrow}{mod}\ z$.

Soit $\mathbb{D} = \frac{\mathbb{Z}[\epsilon]}{(\epsilon^2)}$ l'anneau des entiers duaux, p un entier premier (alors p est irréductible dans $\mathbb{D}$).

Posons :

$$f = f_1 + \epsilon f_2 \in \frac{\mathbb{D}}{p\mathbb{D}}[x], \quad \text{avec } f_i \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x], i = 1, 2$$

et

$$g = g_1 + \epsilon g_2 \in \frac{\mathbb{D}}{p\mathbb{D}}[x], \text{ avec} \quad g_i \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x], i = 1, 2$$

Comment calculer l'inverse de f dans $\frac{\frac{\mathbb{D}}{p\mathbb{D}}[x]}{(g(x))}$?

$f \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$ est inversible dans $\frac{\frac{\mathbb{D}}{p\mathbb{D}}}{(g(x))}$ si et seulement si $\exists f' \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$ tel que $\overline{ff'} = \overline{1} \Leftrightarrow \exists f' = f'_1 + f'_2\epsilon, \exists k = k'_1 + k'_2\epsilon \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$ tel que $ff' - 1 = kg \overset{\longleftrightarrow}{mod}\ p$ dans $\mathbb{D}[x]$

$\Leftrightarrow$

$$\begin{cases} f_1f'_1 - 1 = k_1g_1 \mod p & (\alpha) \\ f_1f'_2 + f_2f'_1 = g_1k_2 + g_2k_1 \mod p & (\beta) \end{cases}$$

$\Leftrightarrow$

$$\begin{cases} f_1f'_1 + (-k_1)g_1 = 1 \mod p & (\alpha) \\ f_1f'_2 + f_2f'_1 = k_2g_1 + g_2k_1 \mod p & (\beta) \end{cases}$$

Supposons que f est inversible dans $\frac{\frac{\mathbb{D}}{p\mathbb{D}}[x]}{(g(x))}$.

$(\alpha) \Leftrightarrow \overline{f_1}\overline{f'_1} = \overline{1}$ dans $\frac{\mathbb{Z}}{p\mathbb{Z}}[x]$.

. En utilisant l'algorithme Euclide Etendu dans $\frac{\mathbb{Z}}{p\mathbb{Z}}[x]$ pour l'équation (α), nous pouvons trouver une solution $(f'_1, -k_1)$ pour (α) : $f_1f'_1 + (-k_1)g_1 = 1 \mod p$ (α).

. En remplaçant dans (β) nous avons

$$(\beta) \Leftrightarrow f_1f'_2 + (-k_2)g_1 = (-f'_1)f_2 + g_2k_1 \mod p.$$

Posons $h = -f'_1f_2 + g_2k_1 \mod p$ alors en multipliant (α) par h on a :

$$f_1(f'_1h) + (-k_1h)g_1 = h \mod p.$$

Ainsi, par identification, on peut choisir $f'_2 = f'_1h \mod p$ et $k_2 = k_1h \mod p$ comme une solution de (β).

Ainsi $f' = f'_1 + \epsilon f'_2 = f'_1 + f'_1h\epsilon = f'_1(1 + h\epsilon)$ est l'inverse de f dans $\frac{\frac{\mathbb{D}}{p\mathbb{D}}[x]}{(g(x))}$ avec $f'_1 = f_1^{-1}$ dans $\frac{\mathbb{D}}{p\mathbb{D}}[x]$

Cette remarque conduit au lemme suivant.

tions suivantes sont équivalentes :

1) f est inversible dans $\frac{\frac{\mathbb{D}}{p\mathbb{D}}[x]}{(g(x))}$,

2) $f \wedge g = 1$ (f est co-premier avec g dans $\frac{\mathbb{D}}{p\mathbb{D}}[x]$),

3) $Re(f) \wedge Re(g) = 1$ ($Re(f)$ est co-premier $Re(g)$ dans $\frac{\mathbb{Z}}{p\mathbb{Z}}[x]$).

Preuve

1) $\Leftrightarrow$ 2) et 2) $\Rightarrow$ 3) sont évidents. Prouvons que 3) $\Rightarrow$ 2).

Ecrire $f = f_1 + \epsilon f_2$ et $g = g_1 + \epsilon g_2$ alors

$$Re(f) \wedge Re(g) = 1 \Leftrightarrow \exists u_1, v_1 \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x]$$

tel que $f_1 u_1 + g_1 v_1 = 1$.

Posons $u_2 = u_1 h \quad v_2 = v_1 h$ où $h = -f_2 u_1 - g_2 v_1$. Définir

$$u = u_1 + \epsilon u_2 = u_1 + \epsilon u_1 h \text{ et } v = v_1 + \epsilon v_2 = v_1 + \epsilon v_1 h.$$

Evaluons $fu + gv$:

nous avons

$$fu + gv = f_1 u_1 + g_1 v_1 + \epsilon((f_1 u_2 + f_2 u_1) + (g_1 v_2 + g_2 v_1)) = 1 + \epsilon(f_1 u_1 h + f_2 u_1 + g_1 v_1 h + g_2 v_1) = 1$$

parce que en remplaçant h par sa valeur on a : $f_1 u_1 h + f_2 u_1 + g_1 v_1 h + g_2 v_1 = 0$.

□

Algorithme Pseudo-Euclide Etendu dans $\frac{\mathbb{D}}{p\mathbb{D}}[x]$

<u>En entrée</u> : $f = f_1 + \epsilon f_2 \in \frac{\mathbb{D}}{p\mathbb{D}}[x],\ f_i \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x], i = 1, 2$ et $g = g_1 + \epsilon g_2 \in \frac{\mathbb{D}}{p\mathbb{D}}[x],\ g_i \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x], i = 1, 2$

<u>En sortie :</u>

- "f n'est pas inversible dans $\frac{\frac{\mathbb{D}}{p\mathbb{D}}[x]}{(g(x))}$" si $pgdc(f_1, g_1) \neq 1$ in $\frac{\mathbb{Z}}{p\mathbb{Z}}[x]$
- $u = u_1 + u_2\epsilon \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$ et $v = v_1 + v_2\epsilon \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$ tel que $fu + gv = 1 \in \frac{\mathbb{D}}{p\mathbb{D}}[x]$, si $gcd(f_1, g_1) = 1$ dans $\frac{\mathbb{Z}}{p\mathbb{Z}}[x]$

1. si $pgdc(f_1, g_1) \neq 1$ dans $\frac{\mathbb{Z}}{p\mathbb{Z}}[x]$ renvoyer "f n'est pas inversible dans $\frac{\frac{\mathbb{D}}{p\mathbb{D}}[x]}{(g(x))}$"
2. si $pgdc(f_1, g_1) = 1 \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x]$
   (a) calculer en utilisant l'algorithme d'Euclide Etendu $(u_1, v_1) \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x]$ tel que $f_1 u_1 + g_1 v_1 = 1 \in \frac{\mathbb{Z}}{p\mathbb{Z}}[x]$
   (b) $h \leftarrow -f_2 u_1 - g_2 v_1$, $u_2 \leftarrow u_1 h$ et $v_2 \leftarrow v_1 h$
   (c) renvoie $u \leftarrow u_1 + \epsilon u_2 = u_1 + \epsilon u_1 h$ et $v \leftarrow v_1 + \epsilon v_2 = v_1 + \epsilon v_1 h$

Remarquons que

$$fu + gv = 1 \Leftrightarrow (fu_1 + gv_1)(1 + h\epsilon) = 1 \Leftrightarrow fu_1 + gv_1 = 1 - h\epsilon$$

□

Exemple

Soit $g(x) = x^{11} - 1$ et $p = 3$

Soit

$$f(x) = -1 + \epsilon + (1 + \epsilon)x^1 + (1 + \epsilon)x^2 + (-1 + \epsilon)x^4 + (1 + \epsilon)x^6 + (1 + \epsilon)x^9 + (-1 + \epsilon)x^{10}$$

L'inverse de f dans $\frac{\mathbb{D}[x]}{(g(x))}$ est :

$1 + (2 + 2\epsilon)x^1 + 2\epsilon x^2 + (2 + \epsilon)x^3 + (2 + \epsilon)x^4 + (1 + 2\epsilon)x^5 + (2 + 2\epsilon)x^7 + x^8 + (2 + \epsilon)x^9$

PROBLEME OUVERT : Trouver un Algorithme Pseudo-Euclide Etendu pour inverser un polynôme f, inversible dans $\frac{\mathbb{D}_z[x]}{(g(x))}$ où $z \in \mathbb{D} \setminus \mathbb{Z}$

Implémentation de l'algorithme de Pseudo-Euclide Etendu en C++

La fonction suivante implémente l'algorithme de Pseudo-Euclide Etendu décrit dans la sous section . Un objet de la classe *polynôme* représente un polynôme à coefficients entier duaux $\sum_{i=0}^{n}(a_i + b_i\epsilon)x^i$ où a et b sont des éléments de la classe ZZ (implémenté dans la librairie NTL([31])).

```
polynome invpolymodp(polynome& f)
{       int i;
    polynome h, k;
    ZZ_pX pd, x, ps, pt, f1, f2, g, h1, h2, k1, k2, w;
    ZZX h11, h22, k11, k22;
    int tab[12] = {-1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1};
    g.rep.SetLength(12);
    for(i = 0; i < 12; i++)
        g.rep[i] = tab[i];
    h = init(f.degre);
    f1 = PartiReel(f); f2 = PartiImag(f); h1 = PartiReel(h);
    h2 = PartiImag(h);
    XGCD(pd, ps, pt, f1, g);
    if(pd==1){
    h1 = ps;
    k1 = pt;
    x=MulMod(h1, f2, g);
    mul(w, x, -1);
    h2=MulMod(h1, w, g);
    k2=MulMod(k1, w, g);
    }
    else
    cout << "Non inversible " << endl;
    conv(h11, h1); conv(h22, h2); conv(k11, k1); conv(k22, k2);
    h = init(f.degre);
    for(i = 0; i <= deg(h1); i++){
        h.coef[i].x = h11.rep[i];
}
    for(i = 0; i <= deg(h2); i++){
        h.coef[i].y = h22.rep[i];
}
```

```
    cout << pd << " "<< h1 << " " << h2 << endl;
    return h;
}
```

DTRU : NTRU SUR LES ENTIERS DUAUX $\mathbb{D} = \mathbb{Z} + \epsilon\mathbb{Z}$

Jusqu'à présent toutes les variantes de NTRU ont été modélisées sur des anneaux intègres qui possèdent tous de bonnes propriétés arithmétiques. Ainsi fort des résultats du chapitre précédent, nous avons decidé de reprendre NTRU sur un anneau qui n'est pas intègre (c'est-à-dire un anneau avec des diviseurs de zéro).

NTRU sur un anneau Pseudo-Euclidien

Rappelons la définition d'un anneau Pseudo-Euclidien.

Un anneau A est pseudo-euclidien s'il existe une application $\varphi : A \to \mathbb{N}$ qui vérifie les propriétés suivantes :

1) $\varphi(z) \geq 0 \quad \forall z \in A$

2) $\forall z \in A$ et pour tout $t \in A - J_A$ où J_A est l'ensemble des diviseurs de zéro nous avons :

$$\varphi(z) \leq \varphi(zt),$$

3) Soit $z \in A$ et soit $t \in A - J_A$(où J_A est l'ensemble de diviseurs de zéro) alors il existe $(q, r) \in A^2$ tel que $z = tq + r$ avec $r = 0$ ou $\varphi(r) < \varphi(t)$

L'application φ définie ci-dessus, est appelée pseudo-norme.

Nous décrivons brièvement le schéma de chiffrement/déchiffrement en utilisant un anneau A non intègre (c'est-à-dire avec des diviseurs de zéro) et expliquons pourquoi il est commode lorsque A est pseudo-euclidien. Soit N un entier ; p, q des éléments de A. On pose

$\mathcal{P} = A[x], \qquad \mathcal{P}_p = \frac{\frac{A}{pA}[x]}{(x^N - 1)}, \qquad \mathcal{P}_q = \frac{\frac{A}{qA}[x]}{(x^N - 1)},$

où (p) et (q) sont des idéaux générés respectivement par p et q. Soient $\mathcal{L}_m$, $\mathcal{L}_g$ des sous-ensembles respectivement de $\mathcal{P}_p$ et $\mathcal{P}_q$, et soient $\mathcal{L}_f$, $\mathcal{L}_r$ des sous-ensembles de $\mathcal{P}$. Nous identifierons librement un élément de l'anneau $\mathcal{P}$ avec sa projection dans $\mathcal{P}_p$ et $\mathcal{P}_q$.

Génération des clés Pour générer les clés nous choisissons deux éléments $f \in \mathcal{L}_f$ et $g \in \mathcal{L}_g$. Supposons que f est inversible dans $\mathcal{P}_p$ et $\mathcal{P}_q$ et soient f_p, f_q ses inverses. Soit $h = p(f_q * g) \in \mathcal{P}_q$ la clé publique.

Chiffrement Pour chiffrer le message $m \in \mathcal{L}_m$, on choisit aléatoirement $\phi \in \mathcal{L}_\phi$ et $\psi \in \mathcal{L}_\phi$ et on calcule $e = \phi * h + \psi * h^2 + m \in \mathcal{P}_q$. e représente le texte chiffré.

Déchiffrement Pour déchiffrer, multiplions e par f^2 pour obtenir $a = p(\phi * f * g + p\psi * g^2) + f^2 * m \in \mathcal{R}_q$.

pour retrouver m.

Discussion Comme il a été indiqué par Kouzmenko dans sa thèse au sujet d'un anneau euclidien(voir[18]), il existe aussi deux problèmes dans la procédure ci-dessus pour un anneau Pseudo-Euclidien, si nous voulons l'appliquer dans la pratique :

1. Comment inverser f ?
2. Comment passer de calcul modulo p au calcul modulo q dans la partie déchiffrement ?

Pour répondre à ces questions, nous devons avoir les éléments suivants :

1. "Algorithme Pseudo-Euclide Etendu" pour inverser f dans $\frac{A}{(p)}[x]$ et $\frac{A}{(q)}[x]$, s'il est inversible.
2. définir $|f|_\infty = \max_{0\leq i\leq N-1} \varphi(f_i)$, ensuite, choisir les paramètres de telle sorte que

$$|p\phi * g + f * m|_\infty < \varphi(q) \qquad (E)$$

et passer du calcul modulo p au calcul modulo q avec $\varphi(q) > \varphi(p)$.

Puisqu'en général, l'algorithme de Pseudo-division ne donne pas un reste unique, nous avons besoin de modifier la procédure ci-dessus. Par exemple (ce sera le cas pour les entiers duaux), il peut exister une constante K telle que si

$$\varphi(x) < \frac{\varphi(q)}{K}$$

alors x est le représentant unique de la classe d'équivalence mod q avec cette propriété. Ensuite, pour que la modification de NTRU fonctionne, il suffit de remplacer $\varphi(q)$ par $\frac{\varphi(q)}{K}$ dans l'équation (E).

Dans la dernière partie du présent paragraphe, pour illustrer ces idées, nous allons montrer comment NTRU peut être modélisé concrètement sur l'anneau des entiers duaux (un anneau non euclidien et non intègre).

NTRU sur les Entiers Duaux

En résumant les resultats de la section et en les combinant avec les discussions de cette section, nous obtenons le théorème suivant. Dans le reste de ce chapitre , chaque fois que nous parlons de réduction de y *mod* x où x est un entier dual, nous signifions la sortie r de l'algorithme de pseudo-division dejà rencontré car on a déjà prouvé que cette sortie est unique pour une même entrée.

Théorème 0.0.7. *Soient $p, q \in \mathbb{D}$ des entiers duaux.*

1. *Soient f, g des polynômes dans $\mathbb{D}[x]$. Supposons que f est inversible dans $\frac{\mathbb{D}}{(p)}[x]$ et $\frac{\mathbb{D}}{(q)}[x]$ avec respectivement f_p et f_q les inverses.*
2. *Définir $h = p(g * f_q) \in \frac{\mathbb{D}}{(q)}[x]$.*
3. *Soit $m \in \frac{\mathbb{D}}{(p)}[x]$ tel que $|m|_\infty < \frac{\varphi(p)}{4}$.*
4. *Soient $\phi, \psi \in \frac{\mathbb{D}}{(q)}[x]$. Définir $e = \phi * h + \psi * h^2 + m \in \frac{\mathbb{D}}{(q)}[x]$.*
5. *Supposons que $|p(\phi * f * g + p\psi * g^2) + f^2 * m|_\infty < \frac{\varphi(q)}{4}$, où les opérations sont polynômiales et se font dans $\mathbb{D}[x]$.*
6. *Alors pour déchiffrer, calculons successivement : $a = f^2 * e(modq)$ et $f_p^2 * a(modp)$*
7. *retourne le texte clair $m = f_p^2 * a(modp)$*

□

Comme nous ne sommes pas en mesure d'inverser jusqu'à présent, en général, un polynôme f dans $\frac{\frac{\mathbb{D}}{z\mathbb{D}}[x]}{(g(x))}$ où $z \in \mathbb{D}\backslash\mathbb{Z}$ mais nous pouvons le faire si $z \in \mathbb{Z}$, nous allons faire une version de **DTRU** que nous pourrons mettre en œuvre en utilisant les propriétés arithmétiques du chapitre . Pour cette version nous considérons que $p,\ q \in \mathbb{Z}$ et nous appelons **DTRU1** l'algorithme de chiffrement correspondant.

Génération des clés

1. Choisir $p,\ q \in \mathbb{Z}$ avec q/p grand.
2. Choisir aléatoirement un polynôme avec de petits coefficients $f \in \mathcal{L}_f \subset \frac{\mathbb{D}[x]}{(x^N-1)}$, inversible $mod\ q$ et $mod\ p$ et on note ses inverses par f_q et f_p, c'est-à-dire, $f_q * f \equiv 1(mod\ q)$ et $f_p * f \equiv 1(mod\ p)$
3. Choisir aléatoirement un polynôme avec de petits coefficients $g \in \mathcal{L}_g \subset \frac{\frac{\mathbb{D}}{(q)}[x]}{(x^N-1)}$.
4. Calculer $h = p.(f_q * g)(mod\ q) \in \frac{\frac{\mathbb{D}}{(q)}[x]}{(x^N-1)}$, h est la clé publique.

   NB : Les polynômes f et g ont généralement de petits coefficients, pendant que h a de grands coefficients.

Chiffrement Supposons que Aissa veuille envoyer un message à Oumar. Aissa exécute les opérations suivantes :

1. sélectionne un message m dans l'ensemble des textes clairs $\frac{\frac{\mathbb{D}}{(p)}[x]}{(x^N-1)}$, où $p \in \mathbb{Z}$: choisit les coefficients $a + b\epsilon$ de m avec $a, b \in \mathfrak{L}_m = \{-\frac{p-1}{2}, \ldots, -1, 0, 1, \ldots, \frac{p-1}{2}\}$ si p est impair et $a, b \in \mathfrak{L}_m = \{-\frac{p}{2} + 1, \ldots, -1, 0, 1, \ldots, \frac{p}{2}\}$ si p est pair.
2. choisit aléatoirement deux polynômes $\phi, \psi \in \mathcal{L}_\phi \subset \frac{\frac{\mathbb{D}}{(p)}[x]}{(x^N-1)}$
3. et utilise la clé publique h de Oumar pour calculer

$$e = \phi * h + \psi * h^2 + m\ (modq) \in \frac{\frac{\mathbb{D}}{(q)}[x]}{(x^N - 1)}$$

4. e est le message chiffré que Aissa va envoyer à Oumar.

Déchiffrement

Supposons que Oumar ait reçu le message e transmis par Aissa et veuille le déchiffrer en utilisant sa clé privé f^2. Pour réussir ceci efficacement, Oumar devrait d'abord calculer le polynôme f_p^2.

1. Pour déchiffrer e, Oumar, calcule d'abord : $a = f^2 * e\ (modq) \in \frac{\frac{\mathbb{D}}{(q)}[x]}{(x^N-1)}$
2. Choisit les coefficients de a entre $-\frac{q}{2}$ et $\frac{q}{2}$.
3. Calcule $m = f_p^2 * a\ (modp) \in \frac{\frac{\mathbb{D}}{(p)}[x]}{(x^N-1)}$

□

Exemples 0.0.3. *Pour l'exemple nous allons utiliser les valeurs suivantes :*

$N = 7$, $p = 3$ *et* $q = 51$

Génération des clés

Oumar veut créer la paire de clés publique/privée en utilisant notre Cryptosystème ***DTRU1****. Premièrement il choisit aléatoirement deux petits polynômes f et g dans $\frac{\mathbb{D}}{(q)}[x]$*

1) *Par exemple*

$g = (-1+\epsilon)x + (-1+\epsilon)x^2 + (1+\epsilon)x^4 + (1+\epsilon)x^6$

2) *Puis il calcule l'inverse f_p de f modulo p et l'inverse f_q de f modulo q en utilisant l'algorithme Pseudo-Euclide Etendu. Il trouve :*

$f_p = 1 + 2\epsilon + (1+2\epsilon)x + (1+2\epsilon)x^2 + (1+2\epsilon)x^3 + 2\epsilon x^4 + (2+\epsilon)x^5 + (1+2\epsilon)x^6$

$f_q = 1 + 41\epsilon + (25+29\epsilon)x + (13+23\epsilon)x^2 + (19+32\epsilon)x^3 + (42+50\epsilon)x^4 + (5+4\epsilon)x^5 + (49+20\epsilon)x^6$

3) *Enfin il calcule le produit*

$h = pg * f_q = 21 - 3\epsilon + (15-6\epsilon)x + (-6-3\epsilon)x^2 + (6-18\epsilon)x^3 + (24-18\epsilon)x^4 - 12x^5 + (3+9\epsilon)x^6$

La clé privée de Oumar est la paire de polynômes f^2 et f_p^2 et sa clé publique est le polynôme h.

Chiffrement

Maintenant, supposons que Aissa veuille envoyer un message

$m = -1 + \epsilon + (1+\epsilon)x + (-1+\epsilon)x^2 + (1+\epsilon)x^3 + (-1+\epsilon)x^5$,

à Oumar en utilisant la clé publique h de Oumar.

1) *Elle choisit d'abord deux polynômes ϕ et ψ aléatoirement de degrés inférieurs ou égaux à 6.*

$\phi = -1 + \epsilon + (1+\epsilon)x + (-1+\epsilon)x^5 + (1+\epsilon)x^6$

et

$\psi = -1 + \epsilon + (1+\epsilon)x^3 + (-1+\epsilon)x^4 + (1+\epsilon)x^6$

2) *Alors son message chiffré e est*

$e = \phi * h + \psi * h^2 + m \ mod(51 + 0\epsilon) = -10 - 8\epsilon + (4+22\epsilon)x + (-16-14\epsilon)x^2 + (22-8\epsilon)x^3 - 12\epsilon x^4 + (-22+10\epsilon)x^5 + (21+15\epsilon)x^6 \ mod(51+0\epsilon)$

Déchiffrement

Oumar reçoit le message chiffré e de Aissa. Il utilise sa clé privée f pour calculer

1) $a = f^2 * e = 10 - 15 + (18+19\epsilon)x + (-5-24\epsilon)x^2 + (-12-23\epsilon)x^3 + (-4+15\epsilon)x^4 + (-13-13\epsilon)x^5 + (8-21\epsilon)x^6 (mod\ 51 + 0\epsilon)$

Noter que, quand Oumar réduit les coefficients de $f^2 * e$ modulo $51 + 0\epsilon$, il choisit des valeurs qui se trouvent entre -24 and 25. Après Oumar réduit les coefficients de a modulo $3 + 0\epsilon$ pour otbtenir

2) $b = 1 + \epsilon x + x^2 + \epsilon x^3 - x^4 + (-1-\epsilon)x^5 - x^6 \ mod(3+0\epsilon)$

Finalement Oumar utilise f_p, pour calculer

3) $\overline{m} = f_p^2 * b = -1 + \epsilon + (1+\epsilon)x + (-1+\epsilon)x^2 + (1+\epsilon)x^3 + (-1+\epsilon)x^5 \ mod(3+0\epsilon)$

Le polynôme $\overline{m}$ est le message m de Aissa. Ainsi Oumar a déchiffré avec succès le message Aissa.

Vitesse de Chiffrement/Déchiffrement

Nous calculons la complexité de chiffrement et de déchiffrement des messages avec DTRU1.

Soientt $a = a_1 + a_2\epsilon$ et $b = b_1 + b_2\epsilon$ deux entiers duaux. Additionner a et b nécessite deux additions des entiers, multiplier a et b nécessite 3 multiplications d'entiers et une addition d'entiers puisque

$$a + b = (a_1 + b_1) + (a_2 + b_2)\epsilon, \ a * b = a_1 b_1 + (a_1 b_2 + a_2 b_1)\epsilon$$

L'algorithme de pseudo-division utilise 2 multiplications des entiers duaux et deux réductions d'entiers, d'où deux opérations élémentaires dans $\mathbb{Z}$. Au total 10 opérations élémentaires dans $\mathbb{Z}$.

éléments aléatoirement $\phi,\ \psi \in \mathcal{L}_\phi$, et après on calcule

$$e = \phi * h + \psi * h^2 + m$$

Il utilise $3N^2$ multiplications d'entiers duaux pour calculer $\phi * h + \psi * h^2$ et $2N$ additions d'entiers Duaux pour obtenir e. Ainsi nous avons besoin de $9N^2$ multiplications d'entiers et $N(3N+4)$ additions entiers au total. D'où,

$$4N(3N+1)$$

operations élémentaires sont nécessaires pour chiffrer.

Dans l'étape de déchiffrement, nous multiplions e par f^2 pour obtenir a, pour lequel nous avons besoin de N^2 multiplications d'entiers duaux, effectuons une réduction modulo p et multiplions a par f_p^2 pour obtenir le résultat final, pour lequel, nous aurons besoin de N^2 multiplications d'entiers duaux. Au total, $6N^2$ multiplications d'entiers, plus $2N^2$ additions d'entiers et $2N$ reductions d'entiers. D'où

$$2N(4N+1)$$

opérations élémentaires sont nécessaires pour déchiffrement.

Paramètres de NTRU

Nous devons choisir les paramètres de sorte que le déchiffrement soit presque toujours possible et rendre les attaques difficiles.

Donc, pour contourner ce problème, nous avons décidé d'utiliser de petites valeurs pour les coefficients de nos polynômes.

1) Exemple pour $p = 3$, nous utilisons les valeurs de l'ensemble suivant : $\{a + \epsilon b,\ a, b \in \{0, 1, -1\}\}$ et générons nos polynômes en suivant la règle ci-dessous :

on définit l'ensemble $\mathcal{L}(d) := \{f | f \in \mathcal{P}$ avec d coefficients egaux à $-1, 1, -\epsilon, \epsilon$ et le reste $0\}$. Soient d_f, d_g, d_ϕ et d_m des entiers. Pour le choix des clés, des éléments aléatoires et des messages, nous considérons les ensembles suivants :

$\mathcal{L}_f = \mathcal{L}(d_f)$, $\mathcal{L}_g = \mathcal{L}(d_g)$, $\mathcal{L}_\phi = \mathcal{L}(d_\phi)$ et $\mathcal{L}_m = \mathcal{L}(d_m)$

Nous devons choisir la taille des paramètres(i.e des ensembles $\mathcal{L}_f$, $\mathcal{L}_g$, $\mathcal{L}_\phi$, $\mathcal{L}_m$) pour éviter l'attaque par force brute et d'autres attaques telles que les attaques contre les réseaux.

2) Exemple pour $p = 2$, nous utilisons les valeurs de l'ensemble suivant : $\{a + \epsilon b,\ a, b \in \{0, 1\}\}$ et générons nos polynômes en suivant la règle ci-dessous :

définir l'ensemble $\mathcal{B}(d) := \{f | f \in \mathcal{P}$ avec d coefficients égaux à $1, \epsilon$ et le reste $0\}$. Soit d_f, d_g, d_ϕ et d_m des entiers. Pour le choix des clés, des éléments aléatoires et des messages, nous considérons les ensembles suivants :

$\mathcal{B}_f = \mathcal{B}(d_f)$, $\mathcal{B}_g = \mathcal{B}(d_g)$, $\mathcal{B}_\phi = \mathcal{B}(d_\phi)$ et $\mathcal{B}_m = \mathcal{L}(d_m)$

Nous devons choisir la taille des paramètres (c'est-à-dire des ensembles $\mathcal{B}_f$, $\mathcal{B}_g$, $\mathcal{B}_\phi$, $\mathcal{B}_m$) pour éviter l'attaque par force brute et d'autres attaques telles que les attaques contre les réseaux.

Attaques par force Brute

Un attaquant peut retrouver la clé privée en essayant tous les $f \in \mathcal{L}_f$ et tester si les coefficients de $f*h$ (modulo q) sont pétits, ou en essayant tous les $g \in \mathcal{L}_g$ et tester si les coefficients de $g*h^{-1}$ (modulo q) sont petits. De la même manière, un attaquant peut retrouver un message en essayant les $r \in \mathcal{L}_r$ possibles et tester si les coefficients de $e - pr * h$(modulo q) sont petits.

Rappelons que nos paramètres sont définis comme suit :

$\mathcal{L}(d) := \{f | f \in \mathcal{P}$ avec d coefficients egaux à $-1, 1, -\epsilon, \epsilon$ et le reste $0\}$. ou

$\mathcal{B}(d) = \{f \in R/f$ a d coeffiecints egaux à $1,\ \epsilon$ et le reste $0\}$

Pour la taille nous avons :

$$|\mathcal{L}(d)| = \begin{pmatrix} N \\ d \end{pmatrix} \begin{pmatrix} N-d \\ d \end{pmatrix} \begin{pmatrix} N-2d \\ d \end{pmatrix} \begin{pmatrix} N-3d \\ d \end{pmatrix} = \frac{N!}{(d!)^4(N-4d)!}$$

$$|\mathcal{B}(d)| = \begin{pmatrix} N \\ d \end{pmatrix} \begin{pmatrix} N-d \\ d \end{pmatrix} \begin{pmatrix} N-d \\ d \end{pmatrix} = \frac{N!}{(d!)^3(N-3d)!}$$

Si k est le paramètre de sécurité, nous devons choisir $\sqrt{|\mathcal{L}(d)|} > 2^k$ ou $\sqrt{|\mathfrak{B}(d)|} > 2^k$ pour éviter les attaques par force brute.

L'attaque réseau

Nous pouvons facilement généraliser l'attaque contre les réseaux([17]) à notre système.

L'attaque des reseaux contre la clé publique

Rappelons que $h = pg * f_q \mod q$ est la clé publique. Nous avons l'équation suivante :

$f * h = pg \mod q$. De cette équation nous avons

$$\begin{cases} f_1 h_1 = pg_1 \mod q \quad (\alpha) \\ f_1 h_2 + f_2 h_1 = pg_2 \mod q \quad (\beta) \end{cases}$$

avec $f = f_1 + \epsilon f_2$, $h = h_1 + \epsilon h_2$ et $g = g_1 + \epsilon g_2$

Pour l'équation (α)

Soit θ une petite valeur. Soit $A = (A_0, A_1, ..., A_{N-1})$ et $B = (B_0, B_1, ..., B_{N-1})$ deux polynômes. Définissons le réseau $\mathfrak{L}_{key} = \{(\theta A, B)/A * h_1 = B \mod q\}$, ainsi (f_1, g_1) est un élément de $\mathfrak{L}_{key}$. Ce réseau est de dimension $2N$.

Comme dans le NTRU classique, la valeur optimale de θ est $\theta = \frac{|f|_\infty}{|g|_\infty}$ et dans ce cas la valeur :

$$c_h = \frac{|\text{vecteur cible}|}{|\text{vecteur court esperé}|} = \sqrt{\frac{2\pi e |f|_\infty . |g|_\infty}{Nq}}$$

mesure combien y a-t-il d'écart entre le réseau associé au cryptosystème à un réseau aléatoire. On doit choisir les différents polynômes tel que $c_h \sim 1$

On a un résultat similaire pour le message.

Avec la clé publique $h = pg * f_q \mod q$, nous avons $f * h = pg \mod q$, et un attaquant peut diviser f au milieu, c'est-à-dire $f = f_a + f_b$ (en ajoutant $N/2$ zéros à f_a et f_b pour avoir des vecteurs de taille N), et calculer

$f_a * h + f_b * h = pg \mod q \Leftrightarrow$

$$\begin{cases} f_{a1}h_1 + f_{b1}h_1 = pg_1 \mod q \quad (1) \\ f_{a1}h_2 + f_{a2}h_1 + f_{b1}h_2 + f_{b2}h_1 = pg_2 \mod q \quad (2) \end{cases}$$

avec $f_a = f_{a1} + \epsilon f_{a2}$, $f_b = f_{b1} + \epsilon f_{b2}$, $h = h_1 + \epsilon h_2$ et $g = g_1 + \epsilon g_2$

L'attaque du milieu consiste de tester tous les polynômes f_{a1} et f_{b1} dont le nombre total de coefficients non nuls égaux à 1 est d_f en calculant $f_{a1}h_1 + f_{b1}h_1 \mod q$ et en testant si le polynôme résultant est de la forme pg_1 avec $g_1 \in \mathcal{L}_{g_1}$.

Soit $\mathcal{L}_1(d) := \{f | f \in \mathcal{P}$ avec d coefficients égaux à $-1, 1$ et le reste $0\}$ et

$\mathfrak{B}_1(d) = \{f \in R/f$ a d coefficients égaux à $1, \epsilon$ et le reste $0\}$.

Si k est le paramètre de sécurité, nous devons choisir $\frac{\sqrt{|\mathcal{L}_1(d)|}}{\sqrt{N}} > 2^k$ ou $\frac{\sqrt{|\mathfrak{B}_1(d)|}}{\sqrt{N}} > 2^k$ où $d = d_{f_1}$ pour éviter les attaques forcées avec rencontre au milieu contre la clé publique de NTRU et le texte chiffré de NTRU (voir [14]).

On a un resultat similaire pour le message.

Niveau de Sécurité

Nous proposons de choisir les paramètres comme suit :

- N est un entier premier > 100
- d_f est le petit entier tel que $\frac{\sqrt{|\mathcal{L}_1(d)|}}{\sqrt{N}} > 2^k$ avec $d_f \leq N/4$ ($pour\ p = 3$) et $d_f \leq N/2$ ($pour\ p = 2$)
- $d_g \approx N/4$ ($pour\ p = 3$) et $d_g \approx N/2$ ($pour\ p = 2$)
- $d_r \approx d_f$
- choisir q de sorte que le déchiffrement fonctionne.

Par exemple, nous proposons les deux tables suivantes pour $p = 2$ et $p = 3$ respectivement.

Securité :	N	p	q	d_f	d_g	d_r
Securité Moyenne$> 2^{80}$	251	2	128	72	71	72
Haute securité $> 2^{120}$	347	2	128	74	94	74
Très haute securité $> 2^{170}$	503	2	256	220	241	220

Securité :	N	p	q	d_f	d_g	d_r
Securité Moyenne $> 2^{80}$	131	3	128	45	47	45
Haute securité $> 2^{120}$	211	3	196	91	95	91
Très haute securité $> 2^{170}$	251	3	256	101	105	101

Il faut juste signaler que nous nous sommes appuyés sur les travaux de **William D. Banks** et de **Igor E. Shparlinski**([1]). Dans ces travaux réalisés en 2002, ils ont proposé une nouvelle variante de NTRU sans inverse de polynômes dans le même corps que NTRU classique.

Dans cette nouvelle généralisation de la version originale du cryptosystème NTRU, on selectionne les parametres (N, p, q) où N est un entier et p et q sont des entiers duaux et quatre ensembles $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r, \mathcal{L}_m$ des polynômes dans l'anneau $\mathcal{P} = \frac{\mathbb{D}[x]}{(x^N-1)}$. Nous notons par $*$ l'operation de multiplication dans l'anneau $\mathcal{P}$. Les paramètres p et q sont des entiers duaux distincts et sont irréductibles avec $\varphi(q) > \varphi(p)$, et les ensembles $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r, \mathcal{L}_m$ sont choisis pour satisfaire la condition suivante

$$\|pr * g + f * m\|_\infty < \frac{\varphi(q)}{4}$$

pour tous les polynômes $f \in \mathcal{L}_f$, $g \in \mathcal{L}_g$, $r \in \mathcal{L}_r$, $m \in \mathcal{L}_m$ où pour un polynôme quelconque

$$F = F_0 + F_1 x^1 + ... + F_{N-1} x^{N-1},$$

nous définissons la norme de F par

$$\|F\|_\infty = \max_{0 \le i \le N-1} \varphi(f_i),$$

Notre généralisation de la version de **William D. Banks** et de **Igor E. Shparlinski**([1]) peut être décrite comme suit.

Création des clés Oumar sélectionne aléatoirement trois polynômes $f \in \mathcal{L}_f$, $g \in \mathcal{L}_g$ et $G \in \mathcal{P}$ tel que G soit inversible modulo q et f aussi soit inversible modulo p. Oumar calcule d'abord les inverses G_q et f_p (en utilisant l'Algorithme de Pseudo-Euclide Etendu), qui vérifient

$$G * G_q \equiv 1 \quad (modq), \qquad f * f_p \equiv 1 \quad (modp)$$

ensuite Oumar calcule les produits

$$h \equiv G_q * g \quad (modq), \qquad H \equiv G_q * f \quad (modq).$$

Oumar publie le couple des polynômes (h, H) comme sa clé publique, en gardant(f, g, G) comme sa clé privée. Le polynôme f_p sera utilisé plus tard, et le polynôme G_q peut être écarté.

Chiffrement. Supposons Aissa veuille envoyer un message secret à Oumar. Aissa selectionne un message m dans $\mathcal{L}_m$ qui est l'espace des messages clairs. Ensuite, Aissa sélectionne aléatoirement un polynôme $r \in \mathcal{L}_r$ et elle utilise la clé publique de Oumar (h, H) pour calculer

$$e \equiv pr * h + H * m \quad (modq).$$

Aissa transmet alors e à Oumar.

Déchiffrement. Oumar reçoit e de la part de Aissa. Pour déchiffrer le message, il calcule d'abord

$$a \equiv G * e \equiv pr * g + f * m \quad (modq),$$

en calculant

$$m \equiv f_p * a \quad (modp)$$

Démonstration

Si tous les coefficients de a sont entre -$\frac{\varphi(q)}{2}$ et $\frac{\varphi(q)}{2}$ alors il vérifie :

$$a \equiv G * e \equiv G * (pr * h + H * m) \quad (modq),$$

$$\equiv pr * G * h + G * H * m \quad (modq),$$

$$\equiv pr * G * (G_q * g) + G * (G_q * f) * m \quad (modq),$$

$$\equiv pr * g) + f * m \quad (modq),$$

De la même manière, le polynôme b vérifie :

$$b \equiv f_p * a \equiv f_p * (pr * g) + f * m) \quad (modp),$$

$$\equiv f_p * f * m \quad (modp),$$

$$b \equiv m \quad (modp),$$

On vérifie facilement que le cas $G = f$ et $\mathbb{D} = \mathbb{Z}$ correspond bien à la version originale du cryptosystème NTRU (dans ce cas, $H = 1$, ainsi la clé publique constitue seulement le polynôme h).

Résumé

Le cryptosystème NTRU a été proposé en 1996 par J. Hoffstein, J.Pipher et J.H. Silverman. La sécurité de NTRU est basée sur certains problèmes difficiles dans les réseaux. L'étude mathématique des réseaux euclidiens remonte à la fin du 19ème siècle, avec les travaux de Hermite et Minkowski sur la géométrie des nombres. NTRU a été généralisé sur plusieurs anneaux qui sont tous intègres. Notre objectif était de généraliser NTRU sur des anneaux non intègres. Pour cela nous avons ciblé l'anneau des entiers duaux $\mathbb{D} = \mathbb{Z} + \epsilon\mathbb{Z}$ qui n'est pas intègre.

Nous avons d'abord réussi à étudier quelques propriétés arithmétiques intéressantes de l'anneau des entiers duaux ($\mathbb{D} = \mathbb{Z} + \epsilon\mathbb{Z}$ avec $\epsilon^2 = 0$). Nous avons rencontrer plusieurs difficultés pour adapter les preuves et les techniques connues sur $\mathbb{Z}$ et $K[x]$, aux entiers duaux à cause de l'existance des diviseurs de zéro dans $\mathbb{D}$. Nous avons aussi étudié quelque propriétés arithmétiques sur $\frac{\mathbb{D}}{p\mathbb{D}}[x]$ où p est un entier premier

Les outils arithmétiques développés sur les entiers duaux nous ont permis de généraliser le protocol NTRU sur un anneau non intègre. Le résultat obtenu a une sécurité similaire à celle de NTRU mais n'est pas plus efficace que NTRU.

Tous les algorithmes dans $\mathbb{D}$, $\mathbb{D}[x]$ et $\frac{\mathbb{D}}{p\mathbb{D}}[x]$ et les opérations dans DTRU sont implémentés en C/C++ en utilisant le package NTL de Victor Shoup.

Mots-clés : Entier Dual, Arithmétique, Cryptographie, réseaux euclidiens, NTRU.

Perspectives de recherches

Nous comptons poursuivre nos recherches sur l'arithmétique et sur d'autre versions de NTRU. Plus particulièrement nous travaillerons sur les points suivants :

- Terminer l'arithmétique sur $\frac{\mathbb{D}}{z\mathbb{D}}[x]$ où $z \in \mathbb{D}\backslash\mathbb{Z}$.
- Faire les autres versions de NTRU.
- Fabriquer une signature dans le modèle de NTRU sur $\frac{\mathbb{D}}{z\mathbb{D}}[x]$ où $z \in \mathbb{D}\backslash\mathbb{Z}$.
- Etudier l'arithmétique de l'anneau $\frac{\mathbb{Z}[t]}{(t^2-t)}$.
- Modéliser NTRU sur $\frac{\mathbb{Z}[t]}{(t^2-t)}$.

Bibliographie

[1] W. D. Banks and I. Shparlinski. *A variant of NTRU with non-invertible polynomials*, INDOCRYPT, 2002

[2] M. G. Camara and D. Sow *Introduction to the arithmetic of the dual integers*, Far East J. Math. Sci.(FJMS)57(1)(2011), 13-39

[3] M. Coglianese, B.-M. Goi, MaTRU *A new NTRU-based cryptosystem, ACISP Progress in Cryptology-INDOCRYPT 2005 :* 6th International Conference on Cryptology in India 2005, Bangalore, India, Springer-Verlag, 2005.

[4] D. Coppersmith and A. Shamir. *Lattice attacks on NTRU*, In Advances in cryptologyUEUROCRYPT 97, volume 1233 of Lecture Notes in Comput. Sci., pages 52-61. Springer, Berlin, 1997.

[5] C. Dwork. Lattices, and their application to cryptography. Lecture notes, Stanford University, 1998. available at http ://www.dim.uchile. cl/ mkiwi/topicos/00/dwork-lattice-lectures.ps.

[6] C. Dwork, M. Naor and O. Reingold *Immunizing Encryption Schemes from Decryptions Errors*, In Eurocrypt 2004, Springer-Verlag (LNCS 3027)(2004),

[7] P. Gaborit, J. Ohler, P. Solé, CTRU, a polynomial analogue of NTRU, Rapport de Recherche, INRIA, 2002, no RR-4621 *http ://www.inria.fr/rrrt/rr-4621.html*

[8] T. E. Gamal. *A public key cryptosystem and a signature scheme based on discrete logarithms.* IEEE Trans. Inform. Theory, 31 :469-472, 1985.

[9] P. A. Grillet *Abstract Algebra(Graduate Texts in Mathematics)* Springer (Second edition) 2007

[10] D. Han *A new lattice attack on NTRU cryptosystem* Trends in Mathematics Information Center for Mathematical Sciences Volume 8, Number 1, June, 2005, Pages 197-205

[11] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU : A Ring Based Public Key Cryptosystem in Algorithmic Number Theory. Lecture Notes in Computer Science 1423, Springer-Verlag, pages 267 to 288, 1998.

[12] J. Hoffstein and J. H. Silverman. Implementation Notes for NTRU PKCS Multiple Transmissions. Technical Report No. 6. available at http ://www.ntru.com.

[13] J. Hoffstein and J. H. Silverman. Protecting NTRU against chosen ciphertext and reaction attacks. NTRU Cryptosystems technical report No 16. available at http ://www.ntru.com.

[14] N. Howgrave-Graham, J. Silverman, A. Singer, and W. Whyte. *NAEP : Provable Security in the Presence of Decryption Failures.* NTRU Cryptosystems 2003, available at http ://www.ntru.com.

[15] N. Howgrave-Graham, J. Silverman, and M. Whyte. *Choosing Parameter Sets for NTRUEncrypt with NAEP and SVES-3.* 2005. available at http : //www.ntru.com.

Private Key. Technical Report No. 4. available at http ://www.ntru.com.

[17] IEEE P1363.1 Public-Key Cryptographic Techniques Based on Hard Problems over Lattices, June 2003. IEEE.

[18] R. Kouzmenko. *Generalizations of the NTRU cryptosystem*, Diploma Project, Winter Semester 2005-2006

[19] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovasz. Factoring Polynomials with rational coefficients. Mathematische *Ann.*, 261 :513-534, 1982.

[20] D. Micciancio. Lattices in Cryptography and Cryptanalysis. Course notes. available at http ://www-cse.ucsd.edu/ daniele/cse207c.

[21] E. Malekian, A. Zakerolhosseini and A. Mashatan. *Qtru : A Lattice Attack Resistant Version of Ntru*,http ://eprint.iacr.org/2009/386.

[22] R. Merkle and M. Hellman. Hiding information and signature in trapdoor knapsacks. IEEE Transactions in Information Theory IT 24, pages 525 a 530, 1978.

[23] A. May and J.H. Silverman. *Dimension Reduction Methods for Convolution Modular Lattices*. In Proceeding of CaCL '01, LNCS, vol. 2146, Springer-Verlag, pp. 110-125, 2001.

[24] M. Nevins, C. KarimianPour and A. Miri. *NTRU over rings beyond Z*, Designs, Codes and Cryptography, August 2009. DOI : 10.1007/s10623-009-9342-7

[25] P.Q. Nguyen and J. Stern. *The Two Faces of Lattices in Cryptology*. In Proceeding of CaCL'01, LNCS, vol. 2146, Springer-Verlag, pp. 148-180, 2001.

[26] A. Nitaj, *NTRU et ses variantes, sécurité et applications.* Workshop C2M, Codes, Cryptologie et leurs Mathématiques.

[27] NTRU Inc. http ://www.ntru.com/

[28] NTRU Cryptosystems website. http ://www.ntru.com.

[29] R. L. Rivest, A. Shamir, and L. M. Adleman. *A method for obtaining digital signatures and public-key cryptosystems*. Communications of the ACM, 21(2) :120-126, 1978.

[30] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digitalsignatures and public key cryptosystems. commun. ACM 21, pages 120 à 126, 1978.

[31] V. Shoup. NTL : A Library for doing Number Theory. http ://www.shoup.net/ntl/.

[32] N. Vats. *Algebraic Cryptanalysis of CTRU Cryptosystem*, COCOON 2008, Dalian, China, Springer, 2008.

Annexe

Dans cet annexe nous avons implementé avec C/C++ et NTL tous les algorithmes de l'arithmétique des entiers Duaux et les algorithmes du cryptosystème NTRU sur les entiers Duaux. Ces implementations n'ont jamais été faites auparavant.

```
#include < NTL/ZZX.h >
#include < NTL/ZZ.h >
#include¡NTL/vec_ZZ.h¿
#include < NTL/ZZ_pX.h >
#include < NTL/ZZ_p.h >
#include < iostream >
#include < time.h >
#define maxpol 30
NTL_CLIENT
using namespace NTL;
using namespace std;

class duaux{
        public :
            ZZ x, y;
        public :
            duaux(ZZ reel=to_ZZ(0), ZZ imag=to_ZZ(0)){x=reel; y=imag;} /*constructeur*/
        friend duaux operator - (duaux, duaux);
        friend duaux operator + (duaux, duaux);
        friend duaux operator * (duaux, duaux);
        duaux conj(){duaux d; d.x = x; d.y = -y;
                return d;
                 };
        ZZ getX(){return x;}
        ZZ getY(){return y;}
         void PseudDiv(duaux &q, duaux &r, duaux &d2, duaux &d1);
         void affiche(){cout << "dual : " << "(" << x << ", " << y <<")" << endl;}
};

class polynome{
```

```
        int degre;
        duaux coef[maxpol];
    public :
        int getDegre(){return degre;}
         duaux saisi(void);
        polynome saisipoly(int);
         polynome add(polynome p, polynome g);
        void affiche(polynome p);
        ZZ_pX PartiReel(polynome &p);
        ZZ_pX PartiImag(polynome &p);
        polynome reconst(const ZZX &a, const ZZX &b);
        polynome invpoly(polynome &f, polynome &g);
        polynome invpolymodp(polynome &f);
};
//dual UN={1,0},ZERO={0,0};

/* ************************************************************
Redefinition de l'addition
************************************************************/
duaux operator + (duaux a, duaux b){
     duaux d;
    d.x = a.x+b.x; d.y = a.y+b.y;
    return d;
  }

/* ************************************************************
Redefinition de la multiplication
************************************************************/
duaux operator * (duaux a, duaux b)
{
     duaux d1;
    d1.x = a.x*b.x; d1.y = a.x*b.y+a.y*b.x;
   return d1;
}

/* ************************************************************
Redefinition de la soustracion
************************************************************/
duaux operator - (duaux a, duaux b){
    duaux d;        d.x = a.x-b.x; d.y = a.y-b.y;       return d;
}
```

```
Definition de la fonction de Pseudo-division
**********************************************************/
void PseudDiv(duaux &q, duaux &r, duaux &d2, duaux &d1) {
        ZZ n;
        n = (d1*d1.conj()).getX();
        duaux y1 = d2*d1.conj();
        ZZ a = y1.getX();
        ZZ b = y1.getY();
        ZZ u, u1;
            u1 = a;
            u = to_ZZ(0);
        DivRem(u, u1, a, n); //a = un + u1
        if(2 * u1 > n) {
            u1-= n;
            u++;
        }
        ZZ v, v1;
            v1 = b;
            v = to_ZZ(0);
         DivRem(v, v1, b, n); //b = vn + v1
        if(2 * v1 > n)
            v1-= n;
            v++;
        }
         q = duaux(u, v);
        r = d2-q*d1;
}

/* ***********************************************************
Fonction de calcul modulo des nombres duaux
************************************************************/
duaux mod(duaux d, duaux z){
        duaux q, r;
        PseudDiv(q, r, d, z);
     return r;
}

/* ***********************************************************
Fonction de calcul de la multication modulo des nombres duaux
************************************************************/
duaux multmod(duaux d1, duaux d2, duaux z){
```

```
        d3 = d1 * d2;
        r = mod(d3, z);
       return r;
}

/* ************************************************************
Fonction de calcul de l'addition modulo des nombres duaux
************************************************************/
duaux addmod(duaux d1, duaux d2, duaux z){
        duaux d3, q, r;
         d3 = d1 + d2;
        r = mod(d3, z);
     return r;
}

/* ************************************************************
Fonction de calcul de la soustraction modulo des nombres duaux
************************************************************/
duaux submod(duaux d1, duaux d2, duaux z){
         duaux d3, q, r;
        d3 = d1 - d2;
        r = mod(d3, z);
    return r;
}


/*****************************************************************
Fonction de calcul de l'inverse d'un nombre dual
*****************************************************************/
duaux InvDuaux(duaux &a, duaux &z){
        duaux b, k;
        ZZ dd, ds, dt, a1, z1, b1, b2, a2, z2, w, k1, k2;
         a1 = a.getX(); a2 = a.getY(); z1 = z.getX(); z2 = z.getY();
        XGCD(dd, ds, dt, a1, -z1);
         b1 = ds;
        k1 = dt;
        w = k1*z2-a2*b1;
        b2 = b1*w; k2 = k1*w;
        b = duaux(b1, b2);
        k = duaux(k1, k2);
      return b;
```

```
/*****************************************************************
Fonction qui permet de saisir un nombre dual
*****************************************************************/
duaux saisi(void){
        duaux d;
        cout << "r= ";
        cin >> d.x;
        cout << "i= ";
        cin >> d.y;
    return d;
}

/* ****************************************************************
Fonction qui initialise un polynome de degre n à 0
*****************************************************************/
polynome init(int n){
        polynome p;
        int i;
        p.degre = n;
        for(i=0;i<=n+1;i++){
                p.coef[i].x=0;
                p.coef[i].y=0;
         }
        return p;
}

/* ****************************************************************
Fonction qui permet de saisir un polynome de degre n
*****************************************************************/
polynome saisipoly(int n){
        polynome p;
        int i;
        duaux a;
        p.degre = n;
        for(i = 0; i< n; i++){
                cout << "Coefficient d ordre "<< i << " ? " << endl;
                a = saisi();
                p.coef[i] = a;
        }
cout << endl;
```

```
}

/*****************************************************************
Fonction de calcul du polynome modulo un nombre dual
*****************************************************************/
polynome polmod(polynome &p, duaux &d){
        int i, degre;
        degre = p.degre;
         for(i=0; i <= degre; i++)
                p.coef[i] = mod(p.coef[i], d);
       return p;
}

/*****************************************************************
Fonction qui calcul le produit de deux polynomes
*****************************************************************/
polynome prodpol( polynome &g, polynome &h, int n){
        polynome p, pr;
        int i, j;
        pr.degre = n;
        p = init(g.degre + h.degre);
        for (i = 0; i < g.degre; i++)
                for (j = 0; j < h.degre; j++)
                        p.coef[i+j] = p.coef[i+j]+g.coef[i]*h.coef[j];
         pr = init(n);
        for(i = 0; i < g.degre + h.degre-1; i++)
                pr.coef[i%n] = pr.coef[i%n]+p.coef[i];
      return pr;
}

/*****************************************************************
Fonction qui calcul l'addition deux deux polynomes dont les coefficiants sont des entiers duaux
*****************************************************************/
polynome add(polynome p, polynome g){
        polynome h;
        int maxdegre = g.degre;
        if(p.degre > maxdegre)
                maxdegre = p.degre;
        h = init(maxdegre);
        for (int i=0;i<maxdegre;i++)
                h.coef[i] = g.coef[i] + p.coef[i];
```

```
}

void affiche(polynome p){
        int i, degre;
        degre = p.getDegre();
        for(i = 0; i < degre; i++)
                cout << "(" << p.coef[i].x << "+" << p.coef[i].y << ")"<< "x" <<i<<"+";
cout << endl;
}

ZZ_pX PartiReel(polynome &pf){
        ZZX pdr;
        ZZ_pX pmod;
        int i;
        for(i =0; i <= pf.degre; i++){
                SetCoeff(pdr, i, pf.coef[i].x);
        }
        conv(pmod, pdr);
        cout << pmod << endl;
    return pmod;
}
ZZ_pX PartiImag(polynome &p){
        ZZX pdi;
        ZZ_pX pmod;
        int j;
        long dp = deg(pdi);
        for(j =0; j <= p.degre; j++){
                SetCoeff(pdi, j, p.coef[j].y);
        }
        conv(pmod, pdi); cout << pmod << endl;
      return pmod;

/*****************************************************************
Fonction de reconstitution d'un polynome à  coefficients avec des entiers duaux
*****************************************************************/
polynome reconst(const ZZX &a, const ZZX &b){
        int j;
        polynome p;
        for(j = 0; j¡= p.degre; j++){
                p.coef[j].x = a.rep[j];
                p.coef[j].y = b.rep[j];
```

```
    return p;
}

/*****************************************************************
FONCTION DE CALCUL MODULO UN ENTIER
*****************************************************************/
ZZX modpolint(ZZX &p, int n){
      int i;
      for(i = 0; i<deg(p)+1; i++)
            p.rep[i] = p.rep[i]%n;
    return p; }

/*****************************************************************************
FONCTION DE CALCUL INVERSE POLYNOME DANS Z/nZ
*****************************************************************************/
ZZX invpolint(ZZX &p, int n){
      ZZ r;
      int tab[5] = -1, 0, 0, 0, 1;
      int i;
      ZZX g, d, s, t;
      for(i = 0; i < 5; i++)
            SetCoeff(g, i, tab[i]);
      XGCD(r, s, t, p, g);
      if(r%n == 1){
            s = modpolint(s, n);
      }
      else cout << "Il ne pas Inversible" << endl;
    return s;
}

/*****************************************************************
Fonction de calcul de l'inverse d'un polynôme modulo un nombre dual p
*****************************************************************/
polynome invpolymodp(polynome &f){
      int i;
      polynome h, k;
      ZZ_pX pd, x, ps, pt, f1, f2, g, h1, h2, k1, k2, w;
      ZZX h11, h22, k11, k22;
       int tab[12] = -1, 0, 0, 0, 0, 0, 0, 1;
      g.rep.SetLength(8);
      for(i = 0; i < 8; i++)
```

```
        h = init(f.degre);
        f1 = PartiReel(f); f2 = PartiImag(f); h1 = PartiReel(h); h2 = PartiImag(h);
        XGCD(pd, ps, pt, f1, g);
        if(pd==1){
                h1 = ps;
                k1 = pt;
                x=MulMod(h1, f2, g);
                mul(w, x, -1);
                h2=MulMod(h1, w, g);
                k2=MulMod(k1, w, g);
        }
         else
                cout << "Non inversible " << endl;
        conv(h11, h1); conv(h22, h2); conv(k11, k1); conv(k22, k2);
        h = init(f.degre);
        for(i = 0; i<=deg(h1); i++){
                h.coef[i].x = h11.rep[i];
        }
        for(i = 0; i<= deg(h2); i++){
                h.coef[i].y = h22.rep[i];
        }
        cout <<"k1= "<< k1 << " k2= " << k2 << endl;
         cout << pd<< " "<< h1 << " " << h2 << endl;
         return h;
}

/*************************************************************
multiplication d'un polynome avec un entier
*************************************************************/
polynome multent(polynome p, ZZ n){
        int i;
        for(i=0; i < p.degre; i++){
                p.coef[i].x*=n;
                p.coef[i].y*=n;
        }
        return p;
}

/*************************************************************
Generation de la clé public
*************************************************************/
```

```
    polynome h, r, t ;
    r=multent(p, n) ;
    t = prodpol(r, q, 7) ;
    h = polmod(t, d) ;
  return h ;
}
```