L'appel à standardisation du NIST pour des primitives cryptographiques post-quantiques : Regard sur le projet crystals

Michel Nguyen
Etudiant en première année de master de mathématiques

Sous la direction de Julien Lavauzelle

Mai 2021

Sommaire

Avant-propos

Ce dossier a été rédigé avec trois mots à l'esprit : découverte, accessibilité et invitation. Découverte car le champ de la cryptographie post-quantique est récent et est en plein expansion. D'où la volonté d'utiliser énormément de sources fiables pour alimenter en informations le dossier. Accessibilité car le dossier cherche à pouvoir être lu par le plus grand nombre de personnes. Lorsque l'on traitera des parties avec des notions mathématiques, on essaiera de les rendre les plus compréhensibles par tous. Invitation car le dossier se veut comme un passeur de relais afin d'encourager le lecteur à poursuivre sa lecture vers d'autres articles scientifiques. Les enjeux autour de la cryptographie post-quantique sont multiples et bien présents, c'est pourquoi il est utile d'avoir un aperçu des avancées actuelles.

"In the history of computer science, however, most important problems have turned out to be either polynomial-time or NP-complete. Thus quantum computers will likely not become widely useful unless they can solve NP-complete problems. Solving NP-complete problems efficiently is a Holy Grail of theoretical computer science which very few people expect to be possible on a classical computer. Finding polynomial-time algorithms for solving these problems on a quantum computer would be a momentous discovery. There are some weak indications that quantum computers are not powerful enough to solve NP-complete problems [Bennett et al. 1994], but I do not believe that this potentiality should be ruled out as yet."

Peter Shor. 1994.

Introduction

En 2017, le *National institute of standards and technology* (NIST) lance la compétition internationale *Post-Quantum Cryptography Standardization* (PQC) dans le but de changer les standards cryptographiques en vue de la nouvelle technologie quantique [25]. Différentes équipes de chercheurs du monde entier ont proposé au total 69 candidats où l'on retrouve des propositions de chiffrement/d'encapsulation de clé et de signature numérique. Actuellement, la compétition continue et prévoit de durer jusqu'aux environs de 2024. Depuis le lancement, trois phases se sont déroulées où le NIST a sélectionné les candidats les plus performants. Aujourd'hui, seuls 7 candidats (3 chiffrements/encapsulations de clé et 4 signatures numériques) sont en compétition [33].

Cette accélération de la recherche scientifique devrait nous interroger. En effet, quelles sont les raisons qui poussent les chercheurs à proposer des primitives résistantes au modèle de calcul quantique alors qu'aucune machine quantique n'est exploitable à l'heure actuelle ? En quoi un ordinateur quantique est-il plus performant qu'un ordinateur classique ? Pourquoi une telle accélération de la recherche scientifique ? Y-a-t-il d'ores et déjà une utilisation industrielle de primitives post-quantiques ?

C'est pourquoi dans ce dossier, nous allons détailler dans un premier temps le contexte, c'est-à-dire qui sont les acteurs et quelles sont les motivations qui animent cette recherche. Dans un second temps, nous allons faire un premier pas vers la cryptographie post-quantique et les domaines mathématiques exploités au sein de la compétition du NIST. Dans un troisième temps, nous allons nous intéresser de près aux réseaux euclidiens qui sont sujets à des problèmes difficiles dans un modèle de calcul quantique. En particulier, le problème *learning with errors* (LWE). Par la suite, nous regarderons deux candidats toujours en lice dans la compétition du NIST : Crystals-Kyber, [8] et Crystals-Dilithium [16]. Et pour terminer, nous proposerons quelques implémentations de schémas de chiffrement basés sur le problème LWE.

Chapitre 1

Contexte

Depuis deux millénaires, de nombreux systèmes de chiffrements ont été utilisés pour assurer l'inintelligibilité des messages. Du temps de Jules César, le chiffrement de César consistait en un décalage de lettres. Aujourd'hui, ce chiffrement est cassé dans le sens où on peut retrouver le message initial à partir d'un chiffré à l'aide d'un ordinateur en un temps raisonnable (par une analyse fréquentielle par exemple). Une leçon à tirer est que la sécurité d'un chiffrement n'est jamais garantie avec le temps. En effet, de nouvelles techniques algorithmiques et mathématiques se développent continuellement afin de casser les chiffrements utilisés. Avec l'apparition des ordinateurs, de nouveaux algorithmes de chiffrement ont pu être standardisés grâce à l'automatisation des calculs et de la puissance des ordinateurs.

Le XXIe siècle se distingue des autres par la numérisation massive des données. Ces données passent par des canaux d'information comme lorsqu'on communique par Internet ou lorsqu'on fait un achat en ligne. Pendant l'échange d'informations, les messages sont chiffrés dans le but d'assurer leur confidentialité, leur intégrité et leur authenticité. C'est pourquoi on cherche en permanence à utiliser des primitives cryptographiques sûres en admettant qu'il existe des adversaires équipés en algorithmes et en technologies. Donc, la recherche autour de la cryptographie est toujours rythmée par la volonté de créer des primitives suffisamment sûres et de corriger les failles de ces primitives.

Deux approches co-existent en cryptographie : l'une à clé privée (nommée symétrique), l'autre à clé publique (nommée asymétrique). Les deux ont leurs propres avantages et inconvénients. Au final, elles se complètent l'un à l'autre. C'est pourquoi, il existe des chiffrements hybrides utilisant les deux.

La cryptographie à clef publique exploite des problèmes dits "difficiles" dans le sens où les fonctions utilisées sont à sens unique, c'est-à-dire qu'il est simple de mener une certaine opération mais difficile de l'inverser. Par recherche exhaustive, on parle de milliers d'années avant de retrouver la clé privée initiale. Deux problèmes difficiles sont majoritairement utilisés actuellement : la factorisation et le logarithme discret. Le premier consiste en la difficulté de retrouver deux entiers premiers p et q à partir de n tel que $n = pq$. L'autre repose sur la difficulté d'extraire a à partir de g^a modulo n'. Ces problèmes sont difficiles en respectant certaines conditions dont le fait que n et n' sont suffisamment grands.

En 1994, le cryptologue américain Shor publie un algorithme capable de casser plusieurs problèmes mathématiques réputés difficiles [41]. En effet, cet algorithme résout ces problèmes en un temps raisonnable (plus précisément polynomial). C'est le cas avec le problème de la factorisation, mais aussi avec le problème du logarithme discret. Cela dit,

cet algorithme ne peut être exécuté qu'à partir d'un modèle de calcul quantique. Ceci chamboule les fondations sur lesquelles s'est construite la cryptographie à clef publique moderne. Par conséquent, la signature numérique est impactée, car elle aussi utilise des problèmes difficiles et dans une moindre mesure, la cryptographie symétrique.

L'existence future d'un ordinateur quantique constitue une menace, bien évidemment pour nos communications actuelles mais aussi pour les communications passées qui ont été chiffrées à l'aide de primitives bientôt caduques. C'est un danger encore plus grave pour des données "sensibles" comme la communication qu'utilise un gouvernement quelconque.

Dès lors, il est utile voire nécessaire de prévoir la situation où les problèmes jugés difficiles aujourd'hui seront cassés par des adversaires équipés en ordinateurs quantiques.

C'est pourquoi en 2017, le NIST décide de réagir en lançant la compétition mondiale *Post-Quantum Cryptography Standardization* afin d'inviter la communauté scientifique à créer des primitives résistantes au modèle de calcul quantique [36]. Plus précisément, les trois primitives suivantes : le chiffrement, l'encapsulation de clé et la signature numérique. Le NIST a un statut et une aura particuliers dans la communauté scientifique car le NIST a déjà organisé auparavant des compétitions mondiales qui ont porté leurs fruits puisque les résultats ont été standardisés. C'est le cas en 1997 lors de la compétition *Advanced Encryption Standard process* dans le but de remplacer le chiffrement symétrique Triple DES bientôt caduc à son époque [32]. Mais aussi en 2008, lors de la compétition *hash function competition* ayant pour objectif de remplacer les fonctions de hachage SHA-1 et SHA-2. Ceci a aboutit à SHA-3 en 2012 qui est toujours utilisé à l'heure actuelle [34].

La communauté scientifique réagit positivement à ce nouvel appel du NIST et propose ainsi 69 propositions, voir figure 1.1.

BIG QUAKE	Giophantus	LOCKER	QC-MDPC-KEM
BIKE	Gravity-SPHINCS	LOTUS	qTESLA
CFPKM	Guess Again	LUOV	RaCoSS
Classic McEliece	Gui	McNie	Rainbow
Compact LWE	HILA5	Mersenne-756839	Ramstake
CRYSTALS-DILITHIUM	HiMQ-3	MQDSS	RankSign
CRYSTALS-KYBER	HK-17	NewHope	RLCE-KEM
DAGS	HQC	NTRUEncrypt	Round2
Ding Key Exchange	KCL	NTRU-HRSS-KEM	RQC
DME	KINDI	NTRU Prime	RVB
DRS	LAC	NTS-KEM	SABER
DualModeMS	LAKE	Odd Manhattan	SIKE
Edon-K	LEDAkem	Ouroboros-R	SPHINCS+
EMBLEM/R.EMBLEM	LEDApkc	Picnic	SRTPI
FALCON	Lepton	Post-quantum RSA Encryption	Three Bears
FrodoKEM	LIMA	Post-quantum RSA Signature	Titanium
GeMSS	Lizard	pqNTRUSign	WalnutDSA
		pqsigRM	

Figure 1.1: Les 69 candidats lors du premier tour [39]

5 candidats ont rapidement été exclus pour cause de similitude ou de faiblesse de sécurité. Parmi les 64 autres candidats, on dénombre 45 chiffrements/encapsulations de clé (ou KEM pour *key encapsulation mechanism*) et 19 signatures numériques.

En janvier 2019, 26 candidats ont été retenus pour le deuxième tour, voir figure 1.2.

Ils ont été sélectionnés selon des critères :

Figure 1.2: Les 26 candidats lors du second tour [39]

- de sécurité (preuves de sécurité valides, résistance à toutes les attaques, complexité algorithmique dans le modèle de calcul classique et quantique) ;
- de performance (taille des paramètres, temps d'exécution des algorithmes, validité du déchiffrement) ;
- de caractéristiques d'implémentation (avantages et désavantages, simplicité de la documentation, flexibilité et simplicité de l'implémentation).

Le critère de la sécurité se divise en plusieurs niveaux et s'applique dans un modèle de calcul classique et quantique, voir tableau 1.1.

Niveau	Description de la sécurité
I	Au moins aussi dur à casser que AES128 (recherche exhaustive de la clé)
II	Au moins aussi dur à casser que SHA256 (recherche de la collision)
III	Au moins aussi dur à casser que AES192 (recherche exhaustive de la clé)
IV	Au moins aussi dur à casser que SHA384 (recherche de la collision)
V	Au moins aussi dur à casser que AES256 (recherche exhaustive de la clé)

Tableau 1.1: Les niveaux de sécurité [39]

Le niveau I, II et III doivent être absolument respectés pour avoir une chance d'être sélectionné. Si une primitive parvient à respecter le niveau IV et V, alors elle possède une très forte sécurité. D'autres critères sont présents tels que :

- ***Drop-in replacements* :** la capacité à pouvoir changer de matériel informatique sans modification du code ;
- **La confidentialité persistante :** la découverte de la clé privée par un adversaire ne compromet pas la confidentialité des communications passées ;
- **La résistance aux attaques par canal auxiliaire :** attaques utilisant des propriétés physiques, qui recherchent et exploitent des failles logicielles ou matérielles.

En juillet 2020, le NIST annonce le troisième tour ainsi que les 7 finalistes sélectionnés, voir tableau 1.2.

	Finalistes	Domaines mathématiques	Commentaires du NIST
Chiffrement/KEM	Kyber	réseaux euclidiens	remplit tous les critères
Chiffrement/KEM	NTRU	réseaux euclidiens	pas aussi efficace que prévu
Chiffrement/KEM	SABER	réseaux euclidiens	remplit tous les critères
Chiffrement/KEM	Classic McEliece	codes correcteurs	grande taille de clés et petite taille des chiffrés
Signature numérique	Dilithium	réseaux euclidiens	équilibré ; problème coreSVP améliorable ?
Signature numérique	Falcon	réseaux euclidiens	équilibré ; problème coreSVP améliorable ?
Signature numérique	Rainbow	syst. poly. multivariés	grande taille de clés publiques ; petites signatures

Tableau 1.2: Les 7 finalistes lors du troisième tour [39]

Il existe 8 candidats alternatifs qui méritent une attention particulière, voir tableau 1.3.

	Finalistes	Domaines mathématiques	Commentaires du NIST
Chiffrement/KEM	BIKE	codes correcteurs	bonne performance mais n'est pas assez stable
Chiffrement/KEM	FrodoKEM	réseaux euclidiens	figure d'alternative parmi les réseaux euclidiens
Chiffrement/KEM	HQC	codes correcteurs	meilleure analyse de la sécurité que BIKE
Chiffrement/KEM	NTRUprime	réseaux euclidiens	conception et sécurité différente de NTRU
Chiffrement/KEM	SIKE	graphes d'isogénies	problème de sécurité et lenteur
Signature numérique	GeMSS	syst. pol. multivariés	grande taille de clés publiques ; petites signatures
Signature numérique	Picnic	crypt. symétrique	n'est pas assez stable mais a du potentiel
Signature numérique	Sphincs+	crypt. symétrique	stable

Tableau 1.3: Les 8 candidats alternatifs lors du troisième tour [39]

Cette compétition est particulière car elle ne cherche pas à désigner un gagnant, mais à retenir plusieurs candidats pour réunir un ensemble de primitives résistantes au modèle de calcul quantique dans le cas où l'une d'entre elles s'avère défectueuse.

"Notre intention est de sélectionner plusieurs options pour plus de standardisation, tout comme éliminer certains candidats moins adaptés... L'objectif de la compétition n'est pas de choisir un gagnant, mais établir les forces et les faiblesses des différentes options, et d'analyser les compromis parmi eux."

Dustin Moody, cryptographe américain du NIST. Pendant la présentation de la compétition PQC en 2016 [35].

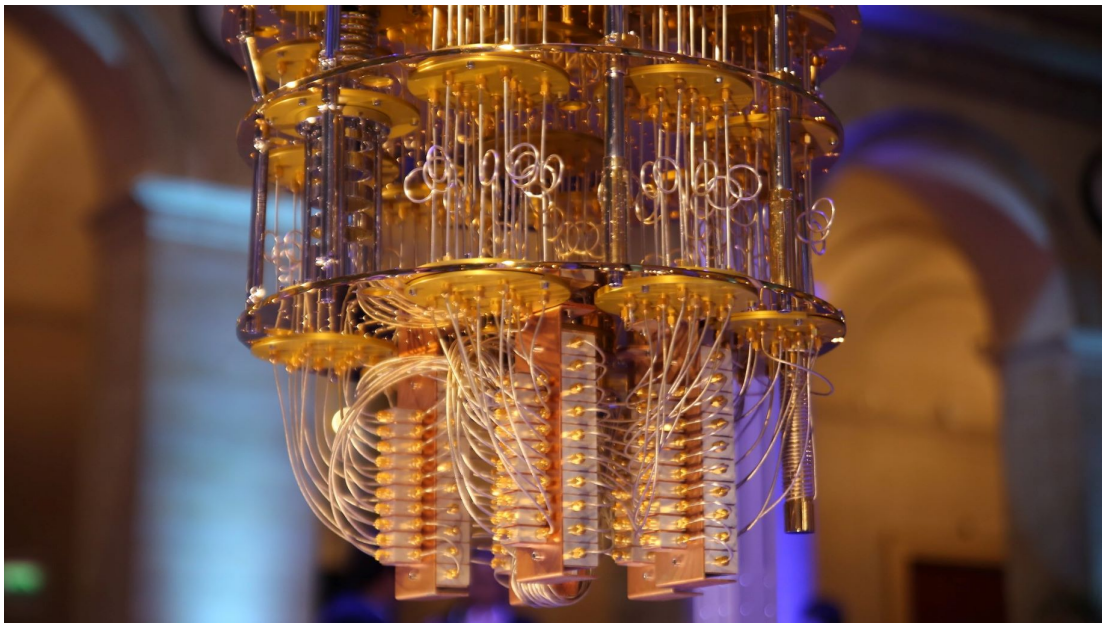
Il est prévu que cette compétition dure jusqu'en 2022 pour une nouvelle standardisation des primitives cryptographiques à l'issue des résultats. Puis, en 2024, une mise en production de ces primitives à destination des industriels sera en vigueur [35]. Au final, cette compétition est un projet d'anticipation pour la lutte contre les potentiels ordinateurs quantiques à l'horizon 2030.

Parmi les 7 candidats, certains sont pressentis pour être standardisés. Il s'agit d'un des trois parmi les chiffrements/KEM : Kyber, NTRU ou bien SABER. Concernant les signatures numériques, il s'agit de Dilithium ou bien Falcon. Il est intéressant à noter que toutes ces primitives sont basés sur des problèmes de réseaux euclidiens car la performance de ceux-ci est prometteuse, voir la source [37] à partir du diapositive 24 et la source [38] à partir du diapositive 15.

Dans notre dossier, on se concentrera sur les candidats Kyber et Dilithium.

Chapitre 2

La cryptographie post-quantique



Un ordinateur quantique [26]

2.1 Un modèle de calcul quantique

Les ordinateurs actuels s'inscrivent dans un modèle de calcul dit "classique" où les opérations sont effectuées sur des séquences de bits (éléments de $\{0,1\}^n$) avec des opérations binaires. Les ordinateurs quantiques se basent sur des propriétés physiques différentes. En effet, ils utilisent des qbits avec des opérations logiques quantiques. Ces qbits sont une superposition quantique de $\{0,1\}$. Ainsi, les ordinateurs quantiques ont le potentiel de réaliser des opérations beaucoup plus efficacement que les ordinateurs classiques [25].

A l'heure actuelle, les ordinateurs quantiques sont imparfaits et loin d'être exploitables. En effet, les meilleurs processeurs quantiques possèdent moins de 100 qbits (l'unité de mémoire de l'information quantique), ce qui est loin de représenter une certaine menace vis-à-vis des chiffrements actuels.

Cela dit, l'évolution de cette technologie progresse à vive allure. Pour illustrer cela, concernant le problème de la factorisation, en 2012 l'ordinateur quantique est capable de

factoriser 21. En 2016, il est capable de factoriser 200 099. En 2019, il est capable de factoriser 1 099 551 473 989 [31].

De plus, concernant le chiffrement RSA, un des chiffrements les plus utilisés actuellement, le co-fondateur de *Institute for Quantum Computing* de l'université de Waterloo, Michele Mosca déclare :

> *"J'estime à 1 chance sur 7 la probabilité de casser RSA-2048 en 2026 et 1 chance sur 2 en 2031"*
>
> Michele Mosca. *Cybersecurity in an era with quantum computers: will we be ready? [30]*

en faisant référence à l'ordinateur quantique. Propos appuyé par le NIST qui table vers l'horizon 2030-2040 pour la création des premiers ordinateurs quantiques exploitables [35].

Ayant connaissance de ces informations, il est clair qu'il est déjà bon d'anticiper cette ère où des ordinateurs seront capables de casser des primitives cryptographiques utilisées. Donc, bien que l'ordinateur quantique n'est pas voué à être vendu massivement dans le commerce, changer les standards permet une protection vis-à-vis de cette potentielle nouvelle menace.

Pour récapituler l'ensemble, nous possédons les algorithmes pour casser des problèmes difficiles mais nous ne possédons pas encore le matériel pour les exécuter.

2.2 Les propositions retenues du NIST

Pour faire face aux ordinateurs quantiques, il peut être envisageable de conserver nos chiffrements actuels tout en augmentant la taille des clefs. Cette possibilité ajoute de la sécurité supplémentaire mais elle pose un défaut : elle alourdit le temps d'exécution de l'algorithme de chiffrement. A titre d'exemple, une proposition d'une modernisation du chiffrement RSA dans le cadre d'un modèle de calcul quantique fait surface en 2017. Cette proposition consiste à générer des clés publiques d'une taille de 1 teraoctet, soit 1000 gigaoctets. [7].

C'est pourquoi il est préférable de s'orienter vers d'autres problèmes mathématiques que la factorisation et le logarithme discret dans le contexte d'un modèle de calcul quantique.

Lors de la compétition du NIST, la plupart des candidats les plus sérieux peuvent être classés selon quatre grands domaines : les réseaux euclidiens, les graphes d'isogénies, les systèmes polynomiaux multivariés et les codes correcteurs. Tous ces problèmes sont classés NP-difficiles. Cela implique qu'il n'existe, à l'heure actuelle, aucun algorithme capable de résoudre ces problèmes dans un modèle de calcul classique comme quantique.

2.2.1 Définitions mathématiques

Pour mieux comprendre les quatre domaines, il nous faut d'abord établir des définitions et des notations des objets mathématiques que nous allons aborder. On suppose comme acquise la définition de groupe abélien, sous-groupe, anneau, polynôme, variable aléatoire, loi de distribution, schéma de chiffrement et schéma de signature.

Définition 1. Corps.
Un corps est un triplet $(K, +, \cdot)$, qu'on note K, formé d'un ensemble et de deux lois de composition $K \times K \to K$, l'une est l'addition $+$, l'autre est la multiplication $\cdot$. Ce triplet est un anneau non nul tel que tout élément non nul possède un inverse pour la multiplication. Autrement dit, pour tout $x \neq 0$ de K, il existe y tel que $x \cdot y = 1$. Cet élément est noté x^{-1}.

Définition 2. Anneau des polynômes.
L'ensemble des polynômes à coefficients dans l'anneau A forme l'anneau $A[X]$, respectant ainsi les propriétés d'un anneau. Par exemple, $\mathbb{Z}[X]$ représente l'ensemble des polynômes à coefficients dans $\mathbb{Z}$.

Définition 3. Matrice.
Une matrice à coefficients dans un ensemble donné E de $n \geq 1$ lignes et $p \geq 1$ colonnes est un tableau rectangulaire de np nombres appartenant à E $(a_{i,j})_{1 \leq i \leq n, 1 \leq j \leq p}$:

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,p} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,p} \end{pmatrix}$$

Le terme général $a_{i,j}$ de cette matrice est situé à l'intersection de la i-ème ligne et de la j-ième colonne. Une matrice à n lignes et p colonnes est dite de type (n, p).

Définition 4. Produit scalaire.
Soient $u = (u_1, ..., u_m)$ et $v = (v_1, ..., v_m)$ deux vecteurs de $\mathbb{R}^m$. Le produit scalaire de u et v est

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

Définition 5. Norme euclidienne.
Soit $u = (u_1, ..., u_m)$ un vecteur de $\mathbb{R}^m$. La norme euclidienne de u est

$$||u||_2 = \sqrt{\langle u, u \rangle} = \sqrt{\sum_{i=1}^{m} u_i^2}$$

Définition 6. Norme "infinie".
Soit $u = (u_1, ..., u_m)$ un vecteur de $\mathbb{R}^m$. La norme "infinie" de u est

$$||u||_\infty = \max(|u_1|, \ldots, |u_m|)$$

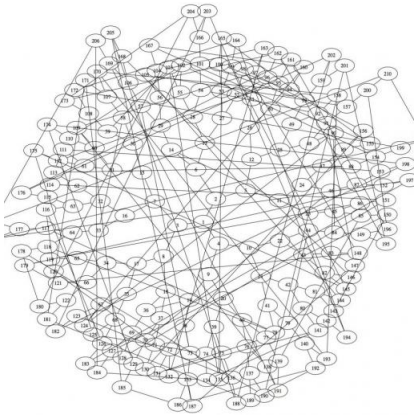
Définition 7. Loi Gaussienne
Une loi gaussienne, ou loi normale, est une loi caractérisée par sa moyenne μ et sa variance σ^2 notée $\mathcal{N}(\mu, \sigma^2)$. Une variable aléatoire X suivant une loi gaussienne a pour densité de probabilité :

$$\forall x \in \mathbb{R}, \quad p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Ce qui caractérise la loi gaussienne, c'est la forme de sa courbe qui est celle d'une cloche. Dans le cas d'une loi gaussienne discrète, x prend valeur dans $\mathbb{Z}$.

Définition 8. Code.
Soit A un ensemble, dit alphabet et l'encodeur qui transforme une séquence en une séquence de n éléments $x := (x_1, \ldots, x_n) \in A^n$, avec $n \geq k$. L'ensemble des mots de code, c'est-à-dire l'ensemble de séquences qui sortent de l'encodeur, s'appelle justement code. Un code de longueur n sur A est un sous-ensemble de A^n.

2.2.2 Les graphes d'isogénies

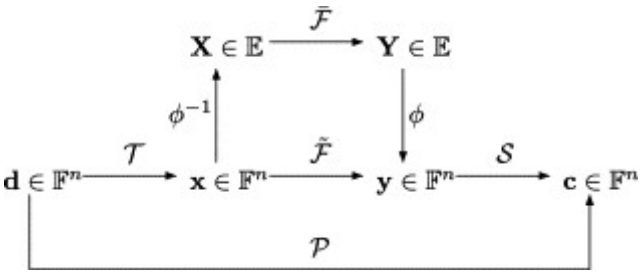


Les graphes d'isogénies

Soit une courbe elliptique qui est l'ensemble des points respectant une équation de Weierstrass définie sur un corps telle que $y^2 = x^3 + ax + b$ tandis qu'une isogénie est un morphisme surjectif et de noyau fini entre des courbes elliptiques. De plus, soient un graphe où les sommets sont des classes d'isomorphismes de courbes elliptiques isogène et les arêtes des isogénies de degrés fixés. Chaque sommet possède le même nombre de voisins. Un problème difficile dans ce domaine est la difficulté de trouver un chemin court qui relie certaines paires de sommets.

Au sein de la compétition PQC, seul le candidat alternatif SIKE exploite ce domaine [27].

2.2.3 Les systèmes polynomiaux multivariés



Les systèmes polynomiaux multivariés

Soit un système de plusieurs équations polynomiales de degré ≥ 2 à plusieurs variables. Dans la cryptographie multivariée, il existe le problème difficile MQ (*multivariate quadratic equations*) qui repose sur la difficulté de calculer des solutions dans ce système.

Le problème MQ
Soit $\mathbb{F}_q$ un corps fini, et deux entiers $m, n \geq 1$. On a un système de m équations quadratiques à n inconnues $x_1, x_2, \ldots, x_n$ tel que :

$$\begin{cases} f_1(x_1, \ldots, x_n) = u_1 \\ \quad \vdots \quad \vdots \\ f_m(x_1, \ldots, x_n) = u_m \end{cases}$$

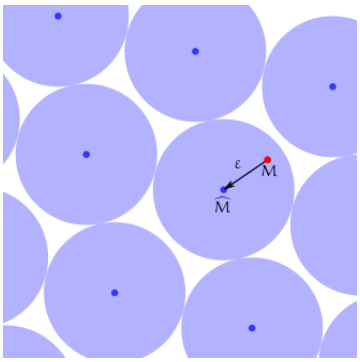
où

$$f_k(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j} x_i x_j + \sum_{i=1}^{n} b_i x_i + c$$

Le problème repose sur la difficulté de trouver une solution du système $s = (s_1, \ldots, s_n) \in \mathbb{F}_q^n$.

Ce domaine est peu représenté parmi les candidats du PQC. En effet, seuls les candidats Rainbow et GeMSS l'exploitent [15, 12].

2.2.4 Les codes correcteurs



Les codes correcteurs

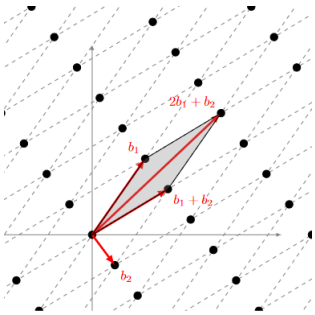
Les codes correcteurs d'erreurs peuvent être sujets à la création de primitives post-quantiques. En effet, ils sont intéressants car il est difficile de décoder un code $\mathcal{C}$ à partir uniquement d'un procédé d'encodage aléatoire.

Un système ancien (1978) mais toujours étudié est le système de McEliece où le choix de codes se porte sur la famille des codes de Goppa binaires.

Dans la compétition PQC, seuls deux candidats exploitent ce domaine : BIKE et HQC [4] [29].

Chapitre 3

Les réseaux euclidiens



Les réseaux euclidiens

Les réseaux euclidiens font partie des quatre grandes familles de propositions du NIST. Un réseau euclidien est un sous-groupe discret $\mathcal{L}$ de dimension n de $(\mathbb{R}^n, +)$. Ce sous-groupe se génère à partir d'une base qui est une famille de n vecteurs libres. Donc, tout élément du réseau euclidien est une combinaison linéaire de cette famille.

Par exemple, prenons les deux points $(2, 0)$ et $(0, 2)$ qui forment une base dans $\mathbb{R}^2$. Cela implique que tous les autres points sont de la forme $a_1(2, 0)+a_2(0, 2)$ avec $a_1, a_2 \in \mathbb{Z}$. Voici un schéma du réseau euclidien généré à partir de cette base :

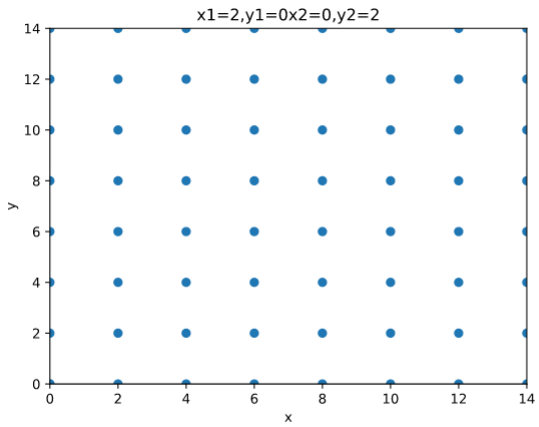


Figure 3.1: Un réseau euclidien à partir des vecteurs $(2, 0)$ et $(0, 2)$ dans $\mathbb{R}^2$ [11]

En généralisant, pour une base $B = \{v_1, \ldots, v_n\}$ de $\mathbb{R}^n$, alors le réseau $\mathcal{L}$ de base B

est l'ensemble :

$$\mathcal{L} = \left\{ \sum_{i=1}^{n} a_i v_i \mid \forall i \in \{1, \ldots, k\}, a_i \in \mathbb{Z} \right\}$$

On remarque que tous les points sont à intervalle régulier. De plus, on définit la distance minimale du réseau telle que $\lambda_1(\mathcal{L}) := \min_{v \in \mathcal{L} \setminus \{0\}} \|v\|_2$.

Les réseaux euclidiens sont un domaine de recherche très actif. Une de ses qualités est d'être le sujet de plusieurs types de primitives à hauts potentiels : chiffrement, chiffrement homomorphe [20], encapsulation de clé et signature numérique.

3.1 Les problèmes SVP, CVP et SIS

Il existe de nombreux problèmes difficiles dans les réseaux euclidiens. Les problèmes suivants sont classés NP-difficiles. On définit $\mathbb{Z}_q^n$ l'ensemble des n-uplets à coefficients dans les entiers relatifs modulo q.

Le problème du vecteur le plus court (SVP)
Posons une base B d'un réseau $\mathcal{L} \subseteq \mathbb{Z}_q^n$. Le problème repose sur la difficulté de trouver un vecteur $v \in \mathcal{L}$ tel que $||v||_2 = \lambda_1(\mathcal{L})$ lorsque la dimension est grande.

Le problème d'approximation du vecteur le plus court ($\mathbf{SVP}_\gamma$)
Posons une base B d'un réseau $\mathcal{L} \subseteq \mathbb{Z}_q^n$ et $\gamma(n) > 1$. Le problème repose sur la difficulté de trouver un vecteur $v \in \mathcal{L}$ tel que $||v||_2 \leq \gamma(n) \cdot \lambda_1(\mathcal{L})$ lorsque la dimension est grande.

Le problème du vecteur le plus proche (CVP)
Posons une base B d'un réseau $\mathcal{L} \subseteq \mathbb{Z}_q^n$ et un vecteur $x \in \mathbb{Z}_q^n$. Le problème repose sur la difficulté de trouver un vecteur $v \in \mathcal{L}$ tel que $||v - x||_2$ soit le plus petit possible.

Le problème de la "solution entière courte" (SIS)
Posons m vecteurs $a_1, \ldots, a_m \in \mathbb{Z}_q^n$. Le problème repose sur la difficulté de trouver $z_1, \ldots, z_m \in \{-1, 0, 1\}$ tels que $z_1 a_1 + ... + z_m a_m = 0$.

3.2 Le cryptosystème NTRU

Il existe un chiffrement basé sur le problème CVP nommé NTRU (*N-th degree Truncated polynomial Ring Units*). Ce chiffrement a été créé en 1998 par les mathématiciens américains Hoffstein, Pipher et Silvermann [22]. L'ensemble considéré par NTRU, c'est-à-dire l'anneau des polynômes, est intéressant à étudier car le chiffrement NTRU inspirera le problème ring-LWE et module-LWE lors de leurs conceptions qu'on traitera plus tard.

Soient les éléments d'un chiffrement NTRU qui sont des éléments de l'anneau des polynômes $\mathcal{R} = \mathbb{Z}_q[X]/(X^n - 1)$, où n, p, q sont les paramètres publics. À un polynôme $A(X) \in \mathbb{R}$, on associe le vecteur a ses coefficients $a \in \mathbb{Z}^n$. On définit $c = a \star b$ comme le vecteur correspondant au produit $A(X)B(X)$:

$$A(X)B(X) = C(X) \in \mathcal{R} \equiv a \star b = c \in \mathbb{Z}^n$$

Les algorithmes 1,2,3 présentent la génération de clefs, le chiffrement et le déchiffrement de NTRU.

Algorithme 1 : `NTRU. Generation de clefs`

Sortie : une paire de clé privée et clé publique (f, h)

1: Tirer $U(X)$ et $V(X)$ deux polynômes dans $\mathcal{R}$, dont les coefficients sont tirés uniformément dans $\{-1, 0, 1\}$
2: Calculer $F(X) = 1 + pU(X)$ et $G(X) = pV(X)$
3: Si $F(X)$ n'est pas inversible mod $X^n - 1$, recommencer pour U et F les étapes 1 et 2
4: Calculer $F(X)^{-1}$ mod $(X^n - 1)$ et $H(X) = G(X)F(X)^{-1}$ mod $(X^n - 1)$
5: La clé privée est $f \in \mathbb{Z}_q^n$ et la clé publique est $h \in \mathbb{Z}_q^n$

Algorithme 2 : `NTRU. Chiffrement`

Entrée : la clé publique h et un message $m \in \mathcal{M} = \{-1, 0, 1\}^n$
Sortie : un chiffré $y \in \mathbb{Z}_q^n$

1: Tirer uniformément $r \in \{-1, 0, 1\}^n$
2: Calculer et retourner $y = r \star h + m$ mod q

Algorithme 3 : `NTRU. Dechiffrement`

Entrée : la clé privée f et un chiffré $y \in \mathbb{Z}_q^n$
Sortie : un message $m \in \mathcal{M} = \{-1, 0, 1\}^n$

1: Calculer $a = y \star f$ mod q
2: Retouner $m' = a$ mod p

Plusieurs candidats du NIST se base sur NTRU comme NTRUEncrypt, NTRU-HRSS-KEM et NTRU Prime [21, 23, 6].

3.3 Le problème *learning with errors*

Learning with errors que l'on abrège LWE est un problème difficile introduit pour la première fois en 2005 par l'américain Oded Regev [40]. Depuis cette date, ce problème attise beaucoup de recherches. Plusieurs types de primitives se basent sur le problème LWE comme le chiffrement, l'encapsulation de clé ou la signature numérique.

Une grande qualité qu'on attribue à ce problème est sa difficulté qui est aussi élevée que le problème SIS. De plus, il est susceptible à de nombreuses améliorations. En effet, après 2005, la communauté scientifique a amélioré le problème LWE en introduisant des familles, plus précisément, des variants du problème LWE dans le but de le rendre plus sûr et plus performant. Les travaux du cryptographe français Damien Stehlé participent à la recherche, notamment en proposant une nouvelle famille nommée integer-LWE [14]. Nous allons voir que ce problème est décisif pour la suite du dossier car LWE est la base du problème exploité dans le projet Crystals en raison de ses qualités.

Le problème *learning with errors* ou l'apprentissage avec erreurs
Soient q un nombre premier, n un entier, $\mathcal{X}_s$ et $\mathcal{X}_e$ des distributions à coefficients dans $\mathbb{Z}_q$. Aussi, une séquence de m échantillons $(a_i, b_i) \in \mathbb{Z}_q^{n+1}$ où chaque a_i est choisi uniformément dans $\mathbb{Z}_q^n$. On pose :

$$bi = \sum_{j=1}^{n} a_{i,j} s_j + e_i \text{ mod } q, \text{ avec } e_i \text{ tiré selon } \mathcal{X}_e \text{ et } s_i \text{ tiré selon } \mathcal{X}_s.$$

Le problème repose sur la difficulté de trouver $s \in \mathbb{Z}_q^n$.

De manière intuitive, s est un secret et e représente un petit bruit aléatoire. Une autre manière de définir LWE est de poser la matrice A des $(a_{i,j})$ telle que $A \in \mathbb{Z}_q^{m \times n}$. Alors il s'agit de

$$\text{trouver } s \text{ à partir de } (A, b) \text{ tel que } As + e \equiv b \bmod q, \text{ avec } e \sim \mathcal{X}_e^m.$$

Pour e, les valeurs sont choisies aléatoirement selon une loi de distribution $\mathcal{X}_e$, souvent une loi Gaussienne discrète centrée en 0 et de petite variance.

3.4 Le chiffrement de Regev

Dans le même temps, Oded Regev introduit le chiffrement de Regev basé sur le problème LWE [40]. Le schéma de chiffrement a une particularité : le déchiffrement est probabiliste. En effet, lors du calcul final du déchiffrement, à savoir $z = v - \sum_{j=1}^{n} u_j s_j$, selon la valeur de ce calcul, on retourne soit le bit 0 ou le bit 1. En réalité, tous les schémas de chiffrement basés sur LWE ou une de ses familles est probabiliste. Concernant le schéma de chiffrement de Regev, il se définit de la manière suivante :

Algorithme 4 : `Regev. Generation de clefs`

Sortie : une paire de clé privée et clé publique (sk, pk)

1: Tirer s uniformément dans $\mathbb{Z}_q^n$
2: Tirer uniformément une matrice $A \in \mathbb{Z}_q^{m \times n}$
3: Tirer $e = (e_1, \ldots, e_m)^T$, où les e_i sont tirés selon $\mathcal{X}_e$
4: Calculer $b = As + e \bmod q$
5: La clé publique est $pk := (A, b)$ et la clé privée est s

Algorithme 5 : `Regev. Chiffrement`

Entrée : la clé publique pk et un message $x \in \{0, 1\}$
Sortie : un chiffré $y \in \mathbb{Z}_q^{n+1}$

1: Définir $r = (r_1, \ldots, r_m)$, où les r_i sont tirés uniformément dans $\{0, 1\}$
2: Si $x = 0$, alors définir $y = (r^T A, \langle r, b \rangle)$
3: Si $x = 1$, alors définir $y = (r^T A, \lfloor \frac{q}{2} \rfloor + \langle r, b \rangle)$
4: Retourner $y = (u, v)$

Algorithme 6 : `Regev. Dechiffrement`

Entrée : la clé privée sk et un chiffré $y = (u, v) \in \mathbb{Z}_q^{n+1}$
Sortie : un message $x \in \{0, 1\}$

1: Calculer $z = v - \sum_{j=1}^{n} u_j s_j$
2: Si $|z| < |\lfloor \frac{q}{2} \rfloor - z|$, retourner $x' = 0$
3: Sinon, retouner $x' = 1$

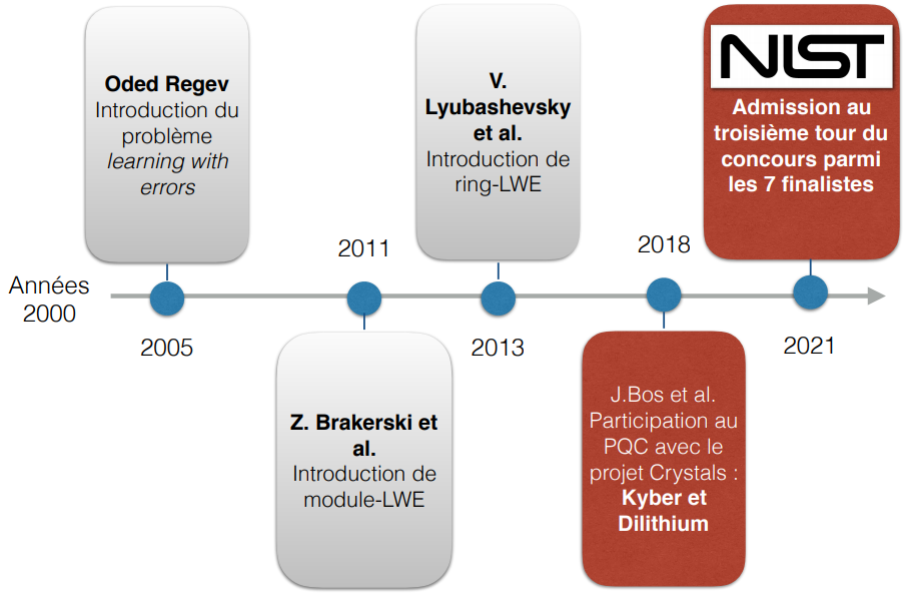


Figure 3.2: LWE, ses familles et participation au PQC du projet Crystals

3.5 Les familles de LWE

Historiquement, plusieurs familles de LWE se sont succédées suivant la naissance du problème LWE en 2005, voir figure 3.2.

Il a fallu quelques années avant que Z. Brakerski et al. introduisent, pour la première fois, le module-LWE en 2011. Autrefois nommé "generalized-LWE", le module-LWE a été utilisé dans le cadre de la construction d'un chiffrement homomorphe [10].

Ensuite, en 2013, V. Lyubashevsky et al. introduisent le ring-LWE qui sonne comme une amélioration du problème LWE [28]. En effet, le ring-LWE considère un anneau de polynômes à coefficients tel que $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$ où n est une puissance de 2, là où LWE ne considère qu'un anneau tel que $\mathbb{Z}_q$. Le problème ring-LWE s'inspire de NTRU défini dans la partie 3.2, dans le sens où ils partagent en commun un anneau structuré $\mathcal{R}$.

Le problème ring-LWE

Défini pour la première fois en 2013, le problème ring-LWE ajoute de la structure en comparaison à LWE en considérant un anneau de polynômes.

Le problème *ring-learning with errors*
Soient q un nombre premier, n une puissance de 2. On a l'anneau des polynômes $\mathcal{R}_q = \mathbb{Z}[X]/(X^n + 1)$. De plus, soient $\mathcal{X}_s$ et $\mathcal{X}_e$ des distributions à coefficients dans $\mathcal{R}_q$. On attribue à a un polynôme aléatoire choisi de manière uniforme dans $\mathcal{R}_q$. On pose :

$$b = as + e \bmod q, \text{ avec } e \text{ tiré selon } \mathcal{X}_e \text{ et } s \text{ tiré selon } \mathcal{X}_s.$$

Le problème repose sur la difficulté de trouver $s \in \mathcal{R}_q$ à partir de (a, b).

Le problème module-LWE

En comparaison avec ring-LWE, module-LWE possède moins de structure. Il s'agit du problème exploité par le candidat Kyber.

Le problème *module-learning with errors*
Soient q un nombre premier, n une puissance de 2. On a l'anneau des polynômes $\mathcal{R}_q = \mathbb{Z}[X]/(X^n + 1)$. De plus, soient k la dimension, $\mathcal{X}_s$ et $\mathcal{X}_e$ des distributions à coefficients dans $\mathcal{R}_q$. Aussi, une séquence de k échantillons $(a_i, b_i) \in \mathbb{Z}_q^{k+1}$ où chaque a_i est choisi uniformément dans $\mathbb{Z}_q^k$. On pose :

$$b_i = \sum_{j=1}^{k} a_{i,j} s_j + e_i \bmod q, \text{ avec } e_i \text{ tiré selon } \mathcal{X}_e \text{ et } s_i \text{ tiré selon } \mathcal{X}_s.$$

Le problème repose sur la difficulté de trouver $s \in \mathcal{R}_q$.

De manière informelle, on peut imaginer que la structure de ring-LWE est plus riche que LWE dans le sens où l'ensemble considéré $\mathbb{Z}_q[X]/(X^n + 1)$ est plus structuré que $\mathbb{Z}_q[X]$.

Il a été prouvé que ring-LWE a des propriétés équivalentes que LWE, voire mieux, car elle permet de réduire la taille de la clé publique par un facteur $\Theta(n)$. Aussi, les calculs s'effectuent plus efficacement. En effet, la multiplication de polynômes a un petit coût en $O(n \log(n))$ en utilisant la transformée de Fourier rapide. De plus, le choix du polynôme $X^n + 1$ où n est une puissance de 2 permet des calculs efficaces sur les architectures modernes.

La dernière famille à présenter est le retour de module-LWE, remis au goût du jour par J. Bos et al. lors du projet Crystals en 2018 [42]. La différence en comparaison avec ring-LWE est de considérer une dimension supérieure à 1 concernant la matrice A. On rappelle que A est une partie de la clé publique où A peut se voir comme une matrice ligne/colonne à coefficients dans $\mathbb{Z}_q$. Autrement dit, si A est de dimension 1, alors on est dans le cas de ring-LWE. Si A est de dimension supérieur à 1, alors il s'agit de module-LWE. Ceci signifie que plusieurs polynômes sont impliqués dans module-LWE contrairement à ring-LWE. Deux améliorations s'offrent à ce changement : la flexibilité et la sécurité.

D'une part, c'est un système flexible en fixant comme anneau des polynômes $\mathcal{R}_q = \mathbb{Z}_{3329}[X]/(X^{256} + 1)$ avec $n = 256$ et $q = 3329$. Il s'agit des paramètres les plus récents en raison de leurs performances. A l'origine, l'équipe Crystals envisageait $q = 7681$ mais le choix de $q = 3329$ permet des performances similaires, voir mieux lors de l'application des transformées de Fourier rapides. Le problème module-LWE est flexible car demander ou retirer plus de bits de sécurité s'effectue en changeant simplement la dimension. Tandis que pour ring-LWE, il ne possède pas cet avantage, c'est-à-dire que pour modifier le nombre de bits de sécurité, il faut choisir un nouvel anneau de polynômes avec des nouveaux paramètres judicieux pour n et q. Ceci implique une certaine lourdeur que ne possède pas module-LWE.

D'autre part, au vu des recherches sur les attaques basées sur le problème NTRU [1], il semblerait que la dimension des matrices considérée ait un impact sur la sécurité contre des attaques. En effet, plus la dimension est grande, plus la sécurité est forte. De plus, si

une attaque efficace vient à apparaître concernant le problème ring-LWE, cette attaque a une chance d'être moins efficace sur le problème module-LWE. Donc, module-LWE semble être plus robuste aux attaques que LWE et ring-LWE.

En définitif, module-LWE incarne une apothéose entre LWE et ring-LWE. En effet, module-LWE se situe entre le peu de structure de LWE et la structure forte de ring-LWE. Autrement dit, à ce jour, module-LWE semble être le meilleur problème en terme de sécurité, de performance et de facilité d'implémentation. C'est pourquoi, en se basant sur ce problème, le projet Crystals a toutes les qualités pour devenir le prochain standard résistant au modèle de calcul quantique. Il existe d'autres familles en étude tel que integer-LWE qui considère l'anneau $\mathbb{Z}_{f(q)}$ mais ont moins d'échos que les deux cités au-dessus [14].

Chapitre 4

Le projet Crystals

Logo du projet Crystals

4.1 L'équipe à l'origine du projet

Le projet Crystals (*Crypotographic Suite for Algebraic Lattices*) a pour ambition de proposer deux primitives cryptographiques, à savoir un KEM et une signature numérique, résistantes au modèle de calcul post-quantique. Pour cela, l'équipe du projet Crystals exploite des problèmes difficiles dans les réseaux euclidiens. Ainsi, à l'occasion de la conférence PQCRYPTO 2017, l'équipe présente, pour la première fois, Crystals-Kyber et Crystals-Dilithium.

La première primitive est un KEM avec une sécurité IND-CCA2 (indistinguabilité face à une attaque à chiffré choisi adaptative), mais il peut être aussi un schéma de chiffrement sémantiquement sûr IND-CPA (attaque à clair choisi), qui se base sur le problème de module-LWE.

La deuxième primitive est un schéma de signature numérique, basé sur le problème du vecteur le plus court (SVP).

L'équipe Crystals soumet ces deux primitives à la compétition PQC, qui rencontrent un franc succès et sont parmi les finalistes. En effet, leurs efficacités sont bonnes en terme de sécurité, de performance et de facilité d'implémentation. L'équipe Crystals est internationale et se compose de 10 cryptographes du milieu universitaire ou bien de grands groupes comme IBM, NXP et Apple, voir tableau 4.1. Dans la suite, nous parlerons de Kyber pour Kyber KEM. S'il s'agit du schéma de chiffrement, alors nous le préciserons.

4.2 Présentation de Crystals-Kyber

L'équipe Crystals décrit Kyber comme le successeur au projet NewHope [2]. En effet, le projet NewHope est un candidat de la compétition PQC qui parvient à être sélectionné

Membres de l'équipe Crystals		
Roberto Avanzi	ARM Limited	Allemagne
Joppe Bos	NXP Semiconductors	Belgique
Léo Ducas	CWI Amsterdam	Pays-Bas
Eike Kiltz	Ruhr University Bochum	Allemagne
Tancrède Lepoint	SRI International	Etats-Unis
Vadim Lyubashevsky	IBM Research Zurich	Suisse
John M. Schank	University of Waterloo	Canada
Peter Schwabe	Radboud University	Pays-Bas
Gregor Seiler	IBM Research Zurich	Suisse
Damien Stehle	ENS Lyon	France

Tableau 4.1: Les membres de l'équipe Crystals [42]

pour le deuxième tour mais n'est pas retenu pour le troisième tour. NewHope se base sur le problème de ring-LWE. De plus, certains membres du projet Crystals ont été des membres du projet NewHope dont Roberto Avanzi, Joppe Bos et Léo Ducas.

Kyber est le candidat favori parmi les finalistes à un tel point que Kyber est déjà intégré dans certaines bibliothèques de primitives cryptographiques d'industriels comme Cloudflare, Amazon et IBM [13] [3] [24].

Dans un premier temps, nous introduisons le schéma de chiffrement Kyber. Dans un second temps, nous présentons Kyber KEM.

4.2.1 Chiffrement Kyber

Tout d'abord, le schéma de chiffrement s'inspire du chiffrement de Regev défini dans la partie 3.4, et exploite le problème module-LWE.

C'est donc un schéma probabiliste composé de trois algorithmes : générations de clefs, chiffrement et déchiffrement. En effet, comme pour le chiffrement de Regev défini dans la partie 3.4, lors du déchiffrement, selon la valeur du calcul final, le bit peut valoir 0 ou 1. La taille de la clé publique est de 1088 octets, et celle de la clé privée est de 2368 octets. Concernant les chiffrés, leurs tailles sont de 1184 octets [9]

Soient k, d_t, d_u, d_v des paramètres positifs, $n = 256$, l'espace des clairs $\mathcal{M} = \{0,1\}^{256}$ et l'espace des chiffrés $\mathcal{C} = \{0,1\}^{256 \cdot kd_u} \times \{0,1\}^{256 \cdot d_v}$. Chaque clair $m \in \mathcal{M}$ peut être vu comme un polynôme dans $\mathcal{R}$ à coefficients dans $\{0,1\}$. De plus, on utilise plusieurs fonctions dans les algorithmes à venir :

Loi binomiale.
On définit la fonction β_η suivant un entier positif η telle que : on a une séquence de η échantillons $(a_i, b_i) \in \{0,1\}$ suivant une loi binomiale centrée. En sortie, on a un élément calculé par $\sum_{i=1}^{\eta}(a_i - b_i) \in \{-1, 0, 1\}$.

Compression et décompression.
On définit la fonction $\text{Compress}_q(x, d)$ telle que

$$\text{Compress}_q(x, d) = \lceil (2^d/q) \cdot x \rfloor \bmod 2^d$$

qui prend en entrée un élément $x \in \mathbb{Z}_q$, un entier positif d et retourne un entier dans $\{0, \ldots, 2^d - 1\}$ où $d < \lceil \log_2(q) \rceil$. De plus, la fonction Decompress_q est définie par

$$\text{Decompress}_q(x, d) = \lceil (q/2^d) \cdot x \rfloor$$

qui doit retourner un élément proche de x lorsqu'on fait appel à

$$x' = \text{Decompress}_q(\text{Compress}_q(x, d), d)$$

L'intérêt des fonctions $\text{Compress}_q(x, d)$ et Decompress_q est d'éliminer quelques bits de poids faible qui n'ont pas d'influence sur la probabilité de faire un déchiffrement correct. Ainsi, cela réduit la taille de la clé publique et celle du chiffré. Pour la suite, on introduit la notation suivante : un vecteur $\mathbf{v}$ est noté en gras et une matrice $\mathbf{A}$ est notée en gras et en majuscule. Le produit scalaire $\langle u, v \rangle$ est noté $\mathbf{uv}$ où les deux vecteurs sont notés en gras.

Algorithme 7 : `Kyber. Generation de clefs`

Sortie : une paire de clé publique et clé privée (pk, sk)

1: Tirer uniformément $\rho \in \{0,1\}^{256}$ et $\sigma \in \{0,1\}^{256}$
2: Étendre ρ à une taille arbitraire pour les attribuer à la matrice $\mathbf{A} \in \mathcal{R}_q^{k \times k}$
3: Étendre σ à une taille arbitraire pour les attribuer au couple $(\mathbf{s}, \mathbf{e}) \in \beta_\eta^k \times \beta_\eta^k$ suivant la loi binômiale β_η
4: Calculer le vecteur $\mathbf{t}$ tel que $\mathbf{t} := \text{Compress}_q(\mathbf{As} + \mathbf{e}, d_t)$
5: Retourner la clé publique $pk := (\mathbf{t}, \rho)$ et la clé privée $sk := \mathbf{s}$

Algorithme 8 : `Kyber. Chiffrement`

Entrée : la clé publique $pk := (\mathbf{t}, \rho)$ et un clair $m \in \mathcal{M}$
Sortie : un chiffré $c := (\mathbf{u}, v) \in \mathcal{C}$

1: Tirer uniformément $r \in \{0,1\}^{256}$
2: Calculer $\mathbf{t}$ tel que $\mathbf{t} := \text{Decompress}_q(\mathbf{t}, d_t)$
3: Étendre ρ à une taille arbitraire pour les attribuer à la matrice $\mathbf{A} \in \mathcal{R}_q^{k \times k}$
4: Étendre r à une taille arbitraire pour les attribuer au triplet $(\mathbf{r}, \mathbf{e}_1, e_2) \in \beta_\eta^k \times \beta_\eta^k \times \beta_\eta$ suivant la loi binomiale β_η
5: Calculer le vecteur $\mathbf{u}$ tel que $\mathbf{u} := \text{Compress}_q(\mathbf{A}^T\mathbf{r} + \mathbf{e}_1, d_u)$
6: Calculer v tel que $v := \text{Compress}_q(\mathbf{t}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m, d_v)$, où $\lceil \frac{q}{2} \rfloor$ désigne un arrondissement de $\frac{q}{2}$
7: Retourner le chiffré $c := (\mathbf{u}, v)$

Algorithme 9 : `Kyber. Dechiffrement`

Entrée : la clé privée $(sk = \mathbf{s})$ et un chiffré $c = (\mathbf{u}, v)$
Sortie : un clair $m \in \mathcal{M}$

1: Calculer le vecteur $\mathbf{u}$ tel que $\mathbf{u} := \text{Decompress}_q(\mathbf{u}, d_u)$
2: Calculer v tel que $v := \text{Decompress}_q(v, d_v)$
3: Si $\text{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1)$ est plus proche de $\lceil \frac{q}{2} \rfloor$ que de 0, alors retourner le bit 1
4: Sinon, retourner le bit 0
5: Retourner le clair $m \in \{0,1\}^{256}$

Notations reprises et versions simplifiées de CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM [9]

4.2.2 Encapsulation de clé Kyber

Une fois qu'on a défini le chiffrement Kyber, après application de la transformation de Fujisaki-Okamoto dont on ne détaillera pas les caractéristiques dans ce dossier [19], on peut définir un schéma d'encapsulation de clé avec une sécurité IND-CCA2. On définit l'encapsulation de clé comme une méthode sécurisée de chiffrer une clé à l'aide d'une autre clé afin de la stocker ou de l'envoyer sur un canal non sécurisé.

Ce schéma est composé de trois algorithmes : génération de clefs qui est identique à celui du PKE, encapsulation de clé et décapsulation de clé. Aussi, le schéma utilise les algorithmes de chiffrement et déchiffrement Kyber. Le schéma se définit de la manière suivante :

Soient deux fonctions de hachages $\mathrm{G} : \{0,1\}^* \to \{0,1\}^{2\times 256}$ et $\mathrm{H} : \{0,1\}^* \to \{0,1\}^{256}$. L'algorithme de génération de clefs est identique à celui du chiffrement à l'exception de la clé privée sk où on ajoute $(\mathbf{t}, \rho)$ et un secret z tiré uniformément dans $\{0,1\}^{256}$. Ainsi, la clé privée est $sk = (\mathbf{s}, z, \mathbf{t}, \rho)$.

Algorithme 10 : `Kyber. Encapsulation de cle`

Entrée : la clé publique $pk = (\mathbf{t}, \rho)$
Sortie : un chiffré $c := (\mathbf{u}, v)$ et une clé pseudo-aléatoire K

1: Tirer uniformément $m \in \{0,1\}^{256}$
2: Calculer un couple $(\hat{K}, r)$ tel que $(\hat{K}, r) := \mathrm{G}(\mathrm{H}(pk), m)$
3: Attribuer au couple $(\mathbf{u}, v)$ le chiffrement Kyber tel que $(u, v) := \text{Chiffrement}((\mathbf{t}, \rho), m; r)$
4: Attribuer à c le couple $(\mathbf{u}, v)$
5: Calculer K tel que $K := \mathrm{H}(\hat{K}, \mathrm{H}(c))$
6: Retourner le couple (c, K)

Algorithme 11 : `Kyber. Decapsulation de cle`

Entrée : la clé secrète $sk = (\mathbf{s}, z, \mathbf{t}, \rho)$ et le chiffré $c = (\mathbf{u}, v)$
Sortie : une clé pseudo-aléatoire K

1: Attribuer à m' le déchiffrement Kyber tel que $m' := \text{Dechiffrement}(s, (\mathbf{u}, v))$
2: Calculer un couple $(\hat{K}', r')$ tel que $(\hat{K}', r') := \mathrm{G}(\mathrm{H}(pk), m')$
3: Attribuer le chiffrement Kyber au couple $(\mathbf{u}', v')$ tel que $(\mathbf{u}', v') := \text{Chiffrement}((t, \rho), m', r')$
4: Si $(\mathbf{u}', v') = (\mathbf{u}, v)$, alors retourner K tel que $K := \mathrm{H}(\hat{K}', \mathrm{H}(c))$
5: Sinon, retourner K tel que $K := \mathrm{H}(z, H(c))$

Notations reprises de CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM [9]

4.2.3 Les versions de Kyber

Kyber se décline en plusieurs versions : Kyber-512, Kyber-768 et Kyber-1024. Ces versions se distinguent en fonction des paramètres de plus en plus grands pour plus de sécurité, voir tableau 4.2 où δ est la probabilité que la décapsulation de clé échoue.

Chaque version offre une bonne sécurité, respectant les conditions de sécurité du NIST établies au tableau 1.1. Plus précisément, d'après les recherches, Kyber-512 est aussi dur

	n	k	q	(d_u, d_v)	δ
Kyber-512	256	2	3329	$(10, 4)$	2^{-139}
Kyber-768	256	3	3329	$(10, 4)$	2^{-164}
Kyber-1024	256	4	3329	$(11, 5)$	2^{-174}

Tableau 4.2: Les différents paramètres de Kyber [5]

à casser que AES-128 (niveau I) ; Kyber-768 est aussi dur à casser que AES-192 (niveau III) ; Kyber-1024 est aussi dur à casser que AES-256 (niveau V). Pour un usage réel, l'équipe Crystals recommande l'utilisation de Kyber-768 qui offre plus de 128 bits de sécurité dans le modèle de calcul classique et quantique.

Nous ne traiterons pas dans le détail les performances, les attaques, les améliorations possibles et les comparaisons à d'autres KEM dans ce dossier. Cela dit, nous vous orientons vers l'article de recherche en question pour en savoir plus [5].

4.3 Présentation de Crystals-Dilithium

Au même titre que Kyber, Dilithium est parmi les trois finalistes au sein de la compétition PQC en tant que signature numérique. Cela se justifie par le fait que Dilithium propose la plus petite taille de clés publiques combinées aux signatures en utilisant les réseaux euclidiens.

Le problème difficile exploité est le problème SVP. Dilithium est conçu de manière à respecter plusieurs critères : une implémentation simple et sécurisée ; des paramètres fixés pour une sécurité à long-terme ; une minimisation de la taille d'un couple composé d'une clé publique et d'une signature ; une facilité d'ajustement des paramètres.

Il existe plusieurs versions de Dilithium que sont Dilithium2, Dilithium3 et Dilithium5. Le nombre correspond au niveau de sécurité défini au tableau 1.1. Concrètement, la valeur des paramètres utilisés change en fonction du niveau de sécurité [17]. Ainsi, la taille des signatures diffère selon la version. Pour la version 2, la taille est de 2420 octets ; pour la version 3, la taille est de 3293 ; pour la version 5, la taille est de 4595.

Le schéma de signature numérique est donné dans la section suivante.

4.3.1 Signature numérique Dilithium

Soient l'espace des clairs $\mathcal{M} = \{0,1\}^{256}$ et l'espace des signatures $\mathcal{S} = \{0,1\}^{256} \times \{0,1\}^{\log_2\binom{256}{\tau}+\tau}$ où τ est un entier positif de telle sorte que la taille de $\{0,1\}^{\log_2\binom{256}{\tau}+\tau}$ soit comprise entre 128 et 256. Soient l'anneau des polynômes $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n+1)$ avec $n = 256$, $q = 2^{23} - 2^{13} + 1$. On a η petit, γ_1, γ_2 et β qui sont des entiers positifs.

De plus, on définit les fonctions, de hachage $\mathrm{H} : \{0,1\}^* \to \{0,1\}^{256}$, HighBits et LowBits. La fonction HighBits prend en entrée un vecteur et un paramètre γ, et retourne les bits de poids fort. Le paramètre γ est une borne des coefficients du vecteur de sortie. La fonction LowBits est identique sauf qu'elle retourne les bits de poids faible.

Ainsi, le schéma de signature numérique se définit par les trois algorithmes suivants.

Algorithme 12 : `Dilithium. Generation de clefs`

Sortie : une paire de clé publique et clé privée (pk, sk)

1: Tirer uniformément k polynômes de $\mathcal{R}$ puis attribuer ses coefficients à la matrice $\mathbf{A} \in \mathcal{R}_q^{k\times\ell}$
2: Tirer uniformément le vecteur $\mathbf{s}_1 \in \mathbb{Z}_\eta^\ell$
3: Tirer uniformément le vecteur $\mathbf{s}_2 \in \mathbb{Z}_\eta^k$
4: Calculer $\mathbf{t}$ tel que $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
5: Retourner la clé publique $pk = (\mathbf{A}, \mathbf{t})$ et la clé privé $sk = (\mathbf{A}, \mathbf{t}, \mathbf{s}_1, \mathbf{s}_2)$

Algorithme 13 : `Dilithium. Signature`

Entrée : un clair $m \in \mathcal{M}$ et une clé privée sk
Sortie : une signature $\sigma = (\mathbf{z}, c) \in \mathcal{S}$

1: Tirer uniformément $\mathbf{y} \in \mathbb{Z}_{\gamma_1-1}^\ell$, où $\gamma \geq 3$
2: Calculer $\mathbf{w}_1$ tel que $\mathbf{w}_1 := \mathrm{HighBits}(\mathbf{A}\mathbf{y}, 2\gamma_2)$
3: Calculer c tel que $c := \mathrm{H}(m \parallel \mathbf{w}_1)$ où c est un polynôme dans $\mathcal{R}_q$
4: Calculer $\mathbf{z}$ tel que $\mathbf{z} := \mathbf{y} + c\mathbf{s}_1$
5: Si $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ ou $\|\mathrm{LowBits}(\mathbf{A}\mathbf{y} - c\mathbf{s}_2, 2\gamma_2)\|_\infty \geq \gamma_2 - \beta$, alors retourner à l'étape 1
6: Retourner la signature $\sigma = (\mathbf{z}, c)$

Algorithme 14 : `Dilithium. Verification`

Entrée : un clair $m \in \mathcal{M}$, une clé publique pk et la signature $\sigma = (\mathbf{z}, c) \in \mathcal{S}$

Sortie : Le booléen "True" ou "False"

1: Calculer $\mathbf{w}'_1$ tel que $\mathbf{w}'_1 := \text{HighBits}(\mathbf{Ay} - c\mathbf{t}, 2\gamma_2)$

2: Si $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ et $c = \text{H}(m \parallel \mathbf{w}'_1)$, alors retourner "True"

3: Sinon, retourner "False"

Notations reprises de Crystals–dilithium: Digital signatures from module lattices [17]

Chapitre 5

Implémentation de schémas de chiffrement en SageMath

Dans ce chapitre, on se propose d'implémenter des schémas de chiffrement basés sur le problème LWE ainsi que des familles de LWE. Les codes sont réalisés sur SageMath 9.2.

Proposition en LWE

Voici une proposition d'un schéma de chiffrement basé sur le problème LWE, inspiré par le schéma de chiffrement de Regev et celui de Kyber. Dans un premier temps, on retrouve le pseudo-code, puis dans un second temps, on présente une proposition composée de quatre algorithmes : approximation, génération de clefs, chiffrement et déchiffrement. L'espace des clairs est $\mathcal{M} = \{0,1\}^{256}$ et l'espace des chiffrés est $\mathcal{C} = \mathbb{Z}_q^k \times \{0,1\}^{256}$.

Algorithme 15 : `LWE. Approximation`

Entrée : une liste l et un modulo q
Sortie : une autre liste $M \in \{0,1\}^{256}$

1: Initialiser une nouvelle liste M
2: Pour chaque élément de l, si la valeur de l'élément est compris entre $\lceil \frac{q}{4} \rfloor$ et $\lceil \frac{3 \cdot q}{4} \rfloor$, alors retourner le bit 1
3: Sinon, retourner le bit 0

Algorithme 16 : `LWE. Generation de clefs`

Entrée : la dimension k et le modulo q
Sortie : une paire de clé publique et clé privée (pk, sk)

1: Tirer uniformément une matrice $\mathbf{A} \in \mathbb{Z}_q^{k \times k}$
2: Tirer un vecteur $\mathbf{s}$ de taille k à partir d'une loi gaussienne centrée en 0 et de petite variance
3: Tirer un vecteur $\mathbf{e}$ de taille k à partir de la même loi gaussienne
4: Calculer $\mathbf{b}$ tel que $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$
5: Retourner la clé publique $pk := (\mathbf{A}, \mathbf{b})$ et la clé privée $sk := \mathbf{s}$

Algorithme 17 : `LWE. Chiffrement`

Entrée : la clé publique pk, le clair $\mathbf{m}$, le modulo q et la dimension k
Sortie : le chiffré $c \in \mathcal{C}$

1: Tirer un vecteur $\mathbf{r}$ de taille k à partir de la loi gaussienne
2: Tirer un vecteur $\mathbf{e}_1$ de taille k à partir de la loi gaussienne
3: Tirer un vecteur $\mathbf{e}_2$ de taille 1 à partir de la loi gaussienne
4: Calculer $\mathbf{u}$ tel que $\mathbf{u} = \mathbf{rA} + \mathbf{e}_1$
5: Calculer $\mathbf{v}$ tel que $\mathbf{v} = \mathbf{rb} + \mathbf{e}_2 + \lceil \frac{q}{2} \rfloor \mathbf{m}$
6: Retourner le chiffré $c := (\mathbf{u}, \mathbf{v})$

Algorithme 18 : `LWE. Dechiffrement`

Entrée : la clé privée $sk = \mathbf{s}$, le chiffré c et le modulo q
Sortie : le clair $m \in \mathcal{M}$

1: Calculer $\mathbf{m}$ tel que $\mathbf{m} = \mathbf{v} - \mathbf{us}$
2: Appliquer la fonction `Approximation` à $\mathbf{m}$ puis retourner le polynôme appartenant à $\mathcal{R}_2$

On utilise les paramètres $sigma = 3$, $q = 3329$, $k = 3$. Voici le code source écrit sur SageMath :

```
import random
from sage.stats.distributions.discrete_gaussian_integer import
    DiscreteGaussianDistributionIntegerSampler

def approx(liste,q):
    M=[]
    for x in liste:
        if(0<=x<=int(q/4)):
            M.append(0)
        elif(int(q/4)<x<=int(q/2)):
            M.append(1)
        elif(int(q/2)<x<=int(3*q/4)):
            M.append(1)
        else:
            M.append(0)
    return M

def generationdeclefs(k,q):
    A=random_matrix(GF(q),k,k)
    D=DiscreteGaussianDistributionIntegerSampler(sigma=sigma)
    s=[D() for x in range(k)]
    s=matrix(k,1,s)
    e=[D() for x in range(k)]
    e=matrix(k,1,e)

    b=A*s+e

    pk=(A,b)
    sk=s
    print("La clé publique est : ",pk)
    print("La clé privée est : ",sk)
    return (pk,sk)

def chiffrement(pk,m,q,k):
    D=DiscreteGaussianDistributionIntegerSampler(sigma=sigma)
    r=[D() for x in range(k)]
    r=matrix(1,k,r)
    e1=[D() for x in range(k)]
    e1=matrix(1,k,e1)
    e2=[D() for x in range(1)]
```

```
    e2=matrix(1,1,e2)

    u=r*pk[0]+e1
    v=[(r*pk[1]+e2)+(int(q/2)*i)for i in m]
    return (u,v)

def dechiffrement(sk,c,q):
    uu=c[0]*sk
    m=[i-uu for i in c[1]]
    m=approx(m,q)
    return m

#************************************************
#*                  AFFICHAGE                   *
#************************************************

# Déclaration de l'écart-type, du modulo et d'une petite dimension
sigma=3
q=3329
k=3

# Appel des algorithmes
clefs=generationdeclefs(k,q)
pk=clefs[0]
sk=clefs[1]
# le clair est tiré uniformément dans {0,1}^{256}
m=[ZZ.random_element(0,2) for i in range(256)]
chiffre=chiffrement(pk,m,q,k)
mm=dechiffrement(sk,chiffre,q)

print("Vérification du schéma de chiffrement :",m==mm)
```

```
La clé publique est :  ([ 893 2908  740]
[  87  115  113]
[ 855  524 2307], [ 129]
[ 197]
[1313])
La clé privée est :  [-3]
[ 3]
[ 1]
Vérification du schéma de chiffrement : True
```

Figure 5.1: Affichage et vérification du schéma de chiffrement

Proposition en ring-LWE

Ensuite, on propose une implémentation d'un schéma de chiffrement basé sur le problème ring-LWE. Il y a cinq algorithmes : approximation, transformation d'une liste en polynôme, génération de clefs, chiffrement et déchiffrement. Dans un premier temps, on établit le pseudo-code, puis dans un second temps, on présente une proposition de code. L'espace des clairs est $\mathcal{M} = \mathcal{R}_2$ et l'espace des chiffrés est $\mathcal{C} = \mathcal{R}_q \times \mathcal{R}_q$.

Algorithme 19 : `ring-LWE. Approximation`

Entrée : une liste l et un modulo q
Sortie : une autre liste $M \in \{0,1\}^{256}$

1: Initialiser une nouvelle liste M
2: Pour chaque élément de l, si la valeur de l'élément est compris entre $\lceil \frac{q}{4} \rfloor$ et $\lceil \frac{3 \cdot q}{4} \rfloor$, alors retourner le bit 1
3: Sinon, retourner le bit 0

Algorithme 20 : `ring-LWE. Transformation d'une liste en polynome`

Entrée : une liste a, la puissance de 2: n et un modulo q
Sortie : un polynôme $tmp \in \mathcal{R}_q$

1: Créer un polynôme tmp où ses coefficients sont les éléments de la liste a
2: Retourner le polynôme tmp

Algorithme 21 : `ring-LWE. Generation de clefs`

Entrée : la puissance de 2: n et le modulo q
Sortie : une paire de clé publique et clé privée (pk, sk)

1: Tirer uniformément une matrice $\mathbf{A} \in \mathcal{R}_q$
2: Tirer un vecteur $\mathbf{s} \in \mathcal{R}_q$ à partir d'une loi gaussienne centrée en 0 et de petite variance
3: Tirer un vecteur $\mathbf{e} \in \mathcal{R}_q$ à partir de la même loi gaussienne
4: Calculer $\mathbf{b}$ tel que $\mathbf{b} = \mathbf{As} + \mathbf{e}$
5: Retourner la clé publique $pk := (\mathbf{A}, \mathbf{b})$ et la clé privée $sk := \mathbf{s}$

Algorithme 22 : `ring-LWE. Chiffrement`

Entrée : la clé publique pk, le clair $\mathbf{m}$, la puissance de 2: n et le modulo q
Sortie : le chiffré $c \in \mathcal{C}$

1: Tirer une matrice $\mathbf{r} \in \mathcal{R}_q$ à partir de la loi gaussienne
2: Tirer une matrice $\mathbf{e}_1 \in \mathcal{R}_q$ à partir de la loi gaussienne
3: Tirer une matrice $\mathbf{e}_2 \in \mathcal{R}_q$ à partir de la loi gaussienne
4: Calculer $\mathbf{u}$ tel que $\mathbf{u} = \mathbf{Ar} + \mathbf{e}_1$
5: Calculer $\mathbf{v}$ tel que $\mathbf{v} = \mathbf{br} + \mathbf{e}_2 + \lceil \frac{q}{2} \rfloor \mathbf{m}$
6: Retourner le chiffré $c := (\mathbf{u}, \mathbf{v})$

Algorithme 23 : `ring-LWE. Dechiffrement`

Entrée : la clé privée $sk = \mathbf{s}$, le chiffré c et le modulo q
Sortie : le clair $m \in \mathcal{M}$

1: Calculer $\mathbf{m}$ tel que $\mathbf{m} = \mathbf{v} - \mathbf{su}$
2: Appliquer la fonction `Approximation` à $\mathbf{m}$ puis retourner le polynôme appartenant à $\mathcal{R}_2$

On utilise les paramètres $sigma = 3$, $n = 16$, $q = 3329$. Voici le code source exploitant le problème ring-LWE :

```
import random
from sage.stats.distributions.discrete_gaussian_integer import
    DiscreteGaussianDistributionIntegerSampler

def approx(liste,q):
    M=[]
    for x in liste:
        if(0<=x<=int(q/4)):
            M.append(0)
        elif(int(q/4)<x<=int(q/2)):
            M.append(1)
        elif(int(q/2)<x<=int(3*q/4)):
            M.append(1)
        else:
            M.append(0)
```

```
15      return M
16
17  def trpolynome(list_a,n,q):
18      Z=ZZ.quotient(q)
19      P=PolynomialRing(Z,"a")
20      a=P.gen()
21      I=P.ideal([a**n+1])
22      S=P.quotient_ring(I,"x")
23      x=S.gen()
24      tmp=0
25      j=0
26      for i in list_a:
27          tmp = tmp + (i * (x**(j)))
28          j += 1
29      return tmp
30
31  def generationdeclefs(n,q):
32      R=PolynomialRing(GF(q),"a")
33      a=R.gen()
34      I=R.ideal([a^n+1])
35      S=R.quotient_ring(I,"x")
36      x=S.gen()
37      M=MatrixSpace(S,1,1)
38
39      A=M.random_element()
40      D=DiscreteGaussianDistributionIntegerSampler(sigma=sigma)
41      s=[D() for x in range(n)]
42      s=matrix(1,1,trpolynome(s,n,q))
43      e=[D() for x in range(n)]
44      e=matrix(1,1,trpolynome(e,n,q))
45
46      b=A*s+e
47
48      pk=(A,b)
49      sk=s
50      print("La clé publique est : ",pk)
51      print("La clé privée est : ",sk)
52      return (pk,sk)
53
54  def chiffrement(pk,m,n,q):
55      D=DiscreteGaussianDistributionIntegerSampler(sigma=sigma)
56      r=[D() for x in range(n)]
57      r=matrix(1,1,trpolynome(r,n,q))
58      e1=[D() for x in range(n)]
59      e1=matrix(1,1,trpolynome(e1,n,q))
60      e2=[D() for x in range(n)]
61      e2=matrix(1,1,trpolynome(e2,n,q))
62
63      u=pk[0]*r+e1
64      X=(pk[1]*r+e2)
65      Y=(int(q/2)*m)
66      v=X+Y
67      return (u,v)
68
69
70  def dechiffrement(sk,n,c,q):
71      m=c[1]-(sk*c[0])
72      coeff=list(m[0][0])
73      m=approx(coeff,q)
74      m=trpolynome(m,n,q)
75      return m
76
77  #***********************************************
78  #*                  AFFICHAGE                  *
79  #***********************************************
80
81  # Déclaration de l'écart-type, de la puissance de 2 et du modulo
82  sigma=3
83  n=16
84  q=3329
```

```
85
86  # Appel des algorithmes
87  clefs=generationdeclefs(n,q)
88  pk=clefs[0]
89  sk=clefs[1]
90
91  # le clair est tiré uniformément dans l'anneau des polynômes Z_2/(X^16+1)
92  m=[randint(0, 1) for i in range(n)]
93  m=trpolynome(m,n,q)
94
95  chiffre=chiffrement(pk,m,n,q)
96  mm=dechiffrement(sk,n,chiffre,q)
97
98  print("Vérification du schéma de chiffrement :",m==mm)
```

```
La clé publique est :  ([1286*x^15 + 2440*x^14 + 277*x^13 + 2761*x^12 + 2854*x^11 + 2120*x^10
+ 534*x^9 + 971*x^8 + 684*x^7 + 1656*x^6 + 663*x^5 + 2000*x^4 + 2761*x^3 + 30*x^2 + 3059*x +
581], [369*x^15 + 2189*x^14 + 1018*x^13 + 3218*x^12 + 320*x^11 + 98*x^10 + 1574*x^8 + 1604*x^
7 + 930*x^6 + 1627*x^5 + 239*x^4 + 934*x^3 + 2708*x^2 + 1103*x + 302])
La clé privée est :  [3326*x^14 + 3*x^13 + 3327*x^12 + 3327*x^10 + 3325*x^9 + x^8 + 3327*x^7
+ 3326*x^6 + 3328*x^5 + 3325*x^4 + 3328*x^3 + 3326*x^2 + x + 3]
Vérification du schéma de chiffrement : True
```

Figure 5.2: Affichage et vérification du schéma de chiffrement

Proposition en module-LWE

Enfin, concernant le schéma de chiffrement basé sur le module-LWE, nous proposons quelques pistes d'implémentation. Soient l'espace des clairs $\mathcal{M} = \mathcal{R}_2$ et l'espace des chiffrés $\mathcal{C} = \mathcal{R}_q \times \mathcal{R}_q$. Une piste intéressante est de suivre le pseudo-code pour le problème LWE en considérant les opérations sur des polynômes. Dans ce cas, on considère un remplissage de la matrice A et des vecteurs s, e dans $\mathbb{Z}_q$. Puis de transformer les lignes de la matrice A en polynômes ainsi que les colonnes des vecteurs s, e par la fonction de transformation. Par conséquent, $A \in \mathbb{Z}_q^{k\times k}$, $s \in \mathbb{Z}_q^k$ et $e \in \mathbb{Z}_q^k$ où $k = 3$ tels que

$$\underbrace{\begin{pmatrix} p_{1,1} & p_{1,2} & p_{1,3} \\ p_{2,1} & p_{2,2} & p_{2,3} \\ p_{3,1} & p_{3,2} & p_{3,3} \end{pmatrix}}_{A} \times \underbrace{\begin{pmatrix} s_{1,1} \\ s_{2,1} \\ s_{3,1} \end{pmatrix}}_{s} + \underbrace{\begin{pmatrix} e_{1,1} \\ e_{2,1} \\ e_{3,1} \end{pmatrix}}_{e} = \underbrace{\begin{pmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \end{pmatrix}}_{b}$$

On obtient ainsi $pk = (A, b)$ et $sk = s$. Ensuite, pour le chiffrement, on suit le pseudo-code où les calculs pour déterminer le couple (u, v) seront à base de polynômes. Lors du maniement des opérations entre polynômes, il faudra faire attention à ce que la dimension des matrices soit cohérente pour appliquer des multiplications entre plusieurs matrices. Enfin, lors du déchiffrement, à l'issue du calcul $v - us$, on est censé avoir un polynôme dans $\mathcal{R}_q$ où on applique la fonction d'approximation pour avoir un polynôme dans $\mathcal{R}_2$. On vérifie si le clair obtenu est bien le même que celui de départ. Si c'est le cas, alors le schéma de chiffrement est opérationnel.

Conclusion

La fin de la compétition PQC approche ; les candidats Crystals-Kyber et Crystals-Dilithium sont prometteurs et remplissent amplement les critères de sécurité, de performance et de caractéristiques d'implémentation. C'est pourquoi il y a une chance qu'ils soient standardisés dans les années à venir pour une mise en place dès 2024 comme ce fut le cas pour la fonction de hachage SHA-3 devenue un standard en 2015. Par conséquent, on dispose de primitives efficaces et sûres lorsque les premiers ordinateurs quantiques feront leur apparition.

Le consensus scientifique autour des réseaux euclidiens démontre la confiance en ses problèmes difficiles. Ainsi, les réseaux euclidiens sont pressentis pour dominer la sphère cryptographique dans un avenir proche.

La technologie quantique est un enjeu d'avenir dont les gouvernements investissent de plus en plus. En particulier, le 26 janvier 2021, le président Macron annonce un investissement de $1,8$ milliards d'euros dans la technologie quantique pour les cinq années à venir. Parmi cette somme, 150 millions d'euros sont consacrés à la cryptographie post-quantique [18].

Bibliographie

[1] Martin Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions. In *Annual International Cryptology Conference*, pages 153–178. Springer, 2016.

[2] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange—a new hope. In *25Th {USENIX} security symposium ({USENIX} security 16)*, pages 327–343, 2016.

[3] Amazon. Round 2 post-quantum tls is now supported in AWS KMS. `https://aws.amazon.com/fr/blogs/security/round-2-post-quantum-tls-is-now-supported-in-aws-kms/`, 2020.

[4] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar Melchor, et al. BIKE: bit flipping key encapsulation. 2017.

[5] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-kyber algorithm specifications and supporting documentation. *NIST PQC Round*, 2:4, 2017.

[6] Daniel J Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime: reducing attack surface at low cost. In *International Conference on Selected Areas in Cryptography*, pages 235–260. Springer, 2017.

[7] Daniel J Bernstein, Nadia Heninger, Paul Lou, and Luke Valenta. Post-quantum rsa. In *International Workshop on Post-Quantum Cryptography*, pages 311–329. Springer, 2017.

[8] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. `https://pq-crystals.org/kyber/index.shtml`, 2017.

[9] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-kyber: a CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.

[10] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[11] William J Buchanan. `https://asecuritysite.com/encryption/lattice_plot`, 2021.

[12] Antoine Casanova, Jean-Charles Faugere, Gilles Macario-Rat, Jacques Patarin, Ludovic Perret, and Jocelyn Ryckeghem. *GeMSS: a great multivariate short signature*. PhD thesis, UPMC-Paris 6 Sorbonne Universités; INRIA Paris Research Centre, MAMBA Team ..., 2017.

[13] Cloudflare. Securing the post-quantum world. `https://blog.cloudflare.com/securing-the-post-quantum-world/`, 2020.

[14] Julien Devevey, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. On the integer polynomial learning with errors problem. Cryptology ePrint Archive, Report 2021/277, 2021. `https://eprint.iacr.org/2021/277`.

[15] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *International Conference on Applied Cryptography and Network Security*, pages 164–175. Springer, 2005.

[16] Léo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. `https://pq-crystals.org/dilithium/index.shtml`, 2017.

[17] Léo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals–dilithium: Digital signatures from module lattices. 2018.

[18] République Française. €1.8 billion in funding for quantum technologies. https://investinfrance.fr/e1-8-billion-in-funding-for-quantum-technologies/, 2021.

[19] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In *International Workshop on Public Key Cryptography*, pages 53–68. Springer, 1999.

[20] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[21] Jeff Hoffstein, Jill Pipher, John M Schanck, Joseph H Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for NTRUEncrypt. In *Cryptographers' Track at the RSA Conference*, pages 3–18. Springer, 2017.

[22] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. NTRU: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*, pages 267–288. Springer, 1998.

[23] Andreas Hülsing, Joost Rijneveld, John M Schanck, and Peter Schwabe. NTRU-HRSS-KEM. *NIST submissions*, 2017.

[24] IBM. World's first quantum computing safe tape drive. https://www.ibm.com/blogs/research/2019/08/crystals/, 2019.

[25] Inria. Comment fonctionne un ordinateur quantique ? https://www.inria.fr/fr/comment-fonctionne-un-ordinateur-quantique, 2020.

[26] INRIA. Quantum computing: a new industrial revolution? https://www.inria.fr/en/Quantum-computing-industrial-revolution, 2020.

[27] Brian Koziel, A-Bon Ackie, Rami El Khatib, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. SIKE'd up: Fast and secure hardware architectures for supersingular isogeny key encapsulation. Cryptology ePrint Archive, Report 2019/711, 2019. https://eprint.iacr.org/2019/711.

[28] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Annual International Conference*

on the Theory and Applications of Cryptographic Techniques, pages 1–23. Springer, 2010.

[29] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loıc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, and Gilles Zémor. Hamming quasi-cyclic (HQC). *NIST PQC Round*, 2:4–13, 2018.

[30] Michele Mosca. Cybersecurity in an era with quantum computers: will we be ready? *IEEE Security & Privacy*, 16(5):38–41, 2018.

[31] NewScientist. Quantum computer sets new record for finding prime number factors. `https://www.newscientist.com/`, 2019.

[32] NIST. Cryptographic standards and guidelines : AES development. `https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development`, 2016.

[33] NIST. Post-quantum cryptography. `https://csrc.nist.gov/Projects/post-quantum-cryptography/`, 2016.

[34] NIST. Hash functions : SHA-3 project. `https://csrc.nist.gov/projects/hash-functions/sha-3-project`, 2017.

[35] NIST. Post quantum cryptography standardization: Announcement and outline of NIST's call for submissions. `https://csrc.nist.gov/Presentations/2016/Announcement-and-outline-of-NIST-s-Call-for-Submis`, 2017.

[36] NIST. The ship has sailed: The NIST post-quantum cryptography "competition". `https://csrc.nist.gov/Presentations/2017/The-Ship-has-Sailed-The-NIST-Post-Quantum-Cryptog`, 2017.

[37] NIST. Let's get ready to rumble -the NIST PQC "competition". `https://csrc.nist.gov/CSRC/media/Presentations/Let-s-Get-Ready-to-Rumble-The-NIST-PQC-Competiti/images-media/PQCrypto-April2018_Moody.pdf`, 2018.

[38] NIST. The 2nd round of the NIST PQC standardization process-opening remarks at PQC 2019. `https://csrc.nist.gov/CSRC/media/Presentations/the-2nd-round-of-the-nist-pqc-standardization-proc/images-media/moody-opening-remarks.pdf`, 2019.

[39] NIST. NIST PQC standardization update - round 2 and beyond. https://csrc.nist.gov/Presentations/2020/pqc-update-round-2-and-beyond, 2020.

[40] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[41] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[42] CRYSTALS Team. https://pq-crystals.org/index.shtml, 2017.