Table of contents

Cryptographie Quantique et Post-Quantique

Introduction et Contexte

Vue d'ensemble du domaine

La cryptographie post-quantique représente un défi majeur pour la sécurité informatique moderne. Ce cours aborde les **menaces posées par l'informatique quantique** à la cryptographie actuelle et les **solutions post-quantiques** développées pour y faire face, en particulier l'algorithme **CRYSTALS-Kyber**.

Problématique centrale

L'avènement des ordinateurs quantiques menace de compromettre la sécurité des systèmes cryptographiques actuels. Cette situation nécessite une transition urgente vers des algorithmes **résistants aux attaques quantiques**.

Enjeux de sécurité

Les systèmes cryptographiques asymétriques actuels (RSA, ECC) reposent sur des problèmes mathématiques difficiles pour les ordinateurs classiques mais résolus efficacement par les ordinateurs quantiques.

Ampleur de la transition

La migration vers la cryptographie post-quantique constituera probablement la **plus grande transition cryptographique de l'histoire**, affectant :

- L'infrastructure Internet mondiale (HTTPS, TLS)
- Les systèmes bancaires et financiers
- Les communications gouvernementales
- Les dispositifs IoT et embarqués
- Les signatures numériques et certificats

Notations générales du document

Symboles mathématiques

- $\mathbb{Z}$: ensemble des entiers
- $\mathbb{R}$: ensemble des nombres réels
- $\mathbb{Z}_q$: entiers modulo q
- $\perp$: indépendance
- $\equiv$: équivalence ou congruence

Conventions de notation

- **Majuscules** : matrices ou ensembles (A, $\mathcal{X}$)
- **Minuscules** : vecteurs ou scalaires (s, e)
- **Gras** : vecteurs ($\mathbf{s}$, $\mathbf{e}$) (optionnel)

Fondements de l'Informatique Quantique

Informatique Classique vs Quantique

Architecture des ordinateurs classiques

Unité de base : le bit

Définition : Un **bit** (binary digit) est l'unité fondamentale d'information dans un ordinateur classique.

Caractéristiques : - Valeurs possibles : **0 ou 1** uniquement - Nombre **fini** d'états discrets - États physiques distingués par des propriétés **électriques ou magnétiques**

Opérations logiques

Les opérations sont effectuées via des **portes logiques** (AND, OR, NOT, XOR, etc.) qui manipulent les bits de manière déterministe.

Traitement de l'information

Pour n bits, il existe 2^n configurations possibles, mais le système ne peut être que dans **une seule** de ces configurations à un instant donné.

Architecture des ordinateurs quantiques

Unité fondamentale : le qubit

Définition : Un **qubit** (quantum bit) est un système quantique à deux niveaux qui peut exister dans une superposition de ses états de base.

Représentation mathématique :

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

où $\alpha, \beta \in \mathbb{C}$ avec $|\alpha|^2 + |\beta|^2 = 1$.

Propriétés quantiques fondamentales

Propriété 1 : Superposition

Un qubit peut exister dans un **nombre infini d'états** correspondant à des superpositions linéaires de ses états de base $|0\rangle$ et $|1\rangle$.

Conséquence : - Avant la mesure : le qubit est dans une superposition - Lors de la **mesure** : le qubit **s'effondre** vers l'un des états de base - Ce phénomène est appelé **décoherence**

Propriété 2 : Intrication (Entanglement)

Définition : Des qubits sont **intriqués** lorsque l'état de l'un ne peut être décrit indépendamment de l'état des autres.

Pour un système de n qubits intriqués : - Espace d'états : 2^n états de base - Croissance **exponentielle** avec n - Un système de 300 qubits aurait plus d'états que d'atomes dans l'univers observable

Parallélisme quantique

Concept fondamental : Un ordinateur quantique peut opérer sur **tous les états de base simultanément** grâce à la superposition.

Exemple : Avec 10 qubits en superposition, une opération quantique agit simultanément sur $2^{10} = 1024$ états.

Limitation importante : Bien que le calcul soit effectué sur tous les états, la mesure ne donne qu'**un seul résultat**. Des algorithmes spéciaux (comme ceux de Shor et Grover) exploitent des interférences quantiques pour amplifier les bonnes réponses.

Différence fondamentale

Représentation de l'information

Point clé : La différence principale entre informatique classique et quantique réside dans la **représentation de l'information** et non dans la vitesse brute de calcul.

Inspiration physique

L'informatique quantique s'inspire du **comportement des particules quantiques** à l'échelle atomique et subatomique.

Nécessité d'un matériel quantique

Important : Un ordinateur quantique **ne peut pas être simulé efficacement** sur un ordinateur classique pour des problèmes non triviaux, en raison de la croissance exponentielle de l'espace d'états.

Défis de l'Informatique Quantique

Défi 1 : Effondrement quantique (Collapsing)

Problème de la mesure

Nature du défi : Le qubit perd sa propriété de **superposition** lors de la mesure, s'effondrant vers un état classique.

Implications algorithmiques

Conséquence pratique : - Les mesures doivent être soigneusement **planifiées** dans les algorithmes - On ne peut pas "observer" l'état intermédiaire d'un calcul sans le détruire - Nécessite des techniques comme la **correction d'erreurs quantiques**

Stratégies de gestion

Les algorithmes quantiques sont conçus pour : - Minimiser le nombre de mesures - Utiliser des interférences quantiques pour amplifier les bonnes réponses - Mesurer uniquement à la fin du calcul

Défi 2 : Sensibilité environnementale

Nature de la fragilité

Problème : Les systèmes quantiques sont extrêmement **sensibles** à leur environnement : - Vibrations mécaniques - Rayonnement électromagnétique - Fluctuations thermiques - Interactions avec l'environnement (décohérence)

Exigences infrastructurelles

Conditions nécessaires : - Températures cryogéniques : souvent près du zéro absolu (15 mK pour les qubits supraconducteurs) - **Isolation électromagnétique** : blindage contre les interférences - **Vide poussé** : pour certaines technologies - **Stabilité mécanique** : tables anti-vibrations

Temps de cohérence

Limitation temporelle : Les qubits ne maintiennent leur état quantique que pendant un temps limité (temps de cohérence), typiquement : - Qubits supraconducteurs : 10-100 s - Ions piégés : secondes à minutes - Qubits topologiques (théoriques) : potentiellement beaucoup plus long

Défi 3 : Architecture des portes logiques

Distinction théorique vs physique

Portes quantiques logiques : - Opérations théoriques parfaites - Décrites mathématiquement par des matrices unitaires - Exemples : Hadamard, CNOT, Toffoli

Portes quantiques physiques : - Implémentation réelle avec du matériel - Soumises aux erreurs et au bruit - Fidélité limitée (typiquement 99-99.9%)

Impact sur les performances

Conséquences : - Nécessité de **correction d'erreurs quantiques** - Overhead important : besoin de nombreux qubits physiques pour un qubit logique - Influence directe sur la **profondeur des circuits** réalisables

Overhead de correction d'erreurs

Pour exécuter des algorithmes complexes comme Shor, on estime qu'il faut : - Des milliers de qubits logiques - Des millions de qubits physiques (avec correction d'erreurs) - Actuellement, les systèmes ont quelques centaines de qubits physiques

Défi 4 : Translation d'algorithmes

Non-transférabilité directe

Constat important : Les algorithmes classiques ne peuvent généralement **pas être traduits directement** en algorithmes quantiques.

Exemple : AES

Advanced Encryption Standard (AES) : - Translation directe **non faisable** efficacement - Nécessite une **refonte complète** de l'approche - Les accélérations quantiques pour AES sont limitées

Nécessité d'un nouveau paradigme

Approche requise : - Développer des **algorithmes appropriés** spécifiquement pour l'informatique quantique - Exploiter les propriétés quantiques (superposition, intrication) - Nouveau paradigme de programmation

État actuel

Situation : - Nombre limité d'algorithmes quantiques connus - Beaucoup de questions **non résolues** - Recherche active pour trouver de nouvelles applications

État de l'art et perspectives

Qubits actuels (2024-2025)

Technologies principales : - **Supraconducteurs** : IBM, Google (50-400 qubits) - **Ions piégés** : IonQ, Quantinuum (32-56 qubits) - **Photoniques** : Xanadu, PsiQuantum - **Atomes neutres** : QuEra, Pasqal

Horizon temporel

Estimations pour un ordinateur quantique cryptographiquement pertinent : - **Optimiste** : 10-15 ans - **Réaliste** : 15-30 ans - **Conservateur** : 30+ ans ou incertain

Facteurs d'incertitude : - Avancées technologiques imprévisibles - Percées théoriques possibles - Investissements massifs en cours

Menaces Quantiques pour la Cryptographie

Algorithmes Cryptographiques Menacés

Cryptographie asymétrique classique

Fondement de la sécurité actuelle

Base mathématique : Les systèmes cryptographiques asymétriques actuels reposent sur des problèmes mathématiques difficiles pour les ordinateurs **classiques** :

Problème 1 : Factorisation des grands nombres - **Utilisé dans** : RSA - **Difficulté classique** : exponentielle ($O(e^{n^{1/3}})$ avec GNFS) - **Difficulté quantique** : polynomiale (algorithme de Shor)

Problème 2 : Logarithme discret - **Utilisé dans** : Diffie-Hellman, DSA, ElGamal - **Difficulté classique** : exponentielle - **Difficulté quantique** : polynomiale (algorithme de Shor)

Problème 3 : Logarithme discret sur courbes elliptiques - **Utilisé dans** : ECDSA, ECDH - **Difficulté classique** : exponentielle - **Difficulité quantique** : polynomiale (variante de Shor)

Systèmes et protocoles affectés

Infrastructure à clé publique : - Chiffrement/déchiffrement asymétrique - Signatures numériques - Authentification des entités - Établissement de clés (key establishment) - Certificats numériques

Protocoles réseau menacés : - **TLS/SSL** : sécurisation des connexions web - **HTTPS** : navigation web sécurisée - **SSH** : accès distant sécurisé - **VPN** : réseaux privés virtuels - **PGP/GPG** : chiffrement d'emails - **Blockchain** : signatures de transactions

Cryptographie symétrique

Résistance relative

Situation différente : La cryptographie symétrique (comme AES) est **moins menacée** mais pas totalement sûre.

Impact de l'algorithme de Grover

Accélération quantique : L'algorithme de Grover offre une accélération **quadratique** pour la recherche exhaustive : - Complexité classique : $O(2^n)$ pour une clé de n bits - Complexité quantique : $O(2^{n/2})$

Conséquence pratique : Un ordinateur quantique réduit effectivement la sécurité de moitié.

Solution pour AES

Mitigation simple : Doubler la taille des clés suffit pour maintenir la sécurité : - AES-128 $\rightarrow$ AES-256 (retrouve niveau de sécurité équivalent) - AES-256 $\rightarrow$ AES-512 (si nécessaire à très long terme)

Conclusion : La cryptographie symétrique reste **viable** dans l'ère post-quantique avec des ajustements simples.

Algorithme de Shor : Menace Principale

Présentation de l'algorithme

Invention et contexte

Développé par : Peter Shor (1994)

Innovation majeure : Premier algorithme quantique à offrir une accélération **exponentielle** pour un problème d'intérêt pratique.

Problèmes résolus

Capacités de Shor : 1. **Factorisation** : Décomposer $N = p \times q$ en facteurs premiers 2. **Logarithme discret** : Résoudre $g^x \equiv h \pmod{p}$ 3. **Problème de Diffie-Hellman**

Complexité et performances

Complexité temporelle

Ordinateur classique (meilleur algorithme connu - GNFS) :

$$O\left(\exp\left((c + o(1))(\ln N)^{1/3}(\ln\ln N)^{2/3}\right)\right)$$

Ordinateur quantique (algorithme de Shor) :

$$O\left((\log N)^2(\log\log N)(\log\log\log N)\right)$$

Essentiellement **polynomial** en $\log N$ (le nombre de bits).

Exemple concret

Pour factoriser un nombre RSA-2048 (617 chiffres décimaux) : - **Classique** : > âge de l'univers avec technologie actuelle - **Quantique** : quelques heures à jours (avec ordinateur quantique suffisant)

Exigences matérielles

Qubits nécessaires

Estimations : Pour casser RSA-2048 avec l'algorithme de Shor : - **Qubits logiques** : ~3000-4000 - **Qubits physiques** (avec correction d'erreurs) : 10-20 millions

État actuel : Les systèmes existants ont ~100-1000 qubits de qualité limitée.

Autres ressources

Besoins supplémentaires : - Temps de cohérence suffisant - Fidélité des portes élevée - Capacité de correction d'erreurs

Scénario "Apocalypse Quantique"

Impact sur l'infrastructure Internet

Effondrement de la confiance

Conséquence majeure : Si un ordinateur quantique cryptographiquement pertinent existait aujourd'hui :

"L'Internet tel que nous le connaissons n'existerait plus"

Systèmes compromis

Services affectés : - **Banques en ligne** : transactions financières - **E-commerce** : paiements sécurisés - **Communications** : emails, messageries - **Cloud** : stockage et calcul - **Gouvernements** : services publics numériques - **Santé** : dossiers médicaux

Menace "Harvest Now, Decrypt Later"

Concept de la menace

Stratégie adversaire : Certains acteurs malveillants pourraient : 1. **Aujourd'hui** : capturer et stocker les communications chiffrées 2. **Future** : déchiffrer avec un ordinateur quantique

Données sensibles à long terme

Particulièrement préoccupant pour : - Secrets d'État à long terme - Données biométriques - Informations médicales - Propriété intellectuelle - Données personnelles sensibles

Urgence de la transition

Implication : La transition vers le post-quantique est urgente **maintenant**, même si les ordinateurs quantiques sont à 10-20 ans.

Algorithme de Grover et Impact Limité

Présentation de Grover

Nature de l'algorithme

Développé par : Lov Grover (1996)

Fonction : Accélération de la **recherche dans une base de données non structurée**.

Accélération quadratique

Performance : - **Classique** : $O(N)$ opérations pour chercher dans N éléments - **Quantique** : $O(\sqrt{N})$ opérations

Application à la cryptographie

Impact sur les chiffrements symétriques

Pour une clé de n bits : - **Recherche exhaustive classique** : 2^n essais - **Recherche avec Grover** : $2^{n/2}$ essais

Réduction de sécurité : Équivaut à diviser par 2 la longueur effective de la clé.

Mitigation simple

Solution : Augmenter la taille des clés : - AES-128 → sécurité effective de 64 bits contre Grover - AES-256 → sécurité effective de 128 bits contre Grover

Conclusion : Ajustement **beaucoup plus simple** que pour la cryptographie asymétrique.

Limitations pratiques

Overhead important

Difficultés d'implémentation : - Nécessite un nombre important de qubits - Requiert de nombreuses opérations quantiques - Circuit profond → sensible aux erreurs

Impact limité en pratique

Pour la cryptographie symétrique, l'avantage de Grover est **moins dramatique** que celui de Shor pour l'asymétrique.

Fondements Mathématiques

Notations et Structures Algébriques

Ensembles de base

Entiers modulo q

Définition :

$$\mathbb{Z}_q = \{0, 1, 2, \dots, q - 1\}$$

Description : Ensemble des **entiers modulo** q avec opérations d'addition et multiplication modulo q.

Propriétés : - Si q est premier : $\mathbb{Z}_q$ est un **corps** (field) - Sinon : $\mathbb{Z}_q$ est un **anneau** (ring)

Opérations : - Addition : $(a + b) \bmod q$ - Multiplication : $(a \cdot b) \bmod q$

Polynômes à coefficients dans $\mathbb{Z}_q$

Notation :

$$\mathbb{Z}_q[X]$$

Définition : Ensemble des polynômes avec coefficients dans $\mathbb{Z}_q$:

$$p(X) = a_0 + a_1X + a_2X^2 + \cdots + a_dX^d$$

où $a_i \in \mathbb{Z}_q$.

Propriété importante : Le degré d n'est **pas borné** (peut être arbitrairement grand).

Opérations : - Addition : coefficient par coefficient - Multiplication : convolution des coefficients, modulo q

Anneaux de polynômes

Anneau de polynômes quotient

Notation :

$$R_q = \mathbb{Z}_q[X]/(X^n + 1)$$

Définition : Anneau des polynômes de degré **strictement inférieur à** n avec coefficients dans $\mathbb{Z}_q$, où la multiplication est effectuée modulo $X^n + 1$.

Représentation d'un élément :

$$p(X) = a_0 + a_1X + \cdots + a_{n-1}X^{n-1}, \quad a_i \in \mathbb{Z}_q$$

Réduction modulo $X^n + 1$

Règle de réduction :

$$X^n \equiv -1 \pmod{X^n + 1}$$

Conséquence :

$$X^{n+k} \equiv -X^k \pmod{X^n + 1}$$

Exemple avec $n = 4$:

$$X^4 \equiv -1$$
$$X^5 \equiv -X$$
$$X^6 \equiv -X^2$$

Multiplication dans R_q

Procédure : 1. Multiplier les polynômes normalement 2. Réduire modulo $X^n + 1$ (ramener degrés $\geq n$) 3. Réduire coefficients modulo q

Exemple : Pour $n = 4$, $q = 17$:

$$(1 + X) \cdot (1 + X^2) = 1 + X + X^2 + X^3$$

$$\begin{gathered}(X^3 + X^2) \cdot (X^2 + X) = X^5 + X^4 + X^4 + X^3 \\ \equiv -X - 1 - 1 + X^3 = X^3 - X - 2 \pmod{X^4 + 1}\end{gathered}$$

Modules

Définition d'un module

Notation :

$$R_q^k$$

Définition : Un **module de rang** k sur R_q : ensemble des k-uplets de polynômes dans R_q.

Élément type :

$$\mathbf{v} = \begin{pmatrix} v_1(X) \\ v_2(X) \\ \vdots \\ v_k(X) \end{pmatrix}, \quad v_i \in R_q$$

Opérations sur les modules

Addition :

$$\mathbf{v} + \mathbf{w} = \begin{pmatrix} v_1 + w_1 \\ \vdots \\ v_k + w_k \end{pmatrix}$$

Multiplication scalaire (par $r \in R_q$) :

$$r \cdot \mathbf{v} = \begin{pmatrix} r \cdot v_1 \\ \vdots \\ r \cdot v_k \end{pmatrix}$$

Matrices sur R_q

Matrice $m \times k$:

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mk} \end{pmatrix}, \quad a_{ij} \in R_q$$

Multiplication matrice-vecteur :

$$\mathbf{A} \cdot \mathbf{v} = \begin{pmatrix} \sum_{j=1}^{k} a_{1j} v_j \\ \vdots \\ \sum_{j=1}^{k} a_{mj} v_j \end{pmatrix}$$

où les multiplications et additions se font dans R_q.

Normes et Polynômes "Petits"

Norme infinie

Définition pour les entiers

Pour $a \in \mathbb{Z}_q$:

$$\|a\|_\infty = \min\{|a|, q - |a|\}$$

C'est la distance minimale à 0 en considérant la réduction modulo q symétrique.

Exemple avec $q = 17$: - $\|3\|_\infty = 3$ - $\|15\|_\infty = \min\{15, 2\} = 2$ (car $15 \equiv -2 \pmod{17}$)

Définition pour les polynômes

Pour $p(X) = \sum_{i=0}^{n-1} a_i X^i \in R_q$:

$$\|p\|_\infty = \max_{0 \leq i < n} \|a_i\|_\infty$$

C'est le **maximum des normes** des coefficients.

Définition pour les vecteurs/modules

Pour $\mathbf{v} = (v_1, \dots, v_k) \in R_q^k$:

$$\|\mathbf{v}\|_\infty = \max_{1 \leq i \leq k} \|v_i\|_\infty$$

Polynômes "petits"

Concept

Définition intuitive : Un polynôme est dit **"petit"** si ses coefficients ont une **magnitude faible** (proches de 0).

Formalisation : Un polynôme $p \in R_q$ est "η-petit" si :

$$\|p\|_\infty \leq \eta$$

où η est typiquement très petit devant q (par exemple $\eta = 2$ ou 3, alors que $q = 3329$).

Rôle dans Kyber

Importance cruciale : Les polynômes petits jouent un rôle **fondamental** dans la sécurité de Kyber : - **Clé secrète** : vecteur de polynômes petits - **Erreurs** : polynômes petits ajoutés pour la sécurité - **Masquage** : les petites erreurs masquent le secret sans empêcher le déchiffrement

Distributions de coefficients

Distribution centrée binomiale : Dans Kyber, les coefficients des polynômes petits sont tirés selon une **distribution centrée binomiale** β_η, qui produit des valeurs dans $\{-\eta, \dots, \eta\}$ avec une distribution proche d'une gaussienne discrète.

Problèmes LWE et Variantes

Learning Without Errors (Configuration de base)

Problème sans erreurs

Configuration : Alice possède un **vecteur secret** $\mathbf{s} = (s_1, \dots, s_n) \in \mathbb{Z}_q^n$.

Information publique : Ensemble d'équations linéaires :

$$\mathbf{a}_i \cdot \mathbf{s} = b_i \pmod q, \quad i = 1, \dots, m$$

où les $\mathbf{a}_i \in \mathbb{Z}_q^n$ sont des vecteurs publics (aléatoires).

Formulation matricielle :

$$\mathbf{A}\mathbf{s} = \mathbf{b} \pmod q$$

où $\mathbf{A}$ est une matrice $m \times n$ et $\mathbf{b}$ est un vecteur de dimension m.

Facilité de résolution

Objectif : Trouver le vecteur secret $\mathbf{s}$ à partir de $(\mathbf{A}, \mathbf{b})$.

Méthode : Élimination de Gauss (ou méthodes d'algèbre linéaire).

Complexité : Polynomial en n et m (typiquement $O(n^3)$ ou moins).

Conclusion : **Facile à résoudre** → ne fournit **aucune sécurité**.

Learning With Errors (LWE)

Ajout d'erreurs aléatoires

Modification : Ajouter une **petite erreur aléatoire** e_i à chaque équation :

$$\mathbf{a}_i \cdot \mathbf{s} + e_i = b_i \pmod q$$

où e_i est tiré d'une distribution d'erreur (typiquement gaussienne discrète ou distribution centrée).

Formulation matricielle :

$$\mathbf{A}\mathbf{s} + \mathbf{e} = \mathbf{b} \pmod q$$

où $\mathbf{e} = (e_1, \dots, e_m)$ est un vecteur d'erreurs.

Difficulté du problème

Problème LWE (version recherche) : Étant donné $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q)$, retrouver $\mathbf{s}$.

Difficulté : Un système **surdéterminé** $(m > n)$ d'équations avec erreurs n'a presque certainement **pas de solution exacte**.

Méthodes classiques : - Élimination de Gauss : ne fonctionne plus - Méthodes d'approximation : très coûteuses

Conclusion : Le secret $\mathbf{s}$ est vraiment **protégé** par les erreurs.

Paramètres importants

Dimension n : taille du vecteur secret **Modulo** q : souvent premier ou puissance de 2 **Nombre d'échantillons** m : nombre d'équations **Distribution d'erreur** : détermine la "taille" des erreurs

Compromis : - Erreurs trop petites → problème trop facile - Erreurs trop grandes → déchiffrement impossible - Il existe un équilibre optimal

Variante : Ring-LWE (R-LWE)

Motivation

Problème de LWE standard : - Clés publiques très grandes : $m \times n$ coefficients - Opérations coûteuses : multiplication matrice-vecteur

Solution : Utiliser des **anneaux de polynômes** pour réduire les tailles.

Définition de Ring-LWE

Configuration : - Anneau : $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ - Secret : $s \in R_q$ (polynôme) - Échantillon R-LWE : $(a, b = a \cdot s + e)$ où $a \in R_q$ aléatoire, $e \in R_q$ petit

Réduction de taille : Un échantillon Ring-LWE = 2 polynômes = $2n$ coefficients vs LWE standard : besoin de m échantillons de n coefficients chacun

Problème R-LWE

Version recherche : Étant donné des paires $(a_i, b_i = a_i s + e_i)$, retrouver s.

Version décision : Distinguer $(a, as + e)$ d'une paire (a, b) aléatoire.

Module-LWE (MLWE)

Généralisation de R-LWE

Motivation : - Ring-LWE : sécurité liée à la structure d'anneau spécifique - Module-LWE : plus flexible, entre LWE et R-LWE

Compromis : - $k = 1$: équivalent à Ring-LWE - $k = n$: proche de LWE standard - k intermédiaire : équilibre taille/sécurité

Définition formelle

Configuration : - Module : R_q^k (vecteurs de k polynômes) - Secret : $\mathbf{s} = (s_1, \dots, s_k) \in R_q^k$ - Matrice publique : $\mathbf{A} \in R_q^{m \times k}$ - Vecteur d'erreur : $\mathbf{e} \in R_q^m$

Échantillon MLWE :

$$(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e})$$

où toutes les opérations sont dans le module R_q^m.

Problème MLWE (recherche)

Objectif : Étant donné $(\mathbf{A}, \mathbf{t})$, retrouver le vecteur secret $\mathbf{s}$.

Difficulté : **Aucun algorithme efficace connu** (classique ou quantique) pour résoudre MLWE avec paramètres appropriés.

Type : MLWE est un **problème de recherche** (search problem).

Flexibilité des paramètres

Paramètre de rang k : - Petit k (2-3) : clés plus petites, sécurité basée sur structure algébrique - Grand k : clés plus grandes, sécurité plus proche de LWE général

Kyber utilise : - Kyber512 : $k = 2$ - Kyber768 : $k = 3$ - Kyber1024 : $k = 4$

Decision Module-LWE (D-MLWE)

Version décision du problème

Même configuration que MLWE : - Module R_q^k - Matrice $\mathbf{A} \in R_q^{m \times k}$

Instance D-MLWE : Une paire $(\mathbf{A}, \mathbf{t})$ où $\mathbf{t} \in R_q^m$.

Question posée

Problème de décision : Déterminer si $(\mathbf{A}, \mathbf{t})$ est : - **Instance valide de MLWE** : $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$ pour un secret $\mathbf{s}$ et des erreurs $\mathbf{e}$ - Ou **paire aléatoire** : $\mathbf{t}$ choisi uniformément dans R_q^m

Importance pour la cryptographie

Propriété de sécurité : Si D-MLWE est difficile, alors $(\mathbf{A}, \mathbf{t})$ ne révèle **aucune information** sur $\mathbf{s}$.

Conséquence : Permet de prouver que la **clé publique** ne fuit pas d'information sur la **clé secrète**.

Difficulté

Aucun algorithme efficace connu pour résoudre D-MLWE (avec bons paramètres).

Type : D-MLWE est un **problème de décision** (decision problem).

Comparaison des variantes

Aspect	MLWE (recherche)	D-MLWE (décision)
Type	Problème de recherche	Problème de décision
Objectif	Trouver $\mathbf{s}$	Distinguer valide/aléatoire
Sortie	Vecteur secret ou échec	Bit (valide ou non)
Utilisation crypto	Construction de schémas	Preuves de sécurité
Relation	Plus difficile	Plus facile (en principe)

Relation importante : Dans la pratique, MLWE et D-MLWE semblent avoir une difficulté similaire pour les paramètres utilisés.

Réseaux Euclidiens (Lattices)

Définition et Propriétés

Définition mathématique

Réseau (Lattice) : Soit $\{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n\}$ une famille de n vecteurs **linéairement indépendants** dans $\mathbb{R}^n$ (la **base** du réseau).

Le **réseau** $\mathcal{L}$ engendré par cette base est :

$$\mathcal{L}(\mathbf{b}_1, \dots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} z_i \mathbf{b}_i \mid z_i \in \mathbb{Z} \right\}$$

En termes simples : Ensemble de toutes les **combinaisons linéaires entières** des vecteurs de base.

Représentation matricielle

Matrice de base :

$$\mathbf{B} = [\mathbf{b}_1 \mid \mathbf{b}_2 \mid \cdots \mid \mathbf{b}_n] \in \mathbb{R}^{n \times n}$$

Le réseau :

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{B}\mathbf{z} \mid \mathbf{z} \in \mathbb{Z}^n\}$$

Propriétés fondamentales

Non-unicité de la base : Un même réseau peut avoir **plusieurs bases différentes**.

Déterminant du réseau :

$$\det(\mathcal{L}) = |\det(\mathbf{B})|$$

Cette quantité est **indépendante** du choix de base.

Volume fondamental : Le déterminant représente le volume du **parallélépipède fondamental**.

Exemple en dimension 2

Base :

$$\mathbf{b}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \mathbf{b}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Réseau :

$$\mathcal{L} = \mathbb{Z}^2$$

C'est le réseau le plus simple : les points à coordonnées entières dans le plan.

Problèmes Classiques sur les Réseaux

Shortest Vector Problem (SVP)

Énoncé : Étant donné un réseau $\mathcal{L}$ (via une base), trouver un vecteur **non nul** dans $\mathcal{L}$ avec **norme euclidienne minimale**.

Formulation mathématique :

$$\text{Trouver } \mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\} \text{ tel que } \|\mathbf{v}\|_2 = \min_{\mathbf{w} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\mathbf{w}\|_2$$

Variante approchée (approx-SVP) : Trouver $\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}$ tel que :

$$\|\mathbf{v}\|_2 \leq \gamma \cdot \lambda_1(\mathcal{L})$$

où $\lambda_1(\mathcal{L})$ est la longueur du plus court vecteur et $\gamma > 1$ est le facteur d'approximation.

Closest Vector Problem (CVP)

Énoncé : Étant donné un réseau $\mathcal{L}$ et un point cible $\mathbf{t} \in \mathbb{R}^n$, trouver le vecteur du réseau **le plus proche** de $\mathbf{t}$.

Formulation mathématique :

$$\text{Trouver } \mathbf{v} \in \mathcal{L} \text{ tel que } \|\mathbf{t} - \mathbf{v}\|_2 = \min_{\mathbf{w} \in \mathcal{L}} \|\mathbf{t} - \mathbf{w}\|_2$$

Variante approchée (approx-CVP) : Trouver $\mathbf{v} \in \mathcal{L}$ tel que :

$$\|\mathbf{t} - \mathbf{v}\|_2 \leq \gamma \cdot \min_{\mathbf{w} \in \mathcal{L}} \|\mathbf{t} - \mathbf{w}\|_2$$

Relation entre SVP et CVP

SVP comme cas particulier : SVP est un **cas particulier** de CVP avec $\mathbf{t} = \mathbf{0}$.

Réductions : - CVP peut se réduire à SVP (avec un certain overhead) - En pratique, les deux problèmes ont une difficulté similaire

Complexité des Problèmes de Réseaux

NP-difficulté

Résultats théoriques : - SVP exact : NP-difficile sous hypothèse de randomisation - **CVP exact** : NP-difficile - Même les **versions approchées** (avec petit facteur d'approximation) sont difficiles

Difficulté en moyenne (Average-Case Hardness)

Propriété remarquable : Les problèmes de réseaux sont **difficiles en moyenne**, pas seulement dans le pire cas.

Distinction importante : - Plupart des problèmes NP : faciles "en moyenne", difficiles seulement dans le pire cas - **Problèmes de réseaux** : difficiles **même en moyenne**

Implication cryptographique : Cette propriété est **cruciale** pour la cryptographie : - Les instances aléatoires sont difficiles - Pas besoin de générer des instances spéciales "difficiles"

Résistance aux attaques quantiques

Propriété fondamentale : Les meilleurs algorithmes connus pour résoudre SVP et CVP, **même sur ordinateurs quantiques**, restent **exponentiels**.

Algorithmes quantiques connus : - Accélération polynomiale au mieux - Pas de percée comme celle de Shor pour la factorisation

Conclusion : Les réseaux offrent une **résistance quantique** intrinsèque.

Lien avec MLWE et D-MLWE

Reformulation comme problèmes de réseaux

Théorème (informel) : Les problèmes MLWE et D-MLWE peuvent être **reformulés** comme des variantes de problèmes de réseaux (SVP, CVP).

Construction du réseau : À partir d'une instance $(\mathbf{A}, \mathbf{t})$ de MLWE, on peut construire un réseau $\mathcal{L}$ tel que : - Résoudre MLWE   résoudre CVP dans $\mathcal{L}$

Réduction de difficulté

Résultat théorique important : La difficulté **moyenne** de MLWE est au moins aussi grande que la difficulté **quantique du pire cas** de certains problèmes de réseaux (comme approx-SVP).

Formulation technique :

$$\text{Worst-case quantum SVP}_\gamma \leq_{\text{reduction}} \text{Average-case MLWE}$$

pour un facteur d'approximation γ approprié.

Intérêt pratique pour la sécurité

Avantages de ce lien :

1. **Évaluation de difficulté** : Les problèmes de réseaux sont **bien étudiés** depuis des décennies
2. **Confiance mathématique** : Compréhension approfondie de leur structure
3. **Résistance quantique** : Aucune attaque quantique efficace connue
4. **Base solide** : Fondement théorique rigoureux pour la sécurité

Conclusion : Ce lien fournit une **base solide** pour affirmer que Kyber est sûr contre les ordinateurs quantiques.

CRYSTALS-Kyber : Architecture et Fonctionnement

Vue d'Ensemble de Kyber

Nature et Classification

Type de système

CRYSTALS-Kyber : - KEM : Key Encapsulation Mechanism (mécanisme d'encapsulation de clés) - **Système cryptographique asymétrique - Post-quantique** : résistant aux attaques quantiques

Paradigme cryptographique

Basé sur les réseaux (Lattice-based) : Kyber appartient à la famille des cryptosystèmes basés sur les problèmes de réseaux euclidiens.

Caractéristiques : - Opère sur des **anneaux de polynômes** R_q - Utilise des **modules** R_q^k - S'appuie sur des **polynômes "petits"**

Fondements de Sécurité

Problèmes mathématiques sous-jacents

Sécurité basée sur : 1. **Module-LWE (MLWE)** : problème de recherche 2. **Decision Module-LWE (D-MLWE)** : problème de décision

Hypothèse de sécurité : Si MLWE et D-MLWE sont **intraitable** (computationally hard), alors Kyber est sûr.

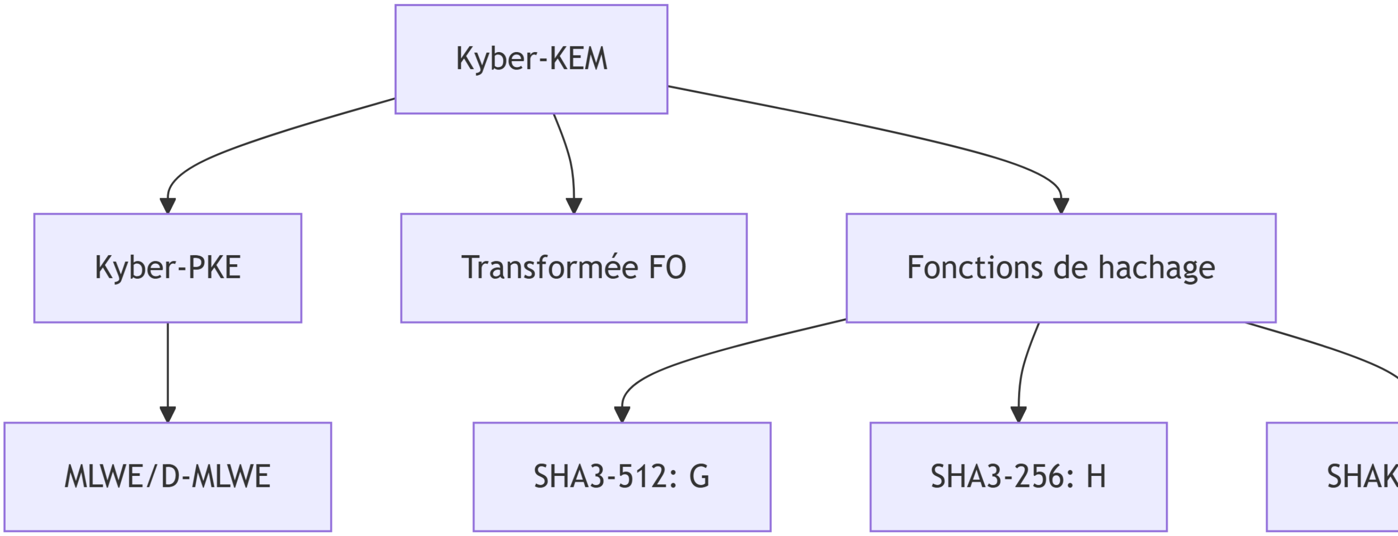
Niveau de sécurité

IND-CCA2 secure : Indistinguishability under Adaptive Chosen Ciphertext Attack

C'est la **plus forte notion de sécurité** pour un KEM : - Résistant aux attaques à texte chiffré choisi - Résistant même quand l'adversaire peut interroger un oracle de déchiffrement

Structure en Couches

Architecture modulaire



Composantes principales : 1. **Kyber-PKE** : chiffrement à clé publique de base 2. **Transformée de Fujisaki-Okamoto (FO)** : conversion PKE → KEM 3. **Fonctions de hachage** : dérivation de clés et randomness

Kyber-PKE : Le cœur cryptographique

Rôle : Fournit les opérations de chiffrement de base.

Opérations : - **KeyGen** : génération de paire de clés - **Encrypt** : chiffrement d'un message - **Decrypt** : déchiffrement

Limitation : Kyber-PKE seul n'est que **IND-CPA** (secure against chosen plaintext attack).

Transformation FO : Renforcement de sécurité

Transformation de Fujisaki-Okamoto : Technique générique pour transformer un PKE (IND-CPA) en KEM (IND-CCA2).

Méthode : - Ajoute des vérifications cryptographiques - Utilise le hachage du message comme randomness - Permet la détection de textes chiffrés malformés

Résultat : Kyber-KEM atteint la sécurité **IND-CCA2**.

Kyber-PKE : Chiffrement de Base

Espace de Texte Clair

Définition du plaintext space

Messages :

$$\mathcal{M} = \{0,1\}^n \subset R_q$$

Interprétation : - Messages = vecteurs de n bits - Encodés comme polynômes dans R_q - Coefficients $\{0,1\}$

Encodage des bits

Procédure d'encodage : Un message $m = (m_0, m_1, \dots, m_{n-1}) \in \{0,1\}^n$ est encodé comme :

$$\text{Encode}(m) = \sum_{i=0}^{n-1} m_i X^i \cdot \left\lfloor \frac{q}{2} \right\rfloor \in R_q$$

Principe : - Bit 0 → coefficient 0 - Bit 1 → coefficient $\lfloor q/2 \rfloor$ (milieu de $\mathbb{Z}_q$)

Décodage des bits

Procédure de décodage : Pour un polynôme $p(X) = \sum a_i X^i \in R_q$:

$$\text{Decode}(p)_i = \begin{cases} 0 & \text{si } |a_i| < q/4 \\ 1 & \text{si } |a_i - \lfloor q/2 \rfloor| < q/4 \end{cases}$$

Tolérance aux erreurs : Le décodage réussit si les erreurs sont $< q/4$ en magnitude.

Propriété Importante : Échec de Déchiffrement

Possibilité d'échec

Avertissement : Le déchiffrement de Kyber-PKE peut **échouer** avec une **petite probabilité**.

Raison de l'échec

Cause : Les **erreurs ajoutées** pour la sécurité peuvent parfois être **trop grandes**.

Mécanisme : 1. Chiffrement ajoute des termes d'erreur e_1, e_2 2. Déchiffrement calcule $m' +$ (erreurs cumulées) 3. Si les erreurs dépassent $q/4$, le décodage échoue

Gestion de la probabilité d'échec

Contrôle par paramètres : La probabilité d'échec est rendue **négligeable** par le choix approprié de : - Modulo q (assez grand) - Distribution d'erreur (assez petite) - Dimension n et rang k

Valeurs typiques : Probabilité d'échec $< 2^{-128}$ ou 2^{-256} (astronomiquement faible).

Schéma Kyber-PKE

Génération de clés (KeyGen)

Entrée : Paramètres (n, k, q, η_1)

Procédure : 1. Générer une matrice aléatoire $\mathbf{A} \in R_q^{k \times k}$ 2. Tirer un vecteur secret $\mathbf{s} \in R_q^k$ avec petits coefficients (distribution β_{η_1}) 3. Tirer un vecteur d'erreur $\mathbf{e} \in R_q^k$ avec petits coefficients 4. Calculer $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e} \in R_q^k$

Sortie : - **Clé publique** : $pk = (\mathbf{A}, \mathbf{t})$ - **Clé secrète** : $sk = \mathbf{s}$

Chiffrement (Encrypt)

Entrée : - Clé publique $pk = (\mathbf{A}, \mathbf{t})$ - Message $m \in \{0, 1\}^n$ - Randomness (pièces aléatoires)

Procédure : 1. Tirer un vecteur aléatoire court $\mathbf{r} \in R_q^k$ (distribution β_{η_1}) 2. Tirer des erreurs $\mathbf{e}_1 \in R_q^k$, $e_2 \in R_q$ (distribution β_{η_2}) 3. Calculer : - $\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1 \in R_q^k$ - $v = \mathbf{t}^T\mathbf{r} + e_2 + \text{Encode}(m) \in R_q$

Sortie : - **Ciphertext** : $c = (\mathbf{u}, v)$

Déchiffrement (Decrypt)

Entrée : - Clé secrète $sk = \mathbf{s}$ - Ciphertext $c = (\mathbf{u}, v)$

Procédure : 1. Calculer $w = v - \mathbf{s}^T\mathbf{u} \in R_q$ 2. Décoder : $m' = \text{Decode}(w)$

Sortie : - Message m' (si déchiffrement réussit) - ou $\perp$ (symbole d'échec)

Correction du déchiffrement

Analyse :

$$\begin{aligned} w &= v - \mathbf{s}^T\mathbf{u} \\ &= (\mathbf{t}^T\mathbf{r} + e_2 + \text{Encode}(m)) - \mathbf{s}^T(\mathbf{A}^T\mathbf{r} + \mathbf{e}_1) \\ &= (\mathbf{A}\mathbf{s} + \mathbf{e})^T\mathbf{r} + e_2 + \text{Encode}(m) - \mathbf{s}^T\mathbf{A}^T\mathbf{r} - \mathbf{s}^T\mathbf{e}_1 \\ &= \mathbf{e}^T\mathbf{r} + e_2 - \mathbf{s}^T\mathbf{e}_1 + \text{Encode}(m) \\ &= \text{Encode}(m) + \underbrace{(\mathbf{e}^T\mathbf{r} + e_2 - \mathbf{s}^T\mathbf{e}_1)}_{\text{erreur totale}} \end{aligned}$$

Condition de succès : Si l'erreur totale reste $< q/4$ en norme infinie, le décodage récupère m correctement.

Kyber-KEM : Mécanisme Complet

Fonctions de Hachage Utilisées

Suite de fonctions SHA-3

Kyber utilise des fonctions de la famille **SHA-3** (Keccak), standardisées par le NIST.

Fonction G : SHA3-512

Algorithme : SHA3-512

Usage dans Kyber : - Dérivation de clés longues (512 bits) - Génération de randomness pour le chiffrement - Hachage de la clé publique

Propriétés : - Sortie : 512 bits (64 octets) - Résistance aux collisions : 256 bits de sécurité

Fonction H : SHA3-256

Algorithme : SHA3-256

Usage dans Kyber : - Hachage standard du ciphertext - Dérivation de la clé partagée finale - Vérification d'intégrité

Propriétés : - Sortie : 256 bits (32 octets) - Résistance aux collisions : 128 bits de sécurité

Fonction J : SHAKE256

Algorithme : SHAKE256 (XOF - eXtendable Output Function)

Usage dans Kyber : - Génération de la matrice publique **A** (déterministe à partir d'une graine) - Expansion de petites graines en longues séquences pseudo-aléatoires

Propriétés : - Sortie de longueur **variable** - Peut générer autant d'octets que nécessaire - Basé sur Keccak

Structure de Kyber-KEM

Opérations principales

1. Génération de clés (KeyGen)

Entrée : Paramètres de sécurité

Sortie : - Clé publique pk - Clé secrète sk (inclut pk, le secret PKE, et des informations auxiliaires)

Procédure : 1. Générer $(pk_{PKE}, sk_{PKE}) \leftarrow$ Kyber-PKE.KeyGen() 2. Calculer $h = H(pk_{PKE})$ 3. Tirer une graine aléatoire z 4. Définir $sk = (sk_{PKE}, pk_{PKE}, h, z)$ 5. Retourner $(pk = pk_{PKE}, sk)$

2. Encapsulation

Entrée : - Clé publique pk

Sortie : - Ciphertext c - Clé partagée K

Procédure : 1. Tirer un message aléatoire $m \in \{0,1\}^{256}$ 2. Calculer $(\bar{K}, r) = G(m \| H(pk))$ (dérivation de clé et randomness) 3. Chiffrer : $c \leftarrow$ Kyber-PKE.Encrypt$(pk, m; r)$ 4. Calculer la clé partagée : $K = H(\bar{K} \| c)$ 5. Retourner (c, K)

3. Decapsulation

Entrée : - Clé secrète $sk = (sk_{PKE}, pk, h, z)$ - Ciphertext c

Sortie : - Clé partagée K

Procédure : 1. Déchiffrer : $m' \leftarrow$ Kyber-PKE.Decrypt(sk_{PKE}, c) 2. Calculer $(\bar{K}', r') = G(m' \| h)$ 3. Rechiffrer : $c' \leftarrow$ Kyber-PKE.Encrypt$(pk, m'; r')$ 4. Si $c = c'$: - Retourner $K = H(\bar{K}' \| c)$ (succès) 5. Sinon : - Retourner $K = H(z \| c)$ (rejet implicite)

Transformation FO et vérification

Principe du rechiffrement : La transformation de Fujisaki-Okamoto nécessite de **rechiffrer** le message déchiffré et de **comparer** avec le ciphertext reçu.

Détection de manipulation : Si $c \neq c'$, cela indique : - Une attaque active (ciphertext modifié) - Ou une erreur de transmission

Rejet implicite : Au lieu de renvoyer une erreur, Kyber retourne une clé pseudo-aléatoire dépendant de z (la graine secrète) et c.

Sécurité : Cette approche empêche les attaques par oracle de déchiffrement (CCA2).

Propriétés de Sécurité

Plaintext Awareness

Concept : Conscience du texte clair (plaintext awareness) signifie qu'un adversaire ne peut créer un ciphertext valide sans "connaître" le plaintext correspondant.

Dans Kyber : La transformation FO force l'adversaire à utiliser Encrypt honnêtement, car toute manipulation est détectée par le rechiffrement.

IND-CCA2 Security

Implication : Plaintext awareness **IND-CCA2** security

Définition IND-CCA2 : Un adversaire ne peut pas distinguer entre : - Encapsulation d'un message choisi - Encapsulation d'un message aléatoire

Même avec accès à un **oracle de decapsulation** (sauf pour le ciphertext challenge).

Niveau de sécurité : C'est la **plus forte notion** pour un KEM.

Probabilité d'échec

Dans Kyber-KEM : La decapsulation peut théoriquement échouer si Kyber-PKE.Decrypt échoue.

Gestion : - Paramètres choisis pour que la probabilité soit **négligeable** ($< 2^{-128}$) - En pratique, n'arrive presque jamais

Comportement en cas d'échec : Le rejet implicite retourne une clé dérivée de z, pas d'erreur explicite.

Paramètres de Kyber

Niveaux de Sécurité

Kyber propose **trois variantes** correspondant à différents niveaux de sécurité NIST :

Kyber512

Paramètres : - $n = 256$ - $k = 2$ - $q = 3329$ - $\eta_1 = 3$, $\eta_2 = 2$

Sécurité : - NIST niveau 1 : équivalent à AES-128 - Résiste à ~2^{143} opérations classiques - Résiste à ~2^{120} opérations quantiques (MAXDEPTH)

Tailles : - Clé publique : 800 octets - Clé secrète : 1632 octets - Ciphertext : 768 octets

Kyber768

Paramètres : - $n = 256$ - $k = 3$ - $q = 3329$ - $\eta_1 = 2$, $\eta_2 = 2$

Sécurité : - NIST niveau 3 : équivalent à AES-192 - Résiste à ~2^{207} opérations classiques - Résiste à ~2^{174} opérations quantiques

Tailles : - Clé publique : 1184 octets - Clé secrète : 2400 octets - Ciphertext : 1088 octets

Kyber1024

Paramètres : - $n = 256$ - $k = 4$ - $q = 3329$ - $\eta_1 = 2$, $\eta_2 = 2$

Sécurité : - NIST niveau 5 : équivalent à AES-256 - Résiste à ~2^{272} opérations classiques - Résiste à ~2^{229} opérations quantiques

Tailles : - Clé publique : 1568 octets - Clé secrète : 3168 octets - Ciphertext : 1568 octets

Choix des Paramètres

Paramètre n : degré des polynômes

Valeur fixe : $n = 256$ pour toutes les variantes

Raison : - Permet des implémentations efficaces (FFT, NTT) - Dimension suffisante pour la sécurité - $X^{256} + 1$ est le polynôme cyclotomique approprié

Paramètre k : rang du module

Variable selon sécurité : - $k = 2$: Kyber512 - $k = 3$: Kyber768 - $k = 4$: Kyber1024

Compromis : - Plus k est grand → plus de sécurité - Plus k est grand → clés et ciphertexts plus grands

Paramètre q : modulo

Valeur fixe : $q = 3329 = 13 \times 256 + 1$ (nombre premier)

Propriétés : - Premier pour avoir un corps $\mathbb{Z}_q$ - Forme $q \equiv 1 \pmod{2n}$ permet NTT efficace - Assez grand pour supporter les erreurs

Paramètres η_1, η_2 : distributions d'erreurs

Valeurs : - $\eta_1 \in \{2, 3\}$: pour les secrets et erreurs KeyGen/Encrypt - $\eta_2 = 2$: pour les erreurs de chiffrement additionnelles

Distribution centrée binomiale β_η : Produit des coefficients dans $\{-\eta, \dots, \eta\}$ avec distribution proche d'une gaussienne discrète.

Compromis : - η plus petit → meilleure probabilité de déchiffrement correct - η plus grand → plus de sécurité

Tableau Récapitulatif

Paramètre	Kyber512	Kyber768	Kyber1024
Niveau NIST	1	3	5
Équivalent	AES-128	AES-192	AES-256
k	2	3	4
η_1	3	2	2
Clé publique	800 B	1184 B	1568 B
Ciphertext	768 B	1088 B	1568 B

Sécurité de Kyber-KEM

Base Théorique de la Sécurité

Hypothèse de sécurité fondamentale

Proposition : La sécurité IND-CCA2 de Kyber-KEM repose sur l'hypothèse que **D-MLWE est intraitable**.

Réduction formelle :

$$\text{D-MLWE difficile} \Rightarrow \text{Kyber-PKE IND-CPA} \Rightarrow \text{Kyber-KEM IND-CCA2}$$

(via transformation FO)

Justification de D-MLWE

Lien avec les réseaux : - D-MLWE se réduit à des problèmes de réseaux (approx-SVP) - Ces problèmes sont **NP-difficiles** - **Difficiles en moyenne** (average-case hard)

Résistance quantique : Aucun algorithme quantique efficace connu pour résoudre approx-SVP avec des facteurs d'approximation utilisés dans Kyber.

Choix des Distributions d'Erreurs

Distributions centrées binomiales

Pourquoi β_η ? - Échantillonnage **efficace** - Propriétés statistiques proches d'une gaussienne discrète - Analyse de sécurité bien comprise

Lien avec la difficulté des réseaux

Propriété importante : Les distributions d'erreurs sont choisies pour que la difficulté de MLWE soit **au moins celle** de résoudre certains problèmes de réseaux dans le pire cas.

Théorème (informel) : Résoudre MLWE en moyenne est au moins aussi difficile que résoudre approx-SVP dans le pire cas (même quantiquement).

Analyse de Sécurité Concrète

Modèles d'attaques considérés

1. Attaques par réseaux (Lattice attacks) : - **BKZ** (Block Korkine-Zolotarev) - **Sieving algorithms** (GaussSieve, etc.) - Estimation du coût pour différentes dimensions

2. Attaques algébriques : - Exploitation de la structure d'anneau - Aucune attaque efficace connue

3. Attaques combinatoires : - Recherche exhaustive - Meet-in-the-middle - Coût exponentiel

Estimation de sécurité

Méthode : Utiliser des outils comme le **Lattice Estimator** pour estimer le coût des meilleures attaques connues.

Résultat pour Kyber768 (exemple) : - Coût classique : ~2^{207} opérations (niveau 3 NIST) - Coût quantique : ~2^{174} opérations

Marge de sécurité : Les paramètres incluent une **marge de sécurité** substantielle.

Attaques d'Implémentation

Attaques par Canaux Auxiliaires (Side-Channel)

Types d'attaques

1. Analyse de consommation électrique (Power Analysis) : - **SPA** (Simple Power Analysis) - **DPA** (Differential Power Analysis) - Exploitation de la corrélation entre consommation et données secrètes

2. Analyse temporelle (Timing Attacks) : - Mesure du temps d'exécution - Détection de branches conditionnelles dépendant du secret - Exemple : if (secret_bit == 1) peut révéler le bit

3. Analyse électromagnétique (EM Analysis) : - Mesure des émissions électromagnétiques - Principe similaire à l'analyse de puissance

4. Attaques par cache : - Exploitation des accès mémoire cache - Flush+Reload, Prime+Probe

Vulnérabilités dans Kyber

Points sensibles : - Génération de clés (échantillonnage du secret) - Déchiffrement (dépend de la clé secrète) - Comparaison dans FO (test $c = c'$)

Exemples de fuites : - Temps variable selon les valeurs secrètes - Accès mémoire dépendant du secret

Contre-mesures

Implémentation en temps constant : - Éviter les branches conditionnelles dépendant du secret - Utiliser des opérations arithmétiques au lieu de if/else - Masking des données sensibles

Blinding : Ajouter du bruit aléatoire aux calculs intermédiaires

Shuffling : Randomiser l'ordre des opérations

Attaques par Injection de Fautes (Fault Attacks)

Principe

Méthode : - Perturber délibérément le calcul (laser, glitch électrique, etc.) - Observer le comportement erroné - Déduire des informations sur le secret

Vulnérabilités dans Kyber

Cibles potentielles : - Corruption du ciphertext c avant comparaison - Modification du résultat de Decrypt - Skip de vérifications critiques

Contre-mesures

Redondance des calculs : Effectuer les opérations critiques plusieurs fois

Détection de fautes : Vérifications d'intégrité, checksums

Code de correction d'erreurs : Encoder les données critiques

Prudence avec les Nouveaux Algorithmes

Maturité limitée

Contexte : Kyber et autres algorithmes post-quantiques sont **relativement nouveaux** (standardisés en 2022-2024).

Implications : - Moins de temps pour découvrir les vulnérabilités d'implémentation - Implémentations encore en évolution - Nouveaux vecteurs d'attaque peuvent émerger

Recommandations

Pour les développeurs : - Utiliser des **bibliothèques certifiées** et auditées - Suivre les **meilleures pratiques** publiées par le NIST - Implémenter des **contre-mesures** dès la conception - Effectuer des **audits de sécurité** réguliers

Pour les utilisateurs : - **Soyez prudent** dans les environnements critiques - Préférer les implémentations **validées** - Considérer une **approche hybride** pendant la transition

Aspects Pratiques et Transition

Crypto-Agilité

Définition et Concepts

Qu'est-ce que la crypto-agilité ?

Définition : La **crypto-agilité** (cryptographic agility) est la capacité d'un système à **basculer facilement** entre différentes primitives cryptographiques sans refonte majeure de l'architecture.

Concrètement : Un système crypto-agile permet de : - Changer d'algorithme de chiffrement - Mettre à jour les mécanismes de signature - Modifier les protocoles de key agreement

avec un minimum de modifications du code et de l'infrastructure.

Principes de conception

Abstraction : Utiliser des interfaces abstraites pour les opérations cryptographiques :

```
interface KEM {
    KeyGen() -> (pk, sk)
    Encapsulate(pk) -> (c, K)
    Decapsulate(sk, c) -> K
}
```

Configuration externe : Spécifier les algorithmes via fichiers de configuration, pas en dur dans le code.

Modularité : Composants cryptographiques interchangeables comme des “plugins”.

Objectifs d’un Système Crypto-Agile

Adaptation rapide aux menaces

Scénario 1 : Vulnérabilité découverte Si un algorithme est cassé ou affaibli : - Basculer rapidement vers une alternative - Minimiser le temps d’exposition - Limiter les perturbations opérationnelles

Scénario 2 : Nouvelle menace Apparition d’un ordinateur quantique cryptographiquement pertinent : - Transition vers post-quantique immédiate - Pas de refonte complète des systèmes

Flexibilité et évolutivité

Avantages à long terme : - Adaptation aux évolutions technologiques - Support de multiples niveaux de sécurité - Personnalisation selon les besoins

Coûts réduits : - Moins de développement pour chaque transition - Infrastructure réutilisable - Formation simplifiée

Importance pour la Transition Post-Quantique

Ampleur de la migration

Défi historique : La transition vers la cryptographie post-quantique sera probablement la **plus grande migration cryptographique de l’histoire**.

Échelle : - Milliards de dispositifs à mettre à jour - Infrastructure Internet mondiale - Systèmes embarqués (IoT, cartes à puce) - Certificats, signatures, protocoles

Incertitude temporelle

Problème : On ne sait pas **quand** un ordinateur quantique cryptographiquement pertinent sera construit.

Implications : - Besoin de préparer la transition **maintenant** - Mais timing exact inconnu (5 ans ? 20 ans ?) - Système agile permet d’être prêt pour toute échéance

Complexité de la transition

Défis techniques : - Compatibilité descendante nécessaire - Coexistence d'algorithmes classiques et post-quantiques - Tests et validation à grande échelle - Formation des développeurs et administrateurs

La crypto-agilité facilite : - Transition progressive - Coexistence de plusieurs algorithmes - Retour en arrière si nécessaire

Recommandations du NIST

Ne Pas Abandonner les Algorithmes Classiques

Approche recommandée

Position du NIST : Ne **pas abandonner complètement** les algorithmes cryptographiques classiques (RSA, ECC) pendant la transition.

Justifications

Raison 1 : Transition plus douce - Permet une **coexistence** progressive - Maintient la compatibilité avec systèmes existants - Réduit les risques de perturbations - Conserve les **optimisations actuelles** (matériel, bibliothèques)

Raison 2 : Nouvelles mathématiques - Cryptographie post-quantique basée sur **mathématiques moins matures** - Sécurité **empirique** plus que théorique dans certains cas - Moins de temps pour analyse cryptanalytique - Risque de vulnérabilités non découvertes

Raison 3 : Incertitude temporelle - Horizon des ordinateurs quantiques **incertain** - Peut-être 10 ans, peut-être 30 ans - Pas de raison d'abandonner sécurité actuelle prématurément - Les algorithmes classiques restent sûrs **aujourd'hui**

Approche Hybride

Concept d'hybridation

Principe : Combiner algorithmes classiques et post-quantiques dans un même système.

Schéma hybride typique : 1. Établir une clé avec algorithme classique (ex: ECDH) 2. Établir une clé avec algorithme post-quantique (ex: Kyber) 3. Combiner les deux clés (ex: $K = \text{KDF}(K_{\text{classique}} \| K_{\text{PQ}})$)

Avantages de l'approche hybride

Sécurité garantie : Le système reste sûr si **au moins un** des deux algorithmes est sûr : - Si ordinateur quantique arrive : post-quantique protège - Si vulnérabilité dans post-quantique : classique protège - **Double protection**

Transition progressive : - Déploiement graduel possible - Test en conditions réelles - Ajustements selon retours d'expérience

Flexibilité : - Adaptation selon applications - Choix du niveau de sécurité - Balance performance/sécurité

Exemples d'hybridation

TLS 1.3 hybride : - Key exchange : X25519 (classique) + Kyber768 (post-quantique) - Signature : ECDSA + Dilithium

SSH hybride : - Authentification : RSA + SPHINCS+ - Key exchange : ECDH + Kyber

Implémentation de Kyber

Considérations de Sécurité

Génération de nombres aléatoires

Importance critique : La sécurité de Kyber dépend fortement de la **qualité du générateur aléatoire**.

Exigences : - Utiliser un **CSPRNG** (Cryptographically Secure Pseudo-Random Number Generator) - Sources d'entropie appropriées (/dev/urandom, RDRAND, etc.) - Graine initiale suffisamment aléatoire

À éviter : Générateurs non cryptographiques (rand(), Math.random()) Graines prévisibles ou réutilisées Sources d'entropie insuffisantes

Contre-mesures side-channel

Implémentation en temps constant : Éviter tout branchement ou accès mémoire dépendant de données secrètes.

Masking : - Ajouter du bruit aléatoire aux calculs - Rend l'analyse de puissance plus difficile

Shuffling et blinding : - Randomiser l'ordre des opérations - Masquer les données intermédiaires

Validation des entrées

Vérifications nécessaires : - Taille des clés publiques reçues - Validité des ciphertexts - Cohérence des paramètres

Protection contre malformations : Rejeter les entrées invalides proprement (sans fuite d'information).

Optimisations de Performance

Number Theoretic Transform (NTT)

Principe : Transformation similaire à la FFT pour la multiplication de polynômes modulo $X^n + 1$.

Avantage : - Multiplication en $O(n \log n)$ au lieu de $O(n^2)$ - Critique pour les performances de Kyber

Implémentation : Utiliser des bibliothèques optimisées (AVX2, NEON sur ARM).

Vectorisation

SIMD (Single Instruction, Multiple Data) : - Traiter plusieurs coefficients simultanément - Extensions : AVX2, AVX-512 (x86), NEON (ARM)

Gain de performance : Facteur 2-4x typiquement.

Matériel dédié

Accélération matérielle : - Coprocesseurs cryptographiques - FPGA - ASICs dédiés

Cas d'usage : Dispositifs haute performance ou très contraints (IoT).

Conformité aux Standards

Spécifications du NIST

Documentation de référence : - FIPS 203 (standard Kyber final) - Spécifications techniques détaillées - Vecteurs de test

Respect des spécifications : Implémenter **exactement** selon le standard pour : - Interopérabilité - Validation - Certification

Certifications

Utiliser des implémentations certifiées : - Bibliothèques auditées (liboqs, PQClean) - Implémentations validées NIST - Code source ouvert et auditable

Processus de validation : - Tests avec vecteurs officiels - Audits de sécurité - Peer review

Mises à jour

Rester à jour : - Suivre les bulletins du NIST - Appliquer les correctifs de sécurité - Surveiller les nouvelles recommandations

Comparaisons avec Autres Approches Post-Quantiques

Familles de Cryptographie Post-Quantique

1. Basés sur les réseaux (Lattice-based)

Algorithmes : - **Kyber** (KEM) - **Dilithium** (signatures) - NTRU, FrodoKEM

Avantages : - **Performances** excellentes (clés, vitesse) - **Bien étudiés** théoriquement - **Flexibles** (différents niveaux de sécurité) - Preuves de sécurité solides

Inconvénients : - Mathématiques relativement récentes - Structure algébrique potentiellement exploitable

Kyber appartient à cette famille.

2. Basés sur les codes (Code-based)

Algorithmes : - **Classic McEliece** - BIKE, HQC

Avantages : - **Très mature** (30+ ans d'étude) - Sécurité bien comprise - Résistance structurelle forte

Inconvénients : - **Très grandes clés** (centaines de Ko) - Moins flexibles en taille

Comparaison avec Kyber : - McEliece : clé publique ~1 Mo vs Kyber768 : ~1 Ko - Kyber préféré pour plupart des applications

3. Basés sur les hash (Hash-based)

Algorithmes : - **SPHINCS+** - XMSS, LMS

Avantages : - Sécurité basée **uniquement sur fonctions de hachage** - Hypothèses minimales - Très conservateur

Inconvénients : - **Signatures volumineuses** (10-50 Ko) - Plus lent - Stateful pour certains (XMSS)

Usage : Principalement pour signatures, pas KEM.

4. Basés sur les isogénies (Isogeny-based)

Algorithmes : - SIKE (**cassé en 2022**) - Nouveaux candidats en développement

Historique : - Prometteur initialement (clés très petites) - SIKE cassé en quelques heures sur PC standard - Famille moins mature

État actuel : - Recherche active mais prudence - Pas recommandé pour déploiement production

Pourquoi Kyber a été Choisi par le NIST

Sélection du NIST

Processus : Compétition post-quantique du NIST (2016-2024) : - Round 1 : 69 candidats - Round 2 : 26 candidats - Round 3 : 7 finalistes - **Kyber sélectionné** en 2022, standardisé en 2024 (FIPS 203)

Critères de sélection

Équilibre performance/sécurité : Kyber offre un excellent compromis : - Tailles de clés **raisonnables** (800-1500 octets) - **Vitesse** élevée (ms pour encaps/decaps) - **Sécurité** prouvée et configurable

Flexibilité : - Trois niveaux de sécurité (512, 768, 1024) - Adaptable aux différents cas d'usage

Maturité relative : - Fondements mathématiques **bien étudiés** (réseaux) - Analyse cryptanalytique intensive - Pas de faiblesse majeure découverte

Implémentabilité : - Peut être implémenté sur matériel **varié** - Des serveurs aux dispositifs embarqués - Optimisations possibles (NTT, SIMD)

Écosystème et adoption

Support industriel : - Bibliothèques disponibles (liboqs, Bouncy Castle, etc.) - Intégration dans TLS, SSH, VPN - Support des navigateurs (Chrome, Firefox)

Standardisation complémentaire : - Dilithium pour signatures (même famille que Kyber) - SPHINCS+ comme backup conservateur

Exemple Pédagogique : Kyber Jouet

Avertissement Important

Nature de l'exemple

IMPORTANT : Les paramètres utilisés ci-dessous sont **minuscules** et fournis **uniquement à des fins pédagogiques**.

Ces paramètres ne sont PAS cryptographiquement sûrs.

Objectif pédagogique

But : Comprendre les **mécanismes** et **calculs** derrière Kyber-KEM sans se perdre dans des nombres gigantesques.

Approche : - Paramètres très réduits - Calculs faisables à la main - Logique identique au vrai Kyber

Configuration de l'Anneau Jouet

Paramètres réduits

Choix des paramètres : - $n = 4$ (au lieu de 256) - $q = 17$ (au lieu de 3329) - $k = 2$ (comme Kyber512) - Erreurs : coefficients dans $\{-1, 0, 1\}$

Anneau de polynômes :

$$R_q = \mathbb{Z}_{17}[X]/(X^4 + 1)$$

Représentation des polynômes

Éléments de R_q :

$$p(X) = a_0 + a_1X + a_2X^2 + a_3X^3, \quad a_i \in \{0, 1, \dots, 16\}$$

Notation vectorielle : On notera parfois $p = [a_0, a_1, a_2, a_3]$ pour simplifier.

Exemple :

$$p(X) = 1 + 3X + 2X^2 + 5X^3 = [1, 3, 2, 5]$$

Réduction modulo $X^4 + 1$

Règle :

$$X^4 \equiv -1 \equiv 16 \pmod{17}$$

Exemple de réduction :

$$X^5 = X \cdot X^4 \equiv X \cdot (-1) = -X \equiv 16X \pmod{X^4 + 1, 17}$$

Étape 1 : Génération de Clés (Récepteur)

Paramètres publics

Matrice publique $\mathbf{A} \in R_q^{2\times 2}$:

Supposons (générée aléatoirement ou de manière déterministe via SHAKE) :

$$\mathbf{A} = \begin{pmatrix} [2, 1, 4, 3] & [5, 2, 1, 6] \\ [3, 5, 2, 1] & [1, 4, 3, 2] \end{pmatrix}$$

(Chaque entrée est un polynôme dans R_q)

Vecteur secret

Tirage d'un petit secret $\mathbf{s} \in R_q^2$:

$$\mathbf{s} = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} [1, -1, 0, 1] \\ [0, 1, -1, 0] \end{pmatrix}$$

Interprétation : - $s_1(X) = 1 - X + X^3$ - $s_2(X) = X - X^2$ - Coefficients petits (dans $\{-1, 0, 1\}$)

Vecteur d'erreur

Tirage d'un petit vecteur d'erreur $\mathbf{e} \in R_q^2$:

$$\mathbf{e} = \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} [0,1,0,-1] \\ [-1,0,1,0] \end{pmatrix}$$

Interprétation : - $e_1(X) = X - X^3$ - $e_2(X) = -1 + X^2$

Calcul de la clé publique

Formule :

$$\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod q$$

Calcul détaillé :

$$\mathbf{t} = \begin{pmatrix} [2,1,4,3] & [5,2,1,6] \\ [3,5,2,1] & [1,4,3,2] \end{pmatrix} \begin{pmatrix} [1,-1,0,1] \\ [0,1,-1,0] \end{pmatrix} + \begin{pmatrix} [0,1,0,-1] \\ [-1,0,1,0] \end{pmatrix}$$

(Les calculs exacts nécessiteraient de faire les multiplications de polynômes dans R_q)

Résultat simplifié (après calculs) :

$$\mathbf{t} = \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} [7,11,2,9] \\ [4,8,15,3] \end{pmatrix}$$

Clés résultantes

Clé publique :

$$pk = (\mathbf{A}, \mathbf{t})$$

Clé secrète :

$$sk = \mathbf{s}$$

Étape 2 : Encapsulation (Émetteur)

Partie 1 : Choix du message et randomness

Message aléatoire : L'émetteur choisit un message $m \in \{0,1\}^4$ qui deviendra la clé partagée :

$$m = [1, 0, 1, 1]$$

Encodage du message :

$$\text{Encode}(m) = \left\lfloor \frac{q}{2} \right\rfloor \cdot m = 8 \cdot [1, 0, 1, 1] = [8, 0, 8, 8]$$

Vecteur aléatoire court $\mathbf{r} \in R_q^2$:

$$\mathbf{r} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} = \begin{pmatrix} [1, 0, -1, 0] \\ [-1, 1, 0, 0] \end{pmatrix}$$

Erreurs $\mathbf{e}_1 \in R_q^2$ et $e_2 \in R_q$:

$$\mathbf{e}_1 = \begin{pmatrix} [0, -1, 0, 1] \\ [1, 0, 0, -1] \end{pmatrix}, \quad e_2 = [0, 1, -1, 0]$$

Partie 2 : Calcul du chiffré

Première composante :

$$\mathbf{u} = \mathbf{A}^T \mathbf{r} + \mathbf{e}_1 \pmod{q}$$

Transposée de A :

$$\mathbf{A}^T = \begin{pmatrix} [2, 1, 4, 3] & [3, 5, 2, 1] \\ [5, 2, 1, 6] & [1, 4, 3, 2] \end{pmatrix}$$

Calcul (simplifié) :

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} [5, 14, 3, 11] \\ [9, 2, 16, 7] \end{pmatrix}$$

Seconde composante :

$$v = \mathbf{t}^T \mathbf{r} + e_2 + \text{Encode}(m) \pmod{q}$$

Calcul (simplifié) :

$$v = [12, 5, 11, 14]$$

Sortie de l'encapsulation

Ciphertext :

$$c = (\mathbf{u}, v)$$

Clé partagée (dérivée de m) :

$$K = H(m) = H([1, 0, 1, 1])$$

(Dans le vrai Kyber, on utilise SHA3-256)

Étape 3 : Décapsulation (Récepteur)

Calcul du message

Formule :

$$w = v - \mathbf{s}^T\mathbf{u} \pmod q$$

Développement :

$$w = v - (s_1 \cdot u_1 + s_2 \cdot u_2)$$

Calcul détaillé :

$$s_1 \cdot u_1 = [1, -1, 0, 1] \cdot [5, 14, 3, 11]$$

(multiplication de polynômes dans R_q)

$$s_2 \cdot u_2 = [0, 1, -1, 0] \cdot [9, 2, 16, 7]$$

Résultat (après calculs) :

$$\mathbf{s}^T\mathbf{u} = [4, 5, 3, 6]$$

Donc :

$$w = [12, 5, 11, 14] - [4, 5, 3, 6] = [8, 0, 8, 8] \pmod{17}$$

Décodage

Rappel de l'encodage :

$$\text{Encode}(m) = [8, 0, 8, 8]$$

Décodage : Pour chaque coefficient de $w = [w_0, w_1, w_2, w_3]$:

$$m_i = \begin{cases} 0 & \text{si } |w_i| < q/4 \approx 4 \\ 1 & \text{si } |w_i - 8| < q/4 \end{cases}$$

Application : - $w_0 = 8 : \ |8 - 8| = 0 < 4 \to m_0 = 1$ - $w_1 = 0 : \ |0| = 0 < 4 \to m_1 = 0$ - $w_2 = 8 : \ |8 - 8| = 0 < 4 \to m_2 = 1$ - $w_3 = 8 : \ |8 - 8| = 0 < 4 \to m_3 = 1$

Message retrouvé :

$$m' = [1, 0, 1, 1] = m$$

Clé partagée

Dérivation :

$$K' = H(m') = H([1, 0, 1, 1]) = K$$

Succès : Les deux parties partagent la même clé K.

Analyse du Mécanisme

Pourquoi ça fonctionne

Développement complet de w :

$$\begin{aligned} w &= v - \mathbf{s}^T\mathbf{u} \\ &= (\mathbf{t}^T\mathbf{r} + e_2 + \text{Encode}(m)) - \mathbf{s}^T(\mathbf{A}^T\mathbf{r} + \mathbf{e}_1) \\ &= ((\mathbf{A}\mathbf{s} + \mathbf{e})^T\mathbf{r} + e_2 + \text{Encode}(m)) - \mathbf{s}^T\mathbf{A}^T\mathbf{r} - \mathbf{s}^T\mathbf{e}_1 \\ &= \mathbf{s}^T\mathbf{A}^T\mathbf{r} + \mathbf{e}^T\mathbf{r} + e_2 + \text{Encode}(m) - \mathbf{s}^T\mathbf{A}^T\mathbf{r} - \mathbf{s}^T\mathbf{e}_1 \\ &= \text{Encode}(m) + \underbrace{(\mathbf{e}^T\mathbf{r} + e_2 - \mathbf{s}^T\mathbf{e}_1)}_{\text{erreur totale}} \end{aligned}$$

Condition de succès : Si l'**erreur totale** reste **petite** ($< q/4$ en norme infinie), le décodage récupère m correctement.

Rôle des petites erreurs

Sécurité : Les erreurs $\mathbf{e}, \mathbf{e}_1, e_2$: - Masquent le secret $\mathbf{s}$ dans $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$ - Rendent MLWE difficile - Empêchent la résolution par algèbre linéaire

Correction : Les erreurs restent assez **petites** pour que : - Elles s'annulent partiellement dans w - Le décodage réussisse malgré le bruit résiduel

Relation avec le Vrai Kyber-KEM

Différences d'échelle

Paramètres réels (Kyber768) : - $n = 256$ (vs 4) - $q = 3329$ (vs 17) - $k = 3$ (vs 2) - Polynômes de 256 coefficients - Vecteurs de 3 polynômes

Tailles : - Clé publique : ~1 Ko (vs quelques octets) - Ciphertext : ~1 Ko (vs quelques octets)

Structure identique

Malgré les différences de taille, la structure est la même :

1. **KeyGen :**
    - Matrice publique $\mathbf{A}$
    - Secret $\mathbf{s}$ petit
    - Clé publique $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$
2. **Encapsulation :**
    - Message aléatoire m
    - Vecteur aléatoire court $\mathbf{r}$
    - $\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$
    - $v = \mathbf{t}^T\mathbf{r} + e_2 + \text{Encode}(m)$
3. **Decapsulation :**
    - Calcul $w = v - \mathbf{s}^T\mathbf{u}$
    - Décodage pour récupérer m
    - Dérivation de $K = H(m)$

Transformations supplémentaires

Dans le vrai Kyber-KEM :

Compression/décompression : Les vecteurs $\mathbf{u}$ et scalaire v sont **compressés** pour réduire la taille du ciphertext.

Transformation FO : - Rechiffrement et vérification $c = c'$ - Rejet implicite si manipulation détectée - Garantit sécurité IND-CCA2

Fonctions de hachage : - G, H, J pour dérivation et vérification - Randomness déterministe (Encrypt déterministe)

Aide-Mémoire et Synthèse

Concepts Clés à Retenir

Informatique quantique

Qubits et propriétés : - **Superposition** : état dans combinaison linéaire de $|0\rangle$ et $|1\rangle$ - **Intrication** : corrélation quantique entre qubits - **Parallélisme** : calcul sur 2^n états simultanément

Défis : - Décohérence lors de la mesure - Sensibilité environnementale - Difficulté d'implémentation

Menaces cryptographiques

Algorithme de Shor : - Résout factorisation et log discret en temps **polynomial** - Casse RSA, ECC, Diffie-Hellman - Nécessite ordinateur quantique de grande échelle

Algorithme de Grover : - Accélération **quadratique** pour recherche - Impact limité sur cryptographie symétrique - Solution : doubler taille des clés

Problèmes post-quantiques

LWE et variantes : - **LWE** : Learning With Errors - **MLWE** : Module-LWE (sur anneaux de polynômes) - **D-MLWE** : version décision

Réseaux euclidiens : - **SVP** : Shortest Vector Problem - **CVP** : Closest Vector Problem - NP-difficiles, difficiles en moyenne - Résistants aux attaques quantiques

Kyber

Nature : - KEM post-quantique - Basé sur MLWE/D-MLWE - Sécurité IND-CCA2

Structure : - Kyber-PKE (cœur cryptographique) - Transformation FO (sécurité CCA2) - Fonctions de hachage (SHA3)

Niveaux de sécurité : - Kyber512 (NIST 1 AES-128) - Kyber768 (NIST 3 AES-192) - Kyber1024 (NIST 5 AES-256)

Formules Essentielles

Structures algébriques

Anneau de polynômes :

$$R_q = \mathbb{Z}_q[X]/(X^n + 1)$$

Module :

$$R_q^k = \text{vecteurs de } k \text{ polynômes dans } R_q$$

Norme infinie :

$$\|p\|_\infty = \max_i |a_i|$$

Problème MLWE

Instance MLWE :

$$(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e})$$

Objectif (recherche) : Trouver $\mathbf{s}$ à partir de $(\mathbf{A}, \mathbf{t})$

Objectif (décision) : Distinguer $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$ de $(\mathbf{A}, \mathbf{t}_{\text{aléatoire}})$

Kyber-PKE

KeyGen :

$$\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$$

$$pk = (\mathbf{A}, \mathbf{t}), \quad sk = \mathbf{s}$$

Encrypt :

$$\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$$

$$v = \mathbf{t}^T\mathbf{r} + e_2 + \text{Encode}(m)$$

$$c = (\mathbf{u}, v)$$

Decrypt :

$$w = v - \mathbf{s}^T\mathbf{u} = \text{Encode}(m) + (\text{petites erreurs})$$
$$m = \text{Decode}(w)$$

Questions Types d'Examen

Questions conceptuelles

Sur l'informatique quantique : 1. Différence entre bit classique et qubit 2. Qu'est-ce que la superposition quantique ? 3. Qu'est-ce que l'intrication ? 4. Quels sont les principaux défis de l'informatique quantique ? 5. Pourquoi ne peut-on pas simuler efficacement un ordinateur quantique sur un ordinateur classique ?

Sur les menaces : 1. Pourquoi les ordinateurs quantiques menacent-ils RSA et ECC ? 2. Qu'est-ce que l'algorithme de Shor et quel est son impact ? 3. Différence entre impact de Shor et Grover 4. Qu'est-ce que la menace "Harvest Now, Decrypt Later" ? 5. Quels systèmes sont affectés par un ordinateur quantique ?

Sur la cryptographie post-quantique : 1. Qu'est-ce que la cryptographie post-quantique ? 2. Pourquoi les problèmes de réseaux sont-ils intéressants ? 3. Différence entre problème worst-case et average-case 4. Qu'est-ce qu'un KEM ? 5. Qu'est-ce que CRYSTALS-Kyber ?

Questions techniques

Sur LWE et variantes : 1. Différence entre Learning Without Errors et Learning With Errors 2. Pourquoi ajouter des erreurs rend le problème difficile ? 3. Qu'est-ce que MLWE et en quoi diffère-t-il de LWE ? 4. Différence entre MLWE (search) et D-MLWE (decision) 5. Quel est le rôle de D-MLWE dans la sécurité de Kyber ?

Sur les réseaux : 1. Définir un réseau euclidien 2. Qu'est-ce que le Shortest Vector Problem (SVP) ? 3. Qu'est-ce que le Closest Vector Problem (CVP) ? 4. Pourquoi SVP et CVP sont-ils NP-difficiles ? 5. Qu'est-ce que la difficulté average-case et pourquoi est-elle importante ? 6. Quel est le lien entre MLWE et les problèmes de réseaux ?

Sur Kyber : 1. Quelle est la structure de Kyber-KEM ? 2. Différence entre Kyber-PKE et Kyber-KEM 3. Rôle de la transformation de Fujisaki-Okamoto 4. Quelles fonctions de hachage sont utilisées et pourquoi ? 5. Qu'est-ce que l'anneau R_q utilisé dans Kyber ? 6. Pourquoi utilise-t-on des polynômes "petits" ?

Questions de calcul

Structures algébriques : 1. Réduire un polynôme modulo $X^n + 1$ 2. Multiplier deux polynômes dans R_q 3. Calculer la norme infinie d'un polynôme 4. Encoder et décoder un message binaire

Exemple jouet : 1. Effectuer KeyGen avec paramètres jouets 2. Calculer l'encapsulation 3. Effectuer la decapsulation 4. Expliquer pourquoi $w = \text{Encode}(m) + (\text{erreurs})$

Questions pratiques

Sécurité : 1. Sur quoi repose la sécurité de Kyber ? 2. Qu'est-ce que la sécurité IND-CCA2 ? 3. Quelles sont les attaques d'implémentation possibles contre Kyber ? 4. Comment se protéger des attaques par canaux auxiliaires ? 5. Pourquoi le déchiffrement peut-il échouer ?

Transition post-quantique : 1. Qu'est-ce que la crypto-agilité et pourquoi est-elle importante ? 2. Quelles sont les recommandations du NIST ? 3. Qu'est-ce qu'une approche hybride classique/post-quantique ? 4. Avantages et inconvénients de l'approche hybride 5. Pourquoi ne pas abandonner RSA/ECC immédiatement ?

Comparaisons : 1. Comparer Kyber avec Classic McEliece 2. Avantages et inconvénients des différentes familles post-quantiques 3. Pourquoi Kyber a-t-il été choisi par le NIST ?

Tableaux Récapitulatifs

Ordinateur classique vs quantique

Aspect	Classique	Quantique
Unité de base	Bit (0 ou 1)	Qubit ($\alpha\|0\rangle + \beta\|1\rangle$)
États possibles	Finis, discrets	Infinis (superposition)
n unités	2^n configurations (une active)	2^n états (tous actifs)
Parallélisme	Séquentiel ou multi-cœur	Intrinsèque (superposition)
Mesure	Lecture sans destruction	Effondrement (perte superposition)
Sensibilité	Robuste	Très sensible
Infrastructure	Standard	Cryogénique, isolation

Comparaison des familles post-quantiques

Famille	Exemples	Avantages	Inconvénients
Réseaux	Kyber, Dilithium	Performances, flexibilité	Math récentes
Codes	McEliece	Très mature	Clés énormes
Hash	SPHINCS+	Hypothèses minimales	Signatures volumineuses
Isogénies	SIKE (cassé)	Clés petites	Moins mature, vulnérable

Paramètres de Kyber

Variante	k	Niveau NIST	Équivalent	Clé publique	Ciphertext
Kyber512	2	1	AES-128	800 B	768 B
Kyber768	3	3	AES-192	1184 B	1088 B
Kyber1024	4	5	AES-256	1568 B	1568 B

Recommandations Pratiques

Pour la transition post-quantique

Étapes clés :

1. **Inventaire :**
    - Identifier tous les systèmes utilisant cryptographie asymétrique
    - Cartographier les dépendances
    - Évaluer l'impact
2. **Priorisation :**
    - Systèmes critiques en premier
    - Données à long terme (vulnérables à "Harvest Now, Decrypt Later")
    - Infrastructure exposée
3. **Crypto-agilité :**
    - Concevoir pour changements futurs
    - Abstractions cryptographiques
    - Configuration externe
4. **Approche hybride :**
    - Combiner classique + post-quantique
    - Transition progressive

   - Tests en conditions réelles
5. **Tests et validation :**
   - Vecteurs de test officiels
   - Audits de sécurité
   - Performance et compatibilité

Pour l'implémentation de Kyber

Sécurité :

- **Générateur aléatoire** : CSPRNG de qualité
- **Contre-mesures side-channel** : implémentation temps constant
- **Validation** : vérifier toutes les entrées
- **Bibliothèques certifiées** : liboqs, PQClean
- **Conformité NIST** : suivre FIPS 203

Performance :

- **NTT** : utiliser pour multiplication de polynômes
- **Vectorisation** : SIMD (AVX2, NEON)
- **Optimisations** : profiling et tuning
- **Matériel** : considérer accélération si besoin

Maintenance :

- **Mises à jour** : suivre bulletins NIST
- **Patches** : appliquer rapidement
- **Veille** : nouvelles attaques, recommandations

Glossaire

- **AES** : Advanced Encryption Standard
- **Average-case hard** : Difficile en moyenne
- **BKZ** : Block Korkine-Zolotarev (algorithme de réduction de base)
- **CCA** : Chosen Ciphertext Attack
- **CPA** : Chosen Plaintext Attack
- **CSPRNG** : Cryptographically Secure Pseudo-Random Number Generator
- **CVP** : Closest Vector Problem
- **D-MLWE** : Decision Module Learning With Errors
- **DPA** : Differential Power Analysis
- **ECC** : Elliptic Curve Cryptography
- **FO** : Fujisaki-Okamoto (transformation)

- **IND-CCA2** : Indistinguishability under Adaptive Chosen Ciphertext Attack
- **IoT** : Internet of Things
- **KEM** : Key Encapsulation Mechanism
- **Lattice** : Réseau euclidien
- **LWE** : Learning With Errors
- **MLWE** : Module Learning With Errors
- **NIST** : National Institute of Standards and Technology
- **NP-hard** : Non-deterministic Polynomial-time hard
- **NTT** : Number Theoretic Transform
- **PKE** : Public Key Encryption
- **Qubit** : Quantum bit
- **RSA** : Rivest–Shamir–Adleman
- **SHA3** : Secure Hash Algorithm 3
- **SHAKE** : Secure Hash Algorithm KECCAK (XOF)
- **SIMD** : Single Instruction, Multiple Data
- **SPA** : Simple Power Analysis
- **SVP** : Shortest Vector Problem
- **TLS** : Transport Layer Security
- **XOF** : eXtendable Output Function