

耐量子計算機暗号

光成滋生, 2025/12/05

概要

量子計算機による暗号解読

- 素因数分解, 離散対数問題など重要な公開鍵暗号の安全性に関わる問題が破られる危険性
 - 具体的なアルゴリズムについては後の講義で

耐量子計算機暗号

- 耐量子暗号, PQC (Post-Quantum Cryptography) とも
- 量子計算機が登場しても安全な古典計算機で動作する暗号技術

PQCへの移行

- 解読できなくても現在利用しているTLSを盗聴してデータを保存
- 将来量子計算機が登場したときに解読される危険性
- PQCへの移行は実用的な量子計算機の登場よりも前に進めるべき

NISTのPQCプロセス

標準化スケジュール

- 2016 : NISTがPQCの標準化プロセスを開始
- 2017 : 69方式受理
- 2019 : 1st round : 26方式通過
- 2020 : 2nd round : 15方式通過
- 2022 : 3rd round : 4方式通過 (PKEとKEMが標準方式に確定)
 - 保留だった4方式が4rd round評価へ
 - 2022/7 そのうちのSIKEが攻撃された
 - 署名は追加公募, 確定したものはドラフト作成中
- 2025 : 標準化文書公開

PQC標準化の状況

確定したもの

- PKC/KEM : CRYSTALS-KYBER (格子) , HQC (符号: 2025/3確定 2027年標準化予定)
- 署名 : CRYSTALS-Dilithium (格子) , FALCON (格子) , SPHINCS+ (ハッシュ)

その他の暗号の種類

符号暗号

- BIKE, Classic, McEliece

同種写像

- SIKE, SIDH: 破れた(2023), CSIDH, SQISign: 評価中, 現在有望と考えられている

多変数多項式

- 鍵サイズが大きい (公開鍵 \geq 3MB, 秘密鍵 \geq 400KB)

主なアルゴリズムの鍵サイズ

一覧

アルゴリズム	種類	公開鍵	秘密鍵	暗号文・署名
ECDH	KEM	32B	32B	32B
ECDSA	署名	32B	64B	64B
MLKEM768 (KYBER-768)	KEM	1184B	2400B	1088B
Dilithium	署名	1952B	4000B	3293B
SPHINCS+	署名	32B	64B	7856B

MLKEM768のブラウザ対応

- 2024/初め: Chrome 124/Edge/FirefoxがKyber-768対応
- 2025/9: iPhone (iOS26) のブラウザも対応
- [Cloudflare Research](#) で確認できる

格子暗号

LWE(Learning With Errors) 問題

- $M_{m,n}(\mathbb{F}_q)$ を縦 m , 横 n の $m \times n$ 次行列全体の集合とする
- $A \in M_{m,n}(\mathbb{F}_q)$ と縦(列)ベクトル $s \in \mathbb{F}_q^n$ を選ぶ
 $b := As + e \pmod{q}$ とし, (A, b) : given のとき s を求めよ
 - ここで e は小さい整数(後述)からなる m 次元縦ベクトル
 - $e = 0$ なら ($m \geq n$ として) 普通の連立方程式を解くだけ
 - 注意: 表記の都合上, 行列やベクトルの縦横が他の書籍と転置しているかもしれない

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} + \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{bmatrix}$$

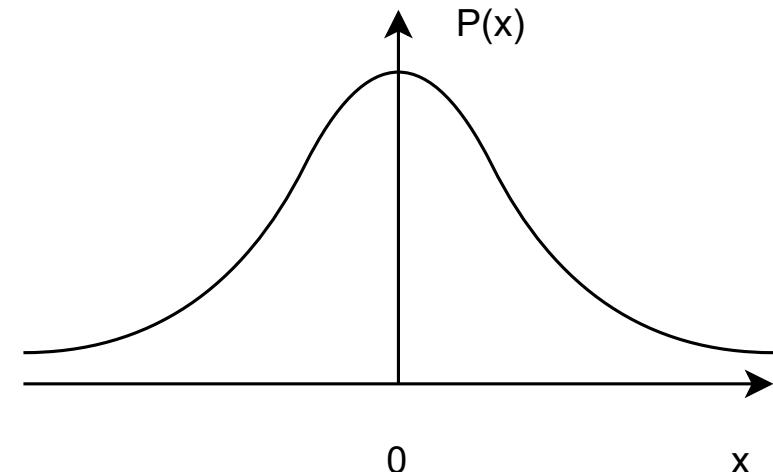
仮定

- 適切なパラメータに対して LWE 問題は困難であることを LWE 仮定という
- (判定版) LWE 仮定
 - 上記の方法で生成した $\{(A, b)\}$ を (計算) LWE 分布という
 - LWE 分布と $(A, b) \xleftarrow{U} M_{m,n}(\mathbb{F}_q) \times \mathbb{F}_q^m$ を区別するのが困難

ノイズの分布

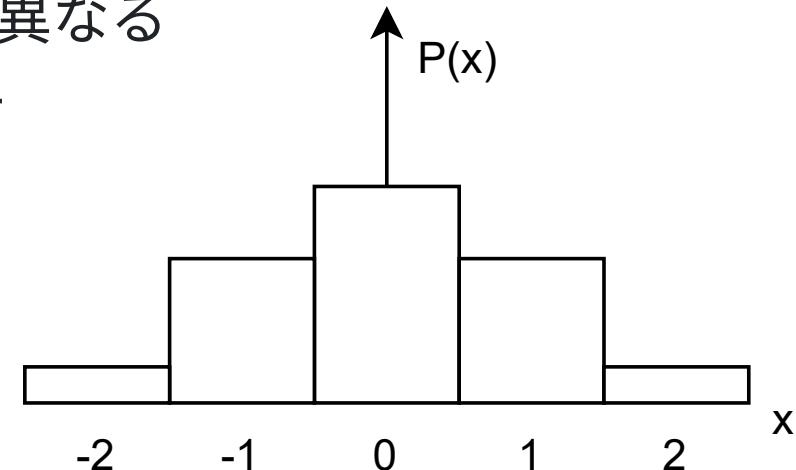
連続ガウス分布

- 平均0, 標準偏差 σ のガウス分布
 - $P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-x^2/2\sigma^2)$



離散ガウス分布

- 整数 x が $\exp(-x^2/2\sigma^2)$ に比例する確率で分布する
 - e は離散ガウス分布からサンプリングする
- 厳密には通常の連続ガウス分布の結果を整数に丸めたものとは異なる
- 扱いが困難なので実装時には適当な境界パラメータ L を決めて $[-L, L]$ の範囲で生成する (e.g., 二項分布)



Regev暗号

鍵生成

- $s \in \mathbb{F}_q^n, A \in M_{m,n}(\mathbb{F}_q), e \in \mathbb{F}_q^m$ を選び $b := As + e$ とする
 - s が秘密鍵で (A, b) が公開鍵

$m \in \{0, 1\}$ の暗号化

- $r \xleftarrow{U} \{0, 1\}^m$ を選び $c := (A^T r, b^T r + (q//2)m)$ が暗号文 ($q//2 := \text{floor}(q/2)$)

$c = (u, v)$ の復号

- $m := [v - s^T u]_q$ が 0 に近ければ $m = 0$, そうでなければ $m = 1$
 - ここで $[a]_q$ は $a \in [0, q - 1]$ を $a \leq q//2$ なら a , それ以外は $a - q//2$ とする

正当性

- $m = (As + e)^T r + (q//2)m - s^T A^T r = e^T r + (q//2)m \approx (q//2)m$
 - $e^T r \ll (q//2)$ なら正しく復号できる

構造化格子

多項式環上の格子

- データサイズ削減のためにより効率のよい格子を利用する
- q : 素数, n : 2のべき, 1の原始 ν 乗根 $\zeta \in \mathbb{F}_q$ (位数が ν : $\zeta^\nu = 1$)
 - MLKEM768では $q = 3329, n = 256, \nu = 256, \zeta = 17$
- $\phi(x) := x^n + 1, R := \mathbb{Z}[x]/(x^n + 1), R_q := \mathbb{F}_q[x]/(x^n + 1)$

MLWE (Module LWE) 問題

- $s \leftarrow \underset{U}{R_q}^k$ とノイズの分布 χ を選ぶ (ベクトルの内積 $s \cdot a = s^T a = a^T s$ に注意)

$$\text{MLWE分布: } A_{s,\chi} := \left\{ (a, s^T a + e) \in R_q^k \times R_q \mid a \leftarrow \underset{U}{R_q}^k, e \leftarrow \chi \right\}$$

MLWE仮定

- MLWE分布と $(a, b) \leftarrow \underset{U}{R_q}^k \times R_q$ を区別するのが困難

Kyber PKEの概要

鍵生成

- $s \in R_q^k, A \in M_k(R_q), e \in R_q^k$ を選び $b := As + e$ とする
 - $sk := s$ が秘密鍵で $pk := (A, b)$ が公開鍵

暗号化

- Encode: 256bitの $m = \sum_i m_i 2^i$ から $\tilde{m} := \sum_{i=0}^{255} m_i (q//2)x^i \in R_q$ を作る
- 乱数 $r \in R_q^k, e_1 \in R_q^k, e_2 \in R_q$ を選び
 $Enc(pk, m) := (u, v) := (A^T r + e_1, r^T b + e_2 + \tilde{m})$ が暗号文

復号と正当性

- $Dec(sk, (u, v)) := v - s^T u = r^T(As + e) + e_2 + \tilde{m} - s^T(A^T r + e_1)$
 $= (r^T As - s^T A^T r) + (r^T e + e_2 - s^T e_1) + \tilde{m}$
- Decode: $r^T e + e_2 - s^T e_1$ は小さいので $q//2$ に近いか否かで \tilde{m} から m_i を0 or 1で復元
- 安全性: Kyber PKEはMLWE仮定の元でIND-CPA安全

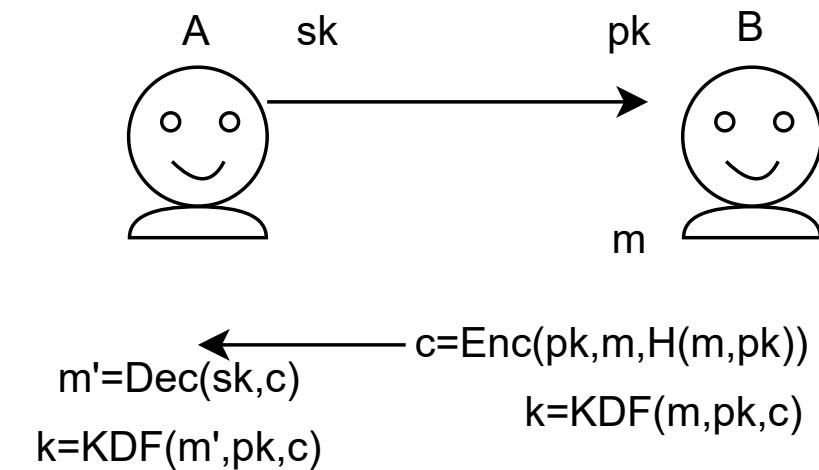
MLKEMの概要

Kyber PKEからIND-CCA2安全な鍵共有(Kyber-KEM)を実現

- 藤崎-岡本変換（の変種）をKEM単体に適用
 - 公開鍵を毎回使い捨てることでDH鍵共有と同じ通信回数

鍵共有

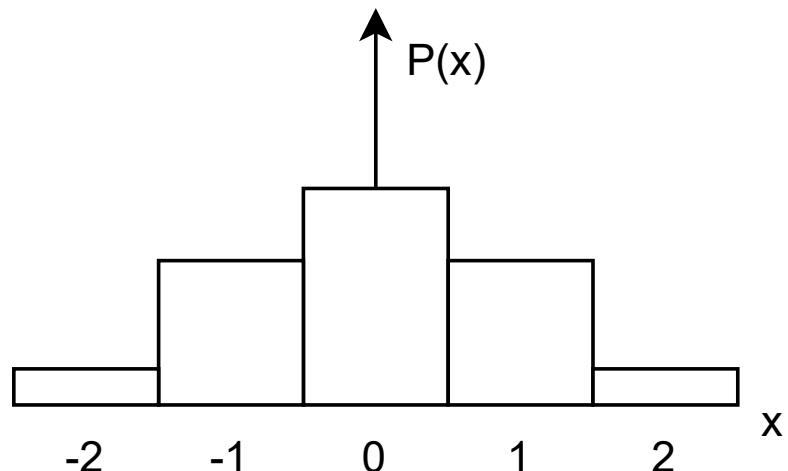
- A: (sk, pk) をKyber PKEで生成し pk をBに送る
- B: ランダムな m を選び
 $c := Enc(pk, m, H(m, pk))$ をAに送る
 - $k := KDF(m, pk, c)$: 共有鍵
- A: $m' := Dec(sk, c), c' := Enc(pk, m', H(m', pk))$
 - $c = c'$ なら $k := KDF(m', pk, c)$: 共有鍵
そうでなければ $KDF(\text{乱数}, pk, c)$ を利用
- 注意: $c \neq c'$ のとき乱数を返すのは鍵の情報 (m) を漏らさないようにするため



MLKEM768 (Kyber-KEM) のパラメータと性能改善

主要パラメータ

- $n = 256, q = 3329 = 13n + 1, k = 3, \eta = 2$
- e は $n = 4, p = 1/2$ の二項分布（範囲は $[-2, 2]$ ）を利用



公開鍵サイズの削減

- A は k^2 個の要素
- SHAKE, SHA-3などからなるXOF, PRFを利用して256bitのseedから各種乱数を生成

多項式乗算の高速化

- n 次元の多項式同士の乗算は素朴にやると $O(n^2)$ 回の演算が必要
- FFTの類似物である NTT (Number Theoretic Transform) を利用して高速化
- 多項式 $f(x) = \sum_i f_i x^i, g(x) = \sum_i g_i x^i$ の積 $h(x) = f(x)g(x) = \sum_i h_i x^i$
 $h_k = \sum_i f_i g_{k-i}$ を畳み込みという

FFT (Fast Fourier Transform) の概要

DFT (Discrete Fourier Transform)

- Fourier変換の離散版
- n を2のべき乗, $w := e^{2\pi\sqrt{-1}/n}$, 複素 n 次行列 $U := (u_{ij}) = (\sqrt{1/n} \cdot w^{ij})_{i,j}$ とする
- $(U\bar{U}^T)_{ik} = \sum_j u_{ij}\bar{u_{kj}} = (1/n) \sum_j w^{j(i-k)} = \delta_{ik}$ より U はユニタリ行列
 - δ_{ik} はクロネッカーのデルタ ($i = k$ なら1, それ以外は0)
 - $i = k$ のとき $\sum_j w^{j(i-k)} = \sum_j 1 = n$
 - $i \neq k$ のとき $\sum_j (w^{i-k})^j = (1 - w^{(i-k)n})/(1 - w^{i-k}) = 0$ (等比数列の和)
- 複素 n 次元ベクトル $a := (a_0, \dots, a_{n-1})^T$ に対して
 $A := F_n(a) := Ua = (1/\sqrt{n})(\sum_i a_i w^{ij})_j$ をDFT,
 $F_n^{-1}(a) := \bar{U}^T A = (1/\sqrt{n})(\sum_i A_i w^{-ij})_j$ を逆DFTという

演算量

- n 次元ベクトルと行列の積なので定義に従うと n^2 回の乗算

計算の分解

添え字の偶数・奇数に分ける

- $m = n/2, n$ 次元の w を w_n とすると $(w_n)^2 = w_m$
- $(\sqrt{n})A_j = \sum_{i=0}^{n-1} a_i w_n^{ij} = \sum_{i=0}^{m-1} a_{2i} w_n^{2ij} + \sum_{i=0}^{m-1} a_{2i+1} w_n^{(2i+1)j}$
 $= \sum_{i=0}^{m-1} a_{2i} w_m^{ij} + w_n^j \sum_{i=0}^{m-1} a_{2i+1} w_m^{ij}$
- $j = 0, 1, \dots, m-1$ のとき $\sqrt{2}A_j = F_m(a_{\text{even}})_j + w_n^j F_m(a_{\text{odd}})_j$
- $j = m, m+1, \dots, n-1$ のとき $w_m^j = w_m^{j-m}, w_n^j = -w_n^{j-m}$ より
 $\sqrt{2}A_j = F_m(a_{\text{even}})_{j-m} - w_n^{j-m} F_m(a_{\text{odd}})_{j-m}$
- $F_m(a_{\text{even}}), F_m(a_{\text{odd}})$ が求まれば $F_n(a)$ が求まる
 - 演算量は $F_m(a)$ 2回と w_n^{j-m} の積 n 回で $2n^2/4 + n = n^2/2 + n$ 回

再帰的に分解

- もう一度分割すると $2((n/2)^2/2 + (n/2)) + n = n^2/4 + 2n$ 回
- これを繰り返すと約 $n \log_2 n$ 回で計算できる: FFTという

FFTを用いた多項式の乗算の高速化

多項式の係数の畳み込み

- n 次多項式 $f(x) := \sum_{i=0}^{n-1} a_i x^i, g(x) = \sum_{i=0}^{n-1} b_i x^i$ の積 $h(x) = f(x)g(x) = \sum_{i=0}^{n-1} c_i x^i$
- $c_k = \sum_{i=0}^k a_i b_{k-i}$ を畳み込みという: そのまま計算すれば $O(n^2)$ 回の乗算
 - 多項式を $x^n + 1$ で割った余りを考えると添え字は $\text{mod } n$ で計算できる
- $C_j := F(a)_j F(b)_j$ とすると
$$\begin{aligned} c_k &:= F^{-1}(C)_k = (1/\sqrt{n}) \sum_j F(a)_j F(b)_j w^{-jk} = (1/\sqrt{n^3}) \sum_{i,j,s} a_i w^{ij} b_s w^{sj} w^{-jk} \\ &= (1/\sqrt{n^3}) \sum_{i,s} a_i b_s (\sum_j w^{j(i+s-k)}) \# k = i + s \text{ となる組合せで先に加算する} \\ &= (1/\sqrt{n^3}) \sum_{i,s} a_i b_s n \delta_{k,i+s} = (1/\sqrt{n}) \sum_i a_i b_{k-i} \text{ (添え字は mod } n \text{ で計算)} \end{aligned}$$
- つまりFFT変換して要素ごとに掛けて逆変換すれば畳み込みになる
$$\sum_i a_i b_{k-i} = \sqrt{n} F^{-1}(F(a)_j F(b)_j)_k$$

演算量

- FFTが3回と n 回の乗算なので $O(n \log_2 n)$

NTTへの応用

整数 \mathbb{F}_q の世界では $\exp(2\pi\sqrt{-1}/n)$ は使えない

- 代わりに \mathbb{F}_q における1の原始 n 乗根 ω を利用する
- 再帰的な分解による演算量の削減も同様に可能

巡回畳込みと負巡回畳み込み

- 多項式の計算が $\mathbb{F}_q[x]/(x^n - 1)$ の中だったら $c_k = \sum_{i=0}^k a_i b_{k-i} + \sum_{i=k+1}^{n-1} a_i b_{k-i+n} = \sum_i a_i b_{(k-i) \bmod n}$
 - 添え字を $\bmod n$ で計算する巡回畳み込みになる
- $\mathbb{F}_q[x]/(x^n + 1)$ の中では $c_k = \sum_{i=0}^k a_i b_{k-i} - \sum_{i=k+1}^{n-1} a_i b_{k-i+n}$: 負巡回畳み込みという

MLKEM768でのNTT

- $\omega = 17$ とすると ω は \mathbb{F}_q ($q = 3329$) における1の原始 $n = 256$ 乗根
 - $\omega^i \neq 1$ for $i \in [1, n)$, $\omega^n = 1$, $\omega^{n/2} = -1$