Implementation and deployment of post-quantum cryptography

Gabriel Chênevert

Computer Science & Mathematics department
JUNIA Engineering School
Catholic University of Lille, France

Overview

Implementation and deployment of post-quantum cryptography

- ML-based post-quantum primitives
- The arithmetic of cyclic polynomials
- Comparision with classical primitives
- Concluding remarks

ML-based
post-quantum primitives

NIST : PQ primitives

August 2024

- ML-KEM (CRYSTALS-Kyber) FIPS 203
- ML-DSA (CRYSTALS-Dilithium) FIPS 204
- SLH-DSA (SPHINCS+) FIPS 205

- Drafts for the future standards were available since 2023.

Pedagogical implementation

Impetus

Enough polynomials and linear algebra to implement Kyber

(F. Valsorda, blog post 7/11/2023)

Initial goal

- Provide feedback on drafts
- Compare with standard implementations
- Assess performances and « cost of quantum-resistance »
  - Development effort
  - Scaling of infrastructures
  - Migration to new primitives

Learning with errors

Asymetrical problem

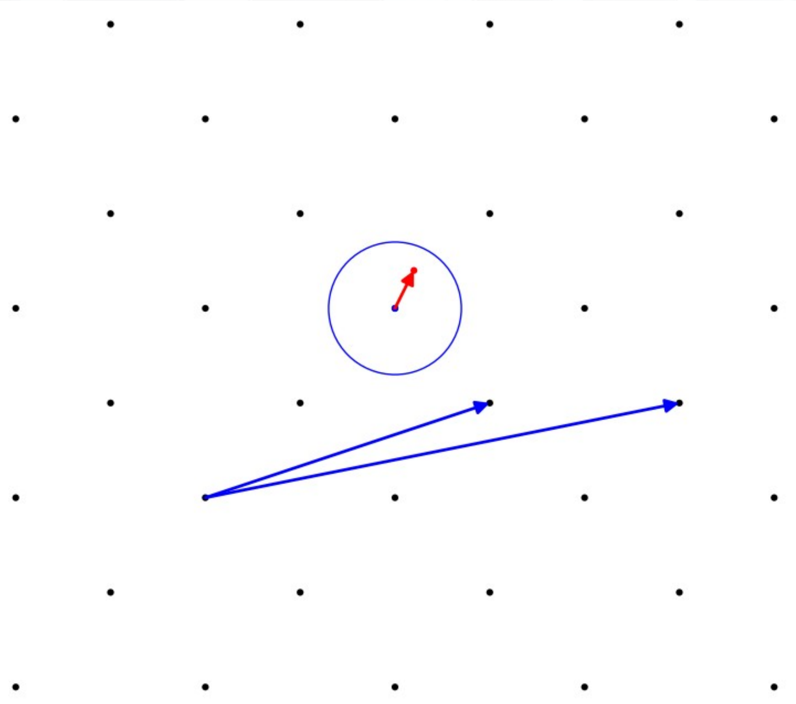
Given a vector $\mathbf{x}$ and matrix $\mathbf{A}$, it's easy to compute $\mathbf{y} = \mathbf{A}\,\mathbf{x}$ (matrix multiplication)

Given the result $\mathbf{y}$ and matrix $\mathbf{A}$, it is still relatively easy to recover $\mathbf{x}$ (Gaussian elimination)

However, if noise is added $\mathbf{y} = \mathbf{A}\,\mathbf{x} + \mathbf{e}$

then it becomes to solve the *noisy system of linear equations*

$$\mathbf{y} \approx \mathbf{A}\,\mathbf{x}$$



High-level description

Public-key encryption

Private key : a secret vector $\mathbf{x}$

Public key : a matrix $\mathbf{A}$ and vector $\mathbf{y}$ such that

$$\mathbf{y} \approx \mathbf{A}\,\mathbf{x}$$

To **encrypt** a message m :

compute $c = m + \mathbf{u} \cdot \mathbf{y}$

where $\mathbf{u}$ is a randomly chosen (row) vector

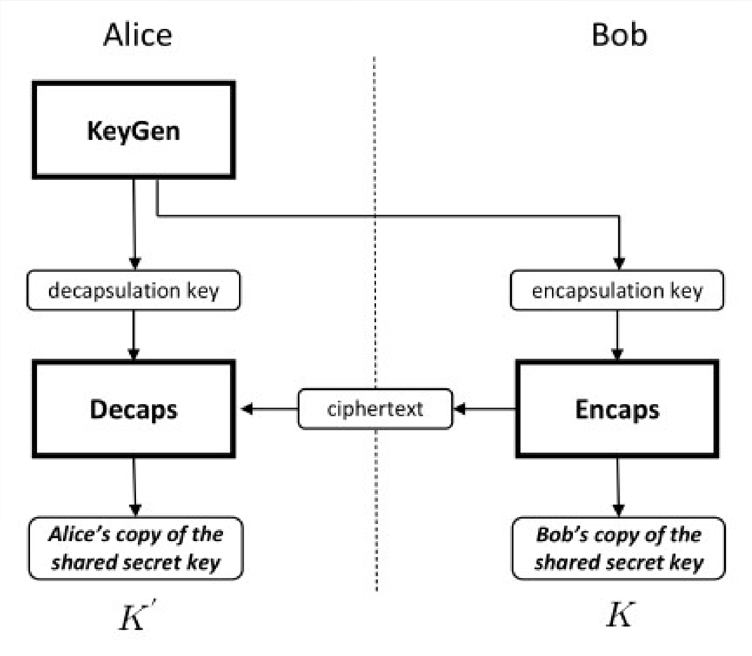
and provide $\mathbf{v} \approx \mathbf{u}\,\mathbf{A}$ as well.

To **decrypt** : compute $\mathbf{u} \cdot \mathbf{y} \approx \mathbf{u}\,(\mathbf{A}\,\mathbf{x}) = (\mathbf{u}\,\mathbf{A})\,\mathbf{x} \approx \mathbf{v} \cdot \mathbf{x}$

to remove it from c and take out extra noise.

ML-KEM and ML-DSA

ML-KEM : Applies the Fujisaki-Okamoto transform to a simple public-key encryption (PKE) scheme as above.



ML-DSA : applies a version of the Fiat-Shamir transform to the asymetrical problem in order to obtain a Schnorr-like signature scheme.

ML = module lattice

Noise is added as low-order bits of coefficients

Instead of integral linear combinations, components are taken in some polynomial ring

=> **MLWE problem**

The arithmetic of cyclic polynomials

Cyclic polynomials

Given a ring R of coefficients, consider polynomial expressions

$$f(X) = f_0 + f_1 X + f_2 X^2 + \cdots$$

with coefficients in R and for which we convene $X^N = 1$

for some integer power N .

Elements of the ring $R[X]/(X^N - 1)$ can be viewed as

vectors with N components arranged in a circle where

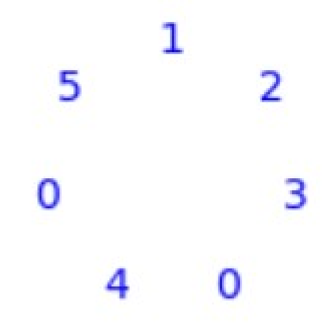
- addition is permormed component-wise

- multiplication by X is a circular permutation

- multiplication in general corresponds to « circular convolution ».

Example : $N = 7$

$$f(X) = 1 + 2X + 3X^2 + 4X^4 + 5X^6$$



Fast convolution

Polynomial multiplication of length-N circular polynomials

takes $\mathcal{O}(N^2)$ operations.

Component-wise multiplication takes only $\mathcal{O}(N)$ ...

And *Fourier transforms* convert convolutions into regular multiplication!

But discrete Fourier transforms take $\mathcal{O}(N^2)$ operations in general...

Unless a *Fast Fourier Transform* (Cooley-Tukey) algorithm is applied, which takes only

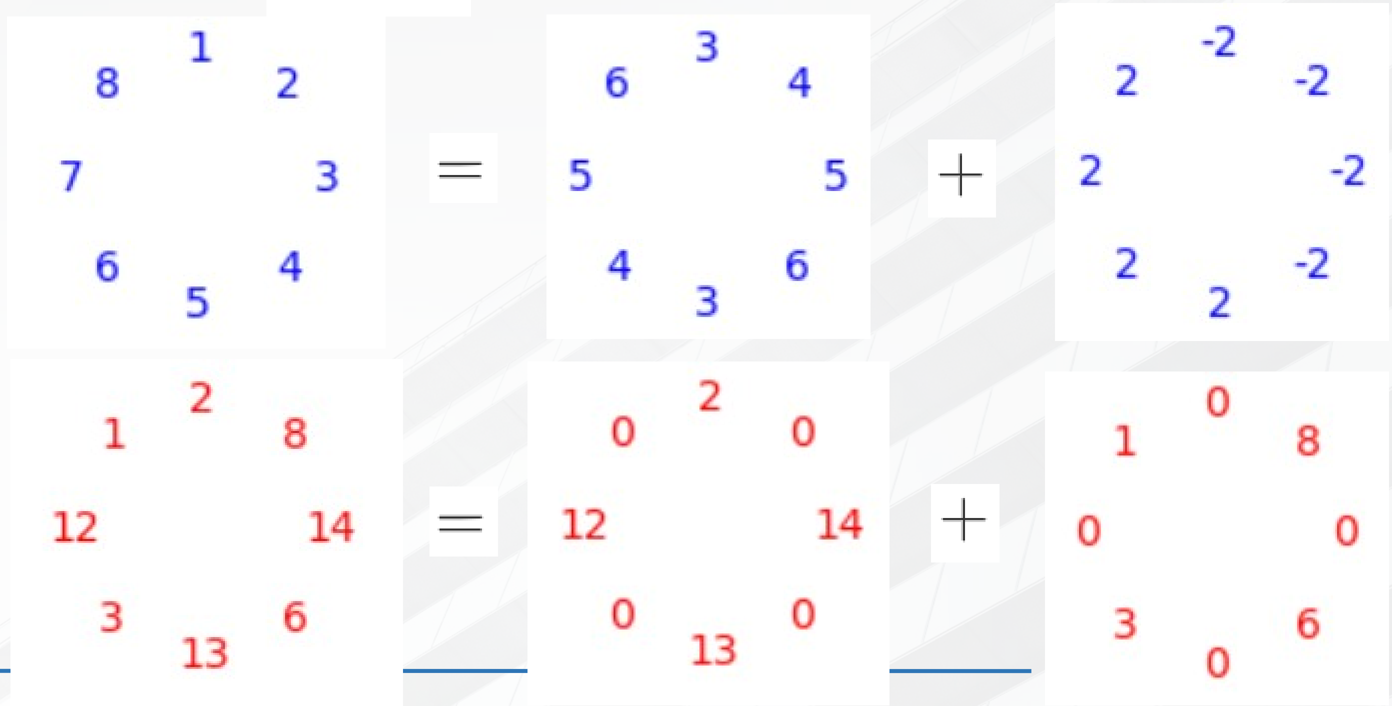
$$\mathcal{O}(N \log N)$$

operations, making circular convolution only marginally slower than regular multiplication.

Antiperiodic polynomials

When $N = 2n$ is even, any circular polynomial can be decomposed into a *periodic* and *antiperiodic* part corresponding to the decomposition of the Fourier transform into *even* and *odd* components.

Example with $N = 8$:



$\zeta = 2 \bmod 17$

Choice of parameters

ML-KEM and ML-DSA work with rings of antiperiodic polynomials $R[X]/(X^n+1)$

In order to compute the Fourier transform, a N-th root of unity ζ

must be chosen in R.

Number-Theoretic Transform : $R = \mathbb{Z}/q\mathbb{Z}$ with $N \mid (q-1)$

ML-DSA :

- $n = 256$
- $q = 8380417 = 2^{23} - 2^{13} + 1$
- $\zeta = 1753$, a 512^{th} root of unity

ML-KEM :

- $n = 256$
- $q = 3329 = 2^8 \cdot 13 + 1$
- $\zeta = 17$, only a 256^{th} root of unity

...

Comparision with classical primitives

Classical primitives

RSA, Diffie-Hellman, DSA, ... : based on **modular exponentiation** with large integers

security level (bits)	public key size (bits)
128	3072
256	15360

EC-based : multiplication on elliptic curves

security level (bits)	public key size (bits)
128	256
256	512

ML-based primitives

ML-KEM

security level (bits)	public key size (bits)
128	13056
256	25344

ML-DSA

security level (bits)	public key size (bits)
128	10496
256	20736

Concluding remarks

ML-based primitives

- These new primitives are considerably harder to understand
- Publication of test vectors and expected values will help developpers comply with the standard
- Having a pedagogical implementation allowing one to easily play with small (insecure!) values will help explain and teach how these primitives work
- Working (?) C++ implementation:
- `https://github.com/chenevert/ML-KEM`

Partners

Scientific-Practical Conference: “Telecommunication and Security”

The conference is funded within the conference grants competition CG-24-220 of the National Science Foundation of Georgia.

Thank you !

- **Email**: gabriel.chenevert@junia.com
- **Webpage**: https://junia.ovh/gch
- **Linkedin:** yes! find me