ETSI/IQC Quantum Safe Cryptography Event

Practical Improvements on BKZ Algorithm

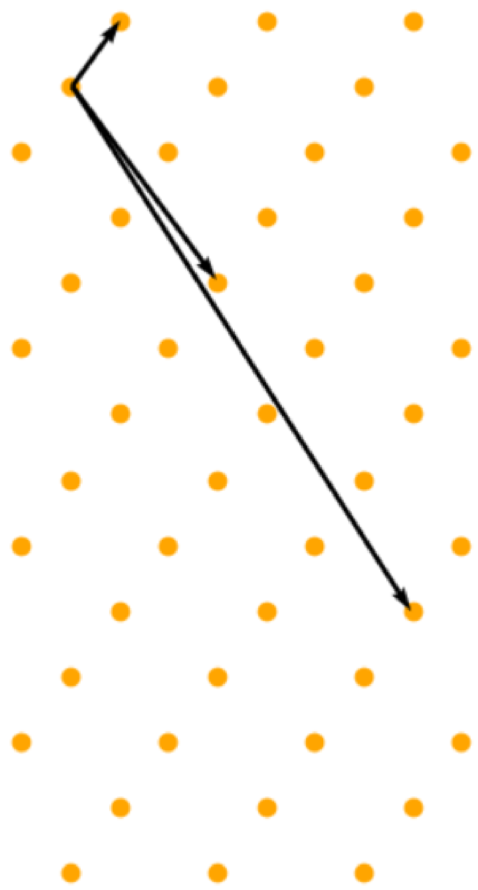
Presented by: Jintai Ding, Ziyu Zhao

15/02/2023

Introduction

Last year, NIST announced first 4 Post Quantum Cryptographic Algorithms. Most of them (Kyber, Dilithium and Falcon) are lattice-based.

- Lattice is discrete subgroup in R^m.
- A lattice L always admits an integral basis. Each point in L can be represented uniquely as an integral linear combination of the basis.
- The security of lattice-based cryptography is mainly based on the hardness of finding short vectors in some lattices.



Introduction

Lattice is attractive for its:

- efficiency, fast encryption and decryption, small key size…
- flexibility, fully homomorphic encryption, functional encryption…
- security, no efficient (quantum) algorithm is known for SVP, provable security…

Introduction

But we do not really understand the practical security of lattice. We need to predict the cost of concrete lattice attacks.

- For small lattices: Sieving & Enum
- Now BKZ is the most efficient algorithm to compute short vectors in large lattices (dimension ~ 1000).
- BKZ is widely used for the security analysis of lattice-based cryptographic algorithms.

Introduction

BKZ algorithm:

- Calls the SVP algorithms (Sieving or Enum) on d dimensional local projected lattices for several times.
- Outputs a rather short vector v.
- Achieves the same root Hermite factor as the SVP subroutines.

$$\left(\frac{||\mathbf{v}||}{\det(L)^{\frac{1}{n}}}\right)^{\frac{1}{n}} \approx \left(\sqrt{\frac{d}{2\pi e}}\right)^{\frac{1}{d}}$$

Introduction

We give four techniques on BKZ, which will provide about 10 times speedup.

- All the lattice-based NIST PQC candidates lose 3 ~ 4 bits of security in concrete attacks.

- We solved some lattice challenges in https://www.latticechallenge.org/ideallattice-challenge The details are listed below:

dim	length	Hermite factor	total cost	based on
656	670275	1.00993^{700}	380 CPUhours	Enum
700	659874	1.00928^{700}	1787 CPUhours	Sieving

Local Basis Processing

It's always a good choice to use local basis processing instead of inserting a single short vector.

- Compute the transform matrix of local processing (on the local projected lattice).
- Apply it on the vectors of the original basis then size reduce the basis.
- The quality of the basis can still be simulated efficiently.
- Mentioned in literature (e.g. [ADH+19]) for sieving based BKZ.

Jumping strategy

What will happen if we work on $L_{[i+s,j+s]}$ after $L_{[i,j]}$?

- The number of the SVP subroutines in each BKZ tour is only 1/s as before.
- How to evaluate the quality?
- Let $B = (b_1, b_2, \cdots, b_n)$ is a basis of L, $(b_1^*, b_2^*, \cdots, b_n^*)$ is the Gram-Schmidt orthogonalization, $B_i = ||b_i^*||^2$. We consider:

$$\mathrm{Pot}(L) = \prod_{i=1}^{n} B_i^{n+1-i}$$

Jumping strategy

- If Geometric Series Assumption (see [Sch03]) is true, Pot is an increase function of $||b_1||$.
- We want to make Pot decrease as fast as possible.
- Run binary search on d and s (by simulation) to find the optimal choice.
- We may get a speed up of $2^{1.65}$ if we use an HKZ-reduction with time complexity $2^{0.386d}$ as the SVP subroutine.

Jumping strategy

MSD	68	69	70	71	72	73	74	75
cpu hours	2.30	2.74	3.27	3.88	4.69	5.69	6.93	8.52
$\Delta \log_2 \text{Pot}$	463	758	1166	1400	1910	2254	2674	2949
$\frac{\Delta \log_2 \text{Pot}}{\text{cost}}$	201	277	357	361	407	396	385	346

(MSD, jumping step)	(72, 1)	(73, 2)	(74, 3)	(75, 4)	(76, 5)	(77, 6)	(78, 7)
cpu hours	4.69	2.84	2.31	2.13	2.20	2.30	2.51
$\Delta \log_2 \text{Pot}$	1910	1797	1787	1962	2059	2084	2241
$\frac{\Delta \log_2 \text{Pot}}{\text{cost}}$	407	633	773	920	930	906	858

Reduce Only When We Need

In practice, we don't need the whole reduced basis.

For the last ⌈n/d⌉ tours of the algorithm, we don't need to visit all the indexes.

Algorithm 2: The last several tours of our BKZ

Input: an n-dimensional lattice L, blocksize d and an SVP algorithm

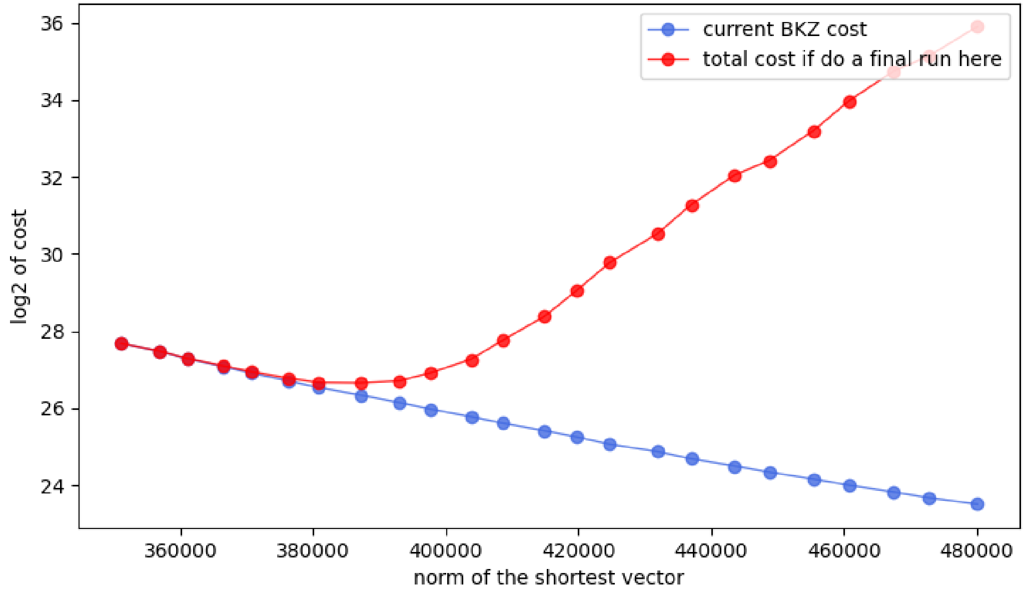
Output: a reduced basis

```
m = ⌈n/d⌉;
for k = 1, 2, ···, m do
    // a BKZ tour on L_[1,n−kd+1]
    for i = 1, 2, ···, n − kd + 1 do
        reduce L_[i,i+d−1] by the SVP algorithm;
return L
```

A Large Final Run

We can choose a much larger dimension d' in the last SVP subroutine (working on [1, d']) to get a much shorter vector.

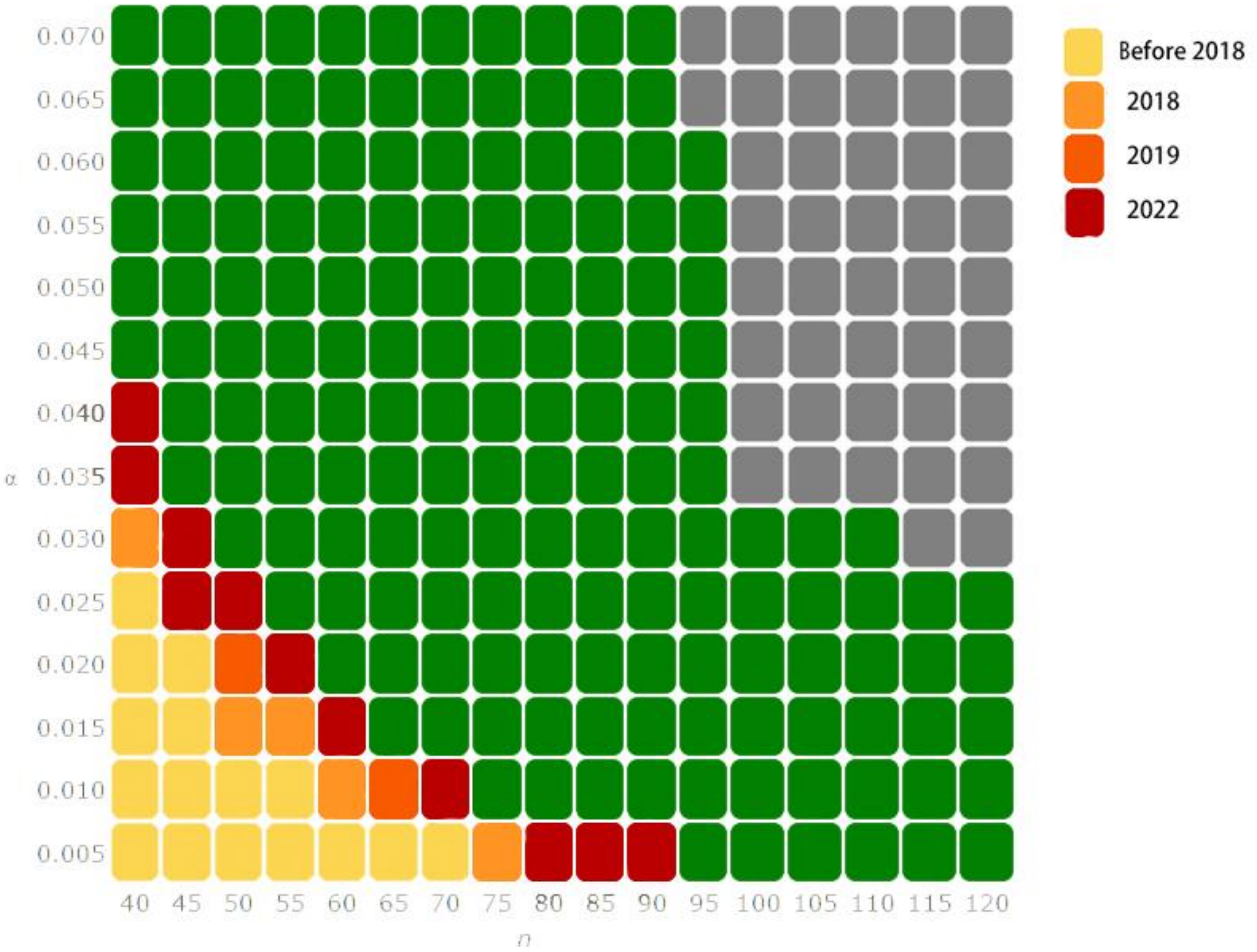
- Save the time for several tours of BKZ with a normal blocksize, about 1 bit.
- One can use the simulator to choose the optimal strategy.



ETSI/IQC Quantum Safe Cryptography Event

We need more work on practical lattice attacks.

- The picture shows the LWE Challenge in recent years.



Thank you!

Questions?