Faster Dual Lattice Attacks by Using Coding Theory

Kevin Carrier, Yixin Shen and Jean-Pierre Tillich

ETIS Laboratory, CY Cergy-Paris University
Royal Holloway, University of London
Project COSMIQ, Inria de Paris

November 21, 2022

Learning with errors (LWE)

Let $n = 4$, $m = 6$ and $q = 17$.

secret

$$A \in \mathbb{Z}_q^{m \times n} \qquad s \in \mathbb{Z}_q^{n} \qquad b \in \mathbb{Z}_q^{m}$$

$$\begin{bmatrix} 14 & 12 & 2 & 5 \\ 5 & 3 & 1 & 7 \\ 14 & 7 & 2 & 5 \\ 0 & 9 & 8 & 4 \\ 8 & 11 & 5 & 12 \\ 5 & 1 & 3 & 14 \end{bmatrix} \times \begin{bmatrix} \ \\ \ \\ \ \\ \ \end{bmatrix} = \begin{bmatrix} 11 \\ 5 \\ 14 \\ 6 \\ 12 \\ 13 \end{bmatrix}$$

Given A and b, find s.

Learning with errors (LWE)

Let $n = 4$, $m = 6$ and $q = 17$.

secret

$$
\underset{A \in \mathbb{Z}_q^{m\times n}}{\begin{bmatrix} 14 & 12 & 2 & 5 \\ 5 & 3 & 1 & 7 \\ 14 & 7 & 2 & 5 \\ 0 & 9 & 8 & 4 \\ 8 & 11 & 5 & 12 \\ 5 & 1 & 3 & 14 \end{bmatrix}} \times \underset{s \in \mathbb{Z}_q^{n}}{\begin{bmatrix} 1 \\ 2 \\ 1 \\ 5 \end{bmatrix}} = \underset{b \in \mathbb{Z}_q^{m}}{\begin{bmatrix} 11 \\ 5 \\ 14 \\ 6 \\ 12 \\ 13 \end{bmatrix}}
$$

Given A and b, find s.

$\rightsquigarrow$ Very easy (e.g. Gaussian elimination) and in polynomial time

Learning with errors (LWE)

Let $n = 4$, $m = 6$ and $q = 17$.

random		**secret**		**noise**		
$A \in \mathbb{Z}_q^{m \times n}$		$s \in \mathbb{Z}_q^n$		$e \in \mathbb{Z}_q^m$		$b \in \mathbb{Z}_q^m$

$$
\begin{bmatrix} 14 & 12 & 2 & 5 \\ 5 & 3 & 1 & 7 \\ 14 & 7 & 2 & 5 \\ 0 & 9 & 8 & 4 \\ 8 & 11 & 5 & 12 \\ 5 & 1 & 3 & 14 \end{bmatrix} \times \begin{bmatrix} 1 \\ 2 \\ 1 \\ 5 \end{bmatrix} + \begin{bmatrix} -3 \\ -1 \\ 2 \\ -3 \\ 3 \\ -1 \end{bmatrix} = \begin{bmatrix} 11 \\ 5 \\ 14 \\ 6 \\ 12 \\ 13 \end{bmatrix}
$$

Learning with errors (LWE)

Let $n = 4$, $m = 6$ and $q = 17$.

random **secret** **noise**

$$A \in \mathbb{Z}_q^{m \times n} \qquad s \in \mathbb{Z}_q^{n} \qquad e \in \mathbb{Z}_q^{m} \qquad b \in \mathbb{Z}_q^{m}$$

$$\begin{bmatrix} 14 & 12 & 2 & 5 \\ 5 & 3 & 1 & 7 \\ 14 & 7 & 2 & 5 \\ 0 & 9 & 8 & 4 \\ 8 & 11 & 5 & 12 \\ 5 & 1 & 3 & 14 \end{bmatrix} \times \begin{bmatrix} \; \\ \; \\ \; \\ \; \end{bmatrix} + \begin{bmatrix} \; \\ \; \\ \; \\ \; \\ \; \\ \; \end{bmatrix} = \begin{bmatrix} 11 \\ 5 \\ 14 \\ 6 \\ 12 \\ 13 \end{bmatrix}$$

Given A and b, find s.

$\leadsto$ Suspected hard problem, even for quantum algorithms

Learning with errors (LWE)

Let $n, m, q \in \mathbb{Z}$ and χ_e, χ_s two distributions over $\mathbb{Z}_q$.

LWE(n, m, q, χ_e, χ_s): probability distribution on $\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$

- sample $A \leftarrow U(\mathbb{Z}_q^{m\times n})$
- sample $s \leftarrow \chi_s^n$
- sample $e \leftarrow \chi_e^m$
- output $(A, As + e)$.

Intuition: $As + e$ is very close to a uniform distribution.

Learning with errors (LWE)

Let $n, m, q \in \mathbb{Z}$ and χ_e, χ_s two distributions over $\mathbb{Z}_q$.

LWE(n, m, q, χ_e, χ_s): probability distribution on $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$

- sample $A \leftarrow U(\mathbb{Z}_q^{m \times n})$
- sample $s \leftarrow \chi_s^n$
- sample $e \leftarrow \chi_e^m$
- output $(A, As + e)$.

Intuition: $As + e$ is very close to a uniform distribution.

Search LWE problem: given $(A, b) \leftarrow \text{LWE}(n, m, q, \chi_e, \chi_s)$, recover s.

Decision LWE problem:
distinguish LWE(n, m, q, χ_e, χ_s) from $U(\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m)$.

Learning with errors (LWE)

Let $n, m, q \in \mathbb{Z}$ and χ_e, χ_s two distributions over $\mathbb{Z}_q$.

LWE$(n, m, q, \chi_e, \chi_s)$: probability distribution on $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$

- sample $A \leftarrow U(\mathbb{Z}_q^{m \times n})$
- sample $s \leftarrow \chi_s^n$
- sample $e \leftarrow \chi_e^m$
- output $(A, As + e)$.

Intuition: $As + e$ is very close to a uniform distribution.

Search LWE problem: given $(A, b) \leftarrow \text{LWE}(n, m, q, \chi_e, \chi_s)$, recover s.

Decision LWE problem:
distinguish LWE$(n, m, q, \chi_e, \chi_s)$ from $U(\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m)$.

Lemma: Search LWE is easy if and only if decision LWE is easy.

Learning with errors (LWE)

LWE$(n, m, q, \chi_e, \chi_s)$: probability distribution on $\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$

- sample $A \leftarrow U(\mathbb{Z}_q^{m\times n})$
- sample $s \leftarrow \chi_s^n$
- sample $e \leftarrow \chi_e^m$
- output $(A, As + e)$.

Learning with errors (LWE)

LWE$(n, m, q, \chi_e, \chi_s)$: probability distribution on $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$

- sample $A \leftarrow U(\mathbb{Z}_q^{m \times n})$
- sample $s \leftarrow \chi_s^n$
- sample $e \leftarrow \chi_e^m$
- output $(A, As + e)$.

Secret distributions χ_s:

- originally uniform in $\mathbb{Z}_q$
- now some distribution of small deviation σ_s (e.g. discrete Gaussian/centered Binormial, $\{-1, 0, 1\}$ whp)
- Fact: small secret is as hard as uniform secret
- small secret allows more efficient schemes

Learning with errors (LWE)

LWE(n, m, q, χ_e, χ_s): probability distribution on $\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$

- sample $A \leftarrow U(\mathbb{Z}_q^{m\times n})$
- sample $s \leftarrow \chi_s^n$
- sample $e \leftarrow \chi_e^m$
- output $(A, As + e)$.

Noise distributions χ_e:

- usually discrete Gaussian/centered Binormial of deviation σ_e
- most schemes (Kyber/Saber/...): σ_e small (≈ 1)

LWE: security and attacks

LWE is fundamental to lattice-based cryptography:

- several lattice-based NIST PQC candidates rely on LWE
- extensive literature
- all evidence points to resistance against quantum attacks

LWE: security and attacks

LWE is fundamental to lattice-based cryptography:

- several lattice-based NIST PQC candidates rely on LWE
- extensive literature
- all evidence points to resistance against quantum attacks

Two types of attacks:

- Primal attacks:
  - more efficient in most cases
- Dual attacks:
  - originally less efficient, now catching up

Contribution: improvement on dual attacks using ideas from codes

Search to distinguish

Very naive attack:

$$\overset{A}{\begin{bmatrix} 8 & 9 & 10 & 12 \\ 5 & 3 & 11 & 3 \\ 0 & 15 & 10 & 4 \\ 15 & 9 & 16 & 15 \\ 1 & 2 & 10 & 8 \\ 11 & 16 & 13 & 9 \end{bmatrix}} \times \overset{s}{\begin{bmatrix} \; \\ \; \\ \; \\ \; \end{bmatrix}} + \overset{e}{\begin{bmatrix} \; \\ \; \\ \; \\ \; \\ \; \\ \; \end{bmatrix}} = \overset{b}{\begin{bmatrix} 8 \\ 3 \\ 11 \\ 15 \\ 2 \\ 15 \end{bmatrix}}$$

Attack:

- get (A, b)

Search to distinguish

Very naive attack: guess secret $\tilde{s}$

$$\underbrace{\begin{bmatrix} 8 & 9 & 10 & 12 \\ 5 & 3 & 11 & 3 \\ 0 & 15 & 10 & 4 \\ 15 & 9 & 16 & 15 \\ 1 & 2 & 10 & 8 \\ 11 & 16 & 13 & 9 \end{bmatrix}}_{A} \times \left(\underbrace{\begin{bmatrix} \; \\ \; \\ \; \\ \; \end{bmatrix}}_{s} - \underbrace{\begin{bmatrix} 3 \\ 11 \\ 0 \\ 14 \end{bmatrix}}_{\tilde{s}} \right) + \underbrace{\begin{bmatrix} \; \\ \; \\ \; \\ \; \\ \; \\ \; \end{bmatrix}}_{e} = \underbrace{\begin{bmatrix} 7 \\ 2 \\ 3 \\ 15 \\ 11 \\ 8 \end{bmatrix}}_{b'}$$

Attack:

- get (A, b)
- guess $\tilde{s}$
- output $b' = b - A\tilde{s}$

Search to distinguish

Very naive attack: guess secret $\tilde{s}$

$$\underbrace{\begin{bmatrix} 8 & 9 & 10 & 12 \\ 5 & 3 & 11 & 3 \\ 0 & 15 & 10 & 4 \\ 15 & 9 & 16 & 15 \\ 1 & 2 & 10 & 8 \\ 11 & 16 & 13 & 9 \end{bmatrix}}_{A} \times \left(\underbrace{\begin{bmatrix} \ \\ \ \\ \ \\ \ \end{bmatrix}}_{s} - \underbrace{\begin{bmatrix} 3 \\ 11 \\ 0 \\ 14 \end{bmatrix}}_{\tilde{s}} \right) + \underbrace{\begin{bmatrix} \ \\ \ \\ \ \\ \ \\ \ \\ \ \end{bmatrix}}_{e} = \underbrace{\begin{bmatrix} 7 \\ 2 \\ 3 \\ 15 \\ 11 \\ 8 \end{bmatrix}}_{b'}$$

Attack:

- get (A, b)
- guess $\tilde{s}$
- output $b' = b - A\tilde{s}$

Good guess ($s = \tilde{s}$):

$$b' = e$$

follows a discrete Gaussian
of small deviation

Search to distinguish

Very naive attack: guess secret $\tilde{s}$

$$\overset{A}{\begin{pmatrix}8&9&10&12\\5&3&11&3\\0&15&10&4\\15&9&16&15\\1&2&10&8\\11&16&13&9\end{pmatrix}} \times \left(\overset{s}{\begin{pmatrix}\ \\ \ \\ \ \\ \ \end{pmatrix}} - \overset{\tilde{s}}{\begin{pmatrix}3\\11\\0\\14\end{pmatrix}} \right) + \overset{e}{\begin{pmatrix}\ \\ \ \\ \ \\ \ \\ \ \\ \ \end{pmatrix}} = \overset{b'}{\begin{pmatrix}7\\2\\3\\15\\11\\8\end{pmatrix}}$$

Attack:

- get (A, b)
- guess $\tilde{s}$
- output $b' = b - A\tilde{s}$

Good guess ($s = \tilde{s}$):

$$b' = e$$

follows a discrete Gaussian of small deviation

Bad guess ($s \neq \tilde{s}$):

$$b' = e + A(s - \tilde{s})$$

follows a uniform[1] distribution (A uniform in $\mathbb{Z}_q^{m \times n}$)

[1] Technically only true for fixed s, random A and $\tilde{s}$

Uniform/Gaussian distinguisher

Given a sampler for χ^m, decide if $\chi = U(\mathbb{Z}_q)$ or $D_{\sigma,q}$ (discrete Gaussian)

Uniform/Gaussian distinguisher

Given a sampler for χ^m, decide if $\chi = U(\mathbb{Z}_q)$ or $D_{\sigma,q}$ (discrete Gaussian)

The entries are independent: given a sample from χ^m we obtain m independent samples from χ.
$\rightsquigarrow$ if m large enough, we know how to distinguish.

Uniform/Gaussian distinguisher

Given a sampler for χ, decide if $\chi = U(\mathbb{Z}_q)$ or $D_{\sigma,q}$ (discrete Gaussian)

Uniform/Gaussian distinguisher

Given a sampler for χ, decide if $\chi = U(\mathbb{Z}_q)$ or $D_{\sigma,q}$ (discrete Gaussian)

Essentially optimal distingusher: use Fourier transform

$$\mathbb{E}_{x \leftarrow \chi}[e^{2i\pi x/q}], \mathrm{Var}_{x \leftarrow \chi}[e^{2i\pi x/q}] \approx \begin{cases} 0, 0 & \text{if } \chi = U(\mathbb{Z}_q) \\ e^{-2\left(\frac{\pi\sigma}{q}\right)^2}, e^{-8\left(\frac{\pi\sigma}{q}\right)^2} & \text{if } \chi = D_{\sigma,q} \end{cases}$$

Uniform/Gaussian distinguisher

Given a sampler for χ, decide if $\chi = U(\mathbb{Z}_q)$ or $D_{\sigma,q}$ (discrete Gaussian)

Essentially optimal distingusher: use Fourier transform

$$\mathbb{E}_{x\leftarrow\chi}[e^{2i\pi x/q}], \mathrm{Var}_{x\leftarrow\chi}[e^{2i\pi x/q}] \approx \begin{cases} 0, 0 & \text{if } \chi = U(\mathbb{Z}_q) \\ e^{-2\left(\frac{\pi\sigma}{q}\right)^2}, e^{-8\left(\frac{\pi\sigma}{q}\right)^2} & \text{if } \chi = D_{\sigma,q} \end{cases}$$

Attack:

- sample $N = \Omega\left(1/\varepsilon^2\right)$ values $x_1, \ldots, x_N$ from χ
- compute

$$S = \frac{1}{N}\sum_{j=1}^{N} e^{2i\pi x_j/q}$$

- Check if $S > e^{-2\left(\frac{\pi\sigma}{q}\right)^2}$

The quantity $\varepsilon = e^{-2\left(\frac{\pi\sigma}{q}\right)^2}$ is called the advantage.

Very naive attack: summary

Very naive attack:

- guess $\tilde{s}$: q^n possiblities
- compute $1/\varepsilon^2$ samples to check guess

Very naive attack: summary

Very naive attack:

- guess $\tilde{s}$: q^n possiblities
- compute $1/\varepsilon^2$ samples to check guess

Complexity estimate:

$$q^n \cdot e^{4\left(\frac{\pi\sigma_e}{q}\right)^2} = \text{too much}$$

Very naive attack: summary

Very naive attack:

- guess $\tilde{s}$: q^n possiblities
- compute $1/\varepsilon^2$ samples to check guess

Complexity estimate:

$$q^n \cdot e^{4\left(\frac{\pi\sigma_e}{q}\right)^2} = \text{too much}$$

Can do better by guessing s in decreasing order of probability[1]:

$$G(\chi_s^n) \cdot e^{4\left(\frac{\pi\sigma_e}{q}\right)^2} \leqslant (1.22\sqrt{2\pi}\sigma_s)^n \cdot e^{4\left(\frac{\pi\sigma_e}{q}\right)^2} = \text{too much}$$

where σ_s deviation of s, $G(\cdot)$ = guessing complexity

[1] The complexity is now the expected running time

Very naive attack: summary

Very naive attack:

- guess $\tilde{s}$: q^n possiblities
- compute $1/\varepsilon^2$ samples to check guess

Complexity estimate:

$$q^n \cdot e^{4\left(\frac{\pi\sigma_e}{q}\right)^2} = \text{too much}$$

Can do better by guessing s in decreasing order of probability[1]:

$$G(\chi_s^n) \cdot e^{4\left(\frac{\pi\sigma_e}{q}\right)^2} \leqslant (1.22\sqrt{2\pi}\sigma_s)^n \cdot e^{4\left(\frac{\pi\sigma_e}{q}\right)^2} = \text{too much}$$

where σ_s deviation of s, $G(\cdot) =$ guessing complexity

Dual attacks: provide an efficient way to only guess a part of the secret

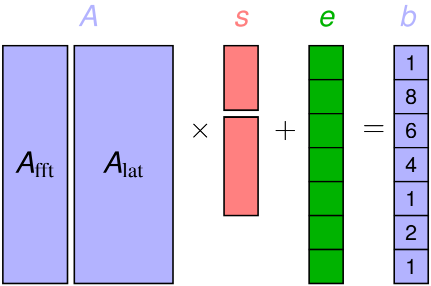
[1] The complexity is now the expected running time

Search to Decision LWE

$$\overset{A}{\begin{bmatrix} 3 & 7 & 2 & 3 & 6 \\ 4 & 1 & 5 & 8 & 4 \\ 1 & 8 & 1 & 8 & 1 \\ 5 & 2 & 5 & 6 & 0 \\ 2 & 1 & 6 & 3 & 0 \\ 8 & 2 & 7 & 3 & 6 \\ 5 & 5 & 6 & 6 & 2 \end{bmatrix}} \times \overset{s}{\begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}} + \overset{e}{\begin{bmatrix} \\ \\ \\ \\ \\ \\ \end{bmatrix}} = \overset{b}{\begin{bmatrix} 1 \\ 8 \\ 6 \\ 4 \\ 1 \\ 2 \\ 1 \end{bmatrix}}$$

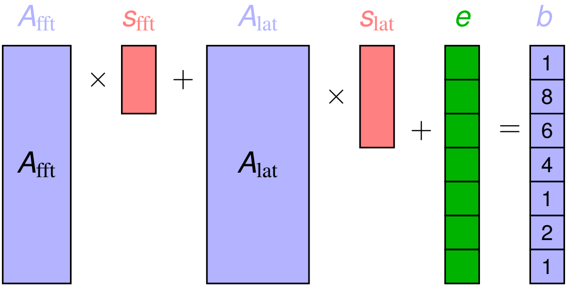
Search to Decision LWE

Split secret: $n = k_{\text{fft}} + k_{\text{lat}}$



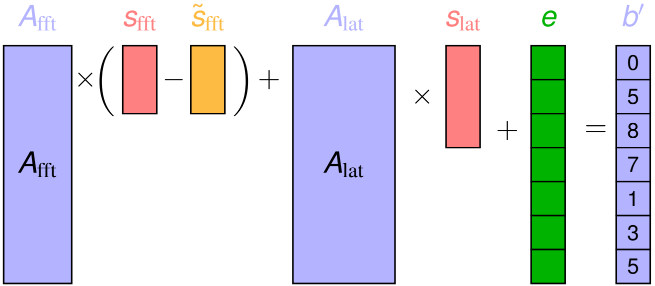
Search to Decision LWE

Split secret: $n = k_{\text{fft}} + k_{\text{lat}}$



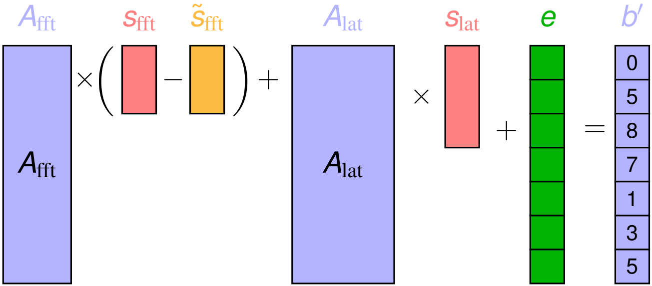
Search to Decision LWE

Split secret: $n = k_{\text{fft}} + k_{\text{lat}}$, guess $\tilde{s}_{\text{fft}}$, output $(A_{\text{lat}}, b' = b - A_{\text{fft}}\tilde{s}_{\text{fft}})$



Search to Decision LWE

Split secret: $n = k_{\text{fft}} + k_{\text{lat}}$, guess $\tilde{s}_{\text{fft}}$, output $(A_{\text{lat}}, b' = b - A_{\text{fft}}\tilde{s}_{\text{fft}})$



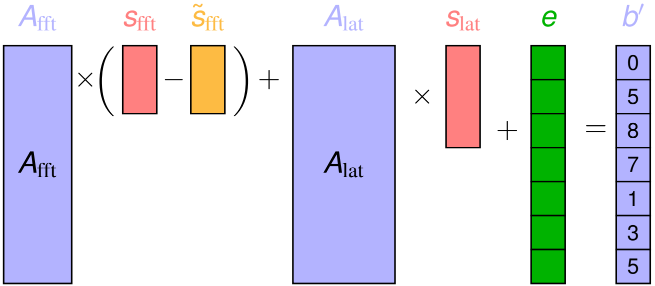
Good guess ($s_{\text{fft}} = \tilde{s}_{\text{fft}}$):

$$b' = A_{\text{lat}} s_{\text{lat}} + e$$

so (A_{lat}, b') follows an LWE distribution

Search to Decision LWE

Split secret: $n = k_{\text{fft}} + k_{\text{lat}}$, guess $\tilde{s}_{\text{fft}}$, output $(A_{\text{lat}}, b' = b - A_{\text{fft}}\tilde{s}_{\text{fft}})$



Good guess ($s_{\text{fft}} = \tilde{s}_{\text{fft}}$):

$$b' = A_{\text{lat}} s_{\text{lat}} + e$$

so (A_{lat}, b') follows an LWE distribution

Bad guess ($s_{\text{fft}} \neq \tilde{s}_{\text{fft}}$):

$$b' = A_{\text{fft}}(s_{\text{fft}} - \tilde{s}_{\text{fft}}) + \cdots$$

so (A_{lat}, b') follows a uniform distribution (A_{fft} uniform)

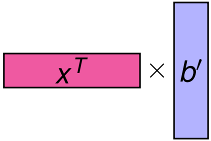
Uniform/LWE distinguisher

Given a sampler for χ, decide if $\chi =$ uniform or LWE.

Uniform/LWE distinguisher

Given a sampler for χ, decide if $\chi =$ uniform or LWE.

- sample (A_{lat}, b') from χ
- compute $x \in \mathbb{Z}_q^m$ such that $x^T A_{\mathrm{lat}} = 0$
- output $x^T b'$



Uniform/LWE distinguisher

Given a sampler for χ, decide if $\chi =$ uniform or LWE.

- sample (A_{lat}, b') from χ
- compute $x \in \mathbb{Z}_q^m$ such that $x^T A_{\text{lat}} = 0$
- output $x^T b'$

$$x^T \times b' = x^T \times \left(A_{\text{lat}} \times s_{\text{lat}} + e \right) = x^T \times e$$

Uniform/LWE distinguisher

Given a sampler for χ, decide if $\chi =$ uniform or LWE.

- sample (A_{lat}, b') from χ
- compute $x \in \mathbb{Z}_q^m$ such that $x^T A_{\mathrm{lat}} = 0$
- output $x^T b'$

$$x^T \times b' = x^T \times \left(A_{\mathrm{lat}} \times s_{\mathrm{lat}} + e \right) = x^T \times e$$

When $\chi =$ LWE:

$$x^T b' = x^T e$$

follows an approximate
Gaussian distribution

Uniform/LWE distinguisher

Given a sampler for χ, decide if $\chi =$ uniform or LWE.

- sample (A_{lat}, b') from χ
- compute $x \in \mathbb{Z}_q^m$ such that $x^T A_{\mathrm{lat}} = 0$
- output $x^T b'$

$$x^T \times b' = x^T \times \left(A_{\mathrm{lat}} \times s_{\mathrm{lat}} + e \right) = x^T \times e$$

When $\chi =$ LWE:

$$x^T b' = x^T e$$

follows an approximate Gaussian distribution

When $\chi =$ Uniform:

$$x^T b'$$

follows a uniform distribution (b' uniform, independent from A_{lat})

Dual attack: naive complexity

Naive dual attack:

- split secret $n = k_{\text{fft}} + k_{\text{lat}}$
- compute dual vectors x and dot products $x^T b$
- guess $\tilde{s}_{\text{fft}}$, subtract guess
- compute $1/\varepsilon^2$ samples to check guess

Dual attack: naive complexity

Naive dual attack:

- split secret $n = k_{\mathrm{fft}} + k_{\mathrm{lat}}$
- compute dual vectors x and dot products $x^T b$
- guess $\tilde{s}_{\mathrm{fft}}$, subtract guess
- compute $1/\varepsilon^2$ samples to check guess

What is ε ?

- e approx Gaussian deviation σ_e
- $x^T b = x^T e$ approx Gaussian deviation $\|x\|\sigma_e$

Dual attack: naive complexity

Naive dual attack:

- split secret $n = k_{\text{fft}} + k_{\text{lat}}$
- compute dual vectors x and dot products $x^T b$
- guess $\tilde{s}_{\text{fft}}$, subtract guess
- compute $1/\varepsilon^2$ samples to check guess

What is ε ?

- e approx Gaussian deviation σ_e
- $x^T b = x^T e$ approx Gaussian deviation $\|x\|\sigma_e$

Complexity estimate:

$$q^{k_{\text{fft}}} \cdot e^{4\left(\frac{\pi\|x\|\sigma_e}{q}\right)^2} + (\text{time to compute many } x)$$

Dual attack: naive complexity

Naive dual attack:

- split secret $n = k_{\mathrm{fft}} + k_{\mathrm{lat}}$
- compute dual vectors x and dot products $x^T b$
- guess $\tilde{s}_{\mathrm{fft}}$, subtract guess
- compute $1/\varepsilon^2$ samples to check guess

What is ε ?

- e approx Gaussian deviation σ_e
- $x^T b = x^T e$ approx Gaussian deviation $\|x\|\sigma_e$

Complexity estimate:

$$q^{k_{\mathrm{fft}}} \cdot e^{4\left(\frac{\pi\|x\|\sigma_e}{q}\right)^2} + (\text{time to compute many } x)$$

$\rightsquigarrow$ we want x to be short

Dual attack: naive complexity

Naive dual attack:

- split secret $n = k_{\text{fft}} + k_{\text{lat}}$
- compute dual vectors x and dot products $x^T b$
- guess $\tilde{s}_{\text{fft}}$, subtract guess
- compute $1/\varepsilon^2$ samples to check guess

What is ε ?

- e approx Gaussian deviation σ_e
- $x^T b = x^T e$ approx Gaussian deviation $\|x\|\sigma_e$

Complexity estimate:

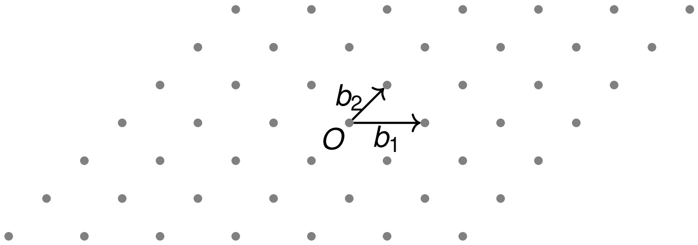
$$q^{k_{\text{fft}}} \cdot e^{4\left(\frac{\pi\|x\|\sigma_e}{q}\right)^2} + (\text{time to compute many } x)$$

$\rightsquigarrow$ we want x to be short $\rightsquigarrow$ lattice reduction

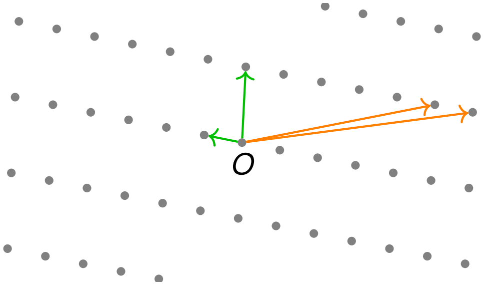
What is a (Euclidean) lattice?

Definition

$\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) = \left\{\sum_{i=1}^{n} x_i \boldsymbol{b}_i : x_i \in \mathbb{Z}\right\}$ where $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ is a basis of $\mathbb{R}^n$.

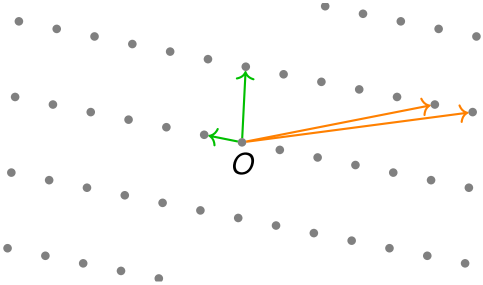


Lattice-based cryptography: fundamental idea



- good basis: private information, makes problem easy
- bad basis: public information, makes problem hard

Lattice-based cryptography: fundamental idea



- good basis: private information, makes problem easy
- bad basis: public information, makes problem hard

Basis reduction: transform a bad basis into a good one
Main tool: BKZ algorithm and its variants

Requires to solve the (approx-)SVP problem in smaller dimensions.

An important optimization

- $b' = b - A_{\mathrm{fft}}\tilde{s}_{\mathrm{fft}}$ comes from search to distinguish reduction
- $x_1, \dots, x_N$ is a list of dual vectors
- $\alpha_j = {x_j}^T b'$ comes from uniform/LWE to uniform/Gaussian red.

To distinguish between unidimensional uniform/Gaussian, we compute

$$F(\tilde{s}_{\mathrm{fft}}) = \sum_{j=1}^{N} e^{\frac{2i\pi}{q}\alpha_j}$$

An important optimization

- $b' = b - A_{\text{fft}}\tilde{s}_{\text{fft}}$ comes from search to distinguish reduction
- $x_1, \ldots, x_N$ is a list of dual vectors
- $\alpha_j = {x_j}^T b'$ comes from uniform/LWE to uniform/Gaussian red.

To distinguish between unidimensional uniform/Gaussian, we compute

$$F(\tilde{s}_{\text{fft}}) = \sum_{j=1}^{N} e^{\frac{2i\pi}{q}\alpha_j} = \sum_{j=1}^{N} e^{\frac{2i\pi}{q}{x_j}^T(b - A_{\text{fft}}\tilde{s}_{\text{fft}})} = \sum_{j=1}^{N} e^{\frac{2i\pi}{q}{x_j}^T b} \cdot e^{-\frac{2i\pi}{q}{x_j}^T A_{\text{fft}}\tilde{s}_{\text{fft}}}$$

Observation: $F(\tilde{s}_{\text{fft}}) = \widehat{T}(\tilde{s}_{\text{fft}})$ Fourier transform of $T({x_j}^T A_{\text{fft}}) = e^{\frac{2i\pi}{q}{x_j}^T b}$

An important optimization

- $b' = b - A_{\text{fft}}\tilde{s}_{\text{fft}}$ comes from search to distinguish reduction
- $x_1, \ldots, x_N$ is a list of dual vectors
- $\alpha_j = {x_j}^T b'$ comes from uniform/LWE to uniform/Gaussian red.

To distinguish between unidimensional uniform/Gaussian, we compute

$$F(\tilde{s}_{\text{fft}}) = \sum_{j=1}^{N} e^{\frac{2i\pi}{q}\alpha_j} = \sum_{j=1}^{N} e^{\frac{2i\pi}{q}{x_j}^T(b - A_{\text{fft}}\tilde{s}_{\text{fft}})} = \sum_{j=1}^{N} e^{\frac{2i\pi}{q}{x_j}^T b} \cdot e^{-\frac{2i\pi}{q}{x_j}^T A_{\text{fft}}\tilde{s}_{\text{fft}}}$$

Observation: $F(\tilde{s}_{\text{fft}}) = \widehat{T}(\tilde{s}_{\text{fft}})$ Fourier transform of $T({x_j}^T A_{\text{fft}}) = e^{\frac{2i\pi}{q}{x_j}^T b}$

Algorithm:

- $T \leftarrow k$-dimensional array set to zero
- $T[{x_j}^T A_{\text{fft}}] \leftarrow e^{2i\pi {x_j}^T b/q}$ for all j
- compute FFT $\widehat{T}$ of T
- check all $\widehat{T}[\tilde{s}_{\text{fft}}]$ against threshold

Complexity: array filling time + FFT time + search time $= O(N + q^{k_{\text{fft}}})$

Dual attack: summary

- split secret $n = k_{\mathrm{fft}} + k_{\mathrm{lat}}$
- compute many dual vectors x
- find $\tilde{s}_{\mathrm{fft}}$ using FFT

Dual attack: summary

- split secret $n = k_{\text{fft}} + k_{\text{lat}}$
- compute many dual vectors x
- find $\tilde{s}_{\text{fft}}$ using FFT

Pick x short in lattice L using BKZ:

$$L = \left\{ x \in \mathbb{Z}^m : x^T A_{\text{lat}} = 0 \bmod q \right\}$$

Complexity estimate:

$$q^{k_{\text{fft}}} + e^{4\left(\frac{\pi \|x\| \sigma_e}{q}\right)^2} + T_{\text{BKZ}}$$

Dual attack: summary

- split secret $n = k_{\mathrm{fft}} + k_{\mathrm{lat}}$
- compute many dual vectors x
- find $\tilde{s}_{\mathrm{fft}}$ using FFT

Pick x short in lattice L using BKZ:

$$L = \left\{ x \in \mathbb{Z}^m : x^T A_{\mathrm{lat}} = 0 \bmod q \right\}$$

Complexity estimate:

$$q^{k_{\mathrm{fft}}} + e^{4\left(\frac{\pi \|x\| \sigma_e}{q}\right)^2} + T_{\mathrm{BKZ}}$$

- BKZ trade-off: short $x \rightsquigarrow$ more expensive algorithm
- best dual attack parameters ($k_{\mathrm{fft}}, \ldots$) found by optimization

Advanced dual attacks

Modulo switching: only guess part of secret modulo p ($p \ll q$)

- reduce guessing complexity
- increase distinguishing cost due to modulo remainders
- makes reduced secret dense

Advanced dual attacks

Modulo switching: only guess part of secret modulo p ($p \ll q$)

- reduce guessing complexity
- increase distinguishing cost due to modulo remainders
- makes reduced secret dense

Hybrid attack: split secret into three parts

- s_{enum}: brute force enumeration by decreasing probability
- s_{fft}: guess by FFT
- s_{lat}: removed by dual attack

Advanced dual attacks

Modulo switching: only guess part of secret modulo p ($p \ll q$)

- reduce guessing complexity
- increase distinguishing cost due to modulo remainders
- makes reduced secret dense

Hybrid attack: split secret into three parts

- s_{enum}: brute force enumeration by decreasing probability
- s_{fft}: guess by FFT
- s_{lat}: removed by dual attack

BKZ with sieving

- obtain many dual vectors at once
- reducing the number of BKZ reductions

Hybrid dual attack

Combine enumeration with dual attack:

- enumerate $s_{\text{enum}} \in \mathbb{Z}_q^{k_{\text{enum}}}$ sampled from $\chi_s^{k_{\text{enum}}}$
  - enumerate all $s_{\text{fft}} \in \mathbb{Z}_q^{k_{\text{fft}}}$ uniform in $\mathbb{Z}_q^{k_{\text{fft}}}$
    - compute a DFT-like sum
    - check if it is above the threshold

Hybrid dual attack

Combine enumeration with dual attack:

- enumerate $s_{\text{enum}} \in \mathbb{Z}_q^{k_{\text{enum}}}$ — sampled from $\chi_s^{k_{\text{enum}}}$
  - enumerate all $s_{\text{fft}} \in \mathbb{Z}_q^{k_{\text{fft}}}$ — uniform in $\mathbb{Z}_q^{k_{\text{fft}}}$
    - compute a DFT-like sum
    - check if it is above the threshold

- guessing complexity: try s_{enum} in decreasing order of probability
- FFT: compute all DFT-sums in one go with a FFT
- dual vectors: compute them once, reuse for all s_{enum}

$$G(\chi_s^{k_{\text{enum}}}) \cdot \left(q^{k_{\text{fft}}} + e^{4\left(\frac{\pi \|x\| \sigma_e}{q}\right)^2} \right) + T_{\text{BKZ}}$$

Hybrid dual attack

Combine enumeration with dual attack:

- enumerate $s_{\mathrm{enum}} \in \mathbb{Z}_q^{k_{\mathrm{enum}}}$ sampled from $\chi_s^{k_{\mathrm{enum}}}$
  - enumerate all $s_{\mathrm{fft}} \in \mathbb{Z}_q^{k_{\mathrm{fft}}}$ uniform in $\mathbb{Z}_q^{k_{\mathrm{fft}}}$
    - compute a DFT-like sum
    - check if it is above the threshold

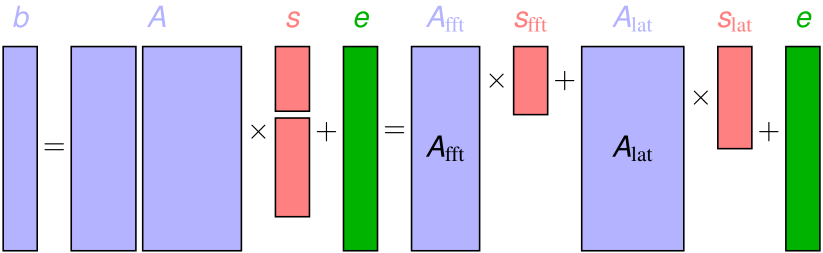
- guessing complexity: try s_{enum} in decreasing order of probability
- FFT: compute all DFT-sums in one go with a FFT
- dual vectors: compute them once, reuse for all s_{enum}

$$G(\chi_s^{k_{\mathrm{enum}}}) \cdot \left(q^{k_{\mathrm{fft}}} + e^{4\left(\frac{\pi \|x\| \sigma_e}{q}\right)^2} \right) + T_{\mathrm{BKZ}}$$

Gain: reduce $k_{\mathrm{lat}} \rightsquigarrow$ decrease BKZ cost

Recall: split secret + dual vector

Combine: split secret



Recall: split secret + dual vector

Combine: split secret

$$b = A \times s + e = A_{\text{fft}} \times s_{\text{fft}} + A_{\text{lat}} \times s_{\text{lat}} + e$$

With: dual vector x such that $x^T A_{\text{lat}} = 0$

$$x^T \times b = x^T \times A_{\text{fft}} \times s_{\text{fft}} + x^T \times e$$

Fundamental equation of dual attack

- split secret, find (x, y) such that $x^T A_{\text{lat}} = 0$ and $y^T = x^T A_{\text{fft}}$

$$x^T \times b = y^T \times s_{\text{fft}} + x^T \times e$$

Fundamental equation of dual attack

- split secret, find (x, y) such that $x^T A_{\text{lat}} = 0$ and $y^T = x^T A_{\text{fft}}$
- guess secret $\tilde{s}$ and subtract

$$x^T \times b - y^T \times \tilde{s}_{\text{fft}} = y^T \times \left(s_{\text{fft}} - \tilde{s}_{\text{fft}} \right) + x^T \times e$$

Fundamental equation of dual attack

- split secret, find (x, y) such that $x^T A_{\text{lat}} = 0$ and $y^T = x^T A_{\text{fft}}$
- guess secret $\tilde{s}$ and subtract

$$\boxed{x^T} \times \boxed{b} - \boxed{y^T} \times \boxed{\tilde{s}_{\text{fft}}} = \boxed{y^T} \times \left(\boxed{s_{\text{fft}}} - \boxed{\tilde{s}_{\text{fft}}} \right) + \boxed{x^T} \times \boxed{e}$$

Good guess ($s_{\text{fft}} = \tilde{s_{\text{fft}}}$):

$$x^T e$$

follows a discrete Gaussian of
small deviation (depends on $\|x\|$)

Fundamental equation of dual attack

- split secret, find (x, y) such that $x^T A_{\mathrm{lat}} = 0$ and $y^T = x^T A_{\mathrm{fft}}$
- guess secret $\tilde{s}$ and subtract

$$x^T \times b - y^T \times \tilde{s}_{\mathrm{fft}} = y^T \times \left(s_{\mathrm{fft}} - \tilde{s}_{\mathrm{fft}} \right) + x^T \times e$$

Good guess ($s_{\mathrm{fft}} = \tilde{s_{\mathrm{fft}}}$):

$$x^T e$$

follows a discrete Gaussian of small deviation (depends on $\|x\|$)

Bad guess ($s_{\mathrm{fft}} \neq \tilde{s_{\mathrm{fft}}}$):

$$y^T (s_{\mathrm{fft}} - \tilde{s_{\mathrm{fft}}}) + x^T e$$

follows a uniform distribution ($y \approx$ uniform in $\mathbb{Z}_q^{k_{\mathrm{fft}}}$)

Fundamental equation of dual attack

- split secret, find (x, y) such that $x^T A_{\mathrm{lat}} = 0$ and $y^T = x^T A_{\mathrm{fft}}$
- guess secret $\tilde{s}$ and subtract

$$x^T \times b - y^T \times \tilde{s}_{\mathrm{fft}} = y^T \times \left(s_{\mathrm{fft}} - \tilde{s}_{\mathrm{fft}} \right) + x^T \times e$$

Good guess ($s_{\mathrm{fft}} = \tilde{s_{\mathrm{fft}}}$):

$$x^T e$$

follows a discrete Gaussian of small deviation (depends on $\|x\|$)

Bad guess ($s_{\mathrm{fft}} \neq \tilde{s_{\mathrm{fft}}}$):

$$y^T (s_{\mathrm{fft}} - \tilde{s_{\mathrm{fft}}}) + x^T e$$

follows a uniform distribution ($y \approx$ uniform in $\mathbb{Z}_q^{k_{\mathrm{fft}}}$)

Problem: cost of distinguishing grows as $q^{k_{\mathrm{fft}}}$

$\rightsquigarrow$ can we change to a modulo $p \ll q$ to reduce the cost?

Modulo switching

- split secret, find (x, y) s.t. $x^T A_{\mathrm{lat}} = 0$ and $y^T = x^T A_{\mathrm{fft}}$, guess $\tilde{s}$

$$x^T \cdot b - y^T \cdot \tilde{s}_{\mathrm{fft}} = y^T \cdot \left(s_{\mathrm{fft}} - \tilde{s}_{\mathrm{fft}} \right) + x^T \cdot e \mod q$$

Modulo switching

- split secret, find (x, y) s.t. $x^T A_{\mathrm{lat}} = 0$ and $y^T = x^T A_{\mathrm{fft}}$, guess $\tilde{s}$
- change modulo to p

$$\frac{p}{q} \boxed{x^T} \cdot \boxed{b} - \frac{p}{q} \boxed{y^T} \cdot \boxed{\tilde{s}_{\mathrm{fft}}} = \frac{p}{q} \boxed{y^T} \cdot \left(\boxed{s_{\mathrm{fft}}} - \boxed{\tilde{s}_{\mathrm{fft}}} \right) + \frac{p}{q} \boxed{x^T} \cdot \boxed{e} \mod p$$

Modulo switching

- split secret, find (x, y) s.t. $x^T A_{\text{lat}} = 0$ and $y^T = x^T A_{\text{fft}}$, guess $\tilde{s}$
- change modulo to p

$$\frac{p}{q} x^T \cdot b - \frac{p}{q} y^T \cdot \tilde{s}_{\text{fft}} = \frac{p}{q} y^T \cdot \left(s_{\text{fft}} - \tilde{s}_{\text{fft}} \right) + \frac{p}{q} x^T \cdot e \mod p$$

Good guess ($s_{\text{fft}} = \tilde{s_{\text{fft}}}$):

$$\frac{p}{q} x^T e$$

follows a discrete Gaussian of small deviation (depends on $\|x\|$)

Bad guess ($s_{\text{fft}} \neq \tilde{s_{\text{fft}}}$):

$$\frac{p}{q} y^T (s_{\text{fft}} - \tilde{s_{\text{fft}}}) + \frac{p}{q} x^T e$$

follows a uniform distribution ($y \approx$ uniform in $\mathbb{Z}_q^{k_{\text{fft}}}$)

Modulo switching

- split secret, find (x, y) s.t. $x^T A_{\text{lat}} = 0$ and $y^T = x^T A_{\text{fft}}$, guess $\tilde{s}$
- change modulo to p

$$\frac{p}{q} x^T \cdot b - \frac{p}{q} y^T \cdot \tilde{s}_{\text{fft}} = \frac{p}{q} y^T \cdot \left(s_{\text{fft}} - \tilde{s}_{\text{fft}} \right) + \frac{p}{q} x^T \cdot e \mod p$$

Good guess ($s_{\text{fft}} = \tilde{s_{\text{fft}}}$):

$$\frac{p}{q} x^T e$$

follows a discrete Gaussian of small deviation (depends on $\|x\|$)

Bad guess ($s_{\text{fft}} \neq \tilde{s_{\text{fft}}}$):

$$\frac{p}{q} y^T (s_{\text{fft}} - \tilde{s_{\text{fft}}}) + \frac{p}{q} x^T e$$

follows a uniform distribution ($y \approx$ uniform in $\mathbb{Z}_q^{k_{\text{fft}}}$)

Problem: $\frac{p}{q} y^T$ is not integral

$\rightsquigarrow$ FFT distinguisher not applicable

Modulo switching (cont)

Notation: $[x]$ = integer part, $\{x\}$ = fractional part, $x = [x] + \{x\}$

$$\frac{p}{q} x^T \cdot b - \frac{p}{q} y^T \cdot \tilde{s}_{\mathrm{fft}} = \frac{p}{q} y^T \cdot \left(s_{\mathrm{fft}} - \tilde{s}_{\mathrm{fft}} \right) + \varepsilon \mod p$$

$$\text{where} \quad \varepsilon = \frac{p}{q} x^T \cdot e$$

Modulo switching (cont)

Notation: $[x]$ = integer part, $\{x\}$ = fractional part, $x = [x] + \{x\}$

$$\frac{p}{q} x^T \cdot b - \left[\frac{p}{q} y^T\right] \cdot \tilde{s}_{\mathrm{fft}} = \left[\frac{p}{q} y^T\right] \cdot \left(s_{\mathrm{fft}} - \tilde{s}_{\mathrm{fft}} \right) + \varepsilon \mod p$$

$$\text{where} \quad \varepsilon = \left\{\frac{p}{q} y^T\right\} \cdot s_{\mathrm{fft}} + \frac{p}{q} x^T \cdot e$$

Modulo switching (cont)

Notation: $[x]$ = integer part, $\{x\}$ = fractional part, $x = [x] + \{x\}$

$$\frac{p}{q}\, x^T \cdot b - \left[\frac{p}{q}\, y^T\right] \cdot \tilde{s}_{\mathrm{fft}} = \left[\frac{p}{q}\, y^T\right] \cdot \left(s_{\mathrm{fft}} - \tilde{s}_{\mathrm{fft}} \right) + \varepsilon \mod p$$

$$\text{where} \quad \varepsilon = \left\{\frac{p}{q}\, y^T\right\} \cdot s_{\mathrm{fft}} + \frac{p}{q}\, x^T \cdot e$$

Good guess ($s_{\mathrm{fft}} = \tilde{s}_{\mathrm{fft}}$):

$$\varepsilon = \left\{\frac{p}{q} x^T\right\} s_{\mathrm{fft}} + \frac{p}{q} x^T e$$

follows an almost discrete Gaussian of small deviation (now depends on $\|x\|$ and $\|s_{\mathrm{fft}}\|$)

Modulo switching (cont)

Notation: $[x]$ = integer part, $\{x\}$ = fractional part, $x = [x] + \{x\}$

$$\frac{p}{q} x^T \cdot b - \left[\frac{p}{q} y^T\right] \cdot \tilde{s}_{\mathrm{fft}} = \left[\frac{p}{q} y^T\right] \cdot \left(s_{\mathrm{fft}} - \tilde{s}_{\mathrm{fft}} \right) + \varepsilon \mod p$$

$$\text{where} \quad \varepsilon = \left\{\frac{p}{q} y^T\right\} \cdot s_{\mathrm{fft}} + \frac{p}{q} x^T \cdot e$$

Good guess ($s_{\mathrm{fft}} = \tilde{s}_{\mathrm{fft}}$):

$$\varepsilon = \{\tfrac{p}{q} x^T\} s_{\mathrm{fft}} + \tfrac{p}{q} x^T e$$

follows an almost discrete Gaussian of small deviation (now depends on $\|x\|$ and $\|s_{\mathrm{fft}}\|$)

Bad guess ($s_{\mathrm{fft}} \neq \tilde{s}_{\mathrm{fft}}$):

$$[\tfrac{p}{q} y^T](s_{\mathrm{fft}} - \tilde{s}_{\mathrm{fft}})$$

not obviously uniform, but saved by the hybrid search hinted at in this presentation

Modulo switching (cont)

Notation: $[x]$ = integer part, $\{x\}$ = fractional part, $x = [x] + \{x\}$

$$\frac{p}{q} x^T \cdot b - \left[\frac{p}{q} y^T\right] \cdot \tilde{s}_{\mathrm{fft}} = \left[\frac{p}{q} y^T\right] \cdot \left(s_{\mathrm{fft}} - \tilde{s}_{\mathrm{fft}} \right) + \varepsilon \mod p$$

where $\varepsilon = \left\{\frac{p}{q} y^T\right\} \cdot s_{\mathrm{fft}} + \frac{p}{q} x^T \cdot e$

Good guess ($s_{\mathrm{fft}} = \tilde{s}_{\mathrm{fft}}$):

$$\varepsilon = \left\{\frac{p}{q} x^T\right\} s_{\mathrm{fft}} + \frac{p}{q} x^T e$$

follows an almost discrete Gaussian of small deviation (now depends on $\|x\|$ and $\|s_{\mathrm{fft}}\|$)

Bad guess ($s_{\mathrm{fft}} \neq \tilde{s}_{\mathrm{fft}}$):

$$\left[\frac{p}{q} y^T\right]\left(s_{\mathrm{fft}} - \tilde{s}_{\mathrm{fft}}\right)$$

not obviously uniform, but saved by the hybrid search hinted at in this presentation

Conclusion: it works but increases the number of samples:

from $4\left(\frac{\pi\|x\|\sigma_e}{q}\right)^2$ to $4\left(\frac{\pi\|x\|\sigma_e}{q}\right)^2 + \frac{1}{3}\left(\frac{\pi\|s_{\mathrm{fft}}\|q}{p}\right)^2$

Going further: using ideas from coding theory

Everyting until this point is in the LWE report by the MATZOV group.

Going further: using ideas from coding theory

Everyting until this point is in the LWE report by the MATZOV group.

Modulo switching: approximate a vector $x \in \mathbb{Z}_q^n$ by

$$x = \frac{q}{p} \cdot [\frac{p}{q}x] + \frac{q}{p}\{\frac{p}{q}x\} = \frac{q}{p} \cdot u + e$$

- $u \in \mathbb{Z}_p^n$: smaller domain (field is smaller)
- $\|e\| \leqslant \frac{q}{p}$: “small error”

Going further: using ideas from coding theory

Everyting until this point is in the LWE report by the MATZOV group.

Modulo switching: approximate a vector $x \in \mathbb{Z}_q^n$ by

$$x = \frac{q}{p} \cdot [\frac{p}{q} x] + \frac{q}{p}\{\frac{p}{q} x\} = \frac{q}{p} \cdot u + e$$

- $u \in \mathbb{Z}_p^n$: smaller domain (field is smaller)
- $\|e\| \leqslant \frac{q}{p}$: “small error”

Our observation: this looks like a special case of lattice codes

$$x = Gu + t$$

- $G \in \mathbb{Z}_q^{n \times m}$: defines a code
- $u \in \mathbb{Z}_q^m$: smaller domain (dimension is smaller)
- $\|t\|$ is small (depends on G)

Applying lattice codes

Recall: find (x, y) such that $x^T A_{\mathrm{lat}} = 0$ and $y^T = x^T A_{\mathrm{fft}}$

$$x^T \times b = y^T \times s_{\mathrm{fft}} + x^T \times e$$

Applying lattice codes

Recall: find (x, y) such that $x^T A_{\mathrm{lat}} = 0$ and $y^T = x^T A_{\mathrm{fft}}$

$$x^T \times b = y^T \times s_{\mathrm{fft}} + x^T \times e$$

Choose a code $G \in \mathbb{Z}_q^{k_{\mathrm{fft}} \times k_{\mathrm{cod}}}$, decode y as

$$y = G \times u + t$$

Applying lattice codes

Recall: find (x, y) such that $x^T A_{\text{lat}} = 0$ and $y^T = x^T A_{\text{fft}}$

$$x^T \times b = y^T \times s_{\text{fft}} + x^T \times e$$

Choose a code $G \in \mathbb{Z}_q^{k_{\text{fft}} \times k_{\text{cod}}}$, decode y as

$$y = G \times u + t$$

New fundamental equation:

$$x^T \cdot b = u^T \cdot G^T \cdot s_{\text{fft}} + t^T \cdot s_{\text{fft}} + x^T \cdot e$$

Lattice codes: fundamental equation

- find (x, y) such that $x^T A_{\mathrm{lat}} = 0$ and $y^T = x^T A_{\mathrm{fft}}$
- choose a code $G \in \mathbb{Z}_q^{k_{\mathrm{fft}} \times k_{\mathrm{cod}}}$, decode $y = Gu + t$

$$x^T \cdot b = u^T \cdot G^T \cdot s_{\mathrm{fft}} + t^T \cdot s_{\mathrm{fft}} + x^T \cdot e$$

Lattice codes: fundamental equation

- find (x, y) such that $x^T A_{\mathrm{lat}} = 0$ and $y^T = x^T A_{\mathrm{fft}}$
- choose a code $G \in \mathbb{Z}_q^{k_{\mathrm{fft}} \times k_{\mathrm{cod}}}$, decode $y = Gu + t$

$$x^T \cdot b = u^T \cdot s_{\mathrm{cod}} + \varepsilon'$$

where

$$s_{\mathrm{cod}} = G^T \cdot s_{\mathrm{fft}} \qquad \varepsilon' = t^T \cdot s_{\mathrm{fft}} + x^T \cdot e$$

Lattice codes: fundamental equation

- find (x, y) such that $x^T A_{\text{lat}} = 0$ and $y^T = x^T A_{\text{fft}}$
- choose a code $G \in \mathbb{Z}_q^{k_{\text{fft}} \times k_{\text{cod}}}$, decode $y = Gu + t$

$$x^T \cdot b = u^T \cdot s_{\text{cod}} + \varepsilon'$$

where

$$s_{\text{cod}} = G^T \cdot s_{\text{fft}} \qquad \varepsilon' = t^T \cdot s_{\text{fft}} + x^T \cdot e$$

Observations:

- we directly guess s_{cod} instead of s_{fft}
- $s_{\text{cod}} = G^T s_{\text{fft}} \in \mathbb{Z}_q^{k_{\text{cod}}}$ has **smaller dimension** $k_{\text{cod}} \ll k_{\text{fft}}$

Lattice codes: fundamental equation

- find (x, y) such that $x^T A_{\text{lat}} = 0$ and $y^T = x^T A_{\text{fft}}$
- choose a code $G \in \mathbb{Z}_q^{k_{\text{fft}} \times k_{\text{cod}}}$, decode $y = Gu + t$

$$x^T \cdot b = u^T \cdot s_{\text{cod}} + \varepsilon'$$

where

$$s_{\text{cod}} = G^T \cdot s_{\text{fft}} \qquad \varepsilon' = t^T \cdot s_{\text{fft}} + x^T \cdot e$$

Observations:

- we directly guess s_{cod} instead of s_{fft}
- $s_{\text{cod}} = G^T s_{\text{fft}} \in \mathbb{Z}_q^{k_{\text{cod}}}$ has **smaller dimension** $k_{\text{cod}} \ll k_{\text{fft}}$
- $\varepsilon = t^T s_{\text{fft}} + x^T e$ follows a discrete Gaussian whose **deviation** depends on $\|x\|$, $\|s_{\text{fft}}\|$ and $\|t\|$
- $\|t\|$ is **small** for a good code G

Lattice codes vs modulo switching

Lattice codes

$$x^T \cdot b = u^T \cdot s_{\text{cod}} + \varepsilon'$$

Modulo switching

$$x^T \cdot b = \left[\frac{p}{q}\, y^T\right] \cdot s_{\text{fft}} + \varepsilon$$

Lattice codes vs modulo switching

Lattice codes

$$x^T \cdot b = u^T \cdot s_{\text{cod}} + \varepsilon'$$

- FFT cost: $q^{k_{\text{cod}}}$
- error ε': Gaussian of stddev

$$\tau_{\text{MS}}^2 = \|x\|^2 \cdot \sigma_e^2 + \|s_{\text{fft}}\|^2 \cdot \frac{q^{2-2\frac{k_{\text{cod}}}{k_{\text{fft}}}}}{2\pi e}$$

for an asymptotically optimal code

Modulo switching

$$x^T \cdot b = \left[\frac{p}{q} \; y^T\right] \cdot s_{\text{fft}} + \varepsilon$$

- FFT cost: $p^{k_{\text{fft}}}$
- error ε: Gaussian of stddev

$$\tau_{\text{LC}}^2 = \|x\|^2 \cdot \sigma_e^2 + \|s_{\text{fft}}\|^2 \cdot \frac{q^2}{12p^2}$$

Lattice codes vs modulo switching

Lattice codes

$$x^T \cdot b = u^T \cdot s_{\mathrm{cod}} + \varepsilon'$$

- FFT cost: $q^{k_{\mathrm{cod}}}$
- error ε': Gaussian of stddev

$$\tau_{\mathrm{MS}}^2 = \|x\|^2 \cdot \sigma_e^2 + \|s_{\mathrm{fft}}\|^2 \cdot \frac{q^{2-2\frac{k_{\mathrm{cod}}}{k_{\mathrm{fft}}}}}{2\pi e}$$

for an asymptotically optimal code

Modulo switching

$$x^T \cdot b = \left[\frac{p}{q}\, y^T\right] \cdot s_{\mathrm{fft}} + \varepsilon$$

- FFT cost: $p^{k_{\mathrm{fft}}}$
- error ε: Gaussian of stddev

$$\tau_{\mathrm{LC}}^2 = \|x\|^2 \cdot \sigma_e^2 + \|s_{\mathrm{fft}}\|^2 \cdot \frac{q^2}{12p^2}$$

Comparison for same FFT cost: $q^{k_{\mathrm{cod}}} = p^{k_{\mathrm{fft}}}$

$$\frac{q^{2-2\frac{k_{\mathrm{cod}}}{k_{\mathrm{fft}}}}}{2\pi e} = \frac{q}{2\pi e p} \approx \frac{q}{17p} \ll \frac{q}{12p}$$

$\rightsquigarrow$ lattice codes are always better than modulo switching!

Other important details

- FFT is more efficient for powers of two
- $q^{k_{\mathrm{cod}}}$ has coarse granularity for big q

$\rightsquigarrow$ use modulo switching to change q to $p = 2^m$ then use lattice codes:
best of both, allow more "continuous" parameter choice

Other important details

- FFT is more efficient for powers of two
- $q^{k_{\mathrm{cod}}}$ has coarse granularity for big q

$\rightsquigarrow$ use modulo switching to change q to $p = 2^m$ then use lattice codes: best of both, allow more “continuous” parameter choice

- optimal codes are expensive but we need a fast decoder
- we only need to decode to a close codeword, not the closest

$\rightsquigarrow$ we suggest to use polar codes which are asymptotically optimal

Other important details

- FFT is more efficient for powers of two
- $q^{k_{\text{cod}}}$ has coarse granularity for big q

$\rightsquigarrow$ use modulo switching to change q to $p = 2^m$ then use lattice codes: best of both, allow more "continuous" parameter choice

- optimal codes are expensive but we need a fast decoder
- we only need to decode to a close codeword, not the closest

$\rightsquigarrow$ we suggest to use polar codes which are asymptotically optimal

- many parameters to choose (p, k_{fft}, k_{cod}, BKZ block size, ...)
- no obvious way to choose them

$\rightsquigarrow$ search for optimal parameters with an optimisation program

Prange bet: motivation

Overall attack so far:

- enumerate $s_{\mathrm{enum}} \in \mathbb{Z}_q^{k_{\mathrm{enum}}}$ sampled from $\chi_s^{k_{\mathrm{enum}}}$
  - perform dual attack with codes and modulo switching and check if s_{enum} was correct

Prange bet: motivation

Overall attack so far:

- enumerate $s_{\mathrm{enum}} \in \mathbb{Z}_q^{k_{\mathrm{enum}}}$ sampled from $\chi_s^{k_{\mathrm{enum}}}$
  - perform dual attack with codes and modulo switching and check if s_{enum} was correct

Expected complexity: $G \cdot T$

- $G =$ expected number of guesses to find s_{enum}
- $T =$ complexity of attack

Prange bet: motivation

Overall attack so far:

- enumerate $s_{\text{enum}} \in \mathbb{Z}_q^{k_{\text{enum}}}$ sampled from $\chi_s^{k_{\text{enum}}}$
  - perform dual attack with codes and modulo switching and check if s_{enum} was correct

Expected complexity: $G \cdot T$

- $G =$ expected number of guesses to find s_{enum}
- $T =$ complexity of attack

Prange bet:

- some values of s_{enum} are much more likely than others (e.g. 0)
- only enumerate a few most likely values
- if it fails, retry with a permutation of the secret

Prange bet: motivation

Overall attack so far:

- enumerate $s_{\text{enum}} \in \mathbb{Z}_q^{k_{\text{enum}}}$ sampled from $\chi_s^{k_{\text{enum}}}$
  - perform dual attack with codes and modulo switching and check if s_{enum} was correct

Expected complexity: $G \cdot T$

- $G =$ expected number of guesses to find s_{enum}
- $T =$ complexity of attack

Prange bet:

- some values of s_{enum} are much more likely than others (e.g. 0)
- only enumerate a few most likely values
- if it fails, retry with a permutation of the secret
- if we do not permute the lattice part (s_{lat}), we can even reuse the BKZ computation just like in the "normal attack"

Prange bet: implementation

New attack: fix betting set Bet

- for each permutation τ that leaves the "lat part " fixed
  - enumerate $s_{\mathrm{enum}} \in$ Bet
    - perform[1] dual attack on τ-permuted instance with codes and modulo switching and check if s_{enum} was correct

[1] Not shown here: dual vectors reused accross iterations since lat part untouched

Prange bet: implementation

New attack: fix betting set Bet

- for each permutation τ that leaves the "lat part " fixed
  - enumerate $s_{\text{enum}} \in$ Bet
    - perform[1] dual attack on τ-permuted instance with codes and modulo switching and check if s_{enum} was correct

Expected complexity: $P \cdot |\,\text{Bet}\,| \cdot T$

- $P =$ expected number of permutations needed
- $T =$ complexity of attack

[1] Not shown here: dual vectors reused accross iterations since lat part untouched

Prange bet: implementation

New attack: fix betting set Bet

- for each permutation τ that leaves the "lat part" fixed
  - enumerate $s_{\mathrm{enum}} \in$ Bet
    - perform[1] dual attack on τ-permuted instance with codes and modulo switching and check if s_{enum} was correct

Expected complexity: $P \cdot |\,\mathrm{Bet}\,| \cdot T$

- $P =$ expected number of permutations needed
- $T =$ complexity of attack

Which bet? $\mathrm{Bet} = \{0\}$ optimal in our case

[1] Not shown here: dual vectors reused accross iterations since lat part untouched

Results

- CC: classical circuit model (most detailed cost)
- CN: intermediate model
- C0: “Core-SVP” cost model

Scheme	MATZOV			Codes w/o Prange			Codes w/ Prange		
	CC	CN	C0	CC	CN	C0	CC	CN	C0
Kyber 512	138.5	133.7	114.8	137.8	133.0	114.0	137.5	132.6	113.9
Kyber 768	195.7	190.4	173.1	192.5	187.2	170.2	191.9	186.7	169.8
Kyber 1024	261.4	255.4	240.7	256.2	250.5	235.7	255.5	249.5	235.5
LightSaber	137.1	132.3	113.1	136.8	131.5	112.3	136.7	131.8	112.2
Saber	201.1	195.1	178.3	199.7	194.9	177.0	199.0	193.8	176.9
FireSaber	263.6	257.7	242.8	259.9	254.4	239.4	259.3	253.9	239.0

- 1 to 5 bit gain without Prange over MATZOV
- further 1 bit gain with Prange bet