All Along the Ring Tower
Algebraic Structures for Fun and Profit

Thomas Prest
joint work w/
{Léo Ducas} ∪ {Thomas Pornin} ∪ {Léo Ducas, Steven Galbraith, Yang Yu}

RISC × PROMETHEUS Seminar, 03/05/2019
I Introduction

II Three Case Studies
   i Generalized Bézout Equations
   ii Generalized Four Square Theorem
   iii Efficient Lattice Decoding

III Conclusion
It is typical in lattice-based cryptography to use matrices with coefficients in $\mathbb{Z}_q[x]/(x^d + 1)$ rather than $\mathbb{Z}_q$:

1. Communication costs typically go $O(d^2) \Rightarrow O(d)$
2. Computation costs typically go $O(d^2) \Rightarrow O(d \log d)$

But in some situations this additional structure seems ineffective:

1. Matrix decomposition (Cholesky, Gram-Schmidt, etc.)
2. Solving equations in a ring which is not a field (e.g. $\mathbb{Z}[x]/(x^d + 1)$)

Algorithms can take time up to $\Theta(d^2)$ or $\Theta(d^3)$. 
The State of Affairs

What naïve solutions do:

1. View $\mathbb{Q}[x]/(x^d + 1)$ as either a $\mathbb{Q}$-linear space of dimension $d$, an extension field of $\mathbb{Q}$ of degree $d$, etc.

2. This ignores the rich structure of cyclotomic rings and fields.

What happens when we open the black box?
For \( d \) a power-of-two, we note:

\[ Q_d = \mathbb{Q}[x]/(x^d + 1) \] the \( d \)-th cyclotomic field

\[ \mathcal{O}_d = \mathbb{Z}[x]/(x^d + 1) \] the \( d \)-th cyclotomic ring

We have this tower of fields:

\[ \mathbb{Q} \subset \mathbb{Q}_2 \subset \cdots \subset \mathbb{Q}_{d/2} \subset \mathbb{Q}_d \]

As well as this chain of isomorphisms:

\[ \mathbb{Q}^d \cong (\mathbb{Q}_2)^{d/2} \cong \cdots \cong (\mathbb{Q}_{d/2})^2 \cong \mathbb{Q}_d \]

At a high level:

\[ \Rightarrow \text{ The field norm and field trace allows to move in the tower of fields} \]

\[ \Rightarrow \text{ Ring isomorphisms allow us to move in the chain of ring isomorphisms} \]
Traces and Norms in Cyclotomic Fields

**Definition:** For a (finite) field extension \( L/K \):

\( \text{Tr}_{L/K} : L \to K \)
\( f \mapsto \sum_{\sigma \in \text{Gal}(L/K)} \sigma(f) \)

\( \text{N}_{L/K} : L \to K \)
\( f \mapsto \prod_{\sigma \in \text{Gal}(L/K)} \sigma(f) \)

**Concretely:** if \( f(x) = f_e(x^2) + x \cdot f_o(x^2) \in \mathbb{Q}_d \), then \( f^\times(x) = f(-x) \) and:

\( \text{Tr}_{\mathbb{Q}_d/\mathbb{Q}_{d/2}}(f) = f + f^\times \)
\( = 2 \cdot f_e(x^2) \)

\( \text{N}_{\mathbb{Q}_d/\mathbb{Q}_{d/2}}(f) = f \cdot f^\times \)
\( = f_e^2(x^2) - x^2 f_o^2(x^2) \)

**Composition properties:**

\( \text{Tr}_{L/K} \circ \text{Tr}_{M/L} = \text{Tr}_{M/K} \)

\( \text{N}_{L/K} \circ \text{N}_{M/L} = \text{N}_{M/K} \)

**Homomorphic properties:**

\( \text{Tr}_{L/K}(a+b) = \text{Tr}_{L/K}(a) + \text{Tr}_{L/K}(b) \)

\( \text{N}_{L/K}(a \cdot b) = \text{N}_{L/K}(a) \cdot \text{N}_{L/K}(b) \)
I Introduction

II Three Case Studies
   i Generalized Bézout Equations
   ii Generalized Four Square Theorem
   iii Efficient Lattice Decoding

III Conclusion
Problem 1 - Completing NTRU Bases

NTRU Lattices:

- Prevalent in lattice-based crypto
- Public key is $A = [1 \ | \ h ]$, for $h = g \times f^{-1} \mod (\varphi, q)$.
- Private key is $B$ such that $B \times A^t = 0 \mod (\varphi, q)$

Some schemes only require a partial trapdoor $B = [g \ | \ -f ]$:

- Fiat-Shamir [ZCHW17], encryption [SHRS17], FHE [LTV12, BLLN13]
Problem 1 - Completing NTRU Bases

NTRU Lattices:

- Prevalent in lattice-based crypto
- Public key is \( A = [ 1 \mid h ] \), for \( h = g \times f^{-1} \mod (\varphi, q) \).
- Private key is \( B \) such that \( B \times A^t = 0 \mod (\varphi, q) \)

Some schemes only require a partial trapdoor \( B = [ g \mid -f ] \):

- Fiat-Shamir [ZCHW17], encryption [SHRS17], FHE [LTV12, BLLN13]

However, some schemes require a full trapdoor \( B = \begin{bmatrix} g & -f \\ G & -F \end{bmatrix} \):

- Hash-then-sign [PFH+17], IBE [DLP14], HIBE [CG17]
- More generally, anything based on trapdoor sampling [GPV08]
Problem 1 - Completing NTRU Bases

NTRU Lattices:

- Prevalent in lattice-based crypto
- Public key is $A = \begin{bmatrix} 1 \\ h \end{bmatrix}$, for $h = g \times f^{-1} \mod (\phi, q)$.
- Private key is $B$ such that $B \times A^t = 0 \mod (\phi, q)$

Some schemes only require a partial trapdoor $B = \begin{bmatrix} g \\ -f \end{bmatrix}$:
- Fiat-Shamir [ZCHW17], encryption [SHRS17], FHE [LTV12, BLLN13]

However, some schemes require a full trapdoor $B = \begin{bmatrix} g & -f \\ G & -F \end{bmatrix}$:
- Hash-then-sign [PFH+17], IBE [DLP14], HIBE [CG17]
- More generally, anything based on trapdoor sampling [GPV08]

**Problem:** Given $f, g \in \mathbb{Z}[x]/(x^d + 1)$, find $F, G \in \mathbb{Z}[x]/(x^d + 1)$ such that:

$$f \cdot G - g \cdot F = q$$
If we can solve the problem projected over $\mathbb{Z}_{d/2}$, i.e.:

$$N_{\mathbb{Z}_{d/2}}(f) \cdot G' - N_{\mathbb{Z}_{d/2}}(g) \cdot F' = 1$$

for some $F', G'$, then we have this relationship over $\mathbb{Z}_d$:

$$f \cdot (f^x G') - g \cdot (g^x F') = 1$$

This leads to a simple algorithm:

1. Project
2. Solve
3. Lift
Outline of the Solver

\[ \mathbb{Z}_d \ni f, g \]

\[ \mathbb{Z}_d \setminus \mathbb{Z}_d/2 \setminus \mathbb{Z}_d/4 \setminus \ldots \setminus \mathbb{Z} \]
Outline of the Solver

\[ \mathbb{Z}_d \ni f, g \]
\[ \mathbb{Z}_d/2 \ni N_{\mathbb{Z}_d/\mathbb{Z}_d/2}(f), N_{\mathbb{Z}_d/\mathbb{Z}_d/2}(g) \]
\[ \vdots \]
\[ \mathbb{Z} \]
Outline of the Solver

\[ \mathbb{Z}_d \ni f, g \]
\[ \mathbb{U} \]
\[ \mathbb{Z}_{d/2} \ni N_{\mathbb{Z}_{d/2}}(f), N_{\mathbb{Z}_{d/2}}(g) \]
\[ \mathbb{U} \]
\[ \mathbb{Z}_{d/4} \ni N_{\mathbb{Z}_{d/4}}(f), N_{\mathbb{Z}_{d/4}}(g) \]
\[ \mathbb{U} \]
\[ \vdots \]
\[ \mathbb{U} \]
\[ \mathbb{Z} \]

At each lower level:
- The coefficients grow (in bitsize) by a factor 2...
- but the number of coefficients is divided by 2.

Space-saving trick: recompute lazily \( N_i(f) \), \( N_i(g) \) at each step.

Allows a linear memory-memory trade-off by a factor \( \ell = \log_2 n \).
Outline of the Solver

\[ \mathbb{Z}_d \ni f, g \]
\[ \mathbb{Z}_d/2 \ni \mathbb{N}_{\mathbb{Z}_d/2}(f), \mathbb{N}_{\mathbb{Z}_d/2}(g) \]
\[ \mathbb{Z}_d/4 \ni \mathbb{N}_{\mathbb{Z}_d/4}(f), \mathbb{N}_{\mathbb{Z}_d/4}(g) \]
\[ \vdots \]
\[ \mathbb{Z} \]

At each lower level:
- The coefficients grow (in bitsize) by a factor 2...
- ...but the number of coefficients is divided by 2.

Space-saving trick: recomputing lazily \( \mathbb{N}_i(f) \), \( \mathbb{N}_i(g) \) at each step.

Allows a linear-time memory trade-off by a factor \( \ell = \log_2 n \).
Outline of the Solver

\[
\begin{align*}
\mathbb{Z}_d &\ni f, g \\
\cup & \\
\mathbb{Z}_d/2 &\ni N_{\mathbb{Z}_d/\mathbb{Z}_d/2}(f), N_{\mathbb{Z}_d/\mathbb{Z}_d/2}(g) \\
\cup & \\
\mathbb{Z}_d/4 &\ni N_{\mathbb{Z}_d/\mathbb{Z}_d/4}(f), N_{\mathbb{Z}_d/\mathbb{Z}_d/4}(g) \\
\cup & \\
\vdots & \\
\cup & \\
\mathbb{Z} &\ni N_{\mathbb{Z}_d/\mathbb{Z}}(f), N_{\mathbb{Z}_d/\mathbb{Z}}(g)
\end{align*}
\]
Outline of the Solver

\[ \mathbb{Z}_d \ni f, g \]

\[ \mathbb{Z}_d / 2 \ni N_{\mathbb{Z}_d / 2} (f), N_{\mathbb{Z}_d / 2} (g) \]

\[ \mathbb{Z}_d / 4 \ni N_{\mathbb{Z}_d / 4} (f), N_{\mathbb{Z}_d / 4} (g) \]

\[ \vdots \]

\[ \mathbb{Z} \ni N_{\mathbb{Z}} (f), N_{\mathbb{Z}} (g) \rightarrow F[\ell], G[\ell] \]
Outline of the Solver

\[
\begin{align*}
\mathbb{Z}_d & \ni f, g \\
\cup \downarrow & \\
\mathbb{Z}_{d/2} & \ni N_{\mathbb{Z}_{d/2}}(f), N_{\mathbb{Z}_{d/2}}(g) \\
\cup \downarrow & \\
\mathbb{Z}_{d/4} & \ni N_{\mathbb{Z}_{d/4}}(f), N_{\mathbb{Z}_{d/4}}(g) \\
\cup \downarrow & \\
\vdots & \vdots \\
\cup \downarrow & \\
\mathbb{Z} & \ni N_{\mathbb{Z}/\mathbb{Z}}(f), N_{\mathbb{Z}/\mathbb{Z}}(g) & \rightarrow & F^{[\ell]}, G^{[\ell]}
\end{align*}
\]
Outline of the Solver

\[ \mathbb{Z}_d \ni f, g \]

\[ \mathbb{Z}_d/2 \ni N_{\mathbb{Z}_d/2}(f), N_{\mathbb{Z}_d/2}(g) \]

\[ \mathbb{Z}_d/4 \ni N_{\mathbb{Z}_d/4}(f), N_{\mathbb{Z}_d/4}(g) \rightarrow F^2, G^2 \]

\[ \vdots \]

\[ \mathbb{Z} \ni N_{\mathbb{Z}/\mathbb{Z}}(f), N_{\mathbb{Z}/\mathbb{Z}}(g) \rightarrow F^\ell, G^\ell \]
Outline of the Solver

\[ \mathbb{Z}_d \ni f, g \]

\[ \mathbb{Z}_d/2 \ni N_{\mathbb{Z}_d/\mathbb{Z}_d/2}(f), N_{\mathbb{Z}_d/\mathbb{Z}_d/2}(g) \rightarrow F[1], G[1] \]

\[ \mathbb{Z}_d/4 \ni N_{\mathbb{Z}_d/\mathbb{Z}_d/4}(f), N_{\mathbb{Z}_d/\mathbb{Z}_d/4}(g) \rightarrow F[2], G[2] \]

\[ \vdots \]

\[ \mathbb{Z} \ni N_{\mathbb{Z}_d/\mathbb{Z}}(f), N_{\mathbb{Z}_d/\mathbb{Z}}(g) \rightarrow F[\ell], G[\ell] \]

Space-saving trick: recompute lazily \( N_i(f) \); \( N_i(g) \) at each step. Allows a linear time-memory trade-off by a factor \( \ell = \log n \).
## Outline of the Solver

| \( \mathbb{Z}_d \) | \( f, g \) | \( F, G \) |
| \( \cup \uparrow \) |
| \( \mathbb{Z}_d/2 \) | \( N_{\mathbb{Z}_d/\mathbb{Z}_d/2}(f), N_{\mathbb{Z}_d/\mathbb{Z}_d/2}(g) \) | \( F[1], G[1] \) |
| \( \cup \uparrow \) |
| \( \mathbb{Z}_d/4 \) | \( N_{\mathbb{Z}_d/\mathbb{Z}_d/4}(f), N_{\mathbb{Z}_d/\mathbb{Z}_d/4}(g) \) | \( F[2], G[2] \) |
| \( \cup \uparrow \) |
| \vdots | \vdots | \vdots |
| \( \cup \uparrow \) |
| \( \mathbb{Z} \) | \( N_{\mathbb{Z}_d/\mathbb{Z}}(f), N_{\mathbb{Z}_d/\mathbb{Z}}(g) \) | \( F[\ell], G[\ell] \) |
### Outline of the Solver

| \( \mathbb{Z}_d \) \( \not\ni \) \( f, g \) | \( \mapsto \) | \( F, G \) |
| \( \cup \) | \( \downarrow \) | \( \uparrow \) |
| \( \mathbb{Z}_d/2 \) \( \not\ni \) \( N_{\mathbb{Z}_d/2}(f), N_{\mathbb{Z}_d/2}(g) \) | \( \mapsto \) | \( F[1], G[1] \) |
| \( \cup \) | \( \downarrow \) | \( \uparrow \) |
| \( \mathbb{Z}_d/4 \) \( \not\ni \) \( N_{\mathbb{Z}_d/4}(f), N_{\mathbb{Z}_d/4}(g) \) | \( \mapsto \) | \( F[2], G[2] \) |
| \( \cup \) | \( \downarrow \) | \( \uparrow \) |
| \( \vdots \) \( \vdots \) \( \vdots \) | \( \vdots \) |
| \( \cup \) | \( \downarrow \) | \( \uparrow \) |
| \( \mathbb{Z} \) \( \not\ni \) \( N_{\mathbb{Z}_d}(f), N_{\mathbb{Z}_d}(g) \) | \( \mapsto \) | \( F[\ell], G[\ell] \) |

At each lower level:

- The coefficients grow (in bitsize) by a factor 2...
- ... but the number of coefficients is divided by 2.

Space-saving trick: recompute lazily \( N^i(f), N^i(g) \) at each step
- Allows a linear time-memory trade-off by a factor \( \ell = \log n \)
Toy Example

sage: f8, g8
-x^7 + 3*x^6 - x^4 + 4*x^3 + 6*x^2 - 2*x - 4,
x^7 - x^6 - 2*x^5 - 4*x^3 - 3*x^2 - x + 7
sage: f4, g4
-51*x^3 + 51*x^2 - 54*x - 17, -33*x^3 - 4*x^2 - 47*x + 57
sage: f2, g2
-2049*x + 3196, -1576*x + 6335
sage: f1, g1
14412817, 42616001
sage: F1, G1
5126443, 15157932
sage: F2, G2
2495*x - 399, 3844*x - 2025
sage: F4, G4
-22*x^3 + 39*x^2 - 23*x - 14, -x^3 - 45*x + 5
sage: F8, G8
-x^7 - x^5 + 3*x^4 + 3*x^3 - 3*x^2 + 4,
2*x^7 - x^6 - x^5 - x^4 - 3*x^3 + x^2 + x - 4
# Performances

<table>
<thead>
<tr>
<th>Method</th>
<th>Time complexity(^1)</th>
<th>Space complexity(^1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resultant [HHGP(^+)03]</td>
<td>$\tilde{O}(d(d^2 + B))$</td>
<td>$O(d^2 B)$</td>
</tr>
<tr>
<td>HNF [SS11]</td>
<td>$\tilde{O}(d^3 B)$</td>
<td>$O(d^2 B)$</td>
</tr>
<tr>
<td>This work (Fast)</td>
<td>$O((dB)^{\log_2 3} \log d)$ [Kara]</td>
<td>$O(d(B + \log d) \log d)$</td>
</tr>
<tr>
<td></td>
<td>$\tilde{O}(dB)$ [SchöStr]</td>
<td></td>
</tr>
<tr>
<td>This work (Compact)</td>
<td>$O((dB)^{\log_2 3} \log^2 d)$ [Kara]</td>
<td>$O(d(B + \log d))$</td>
</tr>
<tr>
<td></td>
<td>$\tilde{O}(dB)$ [SchöStr]</td>
<td></td>
</tr>
</tbody>
</table>

We gain in practice:

- a factor 100 in memory (3 MB → 30 kB)
- a factor 100 in time (2 sec. → 20 msec.)

\(^1\) $B = \log_2 \| (f, g) \|$
Problem 2 - Generalized Four Square Theorem

**Problem:** Given $A \in \mathcal{R}^{n \times n}$, compute $B_1, \ldots, B_k \in \mathcal{R}^{n \times n}$ such that

$$AA^* + \sum_i BB^* = C \cdot I_n$$

**Algorithmic solutions:**

1. $\mathcal{R} = \mathbb{R}$, $k = 1$: Cholesky [Pei10]
2. $\mathcal{R} = \mathbb{R}[x]/(\varphi)$, $k = 1$: Babylonian method [DN12]
3. $\mathcal{R} = \mathbb{Z}$, $k = O(1)$: ia.cr/2019/320
4. $\mathcal{R} = \mathbb{Z}[x]/(x^d + 1)$, $k = O(\log d)$: This talk + ia.cr/2019/320
**Simplified problem:** Given \( a \in \mathbb{Z}[x]/(x^d + 1) \), compute polynomials \( b_1, \ldots, b_{\log_2(d)} \in \mathbb{Z}[x]/(x^d + 1) \) such that for some constant \( C \):

\[
a\bar{a} + \sum_i b_i\bar{b}_i = C,
\]

where \( \bar{\cdot} \) denotes the Hermitian adjoint (in our case, \( \bar{a}(x) = a(x^{-1}) \)).

**Attempt 1:** Galois conjugation and Hermitian adjoint compose nicely:

\[
\text{Tr}_{\mathbb{Q}_d/\mathbb{Q}_{d/2}}(a\bar{a}) = a\bar{a} + (a\bar{a})^\times = a\bar{a} + a^\times\bar{a}^\times \in \mathbb{Z}_{d/2}
\]

😊 We have projected the problem over \( \mathbb{Z}_{d/2} \).

😢 Unfortunately repeating this trick doesn’t scale well.
Attempt 2: Let \( a^*a = g \); \( g \) is self-adjoint so we can write \( g = g_{\text{low}} + \overline{g_{\text{low}}} \). Let \( b(x) = 1 - x \cdot g_{\text{low}}(x^2) \), then:

\[
g + b\overline{b} = g_e(x^2) + x \cdot g_{\text{low}}(x^2) + \overline{x \cdot g_{\text{low}}(x^2)} \\
+ (1 - x \cdot g_{\text{low}}(x^2)) \cdot (1 - x \cdot g_{\text{low}}(x^2)) \\
= (1 + g_e + g_{\text{low}} \cdot \overline{g_{\text{low}}})(x^2)
\]
**Attempt 2:** Let $\tilde{a}a = g$; $g$ is self-adjoint so we can write $g = g_{\text{low}} + \overline{g_{\text{low}}}$. Let $b(x) = 1 - x \cdot g_{o,\text{low}}(x^2)$, then:

$$g + b\overline{b} = g_e(x^2) + x \cdot g_{o,\text{low}}(x^2) + x \cdot \overline{g_{o,\text{low}}(x^2)}$$

$$+ (1 - x \cdot g_{o,\text{low}}(x^2)) \cdot (1 - x \cdot \overline{g_{o,\text{low}}(x^2)})$$

$$= (1 + g_e + g_{o,\text{low}} \cdot \overline{g_{o,\text{low}}})(x^2)$$

- We have projected the problem over $\mathbb{Z}_{d/2}$.
- This trick scales well with repetition.
- It incurs a growth on the coefficients’ sizes...
**A Scalable Solution**

**Attempt 2:** Let $a\bar{a} = g$; $g$ is self-adjoint so we can write $g = g_{\text{low}} + \overline{g_{\text{low}}}$. Let $b(x) = 1 - x \cdot g_{\text{low},\text{low}}(x^2)$, then:

\[
g + b\overline{b} = g_e(x^2) + x \cdot g_{\text{low},\text{low}}(x^2) + \overline{x \cdot g_{\text{low},\text{low}}(x^2)} + (1 - x \cdot g_{\text{low},\text{low}}(x^2)) \cdot (1 - x \cdot g_{\text{low},\text{low}}(x^2))
\]

\[
= (1 + g_e + g_{\text{low},\text{low}} \cdot \overline{g_{\text{low},\text{low}}})(x^2)
\]

😊 We have projected the problem over $\mathbb{Z}_{d/2}$.

😊 This trick scales well with repetition.

😊 It incurs a growth on the coefficients’ sizes...

😊 ... but composes nicely with gadget decomposition:

⇒ We write $g = g_0 + 2 \cdot g_1 + \cdots + 2^k g_k$,

⇒ Then we apply this trick on each $g_i$.

This effectively mitigates the size growth.
Attempt 2: Let \( a \bar{a} = g \); \( g \) is self-adjoint so we can write \( g = g_{\text{low}} + \overline{g_{\text{low}}} \). Let \( b(x) = 1 - x \cdot g_{o,\text{low}}(x^2) \), then:

\[
g + b\bar{b} = g_e(x^2) + x \cdot g_{o,\text{low}}(x^2) + x \cdot \overline{g_{o,\text{low}}(x^2)} \\
+ (1 - x \cdot g_{o,\text{low}}(x^2)) \cdot (1 - x \cdot \overline{g_{o,\text{low}}(x^2)}) \\
= (1 + g_e + g_{o,\text{low}} \cdot \overline{g_{o,\text{low}}})(x^2)
\]

😊 We have projected the problem over \( \mathbb{Z}_{d/2} \).
😊 This trick scales well with repetition.
😊 It incurs a growth on the coefficients’ sizes...
😊 ... but composes nicely with gadget decomposition:

⇒ We write \( g = g_0 + 2 \cdot g_1 + \cdots + 2^k g_k \),

⇒ Then we apply this trick on each \( g_i \).

This effectively mitigates the size growth.

Consequence: We can compute \( b_1, \ldots, b_k \) in \( \mathbb{Z}_d \) such that

\[
a \bar{a} + \sum_i b_i \overline{b_i} = C,
\]

with \( k = \tilde{O}(\log \|g\|_\infty + \log d) \).
Problem 3 - Efficient Lattice Decoding

Problem: Given $B \in \mathbb{Z}^{n \times n}_d$ and $c \in \text{Span}_{Q_d}(B)$, compute $v \in \Lambda(B)$ such that

$$\|v - c\|$$ is small.

Equivalent: Given $B \in \mathbb{Z}^{n \times n}_d$ and $t \in Q^n_d$, compute $z \in \mathbb{Z}^n_d$ such that

$$\|(z - t) \cdot B\|$$ is small.

Algorithmic solutions:

- High quality, $O((nd)^2)$ operations
  (Randomized) nearest plane
  [Bab85, GPV08]

- Lower quality, $O(n^2 d \log d)$ operations
  (Randomized) round-off
  [Bab85, Pei10]

- High quality, $O(n^2 d \log d)$ operations
  Fast Fourier orthogonalization
  ia.cr/2015/1014
How to Find a Close Vector

**Round-Off Algorithm:**

1. \( t \leftarrow c \cdot B^{-1} \)
2. \( z \leftarrow \lfloor t \rfloor \)
3. Output \( v \leftarrow z \cdot B \)

**Nearest Plane Algorithm:**

1. \( t \leftarrow c \cdot B^{-1} \)
2. For \( j = n \) down to 1:
   1. \( \hat{t}_j \leftarrow t_j + \sum_{i>j} (t_i - z_i) \cdot L_{i,j} \)
   2. \( z_j \leftarrow \lfloor \hat{t}_j \rfloor \)
3. Output \( v \leftarrow z \cdot B \)

Output:

---

1 Requires precomputing the Gram-Schmidt orthogonalisation (GSO) of \( B: B = L \cdot \tilde{B} \).
How to Find a Close Vector

Round-Off Algorithm:
1. \( t \leftarrow c \cdot B^{-1} \)
2. \( z \leftarrow \lfloor t \rfloor \)
3. Output \( v \leftarrow z \cdot B \)

Output:

Nearest Plane Algorithm:\(^1\)
1. \( t \leftarrow c \cdot B^{-1} \)
2. For \( j = n \) down to 1:
   1. \( \hat{t}_j \leftarrow t_j + \sum_{i>j} (t_i - z_i) \cdot L_{i,j} \)
   2. \( z_j \leftarrow \lfloor \hat{t}_j \rfloor \)
3. Output \( v \leftarrow z \cdot B \)

Output:

\(^1\)Requires precomputing the Gram-Schmidt orthogonalisation (GSO) of \( B \): \( B = L \cdot \tilde{B} \).
Tricks and Tips (1/2)

Consider the simplified case where we want this to be small:

\[(z - t) \cdot b\]

Using the ring isomorphism \(Q_d \cong (Q_{d/2})^2\), this is equivalent to:

\[
\begin{bmatrix}
  z_e - t_e & z_0 - t_0 \\
\end{bmatrix}
\cdot
\begin{bmatrix}
  b_e & b_o \\
  xb_o & b_e \\
\end{bmatrix}
\]

Why this is nice:

- We can orthogonalize the second row of \(B\) w.r.t. to the first one:

\[
\tilde{b}_2 \leftarrow b_2 - \frac{\langle b_2, b_1 \rangle}{b_2, b_1} \cdot b_1
\]

- We can apply this “break and orthogonalize” trick recursively.
- This structured decomposition then allows a faster nearest plane algorithm.
Additional tricks:

- **Equivalent decomposition:**

\[
\begin{align*}
(B = L \cdot \tilde{B}) & \iff (B \cdot B^* = L \cdot \tilde{B} \tilde{B}^* \cdot L^*) \\
\text{GSO} & \iff \text{LDL decomposition}
\end{align*}
\]

The LDL decomposition is more amenable to a recursive application of our trick; this yields a complexity \(O(d \log^2 d)\).

- **Working only in the FFT domain:** Discarding useless conversions further reduces the total complexity to \(O(d \log d)\).
Summary (non-exhaustive)

Speed-ups in the presence of a ring:

- Most of efficient lattice-based cryptography

Speed-ups in the presence of tower of rings (this talk):

- Using ring isomorphisms: ia.cr/2015/1014
- Using the field norm: ia.cr/2019/015
- Using trace-like properties: ia.cr/2019/230

Exploiting automorphisms:

- Homomorphic encryption
- Zero-Knowledge proofs [dPLS18]
If you cannot trivially exploit the presence of a ring...

... use its particular structure!
L Babai.
On lova’sz’ lattice reduction and the nearest lattice point problem.

Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig.
Improved security for a ring-based fully homomorphic encryption scheme.

Peter Campbell and Michael Groves.
Practical post-quantum hierarchical identity-based encryption.
16th IMA International Conference on Cryptography and Coding, 2017.

Léo Ducas, Vadim Lyubashevsky, and Thomas Prest.
Efficient identity-based encryption over NTRU lattices.


Chris Peikert. An efficient and parallel Gaussian sampler for lattices.

Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang.
Falcon.
John M. Schanck, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe.

Damien Stehlé and Ron Steinfeld.

Xiaoyun Wang and Kazue Sako, editors.

Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, and William Whyte.