Linear Algebra with Errors: On the Complexity of the Learning with Errors Problem

Martin R. Albrecht

joint work with C. Cid, J-C. Faugère, R. Fitzpatrick, and L. Perret

SIAM AG’13
Contents

Introduction

Warm-Up: Deciding Consistency in Noise Free Systems

Solving Decision-LWE

Solving Decision-LWE with Small Secrets
Learning with Errors

Definition (Learning with Errors)

Let $n \geq 1$, $m \gg n$, $q$ odd, $\chi$ be a probability distribution on $\mathbb{Z}_q$ and $s$ be a secret vector in $\mathbb{Z}_q^n$.

Let $e \leftarrow \chi^m$, $A \leftarrow U(\mathbb{Z}_q^{m \times n})$. We denote by $L_{s,\chi}^{(n)}$ the distribution on $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ produced as $(A, A \cdot s + e)$.

Decision-LWE is the problem of deciding whether $A, c \leftarrow U(\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m)$ or $A, c \leftarrow L_{s,\chi}^{(n)}$.

In other words: Is $c$ sampled uniformly randomly or is it $A \cdot s + e$ where typically $e$ is “small”.

Typically, $\chi$ is a discrete Gaussian distribution with small standard deviation.
Learning with Errors

Definition (Learning with Errors)

Let $n \geq 1$, $m \gg n$, $q$ odd, $\chi$ be a probability distribution on $\mathbb{Z}_q$ and $\mathbf{s}$ be a secret vector in $\mathbb{Z}_q^n$.

Let $\mathbf{e} \leftarrow \chi^m$, $\mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{m \times n})$. We denote by $L_{s,\chi}^{(n)}$ the distribution on $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ produced as $(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$.

Decision-LWE is the problem of deciding whether $\mathbf{A}, \mathbf{c} \leftarrow \mathcal{U}(\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m)$ or $\mathbf{A}, \mathbf{c} \leftarrow L_{s,\chi}^{(n)}$.

In other words: Is $\mathbf{c}$ sampled uniformly randomly or is it $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ where typically $\mathbf{e}$ is “small”.

Typically, $\chi$ is a discrete Gaussian distribution with small standard deviation.
Learning with Errors

Definition (Learning with Errors)

Let $n \geq 1$, $m \gg n$, $q$ odd, $\chi$ be a probability distribution on $\mathbb{Z}_q$ and $s$ be a secret vector in $\mathbb{Z}_q^n$.

Let $e \leftarrow \chi^m$, $A \leftarrow \mathcal{U}(\mathbb{Z}_q^{m \times n})$. We denote by $L_s^{(n)}$ the distribution on $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ produced as $(A, A \cdot s + e)$.

Decision-LWE is the problem of deciding whether $A, c \leftarrow \mathcal{U}(\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m)$ or $A, c \leftarrow L_s^{(n)}$.

In other words: Is $c$ sampled uniformly randomly or is it $A \cdot s + e$ where typically $e$ is “small”.

Typically, $\chi$ is a discrete Gaussian distribution with small standard deviation.
Definition (Learning with Errors)

Let \( n \geq 1, m \gg n, q \) odd, \( \chi \) be a probability distribution on \( \mathbb{Z}_q \) and \( \mathbf{s} \) be a secret vector in \( \mathbb{Z}^n_q \).

Let \( \mathbf{e} \leftarrow \chi^m, \mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{m \times n}). \) We denote by \( L_{\mathbf{s}, \chi}^{(n)} \) the distribution on \( \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m \) produced as \( (\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e}) \).

Decision-LWE is the problem of deciding whether \( \mathbf{A}, \mathbf{c} \leftarrow \mathcal{U}(\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m) \) or \( \mathbf{A}, \mathbf{c} \leftarrow L_{\mathbf{s}, \chi}^{(n)} \).

In other words: Is \( \mathbf{c} \) sampled uniformly randomly or is it \( \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \) where typically \( \mathbf{e} \) is “small”.

Typically, \( \chi \) is a discrete Gaussian distribution with small standard deviation.
Applications

- Public-Key Encryption, Digital Signature Schemes
- Identity-based Encryption: encrypting to an identity (e-mail address ...) instead of key
- Fully-homomorphic encryption: computing with encrypted data
- ...
Asymptotic Security

Reduction of worst-case hard lattice problems such as Closest Vector Problem (CVP) to average-case LWE.

But to build cryptosystems we need to understand the hardness of concrete instances: Given $m, n, q$ and $\chi$ how many operations does it take to solve Decision-LWE?
Asymptotic Security

Reduction of worst-case hard lattice problems such as Closest Vector Problem (CVP) to average-case LWE.

But to build cryptosystems we need to understand the hardness of concrete instances: Given $m$, $n$, $q$ and $\chi$ how many operations does it take to solve Decision-LWE?
Solving Strategies

Given $A, c$ with $c = A \cdot s + e$ solve the problem in the primal lattice or the dual lattice.

- Solve the Bounded-Distance Decoding (BDD) problem in the primal lattice: Find $s'$ such that
  \[ \| y - c \| \text{ is minimised, for } y = A \cdot s'. \]

- Solve the Short-Integer-Solutions (SIS) problem in the scaled dual lattice. Find a short $y$ such that
  \[ y \cdot A = 0 \text{ and check if } \langle y, c \rangle = y \cdot (A \cdot s + e) = \langle y, e \rangle \text{ is short.} \]

In this talk

solving SIS using combinatorial techniques and no bound on $m$. 
Solving Strategies

Given $A, c$ with $c = A \cdot s + e$ solve the problem in the primal lattice or the dual lattice.

- Solve the Bounded-Distance Decoding (BDD) problem in the primal lattice: Find $s'$ such that

  $$\|y - c\|$$ is minimised, for $y = A \cdot s'$.

- Solve the Short-Integer-Solutions (SIS) problem in the scaled dual lattice. Find a short $y$ such that

  $$y \cdot A = 0$$ and check if $\langle y, c \rangle = y \cdot (A \cdot s + e) = \langle y, e \rangle$ is short.

In this talk

- solving SIS using combinatorial techniques and
- no bound on $m$. 
Solving Strategies

Given $\mathbf{A}, \mathbf{c}$ with $\mathbf{c} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ solve the problem in the primal lattice or the dual lattice.

- Solve the Bounded-Distance Decoding (BDD) problem in the primal lattice: Find $\mathbf{s}'$ such that
  \[ \|\mathbf{y} - \mathbf{c}\| \text{ is minimised, for } \mathbf{y} = \mathbf{A} \cdot \mathbf{s}'. \]

- Solve the Short-Integer-Solutions (SIS) problem in the scaled dual lattice. Find a short $\mathbf{y}$ such that
  \[ \mathbf{y} \cdot \mathbf{A} = 0 \text{ and check if } \langle \mathbf{y}, \mathbf{c} \rangle = \mathbf{y} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}) = \langle \mathbf{y}, \mathbf{e} \rangle \text{ is short.} \]

In this talk

- solving SIS using combinatorial techniques and no bound on $m$. 

Contents

Introduction

Warm-Up: Deciding Consistency in Noise Free Systems

Solving Decision-LWE

Solving Decision-LWE with Small Secrets
Gaussian elimination

Asume $e = 0$, we hence want to decide whether there is a solution $s$ such that $c = A \cdot s$. We may apply Gaussian elimination to the matrix:

$$[A \mid c] = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & c_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & c_m \end{pmatrix}$$

to recover

$$[\tilde{A} \mid \tilde{c}] = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & c_1 \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} & \tilde{c}_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \tilde{a}_{mn} & \tilde{c}_m \end{pmatrix}$$

If and only if $\tilde{c}_{n+1}, \ldots, \tilde{c}_m$ are all zero, the system is consistent.
Contents

Introduction

Warm-Up: Deciding Consistency in Noise Free Systems

Solving Decision-LWE

Solving Decision-LWE with Small Secrets
The BKW algorithm was first proposed for the Learning Parity with Noise (LPN) problem which can be viewed as a special case of LWE.

**BKW Algorithm II**

We revisit Gaussian elimination:

\[
\begin{bmatrix}
    a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & c_1 \\
    a_{21} & a_{22} & a_{23} & \cdots & a_{2n} & c_2 \\
    \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
    a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} & c_m
\end{bmatrix}
\]

\[=\]

\[
\begin{bmatrix}
    a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & \langle a_1, s \rangle + e_1 \\
    a_{21} & a_{22} & a_{23} & \cdots & a_{2n} & \langle a_2, s \rangle + e_2 \\
    \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
    a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} & \langle a_m, s \rangle + e_m
\end{bmatrix}
\]

\[\Rightarrow\]

\[
\begin{bmatrix}
    a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & \langle a_1, s \rangle + e_1 \\
    0 & \tilde{a}_{22} & \tilde{a}_{23} & \cdots & \tilde{a}_{2n} & \langle \tilde{a}_2, s \rangle + e_2 - \frac{a_{21}}{a_{11}} e_1 \\
    \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
    0 & \tilde{a}_{m2} & \tilde{a}_{m3} & \cdots & \tilde{a}_{mn} & \langle \tilde{a}_m, s \rangle + e_m - \frac{a_{m1}}{a_{11}} e_1
\end{bmatrix}
\]
BKW Algorithm III

- $\frac{a_{i1}}{a_{11}}$ is essentially a random element in $\mathbb{Z}_q$, hence $\tilde{c}_i \leftarrow \mathcal{U}(\mathbb{Z}_q)$.
- Even if $\frac{a_{i1}}{a_{11}}$ is 1 the variance of the noise doubles at every level because of the addition.

Setting

**Problem:** additions and multiplications $\Rightarrow$ noise of $\tilde{c}$ values increases rapidly

**Strategy:** exploit that we have many rows: $m \gg n$. 
We considering $a \approx \log n$ ‘blocks’ of $b$ elements each.

$$
\begin{bmatrix}
  a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & c_0 \\
  a_{21} & a_{22} & a_{23} & \cdots & a_{2n} & c_1 \\
  \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\
  a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} & c_m \\
\end{bmatrix}
$$
For each block we build a table of all $q^b$ possible values.

$$T = \begin{bmatrix} 0 & 0 & a_{13} & \cdots & a_{1n} & c_0 \\ 0 & 1 & a_{23} & \cdots & a_{2n} & c_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ q & q & a_{q^23} & \cdots & a_{q^2n} & c_{q^2} \end{bmatrix}$$
BKW Algorithm VI

We use these tables to eliminate $b$ entries in other rows.

\[
\begin{pmatrix}
  a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & c_0 \\
  a_{21} & a_{22} & a_{23} & \cdots & a_{2n} & c_1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} & c_m
\end{pmatrix}
\begin{pmatrix}
  0 & 0 & a_{13} & \cdots & a_{1n} & c_0 \\
  0 & 1 & a_{23} & \cdots & a_{2n} & c_1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  q & q & a_{q^2 3} & \cdots & a_{q^2 n} & c_{q^2}
\end{pmatrix}
\]

\[
\Rightarrow
\begin{pmatrix}
  a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & c_0 \\
  0 & 0 & a_{23} & \cdots & a_{2n} & \tilde{c}_1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  0 & 0 & \tilde{a}_{m3} & \cdots & \tilde{a}_{mn} & \tilde{c}_m
\end{pmatrix}
\]
BKW Algorithm VII

This gives a time complexity of

\[ \approx (a^2 n) \cdot \frac{q^b}{2} \]

and a memory requirement of

\[ \approx \frac{q^b}{2} \cdot a \cdot (n + 1). \]

A detailed analysis of the algorithm for LWE is available as:

Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick and Ludovic Perret
On the Complexity of the BKW Algorithm on LWE
to appear in \textit{Designs, Codes and Cryptography}. 
The Setting

Assume $\mathbf{s} \leftarrow \mathcal{U}(\mathbb{Z}_2^n)$, i.e. all entries in $\mathbf{s}$ are very small.

This is a common setting in cryptography for performance reasons and because this allows to realise some advanced schemes. In particular, a technique called ‘modulus switching’ can be used to improve the performance of homomorphic encryption schemes.

Modulus Reduction I

Given a sample \((a, c)\) where \(c = \langle a, s \rangle + e\) and some \(p < q\) we may consider

\[
\left( \left\lfloor \frac{p}{q} \cdot a \right\rfloor, \left\lfloor \frac{p}{q} \cdot c \right\rfloor \right)
\]

with

\[
\begin{align*}
\left[ \frac{p}{q} \cdot c \right] &= \left[ \left\langle \frac{p}{q} \cdot a, s \right\rangle + \frac{p}{q} \cdot e \right] \\
&= \left[ \left\langle \left\lfloor \frac{p}{q} \cdot a \right\rfloor, s \right\rangle + \left\langle \frac{p}{q} \cdot a - \left\lfloor \frac{p}{q} \cdot a \right\rfloor, s \right\rangle + \frac{p}{q} \cdot e \right] \\
&= \left\langle \left\lfloor \frac{p}{q} \cdot a \right\rfloor, s \right\rangle + \left\langle \frac{p}{q} \cdot a - \left\lfloor \frac{p}{q} \cdot a \right\rfloor, s \right\rangle + \frac{p}{q} \cdot e \pm [0, 0.5] \\
&= \left\langle \left\lfloor \frac{p}{q} \cdot a \right\rfloor, s \right\rangle + e''.
\end{align*}
\]
Modulus Reduction II

Example

\[ p, q = 10, 20 \]
\[ a = (8, -2, 0, 4, 2, -7), \]
\[ s = (0, 1, 0, 0, 1, 1), \]
\[ \langle a, s \rangle = -7, \]
\[ c = -6 \]
\[ a' = \left\lfloor \frac{p}{q} \cdot a \right\rfloor = (4, -1, 0, 2, 1, -4) \]
\[ \langle a', s \rangle = -4, \]
\[ \left\lfloor \frac{p}{q} \cdot c \right\rfloor = -4. \]
Modulus Reduction III

Typically, we would choose

$$p \approx q \cdot \sqrt{n \cdot \text{Var}(U([-0.5, 0.5]))} \cdot \frac{\sigma_s^2}{\sigma} = q \cdot \sqrt{n/12\sigma_s/\sigma}$$

where $\sigma_s$ is the standard deviation of elements in $s$.

If $s$ is small then $e''$ is small and we may compute with the smaller ‘precision’ $p$ at the cost of a slight increase of the noise rate.

The complexity hence drops to

$$\approx (a^2 n) \cdot \frac{p^b}{2}$$

with $a$ usually is unchanged.
Lazy Modulus Switching I

For simplicity assume $p = 2^\kappa$ and consider the LWE matrix

$$[A \mid c] = \begin{pmatrix}
    a_{1,1} & a_{1,2} & \ldots & a_{1,n} & c_1 \\
    a_{2,1} & a_{2,2} & \ldots & a_{2,n} & c_2 \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    a_{m,1} & a_{m,2} & \ldots & a_{m,n} & c_m
\end{pmatrix}$$

as

$$[A \mid c] = \begin{pmatrix}
    a^h_{1,1} & a^l_{1,1} & a^h_{1,2} & a^l_{1,2} & \ldots & a^h_{1,n} & a^l_{1,n} & c_1 \\
    a^h_{2,1} & a^l_{2,1} & a^h_{2,2} & a^l_{2,2} & \ldots & a^h_{2,n} & a^l_{2,n} & c_2 \\
    \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
    a^h_{m,1} & a^l_{m,1} & a^h_{m,2} & a^l_{m,2} & \ldots & a^h_{m,n} & a^l_{m,n} & c_m
\end{pmatrix}$$

where $a^h_{i,j}$ and $a^l_{i,j}$ denote high and low order bits:

- $a^h_{i,j}$ corresponds to $\lfloor p/q \cdot a_{i,j} \rfloor$ and
- $a^l_{i,j}$ corresponds to $\lfloor p/q \cdot a_{i,j} \rfloor - p/q \cdot a_{i,j}$, the rounding error.
Lazy Modulus Switching II

In order to clear the most significant bits in every component of the $a_i$, we run the BKW algorithm on the matrix $[A \ | \ c]$ but only consider

$$[A, c]^h := \begin{pmatrix} a_{1,1}^h & a_{1,2}^h & \cdots & a_{1,n}^h & c_1 \\ a_{2,1}^h & a_{2,2}^h & \cdots & a_{2,n}^h & c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m,1}^h & a_{m,2}^h & \cdots & a_{m,n}^h & c_m \end{pmatrix}.$$  

when searching for collisions.

We only manage elimination tables for the most significant $\kappa$ bits. All arithmetic is performed in $\mathbb{Z}_q$ but collisions are searched for in $\mathbb{Z}_p$. 
Lazy Modulus Switching III

- We do not apply modulus reduction in one shot, but only when needed.
- As a consequence rounding errors accumulate not as fast: they only start to accumulate when we branch on a component.

We may reduce $p$ by a factor of $\sqrt{a/2}$.

This may translate to huge gains the complexity of BKW is $\approx p^b$ where typically $b \approx n / \log n$. 
Stunting Growth I

Figure: Children, parents and strangers.
Stunting Growth II

Assume $b = 1$ and $a \geq 3$, for the outputs $(\tilde{a}_i, \tilde{c}_i)$ where the first three components are reduced have:

$$
\tilde{a}_i = a_i \text{ from } L^{(n)}_{s, \chi} \\
+ \tilde{a}_0 \text{ with } \tilde{a}_0 \text{ from } T^0 \\
+ \tilde{a}_1 \text{ with } \tilde{a}_1 \text{ from } T^1 \\
+ \tilde{a}_2 \text{ with } \tilde{a}_2 \text{ from } T^2
$$

Considering component $\tilde{a}_{i, (0)}$ we have that

- $a_{i, (0)}$ is uniform in $\mathbb{Z}_q$,
- $\tilde{a}_{0, (0)}$ reduces this to something of size $r = \log_2 q - \log_2 p$
- $\tilde{a}_{1, (0)}$ has size $\log_2 q - \log_2 p$
- $\tilde{a}_{2, (0)}$ has size $\approx \log_2 q - \log_2 p + 1$, and depends on entries on $T^1$. 
We sample many candidates for $\tilde{a}_2$ to find one where $\tilde{a}_{2,(0)}$ is particularly small.

This is easier than for $\tilde{a}_3$ but influences $\tilde{a}_3$. 
Assumption

Let the vectors $\mathbf{x}_i \in \mathbb{Z}_q^\tau$ be sampled from some distribution $\mathcal{D}$ such that

$$\sigma^2 = \text{Var}(\mathbf{x}_i,j)$$

where $\mathcal{D}$ is any distribution on (sub-)vectors observable in our algorithm. Let $\mathbf{y} = \min_{\text{abs}}(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1})$ where $\min_{\text{abs}}$ picks that vector $\mathbf{x}_{\min}$ with $\sum_{j=0}^{b \cdot \ell - 1} |\mathbf{x}_{\min,j}|$ minimal. The standard deviation $\sigma_n = \sqrt{\text{Var}(\mathbf{y}(j))}$ of components in $\mathbf{y}$ satisfies

$$\frac{\sigma}{\sigma_n} \geq c_\tau \sqrt{n} + (1 - c_\tau)$$

with

$$c_\tau = 0.20151418166952917 \sqrt{\tau} + 0.32362108131969386 \approx \frac{1}{5} \sqrt{\tau} + \frac{1}{3}.$$
## Results

<table>
<thead>
<tr>
<th>n</th>
<th>log $\mathbb{Z}_2$</th>
<th>log mem</th>
<th>log $\mathbb{Z}_2$</th>
<th>log mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>40.0</td>
<td>26.2</td>
<td>39.4</td>
<td>25.5</td>
</tr>
<tr>
<td>64</td>
<td>55.9</td>
<td>48.8</td>
<td>52.5</td>
<td>46.0</td>
</tr>
<tr>
<td>128</td>
<td>97.6</td>
<td>90.0</td>
<td>89.6</td>
<td>81.2</td>
</tr>
<tr>
<td>256</td>
<td>182.1</td>
<td>174.2</td>
<td>164.0</td>
<td>156.7</td>
</tr>
<tr>
<td>512</td>
<td>361.0</td>
<td>352.8</td>
<td>305.6</td>
<td>297.9</td>
</tr>
<tr>
<td>1024</td>
<td>705.5</td>
<td>697.0</td>
<td>580.2</td>
<td>572.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>log $\mathbb{Z}_2$</th>
<th>log mem</th>
<th>log $\mathbb{Z}_2$</th>
<th>log mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>40.0</td>
<td>26.1</td>
<td>40.0</td>
<td>26.1</td>
</tr>
<tr>
<td>64</td>
<td>49.2</td>
<td>42.1</td>
<td>47.6</td>
<td>32.0</td>
</tr>
<tr>
<td>128</td>
<td>78.2</td>
<td>70.8</td>
<td>74.2</td>
<td>46.3</td>
</tr>
<tr>
<td>256</td>
<td>142.7</td>
<td>134.9</td>
<td>132.5</td>
<td>67.1</td>
</tr>
<tr>
<td>512</td>
<td>251.2</td>
<td>243.1</td>
<td>241.8</td>
<td>180.0</td>
</tr>
<tr>
<td>1024</td>
<td>494.8</td>
<td>486.5</td>
<td>485.0</td>
<td>407.5</td>
</tr>
</tbody>
</table>

Table: Cost for solving Decision-LWE with advantage $\approx 1$ for BKW and BKZ variants where $q$ and $\sigma$ are chosen as in Regev’s scheme and $s \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$ “log $\mathbb{Z}_2$” gives the number of “bit operations” and “log mem” the memory requirement of $\mathbb{Z}_q$ elements. All logarithms are base 2.
Questions?