Solving LWE with BKW

Martin R. Albrecht [1] Jean-Charles Faugére [2,3]
Robert Fitzpatrick[1,4] Ludovic Perret[2,3]

ISG, Royal Holloway, University of London

INRIA

CNRS

IIS, Academia Sinica, Taipei, Taiwan

PKC 2014, Buenos Aires, Argentina, 28th March 2014

Contents

Learning with Errors

Definition (Learning with Errors)

- Let $n \geq 1$, $m > n$, q modulus, χ 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 $L_{\mathbf{s},\chi}^{(n)}$ the distribution on $\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$ produced as $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$.
- Decision-LWE is the problem of deciding if $(\mathbf{A}, \mathbf{c}) \leftarrow_\$ L_{\mathbf{s},\chi}^{(n)}$ (i.e. $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e}$ where $\mathbf{e}$ is "small") or $(\mathbf{A}, \mathbf{c}) \leftarrow_\$ \mathcal{U}(\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m)$ (i.e. $\mathbf{c}$ is sampled uniformly random).

Learning with Errors

Definition (Learning with Errors)

- Let $n \geq 1$, $m > n$, q modulus, χ 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 $L_{\mathbf{s},\chi}^{(n)}$ the distribution on $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ produced as $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$.
- Decision-LWE is the problem of deciding if $(\mathbf{A}, \mathbf{c}) \leftarrow_\$ L_{\mathbf{s},\chi}^{(n)}$ (i.e. $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e}$ where $\mathbf{e}$ is "small") or $(\mathbf{A}, \mathbf{c}) \leftarrow_\$ \mathcal{U}(\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m)$ (i.e. $\mathbf{c}$ is sampled uniformly random).

Learning with Errors

Definition (Learning with Errors)

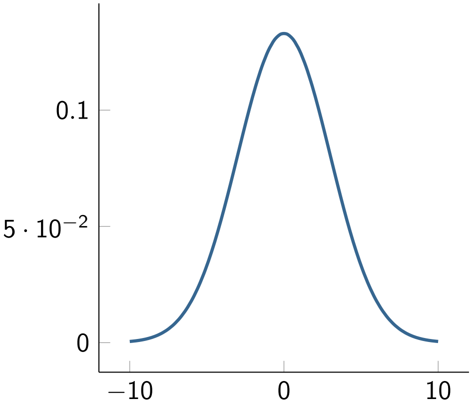
- Let $n \geq 1$, $m > n$, q modulus, χ be a probability distribution on $\mathbb{Z}_q$ and $\mathbf{s}$ be a secret vector in $\mathbb{Z}_q^n$.
- Let $\mathbf{e} \leftarrow_\$ \chi^m$, $\mathbf{A} \leftarrow_\$ \mathcal{U}(\mathbb{Z}_q^{m \times n})$. We denote by $L_{\mathbf{s},\chi}^{(n)}$ the distribution on $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ produced as $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$.
- Decision-LWE is the problem of deciding if $(\mathbf{A}, \mathbf{c}) \leftarrow_\$ L_{\mathbf{s},\chi}^{(n)}$ (i.e. $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e}$ where $\mathbf{e}$ is "small") or $(\mathbf{A}, \mathbf{c}) \leftarrow_\$ \mathcal{U}(\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m)$ (i.e. $\mathbf{c}$ is sampled uniformly random).

Small??

- We represent elements in $\mathbb{Z}_q$ as integers in $[-\lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor]$.
- By "size" we mean $|x|$ for $x \in \mathbb{Z}_q$ in this representation.
- Typically, χ is a discrete Gaussian distribution over $\mathbb{Z}$ considered modulo q with small standard deviation.

Small??

- We represent elements in $\mathbb{Z}_q$ as integers in $[-\lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor]$.
- By "size" we mean $|x|$ for $x \in \mathbb{Z}_q$ in this representation.
- Typically, χ is a discrete Gaussian distribution over $\mathbb{Z}$ considered modulo q with small standard deviation.



Learning with Errors with Matrices

We can generalise this slightly. Given $(\mathbf{A}, \mathbf{C})$ with $\mathbf{C} \in \mathbb{Z}_q^{m\times\ell}$, $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, $\mathbf{S} \in \mathbb{Z}_q^{n\times\ell}$ and $\mathbf{E} \in \mathbb{Z}_q^{m\times\ell}$ do we have

$$\begin{pmatrix} \leftarrow & \ell & \rightarrow \\ & & \\ & \mathbf{C} & \\ & & \\ & & \end{pmatrix} = \begin{pmatrix} \leftarrow & n & \rightarrow \\ & & \\ & \mathbf{A} & \\ & & \\ & & \end{pmatrix} \times \begin{pmatrix} & \mathbf{S} & \end{pmatrix} + \begin{pmatrix} \leftarrow & \ell & \rightarrow \\ & & \\ & \mathbf{E} & \\ & & \\ & & \end{pmatrix}$$

or $\mathbf{C} \leftarrow_\$ \mathcal{U}(\mathbb{Z}_q^{m\times\ell})$.

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 Shortest Vector Problem (SVP) to average-case LWE.

Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé.
Classical hardness of Learning with Errors.
In *STOC '13*, pages 575–584, New York, 2013. ACM.

For cryptosystems we need the hardness of concrete instances:

Given m, n, q and χ how many operations does it take to solve Decision-LWE?

Asymptotic Security

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

Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé.
Classical hardness of Learning with Errors.
In *STOC '13*, pages 575–584, New York, 2013. ACM.

For cryptosystems we need the hardness of concrete instances:

Given m, n, q and χ how many operations does it take to solve Decision-LWE?

Solving Strategies

Given $\mathbf{A}, \mathbf{c}$ with $\mathbf{c} = \mathbf{A} \times \mathbf{s} + \mathbf{e}$ or $\mathbf{c} \leftarrow_{\$} \mathcal{U}(\mathbb{Z}_q^n)$

- solve the Bounded-Distance Decoding (BDD) problem in the primal lattice: Find $\mathbf{s}'$ such that

$$\|\mathbf{y} - \mathbf{c}\| \text{ is minimised, for } \mathbf{y} = \mathbf{A} \times \mathbf{s}'.$$

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

$$\mathbf{y} \times \mathbf{A} = 0 \text{ and check if } \langle \mathbf{y}, \mathbf{c} \rangle = \mathbf{y} \times (\mathbf{A} \times \mathbf{s} + \mathbf{e}) = \langle \mathbf{y}, \mathbf{e} \rangle \text{ is short.}$$

In this talk

1. we solve SIS
2. we use combinatorial techniques and
3. we put no bound on m.

Solving Strategies

Given $\mathbf{A}, \mathbf{c}$ with $\mathbf{c} = \mathbf{A} \times \mathbf{s} + \mathbf{e}$ or $\mathbf{c} \leftarrow_{\$} \mathcal{U}(\mathbb{Z}_q^n)$

- solve the Bounded-Distance Decoding (BDD) problem in the primal lattice: Find $\mathbf{s}'$ such that

$$\|\mathbf{y} - \mathbf{c}\| \text{ is minimised, for } \mathbf{y} = \mathbf{A} \times \mathbf{s}'.$$

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

$$\mathbf{y} \times \mathbf{A} = 0 \text{ and check if } \langle \mathbf{y}, \mathbf{c} \rangle = \mathbf{y} \times (\mathbf{A} \times \mathbf{s} + \mathbf{e}) = \langle \mathbf{y}, \mathbf{e} \rangle \text{ is short.}$$

In this talk

1. we solve SIS
2. we use combinatorial techniques and
3. we put no bound on m.

Solving Strategies

Given $\mathbf{A}, \mathbf{c}$ with $\mathbf{c} = \mathbf{A} \times \mathbf{s} + \mathbf{e}$ or $\mathbf{c} \leftarrow_{\$} \mathcal{U}(\mathbb{Z}_q^n)$

- solve the Bounded-Distance Decoding (BDD) problem in the primal lattice: Find $\mathbf{s}'$ such that

$$\|\mathbf{y} - \mathbf{c}\| \text{ is minimised, for } \mathbf{y} = \mathbf{A} \times \mathbf{s}'.$$

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

$$\mathbf{y} \times \mathbf{A} = 0 \text{ and check if } \langle \mathbf{y}, \mathbf{c} \rangle = \mathbf{y} \times (\mathbf{A} \times \mathbf{s} + \mathbf{e}) = \langle \mathbf{y}, \mathbf{e} \rangle \text{ is short.}$$

In this talk

1. we solve SIS
2. we use combinatorial techniques and
3. we put no bound on m.

Contents

Gaussian elimination I

Assume there is no error, we hence want to decide whether there is a solution $\mathbf{S}$ such that $\mathbf{C} = \mathbf{A} \times \mathbf{S}$. We may apply Gaussian elimination to the matrix:

$$[\mathbf{A} \mid \mathbf{C}] = \left(\begin{array}{cccc|ccc} \mathbf{a}_{11} & \mathbf{a}_{12} & \dots & \mathbf{a}_{1n} & \mathbf{c}_{11} & \dots & \mathbf{c}_{1\ell} \\ \mathbf{a}_{21} & \mathbf{a}_{22} & \dots & \mathbf{a}_{2n} & \mathbf{c}_{21} & \dots & \mathbf{c}_{2\ell} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_{m1} & \mathbf{a}_{m2} & \dots & \mathbf{a}_{mn} & c_{m1} & \dots & c_{m\ell} \end{array}\right)$$

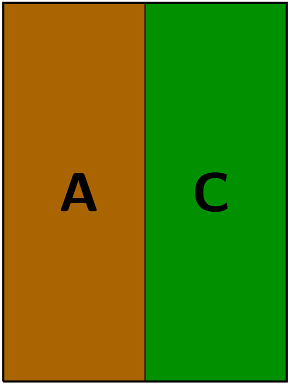
to recover

$$[\tilde{\mathbf{A}} \mid \tilde{\mathbf{C}}] = \left(\begin{array}{cccc|ccc} \mathbf{a}_{11} & \mathbf{a}_{12} & \dots & \mathbf{a}_{1n} & \mathbf{c}_{11} & \dots & \mathbf{c}_{1\ell} \\ 0 & \tilde{\mathbf{a}}_{22} & \dots & \tilde{\mathbf{a}}_{2n} & \tilde{\mathbf{c}}_{21} & \dots & \tilde{\mathbf{c}}_{2\ell} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \tilde{\mathbf{a}}_{rn} & \tilde{\mathbf{c}}_{r1} & \dots & \tilde{\mathbf{c}}_{r\ell} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \tilde{\mathbf{c}}_{m1} & \dots & \tilde{\mathbf{c}}_{m\ell} \end{array}\right)$$

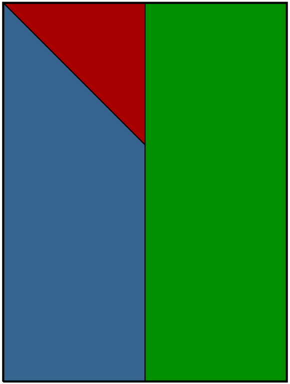
Gaussian elimination II

If and only if $\tilde{\mathbf{c}}_{r+1,1}, \ldots, \tilde{\mathbf{c}}_{m,\ell}$ are all zero, the system is consistent.

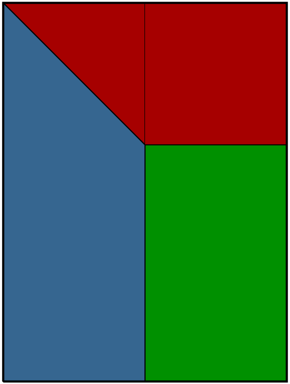
Gaussian elimination III



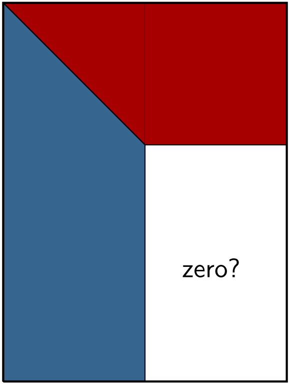
Gaussian elimination IV



Gaussian elimination V



Gaussian elimination VI



Contents

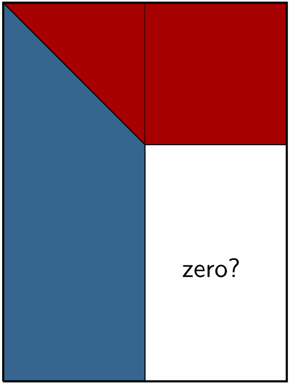
BKW Algorithm I

The BKW algorithm was first proposed for the Learning Parity with Noise (LPN) problem which can be viewed as a special case of LWE over $\mathbb{Z}_2$.

Avrim Blum, Adam Kalai, and Hal Wasserman.
Noise-tolerant learning, the parity problem, and the statistical query model.
J. ACM, 50(4):506–519, 2003.

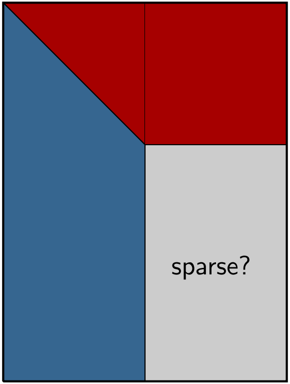
BKW Algorithm II

Goal in noise-free case:



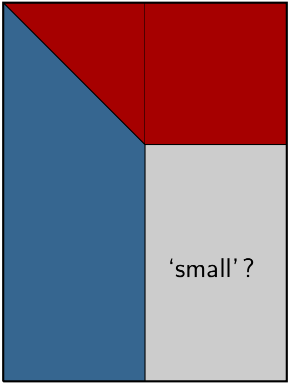
BKW Algorithm III

Goal over $\mathbb{Z}_2$ (LPN):



BKW Algorithm IV

Goal over $\mathbb{Z}_q$ (LWE):



BKW Algorithm V

We revisit Gaussian elimination:

$$\left(\begin{array}{c|c|cccc|c} \mathbf{a}_{11} & \mathbf{a}_{12} & \mathbf{a}_{13} & \cdots & \mathbf{a}_{1n} & c_1 \\ \mathbf{a}_{21} & \mathbf{a}_{22} & \mathbf{a}_{23} & \cdots & \mathbf{a}_{2n} & c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ \mathbf{a}_{m1} & \mathbf{a}_{m2} & \mathbf{a}_{m3} & \cdots & \mathbf{a}_{mn} & c_m \end{array}\right)$$

$$\stackrel{?}{=} \left(\begin{array}{c|c|ccc|c} \mathbf{a}_{11} & \mathbf{a}_{12} & \mathbf{a}_{13} & \cdots & \mathbf{a}_{1n} & \langle \mathbf{a}_1, \mathbf{s}\rangle + \mathbf{e}_1 \\ \mathbf{a}_{21} & \mathbf{a}_{22} & \mathbf{a}_{23} & \cdots & \mathbf{a}_{2n} & \langle \mathbf{a}_2, \mathbf{s}\rangle + \mathbf{e}_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ \mathbf{a}_{m1} & \mathbf{a}_{m2} & \mathbf{a}_{m3} & \cdots & \mathbf{a}_{mn} & \langle \mathbf{a}_m, \mathbf{s}\rangle + \mathbf{e}_m \end{array}\right)$$

BKW Algorithm VI

$$\Rightarrow \left(\begin{array}{c|c|cccc|l} \mathbf{a}_{11} & \mathbf{a}_{12} & \mathbf{a}_{13} & \cdots & \mathbf{a}_{1n} & & \langle \mathbf{a}_1, \mathbf{s}\rangle + \mathbf{e}_1 \\ 0 & \tilde{\mathbf{a}}_{22} & \tilde{\mathbf{a}}_{23} & \cdots & \tilde{\mathbf{a}}_{2n} & & \langle \tilde{\mathbf{a}}_2, \mathbf{s}\rangle + \mathbf{e}_2 - \frac{\mathbf{a}_{21}}{\mathbf{a}_{11}}\mathbf{e}_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \\ 0 & \tilde{\mathbf{a}}_{m2} & \tilde{\mathbf{a}}_{m3} & \cdots & \tilde{\mathbf{a}}_{mn} & & \langle \tilde{\mathbf{a}}_m, \mathbf{s}\rangle + \mathbf{e}_m - \frac{\mathbf{a}_{m1}}{\mathbf{a}_{11}}\mathbf{e}_1 \end{array}\right)$$

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

BKW Algorithm VII

The Problem and its Solution

- **Problem:**
- Noise of $\tilde{\mathbf{c}}_{ij}$ values increases rapidly
- **Strategy:** exploit that we have many rows: $m \gg n$.

BKW Algorithm VIII

We consider $a \approx \log n$ 'blocks' of b elements each.

$$\left(\begin{array}{cc|ccc|c} \mathbf{a}_{11} & \mathbf{a}_{12} & \mathbf{a}_{13} & \cdots & \mathbf{a}_{1n} & c_0 \\ \mathbf{a}_{21} & \mathbf{a}_{22} & \mathbf{a}_{23} & \cdots & \mathbf{a}_{2n} & c_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ \mathbf{a}_{m1} & \mathbf{a}_{m2} & \mathbf{a}_{m3} & \cdots & \mathbf{a}_{mn} & c_m \end{array}\right)$$

BKW Algorithm IX

For each block we build a table of all q^b possible values indexed by $\mathbb{Z}_q^b$.

$$T^0 = \left[\begin{array}{cc|ccc|c} -\lfloor \frac{q}{2} \rfloor & -\lfloor \frac{q}{2} \rfloor & \mathbf{t}_{13} & \cdots & \mathbf{t}_{1n} & c_{t,0} \\ -\lfloor \frac{q}{2} \rfloor & -\lfloor \frac{q}{2} \rfloor + 1 & \mathbf{t}_{23} & \cdots & \mathbf{t}_{2n} & c_{t,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ \lfloor \frac{q}{2} \rfloor & \lfloor \frac{q}{2} \rfloor & \mathbf{t}_{q^2 3} & \cdots & \mathbf{t}_{q^2 n} & c_{t,q^2} \end{array}\right]$$

For each $\mathbf{z} \in \mathbb{Z}_q^b$ we try to find a row in $\mathbf{A}$ such that it contains $\mathbf{z}$ as a subvector at the target indices.

BKW Algorithm X

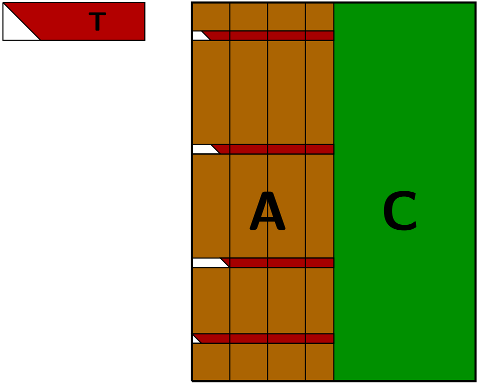
We use these tables to eliminate b entries in other rows. Assume $(\mathbf{a}_{21}, \mathbf{a}_{22}) = (\lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor + 1)$, then:

$$\left(\begin{array}{cc|ccc|c} \mathbf{a}_{11} & \mathbf{a}_{12} & \mathbf{a}_{13} & \cdots & \mathbf{a}_{1n} & c_0 \\ \mathbf{a}_{21} & \mathbf{a}_{22} & \mathbf{a}_{23} & \cdots & \mathbf{a}_{2n} & c_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ \mathbf{a}_{m1} & \mathbf{a}_{m2} & \mathbf{a}_{m3} & \cdots & \mathbf{a}_{mn} & c_m \end{array}\right)$$

$$+ \left[\begin{array}{cc|ccc|c} -\lfloor \frac{q}{2} \rfloor & -\lfloor \frac{q}{2} \rfloor & \mathbf{t}_{13} & \cdots & \mathbf{t}_{1n} & c_{t,0} \\ -\lfloor \frac{q}{2} \rfloor & -\lfloor \frac{q}{2} \rfloor + 1 & \mathbf{t}_{23} & \cdots & \mathbf{t}_{2n} & c_{t,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ \lfloor \frac{q}{2} \rfloor & \lfloor \frac{q}{2} \rfloor & \mathbf{t}_{q^2 3} & \cdots & \mathbf{t}_{q^2 n} & c_{t,q^2} \end{array}\right]$$

$$\Rightarrow \left(\begin{array}{cc|ccc|c} \mathbf{a}_{11} & \mathbf{a}_{12} & \mathbf{a}_{13} & \cdots & \mathbf{a}_{1n} & c_0 \\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{a}}_{23} & \cdots & \tilde{\mathbf{a}}_{2n} & \tilde{c}_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ \mathbf{a}_{m1} & \mathbf{a}_{m2} & \mathbf{a}_{m3} & \cdots & \mathbf{a}_{mn} & c_m \end{array}\right)$$

BKW Algorithm XI



One addition and no multiplications for clearing b columns.

BKW Algorithm XII

This gives a memory requirement of

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

and a time complexity of

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

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
ePrint Report 2012/636, 2012.
to appear in *Designs, Codes and Cryptography*.

BKW Algorithm XIII

Problem: All treatments of BKW 'very slightly heuristic' - we can lose perfect independence between processed samples. In general only a problem for small q - potential for problems if samples are 'inter-added'.

Inter-addition goes some way to resolving the memory requirements but concrete effects of losing independence not investigated.

Contents

The Setting

Assume $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\mathbb{Z}_2^n)$, i.e. all entries in secret $\mathbf{s}$ 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.

Zvika Brakerski and Vinod Vaikuntanathan.
Efficient fully homomorphic encryption from (standard) LWE.
In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011*, pages 97–106. IEEE, 2011.

Modulus Reduction I

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

$$\left(\left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \left\lfloor \frac{p}{q} \cdot c \right\rceil \right)$$

with

$$\begin{aligned}
\left\lfloor \frac{p}{q} \cdot c \right\rceil &= \left\lfloor \left\langle \frac{p}{q} \cdot \mathbf{a}, \mathbf{s} \right\rangle + \frac{p}{q} \cdot e \right\rceil \\
&= \left\lfloor \left\langle \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle + \left\langle \frac{p}{q} \cdot \mathbf{a} - \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle + \frac{p}{q} \cdot e \right\rceil \\
&= \left\langle \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle + \left\langle \frac{p}{q} \cdot \mathbf{a} - \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle + \frac{p}{q} \cdot e \pm [0, 0.5] \\
&= \left\langle \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle + e''.
\end{aligned}$$

Modulus Reduction II

Example

$$
\begin{aligned}
p, q &= 10, 20 \\
\mathbf{a} &= (8, -2, 0, 4, 2, -7), \\
\mathbf{s} &= (0, 1, 0, 0, 1, 1), \\
\langle \mathbf{a}, \mathbf{s} \rangle &= -7, \\
c &= -6 \\
\mathbf{a}' = \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil &= (4, -1, 0, 2, 1, -4) \\
\langle \mathbf{a}', \mathbf{s} \rangle &= -4, \\
\left\lfloor \frac{p}{q} \cdot c \right\rceil &= -4.
\end{aligned}
$$

Modulus Reduction III

Typically, we would choose

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

where σ_s is the standard deviation of elements in $\mathbf{s}$.

If $\mathbf{s}$ 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 being unchanged.

Lazy Modulus Switching I

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

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

as

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

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

Lazy Modulus Switching II

- $\mathbf{a}_{i,j}^h$ corresponds to $\lfloor p/q \cdot \mathbf{a}_{i,j} \rceil$

In order to clear the most significant bits in every component of the $\mathbf{a}_i$, we run the BKW algorithm on the matrix $[\mathbf{A} \mid \mathbf{c}]$ but only consider

$$[\mathbf{A}, \mathbf{c}]^h \quad := \begin{pmatrix} \mathbf{a}_{1,1}^h & \mathbf{a}_{1,2}^h & \dots & \mathbf{a}_{1,n}^h & c_1 \\ \mathbf{a}_{2,1}^h & \mathbf{a}_{2,2}^h & \dots & \mathbf{a}_{2,n}^h & c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{a}_{m,1}^h & \mathbf{a}_{m,2}^h & \dots & \mathbf{a}_{m,n}^h & c_m \end{pmatrix},$$

i.e. the "higher order bits", when searching for collisions.

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

Lazy Modulus Switching III

Example

Let $q, p = 16, 8$ and let $\mathbf{a} = (-3, 2, 4) \in \mathbb{Z}_q^3$.

Instead of searching for a vector $\mathbf{v} = (\pm 3, \cdot, \cdot)$ we ignore the least significant bit.

Hence, both $(\pm 3, \cdot, \cdot)$ and $(\pm 2, \cdot, \cdot)$ will do.

As a consequence we don't necessarily produce a vector $(0, \cdot, \cdot)$ after elimination, but one of $(0, \cdot, \cdot)$ or $(1, \cdot, \cdot)$, i.e. the first component is small.

Lazy Modulus Switching IV

Analogy

An analogy would be linear algebra with floating point numbers, where we define a tolerance when a small number counts as zero. We don't check `x == 0` but `abs(x) < tolerance`. The smaller p the bigger this tolerance.

Lazy Modulus Switching V

Difference with one-shot modulus reduction, i.e. rounding:

- We do not apply modulus reduction in one shot, but only when needed. We compute with high precision but compare with low precision.
- 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 an additional factor of $\sqrt{a/2}$.

Complexity I

BKW

$$\mathcal{O}\left(2^{cn} \cdot n \log_2^2 n\right)$$

Complexity II

BKW + naive modulus switching

$$\mathcal{O}\left(2^{\left(c+\frac{\log_2 d}{\log_2 n}\right)n} \cdot n \log_2^2 n\right)$$

where $0 < d \leq 1$ is a small constant (so $\log d < 0$).

Complexity III

BKW + lazy modulus switching

$$\mathcal{O}\left(2^{\left(c+\frac{\log_2 d - \frac{1}{2}\log_2 \log_2 n}{\log_2 n}\right) n} \cdot n \log_2^2 n\right)$$

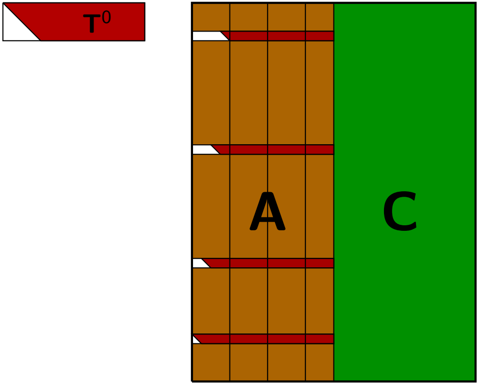
where $0 < d \leq 1$ is a small constant.

Contents

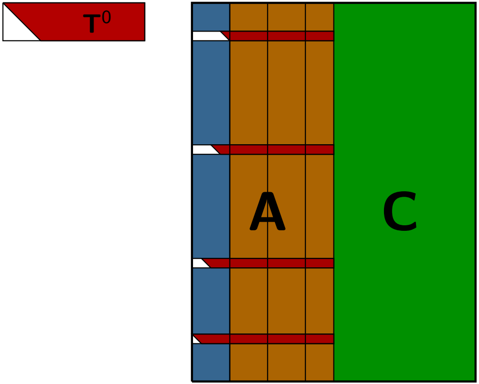
The Problem I

We use the entries in Table T^0 to make the first b components "small". However, as the algorithm proceeds we add up vectors with those small first b components producing vectors where the first b components are not that small any more.

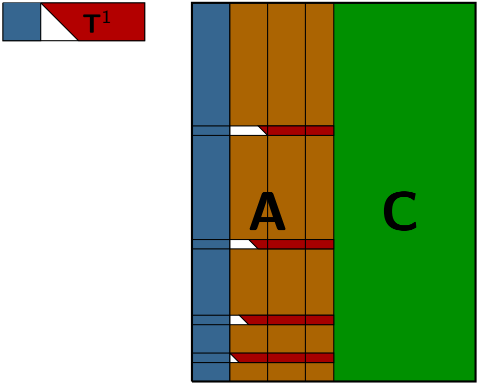
The Problem II



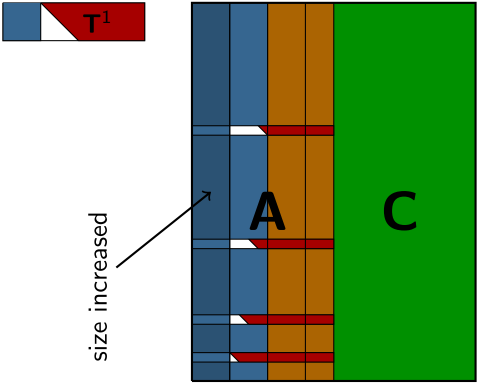
The Problem III



The Problem IV



The Problem V



The Problem VI

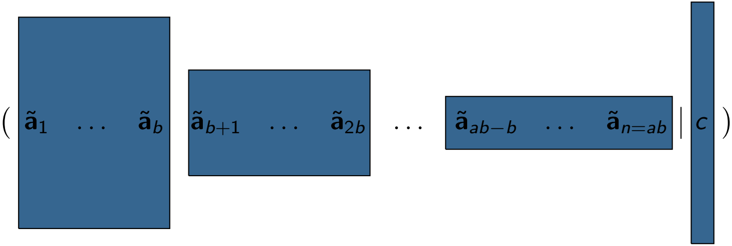
Lemma

Let $n \geq 1$ be the dimension of the LWE secret vector, q be a modulus, $b \in \mathbb{Z}$ with $1 \leq b \leq n$. Let also σ_r be the standard deviation of uniformly random elements in $\mathbb{Z}_{\lfloor q/p \rceil}$. Assuming all samples are independent, the components of $\tilde{\mathbf{a}} = \mathbf{a} - \mathbf{a}'$ returned by $B_{\mathbf{s},\chi}(b, \ell, p)$ satisfy:

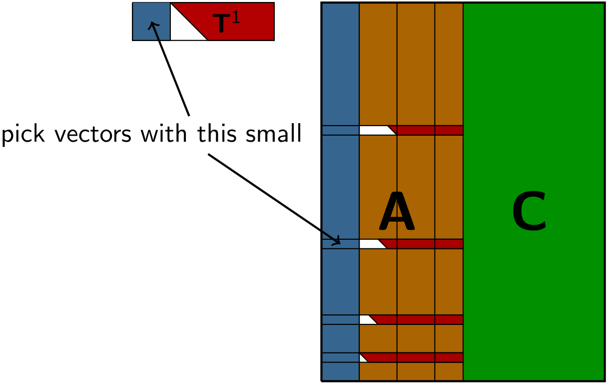
$$\mathrm{Var}(\tilde{\mathbf{a}}_{(i)}) = 2^{\ell - \lfloor i/b \rfloor} \sigma_r^2, \text{ for } 0 \leq \lfloor i/b \rfloor \leq \ell$$

and $\mathrm{Var}\big(\mathcal{U}(\mathbb{Z}_q)\big)$ *for* $\lfloor i/b \rfloor > \ell$.

The Problem VII



Unnatural Selection I



Unnatural Selection II

Finding vectors by chance with the first $bi - b$ components unusually small to populate T^i is easier than finding vectors where the first i components are unusally small.

Impact I

We keep sampling and pick that candidate vector **a** for index **z** in $\mathbf{T}^1$ where the first b components are unusually small.

$\Rightarrow$ We need to establish how much we can expect the size to drop if we sample a given number of times.

Impact II

Assumption (Cowboy)

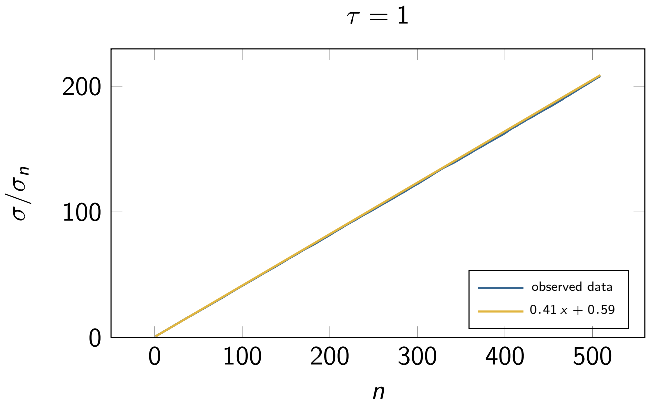
Let the vectors $\mathbf{x}_0, \dots, \mathbf{x}_{n-1} \in \mathbb{Z}_q^\tau$ *be sampled from some distribution* $\mathcal{D}$ *such that* $\sigma^2 = \mathrm{Var}(\mathbf{x}_{i,(j)})$ *where* $\mathcal{D}$ *is any distribution on (sub-)vectors observable in our algorithm. Let* $\mathbf{x}^* = \min_{abs}(\mathbf{x}_0, \dots, \mathbf{x}_{n-1})$ *where* $\min_{abs}$ *picks that vector* $\mathbf{x}^*$ *with* $\sum_{j=0}^{b\cdot\ell-1} |\mathbf{x}^*_{(j)}|$ *minimal. The standard deviation* $\sigma_n = \sqrt{\mathrm{Var}(\mathbf{x}^*_{(0)})} = \dots = \sqrt{\mathrm{Var}(\mathbf{x}^*_{(\tau-1)})}$ *of components in* $\mathbf{x}^*$ *satisfies*

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

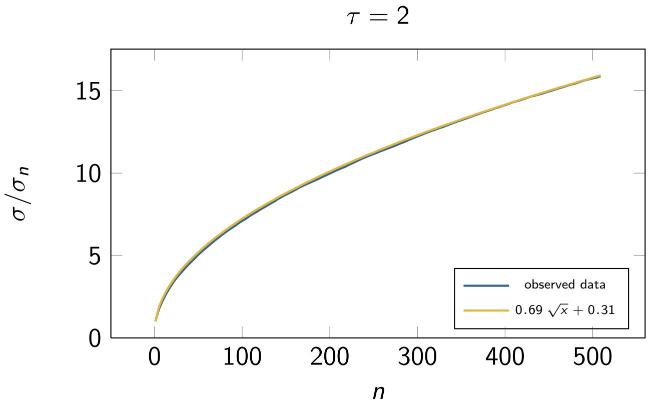
with

$$c_\tau \approx \frac{1}{5}\sqrt{\tau} + \frac{1}{3}.$$

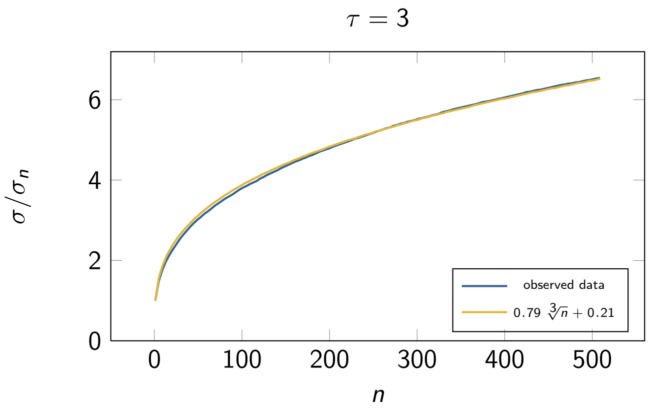
Impact III



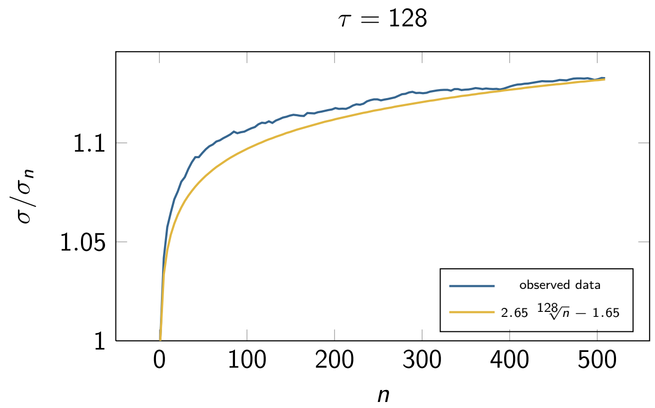
Impact IV


$\tau = 2$
σ/σ_n
15
10
5
0
0
100
200
300
400
500
n
observed data
$0.69\sqrt{x} + 0.31$

Impact V



Impact VI


τ = 128
1.1
1.05
1
σ/σn
0
100
200
300
400
500
n
observed data
2.65 128√n − 1.65

Contents

BKW Variants I

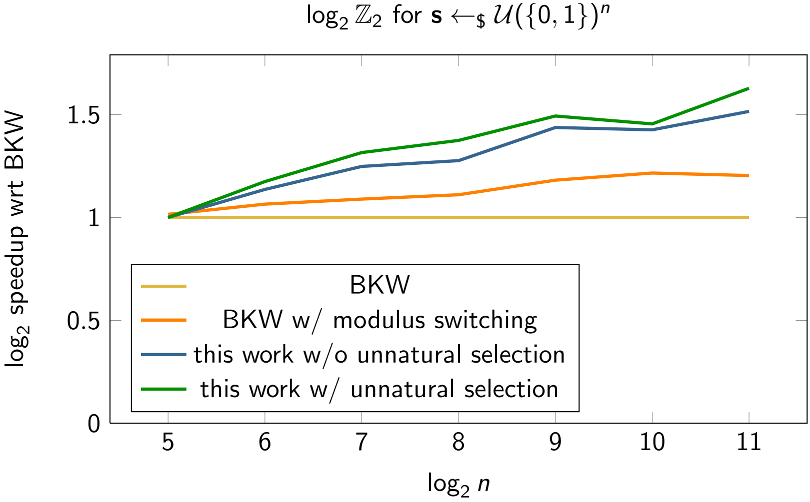
	BKW		+ Mod. Switch	
n	$\log \mathbb{Z}_2$	log mem	$\log \mathbb{Z}_2$	log mem
128	97.6	90.0	89.6	81.2
256	182.1	174.2	164.0	156.7
512	361.0	352.8	305.6	297.9
1024	705.5	697.0	580.2	572.2
2048	1388.7	1379.9	1153.6	1145.3
	This Work (1)		This Work (2)	
n	$\log \mathbb{Z}_2$	log mem	$\log \mathbb{Z}_2$	log mem
128	78.2	70.8	74.2	46.3
256	142.7	134.9	132.5	67.1
512	251.2	243.1	241.8	180.0
1024	494.8	486.5	485.0	407.5
2048	916.4	907.9	853.2	758.9

Table : Cost for solving Decision-LWE with advantage ≈ 1 for BKW and BKZ variants where q and σ are chosen as in Regev's scheme and $\mathbf{s} \leftarrow_\$ \mathcal{U}(\mathbb{Z}_2^n)$

BKW Variants II

"$\log \mathbb{Z}_2$" gives the number of "bit operations" and "$\log \text{mem}$" the memory requirement of $\mathbb{Z}_q$ elements. All logarithms are base 2.

BKW Variants III



. . . and Previous Work I

MITM guess the two halves of the secret and search for a collision

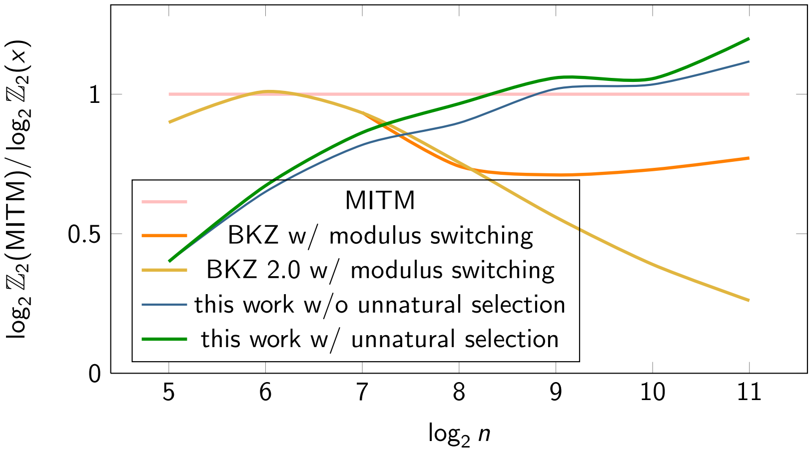
BKZ solve SIS using the BKZ algorithm with $\log_2 T_{sec} = 1.8/\log_2 \delta_0 - 110$.

Yuanmi Chen and Phong Q. Nguyen.
BKZ 2.0: better lattice security estimates.
In *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20, Berlin, Heidelberg, 2011. Springer Verlag.

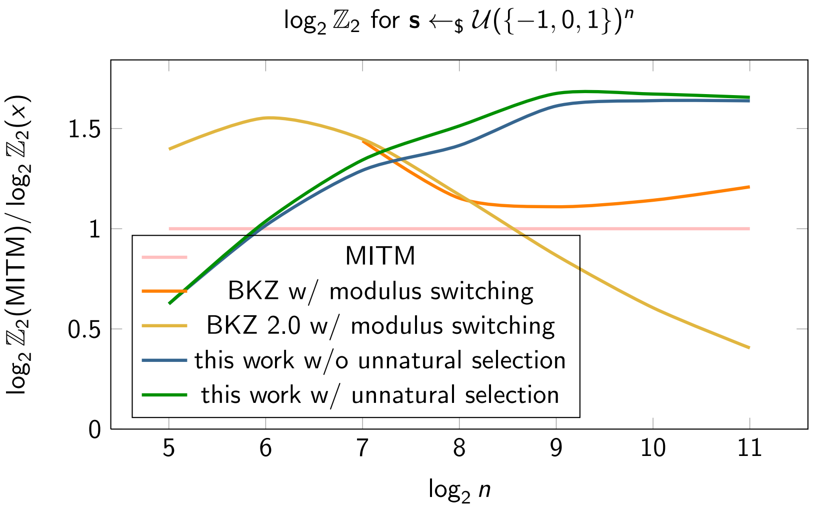
Richard Lindner and Chris Peikert.
Better key sizes (and attacks) for LWE-based encryption.
In *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339, Berlin, Heidelberg, New York, 2011. Springer Verlag.

. . . and Previous Work II

$\log_2 \mathbb{Z}_2$ for $\mathbf{s} \leftarrow_\$ \mathcal{U}(\{0,1\})^n$



. . . and Previous Work III



Fun and Useful Problems

1. BKW (and the most effective lattice attacks) need more samples than what LWE-based cryptosystems offer. We can attempt to deal with this by forming new samples from old samples at the cost of increasing the noise slightly. However, this means our samples are not independent any more. What is the effect of this?
2. The main obstacle to running BKW "in practice" is its demand for memory. With modulus switching and unnatural selection we have a strategy to trade running time for memory to some extend. Can we find configurations where it becomes feasible to run BKW on instances other than very small toy instances?

Conclusion

Questions?