Solving LWE with BKW

Workshop on Tools for Asymmetric Cryptanalysis

Martin R. Albrecht
@martinralbrecht

8 October 2015

Contents

Learning With Errors

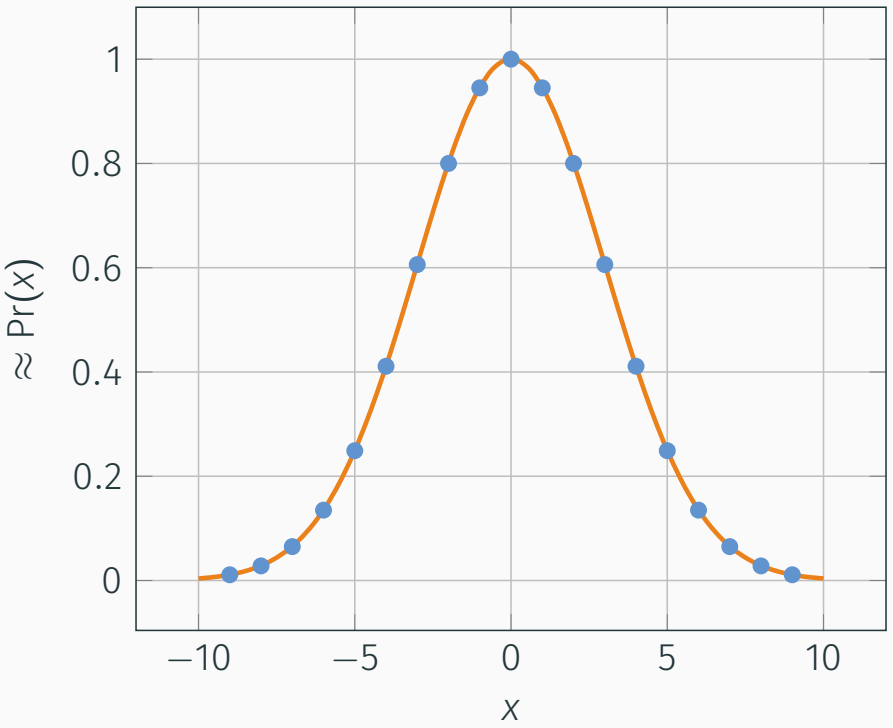
The Learning with Errors (LWE) problem was defined by Oded Regev [Reg05].

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

$$\begin{pmatrix} \\ \mathbf{c} \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \leftarrow & n & \rightarrow \\ & & \\ & \mathbf{A} & \\ & & \end{pmatrix} \times \begin{pmatrix} \\ \mathbf{s} \\ \\ \end{pmatrix} + \begin{pmatrix} \\ \mathbf{e} \\ \\ \\ \end{pmatrix}$$

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

Parameters



- Parameters are:
  - dimension n,
  - modulus q (e.g. $q \approx n^2$),
  - noise size α (e.g. $\alpha q \approx \sqrt{n}$),
  - number of samples m.
- Elements of $\mathbf{A}, \mathbf{s}, \mathbf{e}, \mathbf{c}$ are in $\mathbb{Z}_q$.
- $\mathbf{e}$ is sampled from a discrete Gaussian with width

$$\sigma = \frac{\alpha q}{\sqrt{2\pi}}.$$

Asymptotic Security

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

Zvika Brakerski et al. Classical hardness of learning with errors. In: *45th ACM STOC*. ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 575–584

Asymptotic Security

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

Zvika Brakerski et al. Classical hardness of learning with errors. In: *45th ACM STOC*. ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 575–584

We want to build crypto systems

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

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

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

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

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

If there no error, we may apply Gaussian elimination:

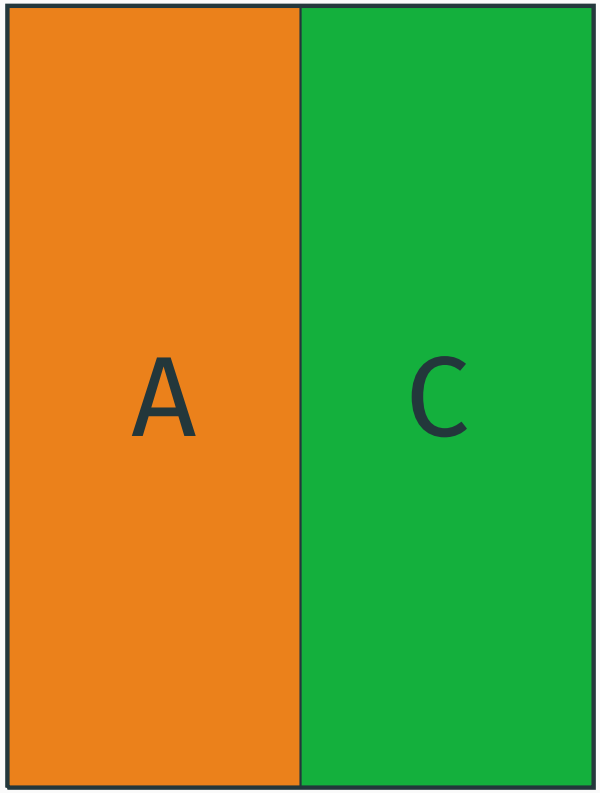
$$[A \mid C] = \left(\begin{array}{cccc|ccc} a_{11} & a_{12} & \cdots & a_{1n} & c_{11} & \cdots & c_{1\ell} \\ a_{21} & a_{22} & \cdots & a_{2n} & c_{21} & \cdots & c_{2\ell} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & c_{m1} & \cdots & c_{m\ell}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

to recover

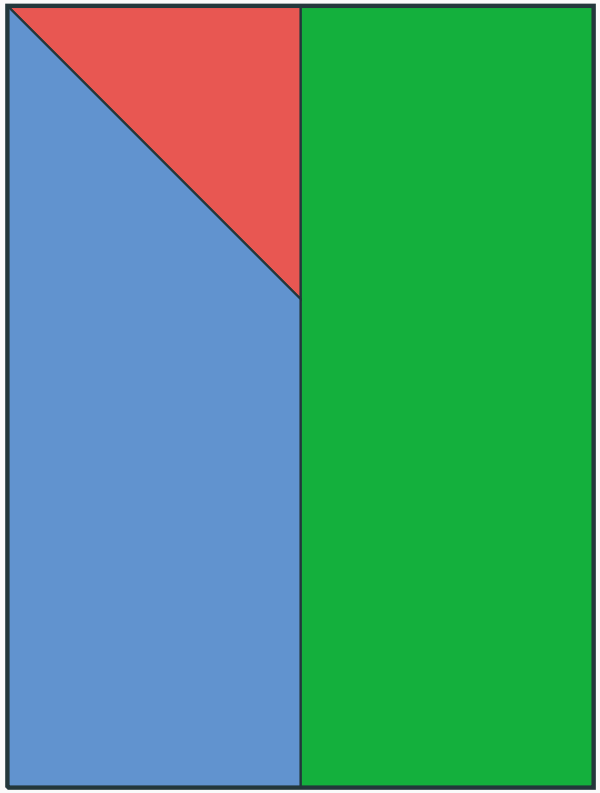
$$[A' \mid C'] = \left(\begin{array}{cccc|ccc} a_{11} & a_{12} & \cdots & a_{1n} & c_{11} & \cdots & c_{1\ell} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{a}_{rn} & \tilde{c}_{r1} & \cdots & \tilde{c}_{r\ell} \\ \vdots & \vdots & \ddots & \vdots & \vdots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

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

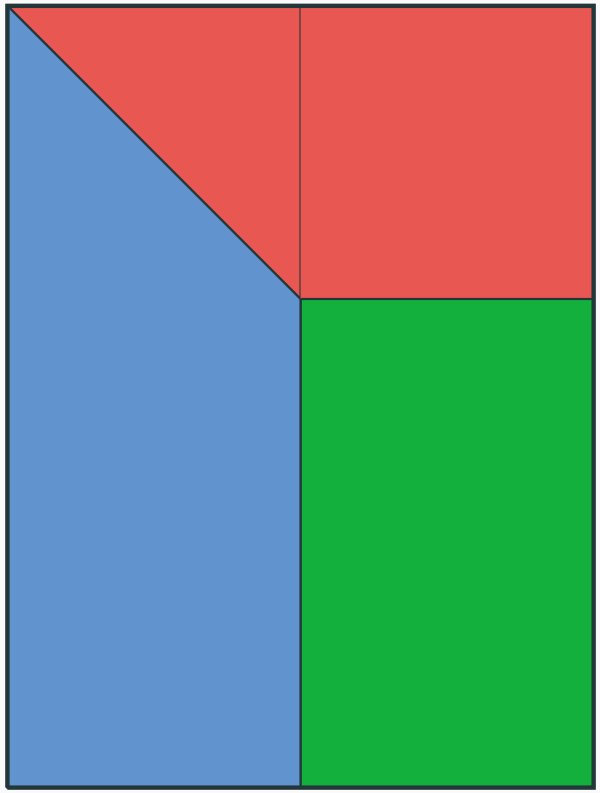
Gaussian elimination II



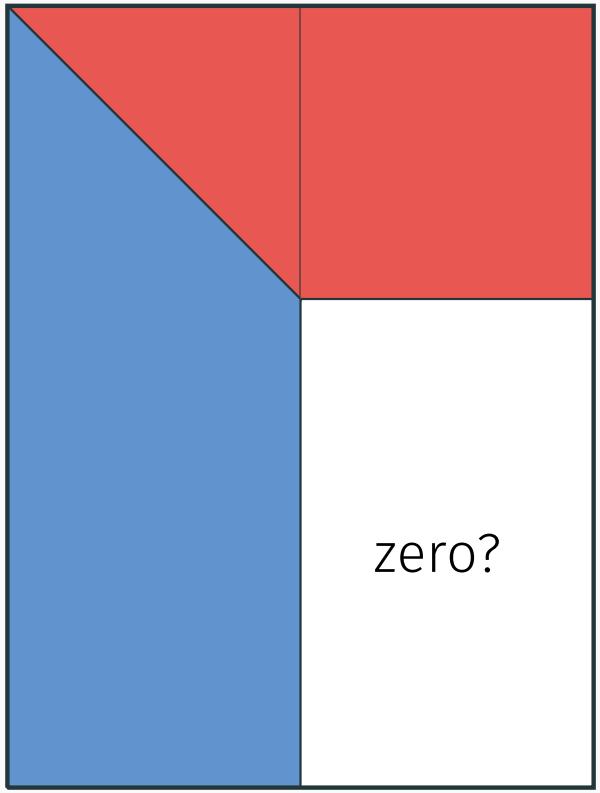
Gaussian elimination III



Gaussian elimination IV



Gaussian elimination V



Contents

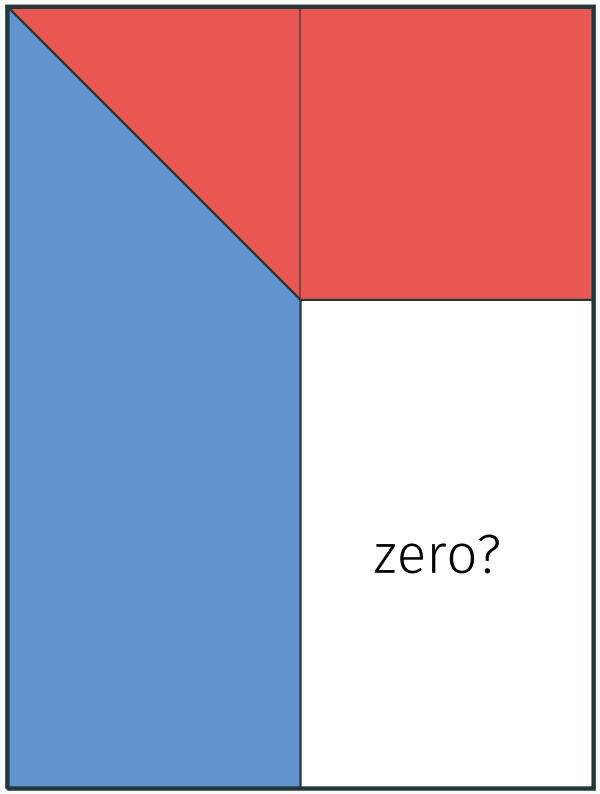
BKW Algorithm I

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. In: *Journal of the ACM* 50.4 (July 2003), pp. 506–519. ISSN: 0004-5411 (print), 1557-735X (electronic). DOI: http://doi.acm.org/10.1145/792538.792543

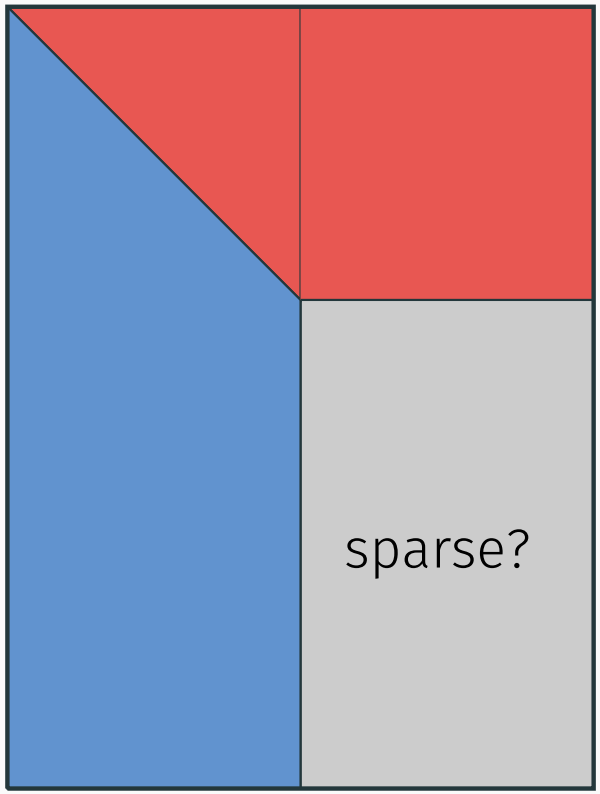
BKW Algorithm II

Goal in noise-free case:



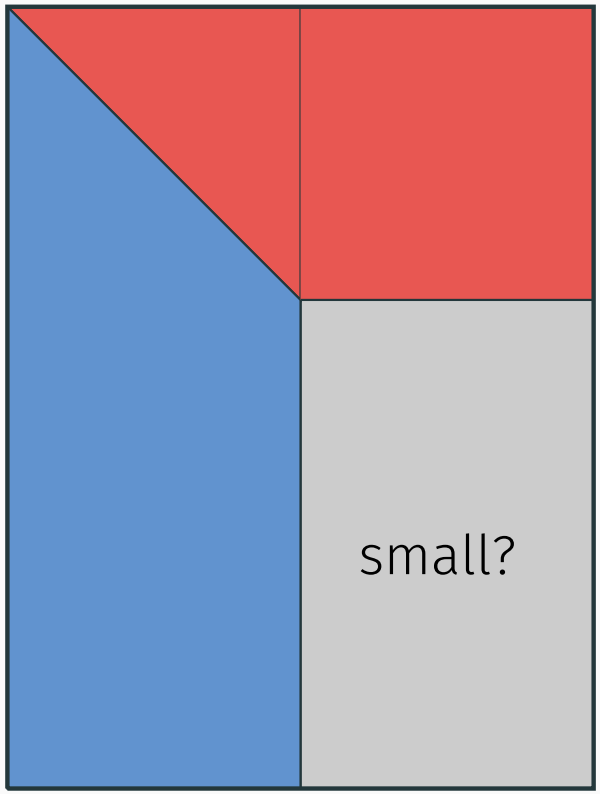
BKW Algorithm III

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



BKW Algorithm IV

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



BKW Algorithm V

We revisit Gaussian elimination:

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

$$\stackrel{?}{=} \left(\begin{array}{c|c|ccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & \langle a_1, s\rangle + e_1 \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} & \langle a_2, s\rangle + e_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} & \langle a_m, s\rangle + e_m \end{array}\right)$$

BKW Algorithm VI

$$\Rightarrow \left(\begin{array}{c|c|ccc|l} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & \langle a_1, s\rangle + e_1 \\ 0 & \tilde{a}_{22} & \tilde{a}_{23} & \cdots & \tilde{a}_{2n} & \langle \tilde{a}_2, s\rangle + e_2 - \frac{a_{21}}{a_{11}} e_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & \tilde{a}_{m2} & \tilde{a}_{m3} & \cdots & \tilde{a}_{mn} & \langle \tilde{a}_m, s\rangle + e_m - \frac{a_{m1}}{a_{11}} e_1 \end{array}\right)$$

- $\frac{a_{i1}}{a_{11}}$ 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.

The Problem and its Solution

- Problem: additions and multiplications increase noise
  $\Rightarrow$ noise of $\tilde{\mathbf{c}}_{ij}$ values increases rapidly
- Strategy: exploit that we have many rows: $m \gg n$.

BKW Algorithm VIII

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

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

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:

BKW Algorithm XI

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

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

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

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 et al. On the Complexity of the BKW Algorithm on LWE. . In: *Designs, Codes and Cryptography* (2015)

Contents

The Setting

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

- common setting in cryptography for performance reasons;
- 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: *52nd FOCS*. ed. by Rafail Ostrovsky. IEEE Computer Society Press, Oct. 2011, pp. 97–106

Intuition

```
double a[m][n];
float b[m][n];
...
/* modulus switching */
for(int i=0; i<m; i++) {
  for(int j=0; j<n; j++) {
    b[i][j] = (float)a[i][j]; // floats are enough
  }
}
...
/* lazy modulus switching */
for(int i=0; i<m; i++) {
  if( abs(a[i][i]) < 0.001 ) { // never compare with zero
    ...
  }
}
```

Modulus Switching 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 Switching II

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 b usually is unchanged.

Lazy Modulus Switching I

All arithmetic is performed in $\mathbb{Z}_q$, but collisions are sought in $\mathbb{Z}_p$.

Martin R. Albrecht et al. Lazy Modulus Switching for the BKW Algorithm on LWE. . In: *PKC 2014*. Ed. by Hugo Krawczyk. Vol. 8383. LNCS. Springer, Mar. 2014, pp. 429–445. DOI: 10.1007/978-3-642-54631-0_25

Lazy Modulus Switching II

Example

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

Instead of searching for a vector $\mathbf{v} = (\pm 3, \cdot, \cdot)$ to eliminate the first component 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 III

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

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

as

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

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

Lazy Modulus Switching IV

- $\mathbf{a}_{i,j}^h$ corresponds to $\lfloor p/q \cdot \mathbf{a}_{i,j} \rceil$ and
- $\mathbf{a}_{i,j}^l$ corresponds to $\lfloor p/q \cdot \mathbf{a}_{i,j} \rceil - p/q \cdot \mathbf{a}_{i,j}$, the rounding error.

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

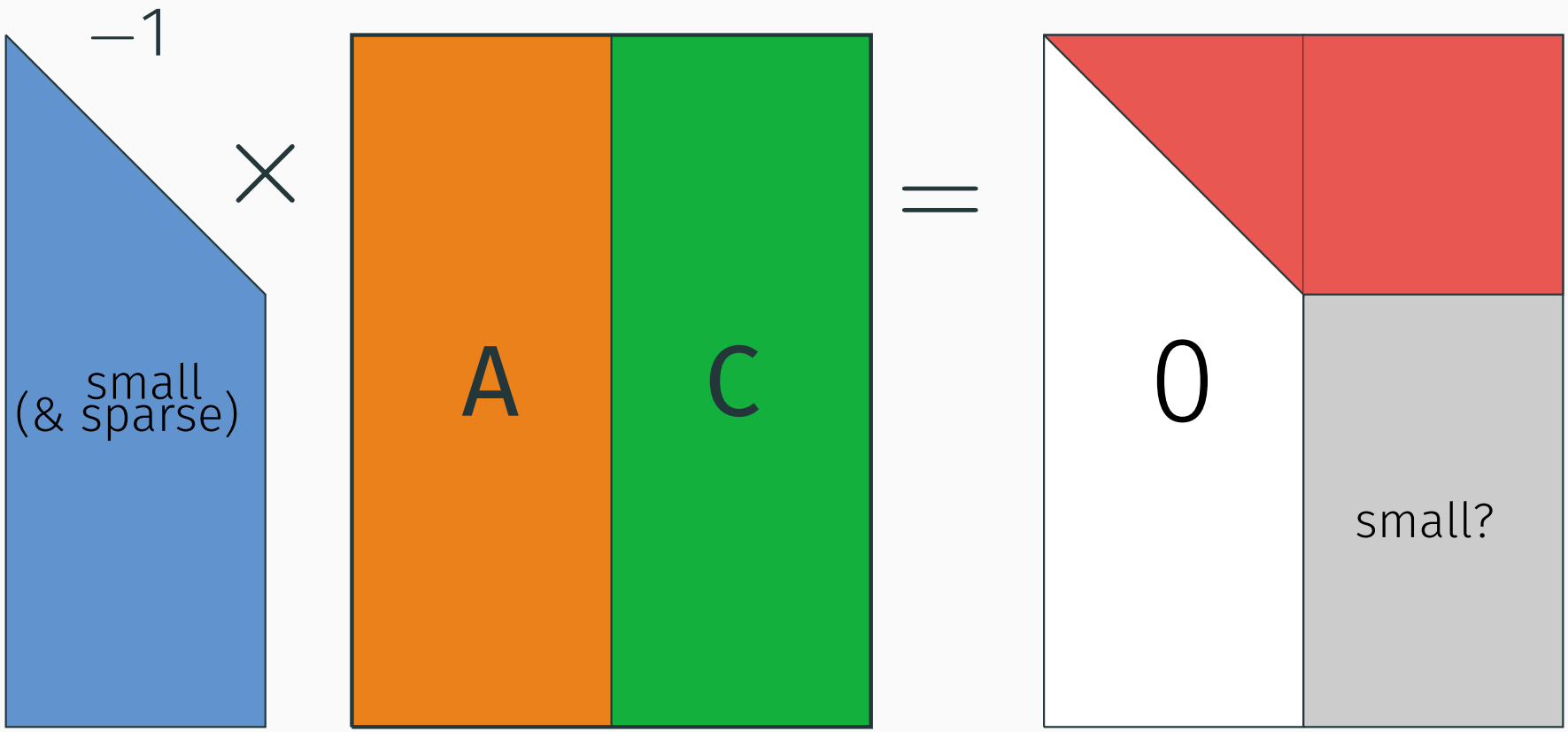
$$[\mathbf{A}, \mathbf{c}]^h \quad := \quad \begin{pmatrix} \mathbf{a}_{1,1}^h & \mathbf{a}_{1,2}^h & \cdots & \mathbf{a}_{1,n}^h & c_1 \\ \mathbf{a}_{2,1}^h & \mathbf{a}_{2,2}^h & \cdots & \mathbf{a}_{2,n}^h & c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{a}_{m,1}^h & \mathbf{a}_{m,2}^h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

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

We only manage elimination tables for the most significant κ bits.

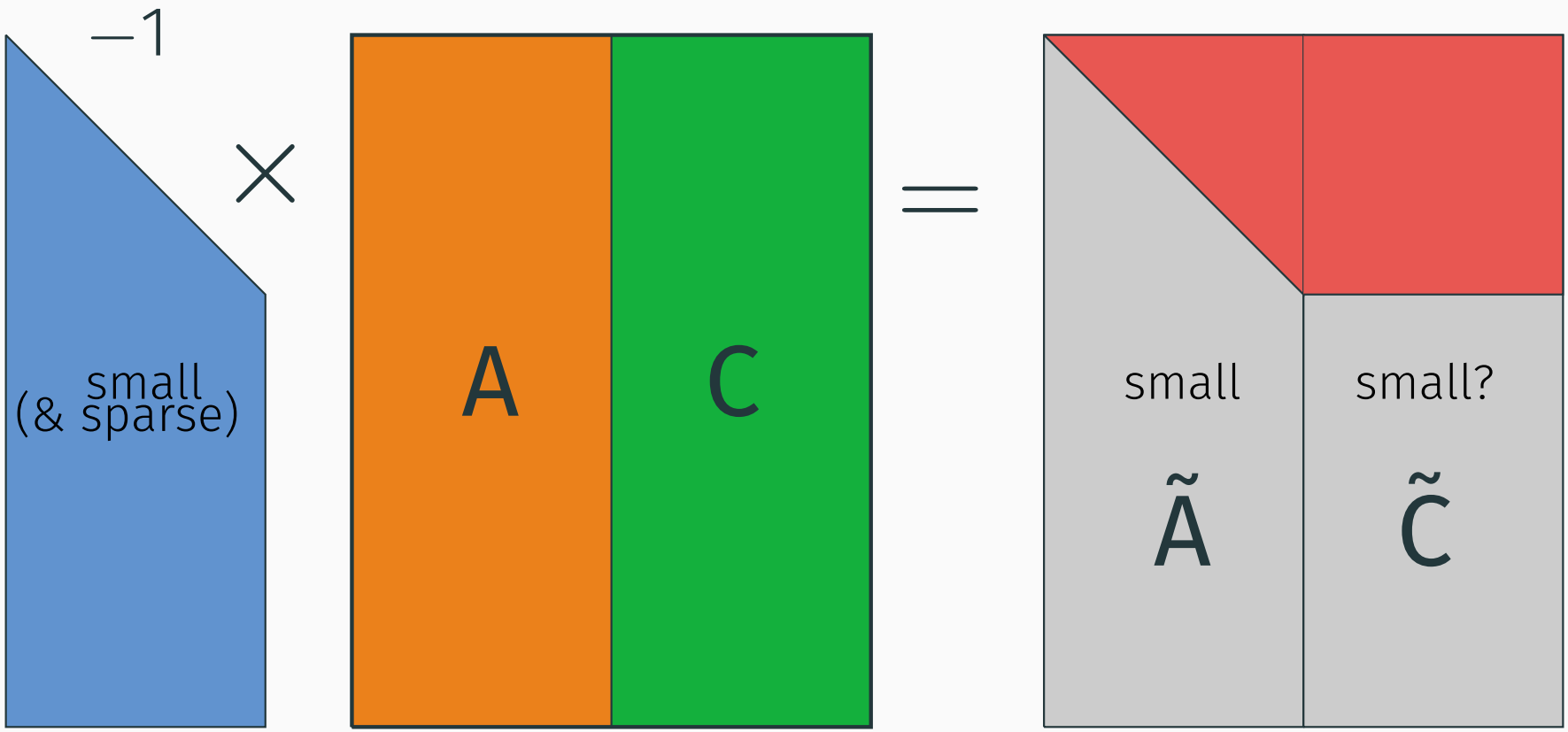
Lazy Modulus Switching V

BKW without lazy modulus switching:



Lazy Modulus Switching VI

BKW with lazy modulus switching ($\tilde{\mathbf{A}} \times \mathbf{S} + \tilde{\mathbf{E}} = \tilde{\mathbf{C}}$):



Lazy Modulus Switching VII

Difference to plain 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.

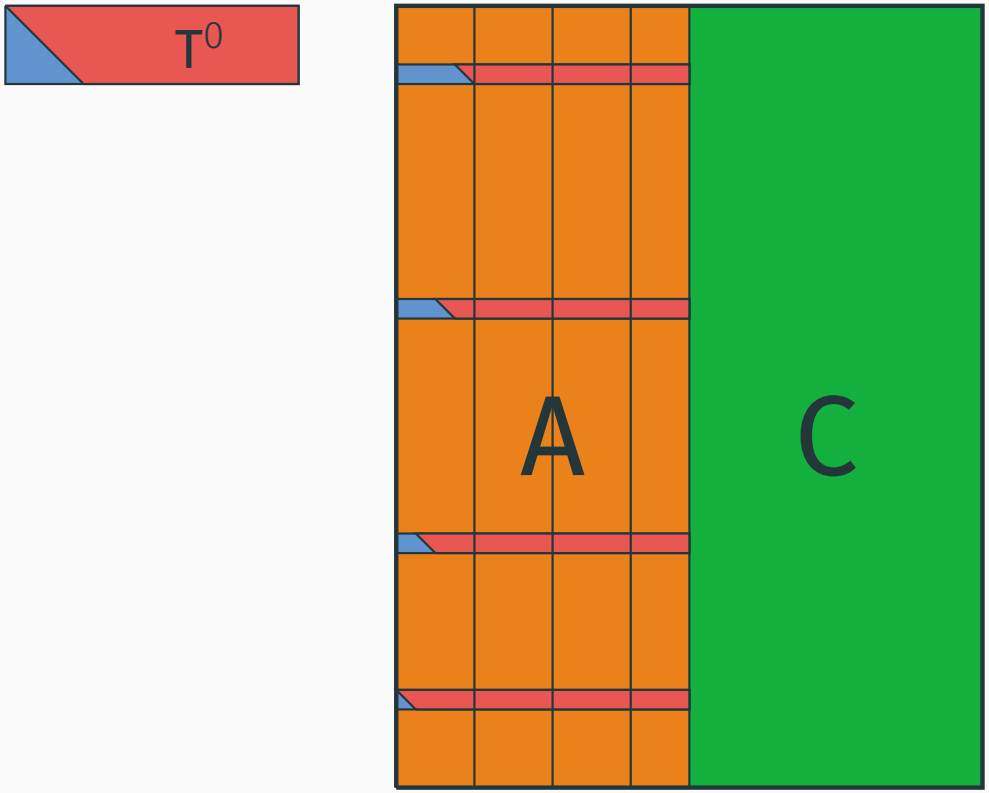
We reduce p by an additional factor of $\sqrt{a/2}$.

Contents

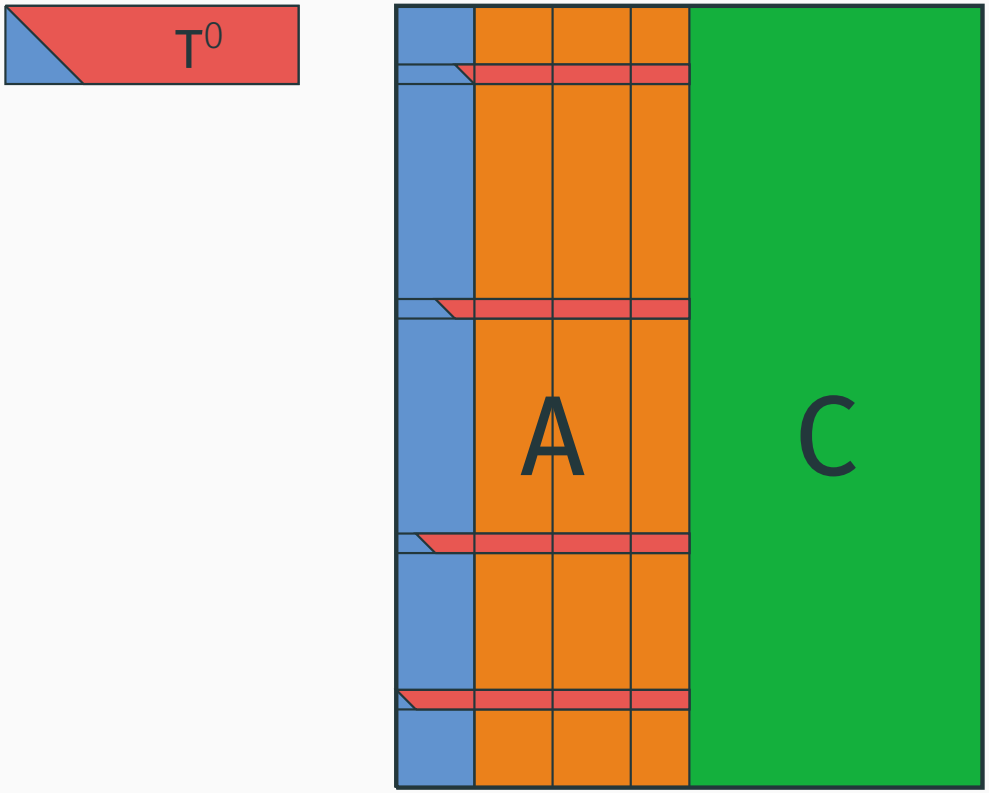
Uneven Noise Contribution I

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.

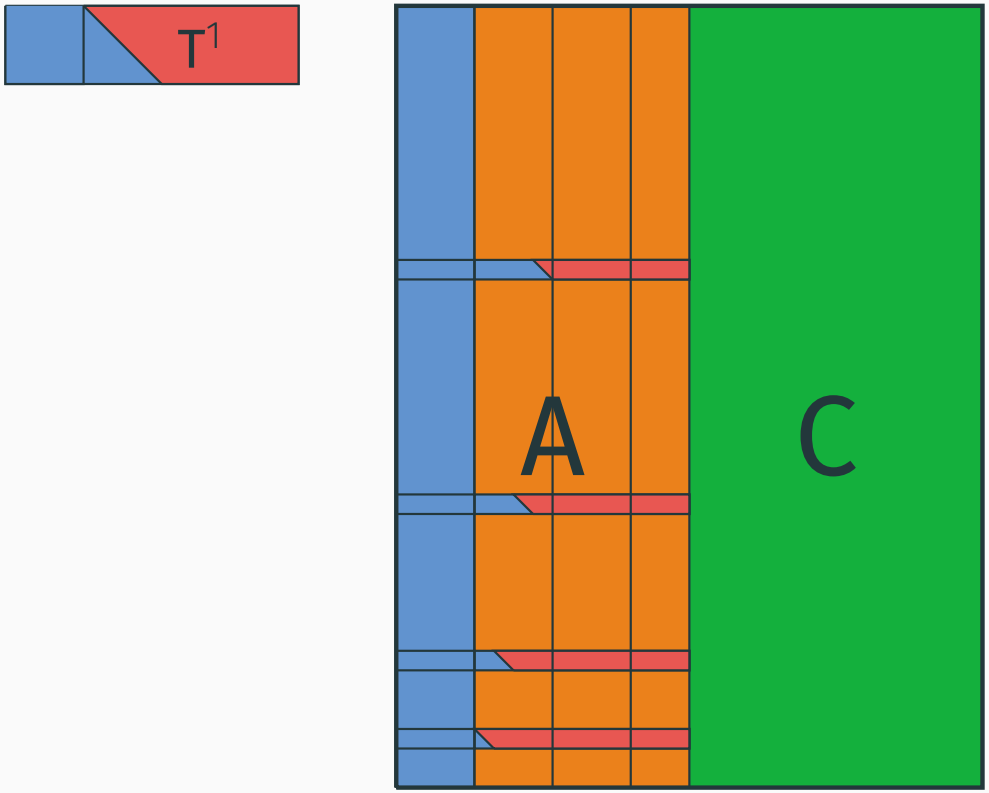
Uneven Noise Contribution II



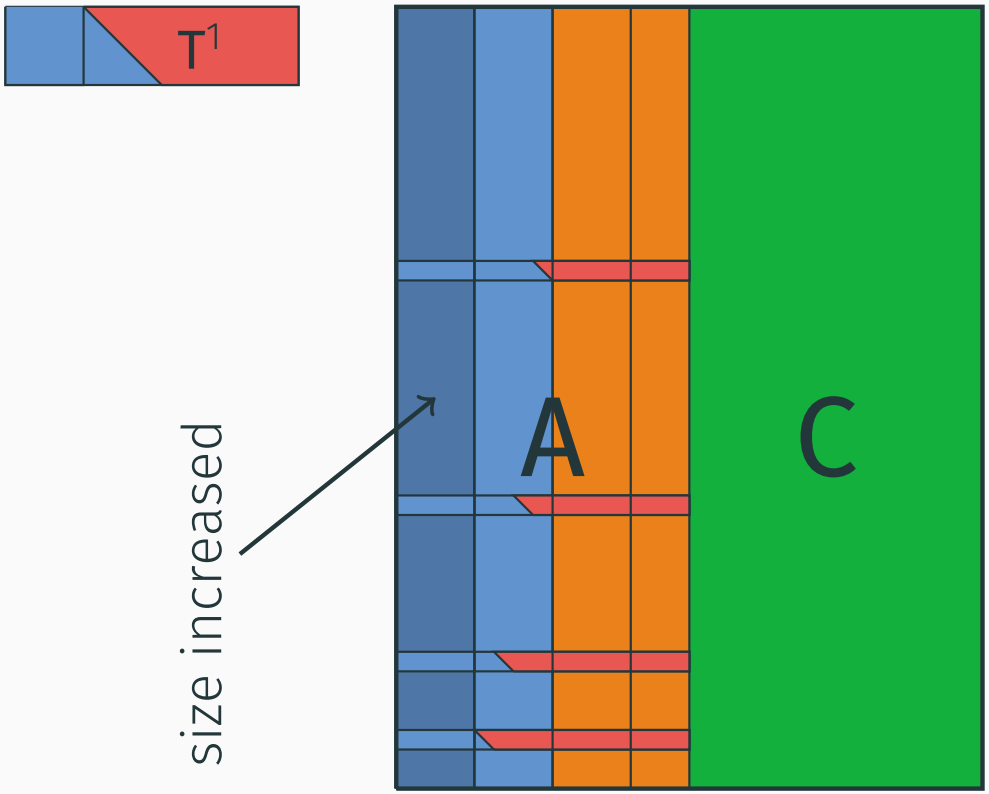
Uneven Noise Contribution III



Uneven Noise Contribution IV



Uneven Noise Contribution V



Approach 1

- One strategy to deal with this issue is to work harder on the first couple of tables to reduce the noise
- This works because for these tables we have exceptionally many samples passing by.

Approach 2

- We could pick decreasing moduli (increasing noise levels) for consecutive blocks to address this problem.
- This, however, would increase the complexity which would now be dominated by the size of the table T^0.
- To compensate for this, we may choose increasing blocksizes b_i for each of the a blocks

Paul Kirchner and Pierre-Alain Fouque. An Improved BKW Algorithm for LWE with Applications to Cryptography and Lattices. In: *CRYPTO 2015, Part I*. ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Aug. 2015, pp. 43–62. DOI: 10.1007/978-3-662-47989-6_3

Coded-BKW

Next talk:

Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-BKW: Solving LWE Using Lattice Codes. In: *CRYPTO 2015, Part I*. ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Aug. 2015, pp. 23–42. DOI: 10.1007/978-3-662-47989-6_2

Fin

Questions?

`https://bitbucket.org/malb/lwe-estimator`