Key Recovery for LWE in Polynomial Time

Kim Laine and Kristin Lauter

UC Berkeley, Microsoft Research

DIMACS Workshop on The Mathematics of Post-Quantum Cryptography
Rutgers University

January 16, 2015

Structure of this talk

1. Introduce the problem (search-)LWE
2. Polynomial time attack
3. Practical performance
4. Security implications
5. Conclusions

Definition of LWE

Learning With Errors (Regev, 2005) is a hard computational problem that several homomorphic cryptosystems are based on.

$q = 2^r$ an integer modulus

n an integer, $\mathbf{s} \in \mathbb{Z}_q^n$ a secret vector chosen uniformly at random

$D_{\mathbb{Z},\sigma}$ (error distribution) the discrete Gaussian distribution centered at 0, with standard deviation σ

Definition 1 (LWE sample)

An LWE sample is a pair $(\mathbf{a}, t) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\mathbf{a}$ is sampled uniformly at random from $\mathbb{Z}_q^n$, $e \leftarrow D_{\mathbb{Z},\sigma}$ and $t = \left[\langle \mathbf{a}, \mathbf{s} \rangle + e\right]_q = \langle \mathbf{a}, \mathbf{s} \rangle_q + e \in (-q/2, q/2)$.

Definition 2 (search-LWE$_{n,r,d,\sigma}$)

Given d LWE samples $(\mathbf{a}_i, t_i)$, the problem search-LWE$_{n,r,d,\sigma}$ is to recover the secret vector $\mathbf{s}$.

Security reductions

- When $q = \text{poly}(n)$, polynomial time *quantum reduction* from worst-case GapSVP (Regev)
- When $q = \text{poly}(n)$, polynomial time *classical reduction* from worst-case of an easier (less studied) variant of GapSVP (Peikert)
- When q is exponential in n, polynomial time *classical reduction* from worst-case GapSVP (Peikert)

In all cases the approximating factor is $\widetilde{O}(nq/\sigma)$. When the approximating factor is polynomial in n, there is no known algorithm for solving GapSVP in polynomial time.

Question 1: What happens in practice when q is exponential in n?

Question 2: What happens in practice when q is "pretty large"?

Connection to lattices

Let Λ be the $(n+d)$-dimensional lattice generated by the rows of the matrix

$$\begin{pmatrix} q & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & q & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & q & 0 & 0 & \cdots & 0 \\ \mathbf{a}_0[0] & \mathbf{a}_1[0] & \cdots & \mathbf{a}_{d-1}[0] & 1/2^N & 0 & \cdots & 0 \\ \mathbf{a}_0[1] & \mathbf{a}_1[1] & \cdots & \mathbf{a}_{d-1}[1] & 0 & 1/2^N & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \mathbf{a}_0[n-1] & \mathbf{a}_1[n-1] & \cdots & \mathbf{a}_{d-1}[n-1] & 0 & 0 & \cdots & 1/2^N \end{pmatrix}.$$

Easy to see:

$$\mathbf{v} = \left[\langle \mathbf{a}_0, \mathbf{s}\rangle_q, \langle \mathbf{a}_1, \mathbf{s}\rangle_q, \ldots, \langle \mathbf{a}_{d-1}, \mathbf{s}\rangle_q, \mathbf{s}[0]/2^N, \mathbf{s}[1]/2^N, \ldots, \mathbf{s}[n-1]/2^N\right] \in \Lambda$$

$$\mathbf{u} = \left[t_0, t_1, \ldots, t_{d-1}, 0, \ldots, 0\right] \notin \Lambda \text{ but is close to } \mathbf{v} \text{ if } N \text{ is big}$$

Connection to lattices

- Assumption: We have access to any number of LWE samples so d can be anything we want.
- "Simply" solve approx-CVP in Λ with input $\mathbf{u}$ to find $\mathbf{v}$ and $\mathbf{s}$.
- Method: Find a good basis for the lattice and use Babai's *nearest planes* method.
- $\dim \Lambda = n + d$ can be huge.
- We might not find the correct vector $\mathbf{v}$.

Problem: The approximation factor might have to be small so need very good basis (use BKZ-2.0 with large blocksize): exponential time?

Polynomial time attack

Claim: If q is large enough and d is chosen appropriately, need to solve CVP only up to exponential factor (in n)!

- So in certain cases: Run LLL (polynomial time in n) and use Babai's method to find candidate for $\mathbf{v}$.
- Recovered vector is almost certainly $\mathbf{v}$.

Explanation: If $q = 2^{O(n)}$ and d chosen appropriately, probability of having a lattice vector within some large radius $< q = 2^{O(n)}$ of $\mathbf{u}$ can be made to be very small.

- This is a restriction on q.
- But we know there is one, namely $\mathbf{v}$.
- So even if LLL-Babai performs exponentially poorly, it will still find a close vector within this radius, which is very likely to be $\mathbf{v}$.

More precise claim:

Let $\delta = \big(1 + (1/2)\,\log_2 n\big)/n$ and suppose $r := \log_2 q > 7(1/2+\delta)n$. Let

$$d = \left\lceil \sqrt{(1/2+\delta)(r+1)n} \right\rceil .$$

Solve ℓ_σ from

$$nd\sqrt{e}\,2^{r-\ell_\sigma} = \sigma \exp\left(\frac{2^{2r-2\ell_\sigma-1}}{\sigma^2}\right) .$$

If

$$(1/2+\delta)n + 2\sqrt{(1/2+\delta)(r+1)n} < \ell_\sigma \,,$$

i.e. σ is small enough, then *search*-$\mathsf{LWE}_{n,r,d,D_{\mathbb{Z},\sigma}}$ can be solved in probabilistic polynomial time in n by computing an LLL-reduced basis of the given basis of Λ and using Babai's nearest planes method to $\mathbf{u}$. The algorithm successfully returns $\mathbf{v}$ with very high probability. The running time is polynomial in n.

Practical performance

Remarks:

- In the proof many very strong approximations are made to guarantee success.
- Does not tell much about practical performance.
- Input basis to LLL has very special form: Performance of LLL-Babai?

Here is a way of measuring the practical performance:

- Guaranteed approx factor in LLL-Babai is $2^{(n+d)/2}$ (used in proof)
- Instead write $2^{\mu(n+d)}$ and go through the proof; get a formula for μ needed to succeed.
- Run examples and compute the required μ. Failed: effectively μ was larger; Succeeded: effectively μ was smaller.
- Gives an idea of how big effectively μ can be expected to be
- Extrapolate behavior of μ to larger examples (do they fail or succeed?)

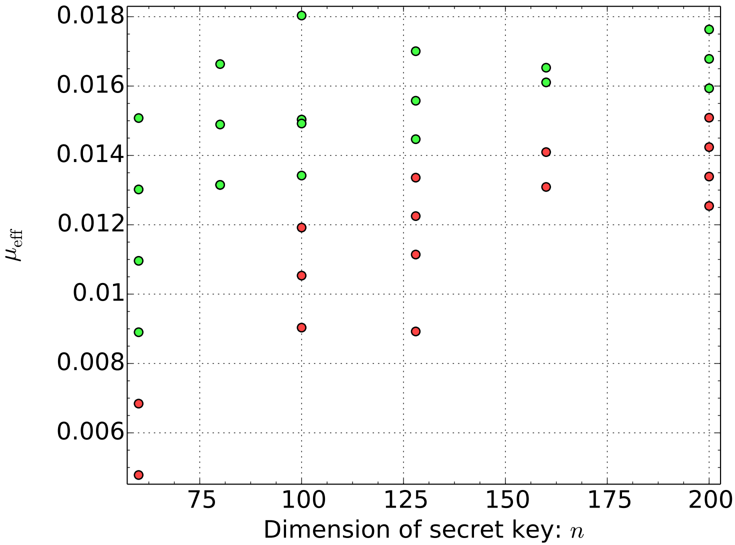
Experimental results

For a particular experiment to succeed, the effective lower bound for μ that needs to be realized is given approximately by

$$\mu_{\text{eff}} := \frac{1}{n+d}\left[-\frac{rn}{d} + r - 1 - \log_2(\lceil\sigma\rceil\sqrt{n+d})\right]$$

Here are the results as a function of n:

Experimental results



Security implications

Ok, so what can be broken? Here are some examples:

n	d	r	σ	Time
80	320	16	3.233	1.6h
100	400	20	6.346	6.8h
128	572	23	3.097	24h
160	540	27	3.077	1d 5h
200	550	31	3.049	2d 13h

Security implications

What happens for larger n? Here $\sigma = 8/\sqrt{2\pi}$.

n	d	r	μ_{eff}
512	1388	65	0.0171
1024	2576	120	0.0176
2048	5152	235	0.0184
4096	10104	465	0.0188

Will these succeed or fail?

Distinguishing attack

- Usually (e.g. Lindner-Peikert, van de Pol-Smart, Lepoint-Naehrig) recommended parameters based on hardness of the distinguishing attack, i.e. distinguishing of valid LWE samples from random data.

Theorem 3

Compute a basis with root-Hermite factor δ for a d-dimensional q-ary lattice. Valid LWE samples can be distinguished from random data with advantage (suppose $d > n$)

$$\exp\left[-\pi\left(\frac{\delta^d\sqrt{2\pi}\,\sigma}{q^{1-n/d}}\right)\right].$$

- Nguyen-Stehle [*LLL on the Average*]: With LLL expect $\delta \approx 1.02$
- With $\delta \approx 1.02$ and r as large as in the examples above, distinguishing advantage is very high!
- So none of this is surprising: *Distinguishing implies key recovery* but in time proportional to q (huge!).
- The interesting result is that key recovery can actually be done so easily in these cases.

Security implications?

Therefore:

- At least with pure LLL the attack does not threaten commonly recommended secure parameters, but how significantly does performance improve if we improve the basis using other methods?
- In some special applications might want to evaluate very deep circuits homomorphically and need very large r, small σ.
- E.g. evaluating some block ciphers homomorphically.
- In these cases you really do need a large enough n.

Security of LWE based cryptosystems depends very strongly on the parameters.

Thank you!