Key Recovery for LWE in Polynomial Time

Kim Laine[1] and **Kristin Lauter**[2]

[1] University of California, Berkeley
`laine@math.berkeley.edu`
[2] Microsoft Research, Redmond
`klauter@microsoft.com`

Abstract. We present a generalization of the Hidden Number Problem and generalize the Boneh-Venkatesan method [BV96, Shp05] for solving it in polynomial time. We then use this to mount a key recovery attack on LWE which runs in polynomial time using the LLL lattice basis reduction algorithm. Success can be guaranteed with overwhelming probability for narrow error distribution when $q \geq 2^{O(n)}$, where n is the dimension of the secret key, and we can give an explicit constant in the exponent, but in practice the performance is significantly better. The same attack can be used to break RLWE for the same parameter ranges.
Two main types of attacks are already known for LWE: the distinguishing attack [MR09] and the decoding attack ([LP11]), which uses the BKZ algorithm. Our key recovery attack is interesting because it runs in polynomial time and yields simple and concrete security estimates for a wide range of parameters depending in a clear and explicit way on the effective approximation factor in the LLL algorithm. We ran the attack for hundreds of LWE instances demonstrating successful key recovery attacks and yielding information about the effective approximation factor as the lattice dimension grows (see Figure 3). For example, we successfully recover the secret key for an instance with $n = 350$ in about 3.5 days on a single machine.

Keywords: Hidden Number Problem, LWE, key recovery, lattice-based cryptography

1 Introduction

Learning with errors (LWE), introduced by Regev in 2005, is a generalization of the *learning parity with noise* problem. Roughly speaking, the problem setting involves a system of d approximate linear equations in n variables modulo q:

$$\begin{aligned}
a_{0,0}s_0 + a_{0,1}s_1 + \ldots + a_{0,n-1}s_{n-1} &\approx t_0 \pmod q \\
a_{1,0}s_0 + a_{1,1}s_1 + \ldots + a_{1,n-1}s_{n-1} &\approx t_1 \pmod q \\
\vdots \qquad\qquad\qquad & \qquad \vdots \\
a_{d-1,0}s_0 + a_{d-1,1}s_1 + \ldots + a_{d-1,n-1}s_{n-1} &\approx t_{d-1} \pmod q
\end{aligned}$$

Two questions can now be asked. The *decision* version of LWE asks to distinguish whether a vector $\mathbf{t} \in \mathbb{Z}_q^d$ is of the form $\left[t_0, t_1, \ldots, t_{d-1}\right]$ or sampled uniformly at random from $\mathbb{Z}_q^d$. The *search* version asks to solve the system, i.e. to find $\mathbf{s} = \left[s_0, s_1, \ldots, s_{n-1}\right]$.
In the seminal paper [Reg09] Regev proved that, in some parameter settings, if *decision*-LWE can be solved in time polynomial in n, then there are polynomial time quantum algorithms for solving worst cases of the lattice problems GapSVP[3] and SIVP[4] with $\gamma = \mathrm{poly}(n)$. These problems are widely believed to be hard with the best known algorithms having exponential complexity in n. In the same paper he proved that when $q = \mathrm{poly}(n)$ there is a rather simple polynomial time search-to-decision reduction when *decision*-LWE can be solved with exponentially good advantage.
Later Peikert [Pei09] presented a purely classical reduction to *search*-LWE in the case $q = \mathrm{poly}(n)$ from a new lattice problem $\mathrm{GapSVP}_{\zeta,\gamma}$, which is an easier variant of GapSVP_γ. Most importantly,

[3] GapSVP_γ takes as input a lattice Λ and a rational number L. The problem is to decide whether the shortest vector in the lattice has length $\lambda_1(\Lambda) < L$ or $\lambda_1(\Lambda) > \gamma \cdot L$. This is essentially a decision version of SVP.
[4] SIVP_γ is the problem of finding a basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ for a lattice Λ, such that $||\mathbf{b}_n|| < \gamma \cdot \lambda_n(\Lambda)$, where $\lambda_i(\Lambda)$ denotes the i-th shortest vector in the lattice.

there is no longer a reduction from the worst-case lattice *search* problem SIVP. When the modulus is $q \geq 2^{n/2}$ the situation is slightly better: *search*-LWE can be classically reduced from the usual worst-case GapSVP.

In [Reg09] Regev also presented a public-key cryptosystem based on LWE. Since then, the LWE problem and its variant *ring*-LWE (RLWE) [LPR13] have become hugely important as building blocks for homomorphic and post-quantum private and public-key primitives and protocols [BV11, Bra12, BGV12, BV14, LP11, LNV, GLN, BLN, LLN, BCNS14].

In this work we define a higher dimensional generalization of the Hidden Number Problem and construct a polynomial time algorithm in the spirit of Boneh and Venkatesan [BV96] (see also [Shp05]) to solve it. We then adapt this same approach to target LWE and obtain a polynomial time key recovery attack to solve *search*-LWE, which applies in the case of exponentially large modulus q and narrow error distribution. For large enough n, we find that success can be guaranteed with high probability roughly when $\log_2 q > 2n$, but that in practice significantly smaller moduli are vulnerable.

Our polynomial time key recovery attack should be viewed in the context of the known attacks on LWE, namely the *distinguishing attack* [MR09], and the *decoding attack* [LP11]. The decoding attack is stronger than the distinguishing attack in that it recovers the secret key, but it relies on the BKZ algorithm so it does not run in polynomial time and its performance is difficult to analyze. In practice, for applications [LNV, GLN, BLN, LLN, BCNS14], LWE parameters are selected to avoid even the distinguishing attack and are not currently vulnerable to our polynomial time key recovery, although vulnerable parameter ranges do appear very naturally in applications to homomorphic encryption. This demonstrates the importance of following the recommended parameters even if one does not care about the threat of the distinguishing attack.

But our attack is interesting for several additional reasons: First, it demonstrates that for large enough q the classical reduction ([Pei09], see Theorem 6 below) is not relevant for cryptography in the sense that both the *search*-LWE and GapSVP problems can be solved in polynomial time for that parameter range (see Remark 5 below).

Second, our attack is efficient enough that we were able to run it for hundreds of LWE instances for different parameter sizes. The results are shown in Figure 3, where the green dots indicate successful secret key recovery, while the red dots indicate failed attempts. These experiments allow us to observe the effective approximation factor in the LLL algorithm. Although theory guarantees that LLL finds a vector of length no more than γ times the length of the shortest vector, where $\gamma = 2^{\mu N}$, N is the lattice dimension and $\mu = \frac{1}{2}$, in practice it is known (see for example [NS06]) that μ can be expected to be much smaller. More correctly, what we observe is the effective approximation factor appearing in Babai's *nearest planes* method [Bab86] given an LLL reduced basis. Secure parameter selection for LWE depends heavily on the asymptotic behavior of the number μ in the LLL-Babai algorithm, and our experiments shown in Figure 3 demonstrate the rough growth of μ as the lattice dimension grows, up to dimension around 800.

Finally, we show how practical the attack is by running it on increasingly large parameter sets. For example, the attack for $n = 350$ terminates successfully in roughly 3.5 days, running on a single machine. The actual running time for the attack in practice matches very closely the predicted running time for optimized LLL implementation, $O(N^4 \log^2 q)$, which makes it easy for us to predict the running time of the attack for larger parameter sizes.

The paper is organized as follows. In Section 2 we study an n-dimensional Generalized Hidden Number Problem (GHNP), which is closely related to *search*-LWE. We describe a generalization of the method of Boneh and Venkatesan [BV96, Shp05] for solving it in polynomial time when the parameters are in certain ranges. Most importantly the modulus q must be exponential in the dimension n. In Section 3 we use the results of Section 2 to mount a polynomial time key recovery attack on *search*-LWE, which is guaranteed to succeed with overwhelming probability for certain LWE parameter ranges. In Section 4 we study the attack in practice and present several examples up to key dimension $n = 350$. We attempt to extrapolate these results to larger n to understand better when a polynomial time attack can be expected to succeed. In Sections 5, 6 we study the security implications of our attack. We observe that vulnerable parameters come up very naturally in applications of LWE to homomorphic cryptography and discuss implications for LWE parameter selection.

2 Generalized Hidden Number Problem

We start by recalling the definition of the *hidden number problem* (HNP) and subsequently describe an n-dimensional generalization of it. Next we generalize the approach of [BV96, Shp05] to find a polynomial time algorithm for solving this *generalized hidden number problem* (GHNP), which is essentially solving an approximate-CVP in a particular lattice using LLL [LLL82] combined with Babai's *nearest planes* method [Bab86]. The main content of the result is to see that while LLL-Babai is only guaranteed to solve CVP up an exponential approximation factor, it is good enough in certain cases to solve GHNP.

Notation. In all of this work we assume that q is an odd prime and $r := \log_2 q$. By $\mathbb{Z}_q$ we denote integers modulo q, but as a set of representatives for the congruence classes we use integers in the interval $(-q/2, q/2)$. By a subscript q we denote the unique representative of an integer modulo q within this interval.

Definition 1. *By* $\mathrm{MSB}_\ell(k)$ *we denote the* ℓ *most significant bits of the integer* k *not counting the sign. For example,*

$$\mathrm{MSB}_4(175) = 160\,, \qquad \mathrm{MSB}_5(-175) = -168\,.$$

Most importantly, we always have

$$\left|k - \mathrm{MSB}_\ell(k)\right| < 2^{\lfloor \log_2 |k| \rfloor + 1 - \ell}\,.$$

Definition 2 (HNP). *Let* $s \in \mathbb{Z}_q$ *be a fixed secret number chosen uniformly at random. Given* d *samples of the form*

$$\Big(a, \mathrm{MSB}_\ell\left(\,[as]_q\,\right)\Big) \in \mathbb{Z}_q \times \mathbb{Z}_q$$

where $a \in \mathbb{Z}_q$ *are chosen uniformly at random, the problem* $\mathrm{HNP}_{r,\ell,d}$ *is to recover* s.

Boneh and Venkatesan [BV96] showed how HNP can be solved in polynomial time. Their method used polynomial time lattice reduction [LLL82] combined with Babai's nearest planes method [Bab86] to solve an approximate-CVP in a particular lattice. The algorithm for solving the HNP was then used to attack the Diffie-Hellman problem in cryptography. More precisely,

Theorem 1 ([BV96, Shp05]). *If* d *and* ℓ *are chosen appropriately,* $\mathrm{HNP}_{r,\ell,d}$ *can be solved in time poly*(r). *For instance, this happens when* $d = \ell = \sqrt{2r}$.

We will generalize Definition 2 and Theorem 1 to n dimensions.

Definition 3 (GHNP). *Let* $\mathbf{s} \in \mathbb{Z}_q^n$ *be a fixed secret vector chosen uniformly at random. Given* d *samples of the form*

$$\Big(\mathbf{a}, \mathrm{MSB}_\ell\left(\langle \mathbf{a}, \mathbf{s}\rangle_q\right)\Big) \in \mathbb{Z}_q^n \times \mathbb{Z}_q\,,$$

where $\mathbf{a} \in \mathbb{Z}_q^n$ *are chosen uniformly at random, the problem* $\mathrm{GHNP}_{n,r,\ell,d}$ *is to recover* $\mathbf{s}$.

In the rest of this section we will describe a probabilistic polynomial time algorithm for solving $\mathrm{GHNP}_{n,r,\ell,d}$ when $r, d \in O(n)$ and n, ℓ are big enough. Our approach is a direct generalization of the method of [BV96].

Notation. We denote the i-th coefficient of a vector $\mathbf{v}$ by $\mathbf{v}[i]$.

We want to solve $\mathrm{GHNP}_{n,r,\ell,d}$ with samples $(\mathbf{a}_i, \mathrm{MSB}_\ell\left(\langle \mathbf{a}_i, \mathbf{s}\rangle_q\right))$, where $i = 0, \dots, d-1$. The first step is to make this into a lattice problem by considering the full $(n+d)$-dimensional lattice $\Lambda_{n,r,\ell,d}$ spanned by the rows of

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

Clearly $\Lambda_{n,r,\ell,d}$ contains the vector

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

Denote

$$u_i = \mathrm{MSB}_\ell\left(\langle \mathbf{a}_i, \mathbf{s}\rangle_q\right). \tag{3}$$

The distance between $\langle \mathbf{a}_i, \mathbf{s}\rangle_q$ and u_i can be bounded using Definition 1:

$$|\langle \mathbf{a}_i, \mathbf{s}\rangle_q - u_i| < 2^{\lfloor \log_2 |\langle \mathbf{a}_i, \mathbf{s}\rangle_q| \rfloor + 1 - \ell} \leq 2^{\lfloor \log_2 (q-1) \rfloor - \ell} < 2^{r-\ell}. \tag{4}$$

The vector

$$\mathbf{u} = \left[u_0, u_1, \ldots, u_{d-1}, 0, \ldots, 0\right] \in \mathbb{R}^{n+d} \tag{5}$$

is not in $\Lambda_{n,r,\ell,d}$, but using (4) we can bound its Euclidean distance from $\mathbf{v}$:

$$||\mathbf{v} - \mathbf{u}|| \leq \sqrt{n+d}\, 2^{r-\ell}.$$

Theorem 2 (LLL-Babai). *Let Λ be a lattice of dimension N. An approximate-CVP in Λ can be solved in polynomial time up to an approximating factor $2^{\mu N}$. A value of $\mu = 1/2$ is guaranteed, but in practice significantly better performance (smaller μ) can be expected.*

Proof. The value $\mu = 1/2$ follows from the result of Babai [Bab86] and the performance guarantee of LLL [LLL82]. The arguments in [NS06] about average perfomance of LLL on random lattices explains why LLL yields in some sense much better bases than the theoretical result of [LLL82] promises. Due to this, the algorithm of Babai can also be expected to yield significantly better results than is guaranteed by theory. Both LLL and Babai's method have complexity polynomial in N. □

The key to solving $\mathrm{GHNP}_{n,r,\ell,d}$ in polynomial time is to argue that, in many cases, the algorithm LLL-Babai in Theorem 2 actually solves approximate-CVP for $\mathbf{u}$ well enough to recover $\mathbf{v}$, from which $\mathbf{s}$ can be read.
Consider what happens if we run LLL-Babai with input $\mathbf{u}$. By Theorem 2 it is guaranteed to output a vector

$$\begin{aligned}\mathbf{w} =& \Big[\langle \mathbf{a}_0, \mathbf{t}\rangle + q\mathbf{k}[0], \langle \mathbf{a}_1, \mathbf{t}\rangle + q\mathbf{k}[1], \ldots, \langle \mathbf{a}_{d-1}, \mathbf{t}\rangle + q\mathbf{k}[d-1], \\ & \mathbf{t}[0]/2^{\ell-1}, \mathbf{t}[1]/2^{\ell-1}, \ldots, \mathbf{t}[n-1]/2^{\ell-1}\Big] \in \Lambda_{n,r,\ell,d}, \end{aligned} \tag{6}$$

where $\mathbf{t} \in \mathbb{Z}^n$, $\mathbf{k} \in \mathbb{Z}^d$, such that

$$||\mathbf{v} - \mathbf{w}|| \leq ||\mathbf{v} - \mathbf{u}|| + ||\mathbf{u} - \mathbf{w}|| \leq \left(1 + 2^{\mu(n+d)}\right) ||\mathbf{v} - \mathbf{u}|| \leq \left(1 + 2^{\mu(n+d)}\right) \sqrt{n+d}\, 2^{r-\ell}. \tag{7}$$

If this is the case, then all differences $(\mathbf{v} - \mathbf{w})[j]$ must lie in the interval

$$\left[-\left(1 + 2^{\mu(n+d)}\right) \sqrt{n+d}\, 2^{r-\ell}, \left(1 + 2^{\mu(n+d)}\right) \sqrt{n+d}\, 2^{r-\ell}\right]. \tag{8}$$

We can assume that $\mathbf{t} \in \mathbb{Z}_q^n$. Namely, let $\mathbf{t}_{\mathrm{red}}$ denote a vector in $\mathbb{Z}_q^n$ that is obtained by reducing the entries of $\mathbf{t}$ modulo q. By replacing $\mathbf{t}$ with $\mathbf{t}_{\mathrm{red}}$ in the definition of $\mathbf{w}$, we obtain a new lattice vector which differs in the first d entries from $\mathbf{w}$ by multiples of q. But adding suitable multiples of the first n generators of the lattice $\Lambda_{n,r,\ell,d}$ (first n rows of the matrix) to this vector yields a lattice vector $\mathbf{w}_{\mathrm{red}}$ whose first d entries are the same as those of $\mathbf{w}$ and whose remaining n entries are possibly smaller of absolute value than those of $\mathbf{w}$.
The first d differences $(\mathbf{v} - \mathbf{w})[j]$ are of the form $\langle \mathbf{a}_j, \mathbf{s} - \mathbf{t}\rangle_q + q\widetilde{\mathbf{k}}[j]$, where $\widetilde{\mathbf{k}} \in \mathbb{Z}^d$. If we assume that

$$\left(1 + 2^{\mu(n+d)}\right) \sqrt{n+d}\, 2^{r-\ell} < \frac{q}{2}, \tag{9}$$

or equivalently that

$$\ell > \log_2\left[\left(1 + 2^{\mu(n+d)}\right) \sqrt{n+d}\right] + 1, \tag{10}$$

then $\widetilde{\mathbf{k}} = \mathbf{0}$, so for the first d differences we obtain the simple conditions

$$\left|\langle \mathbf{a}_j, \mathbf{s} - \mathbf{t}\rangle_q\right| \leq \left(1 + 2^{\mu(n+d)}\right) \sqrt{n+d}\, 2^{r-\ell} < \frac{q}{2}. \tag{11}$$

The last n differences $(\mathbf{v}-\mathbf{w})[j]$ are of the form $(\mathbf{s}-\mathbf{t})[j]/2^{\ell-1}$ and these also need to be contained in the interval (8), but since we know that $\mathbf{s} \in \mathbb{Z}_q^n$ and we can assume that $\mathbf{t} \in \mathbb{Z}_q^n$ as was explained above, then certainly $(\mathbf{s}-\mathbf{t})[j]/2^{\ell-1}$ are in the interval (8).
We now work backwards by fixing a vector $\mathbf{t} \in \mathbb{Z}_q^n$, $\mathbf{t} \neq \mathbf{s}$, and estimate the probability that there is a vector $\mathbf{k} \in \mathbb{Z}^d$ such that $\mathbf{w} \in \mathbb{Z}^{n+d}$ formed from these, as in (6), can be the output of LLL-Babai with input $\mathbf{u}$ in the sense that for the first d differences (11) holds. As was explained above, this is automatic for the last n differences, so we do not need to worry about those. If a vector $\mathbf{a} \in \mathbb{Z}_q^n$ is chosen uniformly at random, then $\langle \mathbf{a}, \mathbf{s}-\mathbf{t}\rangle_q$ is distributed uniformly at random in $\mathbb{Z}_q$, so the probability that $\langle \mathbf{a}, \mathbf{s}-\mathbf{t}\rangle_q$ is in the interval (8) is

$$\frac{2\left\lfloor \left(1+2^{\mu(n+d)}\right)\sqrt{n+d}\,2^{r-\ell}\right\rfloor + 1}{q}\,. \tag{12}$$

So for the fixed vector $\mathbf{t}$, for each $j = 0,\dots,d-1$ independently, the probability that (11) holds is given by (12).

Lemma 1. *The probability that there is a vector $\mathbf{k} \in \mathbb{Z}^d$ such that $\mathbf{w} \in \mathbb{Z}^{n+d}$ formed from $\mathbf{t}$ and $\mathbf{k}$, as in* (6)*, can be the output of LLL-Babai with input $\mathbf{u}$ in the sense that all* (11) *hold is*

$$\leq \left[\frac{2\left\lfloor \left(1+2^{\mu(n+d)}\right)\sqrt{n+d}\,2^{r-\ell}\right\rfloor + 1}{q}\right]^d .$$

The probability is taken over the d vectors $\mathbf{a}_j$ chosen uniformly at random from $\mathbb{Z}_q^n$. □

Next we compute the probability that in addition to $\mathbf{v}$ there are no other vectors $\mathbf{w} \neq \mathbf{v}$ close enough to $\mathbf{u}$ for LLL-Babai to find them. More precisely, we compute the probability that in addition to $\mathbf{s}$, there are no other vectors $\mathbf{t} \neq \mathbf{s}$ that would yield a $\mathbf{w}$ (as in (6)) close enough to $\mathbf{u}$. There are $q^n - 1$ possible vectors $\mathbf{t} \neq \mathbf{s}$ for which the experiment of Lemma 1 can succeed or fail. Using Lemma 1, we immediately get the following result.

Lemma 2. *The probability that $\mathbf{v}$ is the only vector LLL-Babai can output is*

$$> 1 - \left(\frac{q^n-1}{q^d}\right)\left[2\left\lfloor \left(1+2^{\mu(n+d)}\right)\sqrt{n+d}\,2^{r-\ell}\right\rfloor + 1\right]^d ,$$

where the vectors $\mathbf{a}_j$ are chosen uniformly at random from $\mathbb{Z}_q^n$. □

All we need to do is to ensure that the probability in Lemma 2 is very large so that the vector returned by LLL-Babai with input $\mathbf{u}$ is almost certainly the correct vector $\mathbf{v}$, from which $\mathbf{s}$ can be read. To get a concrete result, we ask that this probability is bigger than $1 - 1/2^n$, which yields the inequality

$$2^n\left(\frac{q^n-1}{q^d}\right)\left[2\left\lfloor \left(1+2^{\mu(n+d)}\right)\sqrt{n+d}\,2^{r-\ell}\right\rfloor + 1\right]^d < 1\,.$$

A bit cleaner and just a tiny bit stronger is the inequality

$$2^{(r+1)n-rd}\left[\left(1+2^{\mu(n+d)}\right)\sqrt{n+d}\,2^{1+r-\ell} + 1\right]^d < 1\,. \tag{13}$$

To get an even simpler result, we instead ask that

$$2^{(r+1)n-rd}\left[2^{\mu(n+d)+2+r-\ell}\sqrt{n+d}\right]^{3d/2} < 1\,, \tag{14}$$

which implies (13).

Remark 1. The exponent $3d/2$ could be chosen to be significantly smaller. Namely, for large enough n the exponent can be taken to be any arbitrarily small number bigger than 1. We will discuss this later.

By taking logarithms in (14) we obtain

$$(r+1)n + \frac{rd}{2} + \frac{3d}{2}\left[\mu(n+d) + 2 - \ell + \log_2\sqrt{n+d}\right] < 0\,. \tag{15}$$

For the sake of getting a neat result, we approximate

$$2 + \log_2\sqrt{n+d} \le \varepsilon(n+d)\,, \quad \varepsilon = \frac{2+\log_2\sqrt{n}}{n}\,,$$

to get

$$\begin{aligned}&(r+1)n + \frac{rd}{2} + \frac{3d}{2}\left[\mu(n+d) + 2 - \ell + \log_2\sqrt{n+d}\right]\\ &\le (r+1)n + \frac{rd}{2} + \frac{3d}{2}\left[(\mu+\varepsilon)(n+d) - \ell\right] < 0\,.\end{aligned}$$

This simplifies into

$$3(\mu+\varepsilon)d^2 - \left[3\ell - r - 3(\mu+\varepsilon)n\right]d + 2(r+1)n < 0\,, \tag{16}$$

which is possible when the discriminant is positive:

$$\left[3\ell - r - 3(\mu+\varepsilon)n\right]^2 - 24(\mu+\varepsilon)(r+1)n > 0\,. \tag{17}$$

We assume that $3\ell - r - 3(\mu+\varepsilon)n > 0$, i.e. $\ell > r/3 + (\mu+\varepsilon)n$. In this case solving (17) and using $r > \ell$ yields

$$\ell > \frac{r}{3} + (\mu+\varepsilon)n + \sqrt{\frac{8}{3}(\mu+\varepsilon)(r+1)n}\,, \quad r > \left(\frac{9}{2} + 3\sqrt{2}\sqrt{1+\frac{1}{3(\mu+\varepsilon)n}}\right)(\mu+\varepsilon)n\,. \tag{18}$$

To get a nicer looking result, we use instead the bound

$$r > \frac{21}{2}(\mu+\varepsilon)n\,, \tag{19}$$

which implies the bound for r in (18). Write

$$r = \frac{21}{2}(\mu+\varepsilon)n + C \in O(n)\,,$$

where C is a constant, so $q \in 2^{O(n)}$. The optimal value for d is

$$d = \frac{3\ell - r - 3(\mu+\varepsilon)n}{6(\mu+\varepsilon)} < \frac{2r - 3(\mu+\varepsilon)n}{6(2+\log_2\sqrt{n})}\,n < \frac{3}{2}(\mu+\varepsilon)n^2 + \frac{Cn}{6} \in O(n^2)\,.$$

The last thing to check is that the bound (10) is indeed satisfied, but this follows easily from (15). We have now obtained an analogue of Theorem 1.

Theorem 3. *Let $\varepsilon = \left(2+\log_2\sqrt{n}\right)/n$ and suppose*

$$r > \frac{21}{2}(\mu+\varepsilon)n\,, \quad \ell > \frac{r}{3} + (\mu+\varepsilon)n + \sqrt{\frac{8}{3}(\mu+\varepsilon)(r+1)n}\,, \quad d = \left\lceil \frac{3\ell - r - 3(\mu+\varepsilon)n}{6(\mu+\varepsilon)} \right\rfloor.$$

Then $\mathrm{GHNP}_{n,r,\ell,d}$ can be solved in probabilistic polynomial time in n. A value of $\mu = 1/2$ is guaranteed to work so that the algorithm succeeds with probability at least $1 - 1/2^n$.

Proof. LLL-Babai finds the approximate closest vector in the $(n+d)$-dimensional lattice $\Lambda_{n,r,\ell,d}$ in polynomial time in $n + d \in O(n^2)$. By the arguments above, if r and ℓ satisfy the given (loose) bounds, we can expect the vector given by LLL-Babai to be good enough to recover $\mathbf{s}$ with probability at least $1 - 1/2^n$. According to Theorem 2, LLL-Babai is guaranteed to return the closest vector up to an approximating factor with $\mu = 1/2$, although in practice significantly better performance, i.e. smaller μ, can be expected. □

As was mentioned in Remark 1, the exponent $3d/2$ in (14) can be taken to be any arbitrarily small number bigger than 1 as long as n is large enough. We consider now the extreme case where the exponent is taken to be 1. Then instead of (16) we obtain

$$(\mu+\varepsilon)d^2 - [\ell - (\mu+\varepsilon)n]\, d + (r+1)n < 0\,.$$

The discriminant must be positive, which instead of (18) yields

$$\ell > (\mu+\varepsilon)n + 2\sqrt{(\mu+\varepsilon)(r+1)n}\,, \qquad r > \left(4+\sqrt{15}\sqrt{1+\frac{4}{15(\mu+\varepsilon)n}}\right)(\mu+\varepsilon)n\,.$$

When n is large enough, it suffices to take for example $r > 8(\mu+\varepsilon)n$. In this case $d = O(n)$. As was mentioned earlier, a choice of $\mu = 1/2$ is guaranteed to work [Bab86], but if the parameters of LLL are chosen appropriately, then in fact $\mu \approx 1/4$ will work as long as n is large enough. This means that $r > 2n$ should work when n is large enough.

3 Key Recovery for LWE

In this section we apply Theorem 3 to attack *search*-LWE.

Definition 4 (*search*-LWE). *Let n be a security parameter, $q(n)$ a prime integer modulus, $r := \log_2 q$, and χ an error distribution over $\mathbb{Z}_q$. Let $\mathbf{s} \in \mathbb{Z}_q^n$ be a fixed secret vector chosen uniformly at random. Given access to d samples of the form*

$$\left(\mathbf{a}, \left[\langle \mathbf{a}, \mathbf{s}\rangle + e\right]_q\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_q\,,$$

where $\mathbf{a} \in \mathbb{Z}_q^n$ are chosen uniformly at random and e are sampled from the error distribution χ, the problem search-*$LWE_{n,r,d,\chi}$ is to recover $\mathbf{s}$.*

Definition 5 (*decision*-LWE). *Let n be a security parameter, $q(n)$ a prime integer modulus, $r := \log_2 q$, and χ an error distribution over $\mathbb{Z}_q$. Let $\mathbf{s} \in \mathbb{Z}_q^n$ be a fixed secret vector chosen uniformly at random. Given a vector $\mathbf{t} \in \mathbb{Z}_q^d$, the problem* decision-*$LWE_{n,r,d,\chi}$ is to distinguish with some non-negligible advantage whether $\mathbf{t}$ is of the form*

$$\mathbf{t} = \left[\, [\langle \mathbf{a}_0, \mathbf{s}\rangle + e_0]_q\,, \ldots, [\langle \mathbf{a}_{d-1}, \mathbf{s}\rangle + e_{d-1}]_q \right]\,,$$

where $\mathbf{a}_i \in \mathbb{Z}_q^n$ are chosen uniformly at random and e_i are sampled from the error distribution χ, or whether $\mathbf{t}$ is sampled uniformly at random from $\mathbf{Z}_q^d$.

In practice, the distribution χ is always taken to be a *discrete Gaussian distribution* $D_{\mathbb{Z},\sigma}$. This is the probability distribution over $\mathbb{Z}$ that assigns to an integer x a probability

$$\Pr(x) \propto \exp\left(-\frac{x^2}{2\sigma^2}\right)\,,$$

where σ is the standard deviation. It is efficient, but non-trivial, to sample from such a distribution up to any level of precision [GPV08].

Theorem 4 ([Reg09]). *If* decision-*LWE can be solved with exponentially good advantage in polynomial time and $q = \mathrm{poly}(n)$, then* search-*LWE can be solved in polynomial time.*

Remark 2. Theorem 4 fails for exponentially large moduli. Even if the distinguishing advantage was perfect and takes only polynomial time, the time to recover the secret $\mathbf{s}$ would be proportional to nq, so it would be still exponential in n.

The main result of [Reg09] was that when $q = \mathrm{poly}(n)$ LWE can be proven to be hard in the following sense.

Theorem 5 ([Reg09]). *If $q = poly(n)$, $\sigma > \sqrt{n/(2\pi)}$ and $d = poly(n)$, then there exists a polynomial time quantum reduction from worst-case $GapSVP_{\widetilde{O}(nq/\sigma)}$ to* decision-*$LWE_{n,r,d,D_{\mathbb{Z},\sigma}}$.*

For exponentially large q the following classical reduction can be used.

Theorem 6 ([Pei09]). *If $q \geq 2^{n/2}$, $\sigma > \sqrt{n/(2\pi)}$ and $d = poly(n)$, then there exists a polynomial time classical reduction from worst-case GapSVP$_{\widetilde{O}(nq/\sigma)}$ to* search-*LWE$_{n,r,d,D_{\mathbb{Z},\sigma}}$.*

Remark 3. For smaller values of q security can be based on a classical reduction to an easier and less studied *decision* lattice problem $\mathrm{GapSVP}_{\zeta,\gamma}$ [Pei09].

Remark 4. In practical applications to cryptography the standard deviation σ is often taken to be so small that the reductions do not apply.

To find the LWE secret $\mathbf{s}$ directly using Theorem 3 we need a way to read $\mathrm{MSB}_\ell\left(\langle \mathbf{a}, \mathbf{s}\rangle_q\right)$ from $\left[\langle \mathbf{a}, \mathbf{s}\rangle + e\right]_q$. If σ is small enough and ℓ big enough, this is likely to be possible by simply reading the ℓ most significant bits of $\left[\langle \mathbf{a}, \mathbf{s}\rangle + e\right]_q$ since adding e is unlikely to change them. It is not hard to bound the value ℓ that a particular σ permits (with high probability), but we will instead take a different approach by slightly modifying the proof of Theorem 3. Instead of taking u_i to be the MSB_ℓ parts of the inner products in the LWE samples as in (3), simply take

$$u_i = \left[\langle \mathbf{a}_i, \mathbf{s}\rangle + e_i\right]_q \tag{20}$$

from the LWE samples and form the vector $\mathbf{u}$ just as in (5):

$$\mathbf{u} = \left[u_0, u_1, \ldots, u_{d-1}, 0, \ldots, 0\right] \in \mathbb{R}^{n+d}\,. \tag{21}$$

If the standard deviation σ is so small that the absolute values of e_i are very unlikely to be larger than $2^{r-\ell}$, we can form the vector $\mathbf{v}$ as in (2) and obtain inequalities

$$|\langle \mathbf{a}_i, \mathbf{s}\rangle_q - u_i| < 2^{r-\ell}$$

as in (4), and the rest of the proof goes through without change.
One detail was ignored above. For the argument to work, we need

$$\left[\langle \mathbf{a}_i, \mathbf{s}\rangle + e_i\right]_q = \langle \mathbf{a}_i, \mathbf{s}\rangle_q + e_i\,.$$

In applications of LWE to cryptography this is typically needed for decryption to work correctly. Since the errors are assumed to be small, the probability of this not being true is extremely small. To make things simpler, we assume this to be the case for all LWE samples, although adding it as an additional probabilistic condition would be very easy.

Definition 6. *For all LWE samples in Definitions 4 and 5 we assume*

$$\left[\langle \mathbf{a}, \mathbf{s}\rangle + e\right]_q = \langle \mathbf{a}, \mathbf{s}\rangle_q + e\,.$$

To connect ℓ to the standard deviation σ, we need to know something about the mass of the distribution $D_{\mathbb{Z},\sigma}$ that lies outside the interval $\left(-2^{r-\ell}, 2^{r-\ell}\right)$.

Lemma 3 ([Ban93]). *Let $B \geq \sigma$. Then*

$$\Pr\left[|D_{\mathbb{Z},\sigma}| \geq B\right] \leq \frac{B}{\sigma}\exp\left(\frac{1}{2} - \frac{B^2}{2\sigma^2}\right)\,.$$

According to Lemma 3, the probability that the error has absolute value at least $2^{r-\ell}$ is

$$\leq \sigma^{-1}\,2^{r-\ell}\exp\left(\frac{1}{2} - \frac{2^{2r-2\ell-1}}{\sigma^2}\right)\,.$$

Of course in practice we want the probability of this happening for none of the d samples to be very close to 1.

Lemma 4. *The top ℓ bits of $\langle \mathbf{a}_i, \mathbf{s}\rangle_q$ can be read correctly from all d LWE samples with probability at least*

$$\left[1 - \sigma^{-1}\,2^{r-\ell}\exp\left(\frac{1}{2} - \frac{2^{2r-2\ell-1}}{\sigma^2}\right)\right]^d\,.$$

Now we take ℓ to be the lower bound in Theorem 3 to obtain our main result.

Theorem 7. *Let* $\varepsilon = \left(2 + \log_2 \sqrt{n}\right)/n$ *and suppose* $r > (21/2)(\mu + \varepsilon)n$. *Let*

$$\ell = \left\lceil \frac{r}{3} + (\mu+\varepsilon)n + \sqrt{\frac{8}{3}(\mu+\varepsilon)(r+1)n} \right\rceil, \qquad d = \left\lceil \frac{3\ell - r - 3(\mu+\varepsilon)n}{6(\mu+\varepsilon)} \right\rfloor.$$

Then search-$LWE_{n,r,d,D_{\mathbb{Z},\sigma}}$ *can be solved in probabilistic polynomial time in* n. *A value of* $\mu = 1/2$ *is guaranteed to work so that the algorithm succeeds with probability at least*

$$\left(1 - \frac{1}{2^n}\right)\left[1 - \sigma^{-1}\, 2^{r-\ell} \exp\left(\frac{1}{2} - \frac{2^{2r-2\ell-1}}{\sigma^2}\right)\right]^d.$$

□

Of course the discussion after Theorem 3 applies here also, meaning that success can (roughly speaking) be guaranteed in the sense of Theorem 7 when n is large enough, $r > 2n$ and d is chosen appropriately.

Remark 5. It is important to understand that Theorem 7 does not contradict Theorem 6, because even if σ is large enough for the reduction to apply, for large q it is entirely plausible that $\mathrm{GapSVP}_{\tilde{O}(nq/\sigma)}$ is easy.

4 Practical Performance

In the proofs of Theorems 3 and 7 we performed several very crude estimates to obtain a provably polynomial running time with high probability. In practice we can of course expect the attack to perform significantly better than Theorem 7 suggests. In this section we try to get an idea of what can be expected to happen in practice.

The estimate in (4) is very crude on average. In the proof of Theorem 7 the differences $\left|\langle \mathbf{a}_i, \mathbf{s}\rangle_q - u_i\right|$ are exactly equal to the absolute values of the errors e_i, which are distributed according to $D_{\mathbb{Z},\sigma}$. If instead of using the rows of the matrix (1) we use the rows of

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

to generate the lattice $\Lambda_{n,r,\ell,d}$, the expectation value of $||\mathbf{v} - \mathbf{u}||^2$ is

$$\leq d\,\mathbb{E}\left[D_{\mathbb{Z},\sigma}^2\right] + n\lceil\sigma\rceil^2 = d\left(\sigma^2 + \mathbb{E}\left[D_{\mathbb{Z},\sigma}\right]^2\right) + n\lceil\sigma\rceil^2 \leq (n+d)\lceil\sigma\rceil^2,$$

so we can expect the distance $||\mathbf{v} - \mathbf{u}||$ to be bounded from above by $\sqrt{n+d}\,\lceil\sigma\rceil$.

Another significant improvement to the running time is to define an $(n+d) \times d$ matrix

$$\begin{pmatrix}
q & 0 & 0 & \cdots & 0 \\
0 & q & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & q \\
\mathbf{a}_0[0] & \mathbf{a}_1[0] & \mathbf{a}_2[0] & \cdots & \mathbf{a}_{d-1}[0] \\
\mathbf{a}_0[1] & \mathbf{a}_1[1] & \mathbf{a}_2[1] & \cdots & \mathbf{a}_{d-1}[1] \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\mathbf{a}_0[n-1] & \mathbf{a}_1[n-1] & \mathbf{a}_2[n-1] & \cdots & \mathbf{a}_{d-1}[n-1]
\end{pmatrix} \tag{22}$$

and let $\mathbf{A}$ be its $d \times d$ row-Hermite normal form, i.e. $\mathbf{A}$ is a triangular matrix whose rows generate the same $\mathbb{Z}$-module as the rows of the matrix (22). Let Λ be the full d-dimensional lattice generated by the rows of $\mathbf{A}$. As before, let $u_i = \left[\langle \mathbf{a}_i, \mathbf{s}\rangle + e_i\right]_q$ and set

$$\mathbf{u} = \left[u_0, u_1, \ldots, u_{d-1}\right] \in \mathbb{R}^d. \tag{23}$$

Now use LLL-Babai to find a vector close to $\mathbf{u}$ in the lattice Λ. This recovers a vector which is an integral linear combinations of the rows of $\mathbf{A}$. Simply express this in the original basis, i.e. in terms of the rows of the matrix (22), to recover a candidate for $\mathbf{s}$ as the coefficients of the last n rows. This is the approach that we will work with for the rest of this paper.
In this case we use

$$\mathbf{v} = \left[\langle \mathbf{a}_0, \mathbf{s}\rangle_q, \langle \mathbf{a}_1, \mathbf{s}\rangle_q, \langle \mathbf{a}_2, \mathbf{s}\rangle_q, \ldots, \langle \mathbf{a}_{d-1}, \mathbf{s}\rangle_q\right]$$

and find that the expected distance squared $||\mathbf{v} - \mathbf{u}||^2$ is

$$d\,\mathbb{E}\left[D^2_{\mathbb{Z},\sigma}\right] = d\left(\sigma^2 + \mathbb{E}\left[D_{\mathbb{Z},\sigma}\right]^2\right) = d\sigma^2\,,$$

so that the expected distance $||\mathbf{v} - \mathbf{u}||$ is $\sigma\sqrt{d}$.
A straightforward modification of the calculation yielding (13) shows that to succeed with probability at least p we can expect to need

$$\log_2(1-p) + r(d-n) > d\log_2\left[2\left(1 + 2^{\mu d}\right)\sigma\sqrt{d} + 1\right]\,. \tag{24}$$

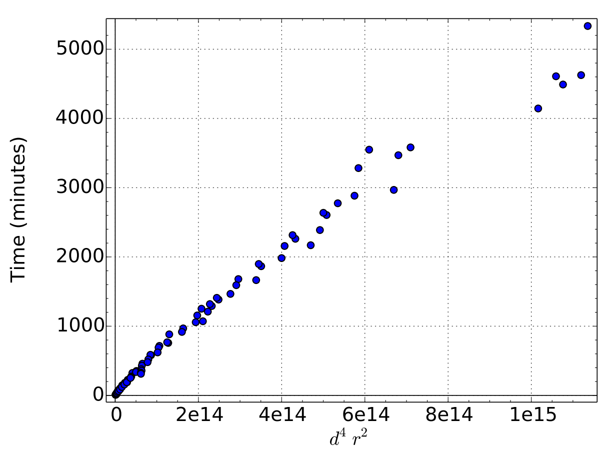
Remark 6. Instead of asking for a high success probability, we might only want to ask to succeed with some positive probability, in which case we take $p = 0$.

4.1 Successful Attacks

All experiments described in the rest of this paper are examples of our key recovery attack run for varying parameter sets. All attacks were run on a 2.6 GHz AMD Opteron 6276 using the floating point variant of LLL [NS06] in PARI/GP [PARI2]. All LWE samples were generated using the LWE oracle implementation in SAGE.
These experiments are intended to demonstrate the key points about our key recovery attack:

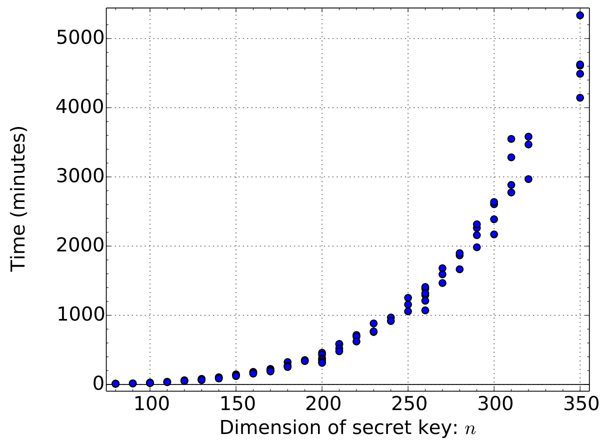
1. The time required to recover the secret key is roughly the running time of LLL, which has been estimated in [NS06] to be approximately $O(d^4r^2)$, where d is the dimension of the lattice and $r := \log_2 q$. This prediction approximates very closely the running time of the attack in practice, which is shown very clearly by the roughly linear graph in Figure 1 when the running time is plotted against d^4r^2.

Fig. 1: Timings for Key Recovery Attacks ($\sigma = 8/\sqrt{2\pi}$, $p = 0$)



2. The attack is practical in the sense that even running on a single machine, an instance of LWE with $n = 350$ can be successfully attacked in roughly 3.5 days. Figure 2 shows the running time of the attack (in minutes) for various n up to size 350.

Fig. 2: Timings for Key Recovery Attacks ($\sigma = 8/\sqrt{2\pi}$, $p = 0$)



3. The range of LWE parameters which can be successfully attacked via this polynomial time key recovery attack depends very intimately on the approximation factor $2^{\mu d}$ in the LLL-Babai algorithm (LLL followed by Babai's nearest planes method). Theorem 2 ([Bab86]) only guarantees $\mu \leq 1/2$, but in practice significantly smaller μ can be expected. Any improvement to the approximation factor in the LLL-Babai algorithm will have a direct and significant impact on which LWE parameters are attackable in polynomial time.
4. Our attack gives an indirect way to measure the effective value of μ in the approximation factor $2^{\mu d}$ of LLL-Babai: Because we can predict whether our attack will succeed or fail fairly accurately based on the value of μ, we can run it on various parameter sets and test whether the secret key was successfully recovered or not. Because the attack is extremely efficient we can run it hundreds of times, for varying parameters, thereby observing effective bounds on μ. We have run these experiments and the results are show in Figure 3. The green dots represent attacks which succeeded, thereby indicating that the effective approximation factor was no more than the plotted value. The red dots represent key recovery attacks which failed. These dots indicate a strong likelihood that for each key dimension n the effective value of μ in the approximation factor lies somewhere between the adjacent green and red dots, although this boundary is fuzzy due to probabilistic effects.

More specifically, to measure the practical performance of LLL-Babai and consequently of the polynomial time key recovery attack, we define a function which is an expression for μ derived from the formula for the likelihood that the attack will succeed (Equation 24):

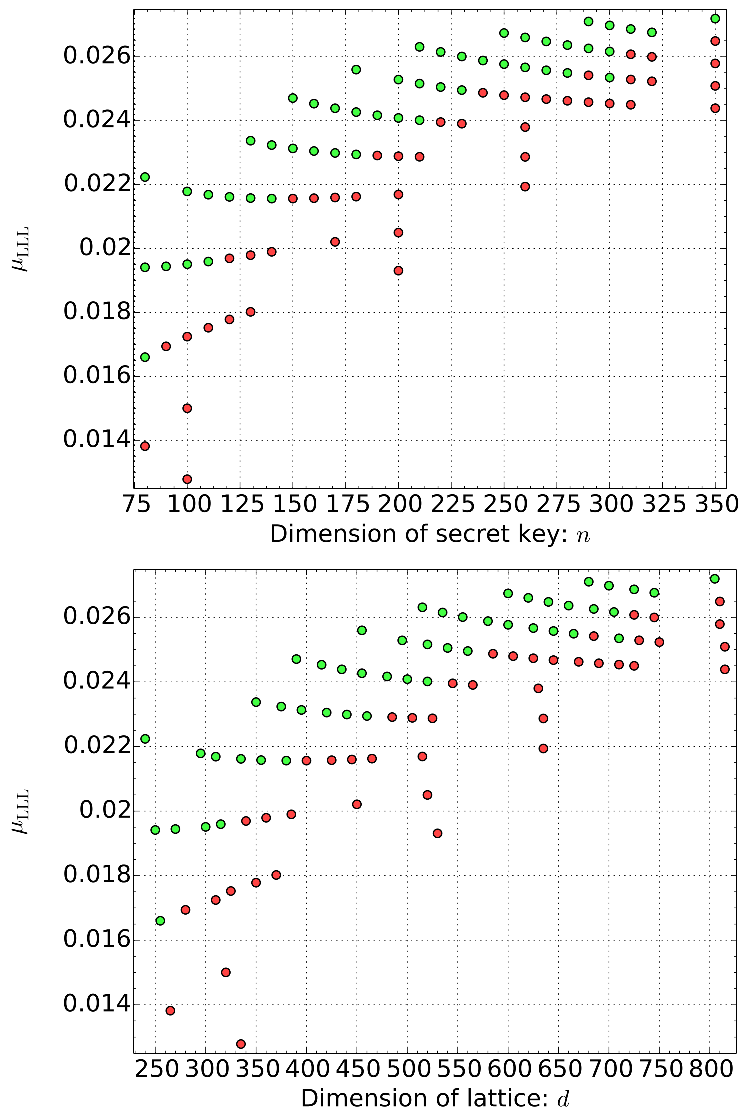
$$\mu_{\text{LLL}}(n, r, d, \sigma, p) := \frac{1}{d} \log_2 \left[\frac{(1-p)^{1/d}\, 2^{r(1-n/d)} - 1}{2\sigma\sqrt{d}} - 1 \right] ,$$

This function measures the effective performance of LLL-Babai in the sense that for an attack to succeed with probability at least p we can expect to need $\mu \leq \mu_{\text{LLL}}$ in the approximation factor $2^{\mu d}$.

The graphs in Figure 3 show a relatively clear boundary in the values of μ_{LLL} between failed and succeeded attacks, which can then be extrapolated to bigger examples. We present the values μ_{LLL} as functions of both and n and d, where d is the dimension of the lattice Λ for which LLL was performed. A green dot indicates that the attack succeeded (correct $\mathbf{s}$ was recovered) and a red dot that the attack failed (incorrect $\mathbf{s}$ was recovered).

The dimension d of course affects μ_{LLL} very strongly, so we want to choose it in an optimal way given all the other parameters, i.e. in a way that maximizes μ_{LLL}. We let d_{opt} be such that $\partial_d \mu_{\text{LLL}}(n, r, d_{\text{opt}}, \sigma, p) = 0$ (rounded to an integer). Then parameter selection in all of the attacks we performed was done by taking $d \approx d_{\text{opt}}$. For a particular value of n the experiments differ mainly in the choice of r, and $d \approx d_{\text{opt}}$ is always computed case-by-case. It is not hard to see

Fig. 3: Effective approximation constant μ in LLL-Babai algorithm ($\sigma = 8/\sqrt{2\pi}$, $p = 0$)



that when the example size increases, the value d_{opt} approaches $2n$. It is also worth pointing out that the effect of p on μ_{LLL} is very small unless p is extremely close to 1.
In Table 1 we show more details of the experiments in Figure 3 that lie at the boundary of succeeding and failing. In all these experiments q is taken to be the smallest prime larger than some power of 2, so the value of r given is a very close approximation but not the exact value.

4.2 Practical Key Recovery

In practice, key recovery in polynomial time can be performed as follows. The LWE problem determines n, r and σ. Now find d_{opt} and see if the corresponding μ_{LLL} is small enough for there to be a chance for the attack to succeed. This can be done e.g. by extrapolating the boundary from Figure 3. For performance reasons you might want to decrease d to be as small as possible so that the attack can still be expected to succeed based on the value of μ_{LLL}. Now observe d

Table 1: Key recovery attacks and running times (in minutes) ($\sigma = 8/\sqrt{2\pi}$, $p = 0$)

	Succeeded				**Failed**			
n	$\log_2 q$	d	μ_{LLL}	Time (min)	$\log_2 q$	d	μ_{LLL}	Time (min)
80	16	255	0.016602	10	15	265	0.013818	9
90	18	270	0.019443	16	17	280	0.016941	15
100	19	300	0.019510	25	18	310	0.017245	24
110	20	315	0.019594	37	19	325	0.017523	33
120	22	340	0.021610	54	21	350	0.019680	56
130	23	355	0.021578	70	22	360	0.019792	68
140	24	380	0.021563	98	23	385	0.019898	86
150	26	395	0.023131	135	25	400	0.021563	121
160	27	420	0.023050	173	26	425	0.021575	157
170	28	440	0.022990	213	27	445	0.021597	190
180	29	460	0.022944	263	28	465	0.021624	252
190	31	480	0.024169	353	30	485	0.022911	338
200	32	500	0.024085	430	31	505	0.022887	379
210	33	520	0.024014	520	32	525	0.022871	480
220	35	540	0.025052	691	34	545	0.023956	621
230	36	560	0.024956	758	35	565	0.023906	767
240	38	580	0.025882	968	37	585	0.024872	917
250	39	600	0.025769	1155	38	605	0.024798	1057
260	40	625	0.025667	1409	39	625	0.024733	1291
270	41	645	0.025576	1592	40	645	0.024674	1466
280	42	665	0.025493	1898	41	670	0.024623	1665
290	44	685	0.026260	2315	43	685	0.025418	2158
300	44	710	0.025350	2388	43	710	0.024537	2169
310	47	725	0.026867	3549	44	725	0.024498	2775
320	48	745	0.026762	3582	47	745	0.025996	3469
350	52	805	0.027193	5335	51	810	0.026491	4626

LWE samples, form the matrix (22), find the row-Hermite normal form $\mathbf{A}$, form the lattice Λ generated by the rows $\mathbf{A}$ and use LLL-Babai to find the closest lattice point to $\mathbf{u}$ (as in (23)), express the closest vector in terms of the original basis (rows of (22)) and read the last n entries to find $\mathbf{s}$.

5 Security Implications

Key recovery for LWE in polynomial time is only possible when the modulus q is very large and σ is very small. Such large moduli (relative to n) do not appear in any recommended parameter ranges [Reg09, LP11, vdPS13, LN14]. These recommendations are typically calculated so that the *distinguishing attack* [MR09, LP11] is infeasible, which gives good reason to believe that also key recovery is infeasible. However, in applications to homomorphic cryptography, where particularly deep circuits need to be homomorphically evaluated, parameters with very large q and small σ might be desirable and ciphertext indistinguishability might be irrelevant. In these cases it is of course particularly important to understand exactly how the polynomial time key recovery attack performs.

We now recall how the distinguishing attack works. Suppose we have observed $d \geq n$ LWE samples, computed the matrix $\mathbf{A}$ and formed the lattice Λ. By Λ^* we denote the dual of Λ, which is the lattice

$$\Lambda^* = \{\mathbf{y} \mid \langle \mathbf{y}, \mathbf{x} \rangle \in \mathbb{Z} \text{ for every } \mathbf{x} \in \Lambda\} \ .$$

Definition 7. *The* root-Hermite factor *of a basis* $B = \{\mathbf{b}_0, \ldots, \mathbf{b}_{d-1}\}$ *of a d-dimensional lattice* Λ *is the number*

$$\delta_B := \left[\frac{||\mathbf{b}_0||}{|\det \Lambda|^{1/d}} \right]^{1/d} ,$$

where $||\cdot||$ *denotes the usual* ℓ_2*-norm.*

The root-Hermite factor seems to be a very efficient way of measuring the quality of a basis (smaller is better). In [NS06] it is argued that with LLL one can expect to find a basis with root-Hermite factor of approximately 1.02 for a random lattice, but for the lattices in our examples the root-Hermite factors of the LLL reduced bases are typically $\delta_B \approx 2^{rn/d^2} < 1.02$.

In the distinguishing attack the goal is to attack *decision*-LWE (Definition 5, i.e. to distinguish with some non-negligible advantage if a vector $\mathbf{u} \in \mathbb{Z}_q^d$ is sampled as in (23) or sampled uniformly at random from $\mathbb{Z}_q^d$. Let $\mathbf{e} \in \mathbb{Z}^d$ be a vector formed from the d errors e_i. If we can find a short integer vector $\mathbf{y} \in q\Lambda^*$ that is non-zero modulo q, then

$$\langle \mathbf{y}, \mathbf{u} \rangle \equiv \langle \mathbf{u}, \mathbf{A}^\top \mathbf{s} \rangle + \langle \mathbf{y}, \mathbf{e} \rangle \equiv \langle \mathbf{y}, \mathbf{e} \rangle \pmod{q} ,$$

so if $\mathbf{y}$ is very short, then $\langle \mathbf{y}, \mathbf{u} \rangle \pmod{q}$ can be expected to be short and distributed as a sample from a Gaussian with standard deviation $||\mathbf{y}||\,\sigma$. To be able to distinguish with high advantage whether the sample is an honest LWE sample, we need $||\mathbf{y}||\,\sigma/q$ to be small enough. If the vector $\mathbf{u}$ is instead chosen uniformly at random from $\mathbb{Z}_q^d$, the inner product can be expected to be uniformly distributed modulo q. If the inner product modulo q is within $(-q/4, q/4)$, the attacker guesses that $\mathbf{u}$ comes from d LWE samples. The distinguishing advantage is (see [LP11])

$$\exp\Big[-\pi \Big(\frac{||\mathbf{y}||\,\sqrt{2\pi}\,\sigma}{q} \Big)^2 \Big] .$$

If the attacker can compute a basis B^* for $q\Lambda^*$ with root-Hermite factor δ_{B^*}, then they can get a distinguishing advantage of at least

$$\exp\Big[-\pi \Big(\frac{\delta_{B^*}^d \,|\det \Lambda^*|^{1/d}\,\sqrt{2\pi}\,\sigma}{q} \Big)^2 \Big] \approx \exp\Big[-\pi \Big(\frac{\delta_{B^*}^d \,\sqrt{2\pi}\,\sigma}{q^{1-n/d}} \Big)^2 \Big] , \tag{25}$$

where we used that $|\det \Lambda^*| \approx q^n$ (see [vdPS13]). In this case the optimal choice for the dimension is $d = \sqrt{nr/\log_2 \delta_{B^*}}$ [MR09]. If q is very large, according to (25) it suffices to find a basis B^* with fairly large root-Hermite factor and still expect a reasonable distinguishing advantage. In [MR09, LP11, vdPS13, LN14] the authors attempt to estimate the computational effort needed for this using various methods. It is conjectured in [GN08] that the computational complexity of lattice reduction using the BKZ algorithm[5] depends mostly on the achieved root-Hermite factor and not so much on the dimension of the lattice. The BKZ simulation algorithm of [CN11] can be used to estimate the performance of BKZ with large blocksize (≥ 50), which is needed to achieve very small root-Hermite factors, and this was used in the recent analysis [LN14] to give updated estimates for the security of LWE.

A series of papers presenting applications of homomorphic encryption ([LNV, GLN, BLN, LLN]) give recommended parameter sizes for RLWE based on the distinguishing attack as explained above. For homomorphic computation, it is often necessary to ensure that q is very large in order to allow for correct decryption after evaluating a deep circuit requiring many homomorphic multiplications. For example, [GLN] recommend two parameter sets for simple machine learning tasks to ensure 80 bits of security, $(n, q) = (4096, 2^{128})$ and $(n, q) = (8192, 2^{340})$, and [BLN] suggests in addition $(n, q) = (2^{14}, 2^{512})$ for evaluating the logistical regression function.

Example 1. In [LN14], estimates for an upper bound on the modulus q that provides 80 bits of security are given (complexity of BKZ-2.0 needed to achieve a small enough root-Hermite factor) with distinguishing advantage 2^{-80} and standard deviation $\sigma = 8/\sqrt{2\pi}$. Their results are presented in Table 2.

LWE parameters with such small q are certainly not vulnerable to our polynomial time key recovery attack. For example, for $n = 1024$ the smallest vulnerable r might be around 140, which is much larger than the upper bound cited in Example 1. Of course these results are not exactly comparable since the attacks are of such a different nature.

[5] The state-of-the-art adaption of BKZ is the algorithm BKZ-2.0 by Chen and Nguyen [CN11].

Table 2: Bounds on $r = \log_2 q$ for 80 bits of security against 2^{-80} distinguishing advantage ($\sigma = 8/\sqrt{2\pi}$)

n	1024	2048	4096	8192	16384
$r \leq$	47.5	95.4	192.0	392.1	799.6

Example 2. For $n = 1024$ and $r \approx 140$, with a lattice basis with root-Hermite factor $\delta_{B^*} \approx 1.02$ which is attainable with LLL for a random lattice according to [NS06], the distinguishing advantage is near perfect according to (25). This means that distinguishing is very easy in these cases, and here we estimate that with our attack we also recover the key in polynomial time.

Example 3. In [LN14], homomorphic evaluation of encryption and decryption circuits for block ciphers is discussed and two homomorphic encryption schemes are compared, `FV` and `YASHE`. As soon as one wishes to perform more than one multiplication, the lower bound on r increases significantly. For example, using `FV`, to be able to do 10 homomorphic multiplications with $n = 1024$ one needs to have $r \geq 229$ to ensure correct decryption. When n is increased the required lower bound for r does increase, but not by as much as would be required for polynomial time key recovery to work. For example, it suffices to take $n \geq 4096$ for homomorphic-`FV` with 10 multiplications to be safe against a polynomial time attack.

According to Example 2 and the comment after Example 1, one way to view the polynomial time key recovery attack is as follows. When q is exponential in n and σ is small, then distinguishing can be done reliably in polynomial time and contrary to what Theorem 4 and Remark 2 suggest, also key recovery can be done in polynomial time.

6 Conclusion

We conclude by presenting some estimates of parameters vulnerable to polynomial time key recovery. Again we use $\sigma = 8/\sqrt{2\pi}$. Based on our experiments (e.g. Figure 3), we suggest some upper bounds for μ_{LLL} with $d = d_{\text{opt}}, p = 0$ that can be expected to be realized for large n. Solving for a lower bound for r from (24) and extrapolating the observations of Figure 2 yields the following results.

Table 3: Estimated smallest r vulnerable to polynomial time key recovery ($\sigma = 8/\sqrt{2\pi}$)

n	$\mu_{\text{LLL}} <$ (est.)	$r >$	Time (est.)
512	0.030	76	25 Days
1024	0.032	147	4 Years
2048	0.035	303	275 Years
4096	0.038	640	19700 Years

We generalized the Hidden Number Problem to higher dimensions and showed how it also admits a polynomial time solution, provided that certain restrictions on the parameters are met. Next we modified the proof to also show a key recovery attack on LWE and found that when the modulus q is large enough and the standard deviation σ of the error distribution is small enough, the LWE secret $\mathbf{s} \in \mathbb{Z}_q^n$ can be recovered in time poly(n) when the number of samples is chosen appropriately. Many examples of successful attacks were presented and used to estimate the effective approximation constant in the LLL algorithm.

References

[Bab86] L. Babai, *On Lovász' lattice reduction and the nearest lattice point problem*, Combinatorica **6** (1986), Issue 1, pp. 1-13.

[Ban93] W. Banaszczyk, *New bounds in some transference theorems in the geometry of numbers*, Mathematische Annalen **296**, no. 1 (1993), pp. 625-635.

[BCNS14] J. Bos, C. Costello, M. Naehrig, D. Stebila, *Post-quantum key exchange for the TLS protocol from the ring learning with errors problem*, IACR Cryptology ePrint Archive Report 2014/599, 2014.

[BLN] J. Bos, K. Lauter, M. Naehrig. Private Predictive Analysis on Encrypted Medical Data, Journal of Biomedical Informatics (2014) DOI 10.1016/j.jbi.2014.04.003

[BGV12] Z. Brakerski, C. Gentry, V. Vaikuntanathan, *(Leveled) fully homomorphic encryption without bootstrapping*, In Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, pp. 309-325, ACM, 2012.

[BV14] Z. Brakerski, V. Vaikuntanathan, *Efficient fully homomorphic encryption from (standard) LWE*, SIAM Journal on Computing **43**, no. 2 (2014), pp. 831-871.

[BV11] Z. Brakerski, V. Vaikuntanathan, *Fully homomorphic encryption from ring-LWE and security for key dependent messages*, In Advances in Cryptology–CRYPTO 2011, pp. 505-524, Springer Berlin Heidelberg, 2011.

[BV96] D. Boneh, R. Venkatesan, *Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes*, Advances in Cryptology—CRYPTO'96, pp. 129-142, Springer Berlin Heidelberg, 1996.

[Bra12] Z. Brakerski, *Fully homomorphic encryption without modulus switching from classical GapSVP*, In Advances in Cryptology–CRYPTO 2012, pp. 868-886, Springer Berlin Heidelberg, 2012.

[CN11] Y. Chen, P. Nguyen, *BKZ 2.0: Better lattice security estimates*, In Advances in Cryptology–ASIACRYPT 2011, pp. 1-20, Springer Berlin Heidelberg, 2011.

[GN08] N. Gama, P. Nguyen, *Predicting lattice reduction*, In Advances in Cryptology–EUROCRYPT 2008, pp. 31-51, Springer Berlin Heidelberg, 2008.

[GPV08] C. Gentry, C. Peikert, V. Vaikuntanathan, *Trapdoors for hard lattices and new cryptographic constructions*, In Proceedings of the fortieth annual ACM symposium on Theory of computing, pp. 197-206. ACM, 2008.

[GLN] T. Graepel, K. Lauter, M. Naehrig. ML Confidential: Machine Learning on Encrypted Data, International Conference on Information Security and Cryptology – ICISC 2012, Lecture Notes in Computer Science **7839**, pages 1–21. Springer Verlag, December 2012.

[LLN] Kristin Lauter, Adriana Lopez-Alt, Michael Naehrig. Private Computation on Encrypted Genomic Data. LatinCrypt 2014 (GenoPri 2014).

[LNV] K. Lauter, M. Naehrig, V. Vaikuntanathan. Can Homomorphic Encryption Be Practical? CCSW 2011, ACM Cloud Computing Security Workshop 2011.

[LLL82] A. Lenstra, H. Lenstra, L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. **261** (1982), no. 4, pp. 515-534.

[LN14] T. Lepoint, M. Naehrig, *A comparison of the homomorphic encryption schemes FV and YASHE*, In Progress in Cryptology–AFRICACRYPT 2014, pp. 318-335, Springer International Publishing, 2014.

[LN13] M. Liu, P. Nguyen, *Solving BDD by enumeration: An update*, In Topics in Cryptology–CT-RSA 2013, pp. 293-309, Springer Berlin Heidelberg, 2013.

[LP11] R. Lindner, C. Peikert, *Better key sizes (and attacks) for LWE-based encryption*, In Topics in Cryptology–CT-RSA 2011, pp. 319-339, Springer Berlin Heidelberg, 2011.

[LPR13] V. Lyubashevsky, C. Peikert, O. Regev, *On ideal lattices and learning with errors over rings*, Journal of the ACM (JACM) **60**, no. 6 (2013), 43.

[MR09] D. Micciancio, O. Regev, *Lattice-based cryptography*, In Post-quantum cryptography, pp. 147-191, Springer Berlin Heidelberg, 2009.

[NS06] P. Nguyen, D. Stehlé, *LLL on the average*, Algorithmic Number Theory, pp. 238-256, Springer Berlin Heidelberg, 2006.

[PARI2] The PARI Group, *PARI/GP version* `2.7.2`, 2014, Bordeaux, available online from `http://pari.math.u-bordeaux.fr/`.

[Pei09] C. Peikert, *Public-key cryptosystems from the worst-case shortest vector problem*, In Proceedings of the forty-first annual ACM symposium on Theory of computing, pp. 333-342, ACM, 2009.

[Reg09] O. Regev, *On lattices, learning with errors, random linear codes, and cryptography*, Journal of the ACM (JACM) **56**, no. 6 (2009): 34.

[Shp05] I. Shparlinski, *Playing "hide-and-seek" with numbers: the hidden number problem, lattices, and exponential sums*, In Proceedings Of Symposia In Applied Mathematics, vol. 62, p. 153, 2005.

[vdPS13] J. van de Pol, N. Smart, *Estimating key sizes for high dimensional lattice-based systems*, In Cryptography and Coding, pp. 290-303, Springer Berlin Heidelberg, 2013.