Key Recovery for LWE in Polynomial Time

Kim Laine\(^1\) and Kristin Lauter\(^2\)

\(^1\) Microsoft Research, USA  \(\text{kim.laine@microsoft.com}\)
\(^2\) Microsoft Research, USA  \(\text{klauter@microsoft.com}\)

Abstract. We discuss a higher dimensional generalization of the Hidden Number Problem and generalize the Boneh-Venkatesan method [BV96] for solving it in polynomial time. We then use this to analyze a key recovery (decoding) attack on LWE which runs in polynomial time using the LLL lattice basis reduction algorithm [LLL82] and Babai’s nearest planes method [Bab86]. We prove that success can be guaranteed with overwhelming probability when the error distribution is narrow enough and \(q \geq 2^{O(n)}\), where \(n\) is the dimension of the secret key. An explicit constant in the exponent is given, but in practice the performance is observed to be significantly better.

Our focus is on attacking the search variant of LWE. Known attacks include combinatorial methods [BKW03, ACFFP13], polynomial system solving (Gröbner basis) methods [AG11, ACFP14], and lattice reduction methods [LP11, LNW13, BG14, LM09]. Typically the performance of the lattice reduction attacks involves estimating the performance and complexity of BKZ-2.0 [CN11], which is difficult. Still another option is to attack the decision version of LWE [MR09] and use the search-to-decision reductions to break the search problem [BLPRS13, MP12]. Our key recovery attack is interesting because it is 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 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, provided that the modulus is large enough, and the error distribution narrow enough.

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 \(q\):

\[
\begin{align*}
a_0,0s_0 + a_{0,1}s_1 + \ldots + a_{0,n-1}s_{n-1} &\approx t_0 \pmod q \\
a_1,0s_0 + a_{1,1}s_1 + \ldots + a_{1,n-1}s_{n-1} &\approx t_1 \pmod q \\
\vdots & \vdots \\
a_{d-1,0}s_0 + a_{d-1,1}s_1 + \ldots + a_{d-1,n-1}s_{n-1} &\approx t_{d-1} \pmod q
\end{align*}
\]

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 \([t_0, t_1, \ldots, t_{d-1}]\) or sampled uniformly at random from \(\mathbb{Z}_q^d\). The search version asks to solve the system, i.e. to find \(\mathbf{s} = [s_0, s_1, \ldots, s_{n-1}]\).

In the seminal paper [Reg09], Regev proved that, in some parameter settings, if search-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 \(\gamma = \text{poly}(n)\). 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 \( q = \text{poly}(n) \) 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 \( q = \text{poly}(n) \) from a new lattice problem GapSVP\(_{\gamma,q}\), which is an easier variant of GapSVP. Most importantly, 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, but such a large \( q \) is typically not realistic for practical applications. The combined work of several authors [Pei09, MP12, BLPRS13] also proves that the problems decision-LWE and search-LWE are classically equal hard (up to a polynomial factor) for practically any modulus \( q \). However, one should be careful with these reductions as they change the LWE parameters, so to solve a particular search-LWE instance using the search-to-decision reductions one needs to be able to solve several possibly significantly harder decision-LWE instances with exponentially good advantage.

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, BV14, LNV11, GLN12, BLN14, LNV11, GLN12, BLN14, LNV11, 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. We should also mention that, independently of us, Galbraith and Shani studied generalizations of the HNP in great detail in [GS15], but the methods used and presented here suffice for our purposes.

Our polynomial time key recovery attack should be viewed in the context of the known attacks on search-LWE, namely the embedding attack [LM09, BG14] and the enumeration attacks [LP11, LNV11, BCNS14]. These attacks typically use the BKZ-2.0 algorithm [LN14], which makes their performance and complexity difficult to analyze. Instead we restrict to using the (polynomial time) LLL algorithm [LLL82], whose performance and complexity are much better understood. This is then combined with the well understood nearest planes method of Babai [Bab86] to recover the secret key. We use clear, explicit and well-known results for these algorithms to analyze the conditions for success. As a result we obtain new insight into the hardness of search-LWE for certain parameter ranges. In particular, we prove that this polynomial time attack succeeds almost certainly when the LWE modulus \( q \) is exponential in the dimension \( n \) and the error distribution is narrow enough (see Theorem 5).

In practice, for applications [LNV11, CLN12, BLN14, LNV11, BCNS14], LWE parameters are selected very conservatively due to the difficulty in analyzing their security, and are of course not vulnerable to our polynomial time key recovery attack. This is done at an immense cost to performance however, so it would be crucial to understand precisely how difficult LWE is to break using the best known methods. In this work we approach the situation from the other end of the hardness spectrum, and establish a good understanding of when exactly LWE becomes easy. We further observe that this typically happens close to when decision-LWE becomes easy. This is not obvious, but also not too surprising due to the rather complicated search-to-decision reductions of [BLPRS13, Mi12].

But our attack is interesting for several additional reasons: First, it demonstrates that for surprisingly small modulus \( q \) and narrow error distribution the classical security reduction [Pei09, see Theorem 5 below] is not relevant for cryptography in the sense that both the search-LWE and GapSVP problems can be solved in polynomial time (see Remark 6 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 for the particular \( q \)-ary lattices that arise from the LWE problem. Although theory guarantees that LLL finds a vector of length no more than \( \gamma \) times the length of the shortest vector, where \( \gamma = \frac{2^{\alpha N}}{N} \), \( N \) is the lattice dimension and \( \mu = 1/2 \), in practice it is known (see for example [NS06]) that \( \mu \) can
be expected to be much smaller. More correctly, what we observe is the effective approximation factor appearing in Babai’s nearest planes method given an LLL reduced basis. Secure parameter selection for LWE depends heavily on the asymptotic behavior of the number \( \mu \) in the LLL-Babai algorithm, and our experiments shown in Figure 3 demonstrate the rough growth of \( \mu \) 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 particular depending on the width of the error distribution.

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 Section 5 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 the 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 MSB\( _\ell (k) \) we denote the \( \ell \) most significant bits of the integer \( k \), not counting the sign. For example, MSB\( _4 (175) = 160 \), and MSB\( _3 (-175) = -168 \). Most importantly, we always have \( |k - \text{MSB}_\ell (k)| < 2^{\left\lfloor \log_2 |k| \right\rfloor + 1 - \ell} \).

**Definition 2 (HNP).** Let \( s \in \mathbb{Z}_q \) be a fixed secret number chosen uniformly at random. Given \( d \) samples of the form \( (a, \text{MSB}_\ell ([as]_q)) \in \mathbb{Z}_q \times \mathbb{Z}_q \) where \( a \in \mathbb{Z}_q \) are chosen uniformly at random, the problem 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 \( \ell \) are chosen appropriately, HNP\( _{r,\ell,d} \) can be solved in time \( \text{poly}(r) \). For instance, this happens when \( d = \ell = \sqrt{2r} \).
We will generalize Definition 2 and Theorem 1 to \( n \) dimensions.

**Definition 3 (GHNP).** Let \( s \in \mathbb{Z}_q^n \) be a fixed secret vector chosen uniformly at random. Given \( d \) samples of the form
\[
\left( a, \text{MSB}_\ell (\langle a, s \rangle_q) \right) \in \mathbb{Z}_q^n \times \mathbb{Z}_q,
\]
where \( a \in \mathbb{Z}_q^n \) are chosen uniformly at random, the problem \( \text{GHNP}_{n,r,\ell,d} \) is to recover \( s \).

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

**Remark 1.** Independently of us, Galbraith and Shani studied generalizations of the HNP in great detail in [CSI15], but the methods used and presented here suffice for our purposes.

**Notation.** We denote the \( i \)-th coefficient of a vector \( v \) by \( v[i] \).

We want to solve \( \text{GHNP}_{n,r,\ell,d} \) with samples \( \left( a_i, \text{MSB}_\ell (\langle a_i, s \rangle_q) \right) \), where \( i = 0,\ldots,d - 1 \). The first step is to make this into a lattice problem by considering the full \((n+d)\)-dimensional lattice \( \mathbb{L}_{n,r,\ell,d} \) spanned by the rows of
\[
\begin{pmatrix}
q 1_{d \times d} & 0_{d \times n} \\
A & 2^{1-\ell} 1_{n \times n}
\end{pmatrix}, \quad A := [a_0, a_1, \ldots, a_{d-1}] \in \mathbb{Z}_q^{n \times d}. \tag{1}
\]

Clearly \( \mathbb{L}_{n,r,\ell,d} \) contains the vector
\[
v = \left[ \langle a_0, s \rangle_q, \langle a_1, s \rangle_q, \ldots, \langle a_{d-1}, s \rangle_q, s[0]/2^{\ell-1}, s[1]/2^{\ell-1}, \ldots, s[n-1]/2^{\ell-1} \right]. \tag{2}
\]

Denote
\[
u_i = \text{MSB}_\ell (\langle a_i, s \rangle_q). \tag{3}
\]

The distance between \( \langle a_i, s \rangle_q \) and \( u_i \) can be bounded using Definition 1
\[
|\langle a_i, s \rangle_q - u_i| < 2^{\lfloor \log_2 (|\langle a_i, s \rangle_q|) \rfloor + 1-\ell} \leq 2^{\lfloor \log_2 (q/2)^{-\ell} \rfloor} < 2^{r-\ell}. \tag{4}
\]

The vector
\[
u = [u_0, u_1, \ldots, u_{d-1}, 0, \ldots, 0] \in \mathbb{R}^{n+d} \tag{5}
\]
is not in \( \mathbb{L}_{n,r,\ell,d} \), but using (4) we can bound its Euclidean distance from \( v \):
\[
||v - u|| \leq \sqrt{n + d} 2^{r-\ell}.
\]

**Theorem 2 (LLL-Babai).** Let \( \mathbb{L} \) be a lattice of dimension \( N \). An approximate-CVP in \( \mathbb{L} \) 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 \( \mu \)) can be expected.

**Proof.** The value \( \mu = 1/2 \) follows from the result of Babai [Bab86] and the performance guarantee of LLL [LLS82]. The arguments in [NS06] about average performance of LLL on random lattices explains why LLL yields in some sense much better bases than the theoretical result of [LLS82] 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 \( \text{GHNP}_{n,r,\ell,d} \) in polynomial time is to argue that, in many cases, the algorithm LLL-Babai in Theorem 2 actually solves approximate-CVP for \( u \) well enough to recover \( v \), from which \( s \) can be read. Consider what happens if we run LLL-Babai with input \( u \). By Theorem 2 it is guaranteed to output a vector
\[
w = \left[ (a_0, t_0) + qk[0], (a_1, t_0) + qk[1], \ldots, (a_{d-1}, t_0) + qk[d-1], t[0]/2^{\ell-1}, t[1]/2^{\ell-1}, \ldots, t[n-1]/2^{\ell-1} \right] \in \mathbb{L}_{n,r,\ell,d}, \tag{6}
\]
where \( t \in \mathbb{Z}^n \), \( k \in \mathbb{Z}^d \), such that
\[
||v - w|| \leq ||v - u|| + ||u - w|| \leq \left( 1 + 2^{\mu(n+d)} \right) \sqrt{n + d} 2^{-\ell}.
\] (7)

If this is the case, then all differences \((v - w)[j]\) must lie in the interval
\[
\left[-\left( 1 + 2^{\mu(n+d)} \right) \sqrt{n + d} 2^{-\ell}, \left( 1 + 2^{\mu(n+d)} \right) \sqrt{n + d} 2^{-\ell}\right].
\] (8)

We can assume that \( t \in \mathbb{Z}_q^n \). Namely, let \( t_{\text{red}} \) denote a vector in \( \mathbb{Z}_q^n \) that is obtained by reducing the entries of \( t \) modulo \( q \). By replacing \( t \) with \( t_{\text{red}} \) in the definition of \( w \), we obtain a new lattice vector which differs in the first \( d \) entries from \( w \) by multiples of \( q \). But adding suitable multiples of the first \( n \) generators of the lattice \( \Lambda_{n+r,d} \) (first \( n \) rows of the matrix) to this vector yields a lattice vector \( w_{\text{red}} \) whose first \( d \) entries are the same as those of \( w \) and whose remaining \( n \) entries are possibly smaller of absolute value than those of \( w \).

The first \( d \) differences \((v - w)[j]\) are of the form \( \langle a_j, s - t \rangle_q + q\tilde{k}[j] \), where \( \tilde{k} \in \mathbb{Z}^d \). If we assume that
\[
\left( 1 + 2^{\mu(n+d)} \right) \sqrt{n + d} 2^{-\ell} < \frac{q}{2},
\] (9)
or equivalently that
\[
\ell > \log_2 \left[ \left( 1 + 2^{\mu(n+d)} \right) \sqrt{n + d} \right] + 1,
\] (10)
then \( \tilde{k} = 0 \), so for the first \( d \) differences we obtain the simple conditions
\[
|\langle a_j, s - t \rangle_q| \leq \left( 1 + 2^{\mu(n+d)} \right) \sqrt{n + d} 2^{-\ell} < \frac{q}{2}.
\] (11)

The last \( n \) differences \((v - w)[j]\) are of the form \((s - t)[j]/2^{\ell-1}\) and these also need to be contained in the interval \( [3] \), but since we know that \( s \in \mathbb{Z}_q^n \) and we can assume that \( t \in \mathbb{Z}_q^n \) as was explained above, then certainly \((s - t)[j]/2^{\ell-1}\) are in the interval \( [3] \).

We now work backwards by fixing a vector \( t \in \mathbb{Z}_q^n \), \( t \neq s \), and estimate the probability that there is a vector \( k \in \mathbb{Z}^d \) such that \( w \in \mathbb{Z}^{n+d} \) formed from these, as in \( [6] \), can be the output of LLL-Babai with input \( u \) 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 \( a \in \mathbb{Z}_q^n \) is chosen uniformly at random, then \( \langle a, s - t \rangle \), is distributed uniformly at random in \( \mathbb{Z}_q \), so the probability that \( \langle a, s - t \rangle_q \), is in the interval \( [3] \) is
\[
\frac{1}{q} \left[ 2 \left( 1 + 2^{\mu(n+d)} \right) \sqrt{n + d} 2^{-\ell} \right] + 1.
\] (12)

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

**Lemma 1.** The probability that there is a vector \( k \in \mathbb{Z}^d \) such that \( w \in \mathbb{Z}^{n+d} \) formed from \( t \) and \( k \), as in \( [6] \), can be the output of LLL-Babai with input \( u \) in the sense that all \( (11) \) hold is
\[
\leq \left[ \frac{2 \left( 1 + 2^{\mu(n+d)} \right) \sqrt{n + d} 2^{-\ell} }{q} + 1 \right]^d.
\]

The probability is taken over the \( d \) vectors \( a_j \), chosen uniformly at random from \( \mathbb{Z}_q^n \).

Next we compute the probability that in addition to \( v \) there are no other vectors \( w \neq v \) close enough to \( u \) for LLL-Babai to find them. More precisely, we compute the probability that in addition to \( s \), there are no other vectors \( t \neq s \) that would yield a \( w \) (as in \( [6] \)) close enough to \( u \). There are \( q^n - 1 \) possible vectors \( t \neq s \) 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 \( v \) is the only vector LLL-Babai can output is
\[
> 1 - \left( \frac{q^n - 1}{q^d} \right) \left[ 2 \left( 1 + 2^{\mu(n+d)} \right) \sqrt{n + d} 2^{-\ell} \right]^d ,
\]
where the vectors \( a_j \) 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 \( u \) is almost certainly the correct vector \( v \), from which \( s \) can be read. To get a concrete result, we ask that this probability is at least \( 1 - 1/2^n \), which implies (2).

Remark 2. 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

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

For the sake of getting a neat result, we approximate

\[
2 + \log_2 \sqrt{n + d} \leq \varepsilon(n + d), \quad \varepsilon = \frac{2 + \log_2 \sqrt{n}}{n},
\]

to get

\[
(r + 1)n + \frac{rd}{2} + \frac{3d}{2} \left( \mu(n + d) + 2 - \ell + \log_2 \sqrt{n + d} \right)
\leq (r + 1)n + \frac{rd}{2} + \frac{3d}{2} \varepsilon(n + d) - \ell \leq 0.
\]

This simplifies into

\[
3(\mu + \varepsilon) d^2 - [3\ell - r - 3(\mu + \varepsilon)n] d + 2(r + 1)n \leq 0,
\]

which is possible when the discriminant is positive:

\[
[3\ell - r - 3(\mu + \varepsilon)n]^2 - 24(\mu + \varepsilon)(r + 1)n \geq 0.
\]

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

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

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

\[
r > \frac{21}{2} (\mu + \varepsilon)n,
\]

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

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

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 \( \varepsilon = (2 + \log_2 \sqrt{n})/n \) and suppose
\[
    r > \frac{21}{2} (\mu + \varepsilon)n, \quad \ell > \frac{r}{3} + (\mu + \varepsilon)n + \sqrt{\frac{8}{3}(\mu + \varepsilon)(r + 1)n}, \quad d = \left\lfloor \frac{3\ell - r - 3(\mu + \varepsilon)n}{6(\mu + \varepsilon)} \right\rfloor.
\]
Then GHNP\(_{n,r,t,d}\) 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 \( 1 - 1/2^\alpha \).

Proof. LLL-Babai finds the approximate closest vector in the \((n+d)\)-dimensional lattice \( \Lambda_{n,r,t,d} \) in polynomial time in \( n + d \in O(n^2) \). By the arguments above, if \( r \) and \( \ell \) satisfy the given (loose) bounds, we can expect the vector given by LLL-Babai to be good enough to recover \( s \) with probability at least \( 1 - 1/2^\alpha \). 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 \( \mu \), can be expected.

As was mentioned in Remark 2, 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
\[
\ell \geq (\mu + \varepsilon)n + 2\sqrt{(\mu + \varepsilon)(r + 1)n}, \quad r > \left( 4 + \sqrt{15} \right) \sqrt{1 + \frac{4}{15(\mu + \varepsilon)n}} (\mu + \varepsilon)n.
\]
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, \( q \) a prime integer modulus, \( r := \log_2 q \), and \( \chi \) an error distribution over \( \mathbb{Z}_q \). Let \( s \in \mathbb{Z}_q^n \) be a fixed secret vector chosen uniformly at random. Given access to \( d \) samples of the form
\[
(a, [a, s] + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q,
\]
where \( a \in \mathbb{Z}_q^n \) are chosen uniformly at random and \( e \) are sampled from the error distribution \( \chi \), the problem search-LWE\(_{n,r,d} \) is to recover \( s \).

This is commonly also expressed as follows. Write the \( d \) coefficients vectors \( a \) as columns of a matrix \( A \in \mathbb{Z}_q^{n \times d} \), the \( d \) errors \( e \) as a column vector \( e \in \mathbb{Z}_q^d \), and the samples \([a, s] + e\) as a column vector \( t \in \mathbb{Z}_q^d \). Then the problem search-LWE\(_{n,r,d} \) is to recover \( s \) from the pair \((A, t)\).

Note that this means solving \( s \) from
\[
A^\top s + e \equiv t \pmod{q}.
\]

Definition 5 (decision-LWE). With \( A, s, \) and \( \chi \) as in Definition 4, solving decision-LWE\(_{n,r,d} \) is to distinguish with some non-negligible advantage whether a pair \((A, t) \in \mathbb{Z}_q^{n \times d} \times \mathbb{Z}_q^d \) is sampled uniformly at random, or if it is of the form \((A, A^\top s + e \pmod{q})\), where \( A \in \mathbb{Z}_q^{n \times d} \) is sampled uniformly at random and \( e \) is sampled from \( \chi^d \).

In practice, the distribution \( \chi \) 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 \( \sigma \) is the standard deviation. It is efficient, but non-trivial, to sample from such a distribution up to any level of precision [GPV08, Pei10].

The main result of [Reg09] was that when \( q = \text{poly}(n) \) LWE can be proven to be hard in the following sense.
Theorem 5 ([Pei09]). If \( q = \text{poly}(n), \sigma > \sqrt{n/(2\pi)} \) and \( d = \text{poly}(n) \), then there exists a polynomial time quantum reduction from worst-case GapSVP \( \tilde{O}(nq/\sigma) \) to search-LWE\(n,r,d,D_{\ell,q} \).

For very large \( q \) the following classical reduction can be used.

Theorem 4 ([Reg09]). If \( q \geq 2^{n/2}, \sigma > \sqrt{n/(2\pi)} \) and \( d = \text{poly}(n) \), then there exists a polynomial time classical reduction from worst-case GapSVP \( \tilde{O}(nq/\sigma) \) to search-LWE\(n,r,d,D_{\ell,q} \). For smaller values of \( q \) security can be based on a classical reduction to an easier and less studied decision lattice problem GapSVP\(\zeta,\gamma \), where again the hardness depends on \( nq/\sigma \) being small.

Remark 3. It is important to realize that the usefulness of these security reductions depends crucially on the ratio \( q/\sigma \) being relatively small. In practical applications the standard deviation \( \sigma \) is often taken to be a small constant, instead of a function of \( q \), so the ratio \( q/\sigma \) becomes very large. This means that for practitioners the reductions typically have unfortunately little significance.

Remark 4. In fact, the problems search-LWE and decision-LWE are essentially equally hard due to the polynomial time search-to-decision reductions of [BLPRS13, MP12, Pei09, Reg09]. However, these reductions typically change the parameters of the LWE instance so that to break a particular search-LWE instance one must break several, possibly significantly harder, decision-LWE instances with exponentially good advantage.

To find the LWE secret \( s \) directly using Theorem 5 we need a way to read MSB\(r\) \( \langle a,s \rangle_q \) from \( \langle (a,s) + e \rangle_q \). If \( \sigma \) is small enough and \( \ell \) big enough, this is likely to be possible by simply reading the \( \ell \) most significant bits of \( \langle (a,s) + e \rangle_q \) since adding \( e \) is unlikely to change them. It is not hard to bound the value \( \ell \) that a particular \( \sigma \) 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\(r\) parts of the inner products in the LWE samples as in \( (5) \), simply take

\[
    u_i = \langle (a_i,s) + e_i \rangle_q
\]

(20)

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

\[
    u = [u_0, u_1, \ldots, u_{d-1}, 0, \ldots, 0] \in \mathbb{R}^{n+d}.
\]

(21)

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

\[
    |\langle (a_i,s) \rangle_q - u_i| < 2^{-\ell}
\]

as in \( (7) \), and the rest of the proof goes through without change.

One detail was ignored above. For the argument to work, we need

\[
    \langle (a_i,s) + e \rangle_q = \langle a_i,s \rangle_q + e_i.
\]

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

\[
    \langle (a,s) + e \rangle_q = \langle a,s \rangle_q + e.
\]

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

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

\[
    \Pr[|D_{\ell,\sigma}| \geq B] \leq \frac{B}{\sigma} \exp\left( \frac{1}{2} \frac{B^2}{2\sigma^2} \right).
\]
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 $\ell$ bits of $(a_i, s)_q$ 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 $\ell$ to be the lower bound in Theorem 6 to obtain our main result.

**Theorem 6.** Let $\varepsilon = (2 + \log_2 \sqrt{n})/n$ and suppose $r > (21/2)(\mu + \varepsilon)n$. Let
\[ \ell = \frac{r}{3} + (\mu + \varepsilon)n + \frac{8}{3}(\mu + \varepsilon)(r + 1)n, \quad d = \left\lceil \frac{3\ell - r - 3(\mu + \varepsilon)n}{6(\mu + \varepsilon)} \right\rceil = \left\lceil \frac{2(r + 1)n}{3(\mu + \varepsilon)} \right\rceil. \]
Then search-LWE$_{n,r,d,D_{z,\sigma}}$ 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
\[ \left( 1 - \frac{1}{2^n} \right)^d \left( 1 - \sigma^{-1} 2^{r-\ell} \exp \left( \frac{1}{2} - \frac{2^{2r-2\ell-1}}{\sigma^2} \right) \right)^d. \]

\[ \square \]

Of course the discussion after Theorem 3 applies here also, meaning that success can (roughly speaking) be guaranteed in the sense of Theorem 6 when $n$ is large enough, $r > 2n$ and $d$ is chosen appropriately.

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

### 4 Practical Performance

In the proofs of Theorems 2 and 6 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 6 suggests. In this section we try to get an idea of what can be expected to happen in practice.

The estimate in 13 is very crude on average. In the proof of Theorem 6 the differences $\| (a_i, s)_q - u_i \|$ are exactly equal to the absolute values of the errors $e_i$, which are distributed according to $D_{z,\sigma}$.

If instead of using the rows of a matrix like that in 11 we use the rows of
\[ \left( q 1_{d \times d} \begin{bmatrix} A & 0_{d \times n} \end{bmatrix} \begin{bmatrix} [\sigma] & 2^{1-|r|} 1_{n \times n} \end{bmatrix} \right), \]
where again $A := [a_0, a_1, \ldots, a_{d-1}] \in Z_q^{n \times d}$ as in Definition 4 to generate the lattice $A_{n,r,\ell,d}$, the expectation value of $\| v - u \|^2$ is
\[ \leq d \mathbb{E} [D_{z,\sigma}^2] + n [\sigma]^2 = d (\sigma^2 + \mathbb{E} [D_{z,\sigma}^2]) + n [\sigma]^2 \leq (n + d) [\sigma]^2, \]
so we can expect the distance $\| v - u \|$ to be bounded from above by $\sqrt{n + d} [\sigma]$.

Another significant improvement to the running time is to define an $(n + d) \times d$ matrix
\[ \left( q 1_{d \times d} \begin{bmatrix} A \end{bmatrix} \right) \]
and let $A_q$ be its $d \times d$ row-Hermite normal form, i.e. $A_q$ is a triangular matrix whose rows generate the same $Z$-module as the rows of the matrix 12. Let $A$ be the full $d$-dimensional lattice generated by the rows of $A_q$. As before, let $u_i = [(a_i, s) + e_i]_q$ and set
\[ u = [u_0, u_1, \ldots, u_{d-1}] \in \mathbb{R}^d. \]
Now use LLL-Babai to find a vector close to \( u \) in the lattice \( \Lambda \), i.e. a vector which is an integral linear combinations of the rows of \( A_q \). Simply express this in the original basis, i.e. in terms of the rows of the matrix \( [22] \), to recover a candidate for \( s \) 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

\[
v = \left[ (a_0, s)_q, (a_1, s)_q, (a_2, s)_q, \ldots, (a_{d-1}, s)_q \right]
\]

and find that the expected distance squared \( ||v - u||^2 \) is

\[
d \mathbb{E} [D_{\Lambda,s}^2] = d \left( \sigma^2 + \mathbb{E} [D_{\Lambda,s}]^2 \right) = d \sigma^2,
\]

so that the expected distance \( ||v - u|| \) 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

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

(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 \( p = 0 \).

Remark 7. Lattices that contain all coordinate vectors of length \( q \) are called \( q \)-ary lattices. The lattice \( \Lambda \) is obviously a \( q \)-ary lattice.

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 \( O(d^4 r^2) \), 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 \( d^4 r^2 \).

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.

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 \( \mu \leq 1/2 \), or \( \mu \leq 1/4 \) (see the discussion after Theorem 2), but in practice significantly smaller \( \mu \) 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. Furthermore, it is crucial to understand how the \( q \)-ary structure (see Remark 7) of the lattice \( \Lambda \) affects the expected performance.

4. Our attack gives an indirect way to measure the effective value of \( \mu \) in the approximation factor \( 2^\mu_d \) of LLL-Babai for \( q \)-ary lattices: Because we can predict whether our attack will succeed or fail fairly accurately based on the value of \( \mu \), 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 \( \mu \). 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 \( \mu \) 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 \( \mu \) derived from the formula for the likelihood that the attack will succeed (Equation 24):

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

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 \( 2^\mu_d \).

An interesting further simplification is obtained by setting \( p = 0 \), which we already mentioned in Remark 6. It is clear from the form of \( \mu_{\text{LLL}} \) that the effect of \( p \) is very small unless \( p \) is extremely close to 1. We use this choice from now on:

\[
\mu_{\text{LLL}}(n, r, d, \sigma, p = 0) \approx \frac{1}{d} \log_2 \left[ \frac{1}{2\sqrt{d}} \cdot \frac{q}{\sigma} \cdot \frac{1}{q^{n/d}} \right].
\]
The graphs in Figure 3 show a relatively clear boundary in the values of $\mu_{LLL}$ between failed and succeeded attacks, which can then be extrapolated to bigger examples. We present the values $\mu_{LLL}$ as functions of both $n$ and $d$, where $d$ is the dimension of the lattice $\Lambda$ for which LLL was performed. A green dot indicates that the attack succeeded (correct $s$ was recovered) and a red dot that the attack failed (incorrect $s$ was recovered).

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

The dimension $d$ of course affects $\mu_{LLL}$ very strongly, so we want to choose it in an optimal way given all the other parameters, i.e. in a way that maximizes $\mu_{LLL}$. We let $d_{opt}$ be such that $\partial_d \mu_{LLL}(n, r, d_{opt}, \sigma, p = 0) = 0$ (rounded to an integer). Parameter selection in all of the attacks we performed was done by taking $d \approx d_{opt}$. For a particular value of $n$ the experiments differ only in the choice of $r$, and $d \approx d_{opt}$ is always computed case-by-case. It is not hard to see that when the example size increases, the value $d_{opt}$ approaches $2n$.

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

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

4.2 Practical Key Recovery

In practice, key recovery in polynomial time can be performed as follows. The LWE problem determines $n$, $r$ and $\sigma$. Now find $d_{\text{opt}}$ and see if the corresponding $\mu_{\text{LLL}}$ 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 $\mu_{\text{LLL}}$. Now observe $d$ LWE samples, form the matrix $[22]$, find the row-Hermite normal form $A_q$, form the lattice $\Lambda$ generated by the rows of $A_q$ and use LLL-Babai to find the closest lattice point to $u$ (as in $[23]$), express the closest vector in terms of the original basis (rows of $[22]$) and read the last $n$ entries to find $s$.

5 Security Implications

Key recovery for LWE in polynomial time is only possible when the ratio $q/\sigma$ is very large, which can be seen for example from $[25]$, and is suggested by Theorems 1 and 5. It is possible that such a situation might never occur, since one could always ensure that $\sigma$ is linear in $q$. For practitioners in the field of homomorphic cryptography the situation looks radically different. LWE parameters with very large $q$ and very small constant $\sigma$ are necessary to allow deeper circuits to be evaluated reasonably efficiently. To make performance of such cryptosystems practical one needs to push the limits of the secure parameter range. The results presented here are one step further towards understanding more precisely how the security of LWE behaves for such extreme parameters, but much more work is still needed to explain how for instance slightly more powerful lattice reduction would change the situation.
Typically the security of LWE-based cryptosystems is evaluated by estimating the complexity and performance of the best known lattice attacks against either search-LWE or decision-LWE. Recall (Remark 4) that these problems are essentially equally difficult, although the practicality of the search-to-decision reductions for an attacker is not clear.

Unfortunately, it is very difficult to give tight security estimates since the best lattice reductions algorithms, such as BKZ-2.0 [CN11], are complicated and not well enough understood. Often only attacks against decision-LWE are considered [MR09] when parameters are selected, even though there are arguments suggesting that in fact attacking search-LWE directly is more efficient [LP11, BG14, LN13].

A series of papers presenting applications of homomorphic encryption ([LNV11, GLN12, BLN14, LLN14]) give recommended parameter sizes for (R)LWE based on attacks against decision-LWE. For example, [GLN12] 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 [BLN14] suggests in addition \( (n, q) = (2^{14}, 2^{312}) \) for evaluating the logistical regression function. In [LN14] several parameters are presented that are estimated to achieve a security level of 80 bits against an advantage of \( 2^{-80} \) for solving decision-LWE. We list these in Table 2.

<table>
<thead>
<tr>
<th>( n )</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
<th>16384</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r \leq )</td>
<td>47.5</td>
<td>95.4</td>
<td>192.0</td>
<td>392.1</td>
<td>799.6</td>
</tr>
</tbody>
</table>

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

We would like to stress that the true security level of these parameters using the best known lattice reduction attacks is not clear, and that large \( n \) such as \( n \geq 8192 \) makes most homomorphic cryptosystems too inefficient for many practical purposes (but not all). Using larger \( q \) and smaller \( n \) would quickly result in huge performance benefits.

**Example 1.** We can try to extrapolate the results of our experiments presented in Table 1 and Figure 3 to guess how large \( q \) needs to be for our attack to work with \( n = 1024 \). More work and experiments are clearly needed to say anything conclusive, but one should be very worried about using anything even close to \( q = 2^{140} \). According to the complexity estimates of LLL [NS06] such an attack would take around 4 years to run using our setup.

**Example 2.** In [LN14], homomorphic evaluation of encryption and decryption circuits for block ciphers is discussed and two homomorphic encryption schemes are compared, the Fan-Vercauteren scheme [FV12] and YASHE [BLLN13]. As soon as one wishes to perform more than one homomorphic multiplication, the lower bound on \( q \) increases significantly. For example, using the Fan-Vercauteren scheme, to be able to do 10 homomorphic multiplications with \( n = 1024 \) one needs to have \( q \geq 2^{229} \) to ensure correct decryption. When \( n \) is increased, the required lower bound for \( q \) does increase, but slowly enough so that eventually a set of parameters is reached that resists all known attacks. For example, it suffices to take \( n \geq 4096 \) to be able to perform 10 homomorphic multiplications with the Fan-Vercauteren scheme and be safe at least against a polynomial time attack.

We conclude with the following interesting observation. Performance estimates for the standard attack against decision-LWE (see e.g. [MR09, LP11]) suggest that the probability for succeeding is given by

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

where \( d = \sqrt{nr/\log_2 \delta} \) [MR09] and \( \delta \) is the root-Hermite factor of a reduced basis of a certain \( d \)-dimensional lattice \( \Lambda \). In [NS06] it is explained that, for a random lattice, LLL can be expected to yield a basis with \( \delta \approx 1.021 \).

\( ^5 \) The lattice in question is the scaled dual of the lattice \( \Lambda \).
Formula (26) again clearly shows how the security level decreases when $q$ increases, and other parameters are held fixed. Setting $\delta = 1.021$ and computing some values of (26), we observe that the probability of successfully breaking decision-LWE becomes high as $q$ increases almost exactly when our key recovery attack can be expected to succeed. In other words, search-LWE seems to become easy almost exactly when decision-LWE becomes easy, for the exact same parameters.

References


