On the Efficiency of Provably Secure NTRU

Daniel Cabarcas[1], Patrick Weiden[2], and Johannes Buchmann[2]

[1] Universidad Nacional de Colombia sede Medellín, Medellín, Colombia
dcabarc@unal.edu.co
[2] Technische Universität Darmstadt, Darmstadt, Germany
{pweiden,buchmann}@cdc.informatik.tu-darmstadt.de

Abstract. It is still a challenge to find a lattice-based public-key encryption scheme that combines efficiency (as e.g. NTRUEncrypt) with a very strong security guarantee (as e.g. the ring-LWE based scheme of Lyubashevsky, Peikert, and Regev LPR-LWE). Stehlé and Steinfeld (EUROCRYPT 11) presented a provably secure variant of NTRUEncrypt (pNE), perhaps the first step towards addressing the challenge. In this paper we thoroughly assess the efficiency of pNE, and investigate whether it can meet those presumed extremes. We show how to select parameters that provide a given security level and we explain how to instantiate pNE. As we compare our instantiation of pNE to NTRUEncrypt and LPR-LWE, we find that pNE is still inferior to both due to the very wide Gaussian distribution used in its key generation.

Keywords: Public-Key Encryption, Efficiency, NTRUEncrypt, Lattice, Learning With Errors, and Discrete Gaussian Distribution.

1 Introduction

Public-key encryption (PKE) based on lattice problems has attracted a lot of attention in the last decade. This is in part due to the need for alternatives to traditional PKE. Most PKE schemes in use today rely on the hardness of either factoring or computing discrete logarithms. However, the trustworthiness of these assumptions has been eroding by improvements in factoring algorithms and by polynomial time quantum algorithms that solve both problems.

The success of lattice-based PKE has also its own virtues. Among lattice-based encryption schemes one can find NTRUEncrypt, which is competitive in terms of practical efficiency with well established PKEs like RSA [17]. Although the most efficient attacks against NTRUEncrypt use lattice algorithms, there exists no formal proof relating the security of the scheme to a lattice problem.

There are also lattice-based PKE schemes such as Regev's [32], that allow for a worst- to average-case reduction from well established lattice problems. Regev proposed a scheme that is secure as long as the learning with errors problem (LWE) is hard on average, and he shows that the LWE problem is as hard as solving a well established lattice problem in its worst case. His original reduction was a quantum reduction, yet more recently it was shown that a classical reduction is also possible [5]. These worst- to average-case reductions

M. Mosca (Ed.): PQCrypto 2014, LNCS 8772, pp. 22–39, 2014.

have also been used to prove other related problems hard, such as the ring-LWE problem [25] and small parameter variants [26], and subsequently, other provably secure PKE schemes have been proposed [36,13,25,23,8]. In any case, the worst- to average-case reduction is considered a very strong security guarantee because it proves that breaking a randomly chosen instance of the scheme is as hard as solving the hardest instance of a well established lattice problem.

The construction of a lattice-based PKE scheme that has both properties —strong security and practical efficiency— is still an interesting challenge. At Eurocrypt 2011, Stehlé and Steinfeld made a first step towards solving this challenge by presenting a provably secure NTRU-variant [35], which we refer to as pNE. The pNE scheme permits a worst- to average-case reduction. Its similarity to NTRUEncrypt raises the questions of how efficient pNE is and of how far the scheme closes the efficiency gap between NTRUEncrypt and a provably secure scheme. Although the authors of pNE state that an instantiation of pNE is likely to be less efficient than NTRUEncrypt, it is still important to determine how much it closes this gap. Moreover, a recent homomorphic implementation of pNE [4] seem to suggest that pNE can be rather efficient.

The pNE and NTRUEncrypt schemes are structurally very similar. Operations take place in the quotient ring $R_q = \mathbb{Z}_q[x]/\mathbf{\Phi}(x)$, where q is a moderately large integer and $\mathbf{\Phi}(x)$ is a degree n polynomial. The secret key $\mathbf{f}$ is sampled from R_q and the public key $\mathbf{h}$ is computed as $\mathbf{h} = p\mathbf{g}/\mathbf{f} \in R_q$, where p is a small integer and $\mathbf{g}$ is also sampled from R_q. Then a message $\mathbf{m} \in R_p$ is encrypted by sampling small elements $\mathbf{e}, \mathbf{e}'$ in R_q, and computing the ciphertext $\mathbf{c} := \mathbf{he} + p\mathbf{e}' + \mathbf{m} \in R_q$. The main difference between pNE and NTRUEncrypt is the distribution used to sample $\mathbf{f}$ and $\mathbf{g}$. Stehlé and Steinfeld show that if $\mathbf{f}$ and $\mathbf{g}$ are sampled from a discrete Gaussian distribution with a large parameter σ, instead of being sampled uniformly at random from a set of small norm polynomials, the public key $\mathbf{h}$ is statistically close to uniform [35]. They then show that the ciphertext is indistinguishable from random assuming the hardness of the LWE problem on average. Finally, they rely on Regev's [32] worst- to average-case reduction from well established lattice problems to conclude that pNE is secure as long as some lattice problems are hard in the worst case.

Our Contribution. In this paper we answer the question of how efficient pNE is and of how far the scheme closes the efficiency gap between NTRUEncrypt and a provably secure scheme. We do so by presenting a thorough assessment of the efficiency of pNE. In order to achieve this, we address five important problems that are of interest in their own right.

1. We show how to select parameters that provide an expected security level. We do not rely on the worst-case hardness in this context because it is unknown how tight the worst- to average-case reduction is. We rather analyze a well-known attack against the average-case problem underlying pNE and deduce the corresponding security level. For example, a lattice dimension of 2048 is required for a bit security level of 144. Although the security level is

not grounded on the worst- to average-case reduction, we make sure it holds for the chosen parameters.
2. Next, we explain how to implement pNE. In particular, we discuss how to adapt best known discrete Gaussian samplers to fit the needs of pNE and we compare their efficiency.
3. We present experimental data on the performance of pNE. It shows that our pNE implementation is over 100 times slower and requires 24 times larger keys than an implementation of NTRUEncrypt by Security Innovation Inc. We consider this gap too large to be overcome through optimization alone.
4. We then move on to compare pNE with the ring-LWE based scheme of Lyubashevsky, Peikert, and Regev [25] (LPR-LWE). We chose LPR-LWE because it appears to be one of the most efficient provably secure lattice-based PKE schemes, based on the parameter selection by Lindner and Peikert [23] and its implementation by Göttert et al. [15]. In order to allow for a fair comparison, we use an analogous method to derive the LPR-LWE parameters and we implement it using the same procedures as in the pNE implementation. This is possible because pNE and LPR-LWE are structurally similar and rely on the same security assumption. It turns out that LPR-LWE is still superior over pNE. LPR-LWE is more than 5 times faster than pNE and pNE uses keys more than 12 times larger than LPR-LWE.
5. Finally, through a careful analysis, we conclude that the main reason pNE is still less efficient than the other schemes is the very wide Gaussian distribution used in key generation, and the unexpected influence this has on the practical security of the scheme.

Related Work. In a related work, Bos et al. [4] analyze the efficiency of a leveled homomorphic encryption scheme based on pNE. We discuss their results in more detail at the end of Section 4. In this paper we do not analyze the performance of NTRUEncrypt since several works highlight its efficiency, see for example [17,29,20]. There has been less scrutiny over LPR-LWE. Although Göttert et al. [15] tested it in software and hardware, the parameters they consider do not support a worst- to average-case reduction (see [23]). In this paper we analyze LPR-LWE's efficiency for a parameter set that does support such a reduction. Finally, we compare the efficiency of four Gaussian samplers over the integers to use the best for our implementation of pNE. The samplers we consider were proposed by Gentry et al. [14], Peikert [30], Knuth and Yao [22] (adapted in [12]), and Buchmann et al. [7]. The samplers of Ducas et al. [11] and (a discrete variant) of Karney [21] we did not consider in this paper due to their less practical efficiency as stated by the authors.

Organization. This paper is organized as follows. Section 2 introduces notation and background material. Section 3 explains how to select parameters that provide an expected security level. Section 4 briefly describes our implementation of pNE, presents experimental data on its performance, and compares it with

both NTRUEncrypt and LPR-LWE. Finally, Section 5 analyzes the efficiency of pNE and draws a conclusion about its main source of inefficiency.

2 Preliminaries

In this section we recall some of the background necessary to support the remainder of the paper. We first describe the assumed hard problem that the pNE scheme relies on, namely the ring-LWE problem. Next, we describe Gaussian sampling techniques, an important building block for implementing pNE. Finally, we describe the pNE scheme itself.

2.1 The Ring-LWE Problem

The security of pNE relies on the assumption that the learning with errors problem over rings (ring-LWE) is hard. The ring-LWE problem was introduced by Lyubashevsky, Peikert and Regev [25] as an adaptation of the LWE problem [32] to ideal lattices.

The *decisional ring-LWE problem* is parametrized by a positive integer q, a polynomial $\mathbf{\Phi}(x) \in \mathbb{Z}[x]$, and a noise distribution χ on $R_q = \mathbb{Z}_q[x]/(\mathbf{\Phi}(x))$. For $\mathbf{s} \in R_q$, define $A_{\mathbf{s}}$ to be the distribution of pairs $(\mathbf{a}, \mathbf{b})$, where $\mathbf{a}$ is uniformly chosen in R_q and $\mathbf{b} = \mathbf{as} + \mathbf{e}$, with $\mathbf{e}$ sampled from χ. The decisional ring-LWE problem is defined as follows: For a uniformly random $\mathbf{s} \in R_q$ (which is kept secret) and given arbitrarily many samples $(\mathbf{a}, \mathbf{b})$, determine whether the samples come from $A_{\mathbf{s}}$, or whether they are uniformly distributed in $R_q \times R_q$. The search variant is to determine $\mathbf{s}$ given arbitrarily many samples from $A_{\mathbf{s}}$. For certain parameters, there is a quantum reduction from worst-case classical lattice problems over ideal lattices to the average-case ring-LWE problem [25].

Several algorithms have been proposed to solve the LWE and ring-LWE problems, see for example [27,23,2,1,24]. Some of these algorithms rely on lattice-basis reductions by algorithms such as BKZ [33] or BKZ 2.0 [9]. Another approach is based on the BKW algorithm [3], a method for solving the learning parity with noise problem. Pruned-enumeration has also been proved to be a viable option [24]. Some algorithms take advantage of the ring structure (e.g. [31]), others are oblivious to it.

In this paper we base all practical security estimates on the well established distinguishing attack [27] using BKZ. Although other attacks might be more effective, it is outside the scope of this paper to compare the effectiveness of the different attacks. Lindner and Peikert [23] heuristically estimate that the running time of the distinguishing attack using BKZ is

$$\log(t_\epsilon) = 1.8/\log(\delta_\epsilon) - 110\,, \tag{1}$$

where δ_ϵ is the so called *Hermite factor* (see Appendix A for more details.) The expression $\log(\delta_\epsilon)$ is polynomial in n, thus $\log(t_\epsilon)$ is also polynomial in n, and therefore t_ϵ is exponential in n. It is also worth noticing that $\log(t_\epsilon)$ is of the order of $1/\log(q)$, a fact that will play an important role in Section 5, where we analyze pNE's efficiency.

2.2 Sampling Discrete Gaussians

For a vector $\mathbf{v} \in \mathbb{R}^n$, a positive real σ, and a lattice $\mathcal{L} \subset \mathbb{R}^n$, let $D_{\mathcal{L},\mathbf{v},\sigma}$ denote the n-dimensional discrete Gaussian distribution over $\mathcal{L}$, centered at $\mathbf{v}$, with parameter σ. For $x \in \mathcal{L}$, $D_{\mathcal{L},\mathbf{v},\sigma}$ assigns probability

$$D_{\mathcal{L},\mathbf{v},\sigma}(\mathbf{x}) := \frac{\rho_{\mathbf{v},\sigma}(\mathbf{x})}{\sum_{\mathbf{z}\in\mathcal{L}} \rho_{\mathbf{v},\sigma}(\mathbf{z})}$$

with $\rho_{\mathbf{v},\sigma}(\mathbf{x}) = \exp\left(-\frac{1}{2}\left\|\mathbf{x}-\mathbf{v}\right\|^2/\sigma^2\right)$. For brevity we write $D_{\mathcal{L},\sigma}$ for $D_{\mathcal{L},\mathbf{0},\sigma}$ and ρ_σ for $\rho_{\mathbf{0},\sigma}$.[1] For practical reasons, we will use a spheric Gaussian distribution, where each coordinate is sampled independently according to the discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ over the integers, and we rely on the fact that $\sum_{z\in\mathbb{Z}} \rho_\sigma(z)$ is constant and hence $D_{\mathbb{Z},\sigma}$ is proportional to ρ_σ.

Several methods have been proposed to sample values from $D_{\mathbb{Z},\sigma}$. We consider the following sampling algorithms: rejection sampling [14], inverting the cumulative distribution function (CDF) [30], the Knuth-Yao algorithm [22,12], and the Ziggurat algorithm [7]. Besides those, there have been developed two other methods quite recently [11,21], which we omit here since the authors state that their methods are slower in practice than existing ones.

We briefly recall the different methods listed above. Let k be some positive real number.[2] In the rejection sampling method, one samples points (x, y) inside the rectangle $B := [-k\sigma, k\sigma)\cap\mathbb{Z}\times[0, 1)$ uniformly at random and outputs x whenever (x, y) is below the graph of ρ_σ.[3] The Ziggurat algorithm is a more advanced rejection sampling algorithm in B. In a precomputation step, one divides the graph of ρ_σ into a partition of horizontal rectangles. Then, one first chooses one of the rectangles and samples a point (x, y) with integer x-coordinate inside this rectangle next (both uniformly at random). Depending on the location inside the rectangle, either x is directly output, rejection sampling is needed or the process is restarted. In the inverse CDF method one precomputes the CDF values $p_z = \Pr[\, D_{\mathbb{Z},\sigma} \le z]$ for all integers $z \in [-k\sigma, k\sigma)$. Then, one samples y uniformly at random in $[0, 1)$ and outputs $x \in [-k\sigma, k\sigma) \cap \mathbb{Z}$ such that $y \in [p_{x-1}, p_x)$. In the Knuth-Yao algorithm one constructs in advance a tree using the binary expansion of the probabilities $\rho_\sigma(z)$ for $z \in [-k\sigma, k\sigma)\cap\mathbb{Z}$ up to some predefined precision. During the sampling process, one walks down the binary tree, using

[1] Some authors use a slightly different definition $\rho_{\mathbf{v},s}(\mathbf{x}) = \exp\left(-\pi\left\|\mathbf{x}-\mathbf{v}\right\|^2/s^2\right)$. The two definitions are equivalent with $s = \sqrt{2\pi}\cdot\sigma$.

[2] The parameter k affects the distribution and the running time: A larger k yields a better fit to $D_{\mathbb{Z},\sigma}$, but increases both storage and rejection rate (and thus running time). Gentry et al. proved that the rejection rate (see description of rejection sampling) is proportional to k and independent of σ [14]. Moreover, they showed that for $k = \omega(\sqrt{\log(n)})$ the output distribution is statistically close to $D_{\mathbb{Z},\sigma}$.

[3] An equivalent view is that one first samples an integer x uniformly at random in the interval $[-k\sigma, k\sigma)$. Then, with probability $\rho_\sigma(x)$ one outputs x, otherwise one restarts.

one uniformly chosen bit at each step to decide which child to move to, and finally outputs the integer of the reached leaf.

2.3 Stehlé and Steinfeld's pNE Scheme

We briefly recall Stehlé and Steinfeld's provably secure encryption scheme pNE [35], which is specified by the following public parameters:

- dimension $n > 8$, a power of 2, which determines the cyclotomic polynomial $\mathbf{\Phi}(x) = x^n + 1$ and the quotient ring $R = \mathbb{Z}[x]/\mathbf{\Phi}(x)$,
- a prime $q > 5$ such that $q \equiv 1 \bmod 2n$, which determines the ciphertext space $R_q = \mathbb{Z}_q[x]/\mathbf{\Phi}(x)$,
- a polynomial $p \in R$ such that p is invertible in R_q and has small coefficients (typically $p = 2$, $p = 3$ or $p = x + 2$), which determines the message space $\mathcal{P} = R/pR$,
- a ring-LWE noise distribution χ,
- and a positive real σ that determines the (n-dimensional sperical) discrete Gaussian distribution $D_{\mathbb{Z}^n,\sigma}$ used in key generation.

The scheme pNE = (KeyGen, Encrypt, Decrypt) is defined as follows.

KeyGen: Sample $\mathbf{f}' \leftarrow D_{\mathbb{Z}^n,\sigma}$, let $\mathbf{f} = p\mathbf{f}' + 1 \bmod q$; if $\mathbf{f} \notin R_q^\times$ resample. Sample $\mathbf{g} \leftarrow D_{\mathbb{Z}^n,\sigma}$; if $\mathbf{g} \notin R_q^\times$ resample. The secret key is $\mathbf{f}$ and the public key is $\mathbf{h} := p\mathbf{g}/\mathbf{f} \in R_q$.

Encrypt($\mathbf{h}, \mathbf{m}$): Sample $\mathbf{e}, \mathbf{e}' \leftarrow \chi$, and return the ciphertext $\mathbf{c} := \mathbf{he} + p\mathbf{e}' + \mathbf{m} \in R_q$.

Decrypt($\mathbf{f}, \mathbf{c}$): Compute $\mathbf{c}' := \mathbf{f} \cdot \mathbf{c} \in R_q$ and return $\mathbf{c}' \bmod p$.

Stehlé and Steinfeld show that pNE is secure as long as some classical lattice problems are hard to solve on quantum computers [35]. First they show that for certain parameter choices, the public key $\mathbf{h}$ is statistically close to uniform. Then, they show that $\mathbf{he} + p\mathbf{e}'$ is basically a sample from a ring-LWE distribution, and hence an IND-CPA attack on pNE can be used to solve ring-LWE. Then, by the worst- to average-case quantum reduction [25], the hardness result follows.[4]

Stehlé and Steinfeld also show that there exist parameter choices for which decryption correctly recovers the plaintext [35]. Let $\mathbf{c}'' = p(\mathbf{ge} + \mathbf{e}'\mathbf{f}) + \mathbf{fm} \in R$. If $\|\mathbf{c}''\|_\infty < q/2$, no wrapping around q occurs, and thus $\mathbf{c}' = \mathbf{c}'' \bmod q = \mathbf{c}''$ and decryption recovers the message $\mathbf{m}$ always.

For pNE to be both secure and correct, the parameters need to be chosen carefully. On the one hand, for the public key $\mathbf{h}$ to be statistically close to uniform, a large parameter σ is required. On the other hand, a large σ increases the size of $\mathbf{c}''$ and hence forces a larger modulus q for decryption to work. Balancing this conflicting forces is an important achievement of the authors of pNE. This balancing act is also decisive for pNE's efficiency as we will show in the following sections.

[4] Here we rely on the original security proof by Stehlé and Steinfeld [35]. However, Brakerski et al. [5] quite recently established a classical worst- to average-case reduction that might apply in this case.

3 Parameter Selection for pNE

We propose concrete parameters for pNE so that it is both correct and secure. For correctness, we name a range of values for the modulus q that guarantee a negligible error rate. Next, we show how to select parameters that provide an expected security level.

- Fix n to be a power of two.
- Fix $p = 2$. This choice provides a useful message space and causes the least possible expansion on the noise.
- Set χ to be the discrete Gaussian distribution $D_{\mathbb{Z}^n,r}$ for some real r (see below). Elements can be efficiently drawn from this distribution and moreover, with n a power of two and $\mathbf{\Phi}(x) = x^n + 1$, the ring-LWE noise distribution can be spherical, and the worst-case reduction still holds [25].
- Set $r = \sqrt{2n/\pi}$, so that ring-LWE is as hard as lattice problems in the worst-case (see [25] for details).
- Set $\sigma = 2n\sqrt{\ln(8nq)q}$. With this, the public key is statistically close to uniform, thus an IND-CPA attack implies solving an instance of ring-LWE [35].
- Choose a prime $q \in [dn^6 \ln(n), 2dn^6 \ln(n)]$, such that $q \equiv 1 \bmod 2n$. We show in Lemma 3 below that $d = 25830$ guarantees correctness of the scheme. Experimentally, we obtain a lower value $d = 2^9$.

Table 1 shows some sets of parameter values computed as described above.

Table 1. Parameter values for pNE, security and error rate estimates. For given values of n, columns two through four show values for parameters q, σ and r that specify an instance of pNE. For a given set of parameters, column seven shows the estimated running time of a distinguishing attack, and columns five and six show the advantage and corresponding Hermite factor, respectively, under which such running time is achieved. Column eight shows the equivalent bit security and column nine the error rate.

Parameters				Advantage	Hermite	Attack time	Equiv. bit	Error
n	$\log q$	$\log \sigma$	r	$\log(1/\epsilon)$	factor δ_ϵ	$\log(T)$ [s]	security	rate
1024	71,90	49,89	25,53	2,72	1,0102	16	38	$O(2^{-n})$
2048	77,28	53,63	36,11	4,63	1,0055	122	144	$O(2^{-n})$
4096	83,30	57,70	51,06	7,85	1,0030	315	338	$O(2^{-n})$

The following two results will be used to prove the correctness of pNE for the proposed parameters.

Lemma 1 ([35], Lemma 11). *Let $n \geq 8$ be a power of two such that $\mathbf{\Phi}(x) = x^n + 1$ splits into n linear factors modulo a prime $q \geq 8n$. Let $\sigma \geq \sqrt{2n\ln(6n)/\pi} \cdot q^{1/n}$ and let $p = 2$. The polynomials $\mathbf{f}$ and $\mathbf{g}$, generated by the KeyGen algorithm, satisfy, with probability $\geq 1 - 2^{-n+3}$,*

$$\|\mathbf{f}\| \leq 8\sqrt{n} \cdot \sigma \quad \textit{and} \quad \|\mathbf{g}\| \leq \sqrt{n} \cdot \sigma \ .$$

Lemma 2 ([28], Lemma 3.1). *Let $n \in \mathbb{N}$. For any real $r = \omega(\sqrt{\log(n)})$, the probability that a polynomial $\mathbf{e}$ chosen according to $D_{\mathbb{Z}^n,r}$ has norm $\|\mathbf{e}\| > r\sqrt{n}$ is $\leq 2^{-n+1}$.*

The following lemma establishes the correctness of pNE for the proposed parameters.

Lemma 3. *Let $p = 2$, n a power of two s.t. $log(n) \geq 3$, $r = \sqrt{2n/\pi}$, $\chi = D_{\mathbb{Z}^n,r}$, $\sigma = 2n\sqrt{\ln(8nq)q}$, and $d \geq 25830$. If q is a prime in $[dn^6 \ln(n), 2dn^6 \ln(n)]$, then pNE correctly recovers plaintexts with probability greater or equal to $1 - 2^{-n+6}$.*

Proof. Let $\mathbf{c}'' = p(\mathbf{ge} + \mathbf{e}'\mathbf{f}) + \mathbf{fm} \in R$. Decryption recovers $\mathbf{m}$ if $\|\mathbf{c}''\|_\infty < q/2$. From Lemma 1, we have that $\|\mathbf{g}\|_2 \leq \sqrt{n} \cdot \sigma$ and $\|\mathbf{f}\|_2 \leq 8\sqrt{n} \cdot \sigma$ with probability $\geq 1 - 2^{-n+3}$. Furthermore, it is $\|p\mathbf{g}\|_2 \leq \sqrt{2n} \cdot \sigma$ and $\|p\mathbf{f}\|_2 \leq 8\sqrt{2n} \cdot \sigma$, both with probability $\geq 1 - 2^{-n+3}$. We also have that

$$\|p\mathbf{ge}\|_\infty \leq \|p\mathbf{ge}\|_2 \leq \sqrt{n}\|p\mathbf{g}\|_2\|\mathbf{e}\|_2 \ .$$

Since $\mathbf{e}$ is drawn from $D_{\mathbb{Z}^n,r}$, it follows from Lemma 2 that $\|\mathbf{e}\|_2 \leq \sqrt{n} \cdot r$ with probability $\geq 1 - 2^{-n+1}$. It follows that $\|p\mathbf{ge}\|_\infty \leq \sqrt{2n} \cdot n\sigma r$ with probability $\geq 1 - 2^{-n+4}$. Similarly, $\|p\mathbf{e}'\mathbf{f}\|_\infty \leq 8\sqrt{2n} \cdot n\sigma r$ with probability $\geq 1 - 2^{-n+4}$. Also, $\|\mathbf{fm}\|_\infty \leq \|\mathbf{fm}\|_2 \leq \sqrt{n}\|\mathbf{f}\|_2\|\mathbf{m}\|_2 = \sqrt{n}\|2\mathbf{f}' + 1\|_2\|\mathbf{m}\|_2$. Since $\mathbf{f}' \leftarrow D_{\mathbb{Z}^n,\sigma}$, $\|2\mathbf{f}' + 1\|_2 \leq \sqrt{2n} \cdot \sigma$ with probability $\geq 1 - 2^{-n+1}$. Since $\mathbf{m} \in R_2$, $\|\mathbf{m}\|_2 \leq \sqrt{n}$. Thus $\|\mathbf{fm}\|_\infty \leq \sqrt{2n} \cdot n\sigma$ with probability $\geq 1 - 2^{-n+1}$. Then

$$\begin{aligned} &\|\mathbf{c}''\|_\infty \leq \sqrt{2n} \cdot n\sigma(9r + 1), \\ &\text{with probability} \geq 1 - 2^{-n+6} \ . \end{aligned} \tag{2}$$

Assuming $\log(n) \geq 3$ and with $r = \sqrt{2n/\pi}$, we have that

$$(9r + 1)^2 \leq \alpha n \text{ with } \alpha = \left(9\sqrt{2/\pi} + 1/\sqrt{2^3}\right)^2 \ . \tag{3}$$

Now, suppose $dn^6 \ln(n) \leq q \leq 2dn^6 \ln(n)$ for some $d \geq 1$. Then

$$\begin{aligned} &\ln(8nq) \leq \ln(16n^8 d) \leq \beta(d) \ln(n), \\ &\text{with } \beta(d) = 8 + \frac{\ln(16) + \ln(d)}{\ln(2^3)} \ . \end{aligned} \tag{4}$$

Then, from (2), (3) and (4), it follows that

$$\begin{aligned} \|2\mathbf{c}''\|_\infty^2 &\leq \left(2\sqrt{2n} \cdot n\sigma(9r + 1)\right)^2 \\ &\leq 8n^3\sigma^2\alpha n \\ &\leq 32\alpha\beta(d)n^6 \ln(n)q \ . \end{aligned}$$

Since $\beta(d) = \mathcal{O}(\ln(d))$, there exists $D \geq 0$ such that for $d \geq D$, $32\alpha\beta(d) \leq d$, and thus $\|2\mathbf{c}''\|_\infty^2 \leq dn^6 \ln(n)q \leq q^2$, and the result follows. We compute the smallest such D by numerical means to be approximately 25830, and it follows that for any $d \geq 25830$ the bound holds. □

Next, for each set of parameters, we calculate a bit security level, based on the distinguishing attack against LWE described in Appendix A. Notice that we do not rely on the worst-case hardness to determine the bit security level of the scheme. This is because it is unknown how tight the worst- to average-case reduction is. We rather analyze the efficiency of the distinguishing attack against the average-case problem underlying pNE. We acknowledge that this approach does not imply that those worst-case lattice problems are hard. However, it provides a plausible estimate for the practical hardness of the average-case problem.

The running time t_ϵ of the distinguishing attack in (1) depends on the desired advantage ϵ. Since an adversary can choose ϵ within a reasonable range, we define the total time of an attack as

$$T = \min\{t_\epsilon/\epsilon \mid \epsilon \in (2^{-80}, 1)\},$$

which we approximate numerically.

From the total time of an attack, we then compute a bit security level b, following the methodology of Howgrave-Graham [19]. Note that the attack time described in Appendix A was estimated by Lindner and Peikert on a 2.3 GHz PC [23]. Assuming that a single block-cipher encryption takes 500 clock cycles, it would take $2^b \cdot \frac{500}{2.3\times 10^9}$ seconds to attack a b-bit block-cipher using brute-force. From this, we obtain that the bit security of a cryptosystem, that can be attacked in no less than T seconds, is given by

$$b = \log(T) + \log\left(2.3 \times 10^9\right) - \log(500).$$

Table 1 shows the bit security level for each set of parameters, as well as the distinguishing advantage and corresponding Hermite factor that minimizes the total attack time.

4 Instantiation and Performance of pNE

In this section we first briefly describe our implementation of pNE, then we present experimental data on its performance, and finally we compare pNE with both NTRUEncrypt and LPR-LWE.

We implemented pNE in C++ using the Number Theory Library (NTL, [34]) for arithmetic in R_q together with the GNU Multiple Precision Arithmetic Library (GMP, [16]) for large integer arithmetic. NTL uses the fast Fourier transform (FFT) for multiplication in the ring R_q. All experiments were performed on a Sun XFire 4440 server with 16 Quad-Core AMD Opteron(tm) Processor 8356 CPUs running at 2.3GHz, having 64GB of memory and running 64bit Debian 7.1. For our experiments we only used one of the 16 cores. We compiled our implementations using GCC v4.7.2-5, NTL v5.5.2-2, and GMP v2:5.0.5+dfsg-2.

In key generation of pNE we must check that $\mathbf{f}$ and $\mathbf{g}$ are invertible in R_q. This is done by choosing $\mathbf{f}, \mathbf{g}$ uniformly at random from R_q and using the native GCD implementation of NTL to test their invertibility. Lemma 10 in [35] proves that the "resample rate" is less or equal to n/q. Our experiments confirm that the resample rate is very small ($< 1/1000$) for our choice of parameters.

Besides R_q arithmetic, the main challenge for implementing pNE is instantiating the Gaussian sampler used in key generation and encryption. We implemented the methods listed in Section 2.2 or adapted provided source code, where available, and tested the implementations in terms of memory size and speed. We also considered two variants of rejection sampling, namely computing ρ_σ on demand and precomputing all possible values of ρ_σ. First, we tested the methods for the rather small value $r = \sqrt{2n/\pi}$ of the Gaussian parameter used in the ring-LWE noise distribution $\chi = D_{\mathbb{Z}^n,r}$ in pNE's KeyGen function. From Table 2, which shows timings and storage requirements for this setting, the Knuth-Yao algorithm appears to be the most efficient algorithm regarding speed. Second, for the much larger value $\sigma \approx 2^8 n^4 \ln^2(n)$ in pNE's KeyGen algorithm, the only method suitable is rejection sampling with ρ_σ computed on demand due to its minimalist storage requirement.

Table 2. Experimental comparison of discrete Gaussian sampling techniques for parameter $\sigma = \sqrt{2n/\pi}$. For each dimension n and each method, the table shows running time in milliseconds and storage in kilobytes. For all n, we used the same Ziggurat with 8192 rectangles in regard to experiments in [7], and for Knuth-Yao we used a precision of 128 bits.

Parameters		Rej. on-demand		Rej precomp.		Inv. CDF		Knuth-Yao		Ziggurat	
n	σ	time	storage	time	storage	time	storage	time	storage	time	storage
1024	25.53	149	0	2.60	4.09	1.07	4.09	0.56	16.47	1.92	262.21
2048	36.11	437	0	6.86	6.36	1.98	6.36	1.03	50.97	2.42	262.21
4096	51.06	1200	0	19.66	9.80	4.04	9.80	2.04	78.69	7.03	262.21

In the remainder of this section we present experimental data on the performance of pNE and compare it to NTRUEncrypt and LPR-LWE's. Table 3 shows timings and sizes for our implementation of pNE.

Table 3. Experimental performance of pNE. For a given set of parameters, column seven shows public key, secret key and ciphertext size in kilobytes, and column eight shows the ciphertext to plaintext ratio. Columns nine to eleven show the running times for KeyGen, Encrypt, and Decrypt in milliseconds, respectively.

Parameters						Sizes [kB]		Running times [ms]		
n	$\log q$	$\log \sigma$	r	bit sec	err rate	pk = sk = ct	ct/pt	KeyGen	Encrypt	Decrypt
1024	71.90	49.89	25.53	38	$O(2^{-n})$	9.22	72	763	5.60	4.12
2048	77.28	53.63	36.11	144	$O(2^{-n})$	19.97	78	1731	12.09	9.51
4096	83.30	57.70	51.06	338	$O(2^{-n})$	43.01	84	3820	26.70	21.37

For comparison, we collected recent figures about NTRUEncrypt in the literature, and we present them in Table 4 (see also Figure 1 for a comparison with pNE and LPR-LWE).

Table 4. Security and performance of NTRUEncrypt with $q = 2048$ and $p = 3$ on a 2GHz CPU. Security estimates were taken from [18] and efficiency measures were provided in private communication by William Whyte of Security Innovation Inc.

		Sizes [kB]				Running times [ms]	
n	bit sec	pk	sk	pt	ct/pt	Encrypt	Decrypt
401	112	0.55	0.20	0.10	11	0.09	0.19
439	128	0.60	0.22	0.11	11	0.10	0.20
743	256	1.02	0.37	0.19	11	0.20	0.40

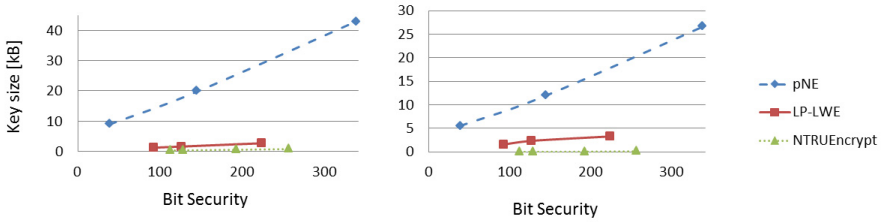

Fig. 1. Encryption running time and public key size against bit security for pNE, NTRUEncrypt and LPR-LWE (parameters as in Tables 3, 4 and 6)

Our pNE implementation is more than 100 times slower and requires 24 times larger keys than an implementation of NTRUEncrypt by Security Innovation Inc. We consider this gap too large to be overcome through optimization of pNE's implementation.

We then move on to compare pNE to the ring-LWE based scheme of Lyubashevsky, Peikert, and Regev [25] (LPR-LWE), which appears to be the most efficient provably secure lattice-based PKE scheme [15]. In order to obtain comparable data, we adapted the implementation by Göttert et al. [15] so as to make it as close as possible to our implementation of pNE. In particular, we use the same Gaussian sampler and the same library for polynomial arithmetic. Table 5 summarizes the results.

Comparing our implementations of pNE with that of LPR-LWE, we conclude that LPR-LWE is significantly more efficient. Take for example pNE with $n = 2048$ which offers 144 bit security and LPR-LWE with $n = 256$ which offers 151 bit security. The public key of pNE is 26 times larger and the secret key 52 times larger. Moreover, key generation of pNE is over 1000 times slower than

Table 5. Experimental performance of LPR-LWE with parameters as proposed by Lindner and Peikert [23]

Parameters					Sizes [kB]			Running times [ms]		
n	q	s	bit sec	error rate	pk = ct	sk	ct/pt	KeyGen	Encrypt	Decrypt
192	4093	8.87	100	1%	0.58	0.29	12	0.79	1.25	0.49
256	4093	8.35	151	1%	0.77	0.38	12	0.98	1.52	0.59
320	4093	8.00	199	1%	0.96	0.48	12	1.40	2.25	0.92

that of LPR-LWE, encryption is over 7 times slower and decryption over 15 times slower.

There are two caveats to this apparently disproportionate difference between pNE and LPR-LWE. First, the error rate of LPR-LWE is very high, at around 1%, while the error rate of pNE is negligible. A 1% error rate could be problematic in a realistic deployment. Second, the Gaussian parameter s in LPR-LWE is small.[5] The worst- to average-case reduction requires $s \geq 2\sqrt{n}$. Moreover, values of s below $\sqrt{n}$ lead to subexponential attacks [2] (see also [10] for other attacks for bounded distribution).[6]

In order to provide a more fair comparison between pNE and LPR-LWE, we computed parameters that guarantee negligible error rate and the worst- to average-case reduction to hold. We follow a methodology adapted from [23] to setup the parameters. We fix n, set $s = 2\sqrt{n}$ and $\delta = 2^{-n}$. Then we find $c > 1$ such that $c \cdot \exp\left((1 - c^2)/2\right) = 1/2$. We then choose the smallest prime q greater than $4cs^2\sqrt{n \ln(2/\delta)}/(\sqrt{2} \cdot \pi)$. These choices guarantee negligible error rate. Finally, we calculate security based on the distinguishing attack as we did for the pNE scheme.

Table 6. Experimental performance of LPR-LWE with conservative parameters that guarantee negligible error rate and the worst- to average-case reduction to hold

Parameters					Sizes [kB]			Running times [ms]		
n	q	s	bit sec	error rate	pk = ct	sk	ct/pt	KeyGen	Encrypt	Decrypt
256	378353	32.00	92	$O(2^{-n})$	1.22	0.61	19	1.02	1.56	0.59
320	590921	35.77	126	$O(2^{-n})$	1.60	0.80	20	1.46	2.36	0.92
512	1511821	45.25	223	$O(2^{-n})$	2.69	1.34	21	2.09	3.29	1.16

Table 6 summarizes the results. It shows parameters that provide low, medium, and high levels of security, as well as experimental data on storage requirements and running times. Although these results show a narrower gap between pNE

[5] In order to be consistent with the notation in [23], here s is the Gaussian parameter for $\rho_s(\mathbf{x}) = \exp\left(-\pi \|\mathbf{x}\|^2 / s^2\right)$ (see Section 2.2).

[6] This is true despite recent results showing the LWE problem hard for small parameters [26], because LPR-LWE does not meet the requirements of the new results [8].

and LPR-LWE, a significant difference in favor of LPR-LWE persists. This difference can also be seen in Figure 1, which depicts encryption running time and public key size against bit security for pNE, LPR-LWE and NTRUEncrypt.

On a recent paper, Bos et al. [4] analyze the efficiency of a leveled homomorphic encryption scheme based on pNE, call it H-pNE. It is difficult to derive a conclusion about pNE's efficiency from the single measurement of their related scheme. However, their results seem to indicate that H-pNE's efficiency is competitive to that of other provably secure ring-LWE homomorphic encryption schemes. We claim that this does not contradict our findings. They report that their H-pNE implementation is about ten times faster than an implementation of a scheme by Brakerski and Vaikuntanathan [6], which is closely related to LPR-LWE. They justify the performance increase partially on better hardware and an optimized implementation. But there are two more factors they do not mention. First, in order to allow homomorphic operations, one must choose a large modulus q to allow additional noise growth. This is true for a homomorphic version of pNE as well as for a homomorphic version of other schemes. Thus, they are comparing schemes with equally high modulus q, while in our analysis we compare to an instance of LPR-LWE with a much smaller q. This means that, while a homomorphic variant of pNE might be comparatively efficient, pNE is less efficient as a stand-alone encryption scheme. The second factor is that the parameter choices they highlight allow for a single multiplicative level of H-pNE, against four levels for the scheme by Brakerski and Vaikuntanathan. This asymmetry is not discussed in the paper by Bos et al.

5 Efficiency Analysis of pNE

We have seen in Section 4 that pNE is less efficient than LPR-LWE or NTRUEncrypt. In this section we analyze why this is the case.

The size of pNE's public key, secret key and ciphertext is given by $n \lceil \log q \rceil$ bits, which is the space required to store one element of R_q. The ciphertext to plaintext ratio is given by $\lceil \log q \rceil$ because a plaintext is encoded into an element of $R_p = R_2$ which stores up to n bits and it is encrypted into an element of R_q of size $n \lceil \log q \rceil$.

In order to better understand the running time of pNE we run experiments for $\log(n) = 4, \ldots, 16$. The results presented in Figure 2 seem to indicate that the running time of the KeyGen, Encrypt and Decrypt algorithms is proportional to $n \log(n) \log(q)$. Actually, we found a strong correlation between their running time and $n \log(n) \log(q)$ in our experiments (Pearson product moment correlation coefficient $r > 0.999$). This confirms that polynomial multiplication and division can be performed in $O(n \log n)$ scalar operations using FFT.

Table 7 shows a breakdown of the running time of key generation, encryption and decryption into their most time-consuming subroutines. The table shows that close to 90% of key generation is spent sampling $\mathbf{f}$ and $\mathbf{g}$ according to a discrete Gaussian distribution, while computing $\mathbf{h}$ takes around 10% of the time.

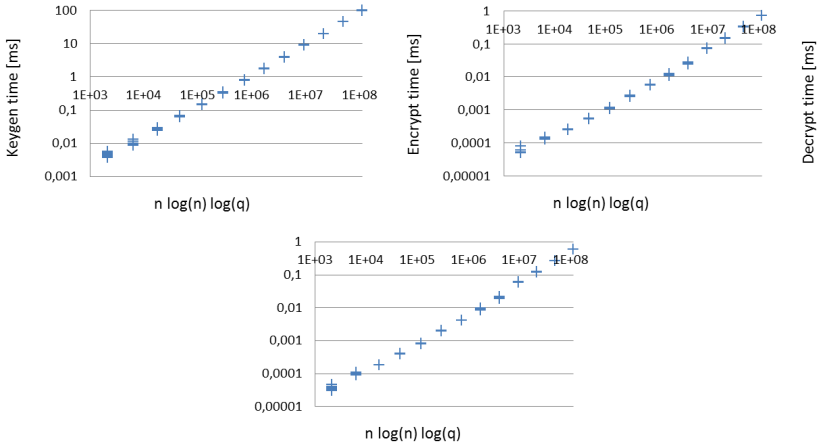

Fig. 2. Running time in milliseconds of pNE's KeyGen, Encrypt and Decrypt versus $n \log(n) \log(q)$, for $\log(n) = 4, \ldots, 16$, respectively

For encryption the tendency is reversed, with little under 20% of the total time spent on sampling Gaussian elements, while computing the ciphertext $\mathbf{c}$ takes a little bit over 80% of the time. This is in part thanks to the efficiency of the Knuth-Yao sampler, which is not possible to be used in key generation. In the case of decryption, the operations in R_q take more than 90% of the time, while the reduction mod p only takes around 5%.

Table 7. Running time breakdown of pNE

	Key Generation		Encryption		Decryption	
n	sampling	arithmetic	sampling	arithmetic	arithmetic	mod p reduction
1024	90.10%	9.90%	18.71%	81.29%	93.83%	6.17%
2048	89.71%	10.29%	16.16%	83.84%	93.65%	6.35%
4096	89.30%	10.70%	14.63%	85.37%	95.39%	4.61%

Although efficiency improvements are certainly possible, the gap between pNE and the other two schemes is too large to be surmounted by optimization alone. In order to understand why this gap is so large we take a closer look at the schemes. The only differences between pNE and NTRUEncrypt are:

i) Operations are performed modulo $x^n + 1$ instead of $x^n - 1$.
ii) The integer modulus q is chosen to be a prime instead of a power of 2.
iii) The secret key polynomials $\mathbf{f}$ and $\mathbf{g}$ are sampled from a discrete Gaussian distribution with parameter $\sigma = 2n\sqrt{\ln(8nq)q}$ instead of being sampled uniformly at random from a set of small norm polynomials.

iv) The ciphertext is $\mathbf{c} = \mathbf{he} + p\mathbf{e}' + \mathbf{m}$ with $\mathbf{e}, \mathbf{e}'$ sampled from a discrete Gaussian distribution with parameter $r = \sqrt{2n/\pi}$, instead of $\mathbf{c} = \mathbf{h}\phi + \mathbf{m}$ with ϕ sampled uniformly at random from a set of small norm polynomials.

Differences i), ii), and iv) cannot be decisive because they are features of LPR-LWE as well, and the latter is much more efficient than pNE. We argue that the main source of inefficiency lays on difference iii). More precisely, efficiency of pNE is hampered in several ways by the large value of σ required to make the public key statistically close to uniform. As shown in the proof of Lemma 3, a large σ induces a large modulus q to allow correct decryption. A large q directly impacts the running times of KeyGen, Encrypt and Decrypt, as they are all proportional to $n\log(n)\log(q)$, as well as it impacts the key sizes, which are proportional to $n\log(q)$.

But large q has an even more decisive negative effect on practical security. From the equations in Appendix A, one can conclude that the running time t of the distinguishing attack depends on q as $\log(t) = \mathcal{O}(1/\log(q))$. Thus, a large q makes the scheme less secure, forcing a large dimension n to obtain a given security level. A large n has an even more dramatic effect on the efficiency of the cryptosystem. Table 8 illustrates the effect of the parameter σ on the efficiency of pNE.

Table 8. Efficiency and security measures of pNE for different values of the parameter σ. Bit security is calculated based on the distinguishing attack and does not take into consideration attacks that may exploit the departure from the security proof.

Parameters				Sizes [kB]		Running times [ms]		
n	$\log(\sigma)$	$\log(q)$	bit sec	pk = sk = ct	ct/pt	KeyGen	Encrypt	Decrypt
512	49	69.22	-22	4.48	70	336	2.58	2.00
512	29	49.71	10	3.20	50	320	2.20	1.59
512	9	29.37	103	1.92	30	305	1.79	1.21
1024	49	71.90	38	9.22	72	759	5.63	4.13
1024	29	51.99	98	6.66	52	737	4.88	3.43
1024	9	31.24	265	4.10	32	689	3.73	2.42

The negative effect that a "wide" Gaussian has on the security of pNE seems counterintuitive as one would expect that a larger key space improves security. Yet, it is the main force dragging pNE's efficiency. It is unclear to us at the moment whether this can be improved while preserving pNE's strong security guarantee —its worst- to average-case reduction.

Acknowledgements. This work was partially supported by CASED (`www.cased.de`). We would like to thank Rachid Elbansarkhani, Andreas Hülsing, Michael Schneider, Damien Stehlé, William Whyte and zgr Dagdelen for useful discussions.

References

1. Albrecht, M.R., Cid, C., Faugère, J.C., Fitzpatrick, R., Perret, L.: On the complexity of the BKW algorithm on LWE. Cryptology ePrint Archive, Report 2012/636 (2012), http://eprint.iacr.org/2012/636/
2. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 403–415. Springer, Heidelberg (2011)
3. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. J. ACM 50(4), 506–519 (2003)
4. Bos, J., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Stam, M. (ed.) IMACC 2013. LNCS, vol. 8308, pp. 45–64. Springer, Heidelberg (2013)
5. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Proceedings of the 45th Annual ACM Symposium on Theory of Computing, STOC 2013, pp. 575–584. ACM, New York (2013)
6. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
7. Buchmann, J., Cabarcas, D., Göpfert, F., Hülsing, A., Weiden, P.: Discrete ziggurat: A time-memory trade-off for sampling from a gaussian distribution over the integers. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 402–417. Springer, Heidelberg (2014)
8. Cabarcas, D., Göpfert, F., Weiden, P.: Provably secure LWE-encryption with uniform secret. Cryptology ePrint Archive, Report 2013/164 (2013), http://eprint.iacr.org/2013/164
9. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011)
10. Ding, J.: Solving LWE problem with bounded errors in polynomial time. Cryptology ePrint Archive, Report 2010/558 (2010), http://eprint.iacr.org/2010/558/
11. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 40–56. Springer, Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-40041-4_3
12. Galbraith, S.D., Dwarakanath, N.C.: Efficient sampling from discrete Gaussians for lattice-based cryptography on a constrained device (2012), preprint available at http://www.math.auckland.ac.nz/~sgal018/gen-gaussians.pdf
13. Gentry, C., Halevi, S., Vaikuntanathan, V.: A simple BGN-type cryptosystem from LWE. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 506–522. Springer, Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-13190-5_26
14. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th Annual ACM Symposium on Theory of Computing, pp. 197–206. ACM Press (May 2006)
15. Göttert, N., Feller, T., Schneider, M., Buchmann, J., Huss, S.A.: On the design of hardware building blocks for modern lattice-based encryption schemes. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 512–529. Springer, Heidelberg (2012)

16. Granlund, T.: The GNU multiple precision arithmetic library, http://gmplib.org/
17. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)
18. Hoffstein, J., Pipher, J., Whyte, W.: A note on hybrid resistant parameter selection for NTRUEncrypt (2010) (unpublished)
19. Howgrave-Graham, N.: A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 150–169. Springer, Heidelberg (2007)
20. Kaps, J.P.: Cryptography for Ultra-Low Power Devices. Ph.D. thesis, Worcester Polytechnic Institute (2006)
21. Karney, C.F.F.: Sampling exactly from the normal distribution. Tech. rep., SRI International (March 2013), http://arxiv.org/abs/1303.6257
22. Knuth, D.E., Yao, A.C.: The complexity of non uniform random number generation. In: Algorithms and Complexity: New Directions and Recent Results, pp. 357–428 (1976)
23. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011)
24. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: An update. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 293–309. Springer, Heidelberg (2013)
25. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010)
26. Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 21–39. Springer, Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-40041-4_2
27. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D.J., Buchmann, J.A., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 147–191. Springer (2008)
28. Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011, pp. 113–124. ACM, New York (2011)
29. O'Rourke, C., Sunar, B.: Achieving NTRU with Montgomery multiplication. IEEE Transactions on Computers 52(4), 440–448 (2003)
30. Peikert, C.: An efficient and parallel Gaussian sampler for lattices. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 80–97. Springer, Heidelberg (2010)
31. Plantard, T., Susilo, W., Zhang, Z.: Lattice reduction for modular knapsack. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 275–286. Springer, Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-35999-6_18
32. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th Annual ACM Symposium on Theory of Computing, pp. 84–93. ACM Press (May 2005)
33. Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical Programming 66, 181–199 (1994)
34. Shoup, V.: Number theory library (NTL) for C++, http://www.shoup.net/ntl/
35. Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 27–47. Springer, Heidelberg (2011)

36. Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 617–635. Springer, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-10366-7_36

A Distinguishing Attack

For the sake of simplicity we will consider the distinguishing attack only against LWE, and not against ring-LWE. In LWE, the samples are constructed as (A, b) with $b = A^\top s + e$, where $A \leftarrow \mathbb{Z}_q^{n \times m}$ and $s \leftarrow \mathbb{Z}_q^n$ are chosen uniformly at random. The ring-LWE problem can be seen as a variant of LWE in which the matrix A has a special structure, where the structure depends on the first part of the ring-LWE samples $\mathbf{a}$ and the underlying ring R_q. We assume that the distinguishing attack does not take advantage of this special structure.

The bottleneck of the distinguishing attack is the computation of a short vector in the lattice

$$\Lambda_q^\perp(A) := \{y \in \mathbb{Z}^m \mid Ay \equiv 0 \bmod q\}.$$

If the distribution χ is a discrete Gaussian, then the advantage of the distinguishing attack is close to $\epsilon = \exp(-2(\pi c r/q)^2)$, where c is the length of the shortest vector the adversary is able to find, and r is the parameter of the discrete Gaussian distribution [23]. The shortest vectors are of length $\delta^m q^{n/m}$, where δ is the so called *Hermite factor*. This length is minimized for $m = \sqrt{n \log(q)/\log(\delta)}$; thus, the shortest vectors we can expect to produce are of length $2^{2\sqrt{n \log(q) \log(\delta)}}$.

The running time of state-of-the-art lattice reduction algorithms (e.g. BKZ) is determined by the Hermite factor δ. In order to obtain an advantage greater or equal to ϵ we need to be able to compute vectors of length less or equal to

$$c_\epsilon = \frac{q}{\pi r}\sqrt{\frac{\ln(1/\epsilon)}{2}}\ ,$$

for which we require a Hermite factor not greater than

$$\delta_\epsilon = 2^{\frac{\log^2(c_\epsilon)}{4n \log q}}.$$

Lindner and Peikert [23] heuristically estimate that the running time of the distinguishing attack using BKZ is

$$\log(t_\epsilon) = 1.8/\log(\delta_\epsilon) - 110\ .$$