Even More Practical Key Exchanges for the Internet using Lattice Cryptography

Vikram Singh[*]
Arjun Chopra[†]

Abstract

In 2014, Peikert described the first practical lattice-based key exchange that is provably secure and provides perfect forward security. However, his presentation lacks concrete proposals for parameters. We aim to provide a clear description of how the algorithm can be implemented along with some analysis for potential parameters.

Previously in 2015, Singh considered the simpler case, as chosen by Bos, Costello, Naehrig and Stebila in 2014, of cyclotomic rings with power-of-two degree. In this work we focus on the case of cyclotomic rings with degree $p-1$ for prime p. This allows for a greater degree of flexibility in choosing lattice dimension, which determines the security level and efficiency of the scheme. We describe the necessary arithmetic setup and then present Peikert's Diffie-Hellman-like key exchange along with security, correctness and implementation analysis.

Keywords: Cryptography, Lattice, Ring-LWE, Ring Learning With Errors, Key Exchange, IKE, TLS

1 Introduction

Lattice-based primitives, and schemes based on ring learning with errors (ring-LWE) [10, 11] in particular, are increasingly viewed as offering the most promising post-quantum alternatives to classical Diffie-Hellman key exchanges or RSA key transport. They are efficient in terms of both computation and key size and provide strong provable security guarantees against classical and quantum adversaries. Most importantly, they have been shown to work within real-world protocols such as Internet Key Exchange (IKE) and Transport Layer Security (TLS) [14, 3].

The scheme presented by Peikert [14] is a Diffie-Hellman-like key exchange based on ring-LWE over a cyclotomic ring R. Alice chooses a private key $s_0 \in R$ and error term $e_0 \in R$, both sampled from a "small" distribution, and uses these to construct her public key $b = a \cdot s_0 + e_0$. If Bob's public key is $c = a \cdot s_1 + e_1$ then they can arrive at an approximate shared secret

[*] vs77814@gmail.com. Principal Consultant, VS Communications.

[†] arjun.chopra.vsc@outlook.com. Associate Consultant, VS Communications.

$s_0 \cdot c = s_0 \cdot a \cdot s_1 + s_0 \cdot e_1 \approx s_0 \cdot a \cdot s_1 + s_1 \cdot e_0 = s_1 \cdot b$ since $s_0 \cdot e_1 - s_1 \cdot e_0$ is small. To convert this approximate agreement into an *exact* agreement, Bob sends Alice an additional string of "masking bits". These masking bits are fed into a "reconciliation" technique which allows Alice to recover enough of Bob's approximate shared secret so that they can both derive an exact shared secret.

Although intended to demonstrate the practicality of ring-LWE, Peikert's description of his key agreement was still at a quite abstract level. Consequently, in [15] we considered the concrete case of power-of-two cyclotomic rings and were able to hide many of the complexities of ring arithmetic and give a cleaner explanation of the system. We now extend our analysis to prime cyclotomic rings where there is a much greater range of potential parameter choices available. This allows smaller keys than the power-of-two case [3, 15] while still avoiding the complication incurred for general cyclotomics. See [11] for the fully general case.

In Section 2 we establish the necessary setup, defining subgaussian random variables, the ring-LWE problem, and the cyclotomic rings we shall use including the power, decoding, and Chinese Remainder bases. We define the error distribution we shall use to randomly draw small ring elements. In Section 3 we sketch the key exchange and describe how masking bits and the reconciliation mechanism are used to generate key from the approximate shared secrets. The reconciliation mechanism is a slight variation on Peikert's, which is cleaner but has the same effect.

In Section 4 we present Peikert's basic key exchange algorithm in detail in the context of our implementation choices. We refer the reader to our earlier work, [15], for a presentation of Peikert's development of provably actively secure key exchanges in order to protect against an adversary that can choose ciphertexts, and his development of provably secure authenticated key exchanges in order to assure each party that the key is agreed only with the holder of the desired certified identity.

In Section 5 we describe our parameter choices along with a consideration of their security, an analysis of correctness, and some initial findings on implementing the schemes. We suggest that our parameters are both more efficient and more secure than any previous work. We conclude in Section 6. Sample `sage` code is available in the Appendix of [15].

2 Preliminaries and definitions

For any integer q, let $\mathbb{Z}_q$ denote the quotient ring $\mathbb{Z}/q\mathbb{Z}$, *i.e.* the ring of integers modulo q. The elements of this ring can be considered in terms of a distinguished set of representatives, *e.g.* the set $\{-(q-1)/2, \ldots, 0, \ldots, q-1\}$.

For any two subsets X, Y of an additive group, we define $-X := \{-x : x \in X\}$ and $X + Y := \{x + y : x \in X, y \in Y\}$. Similarly, we define $x + Y := \{x + y : y \in Y\}$ for a fixed element x.

We define the *infinity norm* on a ring R with basis $Y = \{y_j\}$ to be $||r||_\infty := \max_j(r_j)$ for $r = \sum_j r_j \cdot y_j \in R$. We will be interested in the growth of individual basis coefficients of elements of R and will perform our analysis on this infinity norm $|| \cdot ||_\infty$ rather the Euclidean norm $|| \cdot ||_2$ as this slows for tighter bounds.

2.1 Gaussian and subgaussian distributions

Early exposition of ring-LWE ([10]) relied on Gaussian distributions for all error sampling. For $r > 0$, the Gaussian distribution D_r over $\mathbb{R}$ with parameter r has probability distribution function $\frac{1}{r}e^{(-\pi x^2/r^2)}$. More recent works [11, 14] have updated these ideas to be based on the notion of a *subgaussian distribution.* For any $\delta \geq 0$, we say that a random variable X (or its distribution) over $\mathbb{R}$ is δ*-subgaussian* with parameter $r > 0$ if for all $t \in \mathbb{R}$, the (scaled) moment-generating function satisfies

$$\mathbb{E}[\exp(2\pi t X)] \leq \exp(\delta) \cdot \exp(\pi r^2 t^2).$$

Subgaussians are useful to simplify the analysis and have many nice features, such as the sum of independent subgaussians is another subgaussian whose parameters can be calculated. Most importantly, as noted in [14], any B-bounded centered random variable X (i.e., $\mathbb{E}[X] = 0$ and $|X| \leq B$ always) is 0-subgaussian with parameter $B\sqrt{2\pi}$. Thus simple bounded distributions, such as uniform random from an interval, are subgaussian. We will utilise this fact to provide significant efficiency gains and simplification to our key establishment.

2.2 Ring-LWE

We now recall the ring-LWE probability distribution and (decisional) problem as presented in [14]; see [10] for a more general form.

Definition 1 (Ring-LWE Distribution)**.** For an $s \in R$ and a distribution χ over R, a sample from the Ring-LWE Distribution $A_{s,\chi}$ over $R_q \times R_q$ is generated by choosing $a \leftarrow R_q$ uniformly at random, choosing $e \leftarrow \chi$, and outputting $(a, b = a \cdot s + e)$.

Definition 2 (Ring-LWE Decision)**.** The *decision* version of the ring-LWE problem, denoted $R\text{-}\mathsf{DLWE}_{q,\chi}$,is to distinguish with non-negligible advantage between independent samples from $A_{s,\chi}$, where $s \leftarrow \chi$ is chosen once and for all, and the same number of *uniformly random* and independent samples from $R_q \times R_q$.

The main theorem of [10] can be stated informally as follows:

Theorem 1. Suppose that it is hard for polynomial-time quantum algorithms to approximate (the search version of) the shortest vector problem (SVP) in the worst case on ideal lattices in R to within a fixed poly(n) factor. Then any poly(n) number of samples drawn from the R-LWE distribution are pseudorandom to any polynomial-time (possibly quantum) attacker.

This tells us that distinguishing the Ring-LWE Distribution from random or recovering its secret is hard, provided SVP is hard.

A main benefit of ring-LWE over traditional LWE and other lattice-based techniques is in efficiency. Whereas LWE requires the use of $\Omega(n)$ samples $(a_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, for ring-LWE, typically only a very small number of samples $(a, b) \in R \times R$ are used.[1] Essentially, ring-LWE offers a compact representation for the lattice in question. The reason for this is that each polynomial $v \in R$ represents n vectors in the lattice, one for each multiple $v \cdot x^i$ for $i \in \{0, \ldots, n-1\}$.[2] If a is fixed for all users, then the public key is just the element $b \in R$. We see that this results in smaller public keys for ring-LWE because LWE would have $\Omega(n^2)$ modular values whereas ring-LWE can have only n.

2.3 Cyclotomic rings

Let $R := \mathbb{Z}[\zeta_m] \cong \mathbb{Z}[x]/\langle\Phi_m(x)\rangle$ be the mth cyclotomic ring, where Φ is defined by $x^m - 1 = \prod_{d|m} \Phi_d(x)$ and ζ_m is a primitive mth root of unity. The degree of R is the degree of Φ, which is given by the Euler totient function $n := \phi(m)$. We shall mainly be interested in the quotient ring $R_q := R/qR \cong \mathbb{Z}_q[\zeta_m]$ where all of our operations can be defined.

In order to access a larger set of potential parameter options, in this paper we will focus on the special case m prime. In this case, $n = \phi(m) = m - 1$ and $\Phi_m(x) = 1 + x + x^2 + \cdots + x^n$. We shall utilise prime q such that $q \equiv 1 \pmod{m}$, as is required by the Security Proof stated in Theorem 2.7 of [10]. As in the previous work [15], this also allows arithmetic speed-ups using the Chinese Remainder basis: the ideal $< q >:= qR$ factors as $m-1$ prime ideals which provide the orthogonal components required by the Chinese Remainder Theorem. For the remainder of this paper, we consider m and q of this form unless explicitly stated. We also drop the subscript m on ζ_m.

2.3.1 Bases for cyclotomic rings

In [11], multiple bases for the cyclotomic ring R are defined. As we choose to work with m prime, the power and powerful bases will coincide. Here we will describe the power, decoding, and Chinese remainder bases.

The power basis for R or R_q is the natural basis of powers of ζ:

$$\vec{p} := (\zeta^i)_{i=0,\ldots,n-1}$$

[1] A careful statement of the above worst-case hardness result shows that it deteriorates with the number of samples; fortunately, all our applications require only a small number of samples.

[2] In ring-LWE, each ring element actually represents a so-called *ideal lattice*: a lattice L is an ideal lattice if for all $\mathbf{v} \in L$, $x \cdot \mathbf{v}$ is also in L. Thus one polynomial in R defines a space of n vectors in L.

The decoding basis $\vec{d}$ for the ring of integers R is defined in [11] as the scaled dual of the conjugate power basis. An equivalent and more concrete definition in our case is

$$d_i := \sum_{j=i}^{n-1} \zeta^j \text{ for } i \in \{0, \ldots, n-1\}.$$

By inverting the definition, we have an $O(n)$-complexity change of basis mapping coefficients from the power to the decoding basis of R:

$$\vec{p} \cdot \mathbf{a} \quad = \quad \tilde{\mathbf{d}} \cdot \begin{pmatrix} a_0 \\ a_1 - a_0 \\ a_2 - a_1 \\ \vdots \\ a_{n-1} - a_{n-2} \end{pmatrix}$$

If we write $a \in R$ in the power basis as $a = \vec{p} \cdot \mathbf{a}$, then the canonical embedding[3] defines the Chinese Remainder Theorem matrix CRT as the square matrix

$$\sigma(a) = \mathrm{CRT} \cdot \mathbf{a} = \begin{pmatrix} 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{m-1} & \omega^{(m-1)2} & \cdots & \omega^{(m-1)(n-1)} \end{pmatrix} \cdot \mathbf{a}$$

where the rows are indexed by $\mathbb{Z}_m^*$ and the columns by $\{0, 1, \ldots, n-1\}$ for $n = m-1$. Note that multiplication by CRT can be done by fast Fourier transform methods in $O(n \log n)$ complexity.

In [11], the CRT matrix is defined as a general object over any commutative ring containing a primitive mth root of unity ω. By fixing a specific primitive mth root of unity ζ in $\mathbb{Z}_q$, we can turn the CRT matrix into an important and powerful tool for working in R_q. As q was chosen with $q \equiv 1 \pmod{m}$, the cyclotomic polynomial will factor completely modulo q:

$$x^n + \cdots + x + 1 = (x - \zeta)(x - \zeta^2)(x - \zeta^3) \cdots (x - \zeta^n)$$

for ζ a primitive mth root of unity in $\mathbb{Z}_q$ and $n = m - 1$. Thus, multiplication by CRT gives evaluation of the polynomial at each of the mth roots of unity, that is, at the primitive root ζ and each of its powers. Each polynomial in R_q has a unique representation from evaluating at the roots of unity, so this form can be used as a basis, the Chinese Remainder basis. While addition can be performed component-wise in all bases, the unique thing about the Chinese Remainder basis is that multiplication is also component-wise[4]: for any two polynomials $z, z' \in R_q$ and any $c \in \mathbb{Z}_q$, $z(c) \cdot z'(c) = (z \cdot z')(c)$. Naively multiplication would require summing $O(n^2)$

[3] The canonical embedding $\sigma : R \to H \subset \mathbb{C}^n$ is given by $v \mapsto (v(\omega_m^i))_{i \in \mathbb{Z}_m^\times}$ for $\omega_m := e^{2\pi i/m}$.

[4] Lift z and z' to $\mathbb{Z}_q[x]$, and observe that adding multiples of the cyclotomic polynomial to $z \cdot z'$ does not change the evaluation at any primitive mth root of unity, and this is the only polynomial for which this is true.

cross-products of all pairs of coefficients but use of the Chinese Remainder basis brings that to $O(n)$-complexity.

The Chinese Remainder basis $\vec{c}$ is defined for R_q by

$$\vec{c}^T := \vec{p}^T \cdot \mathrm{CRT}_q^{-1} \text{ , where } \mathrm{CRT}_q := \mathrm{CRT} \pmod q$$

and thus admits $O(n \log n)$ fast Fourier transform method conversion between itself and the power basis. We have shown we can convert between any of these bases and the Chinese Remainder basis in $O(n \log n)$, hence can perform multiplication in $O(n \log n)$.

2.4 Error distributions

In order that each party has some secret information that can be combined in Diffie-Hellman fashion to obtain a shared secret value accessible only to the two parties, it is necessary to produce the short error elements that act as private keys. The aim is to produce errors that are subgaussian with parameters as tight as possible, so that the security parameters of these errors can be nicely controlled once they have been combined to form shared secret values.

Previously ([15]) we generated the error terms by sampling each of the coefficients from a discrete Gaussian with parameter σ or a bounded uniform distribution with bound B. As the power and decoding bases coincide in the power-of-two case there was no ambiguity in which basis to use for the sampling. However, in the prime case we need to choose whether to sample with respect to the power basis or the decoding basis. It transpires that when multiplying error terms the noise growth is worse in the decoding basis (see Section 5.3.2) so we will instead opt for the better correctness properties of the power basis[5].

The discrete Gaussian assigns to each $x \in \mathbb{Z}$ a probability proportional to $e^{-x^2/(2\sigma^2)}$, normalized by the factor $S = 1 + 2\sum_{k=1}^{\infty} e^{-k^2/(2\sigma^2)}$, given by $D_{\mathbb{Z},\sigma}(x) = \frac{1}{S} e^{-x^2/(2\sigma^2)}$. To sample from R_q according to a discretised Gaussian distribution, we use the method in [3, 15]: precompute a lookup table T of size 52 where $T[0] := \lfloor 2^{192}/S \rfloor$ and

$$T[i] := \Big\lfloor 2^{192} \cdot \big(\frac{1}{S} + 2 \sum_{x=1}^{i} D_{\mathbb{Z},\sigma}(x) \big) \Big\rfloor$$

for $i = 1, \dots, 50$, and where $T[51] := 2^{192}$. We sample each coefficient by generating a 192-bit integer t at random, finding the index $\texttt{ind} \in [0, 50]$ such that $T[\texttt{ind}] \leq t < T[\texttt{ind} + 1]$, and then generating an additional random bit for the sign $\texttt{sign} \in \{-1, 1\}$:

[5] This deviates slightly from the sampling method suggested by Ducas and Durmus [4], but it allows us to apply the concrete security estimates from Albrecht et al [1] while still avoiding the Poly-LWE attacks by Elias et al [5].

```
**GaussSample**
Input: standard deviation $\sigma$
Output: error vector $e$
for $i = 1, \ldots, n$
    $t \xleftarrow{\$} (0, 2^{192})$
    $\mathtt{ind}_i = 0$
    repeat $\mathtt{ind}_i$ += 1 until
        $T[\mathtt{ind}_i] \leq t < T[\mathtt{ind}_i + 1]$
    $\mathtt{sign}_i \xleftarrow{\$} \{-1, 1\}$
    $e_i = \mathtt{sign}_i \cdot \mathtt{ind}_i$
$e \leftarrow \vec{p} \cdot (e_1, \ldots, e_n)$
```

As in [15], we will set $\sigma = 8/\sqrt{2\pi} \approx 3.192$ so that the discrete Gaussian $D_{\mathbb{Z}^n,\sigma}$ approximates the continuous Gaussian D_σ extremely well.

In instantiations of ring-LWE, sampling the error terms from Gaussian distributions is typically by far the most expensive operation. As noted in Section 2.2 of [14], uniform sampling from a B-bounded interval is subgaussian with parameter $B\sqrt{2\pi}$, and uniform sampling of each coefficient is both simpler and significantly more efficient:

```
**UniformSample**
Input: bound $B$
Output: error vector $e$
for $i = 1, \ldots, n$
    $e_i \xleftarrow{\$} \{-B, \ldots, B\}$
$e \leftarrow \vec{p} \cdot (e_1, \ldots, e_n)$
```

As the discrete uniform distribution is itself subgaussian and need not attempt to closely approximate a continuous distribution, we have more flexibility in setting the parameter B. In order to provide a fair comparison between the two alternatives, as in [15], we set $B = 5$ so that the standard deviation of the two distributions is approximately equal. Thus we can use a uniform random distribution choosing the coefficients from the set $\{-5, \ldots, 5\}$.

3 Key generation and reconciliation mechanism

In [14], Peikert presents a ring-LWE based Diffie-Hellman-type key exchange algorithm in which two users exchange ring-LWE public keys to arrive at *approximate* or "noisy" agreement on a ring element. In order to (non-interactively) reach *exact* agreement, the second party sends along an additional bit-string (the "masking bits") which can be fed into the "reconciliation" technique from [14] which we will develop in this section.

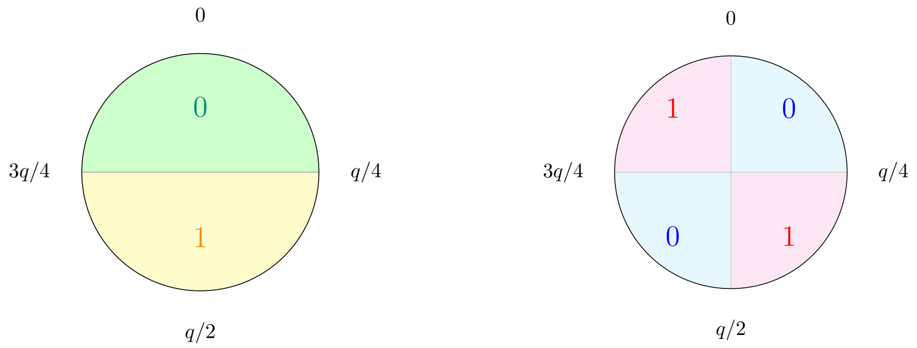
3.1 Description of the exchange

We begin with an informal sketch of how the key exchange works. In the basic key exchange, the first party creates a public key $b = a \cdot s_1 + s_0$ and transmits that to the second party. Upon receipt of the public key b, the second party creates his public key $u = e_0 \cdot a + e_1$ and his version of the approximate shared secret $v = e_0 \cdot b + e_2 = e_0 \cdot a \cdot s_1 + e_0 \cdot s_0 + e_2$. Upon receipt of the public key u and masking bits, the first party forms her version of the shared secret $w = u \cdot s_1 = e_0 \cdot a \cdot s_1 + e_1 \cdot s_1$, and feeds w and the received masking bits into the reconciliation function to recover the key stream. Since s_0, s_1, e_0, e_1, e_2 are all small, $w \approx v$ and the masking bits provide sufficient information for the two parties to exactly agree; however, for adversaries, the masking bits do not give any help in determining what the underlying key bit will be.

Party A		Party B
$pk = b = a \cdot s_1 + s_0$	$\xrightarrow{b}$	
		$pk = u = e_0 \cdot a + e_1$
		$SSV = v = e_0 \cdot b + e_2 =$
		$e_0 \cdot a \cdot s_1 + e_0 \cdot s_0 + e_2$
		key $= \lfloor v \rceil_2$
	$\xleftarrow{(u, \langle v \rangle_2)}$	mask $= \langle v \rangle_2$
$w = u \cdot s_1$		
key $= \mathsf{rec}(w, \langle v \rangle_2)$		

Figure 1: Basic key exchange algorithm.

The key stream that the two users will agree upon will be generated by applying the modular rounding function $\lfloor \cdot \rceil_2$ to each coefficient of the shared secret to round to the closer of 0 or $\frac{q}{2}$. In order for the two parties to achieve exact agreement, the second party also sends over the "masking bits" for his version of the approximate shared secret as generated by the function $\langle \cdot \rangle_2$ which labels which "quadrant" modulo q a coefficient falls into. If an equal number of elements of $\mathbb{Z}_q$ were in each quadrant, then key would be unbiased and the masking bits would give no information about the key. However, q is odd so there is an imbalance. Randomization is used to correct this.



(a) The key stream as generated by the function $\lfloor\cdot\rceil_2$

(b) The masking bits as generated by the function $\langle\cdot\rangle_2$

Figure 2: The key stream and mask bits generating functions.

3.2 Randomized Rounding

For full detail on each of the functions required to generate the key stream and masking bits, the reconciliation function, and proofs of the claims, we refer the reader to [15]. For completeness we provide the necessary definitions and claims here.

We begin by partitioning the elements of $v \in \mathbb{Z}_q$ into four quadrants as

$$\begin{array}{ll} I_0 := \mathbb{Z}_q \cap [0, q/4) & \qquad I_1' := \mathbb{Z}_q \cap [q/4, q/2) \\ I_0' := \mathbb{Z}_q \cap [q/2, 3q/4) & \qquad I_1 := \mathbb{Z}_q \cap [3q/4, q) \end{array}$$

Then we define our *randomized rounding* procedure to make probabilistic nudges on the boundaries of the quadrants in order to balance the relative probabilities of $I_0 \cup I_0'$ and $I_1 \cup I_1'$. This is defined for the two separate cases:

- $\mathbf{q \equiv 1} \pmod{\mathbf{4}}$. $|I_0| > |I_0'| = |I_1| = |I_1'|$ so we need to map out of I_0. If $v = 0$, we draw a uniform random bit and map 0 to either itself or $q-1$ depending on the bit. This moves an element from I_0 to I_1 with 50% probability. Independently, if $v = (q-1)/4$, we map $(q-1)/4$ to either itself or $(q+3)/4$ depending on a random bit, thus moving an element from I_0 to I_1' with 50% probability.

- $\mathbf{q \equiv 3} \pmod{\mathbf{4}}$. $|I_0| = |I_0'| = |I_1'| > |I_1|$ so we need to map into I_1. If $v = 0$, we draw a uniform random bit and map 0 to either itself or $q-1$ depending on the bit. This moves an element from I_0 to I_1 with 50% probability. Independently, if $v = (3q-1)/4$, we map $(3q-1)/4$ to either itself or $(3q+3)/4$ depending on a random bit, thus moving an element from I_0' to I_1 with 50% probability.

This randomized rounding procedure produces a similar effect to the $\mathrm{dbl}(\cdot)$ procedure from [14], without the additional computational burden of mapping all elements of $\mathbb{Z}_q$ into $\mathbb{Z}_{2q}$ and with fewer random bits needing to be sampled. We will denote the randomized rounding of v by $\bar{v}$ and apply it before deriving mask and key bits.

3.3 Reconciliation mechanism

We define the *modular rounding* function $\lfloor \cdot \rceil_2 : \mathbb{Z}_q \to \mathbb{Z}_2$ as

$$\lfloor v \rceil_2 := \lfloor \frac{2}{q} \cdot v \rceil \pmod 2.$$

We will use $\lfloor \bar{v} \rceil_2$ to generate the key stream, which will not be biased due to the use of randomized rounding. As in [14], we define the *cross rounding* function $\langle \cdot \rangle_2 : \mathbb{Z}_q \to \mathbb{Z}_2$ as

$$\langle v \rangle_2 := \lfloor \frac{4}{q} \cdot v \rfloor \pmod 2.$$

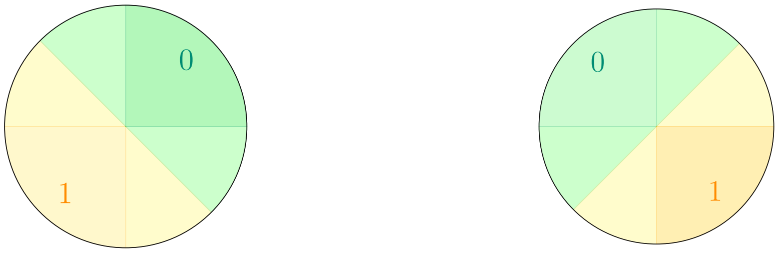
Equivalently, $\langle \bar{v} \rangle_2$ is the $b \in \{0, 1\}$ such that $\bar{v} \in I_b \cup I'_b$. We use $\langle \bar{v} \rangle_2$ to generate the masking bit-string. If v is uniform random in $\mathbb{Z}_q$, then $\langle \bar{v} \rangle_2$ will be biased in $\mathbb{Z}_2$ despite the randomized rounding. Regardless of this bias, however, $\langle \bar{v} \rangle_2$ hides $\lfloor \bar{v} \rceil_2$ perfectly:

Claim 1. If $v \in \mathbb{Z}_q$ is uniform random and randomized rounded as outlined above, then $\lfloor v \rceil_2$ is uniform random given $\langle v \rangle_2$.

Let $E := [-q/8, q/8) \cap \mathbb{Z}$, and define the *reconciliation* function $\mathrm{rec} : \mathbb{Z}_q \times \mathbb{Z}_2 \to \mathbb{Z}_2$ as

$$\mathrm{rec}(w, b) := \begin{cases} 0 & \text{if } w \in I_b + E \pmod q \\ 1 & \text{otherwise.} \end{cases}$$

In effect, the reconciliation function takes the received mask bit b and uses this knowledge of the other party's quadrant to tilt the key stream function $\lfloor \cdot \rceil_2$ from Figure 2 by $q/8$:



(a) The key stream for $b = 0$

(b) The key stream for $b = 1$

Figure 3: The key stream as updated in the reconciliation function.

Provided the difference between shared secrets $|w - \bar{v}|$ is no more than one-eighth q then the shared secrets are sufficiently close that the recipient can infer which quadrant $\bar{v}$ was in, hence infer the key bit $\lfloor \bar{v} \rceil_2$.

Claim 2. If $w = v + e \pmod{q}$ for some $v \in \mathbb{Z}_q$ and $e \in E$, then $\mathrm{rec}(w, \langle v \rangle_2) = \lfloor v \rceil_2$.

We extend the rounding and reconciliation functions to the cyclotomic ring R by applying the functions coordinate-wise to the $\mathbb{Z}_q$-coefficients of the inputs with respect to the power basis. Formally, if $v \in R_q$ has coefficients $v_j \in \mathbb{Z}_q$, then $\lfloor v \rceil_2 := (\lfloor v_j \rceil_2) \in \mathbb{Z}_2^n$ and similarly for $\langle v \rangle_2$. The reconciliation function can be extended to $\mathrm{rec} : R_q \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ by setting $\mathrm{rec}(w, b) := (\mathrm{rec}(w_j, b_j))$ where $w_j \in \mathbb{Z}_q$ are the coefficients of w and $b = (b_j) \in \mathbb{Z}_2^n$.

4 Ring-LWE Key Exchange

In [14], Peikert presents a ring-LWE based Diffie-Hellman-type key exchange algorithm that is provably passively secure, and then shows how to use this basic primitive to construct provably actively secure key exchanges and provably secure authenticated key exchanges. Each of the variants can be presented in the prime setting under consideration in this paper. However, for brevity, we will only present the basic key exchange; see [15] for the other variants.

The passively secure *key encapsulation mechanism* (KEM1) is constructed using the reconciliation technique from Section 3 to allow the two parties to derive an ephemeral key from a pseudorandom value in R_q on which they approximately agree. The parameters for KEM1 will be (m, q, a, χ) where

- m is prime, and so $n = \phi(m) = m - 1$;
- q is an odd prime integer such that $q \equiv 1 \pmod{m}$;
- a is a fixed element of the ring $R_q := \mathbb{Z}/q\mathbb{Z}[\zeta_m] \cong \mathbb{Z}_q[x]/\langle \Phi_m(x) \rangle$; and
- χ is an error distribution over R_q.

We will denote by $\mathbf{Sample}(\chi)$ the function used to sample error terms from χ. For example, this could be $\mathbf{GaussSample}(\sigma)$ for the discrete Gaussian with standard deviation σ, or $\mathbf{UniformSample}(B)$ for the uniform distribution on the interval $\{-B, \ldots, B\}$.

Key generation samples error terms from χ to produce a private key (s_0, s_1), and combines it with the generator a to form the public key. Note that decapsulation only needs s_1 so we can discard s_0.

```
KEM1.Generate
Input: Domain parameters $(m, q, a, \chi)$
Output: Private key $s_1$ and public key $b$
$s_0 \leftarrow \mathbf{Sample}(\chi)$
$s_1 \leftarrow \mathbf{Sample}(\chi)$
$b \leftarrow s_1 \cdot a + s_0 \in R_q$
```

Encapsulation[6] takes the recipient's public key b and produces a shared key μ and a ciphertext c. The shared keys will belong to $\mathcal{K} = \{0,1\}^n$ and their corresponding encapsulations will belong to $\mathcal{C} = R_q \times \mathbb{Z}_2^n$.

```
KEM1.Encapsulate
Input: Recipient's public key $b$
Output: Shared key $\mu$ and encapsulation $c$
$e_0 \leftarrow \mathbf{Sample}(\chi)$
$e_1 \leftarrow \mathbf{Sample}(\chi)$
$e_2 \leftarrow \mathbf{Sample}(\chi)$
$u \leftarrow e_0 \cdot a + e_1 \in R_q$
$v \leftarrow e_0 \cdot b + e_2 \in R_q$
$\bar{v} \leftarrow \mathbf{RandomizedRound}(v)$
$\mu \leftarrow \lfloor \bar{v} \rceil_2 \in \mathbb{Z}_2^n$
$c \leftarrow (u, \langle \bar{v} \rangle_2) \in R_q \times \mathbb{Z}_2^n$
```

Decapsulation forms the approximate shared secret

$$w = u \cdot s_1 = e_0 \cdot a \cdot s_1 + e_1 \cdot s_1 \approx e_0 \cdot a \cdot s_1 + e_0 \cdot s_0 + e_2 = e_0 \cdot b + e_2 = v$$

and uses the reconciliation mechanism from Section 3 to derive the shared secret key μ.

```
KEM1.Decapsulate
Input: Recipient's private key $s_1$ and encapsulation $c$
Output: Shared key $\mu$
$(u, v') \leftarrow c$
$w \leftarrow u \cdot s_1 \in R_q$
$\mu \leftarrow \mathbf{Rec}(w, v')$
```

Lemma 1 (Lemma 4.1 from [14])**.** The KEM1 is IND-CPA secure, assuming the hardness of R-DLWE$_{q,\chi}$ given two samples.

Proof. Analogous to the proof of Lemma 4.1 from [14]. □

[6] The encapsulation and decapsulation functions in [14] include a term g which ours omit. The g is used to control the growth of the error terms in the decoding basis; as we use the power basis it is unnecessary.

5 Analysis

5.1 Instantiating the parameters

We would like to choose concrete parameters that can be tested as practical candidates. Allowing m to be any prime offers much greater flexibility for parameter selection than the power-of-two case. Our aim is to use this flexibility to reduce the public key length while maintaining security and correctness. In order to so, we refer to the guidance in Section 4.4 of [14] on how to set the lattice dimension $n = \phi(m)$ and modulus q.

The security analysis in [14] gives a practical bound on the size of the modulus of $q \approx n^{(3/2)}$. We will also choose $q \equiv 1 \pmod{m}$ to maintain consistency with the proof of security for Ring-LWE. Thus, for a given m we search for the smallest $q > n^{(3/2)}$ that is congruent to 1 modulo m and provides a failure rate of at most 2^{-80}. We then select the pairs (m, q) with the smallest public keys that achieve a range of target security levels between 96 and 256 bits.

						Targets		Public
	m	n	q	σ	B	Security	Correctness	Key Size
I	337	336	32353	3.192	5	96	-80	5040 bits
II	433	432	35507	3.192	5	128	-80	6912 bits
III	541	540	41117	3.192	5	160	-80	8640 bits
IV	631	630	44171	3.192	5	192	-80	10080 bits
V	739	738	47297	3.192	5	224	-80	11808 bits
VI	821	820	49261	3.192	5	256	-80	13120 bits
[15]	1024	512	25601	3.192	5	128	-70	7680 bits
[15]	2048	1024	40961	3.192	5	256	-90	16384 bits
[3]	2048	1024	$2^{32} - 1$	3.192	-	128	-2^{17}	32768 bits

Figure 4: Our parameters compared with previous $m = 2^{\ell}$ parameters.

If we compare these parameter choices with the power-of-two parameters from [15], we can see that we offer options with smaller key sizes for equivalent security targets and provide a larger variety of parameters to meet the needs of different use case scenarios. We also note that the maximum length of the data field of a packet sent over Ethernet is 1500 bytes or 12000 bits. If the public keys are too large, they will not fit into a single packet which increases the possibility of key establishment failures due to fragmentation. Parameter sets **I**–**V** will fit into a single Ethernet packet and offer security ranging from 96 to 224 bits, unlike the 128-bit secure parameters proposed by [3] which would require three maximum sized Ethernet packets. Moreover, each of our parameter sets fit in two Ethernet packets, thus providing greater security with less data overhead than the choice in [3].

5.2 Security

To estimate the security of each parameter set we rely on the analysis from [1], and the associated code [2], which provides approximate costs for a range of different attacks. Although intended for LWE, we will apply it to Ring-LWE by using the approach from [8]. This embeds the Ring-LWE problem in an LWE problem and allows the number of samples to vary. It is possible that this technique may cause issues with attacks which require very large numbers of samples, such as BKW, but these are not among the best attacks on the chosen parameters.

We also update the estimate of the cost of lattice sieving used in the lattice-based attacks. The assumption in [1] is that HashSieve [6] is used which has an asymptotic complexity of $2^{0.3366k+o(k)}$. To set the constant term they quote experimental results from [6] which suggest a cost of $2^{0.45k+12.31}$ cycles for small k. However, more recent results using a parallel version of HashSieve [12] give a least-squares approximation of $2^{0.3377k+17.76}$ cycles for lattice dimensions $86 \leq k \leq 96$. This is a better match for the asymptotic complexity so we will estimate the cost of lattice sieving as $2^{0.3377k+17.76}$ cycles for small k and $2^{0.3366k+17.76}$ cycles more generally.

		MitM			BKW			SIS			Dec			Kannan		
	Target	bop	mem	L	bop	mem	L	bkz2	sieve	L	bop	enum	L	bkz2	sieve	L
I	96	305	301	9	184	176	169	126	116	28	104	88	21	113	**104**	15
II	128	388	384	9	230	222	214	185	150	40	160	144	45	176	**133**	15
III	160	482	478	9	282	273	265	258	184	44	216	200	45	258	**167**	16
IV	192	560	556	9	324	316	308	325	215	52	270	254	45	334	**196**	16
V	224	654	650	10	370	362	353	411	251	59	339	323	46	437	**233**	16
VI	256	725	721	10	408	400	391	481	280	65	395	380	46	521	**261**	16

Figure 5: Attack estimates for parameters with uniform noise ($B = 5$)

		MitM			BKW			SIS			Dec			Kannan		
	Target	bop	mem	L	bop	mem	L	bkz2	sieve	L	bop	enum	L	bkz2	sieve	L
I	96	295	292	9	184	175	168	125	115	28	**102**	86	21	112	103	15
II	128	376	372	9	229	221	213	183	149	40	159	143	45	174	**132**	15
III	160	467	463	9	281	273	264	254	185	48	214	198	45	253	**165**	16
IV	192	542	539	9	323	315	307	321	215	54	266	251	45	330	**194**	16
V	224	633	629	10	369	361	352	407	250	59	336	320	46	431	**231**	16
VI	256	702	698	10	407	399	390	476	277	63	390	374	46	514	**259**	16

Figure 6: Attack estimates for parameters with Gaussian noise ($\sigma = 3.192$)

For the meet-in-the middle (MitM) and BKW attacks, "bops" gives the cost in $\log_2$ cycles, "mem" gives the $\log_2$ memory requirements and L is the $\log_2$ number of LWE samples. For the distinguishing (SIS) and unique SVP (Kannan) attacks, "bkz2" and "sieve" give the cost of using BKZ 2.0 with pruned enumeration and lattice sieving, respectively. For the bounded-distance decoding (Dec) attack, "bops" gives the overall cost of the attack in cycles and "enum" gives the number of enumerations needed in the decoding stage. From Tables 5 and 6 it is clear that the best attacks on our parameters are comfortably above the target security levels.

Since this scheme is designed as a post-quantum proposal, we should additionally consider its security against a quantum attacker. In [7], Laarhoven et al. give a quantum variant of HashSieve with an asymptotic complexity of $2^{0.286k+o(k)}$. We introduce a quantum variant into the code from [2] by utilising this calculated quantum complexity and maintaining the constant as before. While quantum attacks may improve, and perhaps would be expected to once a quantum computer becomes available, these results provide an interesting insight into security in a post-quantum world.

	Security Levels		Security Estimates	
	Classical	Quantum	Classical	Quantum
I	96	90	102	93
II	128	112	132	118
III	160	128	165	146
IV	192	160	194	171
V	224	192	231	202
VI	256	224	259	226

Figure 7: Classical and quantum security for our parameters.

5.3 Correctness of the scheme

For the system described in Section 4, we need each coefficient of the error term to be bounded by $q/8$; that is, we must have that $||s_0e_0 + e_2 - s_1e_1||_\infty < \lfloor q/8 \rfloor$ in order to be guaranteed correctness of the algorithm. This comes from Claim 2 of Section 3, being the maximum difference between shared secret values such that the reconciliation function still produces the correct key. To conduct a direct analysis of the growth of the coefficients of the error term, we will consider the power and decoding bases separately, beginning with a consideration of multiplication in each basis.

5.3.1 Power basis

Multiplication in the power basis is fairly straightforward. Recall that $R := \mathbb{Z}[\zeta] = \mathbb{Z}[x]/\langle\Phi_m(x)\rangle = \mathbb{Z}[x]/\langle x^n + \cdots + x^2 + x + 1\rangle$ for ζ a primitive mth root of unity and $n = \phi(m) = m - 1$. If we multiply $\zeta^i \cdot \zeta^j$ then we just need to reduce modulo the equation $\Phi_m(x) = 1 + x + x^2 + \cdots + x^n$, so $\zeta^n = -1 - \zeta^2 - \cdots - \zeta^{n-1}$. Let $a \in R_q$ be written as $\sum_{i=0}^{n-1} a_i\zeta^i$, and similarly let $b = \sum_{j=0}^{n-1} b_j\zeta^j \in R_q$. The product of any two elements a, b written in the power basis for

R_q is given by

$$\begin{aligned}\sum_{i=0}^{n-1} a_i\zeta^i \cdot \sum_{j=0}^{n-1} b_j\zeta^j &= \sum_{k=0}^{2n-2} \big(\sum_{i+j=k} a_i b_j \big)\zeta^k \\ &= \sum_{k=0}^{n-1} \big(\sum_{i+j=k} a_i b_j \big)\zeta^k + \sum_{i+j=n} a_i b_j \zeta^n + \sum_{k=n+1}^{2n-2} \big(\sum_{i+j=k} a_i b_j \big)\zeta^{(k-m)} \\ &= \sum_{k=0}^{n-1} \big(\sum_{i=0}^{k} a_i b_{k-i} - \sum_{i=1}^{n-1} a_i b_{n-i} + \sum_{i=k+2}^{n-1} a_i b_{m+k-i} \big)\zeta^k\end{aligned}$$

Thus we see that each coefficient of a product of two elements in the power basis consists of a sum of $2n-2$ terms, each term being the product of two independent random variables.

If we create the private keys and error vectors by drawing coefficients of the power basis uniformly at random from $\{-B, \dots, 0, \dots, B\}$, then it is fairly simple to directly analyse the correctness of the system and compute specific probabilities of failure. In this case, each of the a_i and b_j are independent random variables following a uniform distribution over the set $\{-B, \dots, 0, \dots, B\}$. The product of two such random variables is itself a random variable over $\{-B^2, \dots, 0, \dots, B^2\}$ and the sum of such variables yields a convolution of their distributions. Thus the coefficients of a product $a \cdot b$ are random variables with support $\{-(2n-2) \cdot B^2, \dots, 0, \dots, (2n-2) \cdot B^2\}$.

We can calculate the probability that $||s_0e_0 + e_2 - s_1e_1||_\infty < \lfloor q/8 \rfloor$ by considering the correct convolution of probability distributions. If we wish to consider the probability of failure of the key exchange, we must recognise that there are n coefficients, each of which could exceed the $q/8$ bound. However, the $q/8$ bound is only tight for coefficients falling at the boundary of a quadrant: $\{0, \pm\lfloor q/4 \rceil, \lfloor q/2 \rceil\}$. Most coefficients will not lie on or near these boundaries, which will lead to a higher permissible bound. For each of our proposed parameter sets, we calculate the probability that a coefficient exceeds $q/8$ or that a key exchange failure will occur and record these in Table 8.

	I	**II**	**II**	**IV**	**V**	**VI**
Target	-80	-80	-80	-80	-80	-80
Coefficient Failure	-91.751	-86.109	-92.112	-91.146	-89.272	-87.208
Agreement Failure	-91.300	-85.204	-90.985	-89.782	-87.650	-85.401

Figure 8: $\log_2$ failure probabilities for uniform noise ($B = 5$).

If error terms are generated by sampling a Gaussian distribution, then each of the independent random variables a_i, b_j comes from a discretised Gaussian distribution centered at 0 and with variance σ^2. By invoking (a generalisation of) the Central Limit Theorem[7], we can say that

[7] Each of the a_i and b_j is independent; however, when summing up the products of such terms, a given a_i or b_j will appear on average in two such products, creating a weak dependence between the summands.

each of the coefficients of the product is well approximated by a Gaussian distribution centered at 0 with variance $(2n-2)\sigma^4$, and to approximate the coefficients of $s_0e_0 + e_2 - s_1e_1$ we can consider a Gaussian distribution centered at 0 with variance $2(2n-2)\sigma^4 + \sigma^2$. We estimate the probability that a coefficient exceeds $q/8$ or that a key exchange failure will occur and record these in Table 9.

	I	**II**	**II**	**IV**	**V**	**VI**
Target	-80	-80	-80	-80	-80	-80
Coefficient Failure	-88	-83	-89	-88	-86	-84
Agreement Failure	-88	-82	-87	-86	-84	-82

Figure 9: $\log_2$ failure probabilities for Gaussian noise ($\sigma = 3.192$).

5.3.2 Decoding basis

If we instead use the decoding basis for R, then describing the multiplication of two elements is slightly more complicated. Let $\vec{d} = \{d_i\}$ be the decoding basis for R as described in Section 2.3.1. The product of two or more error terms in the decoding basis should also be multiplied by $g := 1 - \zeta$ to help control the growth of these noise terms. The product of $(1-\zeta)$ times two basis elements can be computed using algebra and is given by the formula

$$(1-\zeta) \cdot d_i \cdot d_j = d_{i+j \bmod m} - d_{i-1 \bmod m} - d_{j-1 \bmod m} + d_{n-1}.$$

Let $a \in R_q$ be written as $\sum_{i=0}^{n-1} a_i d_i$ and similarly let $b = \sum_{j=0}^{n-1} b_j d_j \in R_q$ and set $d_n := 0 =: a_n = b_n$. Then the product of two elements a, b in the decoding basis is given by

$$\begin{aligned}(1-\zeta) \cdot \big(\sum_{i=0}^{n-1} a_i d_i\big) \cdot \big(\sum_{j=0}^{n-1} b_j d_j\big) = \\ \sum_{k=0}^{n-2} \big(\sum_{i=0}^{n-1} a_i b_{(k-i) \bmod m} - \sum_{i=0}^{n-1} a_i b_{(k+1) \bmod m} - \sum_{j=0}^{n-1} a_{(k+1) \bmod m} b_j\big) d_k \\ + \big(\sum_{i,j=0}^{n-1} a_i b_j + \sum_{i=0}^{n-1} a_i b_{(m-2-i) \bmod m}\big) d_{n-1}\end{aligned}$$

Thus we see that in this case the average coefficient of a product of two elements in the decoding basis consists of a sum of $3n$ terms, each term being the product of two independent random variables. Moreover, the coefficient of the basis element d_{n-1} is a sum of $n^2 + n$ summands, each the product of two independent random variables. Thus we can see that the error terms grow more rapidly in the decoding basis than in the power basis and so we prefer the use of the power basis.[8] If we wish to quantify the penalty incurred from using the decoding basis

[8] The growth of the average error term in the decoding basis is larger than that of the error terms in the power basis, but the growth of the coefficient of the d_{n-1} term is especially problematic; this observation has been

in another way, we can calculate the probability of failure if secrets are chosen uniformly from random. For example if we consider parameter set **I**, the probability that a coefficient will be over the $q/8$ bound increases from $2^{-91.8}$ in the power basis to $2^{-62.0}$ in the decoding basis, *excluding* the final coefficient of d_{n-1}.

5.4 Timings

We plan to implement the Ring-LWE ephemeral Diffie-Hellman key encapsulation mechanism of Section 4 in the lower level language `C` in order to assess its performance. This code will be made available on the GitHub repository at `https://github.com/vscrypto/ringlwe` and the results presented in an update to this paper.

6 Conclusions

The ring learning with errors problem is a promising cryptographic primitive that is believed to be resistant to attacks by quantum computers. The decision ring-LWE problem naturally leads to a passively secure Diffie-Hellman-like unauthenticated key exchange protocol, which can readily be extended to an actively secure version and an authenticated version. We have examined these key exchange mechanisms and provided both practical analysis and more practical parameter choices than were previously available. By considering prime cyclotomic rings, we are able to produce a wider array of parameter choices and achieve smaller public keys. Once we produce performance timings for an update to this paper we will get a sense for whether there is an increased computational cost, and therefore a trade-off that may be appealing depending on the application. We also provide the first in-depth analysis of a specific usage other than the power-of-two case, which has produced some interesting insights into the functioning of the decoding basis.

We plan to continue and expand upon this work in a number of ways. We plan to undertake further coding implementations to compare the performance against classical mechanisms like RSA or Diffie-Hellman. We would also like to explore related signature algorithms.

References

[1] M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of Learning with Errors. `http://eprint.iacr.org/2015/046`.

borne out by experimentation where the d_{n-1} term can frequently be the cause of failure of the key exchange for various choices of parameters, and the expected correctness of the key exchange is lower for the decoding basis than for the power basis. Happily, the power basis is also simpler to use. If the decoding basis is chosen for some reason, we advocate the dropping of the final coefficient term, as this will markedly improve the performance of the system.

[2] M. R. Albrecht. Sage code for [1]. `https://bitbucket.org/malb/lwe-estimator`.

[3] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. `http://eprint.iacr.org/2014/599`.

[4] L. Ducas and A. Durmus. Ring-LWE in polynomial rings. In *PKC 2012*, pages 34-51. 2012.

[5] Y. Elias, K. E. Lauter, E. Ozman, and K. E. Stange. Provably weak instances of Ring-LWE. In submission. Avaliable at `http://eprint.iacr.org/2015/106`.

[6] T. Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. `http://eprint.iacr.org/2014/744`.

[7] T. Laarhoven, M. Mosca and J. van de Pol. Finding shortest lattice vectors faster using quantum search. `http://eprint.iacr.org/2014/907`.

[8] T. Lepoint and M. Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. In David Pointcheval and Damien Vergaud, editors, *AFRICACRYPT*, volume 8469 of *Lecture Notes in Computer Science*, pages 318-335. Springer, 2014.

[9] R. Linder and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In *CT-RSA'11*, pages 319-339, 2011.

[10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *Journal of the AMC*, 60(6):43:1-43:35, November, 2013. Preliminary version in *EUROCRYPT*, pages 1-23, 2010.

[11] V. Lyubashevsky, C. Peikert, and O. Regev. A Toolkit for Ring-LWE Cryptography. In *EUROCRYPT'13*, pages 35-54. 2013.

[12] A. Mariano, T. Laarhoven, and C. Bischof. Parallel (probable) lock-free HashSieve: a practical sieving algorithm for the SVP. `http://eprint.iacr.org/2015/041`.

[13] D. Micciancio and O. Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmaen, editors, *Post-Quantum Cryptography*, pages 147-191. Springer Berlin Heidelberg, 2009.

[14] C. Peikert. Lattice Cryptography for the Internet. In Michele Mosca, editor, *Proc. 6th International Conference on Post-Quantum Cryptography (PQCrypto) 2014*, LNCS 8772, pages 197-219. Springer, 2014. Full version available at `http://eprint.iacr.org/2014/070`

[15] V. Singh. A Practical Key Exchange for the Internet using Lattice Cryptography. `http://eprint.iacr.org/2015/138`.