PROVABLY WEAK INSTANCES OF RING-LWE

YARA ELIAS, KRISTIN E. LAUTER, EKIN OZMAN, AND KATHERINE E. STANGE

Abstract. The *ring and polynomial learning with errors* problems (Ring-LWE and Poly-LWE) have been proposed as hard problems to form the basis for cryptosystems, and various security reductions to hard lattice problems have been presented. So far these problems have been stated for general (number) rings but have only been closely examined for cyclotomic number rings. In this paper, we state and examine the Ring-LWE problem for general number rings and demonstrate *provably weak instances* of Ring-LWE. We construct an explicit family of number fields for which we have an efficient attack. We demonstrate the attack in both theory and practice, providing code and running times for the attack. The attack runs in time linear in q, where q is the modulus.

Our attack is based on the attack on Poly-LWE which was presented in [EHL]. We extend the EHL-attack to apply to a larger class of number fields, and show how it applies to attack Ring-LWE for a heuristically large class of fields. Certain Ring-LWE instances can be transformed into Poly-LWE instances without distorting the error too much, and thus provide the first weak instances of the Ring-LWE problem. We also provide additional examples of fields which are vulnerable to our attacks on Poly-LWE, including power-of-2 cyclotomic fields, presented using the minimal polynomial of $\zeta_{2^n} \pm 1$.

1. Introduction

Lattice-based cryptography has become a very hot research topic recently with the emergence of new applications to homomorphic encryption. The hardness of the Ring-LWE problem was related to various well-known hard lattice problems [R, MR09, MR04], [LPR, BL+], and the hardness of the Poly-LWE problem was reduced to Ring-LWE in [LPR, DD]. The hardness of the Poly-LWE problem is used as the basis of security for numerous cryptosystems, including [BV, BGV]. The hardness of Ring-LWE was also shown [SS] to form a basis for the proof of security of a variant of NTRU [HPS, IEEE].

In [EHL], the first weaknesses in the Poly-LWE problem were discovered for classes of number fields satisfying certain properties. In addition, a list of properties of number fields were identified which are sufficient to guarantee a reduction between the Ring-LWE and the Poly-LWE problems, and a search-to-decision reduction for Ring-LWE. Unfortunately, in [EHL], no number fields were found which satisfied both the conditions for the attack and for the reductions. Thus [EHL] produced only examples of number fields which were weak instances for Poly-LWE.

The contributions of this paper at a high level are as follows: In Section 3 we strengthen and extend the attacks presented in [EHL] in several significant ways. In Section 4, most importantly, we show how the attacks can be applied also to the Ring-LWE problem. In Section 5, we construct an explicit family of number fields for which we have an efficient attack on Ring-LWE. This represents the first successful attacks on the Ring-LWE problem for number fields with special properties. In addition, in Section 9, we present the first successful implementation of the EHL attack at cryptographic sizes and attack both Ring-LWE and Poly-LWE instances. For example for $n = 1024$ and $q = 2^{31} - 1$, the attack runs in about 13 hours. Code for the attack is given in Appendix A. In Section 6 we give a more

Date: February 12, 2015.

general construction of number fields such that heuristically a large percentage of them will be vulnerable to the attacks on Ring-LWE.

In more detail, we consider rings of integers in number fields $K = \mathbb{Q}[x]/(f(x))$ of degree n, modulo a large prime number q, and we give attacks on Poly-LWE which work when $f(x)$ has a root of small order modulo q. The possibility of such an attack was mentioned in [EHL] but not explored further. In Sections 3.1 and 3.2, we give *two* algorithms for this attack, and in Sections 7 and 7.3 we give many examples of number fields and moduli, some of cryptographic size, which are vulnerable to this attack. The most significant consequence of the attack is the construction of the number fields which are weak for the Ring-LWE problem (Section 6).

To understand the vulnerability of Ring-LWE to these attacks, we state and examine the Ring-LWE problem for general number rings and demonstrate *provably weak instances* of Ring-LWE. We demonstrate the attack in both theory and practice for an explicit family of number fields, providing code and runnning times for the attack. The attack runs in time linear in q, where q is the modulus. The essential point is that Ring-LWE instances can be mapped into Poly-LWE instances, and if the map does not distort the error too much, then the instances may be vulnerable to attacks on Poly-LWE. The distortion is governed by the spectral norm of the map, and we compute the spectral norm for the explicit family we construct in Section 5 and analyze when the attack will succeed. For the provably weak family which we construct, the feasibility of the attack depends on the ratio of $\sqrt{q}/n$. We prove that the attack succeeds when $\sqrt{q}/n$ is above a certain bound, but in practice we find that we can attack instances where the ratio is almost 100 times smaller than that bound. Even for Ring-LWE examples which are not taken from the provably weak family, we were able to attack in practice relatively generic instances of number fields where the spectral norm was small enough (see Section 9).

We investigate cyclotomic fields (even 2-power cyclotomic fields) given by an alternate minimal polynomial, which are weak instances of Poly-LWE for that choice of polynomial basis. Section 7.3 contains numerous examples of 2-power cyclotomic fields which are vulnerable to attack when instantiated using an alternative polynomial basis, thus showing the heavy dependence in the hardness of these lattice-based problems on the choice of polynomial basis. In addition, we analyze the case of cyclotomic fields to understand their potential vulnerability to these lines of attack and we explain why cyclotomic fields are immune to attacks based on roots of small order (Section 8). Finally, we provide code in the form of simple routines in SAGE to implement the attacks and algorithms given in this paper and demonstrate successful attacks with running times (Section 9).

As a consequence of our results, one can conclude that the hardness of Ring-LWE is both *dependent on special properties of the number field* and *sensitive to the particular choice of* q, and some choices may be significantly weaker than others. In addition, for applications to cryptography, since our attacks on Poly-LWE run in time roughly $O(q)$ and may be applicable to a wide range of fields, including even 2-power cyclotomic fields with a bad choice of polynomial basis, these attacks should be taken into consideration when selecting parameters for Poly-LWE-based systems such as [BV, BGV] and other variants. For many important applications to homomorphic encryption (see for example [GLN, BLN]), these attacks will not be relevant, since the modulus q is chosen large enough to allow for significant error growth in computation, and would typically be of size 128 bits up to 512 bits. For that range, the attacks presented in this paper would not run. However, in other applications of Ring-LWE to key exchange for the TLS protocol [BCNS], for example, parameters for achieving 128-bit security are suggested where $n = 2^{10}$ and $q = 2^{32} - 1$, with $\sigma \approx 3$, and these parameters would certainly be vulnerable to our attacks for weak choices of fields and q.

Acknowledgements. The authors are indebted to the organizers of the research conference Women in Numbers 3 (Rachel Pries, Ling Long and the fourth author), as well as to the Banff International Research Station, for bringing together this collaboration. The authors would also like to thank Martin Albrecht for help with Sage.

2. Background on Poly-LWE

Let $f(x)$ be a monic irreducible polynomial in $\mathbb{Z}[x]$ of degree n, and let q be a prime such that $f(x)$ factors completely modulo q. Let $P = \mathbb{Z}[x]/f(x)$ and let $P_q = P/qP = \mathbb{F}_q[x]/f(x)$. Let $\sigma \in \mathbb{R}^{>0}$. The uniform distribution on $P \simeq \mathbb{Z}^n$ will be denoted $\mathcal{U}$. By *Gaussian distribution of parameter* σ we refer to a discrete Gaussian distribution of mean 0 and variance σ^2 on P, spherical with respect to the power basis. This will be denoted $\mathcal{G}_\sigma$. It is important to our analysis that we assume that in practice, elements are sampled from Gaussians of parameter σ truncated at width 2σ.

There are two standard Poly-LWE problems. Our attack solves the *decision* variant, but it also provides information about the secret.

Problem 2.1 (Decision Poly-LWE Problem)**.** *Let $s(x) \in P$ be a secret. The* decision Poly-LWE problem *is to distinguish, with non-negligible advantage, between the same number of independent samples in two distributions on $P \times P$. The first consists of samples of the form $(a(x), b(x) := a(x)s(x) + e(x))$ where $e(x)$ is drawn from a discrete Gaussian distribution of parameter σ, and $a(x)$ is uniformly random. The second consists of uniformly random and independent samples from $P \times P$.*

Problem 2.2 (Search Poly-LWE Problem)**.** *Let $s(x) \in P$ be a secret. The* search Poly-LWE problem*, is to discover s given access to arbitrarily many independent samples of the form $(a(x), b(x) := a(x)s(x) + e(x))$ where $e(x)$ is drawn from a Discrete Gaussian of parameter σ, and $a(x)$ is uniformly random.*

The polynomial $s(x)$ is called the *secret* and the polynomials $e_i(x)$ are called the *errors*.

2.1. Parameter selection.

The selection of parameters for security is not yet a well-explored topic. Generally for Poly-LWE and Ring-LWE, recourse is taken to the parameter recommendations of general LWE, e.g. [PG, RV, BCNS]. Suggested parameters we were able to find include

(1) $P_{LP1} = (n, q, s) = (192, 4093, 8.87)$, $P_{LP2} = (256, 4093, 8.35)$, $P_{LP3} = (320, 4093, 8.00)$ for low, medium and high security, recommended by Lindner and Peikert in [LP];
(2) $P_{GF} = (n, q, s) = (512, 12289, 12.18)$ for high security used in [GF];
(3) $P_{BCNS} = (n, q, s) = (1024, 2^{31} - 1, 3.192)$ suggested in [BCNS] for the TLS protocol. Here, $q = 2^{32} - 1$ was actually suggested but it is not prime. Here, the authors remark that q is taken to be large for correctness but could potentially be decreased.

Note that s reflects the width of the Gaussian: precisely, $s = \sqrt{2\pi}\sigma$ (these variable names are not entirely consistent in the literature).

3. Attacks on Poly-LWE

The attack we are concerned with is quite simple. It proceeds in four stages:

(1) Transfer the problem to $\mathbb{F}_q$ via a ring homomorphism $\phi : P_q \to \mathbb{F}_q$.
(2) Loop through guesses for the possible images $\phi(s(x))$ of the secret.
(3) Obtain the values $\phi(e_i(x))$ under the assumption that the guess at hand is correct.
(4) Examine the distribution of the $\phi(e_i(x))$ to determine if it is Gaussian or uniform.

If f is assumed to have a root $\alpha \equiv 1 \mod q$ or α of small order modulo q, then this attack is due to Eisentraeger-Hallgren-Lauter [EHL].

The first part is to transfer the problem to $\mathbb{F}_q$. Write $f(x) = \prod_{i=1}^{n}(x - \alpha_i)$ for the factorization of $f(x)$ over $\mathbb{F}_q$ which is possible by assumption. By the Chinese remainder theorem, if f has no double roots, then

$$P_q \simeq \prod_{i=1}^{n} \mathbb{F}_q[x]/(x - \alpha_i) \simeq \mathbb{F}_q^n$$

There are n ring homomorphisms

$$\phi : P_q \to \mathbb{F}_q[x]/(x - \alpha_i) \simeq \mathbb{F}_q, \quad g(x) \mapsto g(\alpha_i).$$

Fix one of these, by specifying a root $\alpha = \alpha_i$ of $f(x)$ in $\mathbb{F}_q$. Apply the homomorphism to the coordinates of the ℓ samples $(a_i(x), b_i(x))$, obtaining $(a_i(\alpha), b_i(\alpha))_{i=1,\dots,\ell}$.

Next, loop through all $g \in \mathbb{F}_q$. Each value g is to be considered a guess for the value of $s(\alpha)$. For each guess g, assuming that it is a correct guess and $g = s(\alpha)$, then

$$e_i(\alpha) = b_i(\alpha) - a_i(\alpha)g = b_i(\alpha) - a_i(\alpha)s(\alpha).$$

In the case that the samples were LWE samples and the guess was correct, then this produces a collection $(e_i(\alpha))$ of images of errors chosen according to some distribution. If the distributions $\phi(\mathcal{U})$ and $\phi(\mathcal{G}_\sigma)$ are distinguishable, then we can determine whether the distribution was uniform or Gaussian. Note that $\phi(\mathcal{U})$ will of course be uniform on $\mathbb{F}_q$. If our guess is incorrect, or if the samples are not LWE samples, then the distribution will appear uniform.

Therefore, after looping through all guesses, if all the distributions appeared uniform, then conclude that the samples were not LWE samples; whereas if one of the guesses worked for all samples and always yielded an error distribution which appeared Gaussian, assume that particular g was a correct guess. In the latter case this also yields one piece of information about the secret: $g = s(\alpha) \mod q$.

The attack *will* succeed whenever

(1) q is small enough to allow looping through $\mathbb{F}_q$,
(2) $\phi(\mathcal{U})$ and $\phi(\mathcal{G}_\sigma)$ are distinguishable.

Our analysis hinges on the difficulty of distinguishing $\phi(\mathcal{U})$ from $\phi(\mathcal{G}_\sigma)$, as a function of the parameters σ, n, ℓ, q, and f. Distinguishability becomes easier when σ is smaller (so $\mathcal{U}$ and $\mathcal{G}_\sigma$ are farther apart to begin with), n is smaller and q is larger (since then less information is lost in the map ϕ), and ℓ is larger (since there are more samples to test the distributions). The dependence on f comes primarily as a function of its roots α_i modulo q, which may have properties that make distinguishing easier.

Ideally, for higher security, one will choose parameters that make distinguishing nearly impossible, i.e. such that $\phi(\mathcal{G}_\sigma)$ appears very close to uniform modulo q.

Example. ([EHL]) We illustrate the attack in the simplest case $\alpha = 1$. Assume $f(1) \equiv 0 \mod q$, and consider the distinguishability of the two distributions $\phi(\mathcal{U})$ and $\phi(\mathcal{G}_\sigma)$. Given $(a_i(x), b_i(x))$, make a guess $g \in \mathbb{F}_q$ for the value of $s(1)$ and compute $b_i(1) - g \cdot a_i(1)$. If b_i is uniform, then $b_i(1) - g \cdot a_i(1)$ is uniform for all g. If $b_i = a_i s + e_i$, then there is a guess g for which $b_i(1) - g a_i(1) = e_i(1)$ where $e_i(x) = \sum_{j=1}^{n} e_{ij} x^j$ and $g = s(1)$. Since $e_i(1) = \sum_{j=1}^{n} e_{ij}$, where e_{ij} are chosen from $\mathcal{G}_\sigma$, it follows that $e_i(1)$ are sampled from $\mathcal{G}_{\sqrt{n}\sigma}$ where $n\sigma^2 << q$. The attack can be described loosely as follows: for each sample, test each guess g in $\mathbb{F}_q$ to see if $b_i(1) - g \cdot a_i(1)$ is small modulo q, and only keep those guesses which pass the test. Repeat with the next sample and continue to keep only the guesses which pass.

3.1. **Attack based on a small set of error values modulo q.** In this section, we assume that there exists a root α of f such that α has small order r modulo q, that is $\alpha^r \equiv 1 \mod q$.

Then

$$e(\alpha) = \sum_{i=0}^{n-1} e_i \alpha^i = (e_0 + e_r + e_{2r} + \cdots) + \alpha(e_1 + e_{r+1} + \cdots) + \cdots + \alpha^{r-1}(e_{r-1} + e_{2r-1} + \cdots). \quad (1)$$

If r is small enough, then $e(\alpha)$ takes on only a small number of values modulo q. If so, then we can efficiently distinguish whether a value modulo q belongs to that subset.

Let S be the set of possible values of $e(\alpha)$ modulo q. We assume for simplicity that n is divisible by r. Then the coefficients $e_j + e_{j+r} + \cdots + e_{n-r+j}$ of (1) fall into a subset of $\mathbb{Z}/q\mathbb{Z}$ of size at most $4\sigma n/r$. We sum over r terms, hence, $|S| = (4\sigma n/r)^r$ residues modulo q. For $r = 2$, this becomes $(2n\sigma)^2$.

The attack described below succeeds with high probability if $|S| << q$, that is

$$(4\sigma n/r)^r << q.$$

Algorithm 1 Small set of error values

Input: A collection of ℓ Poly-LWE samples.
Output: A guess g for $s(\alpha)$, the value of the secret polynomial at α; or else **NOT PLWE**; or **INSUFFICIENT SAMPLES**.
The value **NOT PLWE** indicates that the collection of samples were definitely not Poly-LWE samples.
The value **INSUFFICIENT SAMPLES** indicates that there were not enough samples to determine a single guess $s(\alpha)$. In this case, the algorithm may be continued on a new set of samples by looping the remaining surviving guesses on the new samples.

Create an ordered list of elements of S.
Let G be an empty list.
for g from 0 to $q-1$ **do**
    for $(a(x), b(x))$ in the collection of samples **do**
        if $b(\alpha) - ga(\alpha)$ does not equal an element of S **then**
            break (i.e. begin next value of g)
    append g to G (note: occurs only if the loop of samples completed without a break)
if G is empty **then**
    return **NOT PLWE**
if $G = \{g\}$ **then**
    return g
if $\#G > 1$ **then**
    return **INSUFFICIENT SAMPLES**

Proposition 3.1. *Assume that*

$$(4\sigma n/r)^r < q. \quad (2)$$

Algorithm 1 terminates in time at most $\widetilde{O}(\ell q + nq)$, where the $\widetilde{O}$ notation hides the $\log(q)$ factors and the implied constant depends upon r. Furthermore, if the algorithm returns **NOT PLWE**, *then the samples were not valid Poly-LWE samples. If it outputs anything other than* **NOT PLWE**, *then the samples are valid Poly-LWE samples with probability $1 - (\frac{\#S}{q})^\ell$. In particular, this probability tends to 1 as ℓ grows.*

Proof. As discussed above, there are at most q possible values for the elements of S under the assumption (2). To compute each one takes n additions per coefficient (of which there are r), combined with an additional r multiplications and r additions. (Here we have assumed the α^i have been computed; this takes r multiplications.) Each addition or multiplication

takes time at most $\log q$. Therefore, computing S takes time at most $\widetilde{O}(qnr)$. For sorting, it is best to sort as S is computed; placing each element correctly takes $\log q$ time.

The principal double loop takes time at most $\widetilde{O}(\ell q)$. If $b(\alpha)$ and $a(\alpha)$ are precomputed, then for each guess g, the computation of $b(\alpha) - ga(\alpha)$ only costs one multiplication and one subtraction modulo q (i.e. $2\log q$) while it requires only $\log q$ bit comparisons to decide whether this is in the set S.

In Step 4, for later samples, only guesses which were successful in the previous samples (i.e. gave a value which was in the set S) are considered. For a sample chosen uniformly at random, one expects the number of successful guesses to be roughly $\frac{\#S}{q}$. Thus for the second sample, we repeat the above test for only $(\#S)$ guesses. At the ℓ^{th} sample, retaining only guesses which were successful for all previous samples, we expect to test only $(\frac{\#S}{q})^\ell q$ guesses, which very quickly goes to zero. Hence, if we examine ℓ samples, our tolerance for false positives is proportional to $(\frac{\#S}{q})^\ell$.

□

3.2. **Attack based on the size of the error values.** In this section, we describe the most general $\phi : P_q \to \mathbb{F}_q$ attack on the Poly-LWE problem, one which can be carried out in any situation. The rub is that the probability of success will be vanishingly small unless we are in a very special situation. Therefore our analysis actually bolsters the security of Poly-LWE.

Suppose that $f(\alpha) \equiv 0 \mod q$. Let E_i be the event that $b_i(\alpha) - ga_i(\alpha) \mod q$ is in the interval $[-q/4, q/4)$ for some sample i and guess g for $s(\alpha) \mod q$. The main idea is to compare $P(E_i \mid \mathcal{D} = \mathcal{U})$ and $P(E_i \mid \mathcal{D} = \mathcal{G}_\sigma)$. If $\mathcal{D} = \mathcal{U}$, then $b_i(\alpha) - ga_i(\alpha)$ is random modulo q for all guesses g, that is, $P(E_i \mid \mathcal{D} = \mathcal{U}) = \frac{1}{2}$. If $\mathcal{D} = \mathcal{G}_\sigma$, then $b_i(\alpha) - s(\alpha)a_i(\alpha) = e_i(\alpha) \mod q$. We consider

$$e_i(\alpha) = \sum_{j=0}^{n-1} e_{ij}\alpha^j,$$

where e_{ij} is chosen according to the distribution $\mathcal{G}_\sigma$ (truncated at 2σ) and distinguish two cases:

(1) $\alpha = \pm 1$
(2) $\alpha \neq \pm 1$ and α has small order $r \geq 3$ modulo q

Case 1 ($\alpha = \pm 1$).

The error $e_i(\alpha)$ is chosen according to the distribution $\mathcal{G}_{\sigma\sqrt{n}}$ truncated at $2\sigma\sqrt{n}$. Hence

$$-2\sigma\sqrt{n} \leq e_i(\alpha) \leq 2\sigma\sqrt{n}.$$

Therefore, assuming that

$$2\sigma\sqrt{n} < \frac{q}{4},$$

we obtain $P(E_i \mid \mathcal{D} = \mathcal{G}_\sigma) = 1$ for $g = s(\alpha)$. Hence $\mathcal{U}$ and $\mathcal{G}_\sigma$ are distinguishable.

Case 2 ($\alpha \neq \pm 1$ and α has small order $r \geq 3$ modulo q).

The error can be written as

$$e(\alpha) = \sum_{i=0}^{r-1} e_i\alpha^i = (e_0 + e_r + \cdots) + \alpha(e_1 + e_{r+1} + \cdots) + \cdots + \alpha^{r-1}(e_{r-1} + e_{2r-1} + \cdots)$$

where we assume that n is divisible by r for simplicity. For $j = 0, \cdots, r-1$, we have that $e_j + e_{j+r} + \cdots + e_{j+n-r}$ is chosen according to the distribution $\mathcal{G}_{\sqrt{\frac{n}{r}}\sigma}$. As a consequence $e(\alpha)$ is sampled from $\mathcal{G}_{\bar{\sigma}}$ where

$$\bar{\sigma}^2 = \sum_{i=0}^{r-1} \frac{n}{r}\sigma^2\alpha^{2i} = \frac{n}{r}\sigma^2\frac{\alpha^{2r} - 1}{\alpha^2 - 1}.$$

Hence

$$-2\frac{\sqrt{n}}{\sqrt{r}}\sigma\frac{\sqrt{\alpha^{2r}-1}}{\sqrt{\alpha^2-1}} \le e(\alpha) \le 2\frac{\sqrt{n}}{\sqrt{r}}\sigma\frac{\sqrt{\alpha^{2r}-1}}{\sqrt{\alpha^2-1}}.$$

Therefore, assuming that

$$2\frac{\sqrt{n}}{\sqrt{r}}\sigma\frac{\sqrt{\alpha^{2r}-1}}{\sqrt{\alpha^2-1}} < \frac{q}{4}, \tag{3}$$

we obtain $P(E_i \mid \mathcal{D} = \mathcal{G}_\sigma) = 1$ for $g = s(\alpha)$, and uniform and Gaussian are distinguishable. Note that Hypothesis (2) implies in particular that $\alpha^r > q$.

Algorithm 2 Small error values

Input: A collection of ℓ Poly-LWE samples.
Output: A guess g for $s(\alpha)$; or else **NOT PLWE**; or **INSUFFICIENT SAMPLES**. The output **INSUFFICIENT SAMPLES** indicates that more samples are needed to make a determination. In this case, the algorithm can be continued by looping through remaining surviving guesses on new samples.

Let G be an empty list.
for g from 1 to $q-1$ **do**
  for $(a(x), b(x))$ in the collection of samples **do**
    if the minimal residue $b(\alpha) - ga(\alpha)$ does not lie in $[-q/4, q/4)$ **then**
      break (i.e. begin next value of g)
  append g to G (note: occurs only if the loop of samples completed without a break)
if G is empty **then**
  return **NOT PLWE**
if $G = \{g\}$ **then**
  return g
if $\#G > 1$ **then**
  return **INSUFFICIENT SAMPLES**

In each of the two cases, we have given conditions on the size of σ under which $\mathcal{U}$ and $\mathcal{G}_\sigma$ are distinguishable and an attack is likely to succeed. We now elaborate on the algorithm that would be used.

We denote by ℓ the number of samples observed. For each guess g mod q, we compute $b_i - ga_i$ for $i = 1, \dots, \ell$. If there is a guess g mod q for which the event E_i occurs for all $i = 1, \dots, \ell$, then the algorithm returns the guess if it is unique and **INSUFFICIENT SAMPLES** otherwise; the samples are likely valid Poly-LWE samples. Otherwise, it reports that they are certainly not valid Poly-LWE samples.

Proposition 3.2. *Assume that we are in one of the following cases:*

(1) $\alpha = \pm 1$ *and*

$$8\sigma\sqrt{n} < q.$$

(2) α *has small order* $r \ge 3$ *modulo* q*, and*

$$8\sigma\frac{\sqrt{n}}{\sqrt{r}}\frac{\sqrt{\alpha^{2r}-1}}{\sqrt{\alpha^2-1}} < q.$$

Then Algorithm 2 terminates in time at most $\widetilde{O}(\ell q)$*, where the implied constant is absolute. Furthermore, if the algorithm returns* **NOT PLWE***, then the samples were not valid Poly-LWE samples. If it outputs anything other than* **NOT PLWE***, then the samples are valid Poly-LWE samples with probability at least* $1 - (\frac{1}{2})^\ell$.

Proof. The proof is as in Proposition 3.1, without the first few steps. □

We remark that Propositions and Algorithms 3.1 and 3.2 overlap in some cases. For $\alpha = \pm 1$, Algorithm 2 is more applicable (i.e. more parameter choices are susceptible), while for α of other small orders, Algorithm 1 is more applicable.

4. Moving the attack from Poly-LWE to Ring-LWE

We use the term Poly-LWE to refer to LWE problems generated by working in a polynomial ring, and reserve the term Ring-LWE for LWE problems generated by working with the canonical embedding of a number field as in [LPR, LPR13]. In the previous sections we have expanded upon Eisenträger, Hallgren and Lauter's observation that for certain distributions on certain lattices given by Poly-LWE, the ring structure presents a weakness. We will now consider whether it is possible to expand that analysis to LWE instances created through Ring-LWE for number fields besides cyclotomic ones.

In particular, the necessary ingredient is that the distribution be such that under the ring homomorphisms of Section 3, the image of the errors is a 'small' subset of $\mathbb{Z}/q\mathbb{Z}$, either the error values themselves are small, or they form a small, identifiable subset of $\mathbb{Z}/q\mathbb{Z}$. Assuming a spherical Gaussian in the canonical embedding of R or $R^\vee$, we describe a class of number fields for which this weakness occurs. A similar analysis would apply without the assumption that the distribution is spherical in the canonical embedding.

Here, we setup the key players (a number field and its canonical embedding, etc.) for general number fields so that these definitions specialize to those in [LPR13]. There are some choices inherent in our setup: it may be possible to generalize Ring-LWE to number fields in several different ways. We consider the two most natural ways.

4.1. The canonical embedding.

Let K be a number field of degree n with ring of integers R whose dual is $R^\vee$. We will embed the field K in $\mathbb{R}^n$. Note that our setup is essentially that of [DD], rather than [LPR13], but the difference is notational.

Let $\sigma_1, \dots, \sigma_n$ be the n embeddings of K, ordered so that σ_1 through σ_{s_1} are the s_1 real embeddings, and the remaining $n - s_1 = 2s_2$ complex embeddings are paired in such a way that $\overline{\sigma_{s_1+k}} = \sigma_{s_1+s_2+k}$ for $k = 1, \dots, s_2$ (i.e. list s_2 non-pairwise-conjugate embeddings and then list their conjugates following that).

Define a map $\theta : K \to \mathbb{R}^n$ given by

$$\theta(r) = (\sigma_1(r), \dots, \sigma_{s_1}(r), Re(\sigma_{s_1+1}(r)), \dots, Re(\sigma_{s_1+s_2}(r)), Im(\sigma_{s_1+1}(r)), \dots, Im(\sigma_{s_1+s_2}(r))).$$

The image of K is the $\mathbb{Q}$-span of $\theta(\omega_i)$ for any basis ω_i for K over $\mathbb{Q}$. This is not the usual Minkowski embedding, but it has the virtues that 1) the codomain is a real, not complex, vector space; and 2) the spherical or elliptical Gaussians used as error distributions in [LPR13] are, in our setup, spherical or elliptical with respect to the usual inner product. We denote the usual inner product by $\langle \cdot, \cdot \rangle$ and the corresponding length by $|x| = \sqrt{\langle x, x \rangle}$. It is related to the trace pairing on K, i.e. $\langle \theta(r), \theta(s) \rangle = \mathrm{Tr}(r\bar{s})$.

Then R and $R^\vee$ form lattices in $\mathbb{R}^n$.

4.2. Spherical Gaussians and error distributions.

We define a *Ring-LWE error distribution* to be a spherical Gaussian distribution in $\mathbb{R}^n$. That is, for a parameter $\sigma > 0$, define the *continuous Gaussian distribution function* $D_\sigma : \mathbb{R}^n \to (0, 1]$ by

$$D_\sigma(x) := (\sqrt{2\pi}\sigma)^{-n} \exp\left(-|x|^2/(2\sigma^2)\right).$$

This gives a distribution Ψ on $K \otimes \mathbb{R}$, via the isomorphism θ to $\mathbb{R}^n$. By approximating $K \otimes \mathbb{R}$ by K to sufficient precision, this gives a distribution on K.

From this distribution we can generate the *Ring-LWE error distribution* on R, respectively $R^\vee$, by taking a valid discretization $\lfloor \Psi \rceil_R$, respectively $\lfloor \Psi \rceil_{R^\vee}$, in the sense of [LPR13]. Now we have at hand a lattice, R, respectively $R^\vee$, and a distribution on that lattice. The

parameters (particularly σ) are generally advised to be chosen so that this instance of LWE is secure against general attacks on LWE (which do not depend on the extra structure endowed by the number theory).

4.3. **The Ring-LWE problems.** Write $R_q := R/qR$ and $R_q^\vee = R^\vee/qR^\vee$. The standard Ring-LWE problems are as follows, where K is taken to be a cyclotomic field [LPR, LPR13].

Definition 4.1 (Ring-LWE Average-Case Decision [LPR])**.** Let $s \in R_q^\vee$ be a secret. The *average-case decision Ring-LWE problem*, is to distinguish with non-negligible advantage between the same number of independent samples in two distributions on $R_q \times R_q^\vee$. The first consists of samples of the form $(a, b := as + e)$ where e is drawn from $\chi := \lfloor \Psi \rceil_{R^\vee}$ and a is uniformly random, and the second consists of uniformly random and independent samples from $R_q \times R_q^\vee$.

Definition 4.2 (Ring-LWE Search [LPR])**.** Let $s \in R_q^\vee$ be a secret. The *search Ring-LWE problem*, is to discover s given access to arbitrarily many independent samples of the form $(a, b := as + e)$ where e is drawn from $\chi := \lfloor \Psi \rceil_{R^\vee}$ and a is uniformly random.

In proposing general number field Ring-LWE, one of two avenues may be taken:

(1) preserve these definitions exactly as they are stated, or
(2) eliminate the duals, i.e. replace every instance of $R^\vee$ with R in the definitions above.

To distinguish these two possible definitions, we will refer to *dual Ring-LWE* and *non-dual Ring-LWE*. Lyubashevsky, Peikert and Regev remark that for cyclotomic fields, dual and non-dual Ring-LWE lead to computationally equivalent problems [LPR, Section 3.3]. They go on to say that over cyclotomics, for implementation and efficiency reasons, dual Ring-LWE is superior.

Generalising dual Ring-LWE to general number fields is the most naive approach, but it presents the problem that working with the dual in a general number field may be difficult. Still, it is possible there are families of accessible number fields for which this may be the desired avenue.

We will analyse the effect of the Poly-LWE vulnerability on both of these candidate definitions. In fact, the analysis will highlight some potential differences in their security, already hinted at in the discussion in [LPR, Section 3.3].

4.4. **Isomorphisms from $\theta(R)$ to a polynomial ring.** Suppose K is a *monogenic number field*, meaning that R is isomorphic to a polynomial ring $P = \mathbb{Z}[X]/f(X)$ for some monic irreducible polynomial f (f is a *monogenic polynomial*). In this case, we obtain $R = \gamma R^\vee$, for some $\gamma \in R$ (here, γ is a generator of the different ideal), so that $\theta(R^\vee)$ and $\theta(R)$ are related by a linear transformation. Thus a (dual or non-dual) Ring-LWE problem concerning the lattice $\theta(R)$ or $\theta(R^\vee)$ can be restated as a Poly-LWE problem concerning P.

Let α be a root of f. Then R is isomorphic to P, via $\alpha \mapsto X$. An integral basis for R is $1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$. An integral basis for $R^\vee$ is $\gamma^{-1}, \gamma^{-1}\alpha, \gamma^{-1}\alpha^2, \ldots, \gamma^{-1}\alpha^{n-1}$. Let M_α be the matrix whose columns are $\{\theta(\alpha^i)\}$. Let $M_\alpha^\vee$ be the matrix whose columns are $\{\theta(\gamma^{-1}\alpha^i)\}$. If $\mathbf{v}$ is a vector of coefficients representing some $\beta \in K$ in terms of the basis $\{\alpha^i\}$ for $K/\mathbb{Q}$, then $\theta(\beta) = M_\alpha \mathbf{v}$. In other words, $M_\alpha : P \to \theta(R)$ is an isomorphism (where P is represented as vectors of coefficients). Similarly, $M_\alpha^\vee : P \to \theta(R^\vee)$ is an isomorphism.

4.5. **The spectral norm.** Given an $n \times n$ matrix M, its *spectral norm* $\rho = ||M||_2$ is the ℓ_2 norm on its n^2 entries. This is equal to the largest singular value of M. This is also equal to the largest radius of the image of a unit ball under M. This last interpretation allows one to bound the image of a spherical Gaussian distribution of parameter σ on the domain of M by another of parameter $\rho\sigma$ on the codomain of M (in the sense that the image of the ball of radius σ will map into a ball of radius $\rho\sigma$ after application of M).

The normalized spectral norm of M is defined to be $\rho' = ||M||_2 / \det(M)^{1/n}$. The condition number of M is $k(M) = ||M||_2 ||M^{-1}||_2$.

4.6. **Moving the attack from Poly-LWE to Ring-LWE.** Via the isomorphism $M := M_\alpha^{-1}$ (respectively $M := (M_\alpha^\vee)^{-1}$), an instance of the non-dual (respectively dual) Ring-LWE problem gives an instance of the Poly-LWE problem in which the error distribution is the image of the error distribution in $\theta(R)$ (respectively $\theta(R^\vee)$). In general, this may be an elliptic Gaussian distorted by the isomorphism. If the distortion is not too large, then it may be bounded by a spherical Gaussian which is not too large. In that case, a solution to the Poly-LWE problem with the new spherical Gaussian error distribution may be possible. If so, it will yield a solution to the original Ring-LWE problem.

This is essentially the same reduction described in [EHL]. However, those authors assume that the isomorphism is an orthogonal linear map; we are loosening this condition. The essential question in this loosening is how much the Gaussian distorts under the isomorphism. Our contribution is an analysis of the particular basis change.

This distortion is governed by the spectral norm ρ of M. If the continuous Gaussian in $\mathbb{R}^n$ is of parameter σ (with respect to the standard basis of $\mathbb{R}^n$), then the new spherical Gaussian bounding its image is of parameter $\rho\sigma$ with respect to P (in terms of the coefficient representation). The appropriate analysis for discrete Gaussians is slightly more subtle. Loosely speaking, we find that a Ring-LWE instance is weak if the following three things occur:

(1) K is monogenic.
(2) f satisfies $f(1) \equiv 0 \pmod q$.
(3) ρ and σ are sufficiently small

The first condition guarantees the existence of appropriate isomorphisms to a polynomial ring; the second and third are required for the Poly-LWE attack to apply. The purpose of the third requirement is that the discrete Gaussian distribution in $\mathbb{R}^n$ transfers to give vectors $e(x)$ in the polynomial ring having the property that $e(1)$ lies in the range $[-q/4, q/4)$ except with negligible probability; this allows Algorithm 3.2 and the conclusions of Proposition 3.2 to apply.

Let us now state our main result.

Theorem 4.3. *Let K be a number field such that $K = \mathbb{Q}(\beta)$, and the ring of integers of K is equal to $\mathbb{Z}[\beta]$. Let f be the minimal polynomial of β and suppose q is a prime such that f has root 1 modulo q. Finally, suppose that the spectral norm ρ of M_β^{-1} satisfies*

$$\rho < \frac{q}{4\sqrt{2\pi}\sigma n}.$$

Then the non-dual Ring-LWE decision problem for K, q, σ can be solved in time $\widetilde{O}(\ell q)$ with probability $1 - 2^{-\ell}$, using a dataset of ℓ samples.

Proof. Sampling a discrete Gaussian with parameter σ results in vectors of norm at most $\sqrt{2\pi}\sigma\sqrt{n}$ except with probability at most 2^{-2n} [LPR13, Lemma 2.8]. Considering the latter to be negligible, then we can expect error vectors to satisfy $||\mathbf{v}||_2 < \sqrt{2\pi}\sigma\sqrt{n}$ and their images in the polynomial ring to satsify

$$|e(1)| = ||e(x)||_1 < \sqrt{n}||e(x)||_2 < \sqrt{n}\rho\sqrt{2\pi}\sigma\sqrt{n} = \rho\sqrt{2\pi}\sigma n.$$

Therefore, if

$$\rho\sqrt{2\pi}\sigma n < q/4,$$

then we may apply the attack of Section 3.2 that assumes $f(1) \equiv 0 \pmod q$ and that error vectors lie in $[-q/4, q/4)$. □

In what follows, we find a family of polynomials satisfying the conditions of the theorem, and give heuristic arguments that such families are in fact very common. The other cases (other than $\alpha = 1$) appear out-of-reach for now, simply because the bounds on ρ are much more difficult to attain. We will not examine them closely.

4.7. **Choice of σ.** The parameters of Section 2.1 are used in implementations where the Gaussian is taken over $(\mathbb{Z}/q\mathbb{Z})^n$, and security depends upon the proportion of this space included in the 'bell,' meaning, it depends upon the ratio q/σ. In the case of Poly-LWE, sampling is done on the coefficients, which are effectively living in the space $(\mathbb{Z}/q\mathbb{Z})^n$, so this is appropriate. However, in Ring-LWE, the embedding $\theta(R)$ in $\mathbb{R}^n$ may be very sparse (or, $\theta(R^\vee)$ may be very dense). Still, the security will hinge upon the proportion of $\theta(R)/q\theta(R)$ that is contained in the bell. We have not seen a discussion of security parameters for Ring-LWE in the literature, and so we propose that the appropriate meaning of s in this case is

$$s := \sqrt{2\pi}\sigma' := \sqrt{2\pi}\sigma \det(M_\alpha)^{1/n}, \tag{4}$$

where σ' is defined by the above equality. The reason for this choice is that $\theta(R)$ has covolume $\det(M_\alpha)$; a very sparse lattice (corresponding to large determinant) needs a correspondingly large σ so that the same proportion of its vectors lie in the bell.

If ρ represents the spectral norm of M_α^{-1} (which has determinant $\det(M_\alpha)^{-1}$), then

$$\rho' := \rho \det(M_\alpha)^{1/n}$$

is the normalized spectral norm. Therefore $\rho/\sigma = \rho'/\sigma'$. Hence the bound of Theorem 4.3 becomes

$$\rho' < \frac{q}{4sn}. \tag{5}$$

5. Provably weak Ring-LWE number fields

Consider the family of polynomials

$$f_{n,q}(x) = x^n + q - 1$$

for q a prime. These satisfy $f(1) \equiv 0 \pmod q$. By the Eisenstein criterion, they are irreducible whenever $q-1$ has a prime factor that appears to exponent 1. These polynomials have discriminant [M] given by

$$(-1)^{\frac{n^2-n}{2}} n^n (q-1)^{n-1}.$$

Proposition 5.1. *Let n be power of a prime ℓ. If $q-1$ is squarefree and $\ell^2 \nmid ((1-q)^n - (1-q))$ then the polynomials $f_{n,q}$ are monogenic.*

Proof. This is a result of Gassert in [Gassert, Theorem 5.1.4]. As stated, Theorem 5.1.4 of [Gassert] requires ℓ to be an odd prime. However, for the monogenicity portion of the conclusion, the proof goes through for $p = 2$. □

Proposition 5.2. *Suppose that $f_{n,q}$ is irreducible, and the associated number field has r_2 complex embeddings. Then $r_2 = n/2$ or $(n-1)/2$ (whichever is an integer), and the normalized spectral norm of M_α^{-1} is exactly*

$$2^{-r_2/n}\sqrt{(q-1)^{1-\frac{1}{n}}}.$$

Proof. Let a be a positive real n-th root of $q-1$. Then the roots of the polynomial are exactly $a\zeta_{2n}^j$ for j odd such that $1 \le j < 2n$. The embeddings take $a\zeta_{2n}$ to each of the other roots. There is $r_1 = 1$ real embedding if n is odd (otherwise $r_1 = 0$), and the rest are r_2

complex conjugate pairs, so that $n = r_1 + 2r_2$. Suppose for that n is not squarefree. Then the dot product of r-th and s-th columns is

$$Re\left(\sum_{k=0}^{n-1} a^{r+s}\zeta_{2n}^{(r+s)(2k+1)}\right) = 0$$

Therefore, the columns of the matrix are orthogonal to one another. Hence, the matrix is diagonalizeable, and its eigenvalues are the lengths of its column vectors, which is for the r-th column,

$$\left(\sum_{k=0}^{n-1} ||a^r\zeta_{2n}^{2k+1}||^2\right)^{1/2} = \sqrt{na^r}$$

Therefore the smallest singular value of M_α is $\sqrt{n}$ and the largest is $\sqrt{na^{n-1}}$. Correspondingly, the largest singular values of M_α^{-1} is $1/\sqrt{n}$.

A standard result of number theory relates the determinant of M_α to the discriminant of K via

$$\det(M_\alpha) = 2^{-r_2}\sqrt{\mathrm{disc}(f_{n,q})},$$

where $r_2 \leq \frac{n}{2}$ is the number of complex embeddings of K. Combining the smallest singular value with this determinant (the discriminant is given explicitly at the beginning of this section) gives the result. □

Theorem 5.3. *Suppose q is prime, n is an integer and $f = f_{n,q}$ satisfies*

(1) n is a power of the prime ℓ,
(2) $q-1$ is squarefree,
(3) $\ell^2 \nmid ((1-q)^n - (1-q))$,
(4) we have $\tau > 1$, where

$$\tau := \frac{q}{2\sqrt{2}sn(q-1)^{\frac{1}{2}-\frac{1}{2n}}}.$$

Then the non-dual Ring-LWE decision problem for f and s (defined by (4)) can be solved in time $\widetilde{O}(\ell q)$ with probability $1 - 2^{-\ell}$, using a dataset of ℓ samples.

Proof. Under the stated conditions, f has a root 1 modulo q, and therefore Poly-LWE is vulnerable to the attack specified in Algorithm 2. The other properties guarantee the applicability of Theorem 4.3 via Proposition 5.1 and 5.2. □

Under the assumption that $q-1$ is infinitely often squarefree, this provides a family of examples which are susceptible to attack (taking, for example, n as an appropriate power of 2; note that in this case item (3) is automatic).

Interestingly, their susceptibility increases as q increases relative to n. It is the ratio $\sqrt{q}/n$, rather than their overall size, which controls the vulnerability (at least as long as q is small enough to run a loop through the residues modulo q).

The quantity τ can be considered a measure of security against this attack ; it should be small to indicate higher security. For the various parameters indicated in Section 2.1, the value of τ is:

parameters	P_{LP1}	P_{LP2}	P_{LP3}	P_{GF}	P_{BCNS}
τ	0.0136	0.0108	0.0090	0.0063	5.0654

The bound on τ in Theorem 4.3 is stronger than what is required in practice for the attack to succeed. In particular, the spectral norm of the transformation M_α^{-1} does not accurately reflect the average behaviour; it is worst case. As n increases, it is increasingly unlikely that error samples happen to lie in just the right direction from the origin to be

inflated by the full spectral norm. Furthermore, we assumed in the analysis of Theorem 4.3 an overly generous bound on the error vectors.

The proof is in the pudding: in Section 9 we have successfully attacked parameters for which $\tau < 0.02$, including P_{LP1}.

6. Heuristics on the prevalence of weak Ring-LWE number fields

In this section, we argue that many examples satisfying Theorem 4.3 are very likely to exist. In fact, each of the individual conditions is fairly easy to attain. We will see in what follows that given a random monogenic number field, there is with significant probability at least one prime q for which Ring-LWE is vulnerable (i.e. the bound (5) is attained) for parameters comparable to those of P_{BNCS}. Note that in this parameter range, the spectral norm is not feasible to compute directly.

6.1. Monogenicity.

Monogenic fields are expected to be quite common in the following sense. If f is taken to be a random polynomial (i.e. its coefficients are chosen randomly), then it is expected that with probability $\pi^2/6$, P will be the ring of integers of a number field [K]. In particular, if f has squarefree discriminant, this will certainly happen. Furthermore, cyclotomic fields are monogenic, as are the families described in the last section.

However, at degrees $n \sim 2^{10}$, the discriminant of f is too large to test for squarefreeness, so testing for monogenicity may not be feasible. Kedlaya has developed a method for constructing examples of arbitrary degree [K].

6.2. Examples, $n = 2^{10}$, $q \sim 2^{32}$.

Consider the following examples:

$$f(x) = x^{1024} + (2^{31} + 14)x + 2^{31}, \quad q = 4294967311,$$
$$f(x) = x^{1024} + (2^{31} + 2^{30} + 22)x + (2^{31} + 2^{30}), \quad q = 6442450967,$$
$$f(x) = x^{1024} + (2^{31} + 2^{30} + 29)x + (2^{31} + 2^{30} + 5), \quad q = 6442450979.$$

These examples are discussed at greater length in Section 7.2, where it is explained that $f(1) \equiv 0 \pmod{q}$ in each case.

In this size range, it is infeasible to verify the spectral norm of K. In the next few sections we will make persuasive heuristic arguments that it can be expected to have ρ' well within the required bound (5), i.e. $\rho' < 2^{17}$. That is, we expect these examples and others like them to be vulnerable.

6.3. Heuristics for the spectral norm.

To find large q requires taking more complex polynomials f, which in turn may inflate the spectral norm, so the complexity of f must be balanced.

One approach is to consider polynomials of the form $f(x) = x^n + ax + b$. Let us recall a standard result of number theory. For a number field K with r_1 real embeddings and r_2 conjugate pairs of complex embeddings, the determinant of the canonical embedding is $\sqrt{\Delta_K}2^{-r_2}$. Therefore, if $\Delta_K > 2^{2r_2}$ (call this Assumption A), we obtain $\det(M_f) > 1$. Then $||M_f||_2 > 1$. We are interested in the spectral norm of the inverse:

$$||M_f^{-1}||_2 \leq k(M_f)/||M_f||_2 \leq k(M_f),$$

where $k(M_f)$ represents the condition number of M_f. Now,

$$\rho' = ||M_f^{-1}||_2 \det(M_f)^{1/n}.$$

As mentioned above, $\det(M_f)$ is given in terms of Δ_K. Under the assumption that $\mathbb{Z}[X]/f(X)$ is indeed a ring of integers, $\Delta_K = \mathrm{Disc}(f)$ (call this Assumption B). By [M],

$$\mathrm{Disc}(f) = (n-1)^{n-1}a^n + (-1)^{n-1}n^n(b+1)^{n-1}.$$

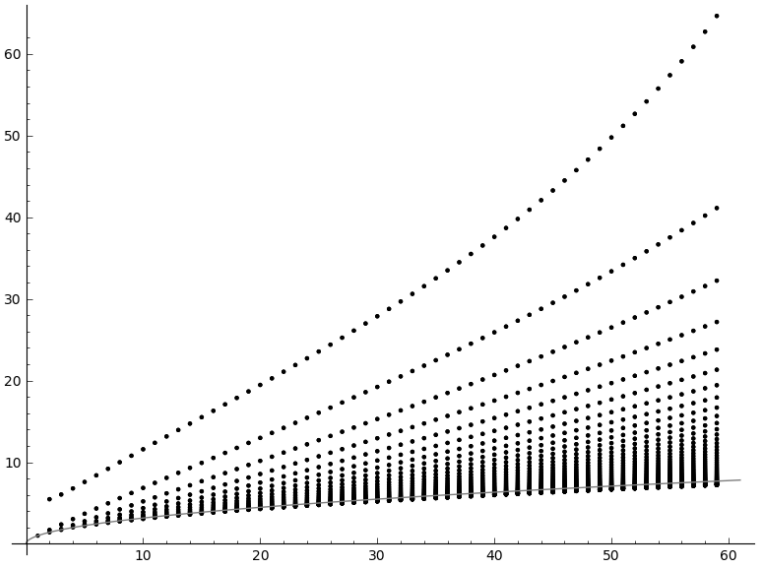
It is evident that by judicious choice of a and b, it is possible to obtain a range of discriminant sizes. We can expect there to be plenty of examples in the range $n^2 < \Delta_K < n^3$ (in this range, Assumption A is satisfied). Then we obtain

$$\rho' \leq 2k(M_f).$$

The condition number of M_f is hard to access theoretically, but heuristically, for random perturbations of any fixed matrix, most perturbations are well-conditioned (having small condition number) [TV]. The matrix M_f is a perturbation of M_p for $p = x^n + 1$. The extent of this perturbation can be bounded in terms of the coefficients a and b, since the perturbation is controlled by the perturbation in the roots of the polynomial. It is a now-standard result in numerical analysis, due to Wilkinson, that roots may be ill-conditioned in this sense, but the condition number can be bounded in terms of the coefficients a and b. This implies that, heuristically, $k(M_f)$ is likely to be small quite frequently.

In conclusion, we expect to find that many $f(x)$ will have ρ' quite small.

6.4. **Experimental evidence for the spectral norm.** Experiments are only feasible in a smaller range that the cryptographic one. The polynomials $x^{32} + ax + b$, $-60 \leq a, b \leq 60$ were plotted on a $\max\{a, b\}$-by-ρ' plane. The result is as follows:



There are some examples with quite high ρ', but the majority cluster low. The grey line is $y = \sqrt{x}$. Therefore, we may conjecture based on this experiment, that we may expect to find plenty of f satisfying $\rho' < \sqrt{\max\{a, b\}}$.

Experimentally, we may guess that the examples of Section 6.2, for which $n = 2^{10}$ and $\max\{a, b\} \leq 2^{30}$, will frequently satisfy $\rho' < 2^{15}$, which is the range required by Theorem 4.3. (Note that the coefficients cannot be taken smaller if f is to have root 1 modulo a prime $q \sim 2^{31}$.)

7. Weak Poly-LWE number fields

7.1. **Finding f and q with roots of small order.** It is relatively easy to generate polynomials f and primes q for which f has a root of given order modulo q. There are two approaches: given f, find suitable q; and given q, find suitable f. Since there are other conditions one may require for other reasons (particularly on f), we focus on the first of these.

Given f, in order to find q such that f has a root of small order (this includes the cases $\alpha = \pm 1$), the following algorithm can be applied.

Algorithm 3 Finding primes q such that $f(x)$ has a root of small order modulo q

Input: A non-cyclotomic irreducible polynomial $f(X) \in \mathbb{Z}[X]$; and an integer $m \geq 1$.
Output: A prime q such that $f(X)$ has a root of order m modulo q.

(1) Let $\Phi_m(X)$ be the cyclotomic polynomial of degree m. Apply the extended Euclidean algorithm to $f(X)$ and $\Phi_m(X)$ over the ring $\mathbb{Q}[X]$ to obtain $a(X)$, $b(X)$ such that
$$a(X)f(X) + b(X)\Phi_m(X) = 1.$$
(Note that 1 is the GCD of $f(X)$ and $\Phi_m(X)$ by assumption.)
(2) Let d be the least common multiple of all the denominators of the coefficients of a and b.
(3) Factor d.
(4) Return the largest prime factor of d.

It is also possible to generate examples by first choosing q and searching for appropriate f. For example, taking $f(x) = \Phi_m(x)g(x) + q$ where $g(x)$ is monic of degree $m - n$ suffices. Both methods can be adapted to find f having any specified root modulo q.

7.2. **Examples, $n \sim 2^{10}$, $q \sim 2^{32}$.** For the range $n \sim 2^{10}$, we hope to find $q \sim 2^{32}$. Examples were found by applying Algorithm 3 to polynomials $f(x)$ of the form $x^n + ax + b$ for a, b chosen from a likely range. Examples are copious and not difficult to find (see Appendix A for code).

Case $\alpha = 1$. A few typical examples of irreducible f with 1 as a root modulo q are:
$$f(x) = x^{1024} + (2^{31} + 14)x + 2^{31}, \quad q = 4294967311,$$
$$f(x) = x^{1024} + (2^{31} + 2^{30} + 22)x + (2^{31} + 2^{30}), \quad q = 6442450967,$$
$$f(x) = x^{1024} + (2^{31} + 2^{30} + 29)x + (2^{31} + 2^{30} + 5), \quad q = 6442450979.$$
These examples satisfy condition 1 of Proposition 3.2 with $\sigma = 3$, hence are vulnerable.

Case $\alpha = -1$. Here is an irreducible f with root -1:
$$f(x) = x^{1024} + (2^{31} + 9)x - (2^{31} + 7), \quad q = 4294967311 \sim 2^{32}.$$
This example similarly satisfies condition 1 of Proposition 3.2 and so is vulnerable.

Case α small order. Here is an irreducible f with a root of order 3:
$$f(x) = x^{1024} + (2^{16} + 2)x - 2^{16}, \quad q = 1099514773507 \sim 2^{40}.$$
This example has $q \sim 2^{40}$; taking this larger q allows us to satisfy (2) of Proposition 3.1 and hence it is vulnerable to Algorithm 1.

7.3. **Examples of weak Poly-LWE number fields with additional properties.** In this section we will give examples of number fields $K = \mathbb{Q}[x]/(f(x))$ which are vulnerable to our attack on Poly-LWE. They will be vulnerable by satisfying one of the following two possible conditions:

R: $f(1) \equiv 0 \pmod{q}$.
R′: f has a root of small order modulo q.

We must also require:

Q: The prime q can be chosen suitably large.

The examples we consider are cyclotomic fields and therefore Galois and monogenic. One should note that guaranteeing these two conditions together is nontrivial in general.

In addition to these, there are additional conditions for the attack explained in [EHL]. The desirable conditions are:

G: K is Galois.
M: K is monogenic.
S: The ideal (q) splits completely in the ring of integers R of K, and $q \nmid [R : \mathbb{Z}[\beta]]$.
O: The transformation between the canonical embedding of K and the power basis representation of K is given by a scaled orthogonal matrix.

Conditions G and S are needed for the Search-to-Decision reduction and Conditions M and O are needed for the Ring-LWE to Poly-LWE reduction in [EHL].

Note that checking the splitting condition for fields of cryptographic size is not computationally feasible in general. However, we are able to give a sufficient condition for certain splittings which is quite fast to check.

Proposition 7.1. *Using the notation as above, if* $f(2) \equiv 0 \pmod q$ *then* q *splits in* R.

Proof. Since $2^{2^{k-1}} \equiv -1 \pmod q$, it follows that $(2^\alpha)^{2^{k-1}} \equiv (-1)^\alpha \equiv -1 \pmod q$ for all odd α in $\mathbb{Z}$. We'll show that $2, 2^3, 2^5, \ldots, 2^m$ where $m = 2^k - 1$ are all distinct mod q, hence showing that $f(x)$ has 2^{k-1} distinct roots mod q i.e. $f(x)$ splits mod q. Assume that $2^i \equiv 2^j \pmod q$ for some $1 \leq i < j \leq 2^k - 1$. Then $2^{j-i} \equiv 1 \pmod q$, which means that the order of 2 modulo q divides $j - i$. However, by the fact below (Lemma 7.2), the order of 2 mod q is 2^k, which is a contradiction since $j - i < 2^k$. □

Lemma 7.2. *Let* q *be a prime such that* $2^{2^{k-1}} \equiv -1 \pmod q$ *for some integer* k. *Then the order of* 2 *modulo* q *is* 2^k.

Proof. Let a be the order of 2 modulo q. By assumption $(2^{2^{k-1}})^2 \equiv 2^{2^k} \equiv 1 \pmod q$. Then $a | 2^k$ i.e. $a = 2^\alpha$ for some $\alpha \leq k$. Say $\alpha \leq k-1$. Then $1 = (2^{2^\alpha})^{2^{k-1-\alpha}} = 2^{2^{k-1}} \equiv -1 \pmod q$, a contradiction. □

The converse of Proposition 7.1 does not hold. For instance, let K be the splitting field of the polynomial $x^8 + 1$ and $q = 401$. Then q splits in R. However $f(2) = 257 \not\equiv 0 \pmod q$.

We now present a family of examples for which $\alpha = -1$ is a root of f of order two. Conditions G, M, S, R′ (order 2) and Q are all satisfied. The field K is the cyclotomic number field of degree $\phi(2^k) = 2^{k-1}$, but instead of the cyclotomic polynomial we take the minimal polynomial of $\zeta_{2^k} + 1$. In each case, q is obtained by factoring $2^{2^{k-1}} + 1$ for various values of k and splitting is verified using Proposition 7.1.

k	2	3	4	5	6	7	7	8	8
q	5	17	257	$65537 \sim 2^{16}$	$6700417 \sim 2^{22}$	$274177 \sim 2^{18}$	$q_5 \sim 2^{45}$	$q_6 \sim 2^{55}$	$q_1 \sim 2^{72}$

k	9	9	10	10	10	11	11	11
q	$q_7 \sim 2^{50}$	$q_2 \sim 2^{205}$	$2424833 \sim 2^{21}$	$q_3 \sim 2^{162}$	$q_4 \sim 2^{328}$	$q_8 \sim 2^{25}$	$q_9 \sim 2^{32}$	$q_{10} \sim 2^{131}$

The examples in the last few rows are of cryptographic size[1], i.e. the field has degree 2^{10} and the prime is of size $\sim 2^{32}$ or greater. These provide examples which are weak against our Poly-LWE attack, by Proposition 3.2.

[1] $q_1 = 5704689200685129054721$, $q_2 = 934616397153579777691635581996068965840512375416381885802803 21$, $q_3 = 74556028256478842083373957362004549187833663426 57$, $q_5 = 67280421310721$, $q_6 = 59649589127497217$ $q_4 = 74164006262753080152478714190193747405994078109751902390582131614441575950470500809281871169394073 7$ $q_7 = 1238926361552897$, $q_8 = 45592577$, $q_9 = 6487031809$, $q_{10} = 46597757852200185432645607430767781928 97$

8. Cyclotomic (in)vulnerability

One of our principal observations is that the cyclotomic fields, used for Ring-LWE, are uniquely protected against the attacks presented in this paper. The next proposition states that the polynomial ring of the m-th cyclotomic polynomial Φ_m will never be vulnerable to the attack based on a root of small order.

Proposition 8.1. *The roots of Φ_m have order m modulo every prime q.*

Proof. Consider the field $\mathbb{F}_q$, q prime. Since $\mathbb{F}_q$ is perfect, the cyclotomic polynomial $\Phi_m(x)$ has $\phi(m)$ roots in an extension of $\mathbb{F}_q$. This polynomial has no common factor with $x^k - 1$ for $k < m$. However, it divides $x^m - 1$. Therefore its roots have order dividing m, but not less than m. That is, its roots are all of order exactly m in the field in which they live. Now, if we further assume that $\Phi_m(x)$ splits modulo q, then its $\phi(m)$ roots are all elements of order m modulo q, so in particular, $m \mid q - 1$. The roots of $\Phi_m(x)$ are all elements of $\mathbb{Z}/q\mathbb{Z}$ of order exactly m. □

The question remains whether there is another polynomial representation for the ring of cyclotomic integers for which f does have a root of small order. This may in fact be the case, but the error distribution is transformed under the isomorphism to this new basis, so this does not guarantee a weakness in Poly-LWE for Φ_m.

However, it is not necessary to search for all such representations to rule out the possibility that this provides an attack. The ring $R_q \cong \mathbb{F}_q^n$ has exactly $n = \phi(m)$ homomorphisms to $\mathbb{Z}/q\mathbb{Z}$. If R_q can be represented as $(\mathbb{Z}/q\mathbb{Z})[X]/f(X)$ with $f(\alpha) = 0$, then the map $R_q \to \mathbb{Z}/q\mathbb{Z}$ is given by $p \mapsto p(\alpha)$ is one of these n maps. It suffices to write down these n maps (in terms of any representation!) and verify that the errors map to all of $\mathbb{Z}/q\mathbb{Z}$ instead of a small subset. It is a special property of the cyclotomics that these n homomorphisms coincide. Thus we are reduced to the case above.

9. Successfully coded attacks

The following table documents Ring-LWE and Poly-LWE parameters that were successfully attacked on a Thinkpad X220 laptop with Sage Mathematics Software [S], together with approximate timings. For code, see Appendix A. The first row indicates that cryptographic size is attackable in Poly-LWE. The second row indicates that a generic example attackable by Poly-LWE is also susceptible to Ring-LWE (see Section 6). We were unable to test the Ring-LWE attack for $n > 256$ only because Sage's inbuilt Discrete Gaussian Sampler was not capable of initializing (thus we were unable to produce samples to test). The last two rows illustrate the τ of Theorem 4.3 that is required for security in practice (approximately $\tau < 0.013$ instead of $\tau < 1$ in theory). In the Ring-LWE rows, parameters were chosen to illustrate the boundary of feasibility for a fixed n. Since the feasibility of the attack depends on the ratio $\sqrt{q}/n$, there is no reason to think larger n are invulnerable (provided q also grows), but we were unable to produce samples to test against. The Poly-LWE example illustrates that runtime for large q is feasible (runtimes for Poly-LWE and Ring-LWE are the same; it is only the samples which differ).

case	f	q	s	τ	samples per run	successful runs	time per run
Poly-LWE	$x^{1024} + 2^{31} - 2$	$2^{31} - 1$	3.192	N/A	40	1 of 1	13.5 hrs
Ring-LWE	$x^{128}+524288x$ $+524285$	524287	8.00	N/A	20	8 of 10	24 sec
Ring-LWE	$x^{192} + 4092$	4093	8.87	0.0136	20	1 of 10	25 sec
Ring-LWE	$x^{256} + 8189$	8190	8.35	0.0152	20	2 of 10	44 sec

References

[IEEE] P1363.1: Standard Specifications for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices, December 2008
`http://grouper.ieee.org/groups/1363/`

[BCNS] Joppe W. Bos, Craig Costello, Michael Naehrig, Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem, Preprint, 2014. http://eprint.iacr.org/2014/599.pdf

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

[BL+] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, Damien Stehlé. Classical Hardness of Learning with Errors, In Proceedings of STOC 2013.

[BV] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, CRYPTO, volume 6841 of Lecture Notes in Computer Science, pages 505–524. Springer, 2011.

[BGV] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping, Cryptology ePrint Archive, Report 2011/277, 2011, ITCS 2012.

[DD] Léo Ducas, Alain Durmus. Ring-LWE in Polynomial Rings. 15th International Conference on Practice and Theory in Public Key Cryptography, PKC 2012, Fischlin, Marc, Buchmann, Johannes, Manulis, Mark (Eds.). Lecture Notes in Computer Science, Vol. 7293,

[EHL] Kirsten Eisentraeger, Sean Hallgren, Kristin Lauter, Weak Instances of PLWE, Proceedings of Selected Areas of Cryptography 2014, Springer LNCS.

[Gassert] T. Alden Gassert, Prime Decomposition in Iterated Towers and Discriminant Formulae, PhD. Thesis, University of Massachusetts, Amherst, 2014

[GF] N. Göttert, T. Feller, M. Schneider, J. Buchmann, S. Huss, On the Design of Hardware Building Blocks for Modern Lattice-Based Encryption Schemes. In Cryptographic Hardware and Embedded Systems CHES 2012, Lecture Notes in Computer Science Volume 7428, 2012, pp 512-529.

[GHS] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead, Cryptology ePrint Archive, Report 2011/566, 2011, Eurocrypt 2012.

[GLN] Thore Graepel, Kristin Lauter, and Michael Naehrig. ML Confidential: Machine Learning on Encrypted Data, International Conference on Information Security and Cryptology – ICISC 2012, Lecture Notes in Computer Science, Springer Verlag, 26 December 2012.

[HPS] Jeff Hoffstein, Jill Pipher, and Joseph Silverman. NTRU: a ring based public key cryptosystem. In Proceedings of ANTS-III, volume 1423 of Lecture Notes in Computer Science, pages 267–288. Springer, June 1998.

[K] Kiran Kedlaya. A construction of polynomials with squarefree discriminants, http://arxiv.org/abs/1103.5728.

[LP] Richard Lindner, Chris Peikert. Better Key Sizes (and Attacks) for LWE-Based Encryption. In Topics in Cryptology CT-RSA 2011 Lecture Notes in Computer Science Volume 6558, 2011, pp 319-339.

[LPR] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. Full version of the 2010 Eurocrypt version.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A Toolkit for Ring-LWE Cryptography, Eurocrypt 2013.

[M] D. W. Masser. 3136. The Discriminants of Special Equations. The Mathematical Gazette, 50(372):158-160 (May 1966).

[MR04] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measure. SIAM J. on Computing, 37(1):267–302 (May 2007). Preliminary version in FOCS 2004.

[MR09] D. Micciancio and O. Regev. Lattice-based cryptography. In Post Quantum Cryptography, pages 147–191. Springer, February 2009.

[PG] T. Pöppelmann and T. Güneysu, Towards Practical Lattice-Based Public-Key Encryption on Reconfigurable Hardware. In Selected Areas in Cryptography – SAC 2013 Lecture Notes in Computer Science 2014, pp 68-85

[R] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. J. ACM, 56(6):1–40, 2009 (Preliminary version STOC 2005)

[RV] S.S. Roy, F. Vercauteren, N. Mentens, D. D. Chen and I. Verbauwhede, Compact Ring-LWE Cryptoprocessor. In Cryptographic Hardware and Embedded Systems CHES 2014 Lecture Notes in Computer Science Volume 8731, 2014, pp 371-391

[RS] M. Rückert and M. Schneider. Selecting secure parameters for lattice-based cryptography. Cryptology ePrint Archive, Report 2010/137, 2010.

[S] W. A. Stein et al. Sage Mathematics Software (Version 6.4.1), The Sage Development Team, 2014, http://www.sagemath.org.

[SS] Damien Stehlé and Ron Steinfeld. Making NTRUEncrypt and NTRUSign as secure as worst-case problems over ideal lattices, Full version Eurocrypt 2011

[TV] Terence Tao, Van Vu. Smooth analysis of the condition number and the least singular value. http://arxiv.org/abs/0805.3167

Appendix A. Appendix: Code

A.1. **Proof of concept for Ring-LWE and Poly-LWE attacks.** The following Sage Mathematical Software [S] code verifies that Algorithm 2 succeeds on the Poly-LWE and Ring-LWE examples of Section 9. Note that Algorithm 1 is a minor modification of Algorithm 2.

This code relies on `DiscreteGaussianDistributionLatticeSampler`, an inbuilt package in Sage. The sampler is incapable of initializing in sufficiently large dimension to fully test the attacks in this paper. See the related trac ticket http://trac.sagemath.org/ticket/17764.

Built into the code are several error checks that will be triggered if insufficient precision is not used.

This code is available in electronic form at http://math.colorado.edu/~kstange/scripts.html.

```
##################################################
# RING-LWE ATTACK #
##################################################

# General preparation of Sage:  Create a polynomial ring and import GaussianSampler, Timer
P.<y> = PolynomialRing(RationalField(), 'y')
from sage.stats.distributions.discrete_gaussian_lattice import DiscreteGaussianDistributionLatticeSampler
RP = RealField(300) # this sets the precision; if it is insufficient, the implementation won't be valid
from sage.doctest.util import Timer

# Give the Minkowski lattice for a given ring determined by a polynomial.
# Also gives a key to which are real embeddings.
def cmatrix(): # returns a matrix, columns basis 1, x, x^2, x^3, ... given in the canonical embedding
    global N, a
    N.<a> = NumberField(f)
    fdeg = f.degree()
    key = [0 for i in range(fdeg)] # 0 = real, 1 = real part of complex emb, 2 = imaginary part
    embs = N.embeddings(CC)
    M = matrix(RP,fdeg,fdeg)
    print "Preparing an embedding matrix:  computing powers of the root."
    apows = [ a^j for j in range(n) ]
    print "Finished computing the powers of the root."
    i = 0
    while i < n:
        em = embs[i]
        if Mod(i,20)==Mod(0,20) or Mod(i,20)==Mod(1,20):
            print "Embedding matrix: ", i, " rows out of ", n, " complete."
        if em(a).imag() == 0:
            key[i] = 0
            for j in range(n):
                M[i,j] = em(apows[j]).real()
            i = i + 1
        else:
            key[i] = 1
            key[i+1] = 2
            for j in range(n):
                M[i,j] = em(apows[j]).real()
                M[i+1,j] = (em(apows[j])*I).real()
            i = i + 2
    return M, key

# Produce a random vector from (Z/qZ)^n
def random_vec(q, dim):
    return vector([ZZ.random_element(0,q) for i in range(dim)])

# Useful function for real numbers modulo q
def modq(r,q):
```

```
    s = r/q
    t = r/q - floor(r/q)
    return t*q

# Call sampler
def call_sampler():
    e = sampler().change_ring(RP)
    return e

# Create samples using a lattice (given by latmat and its inverse),
# a Gaussian sampler on that lattice, secret, prime
def get_sample(latmat, latmatinv, sec, qval, keyval):
    e = call_sampler() # create error, in R^n
    dim = latmat.dimensions()[0] # detect dimension of lattice
    pre_a = random_vec(qval, dim) # create a uniformly randomly in terms of basis in cm
    a = latmat*pre_a # create a, in R^n
    b = vecmul_poly(a,sec,latmat,latmatinv) + e # create b, in R^n
    pre_b = latmatinv*b # move to basis in cm in order to reduce mod q
    pre_b_red = vector([modq(c,qval) for c in pre_b])
    b = latmat*pre_b_red
    return [a, b]

# Global choices: setup a field and prime, sampler.
# Set to dummy values that will be altered when an attack is run
q = 1
n = 1
sig = 1/sqrt(2*pi)
Zq = IntegerModRing(q)
R.<x> = PolynomialRing(Zq)
f = y + 1
N.<a> = NumberField(f)
S.<z> = R.quotient(f) # This is P_q
cm,key = cmatrix()
cmi = cm.inverse()
cm
cm53 = cm.change_ring(RealField(10))
cmqq = cm53.change_ring(QQ)
sampler = DiscreteGaussianDistributionLatticeSampler(cmqq.transpose(), sig)

# Set the parameters for the attack
def setup_params(fval,qval,sval):
    global q,n,sig,f,S,x,z,Zq
    f = fval
    n = f.degree()
    q = qval
    Zq = IntegerModRing(q)
    R.<x> = PolynomialRing(Zq)
    sig = sval/sqrt(2*pi)
    S.<z> = R.quotient(f)
    print "Setting up parameters, polynomial = ", f, " and prime = ", q, " and sigma = ", sig
    print "Verifying properties:  "
    print "Prime?", q.is_prime()
    print "Irreducible? ", f.is_irreducible()
    print "Value at 1 modulo q?", Mod(f.subs(y=1),q)
    return True

# Compute the lattices in Minkowski space
def prepare_matrices():
    global cm, key, cmi, cmqq
    print "Preparing matrices."
    cm,key = cmatrix()
    print "Embedding matrix prepared."
    cmi = cm.inverse()
    print "Inverse matrix found."
    cm53 = cm.change_ring(RealField(10))
    cmqq = cm53.change_ring(QQ)
    print "All matrices prepared."
    return True

# Make a vector in R^n into a polynomial, given change of basis matrix and variable to use
def make_poly(a,matchange,var):
    coeffs = matchange*a  #coefficients of the polynomial are given by the change of basis matrix
    pol = 0
    for i in range(n):
        pol = pol + ZZ(round(coeffs[i]))*var^i # var controls where it will live (what poly ring)
    return pol
```

```
# Make a polynomial into a vector in Minkowski space
def make_vec(fval,matchange):
    if fval == 0:
        coeffs = [0 for i in range(n)]
    else:
        coeffs = [0 for i in range(n)]
        colist = lift(fval).coefficients()
        for i in range(len(colist)):
            coeffs[i] = ZZ(colist[i])
    return matchange*vector(coeffs)

# Multiplication in the Minkowski space via moving to polynomial ring
def vecmul_poly(u,v,mat,matinv):
    poly_u = make_poly(u,matinv,z)
    poly_v = make_poly(v,matinv,z)
    poly_prod = poly_u*poly_v
    return make_vec(poly_prod,mat)

# Create the sampler on the lattice embedded in R^n
def initiate_sampler():
    global sampler
    print "Initiating Sampler."
    sampler = DiscreteGaussianDistributionLatticeSampler(cmqq.transpose(), sig)
    print "Sampler initiated with sigma", RDF(sig)
    return True

# Produce error vectors, just a test to see how they look
def error_test(num):
    print "Testing the error vector production by producing ", num, " errors."
    errorlist = [sampler().norm().n() for _ in range(num)]
    meannorm = mean(errorlist) # average norm
    maxnorm = max(errorlist) # maximum norm
    print "The average error norm is ", RDF(meannorm/( sqrt(n)*sampler.sigma*sqrt(2*pi) )), " times sqrt(n)*s."
    maxratio = RDF(maxnorm/( sqrt(n)*sampler.sigma*sqrt(2*pi) ))
    print "The maximum error norm is ", maxratio, " times sqrt(n)*s."
    if maxratio > 1:
        print "~~~~~~~~~~~~~~~~~~~~~~~ ERROR ~~~~~~~~~~~~~~~~~~~~~~~~"
        print "The errors do not satisfy a proven upper bound in norm."
    return True

# Create the secret
secret = 0
def create_secret():
    global secret
    secret = cm*random_vec(q,n)
    return True

# Produce samples
samps = []
numsamps = 1
def create_samples(numsampsval):
    global samps, numsamps
    samps = []
    print "Creating samples"
    for i in range(numsampsval):
        print "Creating sample number ", i
        samp = get_sample(cm, cmi, secret, q, key)
        samps.append(samp)
    numsamps = len(samps)
    print "Done creating ", numsamps, "samples."
    return True

# Function for going down to q
def go_to_q(a,matchange):
    pol = make_poly(a,matchange,x)
    #print "debug got pol:", pol
    pol_eval = pol.subs(x=1)
    #print "debug eval'd to:", pol_eval, " and then ", Zq((pol_eval))
    return Zq(pol_eval)

# Check to make sure moving to q preserves product -- the last two lines should be equal
def sanity_check():
    print "Initiating sanity check"
    mat = cmi
    pvec1 = random_vec(q,n)
```

```
    vec1 = cm*pvec1
    pvec2 = random_vec(q,n)
    vec2 = cm*pvec2
    vprod2 = vecmul_poly(vec1,vec2,cm,cmi)
    first_thing = go_to_q(vprod2,mat)
    second_thing = go_to_q(vec1,mat)*go_to_q(vec2,mat)
    if first_thing == second_thing:
        print "Sanity confirmed."
    else:
        print "~~~~~~~~~~~~~~~~~~~~~~~ ERROR ~~~~~~~~~~~~~~~~~~~~~~~~"
        print "Sanity problem:", first_thing, " is not equal to ", second_thing, "."
        print "Are you sure your ring has root 1 mod q?"
    return True

# Given a list of elements of Z/qZ, make a histogram and zero count
def histoq(data):
    hist = [0 for i in range(10)] # empty histogram
    zeroct=0 # count of zeroes mod q
    for datum in data:
        e = datum
        if e == 0:
            zeroct = zeroct+1
        histbit = floor(ZZ(e)*10/q)
        hist[histbit]=hist[histbit]+1
    return [hist, zeroct]

# Given a list of vectors in R^n, create a histogram of their
# values in Z/qZ under make_poly, together with a zero count
def histo(data,cmi):
    return histoq([go_to_q(datum,cmi) for datum in data])

# Create a histogram of error vectors, transported to polynomial ring
def histogram_of_errors():
    print "Creating a histogram of errors mod q."
    errs = []
    for i in range(80):
        errs.append(sampler())
    hist = histo(errs,cmi)
    print "The number of error vectors that are zero:", hist[1]
    bar_chart(hist[0], width=1).show(figsize=2)
    return True

# Create a histogram of the a's in the samples, transported to polynomial ring
def histogram_of_as():
    print "Creating a histogram of sample a's mod q."
    a_vals = [samp[0] for samp in samps]
    hist = histo(a_vals,cmi)
    print "The number of a's that are zero:", hist[1]
    bar_chart(hist[0], width=1).show(figsize=2)
    return True

# Create a histogram of errors by correct guess
def histogram_of_errors_2():
    print "Creating a histogram of supposed errors if sample is guessed, mod q."
    hist = histoq([ lift(Zq(go_to_q(sample[1],cmi) - go_to_q(sample[0],cmi)*go_to_q(secret,cmi))) for sample in samps])
    print "The number of such that are zero:", hist[1]
    bar_chart(hist[0], width=1).show(figsize=2)
    return True

# Create the secret mod q
lift_s = 0
def secret_mod_q():
    global lift_s
    lift_s = go_to_q(secret,cmi)
    print "Storing the secret mod q.  The secret is ", secret, " which becomes ", lift_s
    return True

# Algorithm 2
# reportrate controls how often it updates the status of the loop; larger = less frequently
# quickflag = True will run only the secret and a few other values to give a quick idea if it works
def alg2(reportrate, quickflag = False):
    print "Beginning algorithm 2."
    numsamps = len(samps)
    a = [ 0 for i in range(numsamps)]
    b = [ 0 for i in range(numsamps)]
    print "Moving samples to F_q."
```

```
    for i in range(numsamps):
        sample = samps[i]
        a[i] = go_to_q(sample[0],cmi)
        b[i] = go_to_q(sample[1],cmi)
    possibles = []
    winner = [[],0]
    print "Samples have been moved to F_q."
    for i in range(2):
        if i == 0:
            print "!!!!! ROUND 1: !!!!! First, checking how many samples the secret survives (peeking ahead)."
            iterat = [lift_s]
        if i == 1:
            print "!!!!! ROUND 2: !!!!! Now, running the attack naively."
            possibles = []
            if quickflag:
                print "We are doing it quickly (not a full test)."
                iterat = xrange(1000)
            else:
                iterat = xrange(q)
        for g in iterat:
            if Mod(g,reportrate) == Mod(0,reportrate):
                print "Currently checking residue ", g
            g = Zq(g)
            potential = True
            ctr = 0
            while ctr < numsamps and potential:
                e = abs(lift(Zq(b[ctr]-g*a[ctr])))
                if e > q/4 and e < 3*q/4:
                    potential = False
                    if ctr == winner[1]:
                        winner[0].append(g)
                        print "We have a new tie for longest chain:", g, " has survived ", ctr, " rounds."
                    if ctr > winner[1]:
                        winner = [[g],ctr]
                        print "We have a new longest chain of samples survived:", g, " has survived ", ctr, " rounds."
                ctr = ctr + 1
            if potential == True:
                print "We found a potential secret: ", g
                possibles.append(g)
            if g == lift_s:
                if i == 0:
                    print "The real secret survived ", ctr, "samples."
                #break
    print "Full list of survivors of the ", numsamps, " samples:", possibles
    print "The real secret mod q was: ", lift_s
    if len(possibles) == 1 and possibles[0] == lift_s:
        print "Success!"
        return True
    else:
        print "Failure!"
        return False

# Run a simulation.
def shebang(fval,qval,sval,numsampsval,numtrials,quickflag=False):
    global sig
    print "Welcome to the Ring-LWE Attack."
    n = fval.degree()
    print "The attack should theoretically work if the following quantity is greater than 1."
    print "Quantity: ", RDF( qval/( 2*sqrt(2)*sval*n*(qval-1)^( (n-1)/2/n) ) )
    timer = Timer()
    timer2 = Timer()
    timer.start()
    print "********** PHASE 1: SETTING UP SYSTEM "
    setup_params(fval,qval,sval)
    prepare_matrices()
    print "Computing the adjustment factor for s."
    cembs = (n - len(N.embeddings(RR)))/2
    detscale = RP( ( 2^(-cembs)*sqrt(abs(f.discriminant())) )^(1/n) ) # adjust the sigma,s
    sval = sval*detscale
    sig = sig*detscale
    print "Adjusted s for use with this embedding, result is ", sval
    initiate_sampler()
    print "The sampler has been created with sigma = ", sampler.sigma
    print "Sampled vectors will have expected norm ", RDF(sqrt(n)*sampler.sigma)
    error_test(5)
    print "Time for Phase 1: ", timer.stop()
```

```
    timer.start()
    count_successes = 0
    timer2.start()
    for trialnum in range(numtrials):
        print "*~*~*~*~*~*~*~*~*~*~*~* TRIAL NUMBER ", trialnum, "*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~"
        print "********** PHASE 2: CREATE SECRET AND SAMPLES"
        create_secret()
        create_samples(numsampsval)
        sanity_check()
        print "Time for Phase 2: ", timer.stop()
        timer.start()
        print "********** PHASE 3: HISTOGRAMS"
        histogram_of_errors()
        print "The histogram of errors (above) should be clustered at edges for success."
        histogram_of_as()
        print "The histogram of a's (above) should be fairly uniform."
        histogram_of_errors_2()
        print "The histogram of sample errors (above) should be clustered at edges for success."
        print "Time for Phase 3: ", timer.stop()
        timer.start()
        print "********** PHASE 4: ATTACK ALGORITHM"
        secret_mod_q()
        result = alg2(10000,quickflag)
        print "Result of Algorithm 2:", result
        print "Time for Phase 4: ", timer.stop()
        if result == True:
            count_successes = count_successes + 1
        print "*~*~*~*~*~*~*~*~*~*~*~*~*~* ", count_successes, " out of ", trialnum+1, " successes so far. *~*~*~*~*~*"
    totaltime = timer2.stop()
    print "Total time for ", trialnum+1, "trials was ", totaltime
    return count_successes
```

A.2. **Sage code for Algorithm 3.** The following Sage Mathematics Software [S] algorithm returns the largest prime q for which a polynomial f has a root of order m modulo q.

```
x = PolynomialRing(RationalField(), 'x').gen()
def findq(f,m):
    g = x^m-1
    xg = f.xgcd(g)
    cofs = xg[2].coefficients()
    dens = [ a.denominator() for a in cofs ]
    facs = lcm(dens).factor()
    return max([fac[0] for fac in facs ])
```

Yara Elias: Department of Mathematics and Statistics, McGill University, Montreal, Quebec, Canada

E-mail address: yara.elias@mail.mcgill.ca

Kristin E. Lauter:Microsoft Research, One Microsoft Way, Redmond, WA 98052

E-mail address: klauter@microsoft.com

Ekin Ozman: Department of Mathematics, Faculty of Arts and Science, Bogazici University, 34342, Bebek-Istanbul, Turkey

E-mail address: ekin.ozman@boun.edu.tr

Katherine E. Stange: Department of Mathematics, University of Colorado, Campux Box 395, Boulder, Colorado 80309-0395

E-mail address: kstange@math.colorado.edu