A Nonstandard Variant of Learning with Rounding with Polynomial Modulus and Unbounded Samples[*]

Hart Montgomery
Fujitsu Laboratories of America
hmontgomery@us.fujitsu.com

January 22, 2018

Abstract

The *Learning with Rounding Problem* (LWR) has become a popular cryptographic assumption to study recently due to its determinism and resistance to known quantum attacks. Unfortunately, LWR is only known to be provably hard for instances of the problem where the LWR modulus q is at least as large as some polynomial function of the number of samples given to an adversary, meaning LWR is provably hard only when (1) an adversary can only see a fixed, predetermined amount of samples or (2) the modulus q is superpolynomial in the security parameter, meaning that the hardness reduction is from superpolynomial approximation factors on worst-case lattices.

In this work, we show that there exists a (still fully deterministic) variant of the LWR problem that allows for both unbounded queries and a polynomial modulus q, breaking an important theoretical barrier. To our knowledge, our new assumption, which we call the *Nearby Learning with Lattice Rounding Problem* (NLWLR), is the first fully deterministic version of the learning with errors (LWE) problem that allows for both unbounded queries and a polynomial modulus. We note that our assumption is not practical for any kind of use and is mainly intended as a theoretical proof of concept to show that provably hard deterministic forms of LWE can exist with a modulus that does not grow polynomially with the number of samples.

Keywords: Lattices, LWE, Learning with Rounding

1 Introduction

In recent years, the need for quantum-secure cryptographic protocols has seemingly dramatically increased. Many people and organizations, including apparently the NSA, believe that powerful quantum computers will be coming soon. It therefore seems imperative that the cryptographic community develop new, efficient quantum-secure protocols for all of the common use cases of cryptography today.

Most of these protocols are based upon the *Learning with Errors Assumption* (LWE), which was invented by Regev in [Reg05], or its ring variant. Informally, given a fixed random vector $\mathbf{s}$, the LWE assumption states that it is difficult to distinguish samples of the form $(\mathbf{a}_i \mathbf{a}_i{}^\intercal \mathbf{s} + \delta_i)$ from random when the $\mathbf{a}_i$s are uniformly sampled random vectors and the δ_is are fresh samples from a low-norm noise distribution. While it is well-known that the LWE assumption has played a huge role in the development of theoretical crytography, including things like fully homomorphic encryption [BV11], recently LWE has been gaining traction

[*]An edited version of this paper appears in PQCrypto 2018: The Ninth International Conference on Post-Quantum Cryptography.

as an assumption for practical cryptography due to the need for quantum security, including in [BCD+16] and [Pei14]. In fact, it seems like most of the submissions to the NIST post-quantum key exchange competition will be based on LWE (or its ring variants).

But with all of the attention given to the LWE assumption, we think it is worth considering a related, simpler, and more efficient assumption: the learning with rounding (LWR) assumption. The LWR assumption was invented in [BPR12] as a way to build the first nontrivial, parallelizable (and hence low depth) PRFs[1] from lattice assumptions. Informally, given a fixed random vector $\mathbf{s}$, the LWR assumption states that it is difficult to distinguish samples of the form $\left(\mathbf{a}_i \left\lceil \mathbf{a}_i{}^{\intercal}\mathbf{s} \right\rfloor_p\right)$ from random when the $\mathbf{a}_i$s are random vectors sampled uniformly at random. Note that the operation $\lceil \cdot \rfloor_p$ can be thought of as the general rounding operation 'round to the nearest multiple of $\frac{q}{p}$ and then divide by $\frac{q}{p}$'. In the vast majority of cases where the LWE assumption is used, the LWR assumption can be used in its place.

LWR has several advantages over LWE. The most obvious one is that, for fixed dimension and vector distributions (i.e. key and sample distributions), LWR is faster to compute. This is due to the fact that we don't need to sample from the noise distribution when generating an LWR sample, and this noise sampling is typically a major pain point for practical LWE implementations. But LWR is also more resilient than LWE in many ways as well: since it is deterministic, it makes it so an adversary that can somehow trick an oracle into repeating samples (outputting a sample with the same value of $\mathbf{a}_i$ but with a different noise sample) gains nothing. An often-overlooked fact about LWR is that it would likely be much more resistant to side channel attacks than LWE, since noise sampling can be difficult and take a lot of effort to inure against these sorts of attack [RRVV14]. This fact implies that LWR would be a very good post-quantum candidate for certain hardware implementations.

There have been many recent schemes that utilized the power of LWR. A natural use case was lattice-based PRFs, including [BLMR13] and [BP14]. Papers with a practical focus on things like key exchange have also been build using rounding, including [CKLS16] and [DFH+16]. More powerful PRFs that provide increased functionality and also might provide illustrating examples that even touch at things like indistinguishability obfuscation include papers like [BFP+15], [CC17], and [BKM17].

Given all of the benefits of LWR mentioned so far, it seems silly that anyone would ever want to use the LWE assumption instead. However, there is an excellent reason why people use LWE instead of LWR: security. As of now, LWE does not reduce tightly to LWR, and LWR does not have a direct reduction from worst-case lattice problems like LWE does. To reach the same provable security levels as LWE schemes, it is often the case that the parameters of LWR schemes have to be made substantially larger than their LWE counterparts, negating many of the advantages of LWR.

Fundamentally, we think that there does not seem to be any obvious reason why LWR should not be as hard or close to as hard as LWE (although without proofs we can never be sure)[2]. The goal of this work is to continue to try to close the gap between the two problems in terms of hardness. In this work, we do not fully close this gap, or even come up with a primitive that is practically useful–we cannot think of any practical use cases for what we build in this paper. But we do make progress on what we consider an extremely important practical problem in lattice cryptography.

In fact, we think that if the hardness gap between LWR and LWE was closed, then there would be little use for LWE in practice. For instance, if LWR was provably secure for parameters where the 'rounding loss' $\frac{q}{p}$ was not much larger than the noise magnitude for LWE with similar levels of security, many of the very practical NIST post-quantum crypto proposals like Kyber [BDK+17] or Frodo [BCD+16] would probably be redesigned to use

[1]It was previously known how to build completely sequential (and thus high depth) PRFs from PRGs using generic constructions like [GGM84]. It is possible to build a very simple lattice-based PRF using the [GGM84] construction by treating LWE as a PRG.

[2]Caution! This is just an opinion.

LWR. We think that research on the hardness of LWR (or the lack thereof) could have widespread practical application in the long run.

1.1 LWR: Security Background

In order to put our work in context and explain the motivation, it is useful to go over the previous work on LWR and LWR security.

Original LWR Work [BPR12] : the original LWR paper [BPR12] proved the hardness of LWR where the modulus q was superpolynomially larger than the B-bounded noise distribution used in the LWE oracle in the proof. The reduction was straightforward: take an LWE oracle, and round the LWE output. If the rounded LWE samples $\left(\mathbf{a}_i, \lceil \mathbf{a}_i{}^\intercal \mathbf{s} + \delta_i \rfloor_p\right)$ were always equal to the LWR samples $\left(\mathbf{a}_i, \lceil \mathbf{a}_i{}^\intercal \mathbf{s} \rfloor_p\right)$, then the LWR problem was at least as hard as the underlying LWE problem. While this explanation simplifies the reduction, this is the fundamental idea of how it works in this paper.

However, this condition obviously does not hold if the error δ_i causes the rounded LWE output $\lceil \mathbf{a}_i{}^\intercal \mathbf{s} + \delta_i \rfloor_p$ to differ in $\mathbb{Z}_p$ from the rounded LWR output $\lceil \mathbf{a}_i{}^\intercal \mathbf{s} \rfloor_p$. Since we have no way of authoritatively telling whether this bad event happened or not, the authors of [BPR12] required that the modulus q be superpolynomial relative to the B-bounded noise distribution used in the LWE oracle. This ensured that the probability that any $\mathbf{a}_i{}^\intercal \mathbf{s}$ was within B of a 'boundary' where the rounding value changed was negligible for any polynomial amount of LWR samples and allowed the proof to work.

Subsequent Work After the original paper, several further works cleverly improved the state of LWR [AKPW13] [BGM$^+$16] [BLL$^+$15] [AA16]. In general, the authors of these papers showed that, by cleverly sampling and leaking certain secret information and using smart statistical analyses, it was possible to avoid instances where rounded LWE and LWR samples differed in the reduction. These works substantially improved the parameters for LWR in general, but unfortunately still required an apriori bounded number of samples in the case of a polynomial modulus. Let c be a constant[3], $\gamma \geq 1$, let B be the B-bound for the noise distribution of the LWE instance that we reduce to our LWR instance, let p be the rounding parameter (the group $\mathbb{Z}_p$ that we are rounding to), and let κ be a security parameter. The table below (with format borrowed from [AA16]) summarizes the state of the art.

Work	**Unbounded Samples (w)**	**Modulus (q)**	**Advantage Change ($\varepsilon \to \varepsilon'$)**
[BPR12]	Yes	$Bp\kappa^{\omega(1)}$	$\varepsilon - \mathtt{negl}(\kappa)$
[AKPW13]	No	$\gamma Bwp\kappa$	$\varepsilon/(2dw)$
[BGM$^+$16]	No	Bwp	$(\varepsilon/qw)^2$
[BLL$^+$15]	No	Bwp	$(\varepsilon/qw)^2$
[AA16] (1)	No	$Bwp\kappa$	$\varepsilon(wB)^{-c}$
[AA16] (2)	No	$Bwp\kappa$	$\varepsilon(w)^{-c}$

Work	**Dimension Change ($d \to d'$)**	**Straightforward Rounding**	**Uniform Samples**
[BPR12]	d	Yes	Yes
[AKPW13]	$d \log(\gamma)/\log q$	Yes	Yes
[BGM$^+$16]	$d/log_\rho q$	Yes	Yes
[BLL$^+$15]	d	Yes	Yes
[AA16] (1)	d	Yes	Yes
[AA16] (2)	$d - c$	Yes	Yes

[3]for typical choices of parameters

A First Attempt. When attempting to reduce LWE to LWR, it is natural to ask why we cannot just throw out or ignore the samples that are close to these rounding 'boundaries'. We could use probabilistic rejection sampling to make our LWR sample distribution mimic the output of an LWE oracle. This works in essentially the following way (we are leaving out some details here, and what we say isn't exactly correct, but the intuition remains true): recall that a regular LWR oracle would simply just output samples of the form $\left(\mathbf{a}_i, \lceil \mathbf{a}_i{}^\intercal \mathbf{s} \rfloor_p\right)$. Now suppose we have a new oracle that outputs samples in a somewhat similar manner, but with a catch: for every sample, it samples some δ_i from the LWE noise distribution ψ. If $\mathbf{a}_i{}^\intercal \mathbf{s} + \delta_i$ is within a distance of B from the rounding boundary, then we reject the sample and start over. Otherwise, we go on ahead and output $\left(\mathbf{a}_i, \lceil \mathbf{a}_i{}^\intercal \mathbf{s} \rfloor_p\right)$ as usual.

It shouldn't take too much effort to see that the behavior of this new LWR oracle is exactly the same as the following oracle: Take samples from an LWE oracle, and round the second term, getting a term of the form $\left(\mathbf{a}_i, \lceil \mathbf{a}_i{}^\intercal \mathbf{s} + \delta_i \rfloor_p\right)$, but reject the samples if $\mathbf{a}_i{}^\intercal \mathbf{s} + \delta_i$ is within B of a rounding boundary. This oracle is directly simulatable given an LWE oracle, so we can prove hardness directly from LWE using a reduction like this.

Unfortunately, this approach still has one glaring hole: this type of rounding with probabilistic rejection is not fully deterministic. It is true that, for a fixed instance of the problem, every tuple whose first term is $\mathbf{a}_i$ will always have second term $\lceil \mathbf{a}_i{}^\intercal \mathbf{s} \rfloor_p$. However, which $\mathbf{a}_i$s are in fact included in samples is highly nonuniform–and even probabilistic–and even though it is very simple to show that the accumulated distribution of $\mathbf{a}_i$s is computationally hard to distinguish from fully random for one instance of the problem, this still presents significant problems for many applications that require determinism. For instance, if we attempt to build a PRF using this technique, different instantiations of the PRF may have different outputs, since, depending on the noise sampled, some instantiations of the PRF will reject certain $\mathbf{a}_i$s while others will accept them. This could be disastrous for security if an adversary has access to multiple copies of the PRF, since we can guess how close a particular value of $\mathbf{a}_i{}^\intercal \mathbf{s}$ is to the boundary based on the number of outputs versus rejections for a particular sample.

It seems difficult to improve this basic attempt if we wanted to keep the number of queries to be an unbounded polynomial, since we cannot know any extra information about $\mathbf{a}_i{}^\intercal \mathbf{s}$ in the simulation, and we could not leak any non-negligible amount of information per query to help us without risking a complete reveal of the secret $\mathbf{s}$.

Lattice Rounding. Our first core idea is the following: what if, instead of rounding each sample to the nearest multiple of some integer, we group samples together and deterministically round them to a nearby lattice point (not necessarily the nearest, obviously)? We might be able to leak more information that can help us eliminate rounding mistakes this way, as it could potentially be hard for an adversary to determine whether or not a point is close to a boundary created by whatever lattice rounding algorithm we are using (think of Babai's nearest plane algorithm or regular Babai rounding)[4].

However, we have to be careful when doing this. A simulator or adversary cannot have access to both the values of uniformly random $\mathbf{a}_i{}^\intercal \mathbf{s}$ terms and the lattice points gained as a result of rounding, or otherwise something similar to the famous 'learning a hidden parallelepiped' attack in [NR06] seems to occur and the (short) basis of the lattice can be learned. Once a simulator (or an adversary for that matter) has access to the basis we are using to round, then we gain very little from rounding to a lattice instead of to the nearest multiple of some integer. Perhaps some clever trick could thwart this class of attacks, but we could not come up with such an idea.

[4]Proving that an adversary cannot distinguish points close to the 'rounding boundary' of a lattice from uniformly random points without breaking some form of LWE seems like a very interesting open problem.

Restricting the Samples The next main idea is something that arises naturally from our first attempt: suppose we continue to utilize 'lattice rounding', but what if we restrict the possible values of the samples $\mathbf{A}_i$ such that the values $\mathbf{A}_i\mathbf{s}$ are in some kind of ball around points in the lattice that we are rounding to rather than let them be uniform? Note that our samples consist of matrices now instead of vectors. We can use the LWE assumption to show that such a distribution looks random to an adversary, and we can simulate such samples without a trapdoor for the lattice we are rounding to as well: pick a random point in the lattice to which we will 'round' the final value $\mathbf{A}_i\mathbf{s}$ (given a 'bad' basis, of course), add noise to get the 'actual' value of $\mathbf{A}_i\mathbf{s}$, and then output the value of $\mathbf{A}_i$ such that $\mathbf{A}_i\mathbf{s}$ equals the lattice point we picked plus the noise term.

This has the unfortunate consequence that we cannot use a uniform distribution of $\mathbf{A}_i$s, but allows us to potentially prove security because we do not need to know a short basis of the lattice we are rounding to in order to simulate queries. However, we are still unable to issue a challenge query in this regime–in order to find a point close to the lattice, our sampler implicitly needs to know the lattice point.

Finding an Acceptable Output Our solution to the previous problem is the following: rather than ouput samples $\mathbf{A}_i$ and the closest lattice point to $\mathbf{A}_i\mathbf{s}$, what if we only output some limited information about that lattice point, rather than the whole thing? It turns out that if we write the 'nearest' lattice point in $\Lambda(\mathbf{B})$ in the form $\mathbf{Bu}$ (according to some rounding algorithm $\mathcal{A}$–we may not actually output the closest lattice point) and then output $\mathbf{u} \mod 2$, we can actually prove security. We develop a variant of the LWE assumption that reduces from the standard LWE assumption that allows us to do this while maintaining the properties of unbounded (polynomial) samples and a polynomially-sized modulus.

Informally, our assumption says that, for certain $\mathbf{A}_i$s picked so that $\mathbf{A}_i\mathbf{s}$ is close to $\Lambda(\mathbf{B})$, samples of the form

$$\left(\mathbf{A}_i, \lceil \mathbf{A}_i\mathbf{s} \rfloor^{\mathcal{A}}_{\Lambda(\mathbf{B})} \mod 2\right)$$

are indistinguishable from random. Note that our process is parameterized by a lattice $\Lambda(\mathbf{B})$ and a rounding algorithm $\mathcal{A}$. We are obviously glossing over some important details here, but this is the basic idea (and the details are in the body of the paper).

2 Preliminaries

In this section we present some basic material common to many cryptographic papers. A reader familiar with general cryptography and particularly lattices and lattice cryptography can probably safely skip this section. We borrow elements of the presentation from [BGG+14] and [GPV08].

2.1 General Notation

For a random variable X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution of X. If S is a finite set instead of a distribution, we denote by $x \leftarrow S$ the process of sampling a value x according to the uniform distribution over S.

A non-negative function $v(\lambda)$ is *negligible* if for every polynomial $p(\lambda)$ it holds that $v(\lambda) \leq \frac{1}{p(\lambda)}$ for all sufficiently large $\lambda \in \mathbb{N}$.

Statistical Distance: Let Ω be a finite domain, and let X and Y be random variables over Ω. We define the statistical distance between X and Y, denoted $SD(X, Y)$ in the following way:

$$SD(X, Y) = \frac{1}{2} \sum_{\omega \in \Omega} |\Pr[X = \omega] - \Pr[Y = \omega]|$$

We say that two variables X and Y are δ-close if $SD(X, Y) \leq \delta$. If we parameterize X and Y with the security parameter $\lambda \in N$, we can say that two families of distributions X_λ and Y_λ are *statistically indistinguishable* or *statistically close* if $SD(X_\lambda, Y_\lambda)$ is negligible in λ.

Rounding. For an integer $p \leq q$, we define the modular "rounding" function

$$\lceil \cdot \rfloor_p : \mathbb{Z}_q \to \mathbb{Z}_p \text{ that maps } x \to \lfloor (p/q) \cdot x \rceil$$

and extend it coordinate-wise to matrices and vectors over $\mathbb{Z}_q$.

B-Bounded. Let $B = B(n) \in \mathbb{N}$ be a natural number. A family of distributions $\psi \in \{\psi_n\}_{n \in \mathbb{N}}$ is called B-bounded if $\Pr[\psi \in [-B, B]] = 1 - \mathtt{negl}(n)$.

2.2 Lattice Background

Lattice Notation: let q, n, and m be positive integers, and let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be a matrix. We let $\Lambda_q^\perp(\mathbf{A})$ denote the lattice spanned by all $\mathbf{x} \in \mathbb{Z}_q^m$ such that $\mathbf{A} \cdot \mathbf{x} = \mathbf{0} \mod q$. For a vector $\mathbf{u} \in \mathbb{Z}_q^n$, we generalize this and let $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ denote the set of all vectors such that $\mathbf{A} \cdot \mathbf{x} = \mathbf{u}$. Note that this is a coset of $\Lambda_q^\perp(\mathbf{A})$.

Dual Lattice: For any n-dimensional lattice Λ, the *dual lattice*, which we denote Λ^*, is defined to be the lattice spanned by all vectors $\mathbf{x} \in \mathbb{R}^n$ such that, for all $\mathbf{v} \in \Lambda$, $\mathbf{x}^\intercal \cdot \mathbf{v} \in \mathbb{Z}$.

Gram-Schmidt Orthogonalization: let $\mathbf{S}$ be an ordered set of n linearly independent vectors in $\mathbb{R}^n$ such that $S = \{\mathbf{s}_1, ..., \mathbf{s}_n\}$. We let $\tilde{\mathbf{S}} = \{\tilde{\mathbf{s}_1}, ..., \tilde{\mathbf{s}_n}\}$ denote the *Gram-Schmidt Orthogonalization* of $\mathbf{S}$. We define the Gram-Schmidt Orthogonalization in the following way: first, set $\tilde{\mathbf{s}_1} = \mathbf{s}_1$. Then, for each $i \in \{2, n\}$ set each $\tilde{\mathbf{s}_i}$ equal to the component of $\mathbf{s}_i$ orthogonal to the plane spanned by $\{\mathbf{s}_1, ..., \mathbf{s}_{i-1}\}$.

For a lattice Λ, we define the *Gram-Schmidt minimum*, which we denote $\tilde{bl}(\Lambda)$, as

$$\tilde{bl} = \min_{\mathbf{B}} ||\tilde{\mathbf{B}}|| = \min_{\mathbf{B}} \left[\max_{\mathbf{b}_i} ||\tilde{\mathbf{b}_i}|| \right]$$

where the minimum is taken over all (ordered) bases $\mathbf{B}$ of Λ.

Discrete Gaussians. We borrow the elegant presentation style of [GPV08]. For any $s > 0$ define the Gaussian function on $\mathbb{R}^n$ centered at $\mathbf{c}$ with parameter $\mathbf{s}$:

$$\forall x \in \mathbb{R}^n, \quad \rho_{s,\mathbf{c}}(\mathbf{x}) = e^{-\pi ||\mathbf{x} - \mathbf{c}||^2 / s^2}$$

We sometimes omit the subscripts s and c in the case that they are 1 and 0, respectively.

For any $\mathbf{c} \in \mathbb{R}^n$, real $s > 0$, and n-dimensional lattice Λ, we define the discrete Gaussian distribution over Λ as:

$$\forall \mathbf{x} \in \Lambda, \quad \mathcal{D}_{\Lambda, \mathbf{s}, \mathbf{c}}(\mathbf{x}) = \frac{\rho_{\mathbf{s},\mathbf{c}}(\mathbf{x})}{\rho_{\mathbf{s},\mathbf{c}}(\Lambda)}$$

Smoothing Parameter. In [MR04], Micciancio and Regev defined the smoothing parameter: for any n-dimensional lattice Λ and positive real ε, the *smoothing paramter* $\eta_\varepsilon(\Lambda)$ is the smallest real $s > 0$ such that $\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \varepsilon$.

The Gadget Matrix. In [MP12], Micciancio and Peikert invented the gadget vector $\mathbf{g}$ and the gadget matrix $\mathbf{G}$. Let $q = 2^c$ be some integer, and let $c \in \mathbb{Z} = \lceil \log_2 q \rceil$. Recall from [MP12] that the vector $\mathbf{g} \in \mathbb{Z}_q^c$ is defined to be a primitive vector such that $\mathbf{g}_i = 2^{i-1}$, or $\mathbf{g}^\intercal = \left[1|2|4|....|2^{c-1}\right]$. Additionally recall the matrix $\mathbf{S} \in \mathbb{Z}_q^{c \times c}$ was defined such that entries of the form $\mathbf{S}_{i,i} = 2$, $\mathbf{S}_{i+1,i} = -1$, and all other entries are zero. In picture form, we have

$$\begin{bmatrix} 1 & 2 & 4 & .... & 2^{c-1} \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 & 0 & ... & 0 \\ -1 & 2 & 0 & ... & 0 \\ 0 & -1 & 2 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & .... & 0 \end{bmatrix} \mod q$$

If q is not a power of 2, we can modify the last column to be the binary bit decomposition of q. We also will need the standard tensor forms of $\mathbf{g}$ and $\mathbf{S}$ as well. We define the matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times nc}$ to be $\mathbf{G} = g^\intercal \otimes \mathbf{I}_n$ and the matrix $\mathbf{T} \in \mathbb{Z}_q^{nc \times nc}$ to be $\mathbf{T} = \mathbf{S} \otimes \mathbf{I}_n$. Pictorially, we have something like the following (dimensions not to scale):

$$\begin{bmatrix} -\mathbf{g}^\intercal- & 0 & 0 & \dots & 0 \\ 0 & -\mathbf{g}^\intercal- & 0 & \dots & 0 \\ 0 & 0 & -\mathbf{g}^\intercal- & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & -\mathbf{g}^\intercal- \end{bmatrix} \cdot \begin{bmatrix} \mathbf{S} & 0 & 0 & \dots & 0 \\ 0 & \mathbf{S} & 0 & \dots & 0 \\ 0 & 0 & \mathbf{S} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \mathbf{S} \end{bmatrix} = \mathbf{0}^{nc \times c} \mod q$$

We need yet another function from [MP12]. Let the function $g_{\mathbf{G}}(\mathbf{s}, \mathbf{e}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^m \to \mathbb{Z}_q^m$ which is parameterized by some matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be defined as follows:

$$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{G}^\intercal \mathbf{s} + \mathbf{e}$$

Peikert and Micciancio also prove the following:

Lemma 2.1. *Let $\mathbf{s} \in \mathbb{Z}_q^n$ be arbitrary and let $\mathbf{e} \in \mathcal{P}_{1/2}\left(q^{-1}\mathbf{B}\right)$, where $\mathbf{B}$ is a basis of $\Lambda^\perp(\mathbf{G})$. There is an oracle $\mathcal{O}$ for efficiently finding $\mathbf{s}$ and $\mathbf{e}$ given the output of $g_{\mathbf{G}}(\mathbf{s}, \mathbf{e})$. For arbitrary modulus, $||B|| \leq \sqrt{5}$.*

2.3 LWE and Related Assumptions

Definition 2.2. ***Learning with Errors Problem (LWE)****: Consider integers n and q, some distribution ψ over $\mathbb{Z}_q$, and distributions $\mathcal{K}$ and $\mathcal{T}$, both over $\mathbb{Z}_q^n$.*

A $(q, n, \psi, \mathcal{K}, \mathcal{T})$-LWE problem instance consists of access to an unspecified challenge oracle $\mathcal{O}^{LWE}$, being, either, a noisy pseudorandom sampler $\mathcal{O}_{\mathbf{s}}^{LWE}$ carrying some constant random secret key $\mathbf{s} \in \mathbb{Z}_q^n$ sampled from the distribution $\mathcal{K}$, or, a truly random sampler $\mathcal{O}_{\$}^{LWE}$, whose behaviors are respectively as follows:

$\mathcal{O}_{\mathbf{s}}^{LWE}$: *Outputs samples of the form $\left(\mathbf{a}_i, \mathbf{a}_i \cdot \mathbf{s} + \delta_i\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\mathbf{s} \in \mathbb{Z}_q^n$ is a persistent value invariant across invocations sampled by querying the distribution $\mathcal{K}$, $\delta_i \in \mathbb{Z}_q$ consists of a fresh sample from ψ, and $\mathbf{a}_i \in \mathbb{Z}_q^n$ is sampled at random from $\mathcal{T}$.*

$\mathcal{O}_{\$}^{LWE}$: *Outputs samples of the form $\left(\mathbf{a}_i, \mathbf{r}_i\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\mathbf{a}_i \in \mathbb{Z}_q^n$ is sampled at random from $\mathcal{T}$ and $\mathbf{r}_i$ is a uniform random sample from $\mathbb{Z}_q$.*

The $(q, n, \psi, \mathcal{K}, \mathcal{T})$-LWE problem allows repeated queries to the challenge oracle $\mathcal{O}^{LWE}$. We say that an algorithm $\mathcal{A}$ decides the $(q, n, \psi, \mathcal{K}, \mathcal{T})$-LWE problem if

$$\mathbf{Adv}^{LWE}[\mathcal{A}] \stackrel{\mathsf{def}}{=} \left|\Pr[\mathcal{A}^{\mathcal{O}_{\mathbf{s}}^{LWE}} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_{\$}^{LWE}} = 1]\right|$$

is non-negligible for a $\mathbf{s}$ selected appropriately at random from $\mathcal{K}$.

Definition 2.3. ***Learning with Rounding Problem (LWR)****: Consider integers n, p, and q such that $q \geq p$, and distributions $\mathcal{K}$ and $\mathcal{T}$, both over $\mathbb{Z}_q^n$.*

A $(q, p, n, \mathcal{K}, \mathcal{T})$-LWR problem instance consists of access to an unspecified challenge oracle $\mathcal{O}^{LWR}$, being, either, a noisy pseudorandom sampler $\mathcal{O}_{\mathbf{s}}^{LWR}$ carrying some constant random secret key $\mathbf{s} \in \mathbb{Z}_q^n$ sampled from the distribution $\mathcal{K}$, or, a truly random sampler $\mathcal{O}_{\$}^{LWR}$, whose behaviors are respectively as follows:

$\mathcal{O}_{\mathbf{s}}^{LWR}$: *Outputs samples of the form $\left(\mathbf{a}_i, \lceil \mathbf{a}_i \cdot \mathbf{s} \rfloor_p\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$, where $\mathbf{s} \in \mathbb{Z}_q^n$ is a persistent value invariant across invocations sampled by querying the distribution $\mathcal{K}$ and $\mathbf{a}_i \in \mathbb{Z}_q^n$ is sampled at random from $\mathcal{T}$.*

$\mathcal{O}_{\$}^{LWR}$: *Outputs samples of the form $\left(\mathbf{a}_i, \mathbf{r}_i\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$, where $\mathbf{a}_i \in \mathbb{Z}_q^n$ is sampled at random from $\mathcal{T}$ and $\mathbf{r}_i$ is a uniform random sample from $\mathbb{Z}_p$.*

The $(q, p, n, \mathcal{K}, \mathcal{T})$-LWR problem allows repeated queries to the challenge oracle $\mathcal{O}^{LWR}$. We say that an algorithm $\mathcal{A}$ decides the $(q, p, n, \mathcal{K}, \mathcal{T})$-LWR problem if

$$\mathbf{Adv}^{LWR}[\mathcal{A}] \stackrel{\mathsf{def}}{=} \left|\Pr[\mathcal{A}^{\mathcal{O}_{\mathbf{s}}^{LWR}} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_{\$}^{LWR}} = 1]\right|$$

is non-negligible for a $\mathbf{s}$ selected appropriately at random from $\mathcal{K}$.

3 The Main Problem

We now are in a position to formally describe our new assumption. Before we begin with the actual problem, though, we need to formally define lattice rounding.

Lattice Rounding We are going to overload the rounding function. Let q, m, and n be integers with $m \geq n$. For some deterministic rounding algorithm $\mathcal{A}$ and lattice basis $\mathbf{B} \in \mathbb{Z}_q^{m \times n}$, we define the "rounding" function

$$\lceil \cdot \rfloor_{\Lambda(\mathbf{B})}^{\mathcal{A}} : \mathbb{Z}_q^m \to \mathbb{Z}_q^n \text{ that maps } \mathbf{x} \to \mathbf{u}$$

$$\text{such that } \mathcal{A}(\mathbf{x}) = \mathbf{B} \cdot \mathbf{u}$$

Note that, while our rounding algorithm $\mathcal{A}$ is deterministic, it doesn't necessarily have to be perfect at rounding points to close lattice points. Since finding the (exact) closest lattice point is thought to be hard even when arbitrary 'hints' about the lattice are known, we cannot expect any rounding algorithm to always find the closest lattice points. In order to give a general definition, we only require that our rounding algorithm is deterministic.

However, we do note that, in order for our proof to hold, our rounding algorithm must round points within some error factor $\mathbf{e}$ of a lattice point to the correct lattice point, but this $\mathbf{e}$ can be substantially smaller than the shortest vector in the lattice. Examples of $\mathcal{A}$ could be Babai's nearest plane algorithm or Babai rounding with a 'good' lattice basis, for instance, although we use more complicated rounding algorithms in this work.

In addition, note that we output the coefficient vector $\mathbf{u}$ instead of an actual lattice point like $\mathbf{B} \cdot \mathbf{u}$. While in principle we could output a lattice point, it will make the presentation of our results simpler if we define lattice rounding to output coefficient vectors instead of lattice points. This choice also seemingly makes lattice rounding easier to use, as most potential applications would prefer to have a uniformly random vector instead of a uniformly random lattice point as output.

3.1 Nearby Learning with Lattice Rounding Description

We next describe the details of our lattice rounding system.

Nearby Learning with Lattice Rounding

Input:
- Integers n, m, and q such that $m = O(n \log q)$
- A 'noise' distribution $\psi \in \mathbb{Z}_{2q}$
- Algorithm $\mathit{GenTrap}(1^n, 1^m, q) \to \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_{2q}^{\frac{m}{2} \times \frac{m}{2}}$ which outputs a parity check matrix $\mathbf{B} \in Z_q^{m \times n}$ and a 'trapdoor' $\mathbf{T_B}$ as described in lemma 4.5.
- Algorithm $\mathit{Invert}\left(\mathbf{B}, \mathbf{T_B}, \mathbf{b} \in \mathbb{Z}_{2q}^m\right) \to \mathbb{Z}_{2q}^n$ which outputs the vector $\mathbf{u}$ where $\mathbf{b} = \mathbf{B}\mathbf{u} + \mathbf{e}$ for some short vector $\mathbf{e} \in \mathbb{Z}_{2q}^m$ as described in lemma 4.5.

Setup:
- Set $\mathbf{B}, \mathbf{T_B} \leftarrow \mathit{GenTrap}(1^n, 1^m, q)$.
- Set $\mathbf{s} \leftarrow \mathbb{Z}_{2q}^m$.

Output a Sample:
- Set $\mathbf{u}_i \leftarrow \mathbb{Z}_{2q}^n$.
- Set $\boldsymbol{\delta}_i \in \mathbb{Z}_{2q}^m$ by concatenating m samples from ψ.
- Sample $\mathbf{A}_i \in \mathbb{Z}_{2q}^{m \times m}$ by choosing a random $\mathbf{A}_i$ that satisfies $\mathbf{A}_i\mathbf{s} = \mathbf{B}\mathbf{u}_i + \boldsymbol{\delta}_i$.
- **Output** $\left(\mathbf{A}_i, \lceil \mathbf{A}_i\mathbf{s} \rfloor_{\Lambda(\mathbf{B})}^{\mathit{Invert}(\cdot)} \mod 2\right) = (\mathbf{A}_i, \mathbf{u}_i \mod 2)$.

Reconstruct a Sample Given $\mathbf{A}_i$:
- **Output** $\left(\mathbf{A}_i, \lceil \mathbf{A}_i\mathbf{s} \rfloor_{\Lambda(\mathbf{B})}^{\mathit{Invert}(\cdot)} \mod 2\right)$

Some Comments. We note that the algorithms *GenTrap* and *Invert* closely resemble those from [MP12]. Any standard lattice trapdoor algorithm will work here, but we chose to use this one for its ease of use and efficiency. *Invert* is the function that takes the role of our abstract rounding algorithm $\mathcal{A}$, and *GenTrap* gives us the information necessary to run *Invert*.

Our modulus $2q$ is an artifact of our proof technique, so it is not clear that this modulus is absolutely necessary for the problem to be hard. Let k be any constant integer. It is also worth pointing out that the problem can be easily modified to work with modulus kq and outputs of the form $\left(\mathbf{A}_i, \lceil \mathbf{A}_i\mathbf{s}\rfloor_{\Lambda(\mathbf{B})}^{Invert(\cdot)} \mod k\right)$, as we essentially just swap the 2 factor for some other constant k. However, we stick with the $\mod 2$ case because generalizing to k doesn't give us anything too novel from a theoretical perspective and outputting more randomness with each sample still does not make the problem practical.

A natural question to ask is why we need to generate $\mathbf{B}$ with a trapdoor. For many applications of LWR (like PRFs, for instance) we need to recompute the output of a sample $\mathbf{A}_i$ given only the sample $\mathbf{A}_i$ and the key $\mathbf{s}$ (and not the actual output). For basic LWR, this is trivial, but for NLWLR, this is difficult without a trapdoor. Our algorithm 'Reconstruct a Sample Given $\mathbf{A}_i$' handles this functionality and needs to use the lattice trapdoor to work. We comment on this more in our security proof.

Additive Homomorphism. We also note that our outputs are 'almost' homomorphic– i.e., if the error is small enough, then:

$$\lceil \mathbf{A}_i\mathbf{s}\rfloor_{\Lambda(\mathbf{B})}^{Invert(\cdot)} \mod 2 + \lceil \mathbf{A}_j\mathbf{s}\rfloor_{\Lambda(\mathbf{B})}^{Invert(\cdot)} \mod 2 =$$

$$\lceil (\mathbf{A}_i + \mathbf{A}_j)\,\mathbf{s}\rfloor_{\Lambda(\mathbf{B})}^{Invert(\cdot)} \mod 2$$

This means that many standard cryptographic schemes that are built upon LWE can be built (very inefficiently) from the NLWLR assumption as well in a fairly straightforward manner. For instance, Regev-like encryption [Reg05] can be done fairly easily, and we briefly sketch how here: create a public key consisting of k samples of the form $\left(\mathbf{A}_i, \lceil \mathbf{A}_i\mathbf{s}\rfloor_{\Lambda(\mathbf{B})}^{Invert(\cdot)} \mod 2\right)$ and a secret key including $\mathbf{s}$ and $\mathbf{T_B}$.

To encrypt a bit vector $\mathbf{b} \in \mathbb{Z}_2^n$, pick a discrete Gaussian value c_i over $\mathbb{Z}$ for every sample $\mathbf{A}_i$, and output the tuple

$$\left(\sum_{i=1}^{k} c_i\mathbf{A}_i \mod 2q, \sum_{i=1}^{k} c_i \lceil \mathbf{A}_i\mathbf{s}\rfloor_{\Lambda(\mathbf{B})}^{Invert(\cdot)} + \mathbf{b} \mod 2\right)$$

Decryption is done in the natural Regev way. As long as the noise blow-up is controlled, then correctness follows. Uniformity over the choice of lattice point (i.e. $\sum_{i=1}^{k} c_i\mathbf{u}_i$) can be shown from the leftover hash lemma [BDK+11], and the leftover hash lemma over the integers [AGHS13] can be used to show that the noise term (i.e. $\sum_{i=1}^{k} c_i\boldsymbol{\delta}_i$) is distributed as a proper discrete ellipsoid, meaning that we end up with a well-formed, properly chosen sample $\mathbf{A}_i$ and output as our sum. This will require an extremely large k to be used but we already said that this would be inefficient.

While we do not want to spend too much time going through inefficient cryptosystems (thus why we do not offer a formal treatment of the previous argument), we just want to illustrate that this new NLWLR assumption can, in fact, be used to build reasonably powerful cryptosystems.

Parameter Choices. We next put forth a set of parameter choices that offers easy instantiation and good theoretical hardness. Let m, n, and q be integers such that $m \geq 4n\log q$. Let the distribution $\mathcal{D}_{\mathbb{Z},\sigma}$ be a discrete Gaussian distribution with $\sigma \leq \frac{q}{5m\log n}$ (so that the overall distribution over $\mathbb{Z}^m$ is a discrete Gaussian $\mathcal{D}_{\mathbb{Z}^m,\sigma'}$ with parameter $\sigma' \leq \frac{q}{5\sqrt{m}\log n}$).

Existence of Efficient *GenTrap* and *Invert*. In lemmas 4.5 and 4.8 we prove that the algorithms *GenTrap* and *Invert* exist as defined and that *Invert* is deterministic. We prove in lemma 4.10, that, for the choice of parameters mentioned above, the output of *GenTrap* is uniform and that the *Invert* algorithm is correct with all but negligible probability.

Reduction from Standard LWE. Also in lemma 4.10 we show that our construction is at least as hard as standard LWE (uniform samples, uniform key) in dimension n over modulus q and with discrete Gaussian noise with parameter $\frac{\sigma}{3}$ for the parameters described above. If Q is the number of samples an adversary gets, we do lose a factor of Q in our security reduction, but this doesn't make an enormous difference theoretically. This means that, through the LWE reduction of [BLP+13], we have a polynomially-approximate reduction from worst-case lattice problems while still allowing an unbounded (polynomial) number of queries.

We note that our reduction allows for substantially more general parameters than the ones mentioned in this section.

4 Security Proof

In this section we prove the security of what we have called *Nearby Learning with Lattice Rounding*. Our security proof is actually rather straightforward, but has a couple of twists that can make it difficult to follow, so we offer a proof outline below to make it easier to digest. We start by going over an uncommon way to look at the standard LWE problem.

4.1 Reverse LWE

Our proof will become much easier to understand when viewed through lens of what we call 'reverse LWE'. Traditionally, LWE is thought of as the following procedure: given a fixed key $\mathbf{s}$, sample a random sample $\mathbf{a}_i$ and a noise sample δ_i and output $(\mathbf{a}_i, {\mathbf{a}_i}^\intercal \mathbf{s} + \delta_i)$.

However, we can also generate LWE samples in the following way: suppose we are given a fixed key $\mathbf{s}$. First, pick some random value u over the output domain. Then, sample a value δ_i from the noise distribution, and choose a random $\mathbf{a}_i$ subject to the constraint that ${\mathbf{a}_i}^\intercal \mathbf{s} = u - \delta_i$. It should be clear that this distribution is statistically close to that of the traditional LWE distribution, but what this variant of LWE gives us the power to do is to choose our outputs to be distributed in a non-uniform way if we like. Picking us (or, more precisely, vectors of us) from a distribution that is computationally indistinguishable from uniform rather than one that is statistically close to uniform is something that we implicitly exploit in our reduction.

4.2 Proof Outline

The core portion of our security proof has two main steps: first, we show that a particular nonuniform variant of LWE is hard. Then we show that anyone that can distinguish outputs of our nearby learning with lattice rounding oracle from random can break this nonuniform LWE variant. Once we have shown that our output is indistinguishable from random, we show that we can make the rounding mechanism work by appropriately sampling a random lattice and a trapdoor that are compatible with our scheme.

A New Nonuniform LWE. Our reduction starts by showing that the following is a hard (nonuniform) instance of LWE: samples of the form $(\mathbf{a}_i, {\mathbf{a}_i}^\intercal (2\mathbf{s}) + \delta_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_{2q}$, where the operations are done modulo $\mathbb{Z}_{2q}$ but the $\mathbf{a}_i$s and $\mathbf{s}$ are sampled uniformly modulo q. The proof goes approximately as follows: we start with a typical LWE instance over $\mathbb{Z}_q$. Imagine we have samples of the form $(\mathbf{a}_i, {\mathbf{a}_i}^\intercal \mathbf{s} + \delta_i)$. Suppose we multiply the second term by two and 'lift' everything up to $\mathbb{Z}_{2q}$: we now have samples of the form $(\mathbf{a}_i, {\mathbf{a}_i}^\intercal (2\mathbf{s}) + 2\delta_i)$. Note that the second term of the tuple is, by the LWE assumption, indistinguishable over the higher-order bits (the lowest-order bit will be zero since the modulus is a multiple of two). We can then add in additional noise to make the $2\mathbf{s} + 2\delta$ term look uniform since the only nonuniformity is in the lowest-order bit.

This gives us LWE samples of the form $(\mathbf{a}_i, {\mathbf{a}_i}^\intercal (2\mathbf{s}) + \delta_i)$, where both $\mathbf{a}_i$ and $\mathbf{s}$ are uniform over $\mathbb{Z}_q^n$ rather than $\mathbb{Z}_{2q}^n$ that are still indistinguishable from random (for any attentive readers, it is straightforward to show that the 'random' part of the LWE reduction holds as well).

Moving to Rounding. We now have an LWE problem where we can manipulate the lowest-order bits of the key. Given samples of the form $(\mathbf{a}_i, {\mathbf{a}_i}^\intercal (2\mathbf{s}) + \delta_i)$, we can sample a term $\mathbf{t} \in \mathbb{Z}_2^n$ and add in ${\mathbf{a}_i}^\intercal \mathbf{t}$ to the second term of the tuple, which gives us a uniform key over $\mathbb{Z}_{2q}^n$ (but we have knowledge of the lowest-order bits of the key!).

Suppose we now want to sample a single query from our rounding oracle. We can take m LWE samples in the above form and concatenate them as rows into a matrix which we will conveniently call $\mathbf{B} \in \mathbb{Z}_q^n$. We have something of the form $(\mathbf{B}, \mathbf{B}\mathbf{s} + \boldsymbol{\delta})$ where we know the lowest-order bits of $\mathbf{s}$ (in other words, $\mathbf{s} \mod 2$).

Given a random vector $\mathbf{t} \in \mathbb{Z}_{2q}^m$, we can set a matrix $\mathbf{A}_i \in \mathbb{Z}_{2q}^m$ such that $\mathbf{A}_i\mathbf{t} = \mathbf{B}\mathbf{s} - \boldsymbol{\delta}$. Thus we can output a sample $(\mathbf{A}_i, \mathbf{s} \mod 2)$ and have it be a valid output for our nearby learning with lattice rounding oracle since $\mathbf{B}\mathbf{s}$ is the closest point in $\Lambda(\mathbf{B})$ to $\mathbf{A}_i\mathbf{t}$. It is also straightforward to show that $\mathbf{A}_i$ is distributed uniformly at random (and independent of $\mathbf{s} \mod 2$) if the original LWE samples were random.

Unfortunately, given that the secret in our reduction is a close lattice point (which only gives us one query) rather than many queries (as a LWE secret might), we are forced to use a hybrid argument over all of the adversary's queries Q. This gives us a $\frac{1}{Q}$-security loss in our reduction, but we still can output an unbounded number of queries because Q must be polynomial at the end of the day. We defer the details to the formal proof.

Actually Rounding. While the above argument explains why our output is indistinguishable from random, we would like to be able to actually round as well (for a valid $\mathbf{A}_i$, any entity with the requisite secrets $\mathbf{s}$ and $\mathbf{T}_\mathbf{B}$ should be able to produce a valid output) instead of just simulating rounded output. This is essential for sort of any application where multiple parties need to statelessly produce the same outputs given repeated samples $\mathbf{A}_i$ as input[5]. The most basic way to do this would be to sample our lattice basis $\mathbf{B}$ with a trapdoor. Recall that, in [MP12] the authors provide algorithms for sampling a random-looking parity check matrix $\mathbf{B} \in \mathbb{Z}_q^{m\times n}$ and a corresponding trapdoor $\mathbf{T}_B$ such that, given the trapdoor $\mathbf{T}_B$ and a sample of the form $\mathbf{B}\mathbf{u} + \boldsymbol{\delta} \in \mathbb{Z}_q^m$ (for some $\mathbf{u} \leftarrow \mathbb{Z}_q^n$ and some noise vector $\boldsymbol{\delta} \in \mathbb{Z}_q^m$) it is possible to efficiently determine $\mathbf{u}$ with high probability.

Unfortunately, we want an instance of this algorithm where $\mathbf{B}$ still lives in $\mathbb{Z}_q$ but the output samples and the secret $\mathbf{u}$ live in $\mathbb{Z}_{2q}$. As an astute reader might notice, this is slightly annoying to go through all of the details, but conceptually simple. To illustrate, suppose we have a collection of short vectors $\mathbf{x}$ such that $\mathbf{B}\mathbf{x} = \mathbf{0} \mod q$, giving us a form of trapdoor for $\mathbf{B}$. Then $\mathbf{B}(2\mathbf{x}) = \mathbf{0} \mod 2q$, which gives us a slightly worse form of trapdoor. In sum, getting this trapdoor sampler to work for a different modulus of output samples is fairly straightforward, if a bit tedious. We slightly modify the algorithms from [MP12] in order to do this easily and efficiently.

4.3 A New Nonuniform LWE

Our first lemma may seem a bit confusing, but it is really quite simple. As we described in the proof outline, what we are doing is essentially just modulus lifting. We are taking an

[5]like PRFs, for instance. We discuss PRFs in the conclusion.

LWE instance modulo q and raising it to an instance modulo $2q$. The LWE samples remain in $\mathbb{Z}_q$, however, and the key moves to $2 \cdot \mathbb{Z}_q \in \mathbb{Z}_{2q}$.

A brief sketch: this works for the following reason: if $\mathbf{a}_i{}^{\intercal}\mathbf{s} + \delta_i = c \mod q$, then $\mathbf{a}_i{}^{\intercal}(2\mathbf{s}) + 2\delta_i = 2c \mod 2q$. Moreover, it should be immediately clear that the 'doubled' LWE instance is random in all but the lowest-order bit (in which it will always be zero due to the multiple of 2 in the terms and the fact that the modulus is $2q$). Adding some low-norm noise that is uniform over $\mathbb{Z}_2$ rectifies this and gives us the LWE instance we desire.

Lemma 4.1. *Consider integers n and q, and some noise distribution ψ over $\mathbb{Z}_q$. Let the distribution $\mathcal{T}$ be over $\mathbb{Z}_q^n$. Let $\mathcal{U}_q^n$ and $\mathcal{U}_{2q}^n$ denote the uniform distribution over their respective domains, and let $2 \cdot \mathcal{U}_q^n$ be the distribution over $\mathbb{Z}_{2q}^n$ defined in the natural way. Let ψ' be some distribution over $\mathbb{Z}_{2q}^n$ that is statistically close to uniform over $\mathbb{Z}_2$.*

Any adversary that can solve the $\big(2q, n, 2\psi + \psi', 2 \cdot \mathcal{U}_q^n, \mathcal{T}\big)$-LWE problem with advantage ε can be used to solve the $\big(q, n, \psi, \mathcal{U}_q^n, \mathcal{T}\big)$-LWE problem with advantage ε.

Proof. Given an oracle which we denote $\mathcal{O}^{1-LWE}$ for the $\big(q, n, \psi, \mathcal{U}_q^n, \mathcal{T}\big)$-LWE problem, we must show how to simulate an oracle for the $\big(2q, n, 2\psi + \psi', 2 \cdot \mathcal{U}_q^n, \mathcal{T}\big)$-LWE problem, which we denote $\mathcal{O}^{2-LWE}$.

To do this, we show how to convert each output from a $\mathcal{O}^{1-LWE}$ oracle into a valid output from a $\mathcal{O}^{2-LWE}$ oracle. For a vector $\mathbf{a}_i \leftarrow \mathcal{T}$ and a fixed key $\mathbf{s} \in \mathbb{Z}_q^n$ and integers $r_i \leftarrow \mathbb{Z}_q$ and $\delta_i \leftarrow \psi$, recall that outputs from $\mathcal{O}^{1-LWE}$ have one of the two following forms:

$$(\mathbf{a}_i, \mathbf{a}_i{}^{\intercal} \cdot \mathbf{s} + \delta_i), \quad (\mathbf{a}_i, r_i)$$

Suppose we take our output and multiply everything in the second term by two. We note that, at this point, we assume the second term in each tuple lives in $\mathbb{Z}_{2q}^n$ rather than $\mathbb{Z}_q^n$. This gives us the following set of possibilities:

$$(\mathbf{a}_i, \mathbf{a}_i{}^{\intercal} \cdot (2\mathbf{s}) + 2\delta_i), \quad (\mathbf{a}_i, 2r_i)$$

Essentially we are just 'lifting' the second term in each tuple to the high-order bits of a new modulus $2q$. Note that the low-order bit of the second term in the tuples will always be zero at this point. To deal with this, suppose we add in a sample from χ' to the second term of our sample. Let $\delta_i' \leftarrow \chi'$. We get:

$$(\mathbf{a}_i, \mathbf{a}_i{}^{\intercal} \cdot (2\mathbf{s}) + 2\delta_i + \delta_i'), \quad (\mathbf{a}_i, 2r_i + \delta_i')$$

We claim that this is the proper distribution of a $\mathcal{O}^{2-LWE}$ oracle. First, note that $\mathbf{a}_i$ is a valid sample from $\mathcal{T}$, and $2\delta_i + \delta_i'$ is a valid sample from $2\psi + \psi'$. Note as well that $2\mathbf{s}$ is uniformly distributed over the high-order bits $\mathbb{Z}_{2q}^n$ and zero in the lowest-order bit, meaning our LWE secret has the correct distribution. Finally, since δ_i' is statistically close to uniform over $\mathbb{Z}_2$ and $2r_i$ is uniform in all but the lowest bit, we know that $2r_i + \delta_i'$ is statistically close to uniform over $\mathbb{Z}_{2q}$.

Since we have faithfully simulated a $\mathcal{O}^{2-LWE}$ oracle, we can then feed the queries to our adversary and then forward the adversary's response to the challenger, preserving the advantage ε of the adversary in the reduction. ∎

4.4 Reducing Rounding Output to LWE

We now formally define the Nearby Learning with Lattice Rounding Assumption. We note that the output of this distribution (in the 'real case') is identical to that of the output described in section 3.1, as required by definition.

Definition 4.2. ***Nearby Learning with Lattice Rounding (NLWLR):*** *Consider integers m, n, and q, some B-bounded distribution ψ over $\mathbb{Z}_{2q}$, and a distribution $\mathcal{T}$ over $\mathbb{Z}_q^n$.*

Let the matrix $\mathbf{B} \in \mathbb{Z}_q^{m\times n}$ be distributed such that each row is a fresh sample from $\mathcal{T}$. We let $\Lambda(\mathbf{B})$ denote the $2q$-ary lattice with basis $\mathbf{B}$[6].

A $(q, n, \psi, \mathcal{K}, \mathbf{B})$-NLWLR problem instance consists of access to the public matrix $\mathbf{B}$ and an unspecified challenge oracle $\mathcal{O}^{nLWE}$, being, either, a noisy pseudorandom sampler $\mathcal{O}_{\mathbf{s}}^{NLWLR}$ carrying some constant random secret key $\mathbf{s} \in \mathbb{Z}_{2q}^n$ sampled from the distribution $\mathcal{K}$, or, a truly random sampler $\mathcal{O}_{\$}^{NLWLR}$, whose behaviors are respectively as follows:

$\mathcal{O}_{\mathbf{s}}^{NLWLR}$**:** *Outputs samples of the form $(\mathbf{A}_i, \mathbf{z}_i) \in \mathbb{Z}_{2q}^{m\times m} \times \mathbb{Z}_2^m$, where the terms are sampled in the following way: first, recall that $\mathbf{s} \in \mathbb{Z}_q^n$ is a persistent value invariant across invocations sampled by querying the distribution $\mathcal{K}$. Let the value $\mathbf{u}_i$ be sampled randomly from $\mathcal{U}_{2q}^n$, and let $\boldsymbol{\delta}_i \in \mathbb{Z}_{2q}^m$ be sampled by selecting m independent fresh samples from ψ and concatenating them together to form a vector. To output $\mathbf{A}_i$, we use Gaussian elimination to solve for the matrix $\mathbf{A}_i$ that satisifies the following equation:*

$$\mathbf{A}_i \cdot \mathbf{s} = \mathbf{B} \cdot \mathbf{u}_i + \boldsymbol{\delta}_i$$

We then set $\mathbf{z}_i = \mathbf{u}_i \mod 2$ and output that as well.

$\mathcal{O}_{\$}^{NLWLR}$**:** *Outputs samples of the form $(\mathbf{A}_i, \mathbf{z}_i) \in \mathbb{Z}_{2q}^{m\times m} \times \mathbb{Z}_2^m$, where $\mathbf{A}_i \in \mathbb{Z}_q^{m\times n}$ is sampled uniformly at random from $\mathbb{Z}_{2q}^{m\times m}$ and $\mathbf{z}_i$ is a randomly sampled vector in $\mathbb{Z}_2^n$.*

The $(q, n, \psi, \mathcal{K}, \mathbf{B})$-NLWLR problem allows repeated queries to the challenge oracle $\mathcal{O}^{NLWLR}$. We say that an algorithm $\mathcal{A}$ decides the $(q, n, \psi, \mathcal{K}, \mathbf{B})$-NLWLR problem if

$$\mathbf{Adv}^{NLWLR}[\mathcal{A}] \overset{\mathsf{def}}{=} \left|\Pr[\mathcal{A}^{\mathcal{O}_{\mathbf{s}}^{NLWLR}} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_{\$}^{NLWLR}} = 1]\right|$$

is non-negligible for a $\mathbf{s}$ selected appropriately at random from $\mathcal{K}$.

Lemma 4.3. *Consider integers m, n, and q, some B-bounded distribution ψ over $\mathbb{Z}_{2q}$, and a distribution $\mathcal{T}$ over $\mathbb{Z}_q^n$.*

Let the matrix $\mathbf{B} \in \mathbb{Z}_q^{m\times n}$ be distributed such that each row is a fresh sample from $\mathcal{T}$. We let $\Lambda(\mathbf{B})$ denote the $2q$-ary lattice with basis $\mathbf{B}$.

Let Q be the number of queries made to an nLWE oracle. Any adversary that can solve the $(q, n, \psi, \mathcal{U}_q^n, \mathbf{B})$-NLWLR problem with advantage ε can be used to solve the $(2q, n, \psi, 2\cdot\mathcal{U}_q^n, \mathcal{T})$-LWE problem with advantage at least $\frac{1}{Q}\varepsilon$.

Proof. Our proof goal is standard and simple: given an LWE oracle $\mathcal{O}^{LWE}$ that is either $\mathcal{O}_{\mathbf{s}}^{LWE}$ or $\mathcal{O}_{\$}^{LWE}$, we need to simulate an NLWLR oracle $\mathcal{O}^{NLWLR}$ so that we simulate a $\mathcal{O}_{\mathbf{s}}^{NLWLR}$ oracle if we were given a $\mathcal{O}_{\mathbf{s}}^{LWE}$ oracle and a $\mathcal{O}_{\$}^{NLWLR}$ oracle if we were given a $\mathcal{O}_{\$}^{LWE}$ oracle. To do this, we will need to use a hybrid argument, so our reduction will not be exactly advantage-preserving.

Suppose an adversary is restricted to Q queries. For $i \in [0, Q]$ we define a series of hybrid experiments in the following way: let the experiment H_i denote outputting i samples from $\mathcal{O}_{\mathbf{s}}^{NLWLR}$ followed by $Q - i$ samples from $\mathcal{O}_{\$}^{NLWLR}$. As a sanity check, note that H_0 denotes output from a totally 'random' distribution, while H_Q is the 'real' distribution.

As is usual for hybrid arguments, we will show that any adversary that can distinguish the hybrids H_i and H_{i+1} with advantage ε can be used to solve the standard LWE problem with advantage ε. This in turn implies that an adversary that can solve the NLWLR problem can also solve the standard LWE problem with advantage at least $\frac{1}{Q}\varepsilon$, completing the proof.

Generating the Hybrid Challenge. To start, let's consider our oracle $\mathcal{O}^{LWE}$. It will output samples of the form

$$(\mathbf{a}_i, \mathbf{a}_i{}^{\intercal} \cdot (2\mathbf{s}) + \delta_i), \quad (\mathbf{a}_i, r_i) \qquad \in \mathbb{Z}_q^n \times \mathbb{Z}_{2q}$$

[6]We are slightly abusing notation here since B lives in a finite ring, but the intuition is correct.

where $\mathbf{a}_i \leftarrow \mathcal{T}$, $\delta_i \leftarrow \psi$, $r_i \leftarrow \mathbb{Z}_{2q}$, and $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ is fixed for all of the samples. Suppose we query our oracle $\mathcal{O}^{LWE}$ m times and concatenate the results into a matrix. Let the concatenation of the row vectors $\mathbf{a}_i$ be defined to be the matrix $\mathbf{B} \in \mathbb{Z}_q^{m\times n}$. We can write this concatenation in the following way:

$$(\mathbf{B}, \mathbf{B}\cdot(2\mathbf{s}) + \boldsymbol{\delta}), \quad (\mathbf{B}, \mathbf{r}) \qquad \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_{2q}^m$$

Let's define $\mathbf{c} \in \mathbb{Z}_{2q}^n$ to be the second term of the above tuples (which one depends on which version of $\mathcal{O}^{LWE}$ we have been given. Thus, $\mathbf{c} = \{\mathbf{B}\cdot(2\mathbf{s}) + \boldsymbol{\delta}, \mathbf{r}\}$. Next, suppose we sample a vector $\mathbf{x} \leftarrow \mathbb{Z}_2^n$ and add $\mathbf{B}\cdot\mathbf{x}$ to $\mathbf{c}$. We have:

$$\mathbf{c} = \{\mathbf{B}\cdot(2\mathbf{s}+\mathbf{x}) + \boldsymbol{\delta}, \quad \mathbf{r} + \mathbf{B}\cdot\mathbf{x}\}$$

Let $\mathbf{b} = 2\mathbf{s}+\mathbf{x} \in \mathbb{Z}_{2q}^n$. By definition, $\mathbf{b}$ is distributed uniformly at random, and $\mathbf{b} \mod 2 = \mathbf{x}$. If we rewrite this, we have:

$$\mathbf{c} = \{\mathbf{B}\cdot\mathbf{b} + \boldsymbol{\delta}, \quad \mathbf{r} + \mathbf{B}\cdot\mathbf{x}\}$$

Our goal is to make $\mathbf{c}$ into a hybrid-distinguishing query. To do that, suppose we sample some $\mathbf{t} \leftarrow \mathbb{Z}_{2q}^m$. Then we solve for some matrix $\mathbf{A}^*$ such that

$$\mathbf{A}^*\cdot\mathbf{t} = \mathbf{c}$$

Note that if $\mathbf{c} = \mathbf{B}\cdot\mathbf{b} + \boldsymbol{\delta}$, then $\mathbf{A}^*$ will have the distribution of a single sample from the $\mathcal{O}_{\mathbf{s}}^{NLWLR}$. If $\mathbf{c} = \mathbf{r} + \mathbf{B}\cdot\mathbf{x}$, then $\mathbf{A}^*$ will be distributed uniformly at random (and independently of $\mathbf{x}$ since the $\mathbf{r}$ term is uniform over $\mathbb{Z}_{2q}^m$ and independent of $\mathbf{x}$), giving the correct distribution of a single sample of $\mathcal{O}_{\$}^{NLWLR}$. Thus the sample $(\mathbf{A}^*, \mathbf{x})$ is a valid query from $\mathcal{O}^{NLWLR}$ parameterized by the lattice $\mathbf{B}$.

Simulating the Hybrids. Now, suppose that an adversary can distinguish between the hybrid experiments H_i and H_{i+1}. We show that such an adversary can be used to solve the standard LWE problem parameterized in the problem statement. To start, suppose we have run through all of the calculations in the *Generating the Hybrid Challenge* portion of the proof, as we will use that notation here. We first start by sending the adversary the matrix $\mathbf{B}$ as well as the rest of the public parameters.

For hybrid queries 1 through i, we respond in the following way: choose a random vector $\mathbf{z}_i \leftarrow \mathbb{Z}_{2q}^n$ and create a vector $\boldsymbol{\delta}_i \in \mathbb{Z}_{2q}^m$ by sampling ψ a total of m times and concatenating the result into a vector. Then we set the matrix $\mathbf{A}_i \in \mathbb{Z}_q^{m\times m}$ by solving the equation $\mathbf{A}_i\cdot\mathbf{t} = \mathbf{B}\cdot\mathbf{z}_i + \boldsymbol{\delta}_i$. Our query response becomes

$$(\mathbf{A}_i, \quad \mathbf{z}_i \mod 2) \qquad \in \mathbb{Z}_{2q}^{m\times m} \times \mathbb{Z}_2^n$$

Note that this perfectly simulates a query response for the oracle $\mathcal{O}_{\mathbf{s}}^{NLWLR}$ parameterized by the matrix $\mathbf{B}$. For the challenge query $i+1$, we respond with the query

$$(\mathbf{A}^*, \quad \mathbf{x}) \qquad \in \mathbb{Z}_{2q}^{m\times m} \times \mathbb{Z}_2^n$$

As we discussed earlier, this query will be distributed as from $\mathcal{O}_{\mathbf{s}}^{NLWLR}$ if we received the oracle $\mathcal{O}_{\mathbf{s}}^{LWE}$ from the challenger and from $\mathcal{O}_{\$}^{NLWLR}$ if we received the oracle $\mathcal{O}_{\$}^{LWE}$ from the challenger.

For queries $i+2$ through Q, we just respond with uniformly random output in $\mathbb{Z}_q^{m\times m} \times \mathbb{Z}_2^m$, which is exactly the required distribution of $\mathcal{O}_{\$}^{NLWLR}$.

Thus, in our hybrid simulation, we have simulated hybrid experiment H_i if we received the oracle $\mathcal{O}_{\$}^{LWE}$ and hybrid H_{i+1} if we received the oracle $\mathcal{O}_{\mathbf{s}}^{LWE}$. This implies that any adversary that can distinguish the hybrid experiments H_i and H_{i+1} with advantage ε can be used to solve the standard LWE problem as parameterized in the theorem statement with advantage ε, which is the fact we need to complete the proof. ∎

4.5 Generating a Trapdoor

In this section we show that there are suitable mechanisms for us to realize the *GenTrap* and *Invert* functions needed for the nearby learning with lattice rounding assumption to work. As we mentioned in the proof overview, this is really quite straightforward, but still takes a bit of digging through the details in order to build a convincing proof. Our construction of the *GenTrap* and *Invert* functions are almost identical to (and the notation is borrowed from) those of [MP12], but in theory any kind of trapdoor generation could be used.

Previous Work Our starting point will be the following theorem from [MP12] (Theorem 5.1), which allows us to sample a uniform parity check matrix and a corresponding trapdoor:

Lemma 4.4. *There is an efficient randomized algorithm* $GenTrap\left(1^n, 1^m, q\right)$ *that, given any integers* $n \geq 1$*,* $q \geq 2$*, and sufficiently large* $m = O\left(n \log q\right)$ *outputs a parity-check matrix* $\mathbf{A} \in Z_q^{n \times m}$ *and a 'trapdoor'* $\mathbf{R}$ *such that the distribution of* $\mathbf{A}$ *is* $\texttt{negl}\left(n\right)$*-close to uniform. Moreover, there are efficient algorithms Invert and SampleD that with overwhelming probability over all random choices, do the following:*

- ◇ *For* $\mathbf{b}^\intercal = \mathbf{s}^\intercal \mathbf{A} + \mathbf{e}^\intercal$*, where* $\mathbf{s} \in \mathbb{Z}_q^n$ *is arbitrary and either* $||\mathbf{e}|| < q/O\left(\sqrt{n \log q}\right)$ *or* $e \leftarrow \mathcal{D}_{\mathbb{Z}^m, \alpha q, \mathbf{0}}$ *for* $1/\alpha \geq \sqrt{n \log q} \cdot \omega\left(\sqrt{\log n}\right)$*, the deterministing algorithm* $Invert\left(\mathbf{R}, \mathbf{A}, \mathbf{b}\right)$ *outputs* $\mathbf{s}$ *and* $\mathbf{e}$*.*
- ◇ *For any* $\mathbf{u} \in \mathbb{Z}_q^n$ *and a large enough* $s = O\left(\sqrt{n \log q}\right)$*, the randomized algorithm* $SampleD\left(\mathbf{R}, \mathbf{A}, \mathbf{u}, s\right)$ *samples from a distribution that is within* $\texttt{negl}\left(n\right)$ *statistical distance of* $\mathcal{D}_{\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}), s \cdot \omega\left(\sqrt{\log n}\right)}$*.*

We really do not need the full power of this theorem for our applications, but we do need this theorem to work over nonstandard distributions. We list the necessary changes below in red. Note that we do flip the dimensions to make our main proof easier to understand (so we don't have to write transpose everywhere) but this changes nothing technically.

Lemma 4.5. *There is an efficient randomized algorithm* $GenTrap\left(1^n, 1^m, q\right)$ *that, given any integers* $n \geq 1$*,* $q \geq 2$*, and sufficiently large* $m = O\left(n \log q\right)$ *outputs a parity-check matrix* $\mathbf{A} \in Z_q^{m \times n}$ *and a 'trapdoor'* $\mathbf{R}$ *such that the distribution of* $\mathbf{A}$ *is* $\texttt{negl}\left(n\right)$*-close to uniform. Moreover, there is an efficient algorithm Invert that with overwhelming probability over all random choices, does the following:*

- ◇ *For* $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \mod 2q$*, where* $\mathbf{s} \in \mathbb{Z}_{2q}^n$ *is arbitrary and either* $||\mathbf{e}|| < q/O\left(\sqrt{n \log q}\right)$ *or* $e \leftarrow \mathcal{D}_{\mathbb{Z}^m, \alpha q, \mathbf{0}}$ *for* $1/\alpha \geq \sqrt{n \log q} \cdot \omega\left(\sqrt{\log n}\right)$*, the deterministic algorithm Invert* $\left(\mathbf{R}, \mathbf{A}, \mathbf{b}\right)$ *outputs* $\mathbf{s}$ *and* $\mathbf{e}$*.*

As an astute reader might notice, this is not that complicated. Someone familiar with lattice trapdoor generation can probably see exactly how this works, but for the sake of completeness, however, we will go through this in at least a little bit of detail. We start by presenting our *GenTrap* algorithm, which is exactly the same as that of [MP12]. Note that we present the simplified version (without an input $\overline{\mathbf{A}}$ or a tag $\mathbf{H}$) since we do not need these extra features.

Algorithm 1 Efficient algorithm $GenTrap^{\mathcal{D}}(1^n, 1^m, q)$ for generating a parity check matrix $\mathbf{A}$ with a trapdoor $\mathbf{R}$

```
Input: Integers $n$, $m$, $q$, $\overline{m}$, and $w$ such that $m = \overline{m} + w$
  • A distribution $\mathcal{D}$ over $\mathbb{Z}^{\overline{m} \times w}$.
Output: A parity check matrix $\mathbf{A} = \begin{bmatrix} \overline{\mathbf{A}} \\ \mathbf{A}_1 \end{bmatrix} \in Z_q^{m \times n}$ and trapdoor $\mathbf{R} \in \mathbb{Z}^{\overline{m} \times w}$. The steps are:
  • Sample $\overline{\mathbf{A}} \leftarrow \mathbb{Z}_q^{\overline{m} \times n}$
  • Choose a random matrix $\mathbf{R} \in \mathbb{Z}^{w \times \overline{m}}$ from the distrbution $\mathcal{D}$.
  • Set $\mathbf{A} = \begin{bmatrix} \overline{\mathbf{A}} \\ \mathbf{G}^\intercal - \mathbf{R}\overline{\mathbf{A}} \end{bmatrix}$
  • Return $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, $\mathbf{R} \in \mathbb{Z}^{w \times \overline{m}}$.
```

A simple instantiation of the above *GenTrap* algorithm is all that we need. We state the following instantiation choice which is suggested in [MP12]:

Lemma 4.6. *Let the integers m, n, q, $\overline{m}$, and w be defined such that $\overline{m} = w = 2n \log q$, and let the distribution $\mathcal{D}$ output matrices $\mathbf{R} \in \mathbb{Z}^{w \times \overline{m}}$ such that each entry $\mathbf{R}_{ij}$ is equal to one with probability $\frac{1}{4}$, -1 with probability $\frac{1}{4}$, and zero otherwise. Then the matrix $\mathbf{A}$ is distributed δ-close to uniform for some negligible δ.*

Proof. This statement follows immediately from one of the parameter discussion choices in [MP12] section 5.2. ∎

Really the only work we have to do is to change the *Invert* algorithm, and we don't have to change it that much. The key point is that if we keep the error bound by its original parameters (rather than potentially scaling it with the modulus) all of the equations from the original *Invert* algorithm from [MP12] will hold as well. That way we can first solve for $\mathbf{s}$ and $\mathbf{e}$ over the modulus q, but since $||\mathbf{e}|| < \frac{q}{2}$ absolutely, we can plug $\mathbf{e}$ back into our original equation modulo $2q$ and get the full value of $\mathbf{s}$.

Algorithm 2' Efficient algorithm $Invert(\mathbf{A}, \mathbf{R}, \mathbf{b})$ for inverting the function $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$

```
Input: An oracle $\mathcal{O}$ for inverting the function $g_{\mathbf{G}}(\hat{\mathbf{s}}, \hat{\mathbf{e}})$ when $\hat{\mathbf{e}} \in \mathbb{Z}^w$ is suitably small.
  • parity-check matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$
  • $G$-trapdoor $\mathbf{R} \in \mathbb{Z}^{\overline{m} \times kn}$ for $\mathbf{A}$
  • vector $\mathbf{b} = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{A}\mathbf{s} + \mathbf{e}$ for any $\mathbf{s} \in \mathbb{Z}_{2q}$ and suitably small $\mathbf{e} \in \mathbb{Z}^m$
Output: The vectors $\mathbf{s}$ and $\mathbf{e}$. The steps are:
  • Compute $\hat{\mathbf{b}} = [\mathbf{R}||\mathbf{I}]\,\mathbf{b} \mod q$.
  • Get $(\hat{\mathbf{s}} \mod q, \hat{\mathbf{e}}) \leftarrow \mathcal{O}(\hat{\mathbf{b}})$.
  • Get $\mathbf{e} = \mathbf{b} - \mathbf{A}\hat{\mathbf{s}} \mod q$ (interpreted as a vector in $\mathbb{Z}^m$ with entries $||\mathbf{e}_i|| < \frac{q}{2}$).
  • Solve for $\mathbf{s}$ using Gaussian elimination for $\mathbf{A}\mathbf{s} = \mathbf{b} - \mathbf{e} \mod 2q$.
  • Return $\mathbf{s}$ and $\mathbf{e}$.
```

Lemma 4.7. *Suppose that oracle $\mathcal{O}$ in Algorithm $2'$ correctly inverts $g_{\mathbf{G}}(\hat{\mathbf{s}}, \hat{\mathbf{e}})$ for any error vector $\mathcal{P}_{1/2}(q\mathbf{B}^{-1})$ for some $\mathbf{B}$. Then for any $\mathbf{s}$ and $\mathbf{e}$ of length $||e|| < \frac{q}{2||\mathbf{B}||s}$ where $s = \sqrt{s_1(\mathbf{R})^2 + 1}$, algorithm $2'$ correctly inverts $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$, where $s_1(\mathbf{R})$ denotes the largest singular value of the singular value decomposition (SVd) of $\mathbf{R}$ and for our trapdoor $\mathbf{R}$ is no more than $\sqrt{m}$. Moreover, for any $\mathbf{s}$ and random $e \leftarrow \mathcal{D}_{\mathbb{Z}^m, \alpha q}$ where $\frac{1}{\alpha} \geq 2||\mathbf{B}||s \cdot \omega(\sqrt{logn})$, algorithm $2'$ correctly inverts with high probability.*

Proof. Since this theorem is almost exactly Theorem 5.4 from [MP12], the proof is almost exactly the same. The oracle existence follows from lemma 2.1. Why everything holds mod q follows exactly from that proof. The only thing we need to argue is to why we can reconstruct $\mathbf{s}$ over the modulus $2q$. This should be obvious: since we have kept the parameters set so that we can always recover the error e exactly using the $\mod q$ formulation, we can plug it back into our original equation mod $2q$ and solve for $\mathbf{s}$ using Gaussian elimination. ∎

Lemma 4.8. *Let m, n, and q be integers such that $m \geq 4n \log q$. Let $\overline{m} = w = 2n \log q$, and and let the distribution $\mathcal{D}$ output matrices $\mathbf{R} \in \mathbb{Z}^{w \times \overline{m}}$ such that each entry $\mathbf{R}_{ij}$ is equal to one with probability $\frac{1}{4}$, -1 with probability $\frac{1}{4}$, and zero otherwise. Finally, let the distribution $\mathcal{D}_{\mathbb{Z}^m,\sigma}$ be a discrete Gaussian distribution with $\sigma \leq \frac{q}{5\sqrt{m} \log n}$. Then the* Invert *algorithm works with all but negligible probability.*

Proof. This follows directly from lemmas 4.6 and 4.7. ∎

4.6 Putting It All Together

We next combine lemmas from the previous subsections in order to give a direct reduction from a standard LWE instance.

Lemma 4.9. *Let m, n, and q be integers, and let the matrix $\mathbf{B} \in \mathbb{Z}_q^{m \times n}$ have each row sampled from some distribution $\mathcal{T}$. Let Q be the number of queries made to our NLWLR oracle. Let ψ' be a distribution over $\mathbb{Z}_{2q}$ that is statistically close to uniform over $\mathbb{Z}_2$. Any adversary that can solve the $(q, n, 2\psi + \psi', \mathcal{U}_q^n, \mathbf{B})$-nLWE problem with advantage ε can be used to solve the $\left(q, n, \psi, \mathcal{U}_q^n, \mathcal{T}\right)$-LWE problem with advantage $\frac{1}{Q}\varepsilon$.*

Proof. This follows from applying lemmas 4.1 and 4.3 and carefully keeping track of parameters. ∎

We note that this is the standard version of the LWE problem if ψ is picked to be a Gaussian distribution and $\mathcal{T}$ is picked to be uniform. We can now instantiate this assumption so that it is amenable to trapdoor sampling.

Lemma 4.10. *Let m, n, and q be integers such that $m \geq 4n \log q$. Let $\overline{m} = w = 2n \log q$, and and let the distribution $\mathcal{D}$ output matrices $\mathbf{R} \in \mathbb{Z}^{w \times \overline{m}}$ such that each entry $\mathbf{R}_{ij}$ is equal to one with probability $\frac{1}{4}$, -1 with probability $\frac{1}{4}$, and zero otherwise. Finally, let the distribution $\psi = \mathcal{D}_{\mathbb{Z},\sigma}$ be a discrete Gaussian distribution with $\sigma \leq \frac{q}{5m \log n}$. Let $\mathbf{B} \in \mathbb{Z}_q^{m \times n}$ be the output of GenTrap. We also require that $\sigma \geq 3\eta_\varepsilon\left(\mathbb{Z}^m\right)$.*

Then there exist GenTrap and Invert algorithms that are correct with all but negligible probability. In addition, any adversary that can solve the $(q, n, \mathcal{D}_{\mathbb{Z},\sigma}, \mathcal{U}_q^n, \mathbf{B})$-nLWE problem with advantage ε can be used to solve the $\left(q, n, \mathcal{D}_{\mathbb{Z},\frac{\sigma}{3}}, \mathcal{U}_q^n, \mathcal{U}_q^n\right)$-LWE problem with advantage $\frac{1}{Q}\varepsilon$.

Proof. This lemma follows from combining lemmas 4.9 and 4.8. ∎

5 Conclusion and Open Problems

In this work, we developed a deterministic variant of LWE with polynomial modulus and unbounded samples and showed that it is as hard as standard LWE. To our knowledge, this is the first such construction. Below we summarize our results in a table. Let c be a constant[7], $\gamma \geq 1$, let B be the B-bound for the noise distribution of the LWE instance that the LWR instance reduces from, qnd let κ be a (negligible) security parameter. We have:

[7]for typical choices of parameters

Work	**Unbounded Samples (w)**	**Modulus (q)**	**Advantage Change ($\varepsilon \to \varepsilon'$)**
[BPR12]	Yes	$Bp\kappa^{\omega(1)}$	$\varepsilon - \texttt{negl}(\kappa)$
[AKPW13]	No	$\gamma Bwp\kappa$	$\varepsilon/(2dw)$
[BGM$^+$16]	No	Bwp	$(\varepsilon/qw)^2$
[BLL$^+$15]	No	Bwp	$(\varepsilon/qw)^2$
[AA16] (1)	No	$Bwp\kappa$	$\varepsilon(wB)^{-c}$
[AA16] (2)	No	$Bwp\kappa$	$\varepsilon(w)^{-c}$
This	**Yes**	$O\left(B\sqrt{n \log q}\right) \cdot \omega\left(\sqrt{\log n}\right)$	$\varepsilon\left(w\right)^{-1}$

Work	**Dimension Change ($d \to d'$)**	**Straightforward Rounding**	**Uniform Samples**
[BPR12]	d	Yes	Yes
[AKPW13]	$d \log(\gamma)/\log q$	Yes	Yes
[BGM$^+$16]	$d/log_\rho q$	Yes	Yes
[BLL$^+$15]	d	Yes	Yes
[AA16] (1)	d	Yes	Yes
[AA16] (2)	$d - c$	Yes	Yes
This	$d/O\left(\log q\right)$	**No**	**No**

While our construction does offer unbounded samples and a polynomial modulus (and thus a polynomial approximation factor to worst-case lattice problems), it has some rather large drawbacks as well. The fact that the distribution of the samples $\mathbf{A}_i$ is not uniform makes this new assumption much more difficult to use in practice. Additionally, rounding to a lattice rather than to the nearest multiple of some integer p means that we lose most (if not all) of the efficiency advantages of rounding when compared to regular LWE.

It is our hope that future research can be done to eliminate these steps or to provide evidence as to why they are essential to the security of learning with rounding over a polynomial modulus with unbounded samples.

On Building PRFs. Given that we are building a deterministic variant of LWE that supports an unbounded number of queries, a natural question to ask is whether we can build fully parallelizable PRFs using the NLWLR assumption with a polynomial modulus q. Unfortunately, this still seems like it will require a substantial amount of new ideas. Note that even if we could prove that the standard form of LWR was exactly as hard as LWE, then constructing a lattice-based PRF with polynomial modulus would still require new ideas. This is because all of the known parallelizable lattice-based PRFs involve at least some form of subset product LWE. In other words, these PRFs require things of the form

$$\prod_{i=1}^{\ell} \mathbf{A}_{i,b_i}\mathbf{s} + \boldsymbol{\delta}_j$$

for 'random' matrices $\mathbf{A}_i$, a secret key $\mathbf{s}$, input bits b_i, and fresh noise samples $\boldsymbol{\delta}_j$ to be hard. While the clever construction in [BP14] attempts to minimize the noise blowup of these subset products, even it must have these subset products in some (lesser) depth to maintain a healthy amount of parallalelizability.

We consider proving the hardness of this subset product LWE problem for a polynomial modulus (and reducing the hardness of it to a polynomial value for the LWE security reduction parameter α) or providing evidence why it cannot be easily reduced to be an important open problem in this area.

Acknowledgements

We would like to thank Arnab Roy, Avradip Mandal, Sam Kim, David Wu, and Mike Hamburg for discussing lattices with us. We also want to thank the anonymous reviewers of the PQCrypto committee for their useful comments and suggestions on the paper and, in particular, anonymous reviewer #2 for their extremely helpful and detailed review.

References

[AA16] Jacob Alperin-Sheriff and Daniel Apon. Dimension-preserving reductions from LWE to LWR. *IACR Cryptology ePrint Archive*, 2016:589, 2016.

[AGHS13] Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai. Discrete Gaussian leftover hash lemma over infinite domains. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 97–116, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 57–74, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[BCD$^+$16] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In *ACM CCS 16: 23rd Conference on Computer and Communications Security*, pages 1006–1018. ACM Press, 2016.

[BDK$^+$11] Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, François-Xavier Standaert, and Yu Yu. Leftover hash lemma, revisited. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 1–20, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

[BDK$^+$17] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. CRYSTALS - kyber: a cca-secure module-lattice-based KEM. *IACR Cryptology ePrint Archive*, 2017:634, 2017.

[BFP$^+$15] Abhishek Banerjee, Georg Fuchsbauer, Chris Peikert, Krzysztof Pietrzak, and Sophie Stevens. Key-homomorphic constrained pseudorandom functions. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 31–60, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[BGG$^+$14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 533–556, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[BGM$^+$16] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 209–224, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[BKM17] Dan Boneh, Sam Kim, and Hart William Montgomery. Private puncturable prfs from standard lattice assumptions. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, pages 415–445, 2017.

[BLL$^+$15] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 3–24, Auckland, New Zealand, November 30 – December 3, 2015. Springer, Heidelberg, Germany.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 575–584, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 353–370, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, Palm Springs, California, USA, October 22–25, 2011. IEEE Computer Society Press.

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained prfs for nc^1 from LWE. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, pages 446–476, 2017.

[CKLS16] Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song. Lizard: Cut off the tail! practical post-quantum public-key encryption from lwe and lwr. 2016. `http://eprint.iacr.org/2016/1126`.

[DFH$^+$16] Stefan Dziembowski, Sebastian Faust, Gottfried Herold, Anthony Journault, Daniel Masny, and François-Xavier Standaert. Towards sound fresh re-keying with hard (physical) learning problems. In *Advances in Cryptology – CRYPTO 2016, Part II*, Lecture Notes in Computer Science, pages 272–301, Santa Barbara, CA, USA, August 2016. Springer, Heidelberg, Germany.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 464–479, Singer Island, Florida, October 24–26, 1984. IEEE Computer Society Press.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

[NR06] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 271–288, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

[Pei14] Chris Peikert. Lattice cryptography for the internet. In *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, pages 197–219, 2014.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, Maryland, USA, May 22–24, 2005. ACM Press.

[RRVV14] Sujoy Sinha Roy, Oscar Reparaz, Frederik Vercauteren, and Ingrid Verbauwhede. Compact and side channel secure discrete Gaussian sampling. Cryptology ePrint Archive, Report 2014/591, 2014. `http://eprint.iacr.org/2014/591`.