On the (M)iNTRU assumption in the integer case

Jim Barthel[1], Volker Müller[1], and Răzvan Roşie[2]

[1] University of Luxembourg, Luxembourg
{jim.barthel, volker.muller}@uni.lu
[2] Jao Luxembourg, Luxembourg
rosie@jao.eu

Abstract. In AsiaCrypt 2019, Genise, Gentry, Halevi, Li and Micciancio put forth two novel and intriguing computational hardness hypotheses: The inhomogeneous NTRU (iNTRU) assumption and its matrix version MiNTRU. In this work, we break the integer case of the iNTRU assumption through elementary lattice reduction, and we describe how the attack might be generalized to polynomial rings and to the low dimensional MiNTRU assumption with small noise.

Keywords: iNTRU, MiNTRU, cryptanalysis, lattice reduction

1 Introduction

Security reductions form the core of modern cryptography. Appraised by theorists, but largely ignored by programmers, reductions guarantee that specific attacks captured by some security experiment are infeasible. A reduction has two main components: some construction that needs to be proven secure and some problem that is assumed to be hard – usually denoted by the term *assumption*.

The last cryptographic epoch was synonymous with the raise of post-quantum cryptographic assumptions. Among these, lattice assumptions occupy a central role and today most of the provably secure lattice schemes rely on the *Learning-with-Errors* (LWE) problem, as described in the seminal paper of Regev [19]. A second group of assumptions is based on the NTRU problem, as postulated in [9]. While the former group is reducible to standard average-case assumptions, the latter is not. However, often the latter group offers superior practical performance, and results in this area are preferred for implementations. Besides the traditional definitions, there are a wide set of versions used in different sub-areas of cryptography, not all of them being deeply studied.

In this work, we consider two novel versions of the NTRU assumption from [6]. We show a practical attack against the one-dimensional version and generalize it to the multidimensional version with small dimension or small noise. In particular, our attacks show that both problems can directly be reduced to the *shortest vector problem*.

1.1 Contribution 1: Breaking the integer iNTRU assumption

The inhomogeneous NTRU (decision) problem (iNTRU) introduced in [6] consists in distinguishing between a random and a synthetically constructed $(\ell+1)$-tuple.

The synthetically constructed tuple follows the so-called iNTRU distribution that is obtained in two steps: First, a secret invertible ring element $s \in \mathcal{R}/q\mathcal{R}$ is randomly sampled and small error values e_i stemming from a specific error distribution are determined. Second, the tuple is defined by setting $a_0 := e_0/s \mod q$ and the remaining ℓ entries are fixed by $a_i := (2^{i-1} - e_i)/s \mod q$.

We analyse the integer iNTRU problem (i.e., $\mathcal{R} = \mathbb{Z}$) and develop two elementary lattice based distinguisher. Our key idea consists in replacing a_i by $b_i := 2a_i - a_{i+1} = (-2e_i + e_{i+1})/s \mod q$ and making so the entries independent of the blow up term 2^i. This change guarantees the existence of an extremely small vector (smaller than the expected heuristic value) inside well constructed lattices. A vector of this magnitude will not be contained in those lattices if the initial tuple was randomly sampled. Finally, simple lattice reduction spots the difference and even reveals the secret s.

1.2 Contribution 2: Generalizing the one-dimensional attack to the MiNTRU assumption

After introducing the one dimensional version of the inhomogeneous NTRU problem, the authors of [6] generalize it to matrices. The matrix inhomogeneous NTRU (decision) problem (MiNTRU) consists in distinguishing between a randomly sampled and a synthetically constructed matrix. The synthetically constructed matrix is again obtained in two steps: First, a random invertible matrix $\mathbf{S} \in \mathbb{Z}_q^{n\times n}$ is sampled and an error matrix $\mathbf{E} \in \mathbb{Z}_q^{n\times(n(\ell+1))}$ stemming from a specific error distribution is determined. Second, the challenge matrix is defined as $\mathbf{A} := \mathbf{S}^{-1} \times (\mathbf{G} - \mathbf{E}) \mod q$ where $\mathbf{G} = [0|\mathbf{I}|2\mathbf{I}|...|2^{\ell-1}\mathbf{I}]$ is an extended gadget matrix.

We generalize our previous iNTRU distinguisher to the multidimensional case. Again, the method relies on first eliminating the blow up factor 2^{i-1} (hidden in the gadget matrix), but this time even the secret matrix $\mathbf{S}$ will be canceled out, leading to a system of low norm matrices only. From there on, a well constructed lattice reveals again an extremely short vector, which can not be found in case of a random initial matrix. As the involved lattice dimensions are increasing, the output may deviate from the shortest vector, and so the secret will no longer be recovered. Nonetheless, this method has a relatively good success rate if the dimension n is small or if the error distribution for $\mathbf{E}$ is too narrow.

1.3 Disclaimer 1

We highlight that our attacks differ from the one in [12] based on [10].

Our iNTRU attack and its generalization follow standard lattice techniques and do not require advanced sub-lattice constructions. Although the latter may be used to simplify the construction, it is not needed. In particular, we will not get back to the highly useful methods from [10].

We note that besides using a different construction, we also work in a different context than [12]. Indeed, whereas they started from secret Bernoulli matrices

($-1, 0, 1$ entries), we work in a more general context where the secret matrix is sampled completely at random. In addition, our methods allow bypassing far larger entropic noise than their sub-lattice attack, but our approach is still somewhat dependent on the overstretched regime of the iNTRU instantiation.

Unfortunately, compared to their attack, our attacks are less powerful. More precisely, they managed to recover efficiently the secret matrix, thereby breaking the search version of the assumptions. We will be content with attacking the decision version without guarantee of recovering the secret.

Finally, we note that [6] has been updated to bypass the attack from [12] (which was in fact based on a toy example from [6]). In particular, the secret matrix is not a Bernoulli matrix anymore, implying that the size estimate used in [12] does no longer hold, and the final recovering step cannot be applied. We are not aware on how to use their attack on a completely random secret matrix.

1.4 Disclaimer 2

We remark that our attacks are devised for the integer case ($\mathcal{R} = \mathbb{Z}$) only, and that their efficiency for general rings is limited. Albeit they reflect a potential theoretical threat, our constructions are not strong enough to impact the security of recent cryptographic constructions such as [7] or [6] that either use iNTRU with rings of large degree or the MiNTRU with matrices of large dimension. Thereby, our contribution can only be seen as a first indication that the iNTRU (respectively the MiNTRU) assumption is not as hard as other well assumptions (like LWE). Further security analyses may be required to develop practical attacks against iNTRU (respectively MiNTRU) based cryptographic protocols.

1.5 Paper Organization

We start with setting the notations in Section 2 and continue this section with a reminder about lattices. In Section 3, we redefine the iNTRU assumption, and we quickly review its recent use. In Section 4 and Section 5, we develop two complementary lattice attacks against the one-dimensional integer iNTRU assumption. The first one will only be applicable if a specific invertibility condition is satisfied, and the second attack may be used in the opposite case. We complete our analysis in Section 6 with a short description on how to generalize our attacks to the polynomial ring iNTRU assumption and to the low dimensional MiNTRU assumption with low entropic noise. The latter assumption is formally redefined in Appendix B and the detailed attack is outlined in Appendix C. Due to the increasing approximation uncertainty of the shortest vector by lattice reduction, this attack is only guaranteed to work for MiNTRU instantiations of small dimensions or low entropic noise. We complete our analysis in Appendix D with a short comparison of the studied assumptions with the Learning-with-Errors assumption. Fully commented SageMath source codes corresponding to our attacks may be found in Appendix E and Appendix F or at `http://hdl.handle.net/10993/47990`.

2 Preliminaries

2.1 Notations

For a finite set S, we denote its cardinality by $|S|$ and the action of sampling an element x uniformly at random from S is denoted by $x \leftarrow_{\$} S$. When another, non-uniform distribution χ over the support set S is used, we abuse the notation and write $x \leftarrow \chi(S)$ or simply $x \leftarrow \chi$ if the support set is clear from context. We denote by $\|\cdot\| := \|\cdot\|_2$ the real Euclidean norm and by $\log := \log_2$ the base 2 logarithm. For an integer $q \geq 2$, we denote by $\mathbb{Z}/q\mathbb{Z}$ the ring of integers modulo q and we represent it using the 0-centered representation $\mathbb{Z}/q\mathbb{Z} = (-q/2, q/2] \cap \mathbb{Z}$. We denote an ordered list of n elements by $(a_1, \ldots, a_n)$. Lowercase variables in **bold** font, such as $\mathbf{a}$, usually denote (row) vectors and bold uppercase letters, such as $\mathbf{A}$, usually denote matrices.

2.2 Lattice preliminaries

Lattices Let $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{Z}^m$ be linearly independent row vectors. The row lattice generated by the basis $\mathbf{v}_1, \ldots, \mathbf{v}_n$ is the linear span

$$\Lambda = \mathcal{L}(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{v}_i \mid x_1, \ldots, x_n \in \mathbb{Z} \right\}.$$

We call a matrix $\mathbf{B}$ a basis matrix of Λ if Λ is generated by the rows of $\mathbf{B}$. It is well known that two bases $\mathbf{B}, \mathbf{B}'$ generate the same lattice if and only if there is an unimodular matrix $U \in GL(\mathbb{Z}, d)$ such that $\mathbf{B} = U\mathbf{B}'$. The determinant of a lattice Λ is defined by $\det(\Lambda) = \sqrt{\det(\mathbf{B}\mathbf{B}^T)}$ where $\mathbf{B}$ denotes any basis of Λ. Naturally, this determinant is independent of the chosen basis. The rank (or dimension) of a lattice is the dimension as a vector space of the lattice, and a lattice is full rank if it has maximal rank.

Successive minima For $i \in \{1, \ldots, n\}$, we define the i^{th} *successive minimum of* Λ as the smallest $r > 0$ such that Λ contains at least i linearly independent vectors of length bounded by r, $\lambda_i(\Lambda) = \inf\{r \in \mathbb{R}_{>0} : \dim(span(\Lambda \cap B(0, r))) \geq i\}$ where $B(0, r) = \{x \in \mathbb{R}^m : \|x\| \leq r\}$ is the closed ball of radius r around 0. The successive minima are achieved (thus, one may use the minimum instead of the infimum in its definition) and lattice points of norm $\lambda_i(\Lambda)$ are called i-th shortest vectors, but may not be unique. Minkowski's Second Theorem states that for each $1 \leq i \leq n$ the product $\left(\prod_{j=1}^{i} \lambda_j(\Lambda)\right)^{1/i} \leq \sqrt{\frac{n}{2\pi e}} \det(\Lambda)^{1/n}$.

LLL reduction Given $1/4 < \delta < 1$, a lattice basis $\mathbf{v}_1, \ldots, \mathbf{v}_n$ of Λ is LLL reduced with factor δ if the following holds

1. Size reduced: $\left| \frac{\langle \mathbf{v}_i, \mathbf{v}_j^* \rangle}{\langle \mathbf{v}_j^*, \mathbf{v}_j^* \rangle} \right| \leq \frac{1}{2}$ for all $1 \leq j < i < n$;

2. Lovász condition: $\|\mathbf{v}_j^*\|^2 \geq \left(\delta - \left|\frac{\langle \mathbf{v}_j, \mathbf{v}_{j-1}^* \rangle}{\langle \mathbf{v}_{j-1}^*, \mathbf{v}_{j-1}^* \rangle}\right|^2\right) \|\mathbf{v}_{j-1}^*\|^2$ for each $2 \leq j \leq n$;

where $\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*$ denote the Gram-Schmidt orthogonalization of the basis vectors. Traditionally $\delta = 3/4$, but in practice $\delta = 0.99$ is chosen. LLL reduced bases are not unique, but they have many desired properties. Indeed, let $\alpha = \frac{1}{\delta - \frac{1}{4}}$, then

1. $\|\mathbf{v}_j\| \leq \alpha^{\frac{n-1}{2}} \lambda_i(\Lambda)$ for all $1 \leq j \leq i \leq n$;
2. $\det(\Lambda) = \prod_{i=1}^n \|\mathbf{v}_i^*\| \leq \prod_{i=1}^n \|\mathbf{v}_i\| \leq \alpha^{\frac{n(n-1)}{2}} \det(\Lambda)$.

The LLL algorithm [13] outputs a LLL reduced basis of a rank n lattice in $\mathbb{Z}^m$ in time $O(n^5 m \log(K)^3)$ from basis vectors of norm less than K. [3]

Heuristics The Gaussian Heuristic (see [2] and [5]) yields that for a "random" lattice of "large" dimension, we expect the shortest vector to be of norm $\lambda_1(\Lambda) \simeq \sqrt{\frac{n}{2\pi e}} \det(\Lambda)^{1/n}$. Furthermore, in this case, all the lattice minima can be expected to be of approximately the same size.

Q-ary lattices If $q\mathbb{Z}^m \subseteq \Lambda \subseteq \mathbb{Z}^m$ for some $q \in \mathbb{Z}_{\geq 2}$, then Λ is called a q-ary lattice. We remark first that, by definition, every q-ary lattice has full rank $n = m$. Secondly, we observe that the lattice minima of a q-ary lattice are upper bounded by $\lambda_i(\Lambda) \leq q$ for all $i \in \{1, \ldots, m\}$. Given any matrix $\mathbf{A} \in (\mathbb{Z}/q\mathbb{Z})^{k \times m}$, we define the two special q-ary lattices $\Lambda_q(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m | \mathbf{y} = \mathbf{A}^T \mathbf{x} \mod (q)$ for some $\mathbf{x} \in \mathbb{Z}^k\}$ and $\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m | \mathbf{A}\mathbf{y} = 0 \mod (q)\}$. As a matter of fact, any q-ary lattice may be expressed as one of those lattices for some matrix $\mathbf{A} \in (\mathbb{Z}/q\mathbb{Z})^{k \times m}$ and $\det(\Lambda_q(\mathbf{A})) \geq q^{m-k}$ with equality if $\mathbf{A}$ is non-singular. Due to their special structure, q-ary lattices can not be seen as random (as required for the Gaussian heuristic). Nonetheless, [3] states that the Gaussian heuristic appears to hold exceedingly well for such lattices. A bit more precisely, [22, Lemma 2.18] proves that for fixed prime q and $m \geq k$, and for a randomly sampled matrix $\mathbf{A} \in (\mathbb{Z}/q\mathbb{Z})^{k \times m}$, the first lattice minimum is lower bounded by $\min\left\{q,\ \sqrt{\frac{m}{8\pi e}} q^{\frac{m-k}{m}}\right\}$ with probability greater than $1 - 2^{-m}$. We note that $\sqrt{\frac{m}{8\pi e}} q^{\frac{m-k}{m}}$ corresponds to half the Gaussian heuristic if $\mathbf{A}$ is non-singular.

Our lattices Hereinafter, we will use particular q-ary lattices where $\mathbf{A} \in (\mathbb{Z}/q\mathbb{Z})^{1 \times m}$ with a fixed entry equal to 1 and where q is not necessarily a prime. Although, none of the above results perfectly match our setup, we assume that, with noticeable probability, the first lattice minimum satisfies

$$\lambda_1(\Lambda) \geq \min\left\{q,\ \sqrt{\frac{m}{8\pi e}} q^{\frac{m-1}{m}}\right\}. \tag{H}$$

[3] Hereinafter, we will only use the LLL algorithm for lattice reduction. Better results may be achieved using recent results on the BKZ algorithm (see [14]). However, the LLL algorithm suffices for our elementary analysis.

3 The iNTRU assumption

In this section, we (re-)define the inhomogeneous NTRU (iNTRU) assumption, we describe some variants and outline its use.

3.1 The iNTRU assumption

The iNTRU problem has initially been introduced in [6, Section 4.1] formula (3).

Definition 1 (iNTRU distribution). *Let $\mathcal{R}$ be a ring, q any modulus, $\ell = \lceil \log(q) \rceil$ and χ be a symmetric error distribution over $\mathcal{R}$ producing with overwhelming probability elements with norm $\ll q$. Define the* iNTRU *distribution with these parameters to be obtained by the following sampling process*

$$\mathsf{iNTRU} = \left\{ \begin{array}{l} s \leftarrow_{\$} (\mathcal{R}/q\mathcal{R})^{\times} \\ e_i \leftarrow \chi \quad \forall i \in \{0, ..., \ell\} \\ a_0 := e_0/s \mod q \\ a_i := (2^{i-1} - e_i)/s \mod q \quad \forall i \in \{1, ..., \ell\} \end{array} \right\} \tag{1}$$

and denote any such tuple $(a_0, \ldots, a_\ell)$ by iNTRU *tuple.*

Definition 2 (iNTRU search problem). *Given an* iNTRU *tuple $(a_0, \ldots, a_\ell)$ and a modulus q, the* iNTRU *search problem consists in finding the hidden secret s. The* iNTRU *search assumption predicts that s can only be determined with negligible probability.*

Definition 3 (iNTRU decision problem). *Given a tuple $(x_0, \ldots, x_\ell)$ and a modulus q, the* iNTRU *decision problem consists in distinguishing whether the tuple has been sampled using the* iNTRU *distribution or the uniform distribution over $\mathcal{R}/q\mathcal{R}$. The* iNTRU *decision assumption predicts that such a distinction can only be made with negligible probability.*

We highlight that in [6] only the decision variant has been defined. However, in practice, the search variant may be used.

3.2 Further remarks

Hereinafter, we will point out some particular points of the definitions:

1. The iNTRU definition gives no limitation for the modulus q. Indeed, theoretically $\mathcal{R}/q\mathcal{R}$ might only be a ring and does not need to be a field.
2. If $\mathcal{R} = \mathbb{Z}$, then the underlying error distribution χ may be considered to be the discrete Gaussian distribution with variance $\sigma_\chi = O(\sqrt{q})$ (following a suggestion of [6]).
3. One can define shortened iNTRU tuples by removing the first entries of an iNTRU tuple. Especially, the first entry a_0 is sometimes removed.

3.3 Applications

Currently, the iNTRU assumption has only been used in [7] to develop two ring based short integer solution lattice trapdoors. The pseudorandomness of those trapdoors stems directly from the iNTRU assumption. We will fast revise their construction.

Trapdoors Intuitively, a trapdoor is some information that allows to invert a function [4]. For instance, if $f_{\mathbf{a}}(\mathbf{x}) \coloneqq \mathbf{a} \cdot \mathbf{x}^T$, where $\mathbf{a}, \mathbf{x}$ are elements over some polynomial quotient ring $(\mathcal{R}/q\mathcal{R})^m$, then a trapdoor would allow recovering $\mathbf{x}$.

Short Integer Solutions The *Short Integer Solution* (SIS) problem (see [1] for the original integer case, [16] for the ring based definition, and [11] for the module version) is a standard cryptographic problem which, in the ring version, asks to find, for a given vector $\mathbf{a} \in (\mathcal{R}/q\mathcal{R})^m$ and a bound value $\beta \in \mathbb{R}_{>0}$, a vector $\mathbf{x} \in (\mathcal{R}/q\mathcal{R})^m$ such that $f_{\mathbf{a}}(\mathbf{x}) = \mathbf{a} \cdot \mathbf{x}^T = 0$ and $\|\mathbf{x}\|_\rho < \beta$ for a suitable metric $\|\cdot\|_\rho$.

Usually, this problem is tackled by an elementary trapdoor mechanism. More precisely, assume that it is easy to solve the short integer solution problem for $f_{\mathbf{g}}$ where $\mathbf{g}$ is a known vector called the gadget. Assume further to know a low norm matrix $\mathbf{R}$, called a $\mathbf{g}$-trapdoor, such that $\mathbf{a} \cdot \mathbf{R} = \mathbf{g}$. Then, it is easy to solve the initial short integer solution problem. Indeed, one first samples at random any $\mathbf{x}$ such that $f_{\mathbf{g}}(\mathbf{x}) = 0$ and $\|\mathbf{x}\|_\rho < \beta$ (which is supposedly easy to find). Next, one computes $\mathbf{x}' := \mathbf{R} \cdot \mathbf{x}$ and finally hopes that it still fulfills $\|\mathbf{x}'\|_\rho < \beta$, which, due to the low norm entries of $\mathbf{R}$, generally holds. The preimage construction (i.e., finding $\mathbf{x}$) is commonly based on a discrete Gaussian sampling procedure (see [8] for a broad overview).

Despite their utility, such trapdoors may incorporate a security thread. Indeed, as outlined above, anyone knowing the trapdoor may solve the initial problem. Thereby, a necessary security feature required by such a trapdoor is that it is difficult to be guessed or put another way, it should be pseudorandom. [7] constructs two such trapdoors as follows.

Their idea The main idea behind their trapdoors is to use the inherent trapdoor potential of the iNTRU distribution. Concretely, a shortened iNTRU tuple $\mathbf{a} = (a_1, \ldots, a_\ell)$ can be represented as $\mathbf{a} = s^{-1}(\mathbf{g} + \mathbf{e})$ where $\mathbf{g} = (1, 2, 2^2, \ldots, 2^{\ell-1})$ is the gadget vector, $\mathbf{e} = (e_1, \ldots, e_\ell) \leftarrow_{\$} \chi^\ell$ and $s \in (\mathcal{R}/q\mathcal{R})^\times$ (they even choose $s \leftarrow \chi$). Since $s\mathbf{a} = \mathbf{g} + \mathbf{e} \approx \mathbf{g}$, the secret s is almost a $\mathbf{g}$-trapdoor for $\mathbf{a}$, falling only short of the corresponding error vector $\mathbf{e}$.

Their trapdoor generation Unfortunately, such a direct construction might leak some information on the trapdoor (we omit the details here). To bypass this leakage, the authors suggest replacing the gadget $\mathbf{g}$ by an approximate gadget $\mathbf{f} = (2^j, \ldots, 2^{\ell-1})$ for some $j \in \mathbb{N}_{>1}$ (often $j = \lceil \log(q)/2 \rceil$) and to proceed in the usual way, i.e., $\mathbf{a} = s^{-1}(\mathbf{f} + \mathbf{e})$. The $\mathbf{f}$-trapdoor will then consist in $(s, \mathbf{e})$.

The main difference of their constructions are the preimage sampling processes. These rely on different choices of perturbations and provoke a change in the discrete Gaussian sampling step. We omit the technical details about the procedures, as they will not be affected by what follows.

Pseudorandomness In the described construction, trapdoor pseudorandomness stems directly from the pseudorandomness of iNTRU tuples. Indeed, if, for given $\mathbf{a}$ (and implicitly the approximate gadget $\mathbf{f}$), one could retrieve the secret s, then one can also find the error tuple $\mathbf{e}$ and so the desired trapdoor.

3.4 Our contribution

We are going to prove that neither the decision, nor the search variant of the iNTRU assumption are safe in the integer case $\mathcal{R} = \mathbb{Z}$ and so the iNTRU distribution is not pseudorandom. Independent of the chosen modulus and the exact error distribution, our lattice attacks will distinguish with noticeable probability between random tuples and synthetically constructed ones, and they will retrieve the hidden secret in the latter case.

Our first attack is the natural choice when facing a challenge tuple. It will slightly modify the challenge entries and then construct a lattice. In the presence of a random challenge tuple, the shortest vector of this lattice will follow a specific heuristic (close to the Gaussian heuristic) whereas for a synthetically constructed one, the shortest vector will be far smaller and can be used to retrieve the secret. Unfortunately, the involved transformations include a modular inversion which may not be feasible in some cases, and one may bypass the attack by a suitable construction.

Our second attack can be used in case the first attack does not apply. It follows a similar transformation chain, but does not involve a modular inversion. The basic idea of the attack is the same, but this time the second-shortest vector will be compared to the heuristic. Due to this non-standard approach and an increased complexity, the first attack is preferable in most cases.

Although our attacks are conceived for the integer case, one may generalize them for polynomial rings. Such a generalization however needs to be carried out carefully, as the lattice dimension will increase and limits the applicability of the attack. Thereby, the attack is predicted to work for low degree polynomial rings only.

4 Attacking the iNTRU assumption - First approach

In this section, we describe our first lattice attack against the search and decision variant of the iNTRU assumption. We develop the attack for $\mathcal{R} = \mathbb{Z}$ only, but we emphasise that it can be generalized to low degree polynomial rings. Our attack first outlines whether a given tuple stems from the iNTRU distribution and if so it will find the underlying secret s. The development is based on full length challenge tuples but can, through small changes, also be applied to shortened tuples.

4.1 Our first lattice and its properties

Let $(x_0, \ldots, x_\ell)$ be an challenge tuple corresponding either to the uniform or the iNTRU distribution.

Lattice construction First, we slightly modify our challenge tuple and set

$$y_0 := x_0 \mod q \quad \text{and} \quad y_i = 2x_i - x_{i+1} \mod q \quad \forall i \in \{1, \ldots, \ell - 1\}. \tag{2}$$

Let $t \in \{0, \ldots, \ell - 1\}$ be an index such that $\gcd(y_t, q) = 1$. If no such index exists, the subsequent development will not work and our second iNTRU attack needs to be used (c.f. Section 5)[4]. We set

$$z_i := y_t^{-1} y_i \mod q \quad \forall i \in \{0, \ldots, \ell - 1\} \tag{3}$$

where $z_t = 1$. Then, we define the $\ell \times \ell$ q-ary row lattice:

$$\Lambda = \mathcal{L} \begin{pmatrix} z_0 & \ldots & z_{t-1} & 1 & z_{t+1} & \ldots & z_{\ell-1} \\ q & \ldots & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & \ldots & q & 0 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 0 & q & \ldots & 0 \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & 0 & 0 & \ldots & q \end{pmatrix} \tag{4}$$

4.2 Case of a random tuple

Assume that our initial challenge tuple $(x_0, \ldots, x_\ell)$ was sampled uniformly at random. Then, our constructed variables y_i as well as z_i (except $z_t = 1$) will still follow the uniform distribution as they involve only the addition and multiplication of random variables. Thereby, the shortest lattice vector can be expected to follow heuristic **H** and to satisfy

$$\lambda_1(\Lambda) \geq \min \left\{ q, \sqrt{\frac{\ell}{8\pi e}} q^{\frac{\ell-1}{\ell}} \right\}. \tag{5}$$

4.3 Case of a synthetic tuple

Assume next that the initial tuple $(x_0, \ldots, x_\ell)$ has been synthetically constructed following the iNTRU distribution. We will show that in this case the lattice contains a non-trivial short vector being magnitudes smaller than the expected heuristic.

[4] For $\mathcal{R} = \mathbb{Z}$ and random $x_0, \ldots, x_\ell$, the probability that our first attack cannot be used is only $\left(1 - \frac{\phi(q)}{q}\right)^\ell$ where ϕ denotes the Euler totient function. In particular, if q is prime our first attack should always work.

First observation We recall that a synthetically constructed tuple $(a_0, \ldots, a_\ell)$ following the iNTRU distribution satisfies

$$a_0 = \frac{e_0}{s} \mod q \quad \text{and} \quad a_i = \frac{2^{i-1} - e_i}{s} \mod q \quad \forall i \in \{1, \ldots, \ell\} \tag{6}$$

where $e_0, \ldots, e_\ell$ denote random errors sampled from the symmetric error distribution χ producing with overwhelming probability small elements. Thereby,

$$y_0 = \frac{e_0}{s} \mod q \quad \text{and} \quad y_i = \frac{-2e_i + e_{i+1}}{s} \mod q \quad \forall i \in \{1, \ldots, \ell - 1\} \tag{7}$$

where the numerators are still quite small. More precisely, the numerators follow the distribution χ' where:

1. The mean $\mu_{\chi'}$ of χ' is given by
$$\mu_{\chi'} = -2\mu_\chi + \mu_\chi = 0$$
where the first equality stems from the distribution properties of sums of random variables as well as the fact that a_i and a_{i+1} follow the same distribution χ, and the second equality comes from the fact that the mean $\mu_\chi = 0$ since χ is a symmetric distribution (i.e., $-\chi = \chi$).
2. The variance $\sigma_{\chi'}$ of χ' is given by
$$\sigma^2_{\chi'} = 3\sigma^2_\chi$$
since all three variables follow the distribution χ.

In particular, we conclude that since χ produces with overwhelming probability elements with absolute value $\ll q$, so does χ'. Thereby, the numerators are expected to be quite small when compared to the modulus q.

Second observation Continuing to outline the effect of our variable changes leads to the conclusion that

$$z_i = \frac{-2e_i + e_{i+1}}{e'_t} \mod q \quad \forall i \in \{0, \ldots, \ell - 1\} \tag{8}$$

where $e'_t = e_0$ if $t = 0$ and $e'_t = -2e_t + e_{t+1}$ if $t \in \{1, \ldots, \ell - 1\}$. Thus, each z_i is the quotient of two small error elements.

The shortest lattice vector Interestingly, our two observations imply that our lattice contains the vector

$$\mathbf{v} = (e_0, (-2e_1 + e_2), \ldots, (-2e_{\ell-1} + e_\ell)) \tag{9}$$

obtained by multiplying the first row by e'_t and applying the "modulo q reduction", corresponding to an addition of the respective lines, as often as needed.

We are going to show that this vector will be almost surely smaller than the expected heuristic and the canonically short vectors with a single entry q. To do so, first assume that the error entries are upper bounded by some constant $K > 0$. i.e.,

$$\max\{|e_0|, |-2e_1 + e_2|, \dots, |-2e_{\ell-1} + e_\ell|\} \leq K. \tag{10}$$

Then, the size of v may be upper bounded by

$$\|\mathbf{v}\|_2 \leq \sqrt{\ell K^2} \leq \sqrt{\ell}\, K. \tag{11}$$

Proposition 1. *If* $\max\{|e_0|, |-2e_1 + e_2|, \dots, |-2e_{\ell-1} + e_\ell|\} \leq \min\left\{\frac{q}{\sqrt{\ell}}, \frac{1}{\sqrt{8\pi e}} q^{(\ell-1)/\ell}\right\}$*, then the target vector* $\mathbf{v}$ *is shorter than* $\min\left\{q, \sqrt{\frac{\ell}{8\pi e}} q^{\frac{\ell-1}{\ell}}\right\}$.

Proof. Replace K in Equation (11) with the predicted values and compare. □

As in practice the error terms are $O(\sqrt{q})$, the size condition is almost always satisfied. For comparison, the probability that a completely randomly sampled ℓ-tuple would be of this size is lower than $\left(\frac{2\sqrt{\ell}K+1}{q}\right)^{\ell}$, which is rapidly decreasing for small K.

Lattice reduction We need to make sure that our target vector v can also be determined. We will show that upon slightly decreasing the upper bound K, we are guaranteed that ordinary LLL reduction returns a vector that is smaller than the expected heuristic. In general, the first LLL reduced vector with factor δ will not be a smallest lattice vector, but only a good approximation of it. More precisely, the first LLL reduced vector $\mathbf{w}_1$ satisfies theoretically $\|\mathbf{w}_1\| \leq \alpha^{\frac{\ell-1}{2}} \lambda_1$ where λ_1 denotes the length of a shortest lattice vector and $\alpha = \frac{1}{\delta - \frac{1}{4}}$. However, in practice, this artificial blowup is barely observed. We note that for increasing δ, the blow-up factor α decreases. For the sake of explicit results, we consider $\delta = \frac{63}{64} < 0.99$ resulting in $\alpha = \frac{64}{47} < \sqrt{2}$. By Equation (11), we know that $\lambda_1 \leq \|v\|_2 \leq \sqrt{\ell}K$. This implies that

$$\|\mathbf{w}_1\|_2 \leq \alpha^{\frac{\ell-1}{2}} \sqrt{\ell}K \leq 2^{\frac{\ell-1}{4}} \sqrt{\ell}K \leq 2^{\frac{\log(q)}{4}} \sqrt{\ell}K \leq q^{\frac{1}{4}} \sqrt{\ell}K. \tag{12}$$

Proposition 2. *If* $\max\{|e_0|, |-2e_1 + e_2|, \dots, |-2e_{\ell-1} + e_\ell|\} \leq \min\left\{\frac{q^{3/4}}{\sqrt{\ell}}, \frac{1}{\sqrt{8\pi e}} q^{\frac{3\ell-4}{4\ell}}\right\}$*, then* $\|\mathbf{w}_1\|_2$ *is smaller than* $\min\left\{q, \sqrt{\frac{\ell}{8\pi e}} q^{\frac{\ell-1}{\ell}}\right\}$.

Proof. Replace K in Equation (12) by the predicted values and compare. □

As usually $K = O(\sqrt{q})$, we can expect the condition of Proposition 2 to hold in practice. We highlight also that finding another vector of this magnitude is rather improbable, and we can even expect $\mathbf{w}_1 = \pm\mathbf{v}$.

4.4 Conclusion

We conclude that:

1. In case of a random tuple, the first LLL reduced vector can be expected with noticeable probability to be lower bounded by $\min\left\{q,\ \sqrt{\frac{\ell}{8\pi e}} q^{\frac{\ell-1}{\ell}}\right\}$.
2. In case of a synthetic tuple, the first LLL reduced vector is with high probability smaller than the predicted value $\min\left\{q,\ \sqrt{\frac{\ell}{8\pi e}} q^{\frac{\ell-1}{\ell}}\right\}$ and we can even expect it to be equal to $\pm\mathbf{v}$ where $\mathbf{v} = (e_0, (-2e_1 + e_2), \ldots, (-2e_{\ell-1} + e_\ell))$ is our target vector.

Hence, we can distinguish with noticeable probability between a randomly sampled tuple and a synthetically constructed one by simply comparing the length of the first LLL reduced vector with $\min\left\{q,\ \sqrt{\frac{\ell}{8\pi e}} q^{\frac{\ell-1}{\ell}}\right\}$. Furthermore, in case of a synthetically constructed tuple, the first LLL reduced vector is expected to reveal the modified error terms. Choosing the error term in the t-th position e'_t, and computing $\frac{e'_t}{y_t} \mod q = \pm s$ reveals then the hidden secret s. The corresponding SAGEMATH source codes for our first distinguisher (distinguisher1) can be found in Appendix E.

Optional bootstrapping We note that in the whole development, we never assumed to know the precise error distribution. Indeed, multiple passages could have been formalized and simplified when the error distribution was known (e.g., Discrete Gaussian). Besides retrieving the secret s, our method even allows to retrieve the underlying error distribution. Indeed, upon reception of the secret s, one easily reveals the original error values $e_0, \ldots, e_\ell$. Once the errors have been determined, any bootstrapping method may simulate the whole error distribution.

5 Attacking the iNTRU assumption - Second approach

In this section, we describe our second attack against the integer iNTRU assumptions. Whereas our first attack is foremost suitable for prime moduli q, it may not be used under unfortunate circumstances, namely if none of the y_i is invertible. This can be easily determined and in the improbable case this happens, one needs to opt for our second attack. Although slightly more challenging and bound on a more restricted success probability, our second attack will work for any initial challenge input. However, due to its increased complexity and unusual approach, the first method should be used if possible.

5.1 Our second lattice and its properties

Let $(x_0, \ldots, x_\ell)$ be an iNTRU challenge tuple corresponding either to the uniform or the iNTRU distribution.

Lattice construction Similar than in our first attack, we modify our challenge tuple by setting

$$y_0 := x_0 \mod q \quad \text{and} \quad y_i = 2x_i - x_{i+1} \mod q \quad \forall i \in \{1, \ldots, \ell - 1\}. \quad (13)$$

But contrary to the first attack, we stop our modifications and directly construct the $(\ell + 1) \times (\ell + 1)$ q^2-ary row lattice:

$$\Lambda = \mathcal{L} \begin{pmatrix} y_0 q & y_1 q & y_2 q & \ldots & y_{\ell-1} q & 1 \\ q^2 & 0 & 0 & \ldots & 0 & 0 \\ 0 & q^2 & 0 & \ldots & 0 & 0 \\ 0 & 0 & q^2 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & q^2 & 0 \end{pmatrix} \quad (14)$$

Predicted value Assuming the first row of Λ random, heuristic **H** yields that

$$\lambda_1(\Lambda) \geq \min \left\{ q^2, \sqrt{\frac{\ell + 1}{8\pi e}} q^{\frac{2\ell}{\ell+1}} \right\}. \quad (15)$$

However, here our heuristic needs to be considered with precaution as it only applies to random initial matrices $\mathbf{A}$ and prime moduli. Especially for our lattice Λ, where both of those conditions are not satisfied, we must pay attention and indeed, our lattice contains a shorter vector. More precisely, our lattice contains the vector $(0, \ldots, 0, q)$ obtained by multiplying the first row by q and then for each $i \in \{0, \ldots, \ell - 1\}$ subtracting y_i times row $i + 1$ from it. Thus, $\lambda_1 \leq q$. Nonetheless, our heuristic is a good indication for the size of the other successive minima and in particular for λ_2 (not to say that it is the only applicable estimation).

5.2 Case of a random tuple

Let the initial tuple $(x_0, ..., x_\ell)$ be randomly sampled at uniform from $(\mathbb{Z}/q\mathbb{Z})^{\ell+1}$. Then, also the corresponding y_i will follow the uniform distribution and so the first row of our lattice behaves, up to the common factor q and the last entry, almost randomly. We will prove that apart from our trivially short vector $(0, \ldots, 0, q)$, its suitable multiples, and the canonical vectors with a single entry q^2 (if applicable), it is improbable to find another vector smaller than the expected value heuristic.

Lemma 1. *Let $B \leq \frac{q}{2}$ be an integer and $S \subseteq \mathbb{Z}$ be fixed. Choose randomly $r \in \mathbb{Z}/q^2\mathbb{Z}$ and for each $i \in \{0, \ldots \ell - 1\}$, let also $y_i \in \mathbb{Z}/q\mathbb{Z}$ be random. Set*

$$\mathbf{y} = (y_0 q, y_1 q, \ldots, y_{\ell-1} q, 1).$$

Then, the probability that $r \in S$ and the norm of the vector $r\mathbf{y} \mod q^2$ is at most Bq is upper bounded by

$$\mathbb{P}\left(\left(\left\|[r\mathbf{y} \mod q^2]\right\|_2 \leq Bq\right) \cap (r \in S)\right) \leq \sum_{\substack{\beta_\ell=-Bq \\ \beta_\ell \in S}}^{Bq} \frac{\ell(2\lfloor B/\gcd(\beta_\ell, q)\rfloor + 1)}{q^2} \left(\frac{\gcd(\beta_\ell, q)}{q}\right)^\ell.$$

We note that the computation of $\left\|[r\mathbf{y} \mod q^2]\right\|_2$ takes place sequentially and component-wise. More precisely, first $r\mathbf{y}$ is computed, then each component is reduced modulo q^2 to its centrally symmetric representative (i.e., the smallest representative in absolute value), and finally the Euclidean norm is taken on the resulting vector as seen over the integers. The proof of Lemma 1 can be found in Appendix A.

To put the previous lemma into context, we point out that if $S = \{kq \mid k \in \mathbb{Z}\}$, then the probability is smaller than $\frac{1}{\sqrt{q}}$ and if $S = (\mathbb{Z}/q^2\mathbb{Z})^\times$ is the set of units, then the probability is smaller than $\frac{\ell}{q^{\ell-1}}$. Although our lattice contains the trivial short vector $(0, \ldots, 0, q)$ and its multiples, the probability of finding a short vector with nonzero entries on the first $\ell - 1$ entries is rapidly decreasing with increasing q. For small bounds B, we expect with a high probability that such a vector will not even exist. In this case, the best possible guess for the size of the second-shortest vector will be $\min\left\{q^2, \sqrt{\frac{\ell+1}{8\pi e}} q^{\frac{2\ell}{\ell+1}}\right\}$.

5.3 Case of a synthetic tuple

Assume next that the initial tuple $(x_0, \ldots, x_\ell)$ has been synthetically constructed following the iNTRU distribution. Then, we will prove that apart from our trivially short vector $(0, \ldots, 0, q)$, its multiples and the canonical vectors with a single entry q^2, our lattice contains at least one more linearly independent short vector.

Preliminary observation Similar than in our first attack, we note that a tuple $(a_0, \ldots, a_\ell)$ following the iNTRU distribution satisfies

$$a_0 = \frac{e_0}{s} \mod q \quad \text{and} \quad a_i = \frac{2^{i-1} - e_i}{s} \mod q \quad \forall i \in \{1, \ldots, \ell\} \tag{16}$$

where $e_0, \ldots, e_\ell$ denote random errors sampled from the symmetric error distribution χ producing with overwhelming probability small elements and that

$$y_0 = \frac{e_0}{s} \mod q \quad \text{and} \quad y_i = \frac{-2e_i + e_{i+1}}{s} \mod q \quad \forall i \in \{1, \ldots, \ell - 1\} \tag{17}$$

where the numerators follow the symmetric distribution χ' with $\mu_{\chi'} = 0$ and $\sigma_{\chi'} = \sqrt{3}\sigma_\chi$ and are thus still quite small.

Another short vector Our lattice contains the vector

$$\mathbf{v} = (e_0 q, (-2e_1 + e_2)q, \ldots, (-2e_{\ell-1} + e_\ell)q, s) \tag{18}$$

obtained by multiplying the first row by s and applying the "modulo q^2 reduction", corresponding to an addition of the respective lines, as often as needed. We are going to show that this vector will be almost surely smaller than the expected heuristic. To do so, we first assume that the error entries are upper bounded by some constant $K > 0$. i.e.,

$$\max\{|e_0|, |-2e_1 + e_2|, \ldots, |-2e_{\ell-1} + e_\ell|\} \leq K. \tag{19}$$

Then, the size of v may be upper bounded by

$$\|\mathbf{v}\|_2 \leq \sqrt{\ell K^2 q^2 + s^2} \leq \sqrt{\ell K^2 q^2 + q^2} \leq \sqrt{\ell + 1}\, qK. \tag{20}$$

Whenever K is small enough, this upper bound is smaller than the expected heuristic. That this smallness condition is in general no limitation is shown by the following proposition.

Proposition 3. *If* $\max\{|e_0|, |-2e_1 + e_2|, \ldots, |-2e_{\ell-1} + e_\ell|\} \leq \min\left\{\frac{q}{\sqrt{\ell+1}}, \frac{1}{\sqrt{8\pi e}} q^{\frac{\ell-1}{\ell+1}}\right\}$, *then the target vector* $\mathbf{v}$ *is shorter than* $\min\left\{q^2, \sqrt{\frac{\ell+1}{8\pi e}} q^{\frac{2\ell}{\ell+1}}\right\}$.

Proof. Replace K by the upper bound for max in Equation (20). □

As usually the error terms are $O(\sqrt{q})$, our target vector $\mathbf{v}$ will almost surely be smaller than the expected heuristic.

Lattice reduction Finally, it is time to check whether our target vector v can even be determined using ordinary lattice reduction. Hereinafter, we will only consider LLL reduction. An LLL reduced basis $(\mathbf{w}_1, \ldots, \mathbf{w}_{\ell+1})$ satisfies theoretically $\|\mathbf{w}_i\|_2 \leq \alpha^{\frac{\ell}{2}} \lambda_i$ where λ_i denotes the i-th successive minimum of the lattice and $\alpha = \frac{1}{\delta - \frac{1}{4}}$. For the sake of explicit results, we again consider $\delta = \frac{63}{64} < 0.99$ resulting in $\alpha = \frac{64}{47} < \sqrt{2}$.

Let us concentrate on the first LLL output, namely $\mathbf{w}_1$. We recall that our shortest vector will probably be $\mathbf{v}_0 = (0, ..., 0, q)$.[5] Thus, we assume $\lambda_1 \leq q$. This implies that

$$\|\mathbf{w}_1\|_2 \leq \alpha^{\frac{\ell}{2}} q \leq 2^{\frac{\ell}{4}} q \leq 2^{\frac{\log(q)+1}{4}} q \leq (2q)^{\frac{1}{4}} q.$$

Lemma 1 yields that such a short vector can only be found at random with extremely low probability Thus, it seems improbable that apart from $\mathbf{v}_0$ and its

[5] The only possibility for which this would not be the case takes place when $g = \gcd(y_0, \ldots, y_{\ell-1}, q) > 1$ as then $(0, \ldots, 0, \frac{q}{g})$ will be the shortest lattice vector, but this scenario is rather improbable.

multiples, another vector of this magnitude exists, and we expect $\mathbf{w}_1$ to be (a multiple of) $\mathbf{v}_0$.[6]

Next, we consider the second LLL output, namely $\mathbf{w}_2$. Assuming that $\mathbf{w}_1$ is a multiple of $\mathbf{v}_0$, the second LLL output $\mathbf{w}_2$ needs to contain non-zero entries on the first ℓ entries (otherwise it would not be linearly independent from $\mathbf{w}_1$). By Equation (20), we know that $\|\mathbf{v}\|_2 \leq \sqrt{\ell+1}\, qK$ where K denotes the maximal error term and by Proposition 3 we know that our target vector is almost surely smaller than $\min\left\{q^2,\ \sqrt{\frac{\ell}{8\pi e}} q^{\frac{2\ell}{\ell+1}}\right\}$. If we slightly reduce the upper bound K, we obtain a similar result for $\mathbf{w}_2$.

Proposition 4. *If* $\max\{|e_0|, |-2e_1+e_2|, \ldots, |-2e_{\ell-1}+e_\ell|\} \leq \min\left\{\frac{q^{3/4}}{2^{1/4}\sqrt{\ell+1}},\ \frac{1}{2^{3/4}\sqrt{\pi e}} q^{\frac{3\ell-5}{4(\ell+1)}}\right\}$*, then* $\|\mathbf{w}_2\|_2$ *is smaller than* $\min\left\{q^2,\ \sqrt{\frac{\ell}{8\pi e}} q^{\frac{2\ell}{\ell+1}}\right\}$.

Proof. Let $\max\{|e_0|, |-2e_1+e_2|, \ldots, |-2e_{\ell-1}+e_\ell|\} \leq K$. Then we know that $\|\mathbf{w}_2\|_2 \leq \alpha^{\frac{\ell}{2}}\lambda_2 \leq \alpha^{\frac{\ell}{2}}\mathbf{v}$. Using again the fact that $\alpha \leq \sqrt{2}$ implies $\alpha^{\frac{\ell}{2}} \leq (2q)^{\frac{1}{4}}$ and with Equation (20), we conclude

$$\|\mathbf{w}_2\|_2 \leq (2q)^{\frac{1}{4}}\sqrt{\ell+1}\, qK.$$

Replacing K by the claimed upper bounds concludes the proposition. □

The upper bound for K in Proposition 4 is $O(q^{\frac{3}{4}})$ and as usually $K = O(\sqrt{q})$, we can expect the condition to hold in practice. In comparison, using Lemma 1, we conclude that such a short vector will only be found at random with low probability. Thereby, we can even expect $\mathbf{w}_2 = \pm\mathbf{v}$.

5.4 Conclusion

Our cautious lattice analysis gives rise to multiple conclusions:

1. In case of a random tuple as well as in the case of a synthetically constructed one, the first LLL reduced vector will with overwhelming probability be a multiple of $(0, \ldots, 0, q)$.
2. In case of a random tuple, the second LLL reduced vector can be expected to be lower bounded by $\min\left\{q^2,\ \sqrt{\frac{\ell}{8\pi e}} q^{\frac{2\ell}{\ell+1}}\right\}$.
3. In case of a synthetic tuple, the second LLL reduced vector is with a high probability smaller than $\min\left\{q^2,\ \sqrt{\frac{\ell}{8\pi e}} q^{\frac{2\ell}{\ell+1}}\right\}$. Furthermore, we can expect it to be equal to $\pm\mathbf{v}$ where $\mathbf{v} = (e_0 q, (-2e_1+e_2)q, \ldots, (-2e_{\ell-1}+e_\ell)q, s)$ is our target vector.

[6] This conclusion also holds in the case of a random tuple.

Hence, we finally conclude that we can distinguish with noticeable probability between a randomly sampled tuple and a synthetically constructed one by simply comparing the length of the second LLL reduced vector with $\min\left\{q,\ \sqrt{\frac{\ell}{8\pi e}} q^{\frac{2\ell}{\ell+1}}\right\}$. Furthermore, in case of a synthetically constructed tuple, the last entry of this vector should reveal the secret value s or its negative $-s$. Especially if the initial errors are comparably small with respect to q (e.g., $O(\sqrt{q})$), there is a high chance of revealing the secret. The corresponding SAGEMATH source codes for our second distinguisher (distinguisher2) can be found in Appendix E.

Optional bootstrapping As in the first attack, we never assumed to know the precise error distribution. Similar to the first attack, after retrieving the secret s, one can extract the original error values $e_0, \ldots, e_\ell$ and use a suitable bootstrapping method to simulate the whole error distribution.

6 Generalizing our attacks

Our two lattice attacks against the iNTRU assumption have been conceived for the integer case ($\mathcal{R} = \mathbb{Z}$) only. However, it is not difficult to generalize them.

6.1 iNTRU - The general case

In, order to generalize our attacks to polynomial rings, one may simply replace the x_i's by the corresponding polynomial values and carry on with the constructions. The corresponding lattices will then contain the corresponding polynomial coefficients. The arising difficulty is that in the presence of degree d polynomials, the lattice dimension will increase by a factor of d as well. This blow-up heavily impacts the detectable error limits in Proposition 2 and Proposition 4 up to the point where only Bernoulli errors would be detectable (i.e. $-1, 0, 1$). To be precise, degree d polynomials would only lead to the theoretical bound $\|\mathbf{w}_1\|_2 \leq q^{\frac{d}{4}} K$ for the first LLL reduced vector in Equation (12) which, for large d, is not short enough to grant the required upper bound $\|\mathbf{w}_1\|_2 \leq q$. Likewise, for our second distinguisher, the theoretical bound $\|\mathbf{w}_2\|_2 \leq q^2$ will not be fulfilled for large d. As in general LLL performs better in practice than in theory, slightly larger degrees might be achievable, but only the use of stronger reduction algorithms such as BKZ (see [14]) will lead to well-functioning distinguishers.

6.2 MiNTRU

Our attacks can also be generalized to the second assumption introduced in [6], the so called matrix inhomogeneous NTRU (MiNTRU) assumption. The MiNTRU assumption essentially replaces polynomial elements in the iNTRU assumption by integer modular matrices. To be precise, let q be any modulus, $\ell = \lceil \log(q) \rceil$, $m = n(\ell + 1)$, $\mathbf{G} = [0|\mathbf{I}|2\mathbf{I}|...|2^{\ell-1}\mathbf{I}] \in \mathbb{Z}^{n \times m}$ an extended gadget matrix, and χ a symmetric error distribution over $\mathbb{Z}$ producing with overwhelming probability elements with norm $\ll q$. Then, the MiNTRU distribution with these parameters is obtained by the following sampling process

$$\mathsf{MiNTRU} = \begin{Bmatrix} \mathbf{S} \leftarrow (\mathbb{Z}/q\mathbb{Z})^{n\times n}_{invertible} \\ \mathbf{E} \leftarrow \chi^{n\times m} \\ \mathbf{A} := \mathbf{S}^{-1} \times (\mathbf{G} - \mathbf{E}) \mod q \end{Bmatrix}$$

and the MiNTRU decision problem requires to distinguish this distribution from the random distribution over $(\mathbb{Z}/q\mathbb{Z})^{n\times m}$. By decomposing the matrix $\mathbf{A}$ into $(\ell+1)$ individual $n \times n$ matrices $\mathbf{A}_0, \dots \mathbf{A}_\ell$ such that $\mathbf{A} = [\mathbf{A}_0 | \dots | \mathbf{A}_\ell]$, then setting $\mathbf{Y}_0 := \mathbf{A}_0 \mod q$ and $\mathbf{Y}_i = 2\mathbf{A}_i - \mathbf{A}_{i+1} \mod q \quad \forall i \in \{1, \dots, \ell - 1\}$ and finally computing $\mathbf{Z}_i := \mathbf{Y}_t^{-1}\mathbf{Y}_i \mod q \quad \forall i \in \{0, \dots, \ell-1\}$ for some invertible matrix $\mathbf{Y}_t$, removes completely the dependence of the gadget matrix $\mathbf{G}$ and ends up in matrices with entries of quotients of small norm error elements only. Thereby, the same strategy as in the iNTRU attack may be mounted against the underlying decision problem. However, once again, the success chance of our attack is strongly affected by the matrix dimensions. Concretely, the generalized attacks will only work for low dimensional matrices or matrices with low entropic noise. A more detailed analysis may be found in Appendix B and Appendix C.

7 Conclusion

Our simple lattice based distinguishers break with noticeable probability the integer case of the iNTRU assumption. Additionally, their construction yields a theoretical thread for the general iNTRU and the MiNTRU assumption. Nonetheless, this thread is not reflected in practice as the distinguishers don't cope with large dimensions of polynomial rings (iNTRU) or with large matrix dimensions (MiNTRU) as used in recent cryptographic constructions. This work should mainly raise awareness that new hardness hypotheses should be used with caution and questions whether the two studied assumptions can compete with the long-standing Learning-with-Errors (LWE) assumption [19] which does not suffer from the described vulnerability (see Appendix D for a short comparison). We emphasize that, if possible, well-known hardness assumptions should be used.

Acknowledgement

Jim Barthel was supported in part by the Luxembourg National Research Fund through grant PRIDE15/10621687/SPsquared. Răzvan Roşie was supported in part by ERC grant CLOUDMAP 787390.

References

1. M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 99–108. Association for Computing Machinery, 1996.
2. Miklós Ajtai. Generating random lattices according to the invariant distribution, March 2006.

3. Martin Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving lwe by reduction to unique-svp. In Hyang-Sook Lee and Dong-Guk Han, editors, *Information Security and Cryptology – ICISC 2013*, pages 293–310. Springer, 2014.
4. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
5. Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, pages 31–51. Springer, 2008.
6. Nicholas Genise, Craig Gentry, Shai Halevi, Baiyu Li, and Daniele Micciancio. Homomorphic encryption for finite automata. In *Advances in Cryptology – ASIACRYPT 2019*, LNCS, pages 473–502. Springer, December 2019.
7. Nicholas Genise and Baiyu Li. Gadget-based intru lattice trapdoors. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *Progress in Cryptology–INDOCRYPT 2020*, pages 601–623. Springer, 2020.
8. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC '08, page 197–206. ACM, 2008.
9. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic Number Theory*, Third International Symposium, June 21-25, 1998, pages 267–288. Springer, 1998.
10. Paul Kirchner and Pierre-Alain Fouque. Revisiting lattice attacks on overstretched ntru parameters. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 3–26. Springer, 2017.
11. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75, 06 2014.
12. Changmin Lee and Alexandre Wallet. Lattice analysis on mintru problem. Cryptology ePrint Archive, Report 2020/230, 2020. https://ia.cr/2020/230.
13. A. K. Lenstra, H. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.
14. Jianwei Li and Phong Q. Nguyen. A complete analysis of the bkz lattice reduction algorithm. Cryptology ePrint Archive, Report 2020/1237, 2020. https://ia.cr/2020/1237.
15. Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 333–342. Association for Computing Machinery, 2009.
16. Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography Conference*, volume 3876 of *LNCS*, pages 145–166. Springer, 2006.
17. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, page 187–196. Association for Computing Machinery, 2008.
18. M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.
19. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 84–93. Association for Computing Machinery, 2005.
20. Oded Regev. The learning with errors problem (invited survey). In *2010 IEEE 25th Annual Conference on Computational Complexity*, pages 191–204, 2010.
21. Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. *Foundation of Secure Computations*, 1978.
22. Weiqiang Wen. *Contributions to the hardness foundations of lattice-based cryptography*. PhD thesis, Université de Lyon, Computational Complexity, 2018. HAL.

A Proof of Lemma 1

We observe that by the properties of the Euclidean and the infinity norm, we have

$$\mathbb{P}\left(\left(\left\|[r\mathbf{y} \mod q^2]\right\|_2 \leq Bq\right) \cap (r \in S)\right) \leq \mathbb{P}\left(\left(\max\left\{\,\left|[r\mathbf{y} \mod q^2]\right|\,\right\} \leq Bq\right) \cap (r \in S)\right)$$

where the maximum is taken over all the modulo reduced entries of $r\mathbf{y}$. The right expression is equal to

$$\mathbb{P}\left(\left(\bigcap_{i=0}^{\ell-1}\underbrace{\left(\left|[ry_iq \mod q^2]\right| \leq Bq\right)}_{:=C_i}\right) \cap \underbrace{\left(\left|[r \mod q^2]\right| \leq Bq\right)}_{:=C_\ell} \cap (r \in S)\right).$$

Each event $C_0, \ldots, C_{\ell-1}$ in the probability statement can be written as a union of events $C_i = \bigcup_{\beta_i=-Bq}^{Bq}([ry_iq \mod q^2] = \beta_i)$. As this event can only take place whenever β_i is a multiple of q (otherwise, the equality cannot be satisfied), we need only to consider the restricted union of events $\bigcup_{\beta_i=-B}^{B}([ry_iq \mod q^2] = \beta_iq) = \bigcup_{\beta_i=-B}^{B}([ry_i \mod q] = \beta_i)$. Furthermore, $C_\ell = \bigcup_{\beta_\ell=-Bq}^{Bq}([r \mod q^2] = \beta_\ell) = \bigcup_{\beta_\ell=-Bq}^{Bq}(r = \beta_\ell)$ which is restricted to $\beta_\ell \in S$ by the last condition. Thus, our overall probability is equal to

$$\mathbb{P}\left(\left(\bigcap_{i=0}^{\ell-1}\bigcup_{\beta_i=-B}^{B}([ry_i \mod q] = \beta_i)\right) \cap \left(\bigcup_{\substack{\beta_\ell=-Bq\\ \beta_\ell\in S}}^{Bq}(r = \beta_\ell)\right)\right).$$

Reordering the events gives

$$\mathbb{P}\left(\bigcup_{\beta_0=-B}^{B}\cdots\bigcup_{\beta_{\ell-1}=-B}^{B}\bigcup_{\substack{\beta_\ell=-Bq\\ \beta_\ell\in S}}^{Bq}\left(\bigcap_{i=0}^{\ell-1}([ry_i \mod q] = \beta_i) \cap (r = \beta_\ell)\right)\right).$$

As the events are mutually exclusive, this probability is equal to

$$\sum_{\beta_0=-B}^{B}\cdots\sum_{\beta_{\ell-1}=-B}^{B}\sum_{\substack{\beta_\ell=-Bq\\ \beta_\ell\in S}}^{Bq}\mathbb{P}\left(\bigcap_{i=0}^{\ell-1}([ry_i \mod q] = \beta_i) \cap (r = \beta_\ell)\right).$$

Using Bayes' conditional probability rule followed by Euler's rule of interchanging finite sums, this quantity can be rewritten as:

$$\sum_{\beta_0=-B}^{B}\cdots\sum_{\beta_{\ell-1}=-B}^{B}\sum_{\substack{\beta_\ell=-Bq\\ \beta_\ell\in S}}^{Bq}\mathbb{P}(r = \beta_\ell)\,\mathbb{P}\left(\bigcap_{i=0}^{\ell-1}([ry_i \mod q] = \beta_i)\,\middle|\,(r = \beta_\ell)\right)$$

$$= \sum_{\substack{\beta_\ell=-Bq\\ \beta_\ell\in S}}^{Bq}\mathbb{P}(r = \beta_\ell)\sum_{\beta_0=-B}^{B}\cdots\sum_{\beta_{\ell-1}=-B}^{B}\mathbb{P}\left(\bigcap_{i=0}^{\ell-1}([ry_i \mod q] = \beta_i)\,\middle|\,(r = \beta_\ell)\right)$$

Naturally $\mathbb{P}\left(r=\beta_\ell\right)=\frac{1}{q^2}$ for any β_ℓ. It remains to investigate the value of the rightmost probability. To do so, we rewrite $\beta_\ell = g_\ell \beta'_\ell$ where $g_\ell = \gcd(\beta_\ell, q)$. Then, for fixed $\beta_0, \dots, \beta_{\ell-1}, \beta_\ell$:

$$\begin{aligned}
&\mathbb{P}\left(\bigcap_{i=0}^{\ell-1}([ry_i \mod q]=\beta_i) \,\middle|\, (r=\beta_\ell)\right)\\
=&\mathbb{P}\left(\bigcap_{i=0}^{\ell-1}([\beta_\ell y_i \mod q]=\beta_i)\right)\\
=&\mathbb{P}\left(\bigcap_{i=0}^{\ell-1}([g_\ell\beta'_\ell y_i \mod q]=\beta_i)\right)
\end{aligned}$$

The events in this probability will only be satisfiable if β_i is a multiple of g_ℓ, say $\beta_i = \beta'_i g_\ell$. Thus, our cumulative probability rewrites as

$$\begin{aligned}
&\sum_{\substack{\beta_\ell=-Bq\\ \beta_\ell\in S}}^{Bq} \frac{1}{q^2} \sum_{\beta'_0=-\lfloor B/g_\ell\rfloor}^{\lfloor B/g_\ell\rfloor} \cdots \sum_{\beta'_{\ell-1}=-\lfloor B/g_\ell\rfloor}^{\lfloor B/g_\ell\rfloor} \mathbb{P}\left(\bigcap_{i=0}^{\ell-1}([g_\ell\beta'_\ell y_i \mod q]=\beta'_i g_\ell)\right)\\
=&\sum_{\substack{\beta_\ell=-Bq\\ \beta_\ell\in S}}^{Bq} \frac{1}{q^2} \sum_{\beta'_0=-\lfloor B/g_\ell\rfloor}^{\lfloor B/g_\ell\rfloor} \cdots \sum_{\beta'_{\ell-1}=-\lfloor B/g_\ell\rfloor}^{\lfloor B/g_\ell\rfloor} \mathbb{P}\left(\bigcap_{i=0}^{\ell-1}([\beta'_\ell y_i \mod \frac{q}{g_\ell}]=\beta'_i)\right)\\
=&\sum_{\substack{\beta_\ell=-Bq\\ \beta_\ell\in S}}^{Bq} \frac{1}{q^2} \sum_{\beta'_0=-\lfloor B/g_\ell\rfloor}^{\lfloor B/g_\ell\rfloor} \cdots \sum_{\beta'_{\ell-1}=-\lfloor B/g_\ell\rfloor}^{\lfloor B/g_\ell\rfloor} \mathbb{P}\left(\bigcap_{i=0}^{\ell-1}([y_i \mod \frac{q}{g_\ell}]=[\beta'_i\beta'^{-1}_\ell \mod \frac{q}{g_\ell}])\right)
\end{aligned}$$

where we used the fact that $g_\ell = \gcd(\beta_\ell, q)$ which implies that β'_ℓ is invertible modulo $\frac{q}{g_\ell}$. It is now clear that the remaining events are independent as they only depend on y_i. Thus

$$\begin{aligned}
&\mathbb{P}\left(\bigcap_{i=0}^{\ell-1}([y_i \mod \frac{q}{g_\ell}]=[\beta'_i\beta'^{-1}_\ell \mod \frac{q}{g_\ell}])\right)\\
=&\prod_{i=0}^{\ell-1}\mathbb{P}\left([y_i \mod \frac{q}{g_\ell}]=[\beta'_i\beta'^{-1}_\ell \mod \frac{q}{g_\ell}]\right)\\
=&\left(\frac{1}{\frac{q}{g_\ell}}\right)^\ell\\
=&\left(\frac{g_\ell}{q}\right)^\ell.
\end{aligned}$$

Thereby, the cumulative probability is given by

$$\sum_{\substack{\beta_\ell=-Bq\\ \beta_\ell\in S}}^{Bq} \frac{\ell(2\lfloor B/g_\ell\rfloor+1)}{q^2}\left(\frac{g_\ell}{q}\right)^\ell.$$

□

B The MiNTRU assumption

In this section, we (re-)define the matrix inhomogeneous NTRU (MiNTRU) assumption, we describe some variants and outline its use.

B.1 The MiNTRU assumption

The MiNTRU problem has initially been introduced in [6, Section 4.1] in formula (4).

Definition 4 (MiNTRU distribution). *Let q be any modulus, $\ell = \lceil \log(q) \rceil$, $m = n(\ell + 1)$, $\mathbf{G} = [0|\mathbf{I}|2\mathbf{I}|...|2^{\ell-1}\mathbf{I}] \in \mathbb{Z}^{n \times m}$ an extended gadget matrix, and χ a symmetric error distribution over $\mathbb{Z}$ producing with overwhelming probability elements with norm $\ll q$. Define the MiNTRU distribution with these parameters to be obtained by the following sampling process:*

$$\mathsf{MiNTRU} = \begin{Bmatrix} \mathbf{S} \leftarrow (\mathbb{Z}/q\mathbb{Z})^{n\times n}_{invertible} \\ \mathbf{E} \leftarrow \chi^{n\times m} \\ \mathbf{A} := \mathbf{S}^{-1} \times (\mathbf{G} - \mathbf{E}) \mod q \end{Bmatrix} \tag{21}$$

and denote any such matrix $\mathbf{A}$ by MiNTRU matrix.

Definition 5 (MiNTRU search problem). *Given a MiNTRU matrix $\mathbf{A}$ and a modulus q, the MiNTRU search problem consists in finding the hidden secret matrix $\mathbf{S}$. The MiNTRU search assumption predicts that $\mathbf{S}$ can only be determined with negligible probability.*

Definition 6 (MiNTRU decision problem). *Given a matrix $\mathbf{X}$ and a modulus q, the MiNTRU decision problem consists in distinguishing whether the matrix has been sampled using the MiNTRU distribution or the uniform distribution over $(\mathbb{Z}/q\mathbb{Z})^{n\times m}$. The MiNTRU decision assumption predicts that such a distinction can only be made with negligible probability.*

We highlight that in [6] only the decision problem, has been defined. However, in practice, the search variant may be used. Furthermore, in our presentation, we dropped the apostrophes used in [6] to distinguish the above problem from the so called *small secret* MiNTRU.

B.2 Further remarks

Hereinafter, we will point out some particular points of the definitions:

1. The MiNTRU definition gives no limitation for the modulus q, but in practice, q is chosen to be prime.
2. The MiNTRU definition does not fix the underlying error distribution χ, but in practice it may be considered to be the discrete Gaussian distribution with variance $\sigma_\chi = O(\sqrt{q})$ (following a suggestion of [6]).

3. The MiNTRU definition gives no limitation for the matrix dimension n, but in practice $n = O(q^{1/4})$.
4. One may define other variants of the MiNTRU problem by removing the first few $n \times n$ blocks of a MiNTRU matrix. For example, the *small secret* variant in [6] is obtained by removing the first $n \times n$ block encoding the 0 block of the gadget matrix and sampling the secret matrix $\mathbf{S}$ from the error distribution $\chi_{invertible}^{n \times n}$.

B.3 Applications

For now, the MiNTRU assumption has only been used once in the literature, namely in its original formalization paper [6]. The authors use the MiNTRU assumption to prove semantic security of a new homomorphic encryption scheme for finite automata. We will fast revise their construction.

Homomorphic encryption and non-deterministic finite automata The core idea of homomorphic encryption (HE) [21] is to enable computation over encrypted data. A particular case may be the evaluation of encrypted non-deterministic finite automata (NFA) [18] which can be seen as a particular matrix product. Indeed, without diving too far into the literature of homomorphic encryption and finite automata, a homomorphic encryption scheme for non-deterministic finite automata should allow to compute $\left(\prod_{i=k}^{1} \mathbf{M}_i\right) \cdot \mathbf{v}$ over a finite ring of integers[7], where the matrices $\mathbf{M}_i$ and $\mathbf{v}$ are encrypted. [6] constructs such a scheme as follows.

Their basic encryption process To encrypt a message matrix $\mathbf{M} \in (\mathbb{Z}/q\mathbb{Z})^{n \times n}$, they first sample a secret matrix $\mathbf{S} \in (\mathbb{Z}/q\mathbb{Z})_{invertible}^{n \times n}$ and an error matrix $\mathbf{E} \leftarrow \chi^{n \times m}$ with $m = n \times (\ell + 1)$, then they consider the gadget matrix $\mathbf{G} = [0|\mathbf{I}|2\mathbf{I}|...|2^{\ell-1}\mathbf{I}] \in \mathbb{Z}^{n \times m}$ and finally set $\mathbf{C} := \mathbf{S}^{-1} \times (\mathbf{M} \times \mathbf{G} + \mathbf{E}) \mod q$. Decryption is achieved by computing $\mathbf{S} \times \mathbf{C} - \mathbf{E} := \mathbf{M} \times \mathbf{G}$ and recovering $\mathbf{M}$ using a known trapdoor of the gadget matrix $\mathbf{G}$. Similarly, a vector $\mathbf{v} \in (\mathbb{Z}/q\mathbb{Z})^n$ is encrypted by $\mathbf{c} := \mathbf{S}^{-1} \times (\mathbf{v} + \mathbf{e})$ where $\mathbf{e} \leftarrow \chi^n$.

Chained encryption for homomorphic evaluation To finally enable the homomorphic evaluation of an encrypted product $\left(\prod_{i=k}^{1} \mathbf{M}_i\right) \cdot \mathbf{v}$, the authors propose a specific recursive encryption chain. In detail, first $k+1$ secret keys $(\mathbf{S}_i, \mathbf{E}_i)$ $(i \in \{0, \dots, k\})$ are sampled, then $\mathbf{v}$ is encrypted as $\mathbf{c} := \mathbf{S}_0^{-1} \times (\beta\mathbf{v} + \mathbf{e})$ mod q and finally the matrices $\mathbf{M}_i$ are recursively encrypted by $\mathbf{C}_i := \mathbf{S}^{-1} \times (\mathbf{M}_i \times \mathbf{S}_{i-1} \times \mathbf{G} + \mathbf{E}_i) \mod q$ for all $i \in \{1, \dots, k\}$. Homomorphic evaluation is then performed by setting $\mathbf{c}_0 := \mathbf{c}$ and recursively computing $\mathbf{c}_i := \mathbf{C}_i \times \mathbf{G}^{-1}(\mathbf{c}_{i-1})$ for all $i \in \{1, \dots, k\}$ where $\mathbf{G}^{-1}(c_i)$ denotes a discrete Gaussian vector such that $\mathbf{G} \times \mathbf{G}^{-1}(\mathbf{c}_i) = \mathbf{c}_i \mod q$. The decryption of the final ciphertext $\mathbf{c}_k$ corresponds to the desired output $\left(\prod_{i=k}^{1} \mathbf{M}_i\right) \cdot \mathbf{v}$.

[7] Note that the product runs downwards, i.e., from k to 1.

Security of the new scheme It is trivial to observe that semantic security of their homomorphic encryption scheme is guaranteed if the basic encryption procedure is semantic secure. The basic encryption procedure in turn seems to link to the MiNTRU assumption. Indeed, an encrypted matrix $\mathbf{C} := \mathbf{S}^{-1} \times (\mathbf{M} \times \mathbf{G} + \mathbf{E}) \mod q$ consist almost in a MiNTRU matrix $\mathbf{A} := \mathbf{S}^{-1} \times (\mathbf{G} + \mathbf{E}) \mod q$, except for the additional factor $\mathbf{M}$. The formal security reduction is then obtained in two steps: First, pseudorandomness of the small secret MiNTRU variant is deduced from the usual MiNTRU. Then, semantic security of the encryption procedure with a slightly distorted error distribution $\psi[\mathbf{E}, \mathbf{M} \times \mathbf{G}] := \{\mathbf{R} \leftarrow \mathbf{G}^{-1}(\mathbf{M} \times \mathbf{G}),\ \text{output } \mathbf{E} \times \mathbf{R} \mod q\}$ is obtained from the pseudorandomness of the small secret MiNTRU variant. We remark that this error distribution does not correspond to the actual low norm distribution of the encryption procedure but that it produces far larger entropic noise and is so harder to filter out. To further strengthen the MiNTRU assumption (and so semantic security of their scheme), pseudorandomness of MiNTRU with error distribution $\chi^{n \times m} \times \mathbf{B}^{-1}(\mathbf{G})$ is deduced from the pseudorandomness of the n-secret LWE with trapdoor oracle to $\mathbf{B}$. Unfortunately, trapdoor oracle access for $\mathbf{B}$ of the n-secret LWE, whose distribution is defined by $\{(\mathbf{A}, \mathbf{B} := \mathbf{S} \times \mathbf{A} + \mathbf{E}) \ :\ \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{S} \leftarrow \mathbb{Z}_q^{n \times n}, \mathbf{E} \leftarrow \chi^{n \times m}\}$, is a non-standard assumption and its pseudorandomness requires further investigation. Moreover, the given reduction relies again on a small alteration of the error distribution, namely $\chi^{n \times m} \times \mathbf{B}^{-1}(\mathbf{G})$, which may not correspond to the low norm error distribution that is expected in the MiNTRU setting.

B.4 Our contribution

We are going to prove that (at least for small dimensions n) the decision variant of the MiNTRU assumption is not safe and so the MiNTRU distribution is not pseudorandom in that range. Independent of the chosen modulus and the exact low norm error distribution, our lattice attack will distinguish with noticeable probability between a random matrix and a synthetically constructed one.

Our attack can be seen as a matrix generalization of our first iNTRU attack. It will slightly modify the challenge matrix and construct a specific lattice that in view of a synthetic matrix contains a short vector with entries corresponding to low norm error elements only, but in the presence of a random challenge matrix follows a specific heuristic (close to the Gaussian heuristic). Thereby, the distinction of matrices can be made by comparing the shortest vector to the heuristic.

C Attacking the MiNTRU assumption

In this section, we describe a lattice attack against the decision variant of the MiNTRU assumption. Our development is based on full length challenge matrices but the method will, after small changes, also work for shortened matrices. The attack is inspired on our first iNTRU attack. An attack based on our second iNTRU attack is possible but generally not needed (see Remark 1 below).

C.1 Our lattice and its properties

Let $\mathbf{X}$ be a challenge matrix corresponding either to the uniform or the MiNTRU distribution.

Lattice construction First, we decompose our challenge matrix into $(\ell+1)$ individual $n \times n$ matrices $\mathbf{X}_0, \ldots \mathbf{X}_\ell$ such that

$$\mathbf{X} = [\mathbf{X}_0 | \ldots | \mathbf{X}_\ell]. \tag{22}$$

Then, we slightly modify these matrices and set

$$\mathbf{Y}_0 := \mathbf{X}_0 \mod q \;\text{ and }\; \mathbf{Y}_i = 2\mathbf{X}_i - \mathbf{X}_{i+1} \mod q \quad \forall i \in \{1, \ldots, \ell-1\}. \tag{23}$$

Let $t \in \{0, \ldots, \ell-1\}$ be an index such that the matrix $\mathbf{Y}_t$ is invertible modulo q. If no such index exists, the subsequent development will not work.

Remark 1. Let the prime decomposition of q be $q = \prod_{i=1}^{k} q_i^{\alpha_i}$. Then, a random $n \times n$ matrix is invertible modulo q if and only if it is so modulo $q_i^{\alpha_i}$ for all $i \in \{1, \ldots, k\}$ which holds for a single i with probability $\prod_{j=1}^{n}(1 - q_i^{-j\alpha_i})$. Thus, the probability that our first attack cannot be used is $\left(1 - \prod_{i=1}^{k}\prod_{j=1}^{n}(1 - q_i^{-j\alpha_i})\right)^\ell$ and so our first attack should almost always work; in the improbable case it does not, one can generalize our second iNTRU attack being independent of invertible elements.

We proceed by setting

$$\mathbf{Z}_i := \mathbf{Y}_t^{-1}\mathbf{Y}_i \mod q \quad \forall i \in \{0, \ldots, \ell-1\} \tag{24}$$

where $\mathbf{Z}_t = \mathbf{I}$ is the $n \times n$ identity matrix.
We define the $\ell n \times \ell n$ q-ary row lattice:

$$\Lambda = \mathcal{L}\begin{pmatrix} \mathbf{Z}_0 & \ldots & \mathbf{Z}_{t-1} & \mathbf{I} & \mathbf{Z}_{t+1} & \ldots & \mathbf{Z}_{\ell-1} \\ q\mathbf{I} & \ldots & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & \ldots & q\mathbf{I} & 0 & 0 & \ldots & 0 \\ 0 & \ldots & 0 & 0 & q\mathbf{I} & \ldots & 0 \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & 0 & 0 & \ldots & q\mathbf{I} \end{pmatrix} \tag{25}$$

Remark 2. One may be tempted to simply reduce to the one dimensional case by using the determinant function, in other words, to set $z_i = \det(\mathbf{Z}_i)$, and then continuing along the lines of the first iNTRU attack. One would expect that a random matrix $\mathbf{X}$ leads to random z_i and a synthetically constructed matrix $\mathbf{A}$ to internally linked z_i. Indeed, symbolically setting $s = \det(\mathbf{S}^{-1})$ and $e'_i = \det(-2\mathbf{E}_i + \mathbf{E}_i)$ reveals this internal structure. Unfortunately, the compression of the error terms using the determinant blows the final error up and in order to work out, the individual error terms would need to be upper bounded in absolute value by $\min\left\{\frac{q^{3/4n}}{\sqrt{n}\,\ell^{1/2n}}, \frac{1}{\sqrt{n}(8\pi e)^{1/2n}} q^{\frac{3\ell-4}{4\ell n}}\right\}$ which is only satisfied in practice for small n. Thus, it is suggested to use another approach.

Remark 3. The lattice in Equation (25) is a direct generalization of Equation (4). Actually, each entry of Equation (4) is expanded into a full matrix to obtain Equation (25). Nonetheless, the underlying idea is exactly the same.

Heuristic Contrary to our original iNTRU attack, we can not use heuristic **H** as instead of a simple vector, we use a matrix to define our q-ary lattice. We recall that by [22, Lemma 2.18] for fixed prime q and $m \geq n$, and for a randomly sampled matrix $\mathbf{A} \in (\mathbb{Z}/q\mathbb{Z})^{n\times m}$, the first lattice minimum of the q-ary lattice $\Lambda_q(\mathbf{A})$ is lower bounded by $\min\left\{q,\ \sqrt{\frac{m}{8\pi e}} q^{\frac{m-n}{m}}\right\}$ with probability greater than $1 - 2^{-m}$ and that $\sqrt{\frac{m}{8\pi e}} q^{\frac{m-n}{m}}$ is at most as large as half the Gaussian heuristic.

C.2 Case of a random tuple

Assume that our initial challenge matrix $\mathbf{X}$ was sampled uniformly at random. Then, our constructed matrices $\mathbf{X}_i$, $\mathbf{Y}_i$ and $\mathbf{Z}_i$ will still follow the uniform distribution (except for $\mathbf{Z}_t = \mathbf{I}$). Albeit this does not fully correspond to the setup required by the above heuristic as some part of the generating matrix is not random (but equal to the identity matrix $\mathbf{I}$) and q may not be prime, we assume that with noticeable probability, the first lattice minimum satisfies

$$\lambda_1(\Lambda) \geq \min\left\{q,\ \sqrt{\frac{\ell n}{8\pi e}} q^{\frac{\ell-1}{\ell}}\right\}. \qquad \textbf{(H')}$$

C.3 Case of a synthetic tuple

Assume next that the initial challenge matrix $\mathbf{X}$ has been synthetically constructed following the MiNTRU distribution. We will show that in this case the lattice contains a non-trivial short vector being magnitudes smaller than the expected heuristic.

First observation We recall that a synthetically constructed matrix $\mathbf{A}$ following the MiNTRU distribution satisfies $\mathbf{A} = \mathbf{S}^{-1} \times (\mathbf{G} - \mathbf{E}) \mod q$ and so letting

$\mathbf{A} = [\mathbf{A}_0|\ldots|\mathbf{A}_\ell]$ gives

$$\begin{aligned} \mathbf{A}_0 &= \mathbf{S}^{-1}(\mathbf{0} - \mathbf{E}_0) \mod q \quad \text{and} \\ \mathbf{A}_i &= \mathbf{S}^{-1}(2^{i-1}\mathbf{I} - \mathbf{E}_i) \mod q \quad \forall i \in \{1, \ldots, \ell\} \end{aligned} \tag{26}$$

where $\mathbf{E}_0, \ldots, \mathbf{E}_\ell$ denote random error matrices whose entries are sampled from the symmetric error distribution χ producing with overwhelming probability small elements. Thereby,

$$\begin{aligned} \mathbf{Y}_0 &= -\mathbf{S}^{-1}\mathbf{E}_0 = \mathbf{S}^{-1}\mathbf{E}'_0 \mod q \quad \text{and} \\ \mathbf{Y}_i &= \mathbf{S}^{-1}(-2\mathbf{E}_i + \mathbf{E}_{i+1}) = \mathbf{S}^{-1}\mathbf{E}'_i \mod q \quad \forall i \in \{1, \ldots, \ell-1\} \end{aligned} \tag{27}$$

where the entries of $\mathbf{E}'_i$ are still quite small for all $i \in \{0, \ldots, \ell-1\}$. More precisely, they follow the distribution χ' with $\mu_{\chi'} = 0$ and $\sigma' = \sqrt{3}\sigma$. In particular, we conclude that since χ produces with overwhelming probability elements with absolute value $\ll q$, so does χ' and the entries are quite small when compared to the modulus q.

Second observation Continuing to outline the effect of our variable changes leads to the conclusion that

$$\mathbf{Z}_i = \mathbf{E}'^{-1}_t\mathbf{E}'_i \mod q \quad \forall i \in \{0, \ldots, \ell-1\}. \tag{28}$$

Third observation By construction, it is now clear that

$$\mathbf{E}'_t\mathbf{Z}_i = \mathbf{E}'_i \mod q \quad \forall i \in \{0, \ldots, \ell-1\} \tag{29}$$

and so, denoting the j-th row of $\mathbf{E}'_i$ by $\mathbf{e}_i^{(j)}$ yields

$$\mathbf{e}_t^{(j)}\mathbf{Z}_i = \mathbf{e}_i^{(j)} \mod q \quad \forall i \in \{0, \ldots, \ell-1\} \text{ and } \forall j \in \{1, \ldots, n\}. \tag{30}$$

The shortest lattice vector Interestingly, our observations imply that our lattice contain the n linearly independent vectors

$$\mathbf{e}^{(j)} = (\mathbf{e}_0^{(j)}|\ldots|\mathbf{e}_\ell^{(j)}) \quad \forall j \in \{1, \ldots, n\} \tag{31}$$

obtained through a linear combination corresponding to $\mathbf{e}_t^{(j)}$ followed by a suitable "modulo q reduction". We are going to show that these vectors will be almost surely smaller than the Gaussian heuristic. Therefore, first assume that the error entries of the error matrices $\mathbf{E}'_i$ (or only of a specific row of the error matrices) are upper bounded by some constant $K > 0$. Then, the size of $\mathbf{e}^{(j)}$ is upper bounded by

$$\|\mathbf{e}^{(j)}\|_2 \leq \sqrt{\ell n K^2} \leq \sqrt{\ell n}\, K. \tag{32}$$

Proposition 5. *If the error entries of the error matrices* $\mathbf{E}'_i$ *are upper bounded by* $\min\left\{\frac{q}{\sqrt{\ell n}}, \frac{1}{\sqrt{8\pi e}} q^{(\ell-1)/\ell}\right\}$*, then the target vectors* $\mathbf{e}^{(j)} = (\mathbf{e}_0^{(j)} | \dots | \mathbf{e}_\ell^{(j)})$ *are shorter than* $\min\left\{q, \sqrt{\frac{\ell n}{8\pi e}} q^{\frac{\ell-1}{\ell}}\right\}$ *for each* $j \in \{1, \dots, n\}$.

Proof. Replace K in Equation (32) by the predicted values. □

As in practice the error terms are $O(\sqrt{q})$, the size condition is almost always satisfied. The probability that a completely randomly sampled ℓn-vector would be of this size is lower than $\left(\frac{2\sqrt{\ell n}K+1}{q}\right)^{\ell n}$ which is rapidly decreasing in K.

Lattice reduction Unfortunately, the large lattice dimension avoids obtaining $\mathbf{e}^{(j)}$ through usual LLL reduction. Indeed, LLL with $\delta = \frac{63}{64}$ will output $\mathbf{w}_1$ such that $\|\mathbf{w}_1\|_2 \le q^{\frac{n}{4}} 2^{\frac{n-1}{4}} \sqrt{\ell n} K$ which is too loose for a theoretical conclusion.[8]

Through BKZ reduction, the approximation factor could be strongly improved and for small enough block size one may even get a polynomial runtime (see [14, Theorem 2]). Thus, slightly larger dimensions may be treated. To be precise, BKZ with block size β achieves $\|\mathbf{b}_1\|_2 \le \gamma_\beta^{\frac{\ell n}{\beta-1}} \lambda_1$ for its first reduced vector $\mathbf{b}_1$ where γ_β denotes the Hermite constant (better results may be achieved with more recent results). [9] To put this into context, consider the bloc size $\beta = 24$ for which BKZ still finishes in a reasonable amount of time. Then, $\|\mathbf{b}_1\|_2 \le 2^{\frac{2\ell n}{23}} \lambda_1 \le 2^{\frac{2\ell n}{23}} \sqrt{\ell n} K \le (2q)^{\frac{2n}{23}} \sqrt{\ell n} K$ indicating that our method would still work for sufficiently small n or small noise. Unfortunately, this improvement is still not be exact enough for a theoretical conclusion in large dimensions.

Fortunately, lattice reduction algorithms often perform better in practice than in theory and so our method may still be applied for larger lattice dimensions on a speculative basis. Nonetheless, we emphasise that theoretical success can only be shown through exact shortest vector problem solvers whose run-time is exponential in the lattice dimension making the attack impractical.

C.4 Conclusion

We conclude that:

1. In case of a random tuple, the shortest lattice vector can be expected to be lower bounded by $\min\left\{q, \sqrt{\frac{\ell n}{8\pi e}} q^{\frac{\ell-1}{\ell}}\right\}$.
2. In case of a synthetic tuple, the shortest lattice vector is with high probability smaller than $\min\left\{q, \sqrt{\frac{\ell n}{8\pi e}} q^{\frac{\ell-1}{\ell}}\right\}$.

[8] Note that if $n = 1$, then we recover the same bound as in Equation (12).

[9] See for example *Predicting Lattice Results* by Gama and Nguyen in *Advances in Cryptology - Eurocrypt 2008* or *Extended Lattice Reduction Experiments using the BKZ algorithm* by Schneider and Buchmann in *Sicherheit 2010* and other related references.

Hence, we conclude that we could distinguish with noticeable probability between a randomly sampled challenge matrix and a synthetically constructed one by simply comparing the length of the first lattice minimum (or at least a short enough approximation of it) with $\min\left\{q,\ \sqrt{\frac{\ell n}{8\pi e}}q^{\frac{\ell-1}{\ell}}\right\}$. Unfortunately, our development is not strong enough to theoretically prove the well functioning of our distinguisher when relying on polynomial lattice reduction techniques only. The corresponding SAGEMATH source codes can be found in Appendix F.

C.5 Further remarks

Remark 1: Recovering the secret We note that if a MiNTRU matrix is detected, then it is tempting to pad the n shortest lattice vectors together in order to recover the error matrices $\mathbf{E}'_i$ and through linear algebra the original error matrices $\mathbf{E}_i$ and simultaneously the secret matrix $\mathbf{S}$. Unfortunately such a procedure will not work in general. First, the shortest vectors may not correspond to the error vectors (shorter vectors may be obtained through linear combinations of them). Second, lattice reduction may not achieve the shortest vectors. Third, the matrices would only be determined up to a random permutation. Thus, our method is not suitable to solve the search variant of the MiNTRU problem.

Remark 2: Minor improvement To reduce computational power, one needs to decrease the lattice dimension. One may achieve this by not considering ℓ copies of $\mathbf{Z}_i$ but only b copies for some $1 \leq b \leq \ell$. Indeed, as long as the error terms are small enough (c.f. Proposition 5), the construction works out. This can also be formally proven by simply replacing ℓ by b in the whole section. We note that it doesn't matter which copies of $\mathbf{Z}_i$ are chosen as long as $\mathbf{Z}_t$ is part of the chosen set. $\mathbf{Z}_t$ plays a crucial role in the construction as it guarantees that only small linear combinations will be chosen. The other $b-1$ $\mathbf{Z}_i$'s may be chosen randomly. In practice, $b = 5$ is sufficient for $\sigma_\chi = 2\sqrt{q}$.

Furthermore, we are not restricted to choose complete copies of $\mathbf{Z}_i$, but we may pick out individual columns. A mix between columns of different copies will not interfere with the functioning of our method. However, the analysis will change (ℓn needs to be replaced by the suitable value). Nonetheless, as long as $\mathbf{Z}_t$ is chosen completely, the method will work out fine.

The same improvement may be implemented for iNTRU. Unfortunately, this improvement does not solve the true bottleneck of MiNTRU which is the size of $n = O(q^{\frac{1}{4}})$ implying a lattice dimension that is too large for practical attacks.

Remark 3: Dependence of the matrices When developing the matrices $\mathbf{Z}_i = \mathbf{Y}_t^{-1}\mathbf{Y}_i$, one may think to repeat the same construction for another index t' in order to obtain linearly independent matrices $\mathbf{Z}'_i = \mathbf{Y}_{t'}^{-1}\mathbf{Y}_i$. Unfortunately this will not be the case as $\mathbf{Z}'_i = \mathbf{Z}'_t\mathbf{Z}_i$ and so no more information can be generated through such an index change.

D Comparison with multi-secret LWE

In this section, we rapidly compare the (matrix) inhomogeneous NTRU (iNTRU) assumption, with the multi-secret Learning-with-Errors (LWE) assumption. In line with the main objective of our article, we concentrate on the integer case only.

D.1 The multi-secret LWE problem

The LWE problem has initially been introduced in [19]. Its multi-secret variant may be formalized as follows [17].

Definition 7 (Multi-secret LWE distribution). *Let q be any modulus, $m \in \mathbb{Z}_{\geq n}$, and χ an error distribution over $\mathbb{Z}$ producing with overwhelming probability elements with norm $\ll q$. Define the multi-secret* LWE *distribution with these parameters to be obtained by the following sampling process:*

$$\mathsf{LWE} = \left\{ \begin{aligned} &\mathbf{S} \leftarrow (\mathbb{Z}/q\mathbb{Z})^{n\times n} \\ &\mathbf{E} \leftarrow \chi^{n\times m} \\ &\mathbf{A} \leftarrow (\mathbb{Z}/q\mathbb{Z})^{n\times m} \\ &\mathbf{B} := \mathbf{S}\times\mathbf{A}+\mathbf{E} \mod q \end{aligned} \right\} \tag{33}$$

and denote any pair $(\mathbf{A}, \mathbf{B})$ as multi-secret LWE *pair.*

Definition 8 (Multi-secret LWE search problem). *Given a multi-secret* LWE *pair $(\mathbf{A}, \mathbf{B})$ and a modulus q, the multi-secret* LWE *search problem consists in finding the hidden secret matrix $\mathbf{S}$. The multi-secret* LWE *search assumption predicts that $\mathbf{S}$ can only be determined with negligible probability.*

Definition 9 (Multi-secret LWE decision problem). *Given a pair $(\mathbf{X}, \mathbf{Y})$ and a modulus q, the multi-secret* LWE *decision problem consists in distinguishing whether the pair has been sampled using the multi-secret* LWE *distribution or the uniform distribution over $(\mathbb{Z}/q\mathbb{Z})^{n\times m} \times (\mathbb{Z}/q\mathbb{Z})^{n\times m}$. The multi-secret* LWE *decision assumption predicts that such a distinction can only be made with negligible probability.*

D.2 Hardness of multi-secret LWE

The hardness of the LWE problems have been well-studied and many surprising properties such as self-reducibility [19] proving hardness on average have been found. Furthermore, explicit reductions to known supposedly hard lattice problems have been outlined [15, 19]. Those results, and the fact that after 16 years, no efficient general attack has been mounted pushes us to believe that the LWE problems are hard.

However, not all LWE instances are equally hard. Indeed, their hardness depends on the chosen error scale χ. For example, neither the decision, nor the search variant will be solvable for completely random errors, but are surmountable for Bernoulli distributions. The number of samples obtained for an LWE instance has only a small effect on the hardness as one may produce, through linear (re-)combinations, as many valid samples as required [20].

The multi-secret LWE can be directly reduced to the usual LWE problem and the additional information leakage (of the multiple secrets) is supposed to not make the problem easier [17, Lemma 6.2]. Thereby, the multi-secret variant seems to be as hard as the usual LWE.

D.3 Comparison

When comparing the (matrix) inhomogeneous NTRU problems with the multi-secret learning with error problem, one may spot a crucial difference. Whereas in the former case the error is added and then the secret multiplied, the latter switches those two operations. Consisting in a syntactic change of operations, one may still transform one problem into the other by adjusting the error distribution. This adjustment provokes a distorted blow-up leading to heavier entropic noise (see the security part in Appendix B.3). The gadget matrix in the MiNTRU case and the inversion of $\mathbf{S}$ do apparently not have a considerable effect on the hardness, but, intuitively, they give some more information about MiNTRU matrices.

The main comparison point of the multi-secret LWE and MiNTRU stems from our own analysis. Indeed, we explicitly proved that an elementary lattice attack is sufficient to break low dimensional MiNTRU problems (for dimension $n = 1$ it consists in the integer iNTRU problem). On the contrary, the multi-secret LWE seems not to suffer from such a vulnerability. Indeed, m is at most polynomial in n implying that the original decision variant of the LWE is hard, which in turn guarantees hardness of its multi-secret variant.

This difference may or may not have an effect on the security of high degree polynomial rings (for iNTRU) or for high dimensional matrices (for MiNTRU), but it surely shows the existence of different secure parameter bounds when compared to the multi-secret LWE. After all, multiplying and adding is not the same as adding and multiplying!

E Source Code for the iNTRU Attack

```python
#This file is devoted to the codes referenced in Section 3-5 of the article "On the (M)iNTRU assumption in the integer case".

#All references are made with respect to the mentioned article.


# ---------------------------------------------------------------------
# ---------------------------------------------------------------------

#Function Declarations:
#----------------------


#First, we implement an iNTRU tuple sampler (Section 3):

def iNTRU(q,sigma):
    """
    Upon reception of a modulus q and a variance value sigma the function iNTRU samples an iNTRU tuple of length ceil(log(q,2)) using error terms from the 0-centered discrete Gaussian distribution with variance sigma (see Equation (1)).
    """

    # We sample the invertible secret value s:
    s=ZZ(randint(-floor((q-1)/2),floor((q-1)/2)))
    while gcd(s,q)>1:
        s=ZZ(randint(-floor((q-1)/2),floor((q-1)/2)))

    # We sample the error terms:
    from sage.stats.distributions.discrete_gaussian_integer import DiscreteGaussianDistributionIntegerSampler
    D = DiscreteGaussianDistributionIntegerSampler(sigma=sigma)
    E=[]
    for i in range(ceil(log(q,2))+1):
        E.append(ZZ(D()))

    # We construct the iNTRU tuple:
    L=[(E[0]*Integer(Integer(s).inverse_mod(q)))%q]
    for i in range(ceil(log(q,2))):
        L.append(((2^(i)-E[i+1])*Integer(Integer(s).inverse_mod(q)))%q)

    # For comparison purposes, we return the iNTRU tuple L as well as the corresponding secret s and the error terms:
    return (L,E,s)



#Next, we implement the uniform tuple sampler (Section 3):

def uni(q):
    """
    Upon reception of a modulus q the function uni samples a tuple of length ceil(log(q,2)) from the 0-centered uniform distribution over Z/qZ.
    """

    # We sample a vector uniformly at random and return it:
    U=[]
    for i in range(ceil(log(q,2))+1):
        U.append(ZZ(randint(-floor((q-1)/2),floor((q-1)/2))))
    return U



#We implement our first distinguisher (Section 4):

def distinguisher1(q,X):
    """
    Upon reception of a modulus q and a corresponding challenge tuple X distinguisher1 generates the lattice in Equation (4) and approximates the shortest vector. It outputs -1 if the method is not applicable. It outputs (0,0) if it suspects a random challenge tuple and (1,s) if it suspects a synthetic challenge tuple where s is the expected secret.
    """

    #1) We compute the y_i values from the original tuple (see Equation (2)).
    Y=[X[0]]
    for i in range(len(X)-2):
        Y.append((2*X[i+1]-X[i+2])%q)

    #2) We check whether this method can be applied (i.e., if there exists an invertible element in Y).
    t=-1
    for i in range(len(Y)):
        if gcd(Y[i],q)==1:
            t=i
            break
    if t==-1:
        return -1

    #3) We compute the z_i values (see Equation (3)).
    Z=[]
    for i in range(len(Y)):
        Z.append(((Y[t]%q).inverse_mod(q)*Y[i])%q)

    #4) We construct the lattice in Equation (4).
    Lambda=zero_matrix(ZZ,nrows=len(Z),ncols=len(Z))
    for i in range(len(Z)):
        Lambda[0,i]=Z[i]
        if i<t:
            Lambda[i+1,i]=q
        if i>t:
            Lambda[i,i]=q

    #5) We compute the first LLL reduced vector of Lambda:
    v=Lambda.LLL().row(0)

    #6) We compute our heuristic H for Lambda (see Equation
   (5)):
    H=sqrt(len(Z)/(8*pi*e))*q^((len(Z)-1)/len(Z))

    #7) If the first LLL reduced vector is longer than our
   heuristic or q, we conclude that the challenge tuple was
   a random tuple.
    if v.norm()>H or v.norm()>q:
        return (0,0)

    #8) Otherwise, we conclude that it was a synthetic tuple
   and we try to compute the secret (up to its sign).
    else:
        s=(v[t]*(Y[t]%q).inverse_mod(q))%q
        return (1,s)



#We implement our second distinguisher (Section 5):

def distinguisher2(q,X):
    """
    Upon reception of a modulus q and a corresponding
   challenge tuple X distinguisher2 generates the lattice in
    Equation (14) and approximates the shortest vector. It
   outputs (0,0) if it suspects a random challenge tuple and
    (1,s) if it suspects a synthetic challenge tuple where s
    is the expected secret.
    """

    #1) We compute the y_i values from the original tuple (
   see Equation (13)).
    Y=[X[0]]
    for i in range(len(X)-2):
        Y.append((2*X[i+1]-X[i+2])%q)

    #2) We construct the lattice in Equation (14).
    Lambda=zero_matrix(ZZ,nrows=len(Y)+1,ncols=len(Y)+1)
    Lambda[0,len(Y)]=1
    for i in range(len(Y)):
        Lambda[0,i]=Y[i]*q
        Lambda[i+1,i]=q^2

    #3) We compute the second LLL reduced vector of Lambda:
    v=Lambda.LLL().row(1)

    #4) We compute our heuristic H for Lambda (see Equation
    (15)):
    H=sqrt((len(Y)+1)/(8*pi*e))*q^((2*len(Y))/(len(Y)+1))

    #5) If the second LLL reduced vector is longer than our
   heuristic or q, we conclude that the challenge tuple was
   a random tuple.
    if v.norm()>H or v.norm()>q^2:
        return (0,0)

    #6) Otherwise, we conclude that it was a synthetic tuple
   and we try to compute the secret (up to its sign).
    else:
        s=v[len(Y)]%q
        return (1,s)



#
    --------------------------------------------------------------------------

#
    --------------------------------------------------------------------------

#Toy example:
#------------

#We generate an odd prime:
q=next_prime(10000)

#We construct an iNTRU tuple with variance 2*sqrt(q):
```

```
sigma=2*sqrt(q)
(L,E,s)=iNTRU(q,sigma)

#We generate a random tuple:
U=uni(q)

#We use our first distinguisher:
distinguisher1(q,L) # should return (1,+-s%q)
distinguisher1(q,U) # should return (0,0)

#We use our second distinguisher:
distinguisher2(q,L) # should return (1,+-s%q)
distinguisher2(q,U) # should return (0,0)



#
    --------------------------------------------------------------------------------

#Real world example:
#-------------------
#We set the required parameters:
q=randint(2^60,2^64)
sigma=2*sqrt(q)

#We generate an iNTRU tuple and a random one:
(L,E,s)=iNTRU(q,sigma)
U=uni(q)

#We apply our first distinguisher:
distinguisher1(q,L) # should return (1,+-s%q)
distinguisher1(q,U) # should return (0,0)

#We apply our second distinguisher:
distinguisher2(q,L) # should return (1,+-s%q)
distinguisher2(q,U) # should return (0,0)

#
    --------------------------------------------------------------------------------

#Repeated cryptographic example for distinguisher1:
#--------------------------------------------------

#We test how often iNTRU tuple and random tuples
#are correctly recognized by our distinguisher1.
Positive_passed=0
Positive_failed=0
Negative_passed=0
```

```
Negative_failed=0
Secrets_guessed=0
Test_failed=0
for i in range(100):
    print(Positive_passed,Positive_failed,Negative_passed,
   Negative_failed,Secrets_guessed,Test_failed)
    q=randint(2^40,2^65)
    sigma=2*sqrt(q)

    #We generate an iNTRU tuple and a random one:
    (L,E,s)=iNTRU(q,sigma)
    U=uni(q)

    #We apply our distinguisher:
    R=distinguisher1(q,L)
    if R==-1:
        Test_failed=Test_failed+1
    elif R[0]==1:
        Positive_passed=Positive_passed+1
        if (R[1]%q)==(s%q) or (R[1]%q)==(-s%q):
            Secrets_guessed=Secrets_guessed+1
    else:
        Positive_failed=Positive_failed+1
    RU=distinguisher1(q,U)
    if RU==-1:
        Test_failed=Test_failed+1
    elif RU[0]==0:
        Negative_passed=Negative_passed+1
    else:
        Negative_failed=Negative_failed+1
print(Positive_passed,Positive_failed,Negative_passed,
   Negative_failed,Secrets_guessed,Test_failed)

#The final result should show a tuple (A,B,C,D,E,F) where:
# A denotes the number of correctly recognized iNTRU tuples (
   should be high)
# B denotes the number of wrongly interpreted iNTRU tuples (
   should be low)
# C denotes the number of correctly recognized random tuples
   (should be high)
# D denotes the number of wrongly interpreted random tuples (
   should be low)
# E denotes the number of correctly guessed secrets (should
   be high)
# F denotes the number of failed tests due to the
   invertibility condition (should be low)



```

```
# ---------------------------------------------------------------------------

# Repeated cryptographic example for distinguisher2:
#--------------------------------------------------

#We test how often iNTRU tuple and random tuples are correctly recognized by our distinguisher2.
Positive_passed=0
Positive_failed=0
Negative_passed=0
Negative_failed=0
Secrets_guessed=0
for i in range(100):
    print(Positive_passed,Positive_failed,Negative_passed,Negative_failed,Secrets_guessed)
    q=randint(2^40,2^65)
    sigma=2*sqrt(q)

    #We generate an iNTRU tuple and a random one:
    (L,E,s)=iNTRU(q,sigma)
    U=uni(q)

    #We apply our distinguisher:
    (r0,r1)=distinguisher2(q,L)
    if r0==1:
        Positive_passed=Positive_passed+1
        if (r1%q)==(s%q) or (r1%q)==(-s%q):
            Secrets_guessed=Secrets_guessed+1
    else:
        Positive_failed=Positive_failed+1
    if distinguisher2(q,U)[0]==0:
        Negative_passed=Negative_passed+1
    else:
        Negative_failed=Negative_failed+1
print(Positive_passed,Positive_failed,Negative_passed,Negative_failed,Secrets_guessed)

#The final result should show a tuple (A,B,C,D,E) where:
# A denotes the number of correctly recognized iNTRU tuples (should be high)
# B denotes the number of wrongly interpreted iNTRU tuples (should be low)
# C denotes the number of correctly recognized random tuples (should be high)
# D denotes the number of wrongly interpreted random tuples (should be low)
# E denotes the number of correctly guessed secrets (should be high)
```

```


#
    ---------------------------------------------------------------------------

#Conclusion:
#------------

#In our empirical trials (100 synthetic tuples and 100 random
     tuples for each distinguisher) with the setup as stated
    above:
# 1) All iNTRU tuples have been correctly determined (with
    both distinguishers) and all secrets have been found (
    with both distinguishers). (i.e., B=0 and E=100)
# 2) All random tuples have been correctly determined (with
    both distinguishers). (i.e., D=0)
```

F Source Code for the MiNTRU Attack

```
#This file is devoted to the codes referenced in Appendix B-C of the article "On the (M)iNTRU assumption in the integer case".

#All references are made with respect to the mentioned article.


# ----------------------------------------------------------------------
# ----------------------------------------------------------------------

#Function Declarations:
#----------------------

#First, we implement the MiNTRU matrix sampler (Appendix B):

def MiNTRU(q,n,sigma):
    """
    Upon reception of a modulus q, a matrix dimension n and a variance value sigma the function MiNTRU samples a n x (n*ceil(log(q,2)))  MiNTRU matrix using error terms from the 0-centered discrete Gaussian distribution with variance sigma (see Equation (21)).
    """

    # We sample the secret invertible matrix S:
    S=zero_matrix(Integers(q),nrows=n,ncols=n)
    while not S.is_invertible():
        for i in range(n):
            for j in range(n):
                S[i,j]=randint(-floor((q-1)/2),floor((q-1)/2))

    # We sample the error matrix:
    from sage.stats.distributions.discrete_gaussian_integer import DiscreteGaussianDistributionIntegerSampler
    D = DiscreteGaussianDistributionIntegerSampler(sigma=sigma)
    E=zero_matrix(Integers(q),nrows=n,ncols=n*(ceil(log(q,2))+1))
    for i in range(n):
        for j in range(n*(ceil(log(q,2))+1)):
            E[i,j]=D()

    # We construct the extended gadget matrix:
    G=zero_matrix(Integers(q),nrows=n,ncols=n*(ceil(log(q,2))
   +1))
    for i in range(n):
        for j in range((ceil(log(q,2)))):
            G[i,n*(j+1)+i]=2^(j)

    # We construct the MiNTRU matrix:
    A=S.inverse()*(G-E)

    # For comparison purposes, we return the MiNTRU matrix A
   as well as the corresponding secret matrix S and the
   error matrix:
    return (A,E,S)



#Next, we implement the uniform tuple sampler (Appendix B):

def Muni(q,n):
    """
    Upon reception of an odd integer q and a matrix dimension
     n the function Muni samples a n x (n*ceil(log(q,2)))
   matrix from the 0-centered uniform distribution over Z/qZ
   .
    """

    # We sample a matrix uniformly at random and return it:
    U=zero_matrix(Integers(q),nrows=n,ncols=n*(ceil(log(q,2))
   +1))
    for i in range(n):
        for j in range(n*(ceil(log(q,2))+1)):
            U[i,j]=randint(-floor((q-1)/2),floor((q-1)/2))
    return U


#We build our distinguisher (Appendix C):

def distinguisher(q,X):
    """
    Upon reception of a modulus q and a challenge matrix X,
   distinguisher constructs the lattice in Equation (25) and
    approximates the shortest vector. If the distinguisher
   suspects a synthetic challenge matrix, it outputs 0,
   otherwise, it outputs 1.
    """

    #1) We split the matrix X in n x n matrices X_i (see
   Equation (22)):
    XList=[]
    n=X.nrows()
    ell1=Integer(X.ncols()/X.nrows())
    for k in range(ell1):
        XTemp=zero_matrix(Integers(q),nrows=n,ncols=n)
        for i in range(n):
            for j in range(n):
                XTemp[i,j]=X[i,k*n+j]
        XList.append(XTemp)

    #2) We compute the matrices Y_i (see Equation(23)):
    YList=[XList[0]]
    for i in range(ell1-2):
        YList.append(2*XList[i+1]-XList[i+2])

    #3) We check whether this method can be applied (i.e., if there exists an invertible element in Y).
    t=-1
    for i in range(len(YList)):
        if YList[i].is_invertible():
            t=i
            break
    if t==-1:
        return -1

    #4) We compute the matrices Z_i (see Equation (24)):
    ZList=[]
    for k in range(len(YList)):
        ZList.append(YList[t].inverse()*YList[k])

    #5) We construct the lattice basis in Equation (25):
    Lambda=zero_matrix(ZZ,nrows=len(ZList)*n,ncols=len(ZList)*n)
    for k in range(len(ZList)):
        for i in range(n):
            for j in range(n):
                Lambda[i,k*n+j]=ZList[k][i,j]
            if k<t:
                Lambda[(k+1)*n+i,k*n+i]=q
            if k>t:
                Lambda[k*n+i,k*n+i]=q

    #6) We compute the first LLL reduced vector of Lambda:
    v=Lambda.LLL().row(0)

    #7) We compute our heuristic H' for Lambda (see Equation (H')):
    H=sqrt((len(ZList)*n)/(8*pi*e))*q^((len(ZList)-1)/len(ZList))

    #8) If the first LLL reduced vector is longer than our heuristic or q, we conclude that the challenge tuple was a random tuple.
    if v.norm()>H or v.norm()>=q:
        return 0

    #9) Otherwise, we conclude that it was a synthetic tuple.
    else:
        return 1


#We slightly improve our distinguisher (using C.5.Remark 2):

def distinguisher_short(q,X,b):
    """
    Upon reception of a modulus q, a challenge matrix X and a block count b>2, distinguisher_short constructs the lattice in Equation (25) but only with b blocs and approximates the shortest vector. If there are not sufficiently many blocks to meet b blocks, the distinguisher returns -2. If the distinguisher suspects a synthetic challenge matrix, it outputs 0, otherwise, it outputs 1.
    """

    #1) We split the matrix X in n x n matrices X_i (see Equation (22)):
    XList=[]
    n=X.nrows()
    ell1=Integer(X.ncols()/X.nrows())
    if ell1<b:
        return -2
    for k in range(ell1):
        XTemp=zero_matrix(Integers(q),nrows=n,ncols=n)
        for i in range(n):
            for j in range(n):
                XTemp[i,j]=X[i,k*n+j]
        XList.append(XTemp)

    #2) We compute the matrices Y_i (see Equation (23)):
    YList=[XList[0]]
    for i in range(ell1-2):
        YList.append(2*XList[i+1]-XList[i+2])

    #3) We check whether this method can be applied (i.e., if there exists an invertible element in Y).
    t=-1
    for i in range(len(YList)):
        if YList[i].is_invertible():
            t=i
            break
    if t==-1:
        return -1

    #4) We compute a shortened list of matrices Z_i (see Equation (24)):
    ZList=[]
    ZList.append(YList[t].inverse()*YList[t])
    tnew=0
    if t<b:
        for k in range(b):
            if k!=t:
                ZList.append(YList[t].inverse()*YList[k])
    else:
        for k in range(b-1):
            ZList.append(YList[t].inverse()*YList[k])

    #5) We construct a shortened lattice basis (see Equation (25)):
    Lambda=zero_matrix(ZZ,nrows=b*n,ncols=b*n)
    for k in range(b):
        for i in range(n):
            for j in range(n):
                Lambda[i,k*n+j]=ZList[k][i,j]
            if k>0:
                Lambda[k*n+i,k*n+i]=q

    #6) We compute the first LLL reduced vector of Lambda:
    v=Lambda.LLL().row(0)

    #7) We compute our heuristic H' for Lambda (see Equation (H')):
    H=sqrt((b*n)/(8*pi*e))*q^((b-1)/b)

    #8) If the first LLL reduced vector is longer than our heuristic or q, we conclude that the challenge matrix was a random matrix.
    if v.norm()>H or v.norm()>=q:
        return 0

    #9) Otherwise, we conclude that it was a synthetic tuple.
    else:
        return 1



# ----------------------------------------------------------------------------
```

```
#
    --------------------------------------------------------------------------------

#Toy example:
#------------

#We generate an odd prime:
q=next_prime(10000)

#We set a dimension:
n=5

#We construct a MiNTRU matrix with error variance 2*sigma:
sigma=2*sqrt(q)
(A,E,S)=MiNTRU(q,n,sigma)

#We generate a random tuple:
U=Muni(q,n)

#We use our distinguisher:
distinguisher(q,A) # should return 1
distinguisher(q,U) # should return 0

#We apply our improved distinguisher:
distinguisher_short(q,A,5) # should return 1
distinguisher_short(q,U,5) # should return 0

#
    -----------------------------------------------------------------------------------

#Slightly advanced example:
#-------------------


#We set the required parameters:
q=randint(2^15,2^19)
n=randint(4,16)

#We construct a MiNTRU matrix with error variance 2*sigma:
sigma=2*sqrt(q)
(A,E,S)=MiNTRU(q,n,sigma)

#We generate a random matrix:
U=Muni(q,n)

#We apply our distinguisher:
distinguisher(q,A) # should return 1
distinguisher(q,U) # should return 0
```

```

#-->This is probably the limit for this method without
    improvements.

#We apply our improved distinguisher:
distinguisher_short(q,A,5) # should return 1
distinguisher_short(q,U,5) # should return 0


#
    -------------------------------------------------------------------------------

#Weak cryptographic example (may take some time):
#-------------------

#We set the required parameters:
q=randint(2^31,2^33)
n=randint(2^6,2^7)

#We construct a MiNTRU matrix with error variance 2*sigma:
sigma=2*sqrt(q)
(A,E,S)=MiNTRU(q,n,sigma)

#We generate a random tuple:
U=Muni(q,n)

#We apply our improved distinguisher:
distinguisher_short(q,A,5) # should return 1
distinguisher_short(q,U,5) # should return 0
#-->This is probably the limit for the improved method.



#
    -------------------------------------------------------------------------------

#Repeated toy example for distinguisher_short:
#--------------------------------------------------

#We test how often MiNTRU matrices and random matrices are
    correctly recognized by our distinguisher.
Positive_passed=0
Positive_failed=0
Negative_passed=0
Negative_failed=0
Test_failed=0
for i in range(100):
```

```python
    print(Positive_passed,Positive_failed,Negative_passed,Negative_failed,Test_failed)
    q=randint(2^10,2^16)
    n=randint(2,2^5)
    sigma=2*sqrt(q)

    #We generate an iNTRU tuple and a random one:
    (A,E,S)=MiNTRU(q,n,sigma)
    U=Muni(q,n)

    #We apply our distinguisher:
    R=distinguisher_short(q,A,6)
    if R==-1:
        Test_failed=Test_failed+1
    elif R==1:
        Positive_passed=Positive_passed+1
    else:
        Positive_failed=Positive_failed+1
    RU=distinguisher_short(q,U,6)
    if RU==-1:
        Test_failed=Test_failed+1
    elif RU==0:
        Negative_passed=Negative_passed+1
    else:
        Negative_failed=Negative_failed+1
print(Positive_passed,Positive_failed,Negative_passed,Negative_failed,Test_failed)


#The final result should show a tuple (A,B,C,D,F) where:
# A denotes the number of correctly recognized iNTRU tuples (should be high)
# B denotes the number of wrongly interpreted iNTRU tuples (should be low)
# C denotes the number of correctly recognized random tuples (should be high)
# D denotes the number of wrongly interpreted random tuples (should be low)
# F denotes the number of failed tests due to the invertibility condition (should be low)





# ----------------------------------------------------------------------

#Conclusion:
```

```
#------------



#In our empirical trials (100 synthetic matrices and 100 random matrices) with the setup as stated above:
#1) The distinguisher_short with b=6 wrongly declared MiNTRU matrices as random in 48% of the (non-failed) cases.

#2) The distinguisher_short correctly determined all random matrices (in tests that didn't fail).

#3) The distinguisher_short failed in 5% of the cases due to the invertibility condition.


#--> Given the blow-up factor of LLL, the improved distinguisher does moderately well in determining MiNTRU matrices.

#--> More precise reduction algorithms (i.e. BKZ) may strongly improve those statistics.

#--> In any case, the right detection is non-negligible.
```