Practical Lattice Cryptosystems: NTRUEncrypt and NTRUMLS

by

John M. Schanck

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization (Quantum Information)

Waterloo, Ontario, Canada 2015

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Some sections of this work have appeared in publications and pre-prints co-authored by myself, John M. Schanck. Such sections are designated with one of the following symbols:

* Denotes that the section appeared in the paper describing NTRUMLS:

  [36] Jeff Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, and William Whyte. Transcript Secure Signatures Based on Modular Lattices. In Michele Mosca, editor, *Post-Quantum Cryptography*, number 8772 in Lecture Notes in Computer Science, pages 142–159. Springer International Publishing, October 2014.

† Denotes that the section appeared in a recent preprint on choosing NTRUEncrypt parameters:

  [37] Jeff Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing Parameters for NTRUEncrypt. Technical Report 708, 2015.

An annotated list of such sections is provided below.

- Algorithm 1 of Section 3.1, Algorithms 4 and 5 of Section 3.3.3, and Algorithm 6 of Section 3.5 all appear, either verbatim or with minor formatting changes, in [37].
- Section 3.4.4 has been adapted from [37]. A very early draft of the version from [37] was written by my co-authors, the version that appears here is primarily my own.
- Algorithms 7, 8 and 9 of Section 4.3 appear verbatim in [36].
- Sections 4.4 and 4.5 appeared, with minor formatting modifications, in [36]. and were written in collaboration with the other authors of that work. The proof of Proposition 12 of Section 4.4 is due to Joseph Silverman.
- Section 5.5.2 and Section 5.8 appeared in [37].

Abstract

Public key cryptography, as deployed on the internet today, stands on shaky ground. For over twenty years now it has been known that the systems in widespread use are insecure against adversaries equipped with quantum computers – a fact that has largely been discounted due to the enormous challenge of building such devices. However, research into the development of quantum computers is accelerating and is producing an abundance of positive results that indicate quantum computers could be built in the near future. As a result, individuals, corporations and government entities are calling for the deployment of new cryptography to replace systems that are vulnerable to quantum cryptanalysis. Few satisfying schemes are to be found.

This work examines the design, parameter selection, and cryptanalysis of a *post-quantum* public key encryption scheme, NTRUEncrypt, and a related signature scheme, NTRUMLS. It is hoped that this analysis will prove useful in comparing these schemes against other candidates that have been proposed to replace existing infrastructure.

Acknowledgements

I am, first and foremost, indebted to my colleagues at Security Innovation and Brown University. In the order that we met: William Whyte, Mark Etzel, Jeff Hoffstein, Jill Pipher, Joe Silverman, Virendra Kumar, and Zhenfei Zhang. It has been a great pleasure working with each of you.

I owe William Whyte a special thanks. He brought me on at Security Innovation for a 90 day stint writing a reference implementation of "NTRUSign with Perturbations;" I was to start on October 3rd of 2011. William received an email on the 30th of September notifying him that Phong Nguyen and his student, Léo Ducas, had broken the scheme. Needless to say there wasn't much need for a reference implementation after that, so William put me to work developing a new scheme instead. It wasn't a job I was qualified for, but William showed me the ropes and gave me the opportunity to educate myself. I certainly would not be where I am today without his generosity.

Also a special thanks to Jeff Hoffstein for providing me with an abundance of challenge problems in cryptanalysis, and for not being too miffed when I solved the few that I did.

Many thanks to Michele Mosca for advising me throughout my Masters and for alerting me to the need for post-quantum cryptography at the Quantum Key Distribution summer school at the University of Waterloo in the summer of 2011.

Lastly, I would like to express my eternal gratitude to my undergraduate advisor, Herb Bernstein, who taught me to pay close attention to the value ladenness of knowledge work.

Dedication

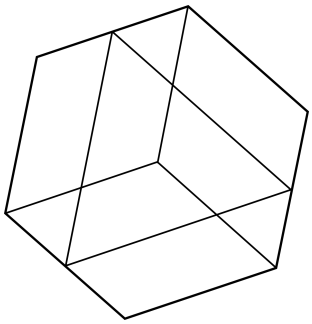

Table of Contents

Chapter 1

Post-quantum cryptography

1.1 A step into the unknown

A visitor to Chicago's Century-of-Progress International Exhibition in the summer of 1933 might have seen a puzzling device: a whirling mass of gear driven wheels built by Derrick N. and Derrick H. Lehmer – a father and son team of mathematicians. The Photoelectric Number Sieve, as it was called, was a calculational tool for use in pure mathematics. The machine had first been demonstrated nearly a year prior before a roomful of their colleagues. It caused something of a sensation, in no small part due to the Lehmers' showmanship. As recalled by his father, before Derrick H. threw the switch that would set the device about its task, he exclaimed: "Here we step out into the unknown!" [50].

He flipped the switch, gears cranked, a light flickered, minutes passed. Then *click*. The device stopped. And the group gathered around learned that

$$5283065753709209 = 59957 \times 88114244437.$$

A small pencil and paper calculation then resolved their noble goal: the complete factorization of $2^{95} + 1$ had been verified [50].

The elder Lehmer would later refer to these excursions in factorization as "hunting big game" in the theory of numbers. This is not pure boasting. From the mathematics to the circuitry, the mechanisms him and his son built were the absolute state of the art. They had constructed "a new kind of calculating machine, applicable to [factorization], in which modern physics has made its contribution to the oldest and least practical branch of mathematics" [49].

Of course, the physics the machine relied on was purely classical; what was truly modern was the engineering. The engineering of 1932 allowed for subtle enough control of a physical system as to employ it in the solution of a mathematical problem too difficult for human calculators. Here's how it worked: The wheels were set based on the number to be factored, then spun at a high rate. When holes in all of them aligned a light would shine through and strike a photoelectric cell. The signal from the photoelectric cell would be amplified to trigger a relay that would disengage the motor. Then the wheels could be turned back until the light shone through once more. A counter identified how many revolutions it had taken to reach that point, and the value of this counter could be used (by a skilled human calculator) to compute a factorization of the input. The device was enormously effective at its task – it would not be outperformed by fully electronic computers until the 1960s.

Less than a century-of-progress later, though, the number theoretic big game of 1932 is no longer so big. The sieves of today – the algorithmic progeny of the Lehmers' contraptions – are far more subtle in their machinations (though more no adroit in their mechanics). A specimen such as the RSA-768 challenge

$$
\begin{aligned}
N = \; & 1230186684530117755130494958384962720772853569595334792197 \\
& 3224521517264005072636575187452021997864693899564749427740 \\
& 6384592519255732630345373154826850791702612214291346167042 \\
& 9214311602221240479274737794080665351419597459856902143413
\end{aligned}
$$

can be teased apart in just a few thousand years of CPU time.

Surely this would excite the Lehmers, but it also excites a great many non-mathematicians today as well. Because over the past thirty years factoring has taken on new, wild, grandiose, inflated importance as one buttress of our information security infrastructure.

Nearly every secure communications channel established on the internet today relies on the hardness of factoring. At the very least to verify the identities of the communicating parties, often for their confidentiality as well. Only one other problem is relied on for public key cryptography at scale, that being the discrete logarithm problem in certain finite abelian groups. For a few more years these problems will serve us well, but by the centennial of the Lehmers' accomplishment they will no longer be difficult enough to base cryptography on with confidence.

In 1994 a mathematician at Bell Labs by the name of Peter Shor proposed an algorithm capable of efficiently factoring arbitrary integers. By all accounts a feat that should have terminated any further research into factoring – the problem was solved. In the same paper he solved the discrete logarithm problem in finite fields, and extensions of his work have

solved the discrete logarithm problem in arbitrary abelian groups. Public key cryptography has been in tatters since then – though we've only very recently started to act as though that were the case.

That's because there was a catch: a machine capable of running Shor's algorithm did not exist in 1994, and many did not think it could be built. The engineering of the day was not ready to control physical systems with the subtlety needed.

Now, 21 years later that's changing. There is widespread belief that the device, a quantum computer, can be built. Some day soon a speech will be made in a voice raised to be heard over the *cha-chink*s of dilution refrigerator compressors; a switch will be flipped, the machine's state will start precessing, and once again we will step out into the unknown.

The quantum computer is among the most daring and beautiful scientific ideas I've yet encountered. To believe that one could be built, only a generation ago, would have been a profound expression of faith in the quantum theory. A faith that all of the little quirks of the theory would be quirks of experiment too – phenomena that had not been observed, that could scarcely be believed, and frequently were not believed. Yet experiment after experiment confirms that the quantum theory is correct, and more importantly experiments are every day confirming that quantum systems can be manipulated with the deft touch needed to encode, to preserve, and to transform information reified by their degrees of freedom.

These extraordinary machines will be built, and – despite what I am about to say – I even believe that they *should* be built. They should be built for the same reason the Lehmers built their Photoelectric Sieve. Out of curiosity, with a sense of adventure, the hope that they will push the boundaries of our knowledge just a little bit further.

But we cannot build the quantum machine for these reasons, because we cannot deny that such a device will be used for cryptanalysis. The factoring and the discrete logarithm problems are no longer an amusement of the pure mathematician – they have been politicized. The unknown that those huddled around the device will step out into when it is turned on will not be some small patch of terrain in the wilds of mathematics, but rather the contents of a diplomat's inbox, or a dissident's chat log, or a corporation's private network.

Who will stand there on that day?
Who will be privileged to use the quantum computer?
To what ends will they use it?
These are questions we must ask ourselves, questions that must shape the work we do.

If the academic community is to push for the development of quantum computers it is our ethical obligation to pursue, in parallel, the development of the information security

infrastructure necessary for a world in which quantum computers exist. To do otherwise would not only be unethical, but, through arms regulations, would likely condemn quantum technologies to military and espionage roles far removed from scientific applications.

So what are we to do?

There is a research program that can be pursued to make the world safe for quantum computers – to make factoring and discrete logarithm curiosities once more. It involves developing cryptosystems that are secure against quantum attackers, but that can be run on present-day computers and present-day networks.

Such is the goal of *post-quantum cryptography.*

There are a number of candidate schemes already in the literature, and the prospects for developing a handful of these to the point where we might have some confidence in deploying them are good.

However, the situation is complicated by the fact that communications sent today may be stored for decryption when quantum computers are finally available. Michele Mosca has distilled the situation into an elegant conditional [60]

Let	x	be the time it will take to retool our existing infrastructure,
	y	be the duration for which our communications must remain confidential, and
	z	be how long we have before large quantum computers are built.
If $x + y > z$,		then we have a serious problem today.

Given the diversity of players interested in developing quantum computers we have little hope of increasing z, and y is for each individual to decide – surely for some it is already too late. The only parameter even partially in our control is x.

1.2 Where we stand

There are a variety of systems already in the literature that are conjectured to be secure against quantum computers. This thesis is an attempt to bring two of those systems closer to deployment through implementation, analysis, and the selection of parameters. I do not claim that these are the best, nor that they should be deployed without further consideration.

There are several excellent introductions to the field of post-quantum cryptography, in particular [12]. To determine x (the time it will take to retool our existing infrastructure) we first need to know what is broken.

Ciphers, hash functions, authenticators, and other *symmetric primitives* are weakened by generic quantum search techniques, but only by a quadratic factor. As an example, consider a preimage attack on a k-bit hash function. If h is an injective function (or is nearly so) with a domain of size 2^k then one can find a preimage of any value in the range of h using Grover's quantum search algorithm. Grover's algorithm makes approximately $2^{k/2}$ calls to a subroutine, and this subroutine can be evaluated by a quantum circuit that is of polynomial size with respect to a circuit for h. This polynomial overhead becomes negligible as k tends to infinity, and one can reasonably say that Grover's algorithm provides a quadratic ($2^k \mapsto 2^{k/2}$) speedup over the generic classical technique of brute force search. In cryptography k will almost always be between 80 and 512, and it seems likely that Grover's algorithm outperforms brute force search for at least some values in this range. However, careful analysis is required to determine whether there is a speedup for any particular (h, k) pair[1].

It should be mentioned that the analysis required to say whether or not a particular symmetric primitive is broken by quantum techniques is not a purely technical analysis – risk analysis should inform our decision of what "broken" means in quantum cryptanalysis (and classical cryptanalysis, for that matter). A 256-bit hash function offers at most 128-bit security against preimage attacks in a worst-case analysis based on Grover's algorithm, but this does not mean we should all rush to adopt wider primitives. I would argue that the risk of someone developing a quantum computer capable of performing 2^{128} operations of any type, let alone 2^{128} Grover iterations with a non-trivial h, in the next 100 years is so astonishingly low as to be entirely dismissable. This is not to say that quantum cryptanalysis has nothing to say about symmetric primitives – there may be more nuanced quantum attacks against individual primitives and protocols – but merely that the generic attacks based on Grover's algorithm are unlikely to be a significant threat in most scenarios.

The case for asymmetric primitives is substantially more complicated. Signatures, public key encryption, and key agreement based on factoring or discrete logarithm problems are hopelessly broken. If fault tolerant quantum computers can be built at all, then it is a near certainty that large enough machines can be built to run Shor's algorithm. Fortunately there are attractive post-quantum schemes in each category.

For signatures, if one's concern for security outweighs one's concern for efficiency, then hash-based signatures such as XMSS and SPHINCS are the clear winners. These will remain secure in a quantum setting as long as preimage resistant hash functions exist in

[1] Many authors ignore the dependence on h, but this is foolish if one cares about concrete estimates rather than asymptotics. Logical quantum bits are expensive in any realistic model of quantum computation. *A priori* one should expect the cost of preimage search on a memory-hard hash function to be greater than the cost of preimage search on a function tuned for high throughput or small circuit complexity.

a quantum setting, and this is overwhelmingly likely. As an added benefit these schemes are mature enough to be deployed today.

Signature schemes based on systems of multivariate quadratic equations appear to provide both efficient operations and a short signature length, but confidence in their security is (perhaps unfairly) low. Signatures based on lattices, particularly ideal- or module-lattices, provide the best known trade-off between key-size, signature-size, and performance. Again, confidence needs to improve before these can be deployed.

For public key encryption the most serious candidate schemes are based on either codes or lattices. Confidence in both codes and lattices without structure is high. The most efficient schemes, however, are again those based on ideal- or module-lattices. Again, confidence in such schemes could be improved.

For key exchange protocols there is a promising scheme based on elliptic curve isogenies [45] that has received comparatively little attention. Lattices with structure are again a prominent contender, and several systems have been proposed for deployment in the near future [13, 64].

In summary, systems based on lattices with exceptional algebraic structure perform well in all categories, but need further investigation before they can be deployed with confidence. Both of the schemes considered below, NTRUEncrypt and NTRUMLS fall into this category.

In the past year the number of voices discussing deployment of post-quantum systems has dramatically increased.

In August 2015 the United States National Security Agency announced plans to incorporate post-quantum cryptography into its "Suite B" list of cryptographic algorithms. This is a list of schemes that the NSA deems suitable for use in protecting classified and unclassified communications [1]. They did not specify which algorithms they are investigating, but the announcement has produced a noticeable uptick in interest in post-quantum cryptography.

In September 2015 the Horizon 2020 funded PQCRYPTO project announced initial recommendations for "long-term secure" post-quantum systems [7]. For public key encryption they recommend a particular parameter set for the McEliece system; a variant of NTRUEncrypt due to Stehle and Steinfeld [75] was also mentioned to be under consideration. For signatures they recommend either of two hash based schemes: XMSS, with any of the parameters specified in the active Internet Draft for that system, or SPHINCS-256.

More recently NIST and NSA have announced that they plan to call for proposals for standards in 2016 or 2017 [14].

Chapter 2

Preliminaries

The focus of this work will be two lattice-based cryptosystems: a public key encryption scheme called NTRUEncrypt, and a signature scheme called NTRUMLS.

For the reader who wants only to understand the cryptographic constructions of Chapters 3 and 4 the definition of a lattice given by Definition 1 will suffice and the remainder of this section may be skipped and referred back to when new concepts are encountered.

Notation Elements of a $\mathbb{Z}$-module are written in bold face, $\boldsymbol{v}$. If the module is presented as a submodule of, say, $\mathbb{Z}^n$, then $\boldsymbol{v}$ will be treated as a row vector where the ith coefficient with respect to the standard basis is then written as v_i. If $\boldsymbol{v}$ is an element of a polynomial ring that is a finitely generated as a $\mathbb{Z}$-module, e.g. $R = \mathbb{Z}[\boldsymbol{x}]/(\boldsymbol{x}^n)$, we use the convention that v_i is the coefficient of $\boldsymbol{x}^i$.

Sets are denoted with a calligraphic font, e.g. $\mathcal{S}$, matrices with upper case roman letters, e.g. M, and integers as lowercase roman letters, q. An exception is the use of N. By convention N is the rank of a particular $\mathbb{Z}$-module used in NTRU.

We will make frequent use of extension of scalars to turn $\mathbb{Z}$-modules into real vector spaces. If this is unfamiliar, for a $\mathbb{Z}$-module L the notation $L \otimes \mathbb{R}$ may simply be thought of as denoting the vector space $\mathrm{span}_{\mathbb{R}}(B)$ for any basis $B \subset L$. Bases will always be ordered. Matrices and ordered sets of vectors are treated interchangeably. In particular, if a basis for a submodule of $\mathbb{Z}^m$ is given as $B = \{\boldsymbol{b}_i\}_{1 \leq i \leq n}$ we may treat B as a matrix in $\mathbb{Z}^{n \times m}$ (hence we use a roman font).

Equivalence modulo q and *reduction* modulo q will be distinguished by parenthesizing $(\mathrm{mod}\ q)$ for equivalence and not for reduction. To be precise, $a = b \bmod q$ is the unique

representative of $b+(q)$ in $\mathbb{Z}\cap[-q/2, q/2)$, in other words a is the smallest integer in absolute value satisfying $a \equiv b \pmod{q}$. This primarily simplifies descriptions of algorithms in which we must work with a specific representative of an equivalence class.

The notation $\mathcal{B}_N(r)$ denotes the subset of $\mathbb{Z}^N$ contained in the euclidean ball of radius r centered at the origin, and $\mathcal{C}_N(r)$ denotes the subset of $\mathbb{Z}^N$ contained in the hypercube of radius r.

An element sampled randomly from a set $\mathcal{S}$ is written $x \leftarrow_{\$} \mathcal{S}$. We will only sample from finite sets, and sampling is always performed with respect to the uniform distribution.

2.1 Generalities on Lattices

The definition of a lattice that one normally encounters in lattice based cryptography is as follows:

Definition 1 (Euclidean lattice)**.** A lattice of rank n is the set of all integer combinations of n linearly independent vectors in $\mathbb{R}^m$, i.e.

$$L = \mathbb{Z}\boldsymbol{b}_1 + \mathbb{Z}\boldsymbol{b}_2 + \cdots + \mathbb{Z}\boldsymbol{b}_n.$$

We will need more robust definitions than one normally encounters on a first tour of lattice crypto. Hence our lattices will not be *just* integer combinations of some set of vectors, or subgroups of $\mathbb{Z}^n$ (or $\mathbb{R}^n$), even though such definitions are sufficient for many purposes.

The treatment here is inspired by Henri Cohen's excellent text [17].

Definition 2 (Lattice)**.** A *lattice* (L, ρ) is a free $\mathbb{Z}$-module L of finite rank with an associated positive definite quadratic form $\rho : L \otimes \mathbb{R} \to \mathbb{R}$.

Recall that ρ is a quadratic form iff the map $\langle \cdot, \cdot \rangle_\rho$ defined by

$$(\boldsymbol{x}, \boldsymbol{y}) \mapsto \tfrac{1}{2}(\rho(\boldsymbol{x} + \boldsymbol{y}) - \rho(\boldsymbol{x}) - \rho(\boldsymbol{y}))$$

is a symmetric bilinear form, i.e. for all $\boldsymbol{x}, \boldsymbol{y} \in L \otimes \mathbb{R}$, we have

1. $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_\rho = \langle \boldsymbol{y}, \boldsymbol{x} \rangle_\rho$
2. $\langle \boldsymbol{x} + \boldsymbol{y}, \boldsymbol{z} \rangle_\rho = \langle \boldsymbol{x}, \boldsymbol{z} \rangle_\rho + \langle \boldsymbol{y}, \boldsymbol{z} \rangle_\rho$

3. $\langle c\boldsymbol{x}, \boldsymbol{z}\rangle_\rho = c\langle \boldsymbol{x}, \boldsymbol{z}\rangle_\rho$.

The condition that ρ is positive definite means that $\rho(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \neq 0$.

We will typically restrict ourselves to the study of euclidean lattices, i.e. those for which the associated quadratic form is the squared euclidean length,

$$\rho(\boldsymbol{v}) = \|\boldsymbol{v}\|^2 = \sum_{i=1}^{n} v_i^2,$$

and the symmetric bilinear form is the usual inner product. For euclidean lattices we will omit the subscript on the inner product and write $\langle \boldsymbol{v}, \boldsymbol{w}\rangle$. Also, provided that doing so is unambiguous, we will refer to L itself as the lattice.

In general, a rank n latttice may be presented as a submodule of $\mathbb{Z}^m$ for some $m \geq n$. A basis for a rank n lattice is any spanning set of size n.

While all free $\mathbb{Z}$-modules of rank n are isomorphic to $\mathbb{Z}^n$, lattices fall into different equivalence classes depending on the associated quadratic form, as the following definition demonstrates.

Definition 3 (Lattice isomorphism)**.** Two lattices (L, ρ) and (L', ρ') are said to be isomorphic if there exists a length-preserving $\mathbb{Z}$-module isomorphism between them, i.e. if there exists a bijection $\phi : L \to L'$ such that $\rho(\boldsymbol{v} + \boldsymbol{w}) = \rho'(\phi(\boldsymbol{v} + \boldsymbol{w})) = \rho'(\phi(\boldsymbol{v}) + \phi(\boldsymbol{w}))$ for all $\boldsymbol{v}, \boldsymbol{w} \in L$.

To emphasise this point let's momentarily examine lattices from a quadratic-form centric viewpoint by looking at the lattice isomorphism induced by an isomorphism between L and $\mathbb{Z}^n$. We'll specify that L is euclidean so we're looking for a lattice isomorphism $(L, \|\cdot\|^2) \to (\mathbb{Z}^n, \rho)$ for some yet unknown ρ. We get an appropriate $\mathbb{Z}$-module isomorphism simply by choosing a basis $B = \{\boldsymbol{b}_i\}_{1\leq i\leq n}$ for L. Every element of L can be written uniquely as an integer combination of the $\boldsymbol{b}_i$, and every integer combination of the $\boldsymbol{b}_i$ gives us an element of the lattice; the isomorphism we're after is given by the correspondence between $\boldsymbol{v} = \sum_{i=1}^n v_i\boldsymbol{b}_i \in L$ and $(v_1, \ldots, v_n) \in \mathbb{Z}^n$. So, keeping in mind that we need to preserve the euclidean length, we see that ρ is given by

$$\begin{aligned}
\rho((v_1, \ldots, v_n)) &= \|\textstyle\sum_{i=1}^n v_i\boldsymbol{b}_i\|^2 \\
&= \left\langle \sum_{i=1}^{n} v_i\boldsymbol{b}_i,\ \sum_{j=1}^{n} v_j\boldsymbol{b}_j \right\rangle \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} v_i v_j \langle \boldsymbol{b}_i, \boldsymbol{b}_j\rangle.
\end{aligned}$$

Or, equivalently, ρ can be expressed by the real symmetric matrix Q:

$$\rho(\boldsymbol{v}) = \boldsymbol{v}Q\boldsymbol{v}^T \quad \text{where } (Q)_{i,j} = \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle.$$

Matrices such as Q are useful enough to get their own name.

Definition 4 (Gram matrix)**.** Let $\{\boldsymbol{b}_i\}_{1 \le i \le n}$ be a basis for a lattice (L, ρ). Then the matrix of scalar products

$$(Q)_{i,j} = \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle_\rho$$

is the Gram matrix of the $\boldsymbol{b}_i$.

Just as the above shows that picking a basis can be viewed as a lattice isomorphism, so can changing bases. A change of basis $B \mapsto B' = UB$ induces the lattice isomorphism $(\mathbb{Z}^n, \rho) \to (\mathbb{Z}^n, \rho')$ where $\rho'(\boldsymbol{v}) = \rho(\boldsymbol{v}U) = \boldsymbol{v}UQU^T\boldsymbol{v}^T$.

It may also be useful to consider orthogonal transformations, such as coefficient permutations, of the ambient space. These may also be seen as lattice isomorphisms: if L is presented as a submodule of $\mathbb{Z}^m$ and $O \in GL_m(\mathbb{R})$, then the map $\boldsymbol{x} \mapsto \boldsymbol{x}O$ induces a lattice isomorphism.

Definition 5 (Volume of lattice)**.** The *volume* of a lattice (L, ρ) is the covolume of L in $L \otimes \mathbb{R}$. If $B = \{\boldsymbol{b}_i\}$ is a basis for L and Q is the Gram matrix associated to ρ then the volume of (L, ρ) is

$$\mathrm{vol}(L, \rho) = \det(Q)^{1/2}.$$

For euclidean lattices (or when ρ is otherwise unambiguous) we will denote the volume of L as $\mathrm{vol}(L)$.

Note that the volume is an invariant of the lattice – isomorphic lattices have equal volumes. A related lattice invariant is the set of successive minima.

Definition 6 (Succesive minima)**.** The *successive minima* of a lattice are a set of n invariants that encode the lengths of the lattice's shortest non-zero vectors. The ith successive minima is the radius of the smallest closed ball that contains i linearly independent vectors of L:

$$\lambda_i(L) = \inf\{r \mid \dim(\mathrm{span}(\mathcal{B}_n(r) \cap L)) \ge i\} \qquad (1 \le i \le n). \tag{2.1}$$

Estimating the successive minima is an issue of fundamental importance in the study of lattices. The following theorem is one of the most essential tools for achieving this task.

Theorem 1 (Minkowski's Convex Body Theorem)**.** *Let L be a euclidean lattice, and let S be a point set of volume* $\mathrm{vol}(S)$ *that is symmetric about the origin, compact, and convex. If* $\mathrm{vol}(S) \geq m2^n\mathrm{vol}(L)$ *then S contains at least m distinct pairs of points in L:* $\{\pm \boldsymbol{u}_i\}_{1\leq i\leq m}$.

Minkowski's convex body theorem immediately yields an upper bound on the first successive minima of a lattice with respect to any norm. The n dimensional (euclidean) ball of radius R has volume

$$\mathrm{vol}(\mathcal{B}_n(R)) = \frac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)} \cdot R^n. \tag{2.2}$$

By Theorem 1 with $S = \mathrm{vol}(\mathcal{B}_n(\lambda_1(L)))$ we have:

$$\lambda_1(L) \leq \frac{2}{\sqrt{\pi}} \left(\mathrm{vol}(L) \cdot \Gamma\left(\frac{n}{2}+1\right)\right)^{1/n}. \tag{2.3}$$

A heuristic variant of this bound, dubbed the Gaussian heuristic, claims that the factor of 2 may be omitted. Many authors additionally use the Sterling approximation for Γ and define the Gaussian heuristic as follows:

Definition 7 (Gaussian heuristic)**.** The Gaussian Heuristic is a claim that

$$\lambda_1(L) \approx \sqrt{\frac{n}{2\pi e}} \cdot \mathrm{vol}(L)^{1/n} \tag{2.4}$$

for "most" lattices. In fact, for random lattices this same approximation is expected to hold for all n successive minima. We define $\mathrm{GH}(L)$ as the right hand side of Eq. 2.4.

Better bounds on the successive minima of a lattice tend to look at information encoded in a given basis of the lattice. Much of this information is extracted through the familiar Gram-Schmidt process.

Definition 8 (Gram-Schmidt orthogonalization)**.** Given an ordered basis $B = \{\boldsymbol{b}_i\}_{1\leq i\leq n}$ of a rank n lattice L the *Gram-Schmidt vectors* of B are denoted by $B^* = \{\boldsymbol{b}_i^*\}_{1\leq i\leq n}$ and defined by

$$\boldsymbol{b}_i^* = \boldsymbol{b}_i - \sum_{j=1}^{i-1} \mu_{i,j}\boldsymbol{b}_j^* \qquad (1 \leq i \leq n) \tag{2.5}$$

wherein the *Gram-Schmidt coefficients*, $\{\mu_{i,j}\}_{1\leq j<i\leq n}$, are defined by

$$\mu_{i,j} = \frac{\langle \boldsymbol{b}_i, \boldsymbol{b}_j^*\rangle}{\langle \boldsymbol{b}_j^*, \boldsymbol{b}_j^*\rangle} \qquad (1 \leq j < i \leq n). \tag{2.6}$$

Lemma 2. *The Gram-Schmidt vectors* $\{\boldsymbol{b}_i^*\}_{1\le i\le n}$ *of any basis of* L *satisfy*

$$|\det(L)| = \prod_{i=1}^{n} \|\boldsymbol{b}_i^*\|.$$

Proof. It is readily seen that $\boldsymbol{b}_i^*$ is the projection of $\boldsymbol{b}_i$ onto the orthogonal complement of $\mathrm{span}_{\mathbb{R}}(\{\boldsymbol{b}_j\}_{1\le j<i}) = \mathrm{span}_{\mathbb{R}}(\{\boldsymbol{b}_j^*\}_{1\le j<i})$. Hence B^* is an orthogonal (but not necessarily orthonormal) basis for $L\otimes\mathbb{R}$. Equation 2.5 inductively expresses the coordinates of the $\boldsymbol{b}_i^*$ in terms of those of the $\boldsymbol{b}_i$ and it can be seen that this change of basis is given by a lower triangular matrix with diagonal entries equal to 1. Thus $\prod_{i=1}^{n}\|\boldsymbol{b}_i^*\| = \det(B^*) = \det(B) = |\det(L)|$. □

Orthogonal projections like those used to define the $\boldsymbol{b}_i^*$ are used frequently enough in lattice cryptanalysis to reserve notation for them. Hence, for any basis of a rank n lattice L we define $\pi_{B,i} : L\otimes\mathbb{R} \to L\otimes\mathbb{R}$ as the projection orthogonal to $\mathrm{span}_{\mathbb{R}}(\{\boldsymbol{b}_j^*\}_{1\le j\le i-1})$. When the basis is clear from context we will write π_i. Note that $\pi_{B,i}(\boldsymbol{v}+\boldsymbol{w}) = \pi_{B,i}(\boldsymbol{v}) + \pi_{B,i}(\boldsymbol{w})$, so it is natural to consider projection as a lattice homomorphism.

Definition 9 (Projected lattice)**.** The ith *projected lattice* of (L,ρ) with respect to the basis B is $(\pi_{B,i}(L),\rho)$ where $\pi_{B,i}(L) = \{\pi_{B,i}(\boldsymbol{v}) \ : \ \boldsymbol{v}\in L\}$. When it will not introduce ambiguities we may write $B^{(i)}$ to denote $\pi_{B,i}(L)$.

Clearly it is always the case that short vectors in L have short projections, i.e. $\rho(\pi_{B,i}(\boldsymbol{v})) \le \rho(\boldsymbol{v})$, so it is natural to attempt to find short vectors in L by "lifting" short vectors from a projected lattice $B^{(i)}$ to L. More generally one can consider a sequence of lattices such as

$$L = B^{(1)} \xrightarrow{\pi_{B,2}} B^{(2)} \xrightarrow{\pi_{B,3}} B^{(3)} \xrightarrow{\pi_{B,4}} \cdots \xrightarrow{\pi_{B,n+1}} B^{(n+1)} = \{0\}$$

and iteratively lift short vectors from $B^{(i)}$ to $B^{(i-1)}$. *Enumeration algorithms* (Section 5.2) proceed precisely in this fashion by performing a depth first search on a tree of vectors that is rooted at $\{0\}$ and has (a subset of) the elements of $B^{(i)}$ at depth i.

There are infinitely many bases for a lattice related by unimodular transformations, some of which are more useful than others. Fortunately there is limit on how "bad" a basis can be, as there exists an efficiently computable normal form for bases of $\mathbb{Z}$-modules, the *Hermite Normal Form* or HNF. For algorithms we can insist that input bases are either in the normal form or are "reduced" in some sense, and this allows us to prove complexity bounds that depend only on invariants of the lattice.

Definition 10 (Hermite Normal Form (adapted from [17]))**.** An $n \times m$ matrix M with integer coefficients is in Hermite Normal Form (HNF) if there exists $r \geq 0$ and a strictly increasing map f from $[r+1, n]$ to $[1, m]$ satisfying the following properties

1. The first r rows of M are equal to 0.
2. $M_{j,f(j)} \geq 1$ for $j \in [r+1, n]$.
3. $M_{k,f(j)} = 0$ for $k \in [f(j)+1, n]$
4. $0 \leq M_{i,f(j)} < M_{j,f(j)}$ for $i \in [0, j-1]$.

When $\det(M) \neq 0$, Definition 10 says that M is in HNF if and only if M is upper triangular with positive diagonal and the entries in each column are non-negative and bounded above by the diagonal entry in that column, i.e. $0 \leq M_{j,i} < M_{i,i}$ for $j > i$.

One final set of invariants that are useful in analyzing lattices are the elementary divisors. These determine the structure of the finite abelian group $\mathbb{Z}^n/L$.

Theorem 3 (Elementary divisor theorem (adapted from [17]))**.** *Let L be a $\mathbb{Z}$-submodule of L', and suppose* $\operatorname{rank}(L) = \operatorname{rank}(L') = n$. *The elementary divisors of L in L' are uniquely determined positive integers $s_1, \ldots, s_n$ that satisfy*

1. *For $1 \leq i < n$ we have $s_{i+1} | s_i$.*
2. *As $\mathbb{Z}$-modules, we have the isomorphism*

$$L'/L \simeq \bigoplus_{1 \leq i \leq n} (\mathbb{Z}/s_i\mathbb{Z}).$$

3. *There exists a $\mathbb{Z}$-basis $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$ of L' such that $(s_1\boldsymbol{v}_1, \ldots, s_n\boldsymbol{v}_n)$ is a $\mathbb{Z}$-basis of L.*

The lattices we will consider have very distinctive elementary divisors.

Definition 11 (q-ary lattice)**.** A full-rank lattice (L, ρ) of rank n for which $q\mathbb{Z}^n \subseteq L \subseteq \mathbb{Z}^n$ is a q-ary lattice.

Any lattice is q-ary for q that is an integer multiple of the least common multiple of the elementary divisors of L, i.e. $\operatorname{lcm}(s_1, \ldots s_n) | q$. An extreme case that occurs frequently in lattice based cryptography is when there exists k for which $s_1 = \cdots = s_k = q$ and $s_{k+1} = \cdots = s_n = 1$.

The q-ary lattices that one frequently encounters in the literature on lattice based cryptography are presented with respect to a matrix $A \in \mathbb{Z}^{m \times n}$, either as $\Lambda_q(A)$ or $\Lambda_q^{\perp}(A)$ where:

$$\Lambda_q(A) = \{\boldsymbol{y} \in \mathbb{Z}^n \ : \ \boldsymbol{x}A \equiv \boldsymbol{y} \pmod{q} \quad \text{has a solution } \boldsymbol{x} \in \mathbb{Z}^m\} \tag{2.7}$$

$$\Lambda_q^{\perp}(A) = \{\boldsymbol{y} \in \mathbb{Z}^n \ : \ A\boldsymbol{y}^T \equiv \boldsymbol{0} \pmod{q}\}. \tag{2.8}$$

2.2 Computational Problems on Lattices

The main computational tasks associated to lattices are the shortest and closest vector problems. The cryptosystems we consider will have reductions *to* these problems, and they will play an essential role in the concrete cryptanalysis of the schemes. Of course, it may be possible to attack the cryptosystems from an entirely different perspective, so these may not be the end of the story.

Definition 12 (The Approximate Shortest Vector Problem, SVP_γ)**.** Given a basis B for a lattice L of rank n, find a non-zero element of $\boldsymbol{v} \in L$ such that

$$\|\boldsymbol{v}\| \leq \gamma \cdot \lambda_1(L)$$

Definition 13 (The Approximate Closest Vector Problem, CVP_γ)**.** Given a basis B for a lattice L of rank n, and an element $\boldsymbol{t} \in L \otimes \mathbb{R}$, find an element of $\boldsymbol{v} \in L$ such that

$$\|\boldsymbol{v} - \boldsymbol{t}\| \leq \gamma \cdot \min_{\boldsymbol{x} \in L} \|\boldsymbol{x} - \boldsymbol{t}\|.$$

Definition 14 (Bounded Distance Decoding, BDD_α)**.** Given a basis B for a lattice L of rank n, and an element $\boldsymbol{t} \in L \otimes \mathbb{R}$ that is promised to be within distance $\alpha \cdot \lambda_1(L)$ of L, find the closest vector to $\boldsymbol{t}$.

2.3 Ideal Lattices and Module Lattices

It is often cumbersome to work with lattices – basic operations such as matrix multiplication take cubic time and quadratic space with respect to the rank. Consequently, there has been significant interest in developing cryptography based on lattices with algebraic structure that enables fast operations and more compact representations.

There are a wealth of interesting algebraic objects that have underlying $\mathbb{Z}$-module structure, and any of these can be turned into a lattice. Group rings and the rings of integers of algebraic number fields are the objects most frequently used to construct compact lattices for cryptographic use. Of these, the integers of cyclotomic fields of prime or power-of-two conductor are by far the most common.

For the remainder of this section, let R be a ring that is a free abelian group of finite rank under addition.

Definition 15 (Module lattice)**.** A lattice (L, ρ) is called a module lattice if there exists an R-module M and a $\mathbb{Z}$-module isomorphism $M \to L$.

Ideal lattices, as one might guess from their name, are a special case of module lattices for which the module under consideration is an R-submodule of R itself.

Definition 16 (Ideal lattice)**.** A lattice (L, ρ) is called an ideal lattice if there exists an ideal $I \subseteq R$ and a $\mathbb{Z}$-module isomorphism $I \to L$.

Note that module lattices are defined as lattices with the structure of R-modules, and not as R-modules with the structure of lattices. As such, they inherit notions such as rank and volume from lattices. For example, "the rank of L" is its rank as a $\mathbb{Z}$-module, not as an R-module.

The notion of q-ary lattice (Definition 11) generalizes readily. Mirroring Equations 2.7 and 2.8 for any $\vec{\boldsymbol{a}} \in R^m$ we define the q-ary module lattices

$$\Lambda_q(\vec{\boldsymbol{a}}) = \{\vec{\boldsymbol{y}} \in R^m \ : \ \boldsymbol{s} \cdot \vec{\boldsymbol{a}} \equiv \vec{\boldsymbol{y}} \pmod{q} \quad \text{has a solution } \boldsymbol{s} \in R\} \tag{2.9}$$

$$\Lambda_q^{\perp}(\vec{\boldsymbol{a}}) = \{\vec{\boldsymbol{y}} \in R^m \ : \ \sum_{i=1}^{m} \boldsymbol{a}_i \boldsymbol{y}_i \equiv \boldsymbol{0} \pmod{q}\}. \tag{2.10}$$

Note that these definitions only apply to cyclic R-modules[1], but they can be extended to arbitrary generating sets of R-modules in the obvious way. We defer making this generalization as will only utilize it briefly when discussing intersections of lattices of the type from Equation 2.9.

[1] A cyclic R-module is one that is generated by a single element of R^m.

2.4 Convolution Polynomial Rings

The first module lattices considered in the context of cryptography came from modules over the group ring of a cyclic group, and these were presented as the ring of integer polynomials modulo $\boldsymbol{x}^N - 1$.

These rings will be of primary importance for us, so we will henceforth define

$$R_N = \mathbb{Z}[\boldsymbol{x}]/(\boldsymbol{x}^N - 1). \tag{2.11}$$

Such rings are occasionally called convolution polynomial rings, owing to the fact that multiplication in R_N has the form of cyclic convolution[2]:

$$\left(\sum_{i=0}^{N-1} f_i \boldsymbol{x}^i\right)\left(\sum_{j=0}^{N-1} g_j \boldsymbol{x}^j\right) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f_i g_j \boldsymbol{x}^{i+j} \equiv \sum_{k=0}^{N-1}\left(\sum_{i=0}^{N-1} f_i g_{k-i}\right)\boldsymbol{x}^k \pmod{\boldsymbol{x}^N - 1}.$$

We represent elements of R_N as either polynomials or as elements of $\mathbb{Z}^N$ according to the natural isomorphism

$$f_0 + f_1\boldsymbol{x} + \cdots + f_{N-1}\boldsymbol{x}^{N-1} \mapsto [f_0, f_1, \ldots, f_{N-1}].$$

To provide the most flexibility we may even mix notation by letting R_N act on $\mathbb{Z}^N$ by right multiplication, i.e. by defining

$$[f_0, f_1, \ldots, f_{N-1}] \cdot \boldsymbol{x} = [f_{N-1}, f_0, \ldots, f_{N-2}]$$

and extending this action by linearity. In this way we obtain an algebra isomorphism between R_N and the sub-algebra of $\mathrm{End}(\mathbb{Z}^N)$ corresponding to *circulant matrices*. The isomorphism is given explicitly by the map $\mathsf{Circ} : R_N \to \mathrm{End}(\mathbb{Z}^N)$:

$$\mathsf{Circ}(\boldsymbol{f}) = \begin{pmatrix} f_0 & f_1 & \cdots & f_{N-1} \\ f_{N-1} & f_0 & \cdots & f_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ f_1 & f_2 & \cdots & f_0 \end{pmatrix}. \tag{2.12}$$

As convolution polynomial rings are the core algebraic objects underlying NTRUEncrypt and NTRUMLS, so some general statements about their structure are in order.

[2]Note that the index arithmetic is modulo N in this equation

First note that $\boldsymbol{x}^N - 1$ is reducible over $\mathbb{Z}$ for all $N > 1$, as can be seen from the familiar formula for the summation of a geometric series: $\boldsymbol{x}^N - 1 = (\boldsymbol{x} - 1)\sum_{i=0}^{N-1} \boldsymbol{x}^i$. For prime N, the polynomial $\sum_{i=0}^{N-1} \boldsymbol{x}^i$ can be seen to be irreducible over $\mathbb{Z}$ by performing the substitution $\boldsymbol{x} \mapsto \boldsymbol{x} + 1$ and applying Eisenstein's criterion. In general, for arbitrary composite N, this polynomial is reducible, but its factorization can be determined using a bit of number theory.

For $d \in \mathbb{Z}_{\geq 2}$ we will denote by $\zeta_d \in \mathbb{C}$ a primitive d^{th} root of unity. Clearly $\boldsymbol{x}^N - 1$ has roots at exactly the N distinct powers of ζ_N; thus

$$\boldsymbol{x}^N - 1 = \prod_{i=1}^{N} (\boldsymbol{x} - \zeta_N^i). \tag{2.13}$$

A standard result in number theory (see, for instance, [78]) is that the minimal polynomial of ζ_d over $\mathbb{Q}$ is given by the so called d^{th} *cyclotomic polynomial*

$$\Phi_d = \prod_{\substack{1 \leq k \leq d \\ (k,d)=1}} (x - \zeta_d^k) \in \mathbb{Q}[x]. \tag{2.14}$$

The minimality of Φ_d establishes that it is irreducible. We will additionally show that Φ_d has integer coefficients, and finally that $\boldsymbol{x}^N - 1$ has a complete factorization over $\mathbb{Z}$ into cyclotomic polynomials.

The roots of Φ_d are also roots of the monic polynomial with integer coefficients $\boldsymbol{x}^d - 1$, hence the roots of Φ_d are algebraic integers. By the well known relationship between the coefficients of a polynomial and elementary symmetric functions of its roots this implies that Φ_d has coefficients that are algebraic integers. Since $\Phi_d \in \mathbb{Q}[x]$ its coefficients must be algebraic integers in $\mathbb{Q}$, hence $\Phi_d \in \mathbb{Z}[x]$.

The claimed factorization of $\boldsymbol{x}^N - 1$ is given by

$$\boldsymbol{x}^N - 1 = \prod_{i=1}^{N} (\boldsymbol{x} - \zeta_N^i) = \prod_{d|N} \prod_{\substack{1 \leq k \leq d \\ (k,d)=1}} (\boldsymbol{x} - \zeta_N^{kN/d}) = \prod_{d|N} \Phi_d. \tag{2.15}$$

For convenience we now list a few easy to compute cyclotomic polynomials:

- $\Phi_1 = \boldsymbol{x} - 1$
- $\Phi_{2^k} = \boldsymbol{x}^{2^{k-1}} + 1$

- $\Phi_p = \boldsymbol{x}^{p-1} + \boldsymbol{x}^{p-2} + \cdots + \boldsymbol{x} + 1 \quad$ for prime p
- $\Phi_n = \prod_{d|n}(\boldsymbol{x}^{n/d} - 1)^{\mu(d)}$ for all N (by Möbius inversion of Equation 2.15).

It is customary at this juncture to point out that not all cyclotomic polynomials have coefficients in $\{-1, 0, 1\}$. In fact an elegant argument due to Schur shows that there are cyclotomic polynomials with coefficients of arbitrary magnitude. In particular, for $t > 2$ primes $p_1 < \cdots < p_t$ such that $p_1 + p_2 > p_t$, there is a coefficient of $\Phi_{p_1 \cdots p_t}$ equal to $1 - t$. As the first integer satisfying the constraints is $3 \cdot 5 \cdot 7 = 105$ it's quite easy to be deceived by small examples. A bit of trivia: it was Emma Lehmer, spouse of the Derrick H. Lehmer we met in the introduction, who gave the first unconditional proof that the coefficients of Φ_N can be arbitrarily large as N ranges over products of just $t = 3$ primes [51].

Returning to our main task, an application of the Chinese remainder theorem reveals the structure of R_N:

Fact 4 (Structure of R_N)**.**

$$R_N = \mathbb{Z}[\boldsymbol{x}]/(\boldsymbol{x}^N - 1) \simeq \prod_{d|N} \mathbb{Z}[\boldsymbol{x}]/(\Phi_d). \tag{2.16}$$

We will quite frequently work in $R_{N,q} = R_N/qR_N$ for a prime or prime-power $q \in \mathbb{Z}$. Another standard result in the study of cyclotomic fields yields the factorization of Φ_d in $\mathbb{Z}[x]/(p)$ for prime p.

Fact 5 (Primes above p in $\mathbb{Q}(\zeta_d)$)**.** *Let $r = \mathrm{ord}_{\mathbb{Z}_d^*}(p)$. Note that $r|\varphi(d)$ where $\varphi(d)$ is Euler's totient function, and $\varphi(d) = \deg(\Phi_d)$. There are $\varphi(d)/r$ irreducible factors of Φ_d in $\mathbb{Z}[x]/(p)$ each of degree r.*

Let $q = p^k$ and let $r(d) = \mathrm{ord}_{\mathbb{Z}_d^*}(p)$. Fact 5 immediately gives us an analogue of Equation 2.16 for $R_{N,q}$:

Fact 6 (Structure of $R_{N,q}$)**.**

$$R_{N,q} \simeq \prod_{d|N} \mathbb{Z}[x]/(q, \Phi_d) \simeq \prod_{d|N} \left(\mathbb{F}_{q^{r(d)}}\right)^{\varphi(d)/r(d)}. \tag{2.17}$$

Finally, calculating the number of invertible elements of R_N/qR_N is entirely analogous to the case of the integers modulo a composite and is given by a suitable generalization of the totient formula.

Fact 7 (Number of invertible elements in $R_{N,q}$)**.**

$$\left|(R_N/qR_N)^*\right| = q^N \prod_{d|N} \left(1 - q^{-r(d)}\right)^{\varphi(d)/r(d)}. \tag{2.18}$$

Chapter 3

NTRUEncrypt

NTRUEncrypt is a public key encryption scheme developed in the mid-90s by Jeff Hoffstein, Jill Pipher and Joe Silverman at Brown University [34]. The scheme was first presented in August 1996 in a minutes-long talk at the rump session of CRYPTO '96, and a paper was submitted to CRYPTO '97 in February of the following year. Perhaps surprisingly, given the enormous impact that NTRUEncrypt has had on the cryptographic community in the 18 years since then, the paper was rejected. It did eventually appear in print in the proceedings of ANTS '98 [39].

Lattice based cryptography, and the use of lattice reduction algorithms as a tool for cryptanalysis, had been considered previously. Lattice reduction, the LLL algorithm in particular, played an essential role in showing that early attempts at building cryptosystems based on the knapsack problem were insecure [4]. It was also used by Don Coppersmith to attack textbook RSA with partial knowledge of the plaintext [19]. On the constructive side, lattices burst onto the scene in 1996 with Ajtai's demonstration of a function that is provably one-way assuming that the shortest vector problem is hard in the *worst-case* [5]. This was quickly followed by a proposal from Ajtai and Dwork for a public key encryption scheme with security that depends on the n^c-unique shortest vector problem being hard in the worst-case [6]. Then at CRYPTO '97 a heuristic, but much more efficient, public key encryption scheme was proposed by Goldreich, Goldwasser, and Halevi [30].

Even the heuristic lattice based systems failed to be competitive with existing systems based on the RSA or discrete logarithm problems. They relied on full rank lattices presented as submodules of $\mathbb{Z}^n$, and hence had public and private keys of length $O(n^2)$ bits, and required matrix multiplication as a basic operation. The primary benefit of NTRUEncrypt, from a practical perspective, is its use of lattices with exceptional algebraic structure. The

module-lattices used in NTRUEncrypt admit linear presentations of keys, and quasilinear multiplication operations.

Practice-oriented systems have largely followed NTRUEncrypt in using lattices with algebraic structure as lattice cryptography has matured. The theoretical community, in parallel, has made great improvements in the efficiency of schemes that can be said to rely only on worst-case assumptions. In between are systems such as Ring-LWE that are provably as hard as worst-case problems on lattices with some algebraic structure.

We are left with a spectrum of schemes along an axis between efficiency and provable security. At the far end towards efficiency is NTRUEncrypt, followed by Ring-LWE instantiated at parameters for which its security proof carries little weight, followed by aggressively optimized unstructured lattice schemes, then Ring-LWE in general, and finally general lattice schemes with a worst-case security assumption.

No significant attacks have emerged that exploit the algebraic structure of NTRUEncrypt as presented below, or Ring-LWE in prime or power-of-two conductor cyclotomic fields, though it is possible to instantiate either scheme insecurely by choosing the ring poorly.

The inventors of NTRUEncrypt have also pursued commercialization and standardization. A patent was filed in 1997 (U.S. Patent No. 6081587) that will expire on August 19, 2017. The patent was assigned to Security Innovation, the current owner, in 2009. In 2013 Security Innovation made the patent free to use in software licensed under the GPL or similar free software licenses. Additional patents were filed for related schemes, such as NTRUSign, and for efficiency enhancements. U.S. Patent No. 7031468 covers the use of "product-form" keys and blinding polynomials, was filed in 2001, and will expire on August 24, 2021.

The primary goal of this chapter is to provide concrete parameter sets for NTRUEncrypt; however it also stands as a self-contained description of the scheme. In Section 3.3.3 we discuss the variant of NTRUEncrypt standardized in the Efficient Embedded Security Standard #1 [2]. Any system seeking to use NTRUEncrypt should make the modifications to the textbook scheme required by this standard. Some details concerning how to do so are omitted here; the interested reader should refer to the standard.

3.1 Primitives

The first step in instantiating NTRUEncrypt is to select the ring in which operations will take place. The canonical choice, and the choice we make here, is

$$R_N = \mathbb{Z}[\boldsymbol{x}]/(\boldsymbol{x}^N - 1).$$

The degree, N, is typically taken to be a prime between four hundred and one thousand. It should be noted that there are instantiations that use other rings such as $\mathbb{Z}[\boldsymbol{x}]/(\boldsymbol{x}^{2^k} + 1)$ and $\mathbb{Z}[\boldsymbol{x}]/(\boldsymbol{x}^N - \boldsymbol{x} - 1)$ – we will not consider these here.

The other public parameters are: q, an integer modulus defining the ring

$$R_{N,q} = R_N/(q) = (\mathbb{Z}/q\mathbb{Z})[\boldsymbol{x}]/(\boldsymbol{x}^N - 1);$$

p, an integer or polynomial of small degree; and subsets of R_N designating the key-, message- and blinding-spaces[1].

Specific considerations for the choice of q and p are discussed in Section 3.4.1. For now assume that they take the common values: q is a power of 2, $q \approx 8N$, and $p = 3$. We will examine several choices for the requisite subsets of R_N, all based around trinary polynomials with a fixed number of coefficients equal to ± 1. To facilitate this discussion we introduce the following sets:

Definition 17 (Common subsets of R_N)**.**

$$\begin{aligned}
\mathcal{T}_N &= \left\{ \text{ trinary polynomials in } R_N \ \right\} \\
\mathcal{T}_N(d, e) &= \left\{ \begin{array}{l} \text{trinary polynomials in } R_N \text{ with} \\ \text{exactly } d \text{ ones and } e \text{ minus ones} \end{array} \right\} \\
\mathcal{P}_N(d_1, d_2, d_3) &= \left\{ \begin{array}{l} \text{product form polynomials in } R_N \\ \boldsymbol{a}_1\boldsymbol{a}_2 + \boldsymbol{a}_3 : \boldsymbol{a}_i \in \mathcal{T}_N(d_i, d_i) \end{array} \right\}.
\end{aligned}$$

The last of these is optional. Product form keys and blinding polynomials were introduced in [41] and are covered by U.S. Patent No. 7031468 until August 2021.

An NTRUEncrypt private key consists of a pair of elements $(\boldsymbol{f}, \boldsymbol{g})$ of R_N, each drawn from one of these sets. A parameter set will specify whether the private key element $\boldsymbol{f}$ is of product form or not. If it is then $\boldsymbol{f}$ is chosen as $\boldsymbol{f} = 1 + p\boldsymbol{F}$, otherwise $\boldsymbol{F} \in \mathcal{P}_N(d_1, d_2, d_3)$ or $\boldsymbol{F} \in \mathcal{T}_N(d_f, d_f)$.

[1] Blinding polynomials are used for ciphertext randomization

The element $\boldsymbol{g}$ is chosen from $\mathcal{T}_N(d_g+1, d_g)$. In all cases, d_f, d_g, d_1, d_2, and d_3 are chosen such that an element drawn from one of the aforementioned sets will have roughly $N/3$ coefficients equal to $+1$, $N/3$ coefficients equal to 0, and $N/3$ coefficients equal to -1. Product form elements may end up with coefficients outside of $\{-1, 0, 1\}$, however they will still be expected to have 1-norm of roughly $2N/3$.

An NTRUEncrypt public key is $\boldsymbol{h} \in R_N$ with coefficients in $[-q/2, q/2)$ such that

$$\boldsymbol{f}\boldsymbol{h} \equiv \boldsymbol{g} \pmod{q}.$$

Well chosen parameter sets ensure that $\boldsymbol{h}$ exists with overwhelming probability with respect to a uniform choice of $(\boldsymbol{f}, \boldsymbol{g})$ in the private key space. Equivalently, well chosen parameter sets ensure that almost all choices of $\boldsymbol{f}$ are invertible in $R_{N,q}$.

As NTRUEncrypt is a public key encryption scheme its essential primitives are key generation, encryption, and decryption. We have essentially covered key generation already, but the basic sketch of the process is shown in Algorithm 1 for completion.

Algorithm 1 NTRUEncrypt Key Generation

Input: A full set of NTRUEncrypt parameters.

repeat
  $\boldsymbol{F} \leftarrow_{\$} \mathcal{P}_N(d_1, d_2, d_3)$ or $\mathcal{T}_N(d_f, d_f)$
  $\boldsymbol{f} = 1 + p\boldsymbol{F} \in R_{N,q}$
until $\boldsymbol{f}$ is invertible in $R_{N,q}$
repeat
  $\boldsymbol{g} \leftarrow_{\$} \mathcal{T}_N(d_g+1, d_g)$
until $\boldsymbol{g}$ is invertible in $R_{N,q}$
$\boldsymbol{h} = \boldsymbol{f}^{-1}\boldsymbol{g} \in R_{N,q}$

Output: Private key $(\boldsymbol{f}, \boldsymbol{g})$, Public key $(1, \boldsymbol{h})$

We will delay giving algorithmic descriptions of encryption and decryption until Section 3.3.3, as there are some details that must be covered before we can sketch primitives that are secure. Here we give just a flavor of how the operations work.

The encryption procedure will output elements of $R_N/(p)$, hence the message space must be some set of coset representatives for $(p)R_N$ in R_N. For $p = 3$ we select the set of minimal norm representatives: the trinary polynomials[2] $\mathcal{T}_N$.

[2]In practice we will restrict to a subset of fixed hamming weight, see Section 3.4.3.

Encryption of $\boldsymbol{m} \in \mathcal{T}_N$ to the public key $\boldsymbol{h}$ is a randomized process that involves selecting a "blinding polynomial" from one of the sets of Definition 17. A typical instantiation will define the set of blinding polynomials as either $\mathcal{T}_N(d_f, d_f)$ or $\mathcal{P}_N(d_1, d_2, d_3)$. The encrypting party selects the blinding polynomial, $\boldsymbol{r}$, uniformly at random from this set and computes the ciphertext as

$$\boldsymbol{c} = p\boldsymbol{r}\boldsymbol{h} + \boldsymbol{m}.$$

Decryption is deterministic, but is not be guaranteed to succeed for all parameterizations. The decrypting party computes

$$\boldsymbol{f}\boldsymbol{c} \equiv p\boldsymbol{r}\boldsymbol{g} + \boldsymbol{f}\boldsymbol{m} \pmod{q},$$

then computes the minimal norm representative of $\boldsymbol{f}\boldsymbol{c} + (q)$ and calls this $\boldsymbol{a}$. If the parameters have been chosen well this satisfies

$$\boldsymbol{a} \equiv \boldsymbol{f}\boldsymbol{m} \pmod{p},$$

with overwhelming probability over the encrypting party's choice of $\boldsymbol{r}$. Supposing that this is the case, from $\boldsymbol{f} = 1 + p\boldsymbol{F}$ we obtain

$$\boldsymbol{a} \equiv (1 + p\boldsymbol{F})\boldsymbol{m} \equiv \boldsymbol{m} \pmod{p},$$

thus the decryption process has succeeded in recovering $\boldsymbol{m}$.

Note that it is possible to instantiate NTRUEncrypt with an arbitrary $\boldsymbol{f}$ that is invertible modulo p. In this case one must perform a final multiplication by $\boldsymbol{f}^{-1} \bmod p$ to recover $\boldsymbol{m}$.

3.2 The NTRU Lattice

NTRUEncrypt is readily described as a lattice based encryption scheme in which the main objects interest are R_N-module lattices of the type given by Equation 2.9. Specifically, the lattice associated to a user's key is

$$\Lambda_q\left((\boldsymbol{f}, \boldsymbol{g})\right) = \{(\boldsymbol{a}, \boldsymbol{b}) \in R_N^2 \ : \ \exists \boldsymbol{s} \in R_N, \ \boldsymbol{s} \cdot (\boldsymbol{f}, \boldsymbol{g}) \equiv (\boldsymbol{a}, \boldsymbol{b}) \pmod{q}\}.$$

The following lemma encapsulates some basic facts about these lattices.

Lemma 8. *Suppose $\boldsymbol{f}$ and $\boldsymbol{g}$ are invertible modulo q. Let $\boldsymbol{f}_q \in R_N$ be such that $\boldsymbol{f}\boldsymbol{f}_q \equiv 1 \pmod{q}$, and $\boldsymbol{h} \in R_N$ be such that of $\boldsymbol{h} \equiv \boldsymbol{g}\boldsymbol{f}_q \pmod{q}$. Then*

$$\Lambda_q((\boldsymbol{f}, \boldsymbol{g})) = \Lambda_q((1, \boldsymbol{h}))$$

and is an R_N-module lattice of rank $2N$ and volume q^N.

Proof. We will handle the equality of the two lattices first. We have

$$\begin{aligned}
\Lambda_q((\boldsymbol{f}, \boldsymbol{g})) &= \{(\boldsymbol{a}, \boldsymbol{b}) \ : \ \exists \boldsymbol{s} \in R_N, \ \boldsymbol{s} \cdot (\boldsymbol{f}, \boldsymbol{g}) \equiv (\boldsymbol{a}, \boldsymbol{b}) \pmod{q}\} \\
&= \{(\boldsymbol{a}, \boldsymbol{b}) \ : \ \exists \boldsymbol{s} \in R_N, \ (\boldsymbol{s}\boldsymbol{f}) \cdot (1, \boldsymbol{g}\boldsymbol{f}_q) \equiv (\boldsymbol{a}, \boldsymbol{b}) \pmod{q}\} \\
&= \{(\boldsymbol{a}, \boldsymbol{b}) \ : \ \exists \boldsymbol{s}' \in R_N, \ \boldsymbol{s}' \cdot (1, \boldsymbol{h}) \equiv (\boldsymbol{a}, \boldsymbol{b}) \pmod{q}\} \\
&= \Lambda_q((1, \boldsymbol{h})).
\end{aligned}$$

In the third line we have used the fact that if $\boldsymbol{s} \in R_N$ then so is $\boldsymbol{s}\boldsymbol{f}$.

For the rank $2N$ claim, one can easily verify that the rows of

$$\begin{pmatrix} \mathsf{Circ}(1) & \mathsf{Circ}(\boldsymbol{h}) \\ \mathsf{Circ}(q) & 0 \\ 0 & \mathsf{Circ}(q) \end{pmatrix} \tag{3.1}$$

are a spanning set for $\Lambda_q((1, \boldsymbol{h}))$. However, since $(q, 0) = q \cdot (1, \boldsymbol{h}) - \boldsymbol{h} \cdot (0, q)$ the $2N$ rows of

$$H = \begin{pmatrix} \mathsf{Circ}(1) & \mathsf{Circ}(\boldsymbol{h}) \\ 0 & \mathsf{Circ}(q) \end{pmatrix} \tag{3.2}$$

are sufficient, and the lattice is of rank at most $2N$. Finally, the lattice must be of rank at least $2N$, hence equal to $2N$, since $q\mathbb{Z}^{2N} \subseteq \Lambda_q((1, \boldsymbol{h}))$.

Finally, for the volume, since H is a basis for the lattice the volume of the lattice is given by

$$\det(HH^T)^{1/2} = \det(H) = q^N.$$

□

One may also check that if $\boldsymbol{h}$ is chosen as the unique representative of $\boldsymbol{g}\boldsymbol{f}_q + (q)$ with coefficients in $[0, q-1]$, then the matrix 3.2 is in Hermite normal form by Definition 10.

The choices of $\boldsymbol{f}$ and $\boldsymbol{g}$ suggested above lead to lattices that are somewhat atypical among lattices of rank $2N$ and volume q^N. The Gaussian heuristic (Definition 7) estimates the first minima of $L = \Lambda_q((\boldsymbol{f}, \boldsymbol{g}))$ as

$$\mathrm{GH}(L) = \sqrt{\frac{2N}{2\pi\epsilon}} \det(L)^{1/2N} = \sqrt{\frac{qN}{\pi\epsilon}},$$

however we know that there are vectors in the lattice of norm

$$\|(\boldsymbol{f}, \boldsymbol{g})\|_2 \approx \sqrt{4N/3}.$$

Assuming that $q \approx 8N$ and that $\lambda_1(L) = \|(\boldsymbol{f}, \boldsymbol{g})\|$ then we see that the shortest vectors of NTRU lattices are unusually short. We have $\mathrm{GH}(L)/\lambda_1(L) \approx \sqrt{N}$, whereas for random lattices we would expect this quantity to be approximately 1.

3.3 Standardized NTRUEncrypt

The simple "textbook" variant of NTRUEncrypt is not fit for use on the Internet or in other potentially adversarial environments. This is commonly the case with public key cryptosystems – it would be inadvisable to use unpadded RSA, for instance. For production systems one should implement NTRUEncrypt as specified in the most recent version of the Efficient Embedded Security Standard #1, currently v3.1 [2]. The standard specifies, in particular, the Short Vector Encryption Scheme (SVES), a CCA-2 secure variant of NTRUEncrypt. It also standardizes auxiliary functions, some of which are security critical, and provides explicit parameter sets.

3.3.1 Additional parameters

This section provides a list of the additional parameters needed for SVESfor reference in the following sections.

SVES Parameter (d_m)**.** *The minimum number of* $+1$ *coefficients,* -1 *coefficients, and* 0 *coefficients that a polynomial* $\boldsymbol{m}$ *must have in order to be used for* SVES *encryption.*

SVES Parameter ($igfC$)**.** *A constant depending on* N *such that there exists an integer* k *for which* $kN/2^c$ *is close to* 1*. Used for sampling uniformly from* $[0, N-1]$*.*

SVES Parameter ($minCallsMask$)**.** *The number of calls that the Mask Generation Function must make to ensure a negligible probability of exhausting its bit pool while generating a mask.*

SVES Parameter ($minCallsR$)**.** *The number of calls that the Index Generation Function must make to ensure a negligible probability of exhausting its bit pool while generating a blinding polynomial. Depends on* $igfC$*.*

Remark. Both $minCallsMask$ and $minCallsR$ serve to prevent attacks based on timing the decryption process, such as that presented in [73].

SVES Parameter ($bLen$)**.** *The number of octets provided to the Blinding Polynomial Generation Function for ciphertext randomization.*

Remark. Historically $bLen$ has been taken to be equal to the security parameter, however in recent parameter sets [37] it has been taken to be twice that so as to resist quantum attacks.

SVES Parameter ($mLen$)**.** *The maximum length of a plaintext in bits.*

SVES Parameter ($hTruncLen$)**.** *The number of bits of the public key to include in calls to the Blinding Polynomial Generation Function.*

Remark. Realistically, $bLen$, $mLen$, and $hTruncLen$ will always be chosen as a multiple of 8, and it might be better to define these explicitly as a number of octets instead of bits.

3.3.2 Support functions

Mask Generating Function

The use of $\mathbb{Z}[\boldsymbol{x}]/(\boldsymbol{x}^N - 1)$ rather than, say, $\mathbb{Z}[\boldsymbol{x}]/(\Phi_N)$ requires that we be careful not to leak information through the ring homomorphism $\mathbb{Z}[\boldsymbol{x}]/(\boldsymbol{x}^N - 1) \to \mathbb{Z}[\boldsymbol{x}]/(\boldsymbol{x} - 1)$ given by evaluation at 1.

The blinding polynomial $\boldsymbol{r}$ is generated such that $\boldsymbol{r}(1) = 0$. Hence a ciphertext $\boldsymbol{c} \equiv p\boldsymbol{r}\boldsymbol{h} + \boldsymbol{m} \pmod{q}$ satisfies

$$\begin{aligned} \boldsymbol{c}(1) &\equiv p \cdot \boldsymbol{r}(1) \cdot \boldsymbol{h}(1) + \boldsymbol{m}(1) \pmod{q} && (3.3)\\ &\equiv p \cdot 0 \cdot \boldsymbol{h}(1) + \boldsymbol{m}(1) \pmod{q} && (3.4)\\ &\equiv \boldsymbol{m}(1) \pmod{q}. && (3.5) \end{aligned}$$

As a consequence, the sketch of NTRUEncrypt provided above is not even IND-CPA secure – an adversary can simply select two messages such that $\boldsymbol{m}(1) \neq \boldsymbol{m}'(1)$ and they will be able to distinguish their encryptions. In order for NTRUEncrypt to satisfy any reasonable definition of security it is necessary that messages be padded prior to encryption.

A Mask Generating Function (MGF) for a set $\mathcal{S}$ takes as input a string $seed$ of $seedLen$ bits (potentially $\ll \log_2(|\mathcal{S}|)$) and outputs an element of $\mathcal{S}$ "uniformly at random." Of

course, no actual function can satisfy this definition, but one can reasonably approximate an ideal MGF using a hash function. EESS #1 defines a mask generating function, MGF-TP-1 (Algorithm 2), for the set of trinary polynomials, $\mathcal{T}_N$.

Roughly, rather than computing the ciphertext as $\boldsymbol{c} = p\boldsymbol{rh} + \boldsymbol{m}$ the encrypting party first encodes $p\boldsymbol{rh}$ as a bitstring R, and computes $\boldsymbol{m}' \in \mathcal{T}_N$ such that

$$\boldsymbol{m}' \equiv \boldsymbol{m} + \text{MGF-TP-1}(R) \pmod{p}.$$

This final value is used to produce the ciphertext in the typical manner $\boldsymbol{c} = p\boldsymbol{rh} + \boldsymbol{m}'$.

The ordinary decryption procedure recovers $\boldsymbol{m}'$ and consequently $p\boldsymbol{rh}$. The decrypting party is then able to compute $\boldsymbol{m} = \boldsymbol{m}' - \text{MGF-TP-1}(\text{“}p\boldsymbol{rh}\text{”}) \bmod p$.

It is essential that MGF-TP-1 operate in constant time. Hence you will see in Algorithm 2 that seed expansion is performed at the beginning of the routine. The amount of expansion, controlled by $minCallsMask$ must be sufficient to ensure that the pool of bits is large enough to generate a trinary polynomial of degree N with all but negligible probability.

The appropriate value of $minCallsMask$ depends on the output length of the hash function. Five uniform trits can be extracted from eight uniform bits with probability $3^5/2^8 = 243/256$. Hence for an ℓ bit output $minCallsMask$ should be the smallest k for which

$$\sum_{i=0}^{\lfloor N/5 \rfloor} \binom{k\ell/8}{i} \left(\frac{13}{256}\right)^i \left(\frac{243}{256}\right)^{k\ell/8-i}$$

is negligible.

Algorithm 2 MGF-TP-1

Input: $seed \in \{0,1\}^*, minCallsMask, N$

$Z = \mathsf{Hash}(seed)$
$buf = \mathsf{Hash}(Z|0) \mid \mathsf{Hash}(Z|1) \mid \ldots \mid \mathsf{Hash}(Z|minCallsMask)$
$\boldsymbol{v} = 0 \in \mathbb{Z}[x]$
$i = 0$
for each octet O in buf **do**
  if $O < 3^5$ **then**
    Choose $(c_0, \ldots, c_4) \in \{0,1,2\}^5$ such that $\sum_{j=0}^{4} c_j \cdot 3^j = O$
    Let $c'_j \in \{-1, 0, 1\}$ be the minimal integer representative of c_j mod 3.
    $\boldsymbol{v} = \boldsymbol{v} + c'_0\boldsymbol{x}^i + \cdots + c'_4\boldsymbol{x}^{i+4}$
    $i = i + 5$
  end if
end for

Output: $\boldsymbol{v}$ mod $\boldsymbol{x}^N$ {Truncate $\boldsymbol{v}$ to its first N coefficients}

Remark. Several small details have been omitted from Algorithm 2. See [2] for a complete specification. Note that the procedure expands the seed to a fixed length up front. This is to avoid timing attacks that analyze the number of calls made to the hash function during mask generation. See section 3.4.5 for details.

Index Generating Function

In order to generate both keys and blinding polynomials one must be able to sample uniformly from $\mathcal{T}_N(d, e)$. EESS #1 defines routines for such sampling that make use of an Index Generating Function, IGF-2. An Index Generating Function is a method for sampling a fixed size subset of $[0, N-1]$ – a procedure that is slightly more difficult to do in constant time than mask generation since the indices cannot be sampled independently.

While there are a variety of possible techniques, it can be tedious to determine probabilistic upper bounds on the number of bits any given technique will consume. In order to avoid timing attacks the procedure must consume a fixed amount of randomness regardless of the subset generated, in particular it must expand the input *seed* to a fixed length via a fixed number of calls to the hash function.

Algorithm 3 IGF-2

Input: $seed \in \{0,1\}^*, minCallsR, igfC, N$

$Z = \mathsf{Hash}(seed)$
$buf = \mathsf{Hash}(Z|0) \mid \mathsf{Hash}(Z|1) \mid \ldots \mid \mathsf{Hash}(Z|minCallsR)$
$indices = \varnothing$
for each $igfC$-bit unsigned integer k in buf **do**
  if $k < 2^{igfC} - (2^{igfC} \bmod N)$ **then**
    $n = k \bmod N$ {in $[0, N-1]$}
    if $n \notin indices$ **then**
      $indices = indices \cup k$
    end if
  end if
end for

Output: $indices$

As simple as generating random elements of $\mathcal{T}_N(d,e)$ may seem, there are perhaps some open problems here. It has been noted[3] that a naive method of tracking used indices, e.g. initializing a list of zeros of length N and switching entries to 1 after they has been selected, could leak the selected subset through a cache timing attack. The obvious alternative is to store a list of the selected elements, then perform a pairwise comparison with each new candidate for entry. This $O(N^2)$ variant comes with a significant performance degradation, particularly for parameter sets that use trinary, rather than product-form, elements. It may be worthwhile to consider a different IGF entirely.

The parameter $minCallsR$ is derived in Section 3.4.5.

Other functions

Before describing SVES we need two more utility function, one

$$\mathsf{Encode}_N : \{0,1\}^{bLen(N)} \times \{0,1\}^{mLen(N)} \to \mathcal{T}_N,$$

for converting bit strings to trinary ring elements. Unlike the mask generating function, Encode_N makes no attempt to produce coefficients that are uniformly distributed in $\{-1,0,1\}$. It does however need to be an invertible transformation, and we will denote

[3] Scott Fluher, private communication. November 2015

its inverse by Decode_N. In actuality, Encode_N also takes a short string representing the parameter set as input, but we will omit this detail here.

The second utility function, BGF_N, uses $\mathsf{IGF\text{-}2}$ to produce a blinding polynomial from a bitstring and the public key,

$$\mathsf{BGF}_N : \{0,1\}^{bLen(N)} \times \{0,1\}^{mLen(N)} \times R_N \to \mathcal{S},$$

where $\mathcal{S}$ is either $\mathcal{T}_N(d_f, d_f)$ or $\mathcal{P}_N(d_1, d_2, d_3)$ depending on the parameter set. The trivial implementation is satisfactory; assuming $\mathsf{IGF\text{-}2}$ provides its output in random order, simply take the first, say, d_f coefficients from $\mathsf{IGF\text{-}2}$ as the indices of the positive coefficients and the next d_f coefficients as the indices of the negative coefficients.

3.3.3 SVES

SVES key generation is identical to Algorithm 1. Encryption and Decryption are described in Algorithms 4 and 5.

Algorithm 4 NTRUEncrypt SVES Encryption (sketch)

Input: Public key $\boldsymbol{h}$, message $M \in \{0,1\}^{mLen}$, and a parameter set with $p = 3$.

1: **repeat**
2: $\quad b \leftarrow_{\$} \{0,1\}^{bLen}$
3: $\quad \boldsymbol{m}' = \mathsf{Encode}_N(b, M)$
4: $\quad \boldsymbol{r} = \mathsf{BGF}_N(b, M, \boldsymbol{h})$
5: $\quad \boldsymbol{R} = p \cdot \boldsymbol{r} \cdot \boldsymbol{h} \bmod q$
6: $\quad \boldsymbol{v} = \mathsf{MGF\text{-}TP\text{-}1}(\boldsymbol{R})$
7: $\quad \boldsymbol{m} = \boldsymbol{m}' + \boldsymbol{v} \bmod p$
8: **until** The number of $+1$s, -1s, and 0s in $\boldsymbol{m}$ are each $\geq d_m$.
9: $\boldsymbol{c} = \boldsymbol{R} + \boldsymbol{m}$

Output: Ciphertext $\boldsymbol{c}$

Algorithm 5 NTRUEncrypt SVES Decryption (sketch)

Input: Key pair $((\boldsymbol{f}, \boldsymbol{g}), (1, \boldsymbol{h}))$, ciphertext $\boldsymbol{c} \in R_{N,q}$, and a parameter set with $p = 3$.

$\boldsymbol{m} = \boldsymbol{f}\boldsymbol{c} \bmod p$
$\boldsymbol{R} = \boldsymbol{c} - \boldsymbol{m}$
$\boldsymbol{v} = \mathsf{MGF\text{-}TP\text{-}1}(R)$ where R is a bitstring representing $\boldsymbol{R}$
$\boldsymbol{m}' = \boldsymbol{m} - \boldsymbol{v} \bmod p$
$(b, M) = \mathsf{Decode}_N(\boldsymbol{m}')$
$\boldsymbol{r} = \mathsf{BGF}_N(b, M, \boldsymbol{h})$
if $p\boldsymbol{r}\boldsymbol{h} = \boldsymbol{R} \bmod q$ and the number of $+1$s, -1s, and 0s in $\boldsymbol{m}$ are each $\geq d_m$ **then**
  $result = M$
else
  $result = \perp$
end if

Output: $result$

3.4 SVES Parameters

3.4.1 Choice of N, q, and p

The earliest NTRUEncrypt parameter sets, those in the manuscript circulated at Crypto '96 and in the paper from ANTS '98 [34] [39], make use of prime N. Sophie-Germain prime, in particular, were recommended as these would maximize the probability that $\boldsymbol{f}$ and $\boldsymbol{g}$ were invertible modulo any prime factors of q or p. It was later suggested, in a technical report on parameter selection [72], that N could be taken to be a power of 2 as this would speed the computation of convolutions using FFT-based techniques. Gentry quickly demonstrated that composite N were inadmissible [28].

For all $\boldsymbol{\phi}$ that divide $\boldsymbol{x}^N - 1$ there is a ring homomorphism $R_N \to \mathbb{Z}[x]/(\boldsymbol{\phi})$, and one must consider this structure when evaluating the complexity of problems such as SVP and CVP in R_N-module lattices. It was observed by Gentry that, for composite N, there are ring homomorphisms $\Theta_d : R_N \to \mathbb{Z}[\boldsymbol{x}]/(\boldsymbol{x}^d - 1)$ for $d|N$ that have the potential to preserve a significant amount of geometric information. Provided that N/d is small, short elements will be mapped to short elements, and it may be possible to recover the secret key using lattice reduction in dimension $2d$ [28].

As a consequence, NTRUEncrypt is always instantiated with prime N. For such N the

101,	139,	149,	163,	173,	197,	211,	269,	293,	317
379,	389,	461,	509,	557,	653,	677,	701,	773,	797
821,	859,	907,	941,	1061,	1109,	1123,	1229,	1277,	1291
1301,	1373,	1483,	1493,	1637,	1733,	1747,	1901,	1949,	1973

Table 3.1: First 40 primes $N > 100$ for which $\mathrm{ord}_{(\mathbb{Z}/N\mathbb{Z})^*}(2) = \mathrm{ord}_{(\mathbb{Z}/N\mathbb{Z})^*}(3) = (N-1)$

ring modulus factors into irreducibles over $\mathbb{Q}$ as

$$\boldsymbol{x}^N - 1 = (\boldsymbol{x} - 1)\Phi_N = (\boldsymbol{x} - 1)(\boldsymbol{x}^{N-1} + \cdots + \boldsymbol{x} + 1).$$

The probability that a random element of R_N is invertible modulo q is maximized when Φ_N is irreducible modulo q. In fact, in this case, an element $\boldsymbol{v}$ will be invertible in $R_{N,q}$ provided that $\boldsymbol{v}(1) \not\equiv 0 \pmod{q}$. As we saw in Section 2.4, if e is the order of r in $(\mathbb{Z}/N\mathbb{Z})^*$ then Φ_N modulo r has e factors of degree $(N-1)/e$ for any prime r. Thus we should choose N, p, and q in such a manner as to ensure that the order of each prime factor of pq is large, say equal to $(N-1)$ or $(N-1)/2$, in $(\mathbb{Z}/N\mathbb{Z})^*$. This will ensure that there is a negligible probability of failure per loop iteration in Algorithm 1. Similar considerations apply for other choices of q.

The only other restrictions on p and q are that they generate coprime ideals of R_N. If p and q are both integers then we simply require that $\gcd(p, q) = 1$. For SVES we fix $p = 3$ and only consider q that are a power of 2. This choice is motivated by the need for fast arithmetic modulo q, and by the impact of p on decryption failure probability (see Section 3.4.4).

Table 3.1 contains a list of primes suitable for use as N when $p = 3$ and q is a power of 2. Appendix A.3 contains two larger lists that ignore the order of 3 in $(\mathbb{Z}/N\mathbb{Z})^*$. When $\boldsymbol{f}$ is of the form $1 + p\boldsymbol{F}$, these latter lists provide more flexibility in the choice of N without increasing the risk of generating non-invertible $\boldsymbol{f}$.

The justification for a choice of N and q is largely up to security considerations; we will address these in Chapter 5.

3.4.2 Private key parameters

We will only consider private keys that are drawn from one of the sets in Definition 17.

In order to maximize the size of the key space, while keeping a prescribed number of ± 1s in g, we take

$$d_g = \lfloor N/3 \rceil,$$

and for non-product form $\boldsymbol{f}$ we select $d_f = d_g$. For product form $\boldsymbol{f}$ we would like the one-norm of $\boldsymbol{F}_1\boldsymbol{F}_2 + \boldsymbol{F}_3$ to be roughly $2N/3$. A simple calculation shows that it is expected to be $\approx 4d_1d_2 + 2d_3$. So we let $\alpha \in \mathbb{R}$ be the unique positive root of

$$2x^2 + x - N/3,$$

and take d_1 to be $\lceil \alpha \rceil$, i.e.

$$d_1 = \left\lceil \sqrt{\frac{3+8N}{48}} - \frac{1}{4} \right\rceil.$$

The choice of d_2 and d_3 is now somewhat arbitrary. But keeping with our goal of having $2d_1d_2 + d_3 \approx N/3$ we might take

$$d_2 = \left\lceil \frac{N}{6d_1} - \frac{1}{2} \right\rceil$$

and

$$d_3 = \max\left(\left\lceil \frac{N}{3} - 2d_1d_2 \right\rceil, \left\lceil \frac{d_1}{2} + \frac{1}{2} \right\rceil \right).$$

3.4.3 Minimum message weight

Since ciphertexts reveal the value of $\boldsymbol{m}(1)$ (see Eq. 3.3), we must ensure that extreme values of $\boldsymbol{m}(1)$ do not help an adversary launch a plaintext recovery attack. A very small (or large) value for $\boldsymbol{m}(1)$ signals that the message has a low (resp. high) number of coefficients equal to $+1$, and this could be exploited by an adversary to reduce the combinatorial search space for the message. In order to constrain $\boldsymbol{m}(1)$ to a narrow range, the SVES encryption routine will reject a ciphertext and generate a new one with fresh randomness unless sufficiently many coefficients take each value in $\{-1, 0, 1\}$.

The choice of d_m also affects the decryption failure probability. Without the message weight restriction an adversary could attempt to increase their probability of triggering a decryption failure by searching for valid ciphertexts with large Hamming weight messages.

Finally the choice of d_m affects the efficiency of the scheme, as a violation of the d_m constraint will force an entirely new ciphertext to be generated.

Let

$$I(d_m) = \{(i,j) : d_m \le i < (N - 2d_m),\ d_m \le j < (N - d_m - i)\}.$$

For efficiency we will only consider d_m satisfying:

$$2^{-10} \geq 1 - 3^{-N} \left(\sum_{(i,j) \in I(d_m)} \binom{N}{i} \binom{N-i}{j} \right). \tag{3.6}$$

Security considerations will force us to take the maximum value satisfying Eq. 3.6.

3.4.4 Probability of Decryption Failure in SVES†

Having fixed the parameters we can finally give a meaningful probability that the SVES decryption algorithm succeeds.

In order for the decryption of $\boldsymbol{c} \equiv p\boldsymbol{r}\boldsymbol{h} + \boldsymbol{m} \pmod q$ to succeed, it must be the case that reducing $\boldsymbol{c}\boldsymbol{f}$ modulo q recovers $p\boldsymbol{r}\boldsymbol{g} + \boldsymbol{m}\boldsymbol{f}$ exactly. A sufficient condition is that

$$\|p\boldsymbol{r}\boldsymbol{g} + \boldsymbol{m}\boldsymbol{f}\|_\infty < q/2. \tag{3.7}$$

By the triangle inequality we have

$$\|p\boldsymbol{r}\boldsymbol{g} + \boldsymbol{m}\boldsymbol{f}\|_\infty \leq \|p\|_1 \|\boldsymbol{r}\|_1 \|\boldsymbol{g}\|_\infty + \|\boldsymbol{f}\|_1 \|\boldsymbol{m}\|_\infty, \tag{3.8}$$

and specifying the spaces from which each term is drawn leads us to Fact 9.

Fact 9. *Suppose $p \in \mathbb{Z}$, $\boldsymbol{g} \in \mathcal{T}_N$, and $\boldsymbol{m} \in \mathcal{T}_N$. Then decryption succeeds with certainty if*

$$q > 2(p\|\boldsymbol{r}\|_1 + \|\boldsymbol{f}\|_1).$$

For a parameterization in which product form elements are drawn from $\mathcal{P}_N(d_1, d_2, d_3)$, trinary elements are drawn from $\mathcal{T}_N(\lfloor N/3 \rceil, \lfloor N/3 \rceil)$, and $\boldsymbol{f} = 1 + p\boldsymbol{F}$ this is equivalent to:

$$\begin{aligned} \textit{(product form } \boldsymbol{r}, \boldsymbol{F}\textit{)} \quad & q > 8p(2d_1 d_2 + d_3) + 2, \\ \textit{(trinary } \boldsymbol{r}, \boldsymbol{F}\textit{)} \quad & q > 8p\lfloor N/3 \rceil + 2. \end{aligned}$$

These bounds are reasonably tight, in that we can exhibit $\boldsymbol{r}$ and $\boldsymbol{m}$ that would cause a decryption failure with any q violating the relevant inequality. However such pairs are extremely rare, and using SVES the decrypting party can be reasonably assured that even an adversarial encrypting party would have had no control over the choice of $\boldsymbol{r}$ or $\boldsymbol{m}$, except indirectly through the choice of b in Line 2 of Algorithm 4. As such, it is fair to model $\boldsymbol{r}$ and $\boldsymbol{m}$ as being uniformly distributed over their appropriate sample spaces. By

doing so it becomes meaningful to ask what the probability of decryption failure is for a given choice of q.

Furthermore, since ciphertext expansion scales roughly as $N \log_2(q)$, and ciphertext size is one of the main drawbacks of NTRUEncrypt, it is advantageous to consider whether q can be decreased by allowing a small probability of decryption failure.

We will argue that the probability

$$\text{Prob}\,(\text{ a given coefficient of } \boldsymbol{rg} + \boldsymbol{mF} \text{ has absolute value } \geq c\ ) \tag{3.9}$$

can be analyzed rather well by an application of the central limit theorem. This was done for the case of trinary $\boldsymbol{r}, \boldsymbol{g}, \boldsymbol{m}, \boldsymbol{F}$ in [33].

In what follows suppose $\boldsymbol{r}$ and $\boldsymbol{F}$ are drawn uniformly from $\mathcal{P}_N(d_1, d_2, d_3)$, i.e. $\boldsymbol{r} = \boldsymbol{r}_1\boldsymbol{r}_2 + \boldsymbol{r}_3$ and $\boldsymbol{F} = \boldsymbol{F}_1\boldsymbol{F}_2 + \boldsymbol{F}_3$ with each of $\boldsymbol{r}_i$ and $\boldsymbol{F}_i$ having d_i coefficients equal to $+1$ and d_i coefficients equal to -1. Furthermore suppose $\boldsymbol{g}$ is drawn uniformly from $\mathcal{T}_N(d_g + 1, d_g)$, and $\boldsymbol{m}$ is drawn uniformly from $\mathcal{T}_N$ subject to the d_m constraint.

Let X denote the constant coefficient of $\boldsymbol{rg} + \boldsymbol{mF}$. The spaces from which $\boldsymbol{r}$ and $\boldsymbol{m}$ are drawn are invariant under permutations of indices, so the choice to analyze this particular coefficient is without loss of generality[4]. Each of the four summands in

$$X = (\boldsymbol{r}_1\boldsymbol{r}_2\boldsymbol{g})_0 + (\boldsymbol{r}_3\boldsymbol{g})_0 + (\boldsymbol{F}_1\boldsymbol{F}_2\boldsymbol{m})_0 + (\boldsymbol{F}_3\boldsymbol{m})_0, \tag{3.10}$$

has mean zero since $\boldsymbol{r}_i(1) = \boldsymbol{F}_i(1) = 0$. Furthermore each summand is itself a sum of either $4d_1d_2$ or $2d_3$ coefficients of $\boldsymbol{g}$ or $\boldsymbol{m}$ (with repetition allowed).

For instance,

$$(\boldsymbol{r}_1\boldsymbol{r}_2\boldsymbol{g})_0 = \sum_{i,j} (\boldsymbol{r}_1)_i (\boldsymbol{r}_2)_j (\boldsymbol{g})_{-(i+j)}$$

and only the $4d_1d_2$ pairs of indices corresponding to non-zero coefficients of $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ contribute to the sum. We can think of each index pair as selecting a sign $\epsilon(i)$ and an index $a(i)$ and rewrite the sum as

$$(\boldsymbol{r}_1\boldsymbol{r}_2\boldsymbol{g})_0 = \sum_{i=1}^{4d_1d_2} \epsilon(i)(\boldsymbol{g})_{a(i)}.$$

[4]Note that while the individual coefficients have the same distribution, they are not independent. Eventually we will argue, by a union bound, that the probability of decryption failure is less than or equal to N times the probability that a single coefficient is too large. So we can ignore inter-coefficient dependencies in our analysis.

While the terms in this sum are not formally independent (since the indices $a(i)$ may not be distinct, and $\boldsymbol{g}$ has a prescribed number of non-zero coefficients) extensive experiments show that the variance of $(\boldsymbol{r}_1\boldsymbol{r}_2\boldsymbol{g})_0$ is still well approximated by treating $(\boldsymbol{g})_{a(i)}$ as a random coefficient of $\boldsymbol{g}$, i.e. as taking a non-zero value with probability $(2d_g+1)/N$. Doing this we calculate the variance as:

$$\mathbb{E}\left[(\boldsymbol{r}_1\boldsymbol{r}_2\boldsymbol{g})_0^2\right] \approx \sum_{i=1}^{4d_1d_2} \mathbb{E}\left[(\epsilon(i)(\boldsymbol{g})_{a(i)})^2\right] = \sum_{i=1}^{4d_1d_2} \mathbb{E}\left[(\boldsymbol{g})_{a(i)}^2\right] = 4d_1d_2 \cdot \frac{2d_g+1}{N}.$$

Identical arguments can be applied to approximate the variances of the other terms of Eq. 3.10. Some care must be taken with the terms involving $\boldsymbol{m}$ as it could be chosen adversarily to maximize its Hamming weight and hence the probability of a decryption failure. Due to the d_m constraint (Section 3.4.3), the number of non-zero coefficients of $\boldsymbol{m}$ cannot exceed $N - d_m$. As such we model the coefficients of $\boldsymbol{m}$ as taking ± 1 each with probability $(1 - d_m/N)$ and 0 with probability d_m/N.

With these considerations the variances of the four summands are found to be

$$\sigma_1^2 = \mathbb{E}\left[(\boldsymbol{r}_1\boldsymbol{r}_2\boldsymbol{g})_0^2\right] = 4d_1d_2 \cdot \frac{2d_g+1}{N}, \qquad \sigma_3^2 = \mathbb{E}\left[(\boldsymbol{F}_1\boldsymbol{F}_2\boldsymbol{m})_0^2\right] = 4d_1d_2 \cdot \frac{N-d_m}{N},$$
$$\sigma_2^2 = \mathbb{E}\left[(\boldsymbol{r}_3\boldsymbol{g})_0^2\right] = 2d_3 \cdot \frac{2d_g+1}{N}, \qquad \sigma_4^2 = \mathbb{E}\left[(\boldsymbol{F}_3\boldsymbol{m})_0^2\right] = 2d_3 \cdot \frac{N-d_m}{N}.$$

Using our assumption that it is reasonable to treat the coefficients of each summand as being independent and identically distributed, the central limit theorem suggests that each summand will be normally distributed. The distribution of X is then given by the convolution of the four constituent normal distributions, so it will also be normal with variance

$$\sigma^2 = \sigma_1^2 + \sigma_2^2 + \sigma_3^2 + \sigma_4^2 = (4d_1d_2 + 2d_3) \cdot \frac{N - d_m + 2d_g + 1}{N}. \tag{3.11}$$

Applying the same line of argumentation in the non-product form (trinary) case yields,

$$\sigma_5^2 = \mathbb{E}\left[(\boldsymbol{r}\boldsymbol{g})_0^2\right] = 2d_r \cdot \frac{2d_g+1}{N}, \qquad \sigma_6^2 = \mathbb{E}\left[(\boldsymbol{F}\boldsymbol{m})_0^2\right] = 2d_f \cdot \frac{N-d_m}{N}, \tag{3.12}$$

$$\sigma^2 = \sigma_5^2 + \sigma_6^2. \tag{3.13}$$

The probability that a normally distributed random variable with mean 0 and standard deviation σ exceeds c in absolute value is given by the complementary error function, specifically $\texttt{erfc}(c/(\sqrt{2}\sigma))$. Applying a union bound, the probability that any of the N coefficients of $\boldsymbol{r}\boldsymbol{g} + \boldsymbol{m}\boldsymbol{F}$ is greater than c is bounded above by $N \cdot \texttt{erfc}(c/(\sqrt{2}\sigma))$.

Decryption failure can only occur if a coefficient of $\boldsymbol{rg}+\boldsymbol{mF}$ exceeds $(q-2)/2$. Putting this all together we have Lemma 10 (marked with an asterisk due to the heuristics used above).

Lemma* 10. *For* SVES *with parameters* N, q, d_g, d_m *and* (df, dr) *or* (d_1, d_2, d_3)

$$\Pr\left[Decryption\ fails\right] \leq N \cdot \texttt{erfc}((q-2)/(2\sqrt{2} \cdot p \cdot \sigma)) \tag{3.14}$$

where σ *as in Eq. 3.11 (product form) or Eq. 3.13 (trinary).*

For an alternative instantiation identical to SVES *except for the use of* $\boldsymbol{f} = 1 + \boldsymbol{F}$

$$\Pr\left[Decryption\ fails\right] \leq N \cdot \texttt{erfc}((q-2)/(2\sqrt{2} \cdot \sigma)) \tag{3.15}$$

with $\sigma^2 = p^2(\sigma_1^2 + \sigma_2^2) + \sigma_2^2 + \sigma_4^2$ *or* $\sigma^2 = p^2\sigma_5^2 + \sigma_6^2$.

3.4.5 Number of IGF calls

Recall that in NTRU-SVES-Decrypt (Algorithm 5) the blinding polynomial, $\boldsymbol{r}$, is reconstructed from a ciphertext, $\boldsymbol{c} = p\boldsymbol{rh} + \boldsymbol{m}$, during decryption. This process involves querying the Index Generating Function with a value that depends on the private key (see Line 3 of Algorithm 5). Silverman and Whyte presented a timing attack on NTRUEncrypt that exploits non-constant time implementations of the Index Generating Function [73]; we defer to the original paper for details of the attack. The greatest source of non-constant time behavior in IGF comes from the rejection sampling required to sample a uniform random subset of $[0, N)$, hence from the variable number of hash functions required to produce a sufficiently long bitstring from the IGF input. The countermeasure recommended in [73] and in EESS #1 is simply to fix the number of hash function calls made as a public parameter. Doing so requires that we compute a number of uniform random bits sufficient extract $2d_r$ unique indices in $[0, N)$ with overwhelming probability.

In practice we let t and c be positive integers such that tN/c is less than, but close to, 1. So one uniform sample from $[0, c)$ can be converted into one uniform sample from $[0, N)$ with probability $1 - tN/c$. Silverman and Whyte define the probability

$$P_{c,N,t}(k, d) = \text{Prob}\begin{pmatrix}\text{A set of } k \text{ integers, each chosen uniformly in } [0, c) \\ \text{contains exactly } d \text{ integers in } [0, nN] \text{ whose} \\ \text{values are distinct modulo } N\end{pmatrix}.$$

If c is a power of two, then the probability that $k \cdot \log c$ random bits are insufficient to produce a d-subset of $[0, N)$ is $\sum_{i=1}^{d-1} P_{c,N,t}(k, d)$. Hence given N, t, c, and d we can search for k such that this cumulative probability is negligible in the security parameter.

Silverman and Whyte give a simple recursion for the probability $P_{c,N,t}(k,d)$ that can be evaluated via dynamic programming in time and space that is quadratic in k. While not overly burdensome, this algorithm can consume a significant amount of memory when used as a subroutine in a search for optimal parameters. As an alternative we present an approximation that can be computed with constant space. The approximation is of sufficient quality to completely supplant the exact method in the case of $N = c$. In other cases the approximation should be used to provide an upper bound on k before calling the exact routine.

Dupuis et al. [22] provide refined large deviation asymptotics for occupancy problems, in particular they study the large deviation probability that fewer than d out of N urns are occupied after k balls have been thrown into them at uniformly random.

Fix k, the number of balls to be thrown, N the total number of urns, and d the minimum number of urns that must be filled. Define $\theta = k/N$ and $\xi = d/N$. Let ρ be the unique positive root of

$$\theta = -\frac{1}{\rho}\log(1-\rho\xi),$$

and let

$$\sigma = \sqrt{\xi/(1-\rho\xi)-\theta},$$

and let

$$J(\theta) = (\theta-\xi)\log\rho + (1-\xi)\log(1-\xi) - \frac{(1-\rho\xi)}{\rho}\log(1-\rho\xi).$$

By [22, Theorem 2.1], for sufficiently large N, the probability that more than k balls would be required to occupy at least d urns is given by

$$p_1(k) \approx \frac{e^{-N\cdot J(\theta)}}{\sqrt{2\pi\sigma^2 N}}\left(\frac{\rho}{\rho-1}\right)\sqrt{\frac{1-\rho\xi}{1-\xi}}.$$

And in the limit as $N \to \infty$ the two sides are equal.

One can use this estimate to quickly search for a k for which $p_1(k)$ is negligible. As one typically only has access to random bits, and N is not typically a power of two, one must also calculate the number of bits required to generate k uniform samples in $[0, tN)$ with overwhelming probability. This is a simple calculation involving the binomial cumulative distribution function with parameter tN/c. Combining these estimates gives a conservative upper bound on the number of bits required to sample a uniform random d-subset of $[0, N)$. A single application of the exact routine recommended by Silverman and Whyte can then be used to improve this bound if necessary.

3.5 Explicit algorithm for computing parameters†

Algorithm 6 determines the smallest recommended N from Table A.1 or Table A.2 that allows for k bit security. Additional details, such as recommendations on how to efficiently perform the search in Line 17, may be found in our implementation which is available at https://github.com/NTRUOpenSourceProject/ntru-params.

Algorithm 6 NTRUEncrypt parameter generation

Input: Desired security level k.

1: Let n_j be the jth value, ordered by magnitude, from either Table A.1 or Table A.2.
2: Set $j = 1$.
3: Set $N = n_j$.
4: Set $d_g = \lfloor \frac{N}{3} \rceil$.
5: Set $d_1 = \left\lceil \frac{1}{4}\left(\sqrt{1+\frac{8N}{3}} - 1\right)\right\rceil$ {The next integer above the positive root of $2x^2 + x - N/3$.}
6: Set $d_2 = \left\lceil \left(\frac{N}{3} - d_1\right)/(2d_1)\right\rceil$.
7: Set $d_3 = \max\left(\left\lceil \frac{d_1+1}{2}\right\rceil, \left\lceil \frac{N}{3} - 2d_1 d_2\right\rceil\right)$.
8: Set d_m to be the largest value satisfying Equation 3.6.
9: Set $k_1 = \left\lfloor \frac{1}{2}\log_2\left(|\mathcal{P}_N(d_1, d_2, d_3)|/N\right)\right\rfloor$. {Cost of direct combinatorial search gives an upper bound on the security.}
10: **if** $k_1 < k$ **then**
11: Increment j.
12: Goto line 3.
13: **end if**
14: Set σ according to Equation 3.11:

$$\sigma = \left((4d_1d_2 + 2d_3) \cdot \frac{N - d_m + 2d_g + 1}{N}\right)^{1/2}.$$

15: Set q to be the smallest power of 2 satisfying

$$N \cdot \texttt{erfc}\left((q-2)/(6\sqrt{2}\sigma)\right) < 2^{-k_1}.$$

{Estimate security}
16: Search for a hybrid parameter K that minimizes the maximum of the cost estimates for hybrid attacks. Equation 5.19 gives the cost of preprocessing a basis for the hybrid attack, and Equation 5.18 gives the cost of combinatorial search on the key space.
17: Let k_2 be the security estimate from Line 16.
18: **if** $k > \min(k_1,\ k_2)$ **then**
19: Increment j.
20: Go to Line 3.
21: **end if**
22: Let $q' = q/2$.
23: **if** $N \cdot \texttt{erfc}\left((q'-2)/(6\sqrt{2}\sigma)\right) < 2^{-k}$ **then**
24: Set $q = q'$
25: Go to Line 16.
26: **end if**

Output: $[N, q, d_1, d_2, d_3, d_g, d_m]$.

3.6 Parameters for NTRUEncrypt

3.6.1 EESS #1 v2

The parameters in Table 3.2 were originally recommended in EESS #1 version 2 in 2007 and are still believed to meet the security level advertised in 2007 against classical adversaries. More recent versions of EESS #1 have deprecated the use of SHA-1, and hence the $N = 401$ and $N = 439$ parameter sets have been replaced. The security estimates presented here are justified in Chapter 5.

EESS #1 Parameter Sets and Security Estimates											
							Hybrid Attack		Product form	$\log_2$	Advertised
N	q	d_1	d_2	d_3	d_g	d_m	K	Cost	search cost	dec. fail prob.	security
401	2048	8	8	6	133	101	154	**116**	145	-217	112
439	2048	9	8	5	146	112	175	**133**	147	-195	128
593	2048	10	10	8	197	158	264	204	**193**	-139	192
743	2048	11	11	15	247	204	360	280	**256**	-112	256

Table 3.2

3.6.2 New parameters in EESS #1 v3

The parameters above do not take quantum adversaries into consideration. The time/space tradeoff in the hybrid attack (Section 5.5) can be replaced by a Grover search to achieve the same asymptotic time complexity as the hybrid attack with a space complexity that is polynomial in N. One may expect that a quantum time/space tradeoff could do even better, however this seems unlikely given the failure of quantum time/space tradeoffs against collision problems in other domains [10]. Several proposals in this direction have been made, such as [24], however these assume unrealistic models of quantum computation. For now, it seems that the best quantum attack on NTRUEncrypt is the hybrid attack with meet-in-the-middle search replaced by Grover search in the Kth projected lattice.

Fluhrer has noted that there are weaknesses in the EESS #1 parameter sets assuming worst-case cost models for quantum computation [24]. In particular, if one Grover iteration is assigned cost equivalent to one classical operation then attacks on the hash functions used in key generation and encryption can break the EESS #1 parameter sets.

Post-Quantum Parameter Sets and Security Estimates												
							Estimate 2		Product form	$\log_2$	Classical	Quantum
N	q	d_1	d_2	d_3	d_g	d_m	K	Cost	search cost	dec. fail prob.	security est.	security est.
443	2048	9	8	5	148	115	177	**134**	147	-196	128	128
509	2048	9	9	8	170	134	214	**164**	176	-171	160	128
587	2048	10	10	8	196	157	260	201	**193**	-139	192	128
743	2048	11	11	15	247	204	360	280	**256**	-112	256	128

Table 3.3

Developing a realistic quantum cost model is outside the scope of this work. However, we can easily provide parameter sets that are secure in Fluhrer's model by using a 256 bit hash function and sufficiently large seeds. The parameters in Table 3.3 all have 128 bit security against quantum adversaries assuming that that BGF and MGF are instantiated with SHA-256, and that $bLen$ is taken to be 256 in Algorithm 4. One should also ensure that any deterministic random bit generators used in key generation or encryption are instantiated with at least 256 bits of entropy from a secure random source.

The parameter sets for $N = 443$, $N = 509$, and $N = 587$ in Table 3.3 are new, $N = 743$ is the same as ees743ep1 from IEEE 1363.1 [3]

3.6.3 Parameters without decryption failure

One can sacrifice some security against lattice reduction attacks in order to completely eliminate decryption failures; see Table 3.4. In the case of $N = 743$ this has no consequences for the security of the scheme as combinatorial attacks outperform lattice reduction attacks for that parameter set.

							Estimate 2		Product form		Classical	Quantum
N	q	d_1	d_2	d_3	d_g	d_m	K	Cost	search cost	dec. fail prob.	security est.	security est.
587	4096	10	10	8	196	157	239	**184**	193	0	184	128
743	4096	11	11	15	247	204	332	258	**256**	0	256	128

Table 3.4: NTRUEncrypt parameter sets that eliminate decryption failures

3.6.4 Paramters with non-trivial $\boldsymbol{f}$ mod p

If one takes $\boldsymbol{f} = 1 + \boldsymbol{F}$ with $\boldsymbol{F} \in \mathcal{T}_N(d_f, d_f)$ or $\boldsymbol{F} \in \mathcal{P}_N(d_1, d_2, d_3)$ rather than $\boldsymbol{f} = 1 + p\boldsymbol{F}$ it is sometimes possible to decrease q and achieve a higher level of security. This comes at the cost of a more computational intensive decryption procedure (an extra multiplication in $R_N/(p)$) and a higher probability of decryption failure.

For some N the resulting decryption failure probability is still small enough that it may be acceptable for some use cases. Table 3.5 presents product form parameter sets for this variant.

N	q	d_1	d_2	d_3	d_g	d_m	Hybrid Attack K	Hybrid Attack Cost	Product form search cost	$\log_2$ dec. fail prob.	Classical security est.	Quantum security est.
401	1024	8	8	6	133	101	169	**129**	145	-90	129	128
443	1024	9	8	5	148	115	194	148	**147**	-81	147	128

Table 3.5: NTRUEncrypt parameter sets with non-negligible decryption failure rates

Chapter 4

NTRUMLS

The development of lattice based signature schemes has been substantially more fraught than that of encryption schemes. A signature scheme was proposed in the same paper that presented the GGH encryption scheme [30], and later an efficient instantiation, NTRUSign, was proposed using NTRU lattices [38]. These schemes were completely broken by Nguyen and Regev in 2006 [61]. They demonstrated that statistical information contained in transcripts of signature/message pairs could be exploited to reveal the "shape" of the signer's private basis. These attacks were improved by Ducas and Nguyen in [21] to squash several countermeasures to the original attack that has been proposed.

The first successful signature schemes based on lattices were proposed by Gentry, Peikert, and Vaikuntanathan in 2008[29]. They introduced a notion of a pre-image sampleable trapdoor function (PSF) and showed how to instantiate such a function using lattices. In their scheme, a user's public key is a PSF f and a signature is a point $\boldsymbol{y}$ such that $f(\boldsymbol{y})$ is equal to the hash of the message to be signed. The signer, using their trapdoor information for f, is able to sample such a $\boldsymbol{y}$ from a fixed public distribution in a manner that reveals nothing that an adversary could not have learned from f itself.

A second method for developing secure lattice signature schemes was developed by Lyubashevsky in a series of papers starting in 2009 [56, 57]. These schemes are based on the Fiat-Shamir transform and do not require the use of trapdoor functions.

A Fiat-Shamir type signature scheme, PASS, developed by Hoffstein, Pipher, and Silverman in the mid 90s, was revived as $\mathsf{PASS}_{\mathtt{RS}}$ in 2013 using Lyubashevsky's rejection sampling technique to remove information from the transcript that had plagued earlier variants [35].

The hardness assumption underlying $\mathsf{PASS_{RS}}$ has not received the same amount of attention as assumptions such as (Ring-)SIS, (Ring-)LWE, or the NTRU assumption. $\mathsf{PASS_{RS}}$ is unlikely to receive much attention given the existence of schemes based on more standard assumptions with comparable efficiency. In late 2013 Jeffery Hoffstein set out to develop a new transcript secure signature scheme that was more "NTRU-like" than PASS. The result was a hash-and-sign type scheme called NTRUMLS [36].

The core idea of NTRUMLS is that a short basis for a lattice can be used to find lattice points with coefficient vectors that satisfy arbitrary equivalence relations modulo a prime p. Under suitable assumptions, the ability to produce a lattice point satisfying a random relation is proof that one is in possession of a short basis.

The key recovery problem for NTRUMLS is identical to NTRUEncryptfor the same values of N, q and p. Security against forgery attacks requires a new assumption, but the best known attack is an approximate closest vector problem in the intersection of L and $p\mathbb{Z}^n$.

The technique can also be applied to general lattices and NTRUMLS can be though of as simply an explicit instantiation in NTRU lattices. We'll first describe the generic technique at a high level, then explore the specifics through NTRUMLS.

4.1 Modular Lattice Signatures (MLS)

The public parameters are N, p, r defined as follows. A signer's secret key is a short basis, M, for a lattice L of rank n, their public key is the HNF basis for L. Messages to be signed are elements of $(\mathbb{Z}/p\mathbb{Z})^n$. A signature on $\boldsymbol{m}$ will be a lattice point inside a hypercube of radius r with a coefficient vector that is congruent to $\boldsymbol{m}$ modulo p.

There are two restrictions on the lattices that may be used. First, it must be possible to efficiently sample a lattice point uniformly at random from those within $\mathcal{B}_N(r)$, the origin centered hypercube of radius r. Second, the map from $L \to (\mathbb{Z}/p\mathbb{Z})^n$ given by reduction modulo p must be onto. Otherwise it may not be possible to find a valid signature for every message. A large class of lattices satisfying the first condition are q-ary lattices with $q = r + \delta$ for some small positive δ. The second condition requires that any basis for the lattice is invertible mod p.

To sign a message point $\boldsymbol{m} \in (\mathbb{Z}/p\mathbb{Z})^n$, the signer first chooses a lattice point $\boldsymbol{s}$ uniformly at random in the origin centered hypercube of radius r. They then compute $\boldsymbol{a}$ such that

$$\boldsymbol{a} \equiv (\boldsymbol{m} - \boldsymbol{s})M^{-1} \pmod{p}.$$

The lattice point $\boldsymbol{\sigma} = \boldsymbol{s} + \boldsymbol{a}M$ is taken to be a candidate signature for $\boldsymbol{m}$. The final crucial step in signing is to reject candidate signatures that may reveal partial information about the initial point $\boldsymbol{s}$ that was chosen in generating the signature. This is fully explained in Section 4.4, but is done by a simple infinity-norm based rejection sampling procedure.

The principal benefit of MLS is its simplicity compared with other lattice based signature schemes. There is no need to sample from a complicated distribution, such as a discrete Gaussian distribution. Furthermore, while the scheme requires rejection sampling, the rejection criteria is a simple infinity norm check.

4.2 NTRUMLS

NTRUMLS is a compact and efficient instantiation of MLS in NTRU lattices. As in Section 3.2 an NTRU lattice is an R_N-module lattice of the form

$$\Lambda_q\left((\boldsymbol{f}, \boldsymbol{g})\right)$$

from Equation 2.9, where $\boldsymbol{f}$ and $\boldsymbol{g}$ are known to be particularly short.

The pair $(\boldsymbol{f}, \boldsymbol{g}) \in R_N^2$ serves as a user's private key. It is necessary for $\boldsymbol{f}$ to be invertible modulo q, and for $\boldsymbol{g}$ to be invertible modulo p. As in NTRUEncrypt we may consider both product form and trinary instantiations, i.e. the $\boldsymbol{f}$ component may be chosen as

$$\boldsymbol{f} = p\boldsymbol{F} \quad \text{with} \quad \boldsymbol{F} \in \mathcal{T}_N(d_f + 1, d_f),$$

or as

$$\boldsymbol{f} = p(1 + \boldsymbol{F}) \quad \text{with} \quad \boldsymbol{F} \in \mathcal{P}_N(d_1, d_2, d_3).$$

Note the small difference with NTRUEncrypt keys: an NTRUMLS key always satisfies $\boldsymbol{f} \equiv 0 \pmod{p}$. Our analysis below will assume $\boldsymbol{F} \in \mathcal{T}_N$, minor alterations to the proofs are occasionally needed for product-form $\boldsymbol{F}$.

Ignoring this small difference, NTRUMLS key generation is identical to NTRUEncrypt key generation. The $\boldsymbol{g}$ component is an element of $\mathcal{T}_N(d_g + 1, d_g)$, and the public key is

$$\boldsymbol{h} = \boldsymbol{g}/\boldsymbol{f} \bmod q.$$

Note that the conditions of Lemma 8 are met, so $\Lambda_q\left((\boldsymbol{f}, \boldsymbol{g})\right) = \Lambda_q\left((1, \boldsymbol{h})\right)$, and this lattice is of rank N and volume q^N.

We define the subset of R_N contained in a hypercube of radius k as:

$$\mathcal{C}(k) = \{\boldsymbol{f} \in R_N : \|\boldsymbol{f}\|_\infty \leq k\}. \tag{4.1}$$

For example $\mathcal{C}(1)$ is precisely the set of trinary polynomials $\mathcal{T}_N$.

We also define the subset of lattice points for which the first coordinate is within a hypercube of radius k and the second is within a hypercube of radius ℓ:

$$\mathcal{L}(k, \ell) = \Lambda_q\left((\boldsymbol{f}, \boldsymbol{g})\right) \cap (\mathcal{C}(k) \times \mathcal{C}(\ell)). \tag{4.2}$$

NTRUMLS Parameter (A)**.** *The largest integer such that $\mathcal{C}(A) \subseteq \mathcal{C}(q/2)$ and $\mathcal{C}(A)$ contains an equal number of points in each coset of $p\mathbb{Z}^N$.*
Explicitly, let

$$A' = \left\lfloor \frac{q-p}{2p} \right\rfloor, \quad \textit{and} \quad A = \left\lfloor pA' + \frac{p}{2} \right\rfloor.$$

Then

$$\mathcal{C}\left(pA' + \frac{p}{2}\right) = \mathcal{C}(A) \subseteq \mathcal{C}(q/2),$$

To ease notation we will also keep this definition of A' throughout.

NTRUMLS Parameter (B_s, B_t)**.** *Positive integers such that, for every key $(\boldsymbol{f}, \boldsymbol{g})$, the probability that*

$$\boldsymbol{a} \cdot (\boldsymbol{f}, \boldsymbol{g}) \in \mathcal{L}(B_s, B_t)$$

is non-negligible with respect to a uniform choice of $\boldsymbol{a} \in \mathcal{C}(1)$.

In generating a signature one first applies a hash function (more specifically, a mask generating function as defined in Section 3.3.2) to the message and public key to obtain a point $(\boldsymbol{s}_p, \boldsymbol{t}_p) \in \mathcal{C}\left(\frac{p}{2}\right) \times \mathcal{C}\left(\frac{p}{2}\right)$. Then one samples a lattice point $(\boldsymbol{s}, \boldsymbol{t})$ uniformly at random from $\mathcal{L}(A, \frac{q}{2} - B_t)$ conditioned on $(\boldsymbol{s}, \boldsymbol{t}) \equiv (\boldsymbol{s}_p, \boldsymbol{t}_p) \pmod{p}$.

For NTRU lattices this is best accomplished by first sampling $\boldsymbol{s}_0$ uniformly at random from $\mathcal{C}(A)$ conditioned on $\boldsymbol{s}_0 \equiv \boldsymbol{s}_p \pmod{p}$. The signer then computes $\boldsymbol{t}_0 = \boldsymbol{s}_0 \boldsymbol{h} \bmod q$ and

$$\boldsymbol{a} = \boldsymbol{g}^{-1} \cdot (\boldsymbol{t}_0 - \boldsymbol{t}_p) \bmod p.$$

Finally the signer outputs $(\boldsymbol{s}, \boldsymbol{t}) = (\boldsymbol{s}_0, \boldsymbol{t}_0) + \boldsymbol{a} \cdot (\boldsymbol{f}, \boldsymbol{g})$ as a signature iff

1. $\boldsymbol{a} \cdot (\boldsymbol{f}, \boldsymbol{g}) \in \mathcal{L}(B_s, B_t)$, and

2. $(\boldsymbol{s}, \boldsymbol{t}) \in \mathcal{L}(A - B_s, \frac{q}{2} - B_t)$.

Verification simply involves hashing the message to $(\boldsymbol{s}_p, \boldsymbol{t}_p)$, calculating $\boldsymbol{t} = \boldsymbol{sh} \bmod q$ and checking that

1. $(\boldsymbol{s}, \boldsymbol{t}) \in \mathcal{L}(A - B_s, \frac{q}{2} - B_t)$, and
2. $(\boldsymbol{s}, \boldsymbol{t}) \equiv (\boldsymbol{s}_p, \boldsymbol{t}_p) \pmod p$.

Remark. Note the two asymmetries between the "s-side" and the "t-side," first in the use of different bounds B_s and B_t, then in the norm of valid signatures $A - B_s$ and $\frac{q}{2} - B_t$. The use of B_s and B_t is due to the difference in the norms of $\boldsymbol{f}$ and $\boldsymbol{g}$. We have $\|\boldsymbol{af}\|_\infty \approx 3\|\boldsymbol{ag}\|_\infty$, hence optimizing the probability of acceptance in signing necessitates the use of different bounds.

The second asymmetry, the use of $A - B_s$ instead of $\frac{q}{2} - B_s$ did not appear in [36], but has come from our experience implementing NTRUMLS. To illustrate why A is used instead of $\frac{q}{2}$ suppose q is even. If there is any chance that $\|\boldsymbol{s}_0 + \boldsymbol{s}_p\|_\infty = q/2$ then reducing $\boldsymbol{s}$ modulo q might send $q/2$ to $-q/2$ and cause the s-side equivalence check to fail during verification. This can be mitigated by simply not reducing $\boldsymbol{s}$ modulo q, but doing so complicates the encoding of signatures for transmission. The use of A complicates our description but simplifies the implementation.

4.3 NTRUMLS Algorithms*

Key generation will look familiar– it is essentially identical to Algorithm 1.

Algorithm 7 NTRUMLS Product Form Key Generation

Input: A full set of NTRUMLS parameters.

```
repeat
    F ←$ T_N(d_f + 1, d_f) or P_N(d_1, d_2, d_3)
    f = pF ∈ R_{N,q}
until f is invertible in R_{N,q}
repeat
    g ←$ T_N(d_g + 1, d_g)
until g is invertible in R_{N,q} and R_{N,p}
h = f^{-1} g ∈ R_{N,q}
```

Output: Private key $(\boldsymbol{f}, \boldsymbol{g})$, Public key $(1, \boldsymbol{h})$

Algorithm 8 NTRUMLS Signature Algorithm

Input: $(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{h}, \mu)$, where $(\boldsymbol{f}, \boldsymbol{g})$ is a private key, $\boldsymbol{h}$ is the corresponding public key, and $\mu \in \{0,1\}^*$ is a document to be signed.

$(\boldsymbol{s}_p, \boldsymbol{t}_p) \longleftarrow \mathsf{Hash}(\boldsymbol{h}, \mu)$
repeat
  $\boldsymbol{r} \xleftarrow{\$} \mathcal{C}(A')$
  $\boldsymbol{s}_0 = \boldsymbol{s}_p + p\boldsymbol{r}$
  $\boldsymbol{t}_0 = \boldsymbol{h}\boldsymbol{s}_0 \bmod q$
  $\boldsymbol{a} = \boldsymbol{g}^{-1}(\boldsymbol{t}_p - \boldsymbol{t}_0) \bmod p$
  $(\boldsymbol{s}, \boldsymbol{t}) = (\boldsymbol{s}_0, \boldsymbol{t}_0) + \boldsymbol{a} \cdot (\boldsymbol{f}, \boldsymbol{g})$
until $\|\boldsymbol{a}\boldsymbol{f}\|_\infty \leq B_s$ and $\|\boldsymbol{a}\boldsymbol{g}\|_\infty \leq B_t$ and $\|\boldsymbol{s}\|_\infty \leq A - B_s$ and $\|\boldsymbol{t}\|_\infty \leq \frac{q}{2} - B_t$

Output: $(\boldsymbol{s}, \boldsymbol{t}, \mu)$

Remark. Since $\boldsymbol{t} \equiv \boldsymbol{h} * \boldsymbol{s} \pmod q$ it does not need to be published explicitly. Furthermore since $\boldsymbol{s} \equiv \boldsymbol{s}_p \pmod p$ and $\boldsymbol{s}_p$ can be obtained by hashing $\boldsymbol{h}$ with the message, the signer can simply publish $(\boldsymbol{s} - \boldsymbol{s}_p)/p$ as the signature. The resulting signature is of length $N\lceil \log_2 q/p \rceil$ bits.

Algorithm 9 NTRUMLS Verification Algorithm

Input: $(\boldsymbol{s}, \boldsymbol{t}, \mu, \boldsymbol{h})$

valid $\longleftarrow$ yes
$(\boldsymbol{s}_p, \boldsymbol{t}_p) \longleftarrow \mathsf{Hash}(\boldsymbol{h}, \mu)$
if $\boldsymbol{t} \not\equiv \boldsymbol{h} * \boldsymbol{s} \pmod q$ **then**
  valid $\longleftarrow$ no
end if
if $\|\boldsymbol{s}\| > \dfrac{q}{2} - B_s$ or $\|\boldsymbol{t}\| > \dfrac{q}{2} - B_t$ **then**
  valid $\longleftarrow$ no
end if
if $(\boldsymbol{s}, \boldsymbol{t}) \not\equiv (\boldsymbol{s}_p, \boldsymbol{t}_p) \pmod p$ **then**
  valid $\longleftarrow$ no
end if

Output: *valid*

Proposition 11. *The Signing Algorithm produces signatures that are verified as valid by the Verification Algorithm.*

Proof. This is an easy exercise. □

4.4 NTRUMLS Transcript Security*

In this section we prove that, under a reasonable assumption, a transcript of signatures created using the signing algorithm contains no information that is not already available to someone who knows the public verification key $\boldsymbol{h}$. We do this by showing that an honest signer produces signatures that are uniformly distributed on $\mathcal{L}\left(A - B_s, \frac{q}{2} - B_t\right)$. We are able to show that for any document hash, $(\boldsymbol{s}_p, \boldsymbol{t}_p)$, the signer's distribution is precisely the uniform distribution on the subset of signature points in $(\boldsymbol{s}_p, \boldsymbol{t}_p) + p\mathbb{Z}^{2N}$ (Proposition 12). For uniformity on the entire signature region we must assume that each coset of $p\mathbb{Z}^{2N}$ contains roughly the same number of signature points (Assumption 15).

We further show that a party who knows $\boldsymbol{h}$ alone can produce a transcript of pairs

$$(\text{Valid Signature}_i, \text{Document Hash}_i)_{i=1,2,3,\ldots}$$

that is statistically indistinguishable from an analogous transcript produced using the signing algorithm and the private key $(\boldsymbol{f}, \boldsymbol{g})$. Specifically, the signature points produced by such a party are uniform on $\mathcal{L}\left(A - B_s, \frac{q}{2} - B_t\right)$, and the document hashes (obtained by reducing the signature coefficients modulo p) are uniform on $\mathcal{C}(p/2)$.

We start by analyzing the transcript created using the signing algorithm and $(\boldsymbol{f}, \boldsymbol{g})$. We note that the rejection sampling condition is what allows us to prove that the resulting signatures are uniformly distributed in a certain space of allowable signatures.

We assume that our hash function outputs document hashes

$$(\boldsymbol{s}_p, \boldsymbol{t}_p) \in \mathcal{C}(p/2)^2$$

that are uniformly distributed on $\mathcal{C}(p/2)^2$. We use Steps 3 through 7 of the Signing Algorithm to define a signing function

$$(\boldsymbol{s}, \boldsymbol{t}) = \sigma'(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \boldsymbol{r}).$$

Thus σ' is a map

$$\sigma' \quad : \quad \overbrace{\mathcal{C}(p) \times \mathcal{C}(1)}^{\text{private key } (\boldsymbol{f}, \boldsymbol{g})} \times \overbrace{\mathcal{C}\left(\frac{p}{2}\right) \times \mathcal{C}\left(\frac{p}{2}\right)}^{\text{document hash } (\boldsymbol{s}_p, \boldsymbol{t}_p)} \times \overbrace{\mathcal{C}(A')}^{\text{random element } \boldsymbol{r}} \quad \longrightarrow \quad \underbrace{\mathcal{L}\left(A + B_s, \frac{q}{2} + B_t\right)}_{\text{potential signature } (\boldsymbol{s}, \boldsymbol{t})}$$

given explicitly by

$$\sigma'(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \boldsymbol{r}) = (\boldsymbol{s}_0 + \boldsymbol{a}\boldsymbol{f}, \boldsymbol{t}_0 + \boldsymbol{a}\boldsymbol{g}), \tag{4.3}$$

where

$$\boldsymbol{s}_0 = \boldsymbol{s}_p + p\boldsymbol{r}, \tag{4.4}$$

$$\boldsymbol{t}_0 \equiv \boldsymbol{h}\boldsymbol{s}_0 \pmod{q} \quad \text{with } \boldsymbol{t}_0 \in \mathcal{C}(q/2), \tag{4.5}$$

$$\boldsymbol{a} \equiv \boldsymbol{g}^{-1}(\boldsymbol{t}_p - \boldsymbol{t}_0) \pmod{p} \quad \text{with } \boldsymbol{a} \in \mathcal{C}(p/2). \tag{4.6}$$

We will write

$$\Omega' = \mathcal{C}(p) \times \mathcal{C}(1) \times \mathcal{C}\left(\frac{p}{2}\right) \times \mathcal{C}\left(\frac{p}{2}\right) \times \mathcal{C}(A')$$

for the domain of σ'.

We now introduce rejection sampling by defining

$$\Omega_{B_s,B_t} = \left\{ (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \boldsymbol{r}) \in \Omega' : \begin{array}{r} (\boldsymbol{s}, \boldsymbol{t}) := \sigma'(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \boldsymbol{r}) \\ = (\boldsymbol{s}_0 + \boldsymbol{a} * \boldsymbol{f}, \boldsymbol{t}_0 + \boldsymbol{a} * \boldsymbol{g}), \\ \|\boldsymbol{s}\| \le A - B_s, \|\boldsymbol{t}\| \le \frac{q}{2} - B_t, \\ \|\boldsymbol{a} * \boldsymbol{f}\| \le B_s, \|\boldsymbol{a} * \boldsymbol{g}\| \le B_t \end{array} \right\}.$$

The restriction of σ' to Ω_{B_s,B_t}, which we denote by σ, is then a map

$$\sigma : \Omega_{B_s,B_t} \longrightarrow \mathcal{L}\left(A - B_s, \frac{q}{2} - B_t\right).$$

The following proposition says that every signature that is valid for the document hash $(\boldsymbol{s}_p, \boldsymbol{t}_p)$ has the same number of preimages in $\mathcal{C}(A')$.

Proposition 12. *The signature function σ has the following property: For a given*

$$\begin{aligned} \textit{private key} \quad & (\boldsymbol{f}, \boldsymbol{g}), \\ \textit{document hash} \quad & (\boldsymbol{s}_p, \boldsymbol{t}_p) \in \mathcal{C}\left(\frac{p}{2}\right) \times \mathcal{C}\left(\frac{p}{2}\right), \end{aligned}$$

the output of σ, when queried on uniformly random $\boldsymbol{r} \in \mathcal{C}(A')$, is uniformly distributed over the set

$$\left\{ (\boldsymbol{s}, \boldsymbol{t}) \in L_h\left(A - B_s, \frac{q}{2} - B_t\right) : (\boldsymbol{s}, \boldsymbol{t}) \equiv (\boldsymbol{s}_p, \boldsymbol{t}_p) \bmod p \right\}$$

of valid signatures for $(\boldsymbol{s}_p, \boldsymbol{t}_p)$. Equivalently, the size of the set

$$\{\boldsymbol{r} \in \mathcal{C}(A') : \sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \boldsymbol{r}) = (\boldsymbol{s}, \boldsymbol{t})\}$$

is the same for all

$$(\boldsymbol{s}, \boldsymbol{t}) \in \mathcal{L}\left(A - B_s, \frac{q}{2} - B_t\right) \quad \textit{satisfying} \quad (\boldsymbol{s}, \boldsymbol{t}) \equiv (\boldsymbol{s}_p, \boldsymbol{t}_p) \pmod{p}.$$

Proof. Since we know from Proposition 11 that $\sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \boldsymbol{r})$ is congruent to $(\boldsymbol{s}_p, \boldsymbol{t}_p)$ modulo p, it is clear that there is zero probability of generating the signature $(\boldsymbol{s}, \boldsymbol{t})$ if $(\boldsymbol{s}, \boldsymbol{t}) \not\equiv (\boldsymbol{s}_p, \boldsymbol{t}_p) \pmod{p}$. So we assume henceforth that

$$(\boldsymbol{s}, \boldsymbol{t}) \equiv (\boldsymbol{s}_p, \boldsymbol{t}_p) \pmod{p}. \tag{4.7}$$

The random element $\boldsymbol{r}$ used to generate a signature is chosen uniformly from the set $\mathcal{C}(A')$, so there are $(2A'+1)^N$ possible choices for $\boldsymbol{r}$. Hence the probability of obtaining $(\boldsymbol{s}, \boldsymbol{t})$ as a signature on $(\boldsymbol{s}_p, \boldsymbol{t}_p)$ is equal to $(2A'+1)^{-N}$ times the number of elements in the set

$$\Sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}, \boldsymbol{t}) = \big\{\boldsymbol{r} \in \mathcal{C}(A') : \sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \boldsymbol{r}) = (\boldsymbol{s}, \boldsymbol{t})\big\}. \tag{4.8}$$

The key to counting the size of the set $\Sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}, \boldsymbol{t})$ is the bijection described in the following lemma.

Lemma 13. *Let*

$$\mathcal{V} = \left\{\boldsymbol{b} \in \mathcal{C}\left(\frac{p}{2}\right) : \big\|\boldsymbol{b} * \boldsymbol{f}\big\| \le B_s \text{ and } \big\|\boldsymbol{b} * \boldsymbol{g}\big\| \le B_t\right\},$$

and let

$$(\boldsymbol{s}, \boldsymbol{t}) \in \mathcal{L}\left(A - B_s, \frac{q}{2} - B_t\right) \quad \text{satisfy} \quad (\boldsymbol{s}, \boldsymbol{t}) \equiv (\boldsymbol{s}_p, \boldsymbol{t}_p) \pmod{p}.$$

Then there is a well-defined bijection of sets

$$\begin{aligned} \phi : \mathcal{V} &\longrightarrow \Sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}, \boldsymbol{t}), \\ \boldsymbol{b} &\longmapsto \frac{\boldsymbol{s} - \boldsymbol{s}_p}{p} - \boldsymbol{b}\frac{\boldsymbol{f}}{p}. \end{aligned} \tag{4.9}$$

Proof. First, since the coefficients of $\boldsymbol{s} - \boldsymbol{s}_p$ are multiples of p, and similarly $\boldsymbol{f}$ has coefficients divisible by p, we see that the polynomial on the right-hand side of (4.9) has coefficients in $\mathbb{Z}$.

We next need to show that $\phi(\boldsymbol{b}) \in \Sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}, \boldsymbol{t})$, which by the definition of $\Sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}, \boldsymbol{t})$ means showing that $\phi(\boldsymbol{b}) \in \mathcal{C}(A')$ and

$$\sigma\big(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \phi(\boldsymbol{b})\big) = (\boldsymbol{s}, \boldsymbol{t}).$$

First note that because $\boldsymbol{s} \in \mathcal{C}(A - B_s)$, $\boldsymbol{s}_p \in \mathcal{C}\left(\frac{p}{2}\right)$, and $\boldsymbol{b}\boldsymbol{f} \in \mathcal{C}(B_s)$, the triangle inequality gives

$$\big\|\phi(\boldsymbol{b})\big\| = \left\|\frac{1}{p}\left(\boldsymbol{s} - \boldsymbol{s}_p - \boldsymbol{b}\boldsymbol{f}\right)\right\| \le \left\lfloor \frac{A - B_s + \frac{p}{2} + B_s}{p} \right\rfloor = A'.$$

The use of the floor function is justified by noting that $\phi(\boldsymbol{b})$ has integer coefficients. This establishes that $\phi(\boldsymbol{b}) \in \mathcal{C}\,(A')$.

Next we use the four formulas (4.3)–(4.6) to compute the signature $\sigma\big(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \phi(\boldsymbol{b})\big)$:

$$\begin{aligned}
\boldsymbol{s}_0 &= \boldsymbol{s}_p + p\phi(\boldsymbol{b}) \\
&= \boldsymbol{s}_p + p\left(\frac{\boldsymbol{s} - \boldsymbol{s}_p}{p} - \boldsymbol{b}\frac{\boldsymbol{f}}{p}\right) \\
&= \boldsymbol{s} - \boldsymbol{b}\boldsymbol{f}, && (4.10)\\
\boldsymbol{t}_0 &\equiv \boldsymbol{h}\boldsymbol{s}_0 \pmod{q} \\
&\equiv \boldsymbol{h}(\boldsymbol{s} - \boldsymbol{b}\boldsymbol{f}) \pmod{q} \\
&\equiv \boldsymbol{h}\boldsymbol{s} - \boldsymbol{b}\boldsymbol{g} \pmod{q} \quad \text{since } \boldsymbol{h} \equiv \boldsymbol{f}^{-1}\boldsymbol{g} \pmod{q}, \\
&\equiv \boldsymbol{t} - \boldsymbol{b}\boldsymbol{g} \pmod{q} \quad \text{since } (\boldsymbol{s}, \boldsymbol{t}) \in \mathcal{L}. && (4.11)
\end{aligned}$$

Since $(\boldsymbol{s}, \boldsymbol{t}) \in \mathcal{L}\left(A - B_s, \frac{q}{2} - B_t\right)$ and $\boldsymbol{b} \in \mathcal{C}$, we have

$$\begin{aligned}
\big\|\boldsymbol{s}_0\big\| &\le \big\|\boldsymbol{s}\big\| + \big\|\boldsymbol{b} * \boldsymbol{f}\big\| = A - B_s + B_s = A, \\
\big\|\boldsymbol{t}_0\big\| &\le \big\|\boldsymbol{t}\big\| + \big\|\boldsymbol{b} * \boldsymbol{g}\big\| = \frac{q}{2} - B_t + B_t = \frac{q}{2},
\end{aligned}$$

i.e. (4.11), similar to (4.10), is an equality, not just a congruence. Continuing with the computation of $\sigma\big(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \phi(\boldsymbol{b})\big)$, we use (4.7) to compute

$$\boldsymbol{a} \equiv \boldsymbol{g}^{-1}(\boldsymbol{t}_p - \boldsymbol{t}_0) \equiv \boldsymbol{b} \pmod{p}.$$

(Note that $\boldsymbol{t} \equiv \boldsymbol{t}_p \pmod{p}$ from (4.6).) Since both $\boldsymbol{a}$ and $\boldsymbol{b}$ are in $\mathcal{C}(p/2)$, this tells us that $\boldsymbol{a} = \boldsymbol{b}$.

We now use (4.3) to compute the signature

$$\begin{aligned}
\sigma\big(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \phi(\boldsymbol{b})\big) &= (\boldsymbol{s}_0 + \boldsymbol{a}\boldsymbol{f}, \boldsymbol{t}_0 + \boldsymbol{a}\boldsymbol{g}) \quad \text{definition of } \sigma, \\
&= (\boldsymbol{s} - \boldsymbol{b}\boldsymbol{f} + \boldsymbol{a}\boldsymbol{f}, \boldsymbol{t} - \boldsymbol{b}\boldsymbol{g} + \boldsymbol{a}\boldsymbol{g}) \\
&\qquad\qquad \text{from (4.10) and (4.11)}, \\
&= (\boldsymbol{s}, \boldsymbol{t}) \quad \text{since } \boldsymbol{a} = \boldsymbol{b}.
\end{aligned}$$

Hence directly from the definition (4.8) of the set $\Sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}, \boldsymbol{t})$, we see that

$$\phi(\boldsymbol{b}) \in \Sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}, \boldsymbol{t}).$$

We next fix an $\boldsymbol{r} \in \Sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}, \boldsymbol{t})$ and compute how many $\boldsymbol{b} \in \mathcal{C}(p/2)$ satisfy $\phi(\boldsymbol{b}) = \boldsymbol{r}$. Since all coefficients of the polyomials $\boldsymbol{s} - \boldsymbol{s}_p$ and $\boldsymbol{f}$ are divisible by p, to ease notation we write

$$\boldsymbol{s} - \boldsymbol{s}_p = p\boldsymbol{S} \quad \text{and} \quad \boldsymbol{f} = p\boldsymbol{F}.$$

We recall that by assumption, the polynomial $\boldsymbol{F}$ is invertible modulo p. We have

$$\begin{aligned} \phi(\boldsymbol{b}) = \boldsymbol{r} &\iff \boldsymbol{S} - \boldsymbol{b}\boldsymbol{F} = \boldsymbol{r} \\ &\iff \boldsymbol{b} \equiv \boldsymbol{F}^{-1}(\boldsymbol{S} - \boldsymbol{r}) \pmod{p} \quad \text{and} \quad \|\boldsymbol{b}\| \leq \frac{p}{2}. \end{aligned}$$

There is thus exactly one value of $\boldsymbol{b}$ in $\mathcal{C}(p/2)$ satisfying $\phi(\boldsymbol{b}) = \boldsymbol{r}$, namely the unique element of $\mathcal{C}(p/2)$ that is congruent to $\boldsymbol{F}^{-1}(\boldsymbol{S} - \boldsymbol{r})$ modulo p. This shows that ϕ is bijective, which concludes the proof of Lemma 13. □

Resuming the proof of Proposition 12, we have, for all $(\boldsymbol{s}, \boldsymbol{t}) \equiv (\boldsymbol{s}_p, \boldsymbol{t}_p) \pmod{p}$,

$$\mathrm{Prob}_{\boldsymbol{r} \leftarrow \mathcal{C}(A')} \left(\begin{array}{c|l} \text{signature} & \text{private key is } (\boldsymbol{f}, \boldsymbol{g}) \text{ and} \\ \text{is } (\boldsymbol{s}, \boldsymbol{t}) & \text{document hash is } (\boldsymbol{s}_p, \boldsymbol{t}_p) \end{array} \right) = \frac{\#\Sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}, \boldsymbol{t})}{\#\mathcal{C}(A')} = \frac{\#\mathcal{C}}{\#\mathcal{C}(A')},$$

where the penultimate equality follows from Lemma 13. This completes the proof of Proposition 12. □

To give a complete proof of transcript security we need a slightly stronger version of Proposition 2 to be true:

Proposition 14. *The distribution of signatures produced by querying σ on uniformly random $(\boldsymbol{s}_p, \boldsymbol{t}_p) \in R(p/2)^2$ and uniformly random $\boldsymbol{r} \in \mathcal{C}(A')$ is indistinguishable from the uniform distribution on $\mathcal{L}\left(A - B_s, \frac{q}{2} - B_t\right)$.*

Proposition 14 is an immediate consequence of Proposition 12 under the assumption that, for any given $\boldsymbol{h}$, the number elements of $\mathcal{L}\left(A - B_s, \frac{q}{2} - B_t\right)$ in each coset of $p\mathbb{Z}^{2N}$ is constant. This certainly fails to be the case for some lattices, for instance $\boldsymbol{h} = 1$ has vectors in only p^N distinct cosets. However, it seems likely that this assumption holds for the lattices used in NTRUMLS.

Assumption 15. *The elements of $\mathcal{L}(A - B_s, \frac{q}{2} - B_t)$ are equidistributed among cosets of $p\mathbb{Z}^{2N}$.*

We conclude this section by noting that any party with access to $\boldsymbol{h}$ can sample the uniform distribution on $\mathcal{L}\left(A - B_s, \frac{q}{2} - B_t\right)$. One simply generates random $\boldsymbol{s} \in \mathcal{C}(A - B_s)$ until $\boldsymbol{h} * \boldsymbol{s} \in \mathcal{C}(\frac{q}{2} - B_t)$. Since the signing region contains a large fraction of $\mathcal{L}\left(A, \frac{q}{2}\right)$, this suceeds after a small number of iterations. A transcript of

$$((\boldsymbol{s}, \boldsymbol{t})_i, (\boldsymbol{s}_p, \boldsymbol{t}_p)_i)_{i=1,2,3,\ldots}$$

where $(\boldsymbol{s}, \boldsymbol{t})_i$ is produced in this manner and $(\boldsymbol{s}_p, \boldsymbol{t}_p)_i = (\boldsymbol{s}, \boldsymbol{t})_i \pmod p$ is uniformly distributed on $\mathcal{L}\left(A - B_s, \frac{q}{2} - B_t\right) \times \mathcal{C}(p/2)$ by Assumption 15. By Proposition 14, and the assumption that the output of Hash is uniform on $\mathcal{C}(p/2)^2$, this transcript is indistinguishable from one produced by an honest signer. The only difference between the two transcripts is that the party who used $\boldsymbol{h}$ alone does not know messages, μ_i, such that $\mathsf{Hash}(\boldsymbol{h}, \mu_i) = (\boldsymbol{s}_p, \boldsymbol{t}_p)_i$.

4.5 Probability of Generating a Valid Signature*

To simplify our analysis we let $B = \lceil p^2 N/4 \rceil$ and take

$$B_s = B_t = B.$$

With this assumption there is zero probability of rejecting a candidate signature due to $\left\|\boldsymbol{a} * \boldsymbol{s}\right\| > B_s$ or $\left\|\boldsymbol{a} * \boldsymbol{t}\right\| > B_t$, but the probability of rejection due to non-inclusion in $Ł(A - B_s, \frac{q}{2} - B_t)$ is significant. Regardless, we can show that the probability of generating a valid signature is approximately $e^{-8/k}$, which is still practical. Further, the probability of rejection can be made significantly lower by fine-tuning B_s and B_t; our proposed parameters in Section 4.6 reflect this optimization.

For this section we assume that the various parameters satisfy the conditions given in Table 4.1.

The rejection criterion says that we only accept signatures whose norm is smaller than $q/2 - B$, so we want q to be a lot larger than B, or it will be too hard to find an acceptable signature. We consider the infinity norm of a potential signature

$$(\boldsymbol{s}, \boldsymbol{t}) = (\boldsymbol{s}_0, \boldsymbol{t}_0) + (\boldsymbol{a}\boldsymbol{f}, \boldsymbol{a}\boldsymbol{g})$$

produced in Step 7 of the signing algorithm. The coefficients of $\boldsymbol{s}_0$ and $\boldsymbol{t}_0$ are in $\mathcal{C}(q/2)$, the coefficents of $\boldsymbol{a}\boldsymbol{f}$ are in $\mathcal{C}(p^2N/4)$, and the coefficients of $\boldsymbol{a}\boldsymbol{g}$ are in $\mathcal{C}(pN/2)$. Hence the coefficients of an $(\boldsymbol{s}, \boldsymbol{t})$ pair produced by Step 7 satisfy

$$\|(\boldsymbol{s}, \boldsymbol{t})\|_\infty \leq \frac{q}{2} + B. \tag{4.12}$$

N	a prime, say, $200 < N < 5000$
p	a small prime chosen so that $N \log_2(p)$ is greater than the desired bit security
B	$\leq \lceil p^2 N/4 \rceil$
k	a small constant, say $2 \leq k \leq 50$
q	an integer coprime with p and satisfying $q \approx kNB \approx kp^2N^2/4$

Table 4.1: Parameter guidelines

We will make the simplifying assumption[1] that the coefficients of $\boldsymbol{s}$ and $\boldsymbol{t}$ are equally likely to take on each of the values in the interval (4.12). The rejection criterion says that we only accept signatures whose coefficents are at most $q/2 - B$. Since we need all $2N$ of the coefficients of $(\boldsymbol{s}, \boldsymbol{t})$ to satisfy this condition, we find that

$$\text{Prob}\big((\boldsymbol{s}, \boldsymbol{t}) \text{ is accepted}\big) \approx \left(\frac{q/2 - B}{q/2 + B}\right)^{2N}.$$

Using the chosen value

$$q \approx \frac{kp^2N^2}{4} \approx kNB$$

from Table 4.1, we find that

$$\begin{aligned}\text{Prob}\big((\boldsymbol{s}, \boldsymbol{t}) \text{ is accepted}\big) &\approx \left(\frac{1 - 2B/q}{1 + 2B/q}\right)^{2N} \\ &\approx \left(\frac{1 - 2/kN}{1 + 2/kN}\right)^{2N} \\ &\approx e^{-8/k},\end{aligned}$$

where for the last equality we use the estimate $(1 + t/n)^n \approx e^t$, valid when t is small and n is large.

[1] In actuality, the coefficients of the products $\boldsymbol{af}$ and $\boldsymbol{ag}$ tend to cluster more towards 0, since they are more-or-less hypergeometrically distributed.

4.6 NTRUMLS Parameters

Ideally the parameters that NTRUMLS has in common with NTRUEncrypt, namely N, p, q, d_f, d_g, would be derivable in exactly the same manner for each system. This would provide some assurance that at least the key recovery problem in NTRUMLS is precisely as hard as key recovery in NTRUEncrypt. In order to attain a low rejection rate during signing it is, unfortunately, necessary to take q larger than one would for the corresponding NTRUEncrypt parameter set. As the ratio of N to q has a strong impact on security, this implies that NTRUMLS will generally be weaker than NTRUEncrypt for the same choice of N.

In the paper describing NTRUMLS the impact of such a large q was ignored, and consequently the bit security estimates were severely overestimated. In Table 4.2 you can find the original parameter sets, the original security estimates, and the revised security estimates following the analysis of Chapter 5. The revised security estimates are based on the cost of key recovery using the hybrid attack of Section 5.5 and Equation 5.19.

	Set #1	Set #2	Set #3	Set #4
N	401	439	593	743
p	3	3	3	3
$\log_2 q$	18	19	19	20
B_s	240	264	300	336
B_t	80	88	100	112
d_1, d_2, d_3	8,8,6	9, 8, 5	10, 10, 8	11, 11, 15
Key & signature size (bytes)	853	988	1335	1765
$\approx$ Prob[accept]	38%	55%	41%	53%
$\approx$ bit security [36]	112	128	192	256
$\approx$ revised bit security	65	70	110	146

Table 4.2: Sample NTRUMLS Parameters from [36]

These parameters were chosen to mirror the existing EESS #1 parameter sets for NTRUEncrypt at the time. A revised set, using the same N that can be found in the more recent version of EESS #1 , can be found in Table 4.3. The main difference here is that q has been decreased, with a corresponding increase in the rejection probability, even when N has been increased. The evaluation of these parameter sets, and other improvements to the NTRUMLS scheme, is a topic of ongoing research.

	Set #1	Set #2	Set #3	Set #4
N	443	563	743	907
p	3	3	3	3
$\log_2 q$	16	16	17	17
B_s	138	174	186	225
B_t	46	58	62	75
d_1, d_2	9,8,5	10, 9, 8	11, 11, 6	13, 12, 7
Public key size (bytes)	886	1126	1579	1927
Signature size (bytes)	831	1056	1486	1814
$\approx$ Prob[accept]	8%	2%	6%	2%
$\approx$ bit security	88	126	179	269

Table 4.3: Revised NTRUMLS Parameters

Chapter 5

Cryptanalysis

We now move on to the cryptanalysis of NTRUEncrypt and NTRUMLS. Both schemes depend on the hardness of the approximate closest and shortest vector problems in R_N-module lattices, and our primary goal will be to estimate the concrete difficulty of such problems. We begin with a discussion of a simple approximation algorithm, Babai's Nearest Plane algorithm, for the Closest Vector Problem. This algorithm will be used extensively by the more sophisticated techniques of the later sections.

5.1 Approximating a closest vector

Recall the setup to a Closest Vector Problem: we are given a basis $B = \{\boldsymbol{b}_i\}$ of a rank n lattice L, and a target point $\boldsymbol{v} \in L \otimes \mathbb{R}$. The goal is to find $\boldsymbol{x} \in L$ such that $\|\boldsymbol{v} - \boldsymbol{x}\|$ is minimized.

Perhaps the most obvious approach is to express $\boldsymbol{v}$ as $\sum_{i=1}^{n} \alpha_i \boldsymbol{b}_i$, with $\alpha_i \in \mathbb{R}$, and then let $\boldsymbol{x}$ be the point obtained by rounding each α_i to the nearest integer. While conceptually simple, a treatment in which each coefficient is rounded independently is tantamount to assuming that the input basis is orthogonal. Such an approach fails to leverage the available information about the non-orthogonality of the input basis.

An alternative approach, the Nearest Plane algorithm, due to Babai [8], addresses the issue of the non-orthogonality of the input basis using the Gram-Schmidt vectors. Specifically, it solves a sequence of one-dimensional CVP instances in projected sublattices spanned by individual Gram-Schmidt vectors.

Babai's nearest plane algorithm starts by solving the one-dimensional CVP for $\pi_{B,n}(\boldsymbol{v})$ in $B^{(n)} = \pi_{B,n}(L)$. It then iteratively extends this solution to an approximate closest vector in $B^{(n-1)}$, then $B^{(n-2)}$, and so on until it reaches $B^{(1)} = L$. Partial solutions are lifted from $B^{(i)}$ to $B^{(i-1)}$ by the natural injective map given by

$$\pi_i(\boldsymbol{b}_k) \mapsto \pi_{i-1}(\boldsymbol{b}_k) \quad \forall k \in [0, N-1].$$

This map clearly does not increase the length of the vector's projection in the $B^{(i)} \otimes \mathbb{R}$ subspace, however it does introduce some component in the $\pi_{i-1}(\boldsymbol{b}_{i-1}) = \boldsymbol{b}^*_{i-1}$ direction. Hence, through this lifting, a partial solution in $B^{(i)}$ yields a CVP instance in $B^{(i-1)}$.

While this sounds complicated, the algorithm itself (Alg. 10) is the essence of simplicity. Each one-dimensional CVP instance is solved by rounding an easily computed real number to the nearest integer.

Algorithm 10 Babai's Nearest Plane

Input: $(B, B^*, \boldsymbol{t})$ where
B is a basis of a lattice L of rank n.
B^* is the Gram-Schmidt orthogonalization of B.
$\boldsymbol{t} \in L \otimes \mathbb{R}$ is the target point.
Output: A vector in L close to $\boldsymbol{t}$.

```
w = 0
for i = n to 1 do
    c = ⟨t − w, b*_i⟩ / ⟨b*_i, b*_i⟩
    w = w + ⌊c⌉ b_i
end for
return w
```

The output of Algorithm 10, hereafter the *Babai point* of $\boldsymbol{t} \in L \otimes \mathbb{R}$ with respect to the basis B, will approximate the true closest vector to $\boldsymbol{t}$ by a factor that depends crucially on the "shape" of the input basis. Babai proved that when B is LLL reduced (see Section 5.3) the distance between the $\boldsymbol{t}$ and the corresponding Babai point is no more than $2^{n/2-1}\|\boldsymbol{b}^*_n\|$ times the distance between $\boldsymbol{t}$ and the closest lattice point [8].

5.2 Enumeration

There are two ways to improve the quality of the approximation attained by Babai's Nearest Plane algorithm. The first, which we will address in coming sections, is to start with a

"better" basis. The second is to replace the choice of $\lfloor c \rceil$ in Line 4 with a search over integers in some interval about c.

The enumeration algorithms of Fincke-Pohst [23], Kannan [47], Schnorr-Euchner [66], and Gama-Regev-Nguyen [26], are all variations on this theme. These algorithms differ either in the initial preprocessing performed on the input basis, or in the strategy used to control the size of the search space.

An *enumeration tree* for a basis B is defined by a partial order $\leq$ on $\overline{L} = \cup_{i=0}^{n} B^{(i)}$ where $\boldsymbol{x} \leq \boldsymbol{y}$ iff there exists $i \in [0, n]$ such that $\pi_{B,i}(\boldsymbol{y}) = \boldsymbol{x}$. This partial order defines a tree in which the nodes of depth i are elements of $B^{(n-i+1)}$; in particular, the root node is $B^{(n+1)} = \{0\}$ and the leaves are elements of $B^{(1)} = L$.

Since $\pi_{B,i}(\boldsymbol{y}) \leq \boldsymbol{y} + c\boldsymbol{b}_{i-1}^*$ for $c \in \mathbb{Z}$ the tree is of infinite width, however we can "prune" the tree down to a finite set by imposing a norm constraint on the leaf nodes. For example, let $r \in \mathbb{R}$ and consider only leaf nodes in $L \cap \mathcal{B}(r)$. Since the norm is monotonic with respect to the partial order on $\overline{L}$ this intersection yields a finite tree.

In slightly greater detail, if a lattice point $\boldsymbol{x}$ has squared length at most r, then, given the ith projection of $\boldsymbol{x}$, the admissible values for x_{i-1} are constrained by the requirement that

$$x_{i-1}^2 \leq \frac{r - \|\pi_{B,i}(\boldsymbol{x})\|^2}{\|\boldsymbol{b}_{i-1}^*\|^2} - \sum_{j=i}^{n} \mu_{j,i} x_j.$$

This can then be iterated to obtain restrictions on all values $j < i$. Alternatively, we may start from the root node, select $r \geq 0$, and obtain bounds on all n coefficients:

$$\begin{aligned} x_n^2 \|\boldsymbol{b}_n^*\|^2 &\leq r \\ (x_{n-1} + \mu_{n,n-1} x_n)^2 \|\boldsymbol{b}_{n-1}^*\|^2 &\leq r - x_n^2 \|\boldsymbol{b}_n^*\|^2 \\ &\vdots \\ \left(x_1 + \sum_{j=2}^{n} \mu_{j,1} x_j\right)^2 \|\boldsymbol{b}_1\|^2 &\leq r - \sum_{k=2}^{n} \left(\left(x_k + \sum_{j=k+1}^{n} \mu_{j,k} x_j \right)^2 \|\boldsymbol{b}_k^*\|^2 \right). \end{aligned}$$

One's goal in selecting r is to minimize the number of candidates that must be enumerated, and hence the expected running time, while keeping at least one non-trivial leaf in the pruned tree. Clearly the optimum value is $r = \lambda_1(L)$, but in general this quantity is unknown. Kannan showed in [47] that using an LLL reduced basis and taking r to be the length of the first vector of said basis results in a runtime of $n^{O(n)}$. Of course these

inequalities may be quite slack for an actual shortest vector. The first, with $r = \lambda_1(L)$ for instance, would have one search for a shortest vector that is parallel to $\boldsymbol{b}_n^*$ even when the length of $\boldsymbol{b}_n^*$ would suggest that such a vector is unlikely to exist.

Using the Gaussian Heuristic one expects the number of nodes at level k to be [27]:

$$H_k = \frac{1}{2} \cdot \frac{\mathrm{vol}(\mathcal{B}_k(\sqrt{r}))}{\prod_{i=n+1-k}^{n} \|\boldsymbol{b}_i^*\|}, \tag{5.1}$$

and the total number of nodes enumerated to be approximately $\sum_{i=1}^{n} H_k$. Various rigorous bounds are known for different choices of r; see [27] and references therein.

A slight change in strategy makes it possible to consider exponentially smaller (though still exponentially sized) search spaces. Rather than a constant bound, $\pi_{B,i}(\boldsymbol{x}) \leq r$, applied at all levels, one can choose n bounds $r_n \leq r_{n-1} \leq \cdots \leq r_1$ and ask only that $\pi_{B_i}(\boldsymbol{x}) \leq r_i$. Pruning strategies of this sort were first considered by Schnorr and Euchner [66], and then expanded upon by Schnorr and Hörner [67]. The later work considered pruning strategies for which the probability that a non-trivial leaf would remain was less than 1.

Determining optimal choices for the r_i can be quite difficult. The current state of the art is due to Gama, Nguyen, and Regev [27]. They suggest minimizing formulae similar to Eq. 5.1 with the k-ball replaced by a particular intersection of cylinders. A surprising outcome of their work is that one can get a roughly $2^{n/2}$ reduction in the number of nodes enumerated while maintaining a non-negligible probability of success. Doing so involves choosing very aggressive bounds, a strategy they call *extreme pruning*,

We will revisit extreme pruning after discussing lattice reduction; it is an essential subroutine of the BKZ-2.0 algorithm and informs our "bit-security" estimates for NTRU-Encrypt and NTRUMLS parameters.

5.3 Lattice reduction

Bounds on the runtime of enumeration algorithms depend crucially on the "quality" of the input basis. One's experience in dimensions one, two, and three might suggest that one should be able to define a "best" or shortest basis for a lattice, and that this best basis should consist of minimal elements, those contained in the ball of radius $\lambda_n(L)$. This intuition is flawed; Conway and Sloane proved in 1995 that there is a lattice of rank $n = 11$ that is generated by its minimal vectors but for which no basis may be found in that set [18]. Regardless, there are ways to meaningfully rank the quality of bases. In particular

there are ways to say that one basis is more "reduced" than another. It is even possible, in some sense, to define a set of bases that are best. The important question then becomes, for any particular definition of reduction: is there an efficient algorithm for producing a "reduced" basis?

Lattice reduction has a long history, but the major breakthrough came in 1982, when Lenstra, Lenstra, and Lovász introduced a new notion of reduction and a polynomial time algorithm to find a basis satisfying it [52].

Using the notation of Definition 8 the definition they gave was

Definition 18 (LLL reduced basis)**.** A basis $\{\boldsymbol{b}_i\}_{1\leq i\leq n}$ of a rank n lattice is *LLL reduced* if

1. For all $1 \leq j < i \leq n$
$$|\mu_{i,j}| \leq \frac{1}{2} \tag{5.2}$$
2. For all $1 \leq i \leq n$
$$\left(\frac{3}{4} - \mu_{i+1,i}^2\right) \|\boldsymbol{b}_i^*\|^2 \leq \|\boldsymbol{b}_{i+1}^*\|^2. \tag{5.3}$$

Equation 5.2 is typically referred to as the *size-reduction* condition, and a basis that meets only this condition is deemed *size-reduced*. Equation 5.3 is referred to as the Lovàsz condition. The constant $\frac{3}{4}$ can be replaced by any value $\delta \in (\frac{1}{4}, 1)$, and we get a corresponding notion of δ-LLL reduction. A δ close to $1/4$ gives the fast runtime, while a δ close to 1 gives a better reduction.

The following lemma encapsulates a few well known facts about LLL-reduced bases. A proof may be found in [17], or in practically any reference on LLL.

Lemma 16 (LLL approximation factors)**.** *If $\{\boldsymbol{b}_i\}$ is an LLL reduced basis of a rank n lattice L, then*

$$\|\boldsymbol{b}_1\| \leq 2^{(n-1)/2} \cdot \lambda_1(L), \tag{5.4}$$
$$\|\boldsymbol{b}_1\| \leq 2^{(n-1)/4} \cdot \mathrm{vol}(L)^{1/n}. \tag{5.5}$$

The figure of merit typically used to describe lattice reduction algorithms, since $\lambda_1(L)$ is rarely known, is the analogue of the constant in Equation 5.5 and is called the *Hermite factor*.

Definition 19 (Hermite factor)**.** Given a basis for a lattice of rank n an algorithm achieves *Hermite factor* δ^n, or *root Hermite factor* δ, if it returns a basis with $\|\boldsymbol{b}_1\| \leq \delta^n \mathrm{vol}(L)^{1/n}$.

This is not to be confuesd with Hermite's constant γ_n for lattices in $\mathbb{Z}^n$, which is the least value such that for all $L \subseteq \mathbb{Z}^n$ and $1 \leq d \leq n$

$$\left(\prod_{i=1}^{d} \lambda_i(L)\right)^{1/d} = \sqrt{\gamma_n}\mathrm{vol}(L)^{1/n}.$$

If the inequality of Equation 5.3 is tight for all i, the lengths of the Gram-Schmidt vectors of an LLL-reduced basis decay geometrically. The Geometric Series Assumption (GSA), introduced by Schnorr in [68], is the assumption that these inequalities are, on average, satisfied with equality.

Definition 20 (GSA)**.** For "LLL-like" reduction methods there is a constant $\eta < 1$ such that reduced bases satisfy, on average,

$$\|\boldsymbol{b}_i^*\|/\|\boldsymbol{b}_1\| = \eta^{i-1}.$$

The validity of this assumption should be checked experimentally in low dimensions for any family of lattices to which one wishes to apply it. For NTRU, and other q-ary lattices, the presence of easy-to-find vectors of length q causes the GSA to fail for small i unless exceptionally strong lattice reduction is used. However, there typically exists an index beyond which the assumption holds, and a suitable variant of the assumption may be used.

It is often useful to translate between the GSA slope η and the root Hermite factor δ that would yield such a slope. Substituting $\|\boldsymbol{b}_1\| = \delta^n \mathrm{vol}(L)^{1/n}$ into the GSA assumption yields

$$\|\boldsymbol{b}_i^*\| = \eta^{-(i-1)}\delta^m \det(\Lambda)^{1/m}.$$

Furthermore since the determinant of the lattice is invariant under change of basis and equal to the product of the Gram-Schmidt Lengths we have

$$1 = \frac{1}{\det(\Lambda)} \prod_{i=1}^{m} \|\boldsymbol{b}_i^*\| = \eta^{-m(m-1)/2}\ \delta^{m^2}.$$

Hence

$$\eta = \delta^{2m/(m-1)}. \tag{5.6}$$

Stronger definitions of reduction are also known, although not ones that achieve a subexponential approximation factor in polynomial time. The strongest notion is that of Korkine-Zolatarev reduction. Briefly, a Korkine-Zolatarev reduced basis is one that is sized reduced, and satisfies

$$\|\boldsymbol{b}_i^*\| = \lambda_1(B^{(i)}).$$

That is to say $\boldsymbol{b}_i^*$, the first non-zero vector of the projected basis $\pi_{B,i}(B)$ is a shortest vector in the lattice $B^{(i)}$. This is hopelessly difficult to achieve for large dimensions. A far more useful notion of reduction applies the Korkine-Zolotarev condition to each of the n sublattices spanned by successive blocks of (at most) k basis vectors from the input. The LLL algorithm is a special case with $k = 2$.

Definition 21 (Block Korkine-Zolatarev (BKZ) reduced with blocksize k)**.** A basis $\{\boldsymbol{b}_i\}_{1 \le i \le n}$ of a rank n lattice is *BKZ reduced with blocksize* k if it is size reduced and

$$\|\boldsymbol{b}_i^*\| = \lambda_1\left(\pi_{B,i}(\{\boldsymbol{b}_i, \ldots, \boldsymbol{b}_j\})\right) \tag{5.7}$$

where $j = \min(i + k, n)$.

As the definition makes clear, achieving BKZ reduction involves solving SVP in projected sublattices of rank (at most) k. Schnorr proposed this definition in [65], but it was several years before Schnorr and Euchner published an algorithm achieving the BKZ definition [66].

Algorithm 11 BKZ

Input: An LLL reduced basis B_{in} and a blocksize β
Output: A BKZ-β reduced basis B

```
B = B_in
repeat
    for k = 1 to n − 1 do
        Find b_new s.t. π_{B,k}(b_new) is a shortest vector in span_Z(π_{B,k}({b_k, ..., b_{k+β}}))
        Insert b_new into B at index k, remove linear dependency with LLL.
    end for
until no change in B occurs
return w
```

The runtime of their algorithm is difficult to analyze. It is not known, for instance, whether the number of calls to the SVP subroutine is even polynomially bounded. However in practice [26], and in reasonable theoretical models [32], this appears to be the case.

5.3.1 BKZ Simulation

For a given β we would like to determine the Hermite factor achieved by BKZ-β and the cost of achieving that factor in terms of, say, the number of calls to the enumeration subroutine. Any method of doing this will rely on heuristics that must be confirmed, by experiment, for any particular class of lattices.

A simulation based method for determining the Hermite factor achieved by BKZ-β was proposed by Chen and Nguyen in [16]. It makes extensive use of both the Gaussian Heuristic and the Geometric Series Assumption, and additionally requires experimentally derived average Gram-Schmidt vector lengths for random Korkine-Zolatarev reduced bases in small dimension (typically ≤ 50).

Very recently Micciancio and Walter have proposed a "self-dual BKZ" algorithm the analysis of which suggests that simulation is largely unnecessary [59]. In large dimension, say $\beta > 100$, their experiments with self-dual BKZ indicate that the mean value of the root Hermite factor achieved by block reduction with blocksize β is

$$\delta(\beta) \approx \mathrm{GH}(\mathbb{Z}^\beta)^{1/(\beta-1)}.$$

They emphasize that this is the *mean* root Hermite factor for a certain class of random lattices, and little is known about, say, the standard deviation.

5.4 Meet-in-the-middle attacks

The private keys of NTRUEncrypt and NTRUMLS are sparse enough that direct combinatorial attacks on them are often competitive with lattice based techniques.

The simplest approach is to enumerate candidates $\boldsymbol{v}$ and check whether $\boldsymbol{v} \cdot (1, \boldsymbol{h})$ is short. If $\mathcal{S}$ is the space from which the true private key $\boldsymbol{f}$ component was drawn then by enumerating $\boldsymbol{v} \in \mathcal{S}$ this approach will recover the private key after $O(|\mathcal{S}|/N)$ attempts. The optimal $\mathcal{S}$ for such an attack is small enough that it is unlikely that any choice of $\boldsymbol{v}$ other than $\boldsymbol{f} \cdot \boldsymbol{x}^k$ would yield a short vector, hence the stated complexity.

One can do substantially better by leveraging a "meet-in-the-middle" style time/memory tradeoff. Let $\mathcal{S}'$ be a set such that $\mathcal{S} \subseteq \{\boldsymbol{a} - \boldsymbol{b} \ : \ \boldsymbol{a}, \boldsymbol{b} \in \mathcal{S}'\}$ and suppose that $\boldsymbol{f} = \boldsymbol{s}_1 - \boldsymbol{s}_2$ with $\boldsymbol{s}_1, \boldsymbol{s}_2 \in \mathcal{S}'$. Then since $\boldsymbol{f} \cdot (1, \boldsymbol{h}) = (\boldsymbol{f}, \boldsymbol{g}) \bmod q$ we have

$$\boldsymbol{s}_1 \cdot (1, \boldsymbol{h}) = (\boldsymbol{f}, \boldsymbol{g}) + \boldsymbol{s}_2 \cdot (1, \boldsymbol{h}) \bmod q.$$

Since $\boldsymbol{f}$ and $\boldsymbol{g}$ have small coefficients we infer that

$$\boldsymbol{s}_1 \cdot (1, \boldsymbol{h}) \approx \boldsymbol{s}_2 \cdot (1, \boldsymbol{h}) \bmod q,$$

and hence we can find $\boldsymbol{f}$ by searching for approximate collisions in $\mathcal{S}' \cdot (1, \boldsymbol{h})$.

The adaptation of meet-in-the-middle search algorithms to the structure of binary NTRU keys is due to Odlyzko and described in [71]. Generalizations to other private key types are described by Howgrave-Graham in [42]. In the best case $|\mathcal{S}'| = O(\sqrt{|\mathcal{S}|})$, so under the assumption that all approximate collisions can be detected, a meet in the middle search on the full product form NTRUEncrypt key space would require both time and memory of order $O(\sqrt{|\mathcal{P}_N(d_1, d_2, d_3)|})$. Similarly for trinary keys the cost would be $O(\sqrt{|\mathcal{T}_N(d_f + 1, d_f)|})$.

Of course, it's possible that the map $\boldsymbol{s} \mapsto \boldsymbol{s} \cdot (1, \boldsymbol{h})$ is not optimal for performing a collision search, and in fact this appears to be the case. Suppose one has guessed the last K coefficients of a shortest vector $\boldsymbol{v}$, and let $\boldsymbol{w}$ be the corresponding zero-prefixed vector, i.e. $w_i = 0$ for $i \in [0, n-K]$ and $w_i = v_i$ for $i \in [n-K, n-1]$. Then the Babai point of $\boldsymbol{w}$ with respect to a suitably reduced basis may be $\boldsymbol{v}$.

Schnorr's "Generalized Birthday Sampling" [68] and Howgrave-Graham's "Hybrid attack" [42] both apply a meet-in-the-middle collision search to vectors output by the Nearest Plane algorithm, as we shall see in the next section.

5.5 The hybrid attack

How should one incorporate knowledge of a particular coefficient distribution into pruned enumeration? Such knowledge is common for lattice cryptosystems – if for no other reason than that the key generation procedure is known. For NTRUEncrypt and NTRUMLS the fact that the private keys are sampled from a distribution on sparse trinary polynomials provides a tremendous hint for enumerating short vectors. However, the common pruning strategies are too coarse to exploit this structure.

In 2007 Nick Howgrave-Graham developed an attack on NTRUEncrypt that led to a major re-evaluation of the parameter sets that had been proposed until that time [42]. The attack was presented as a combination of lattice reduction based preprocessing and meet-in-the-middle combinatorial search, and was dubbed "the hybrid attack." It makes essentially no use of the algebraic structure of the NTRU lattice under attack, and can be readily generalized to arbitrary lattices. In the following section we present a general exposition

using the language of projected lattices (Definition 9), and re-interpret the attack as a form of pruned enumeration. In the sequel we present a more concrete description that follows Howgrave-Graham's paper.

5.5.1 Preprocessing for general lattices

The hybrid attack attack begins by partitioning the lattice into a set of projected sublattices lying in orthogonal subspaces. The input is a Hermite normal form basis, $H = \{\boldsymbol{h}_i\}_{1 \leq i \leq n}$, for a euclidean lattice L of rank n. Indices r_1 and r_2 are chosen such that $1 \leq r_1 < r_2 < n$, and the basis is partitioned into three sets:

$$\begin{aligned} H_1 &= \{\boldsymbol{h}_i : 1 \leq i < r_1\} \\ H_2 &= \{\boldsymbol{h}_i : r_1 \leq i < r_2\} \\ H_3 &= \{\boldsymbol{h}_i : r_2 \leq i < n\}. \end{aligned}$$

The aforementioned projected sublattices are

$$\begin{aligned} L_1 &= \mathrm{span}_{\mathbb{Z}}(\pi_{H,1}(H_1)) = \mathrm{span}_{\mathbb{Z}}(H_1) \\ L_2 &= \mathrm{span}_{\mathbb{Z}}(\pi_{H,r_1}(H_2)) \\ L_3 &= \mathrm{span}_{\mathbb{Z}}(\pi_{H,r_2}(H_3)). \end{aligned}$$

Each serves a distinct purpose: vectors are enumerated in L_3, lifted to L after computing an approximate closest vector in L_2, and finally "binned" according to their value in $\mathbb{Z}^n/L_1$. With suitable assumptions on the enumerated vectors, and the quality of the approximate closest vectors found in L_2, the difference between the vectors forming an approximate collision under the binning function will be a short vector in L.

Before delving into the details it should be noted that, in the author's opinion, no analysis of the hybrid attack presented thus far is entirely satisfactory. The discussion here will not substantially improve the matter, but it is hoped that future work will answer some of the outstanding questions related to the attack's probability of success as a function of the effort spent in lattice reduction and enumeration.

The three lattices, L_1, L_2, and L_3 lie in mutually orthogonal subspaces of $L \otimes \mathbb{R}$. Hence each element of L_3, when presented as an integer combination of $\pi_{H,r_2}(H_3)$, can be mapped to a point in $\text{Ł}_2 \otimes \mathbb{R}$ by composing the natural injective map from $L_3 \to L$ given by

$$\sum_{i=r_2}^{n} v_i \pi_{H,r_2}(\boldsymbol{h}_i) \mapsto \sum_{i=r_2}^{n} v_i \boldsymbol{h}_i$$

with the projection $\pi_{H,r_1} : L \to L_2 \otimes \mathbb{R}$. We will refer to a point in $L_2 \otimes \mathbb{R}$ produced in this way as the CVP Instance *induced by* the element of L_3.

The attack proceeds in two stages:

1. Lattice reduction is applied to L_2 to produce a reduced basis B_2.
2. A subset $\mathcal{S} \subset L_3$ is identified for which the projection of a shortest vector in L is likely to be found in $\mathcal{S} \oplus \mathcal{S}$ (where $\oplus$ is the Minkowski sum). For each element of $\boldsymbol{v} \in \mathcal{S}$, the Babai point for the CVP instance induced by $\boldsymbol{v}$ with respect to the basis $H_1 \cup B_2$ is computed and stored in such a manner that approximate collisions between these points can be detected.

If $\boldsymbol{v}$ is a shortest vector in L then the quality of the reduction in Step 1 must be such that the CVP instance induced by $\pi_{H,r_2}(\boldsymbol{v})$ can be solved *exactly* using Babai's Nearest Plane algorithm (Algorithm 10). A sufficient condition on the quality of the reduction is given by Proposition 17.

Proposition 17 (Furst and Kannan [25])**.** *Suppose L is a euclidean lattice of rank n with a basis $\{\boldsymbol{b}_i\}_{1 \le i \le n}$ satisfying*

$$\|\boldsymbol{b}_i^*\| \ge 2k \quad \forall i.$$

Let $\boldsymbol{x} \in L \otimes \mathbb{R}$. There is at most one lattice point $\boldsymbol{v} \in L$ such that $\|\boldsymbol{x} - \boldsymbol{v}\| \le k$, and there is a polynomial time algorithm (Algorithm 10) that finds this point if it exists.

Proof. Uniqueness of $\boldsymbol{v}$ is argued by contradiction. Suppose there is a second lattice point $\boldsymbol{v}' \neq \boldsymbol{v}$ such that $\|\boldsymbol{x} - \boldsymbol{v}'\| \le k$. Then $\boldsymbol{v} - \boldsymbol{v}' \in L$ and $\|\boldsymbol{v} - \boldsymbol{v}'\| \le 2k$. Since $\boldsymbol{v} - \boldsymbol{v}'$ is a nonzero lattice point, it has a component in the $\boldsymbol{b}_i^*$ direction, for some i, that is at least 1 in magnitude. Hence

$$\frac{\langle \boldsymbol{b}_i^*, \boldsymbol{v} - \boldsymbol{v}' \rangle}{\|\boldsymbol{b}_i^*\|} \ge \|\boldsymbol{b}_i^*\| > 2k,$$

contradicting $\|\boldsymbol{v} - \boldsymbol{v}'\| \le 2k$.

Suppose $\boldsymbol{v}$ exists, we will show that Algorithm 10 finds it. The projection of $\boldsymbol{v} - \boldsymbol{x}$ in the $\boldsymbol{b}_n^*$ direction is of length at most k. Express $\boldsymbol{v} = \sum_{i=1}^n \alpha_i \boldsymbol{b}_i$ with $\alpha_i \in \mathbb{Z}$ and $\boldsymbol{x} = \sum_{i=1}^n \beta_i \boldsymbol{b}_i^*$ with $\beta_i \in \mathbb{R}$. Let j be the largest index such that α_j is non-zero. We have

$$\frac{\langle \boldsymbol{b}_j^*, \boldsymbol{v} - \boldsymbol{x} \rangle}{\|\boldsymbol{b}_j^*\|} = |\alpha_j - \beta_j| \|\boldsymbol{b}_j^*\| \le k \quad \text{hence} \quad |\alpha_j - \beta_j| \le \frac{k}{\|\boldsymbol{b}_j^*\|} < \frac{k}{2k} < 1/2$$

Hence the $\alpha_j = \lfloor \beta_j \rceil$ is the integer multiple of $\boldsymbol{b}_j$ selected in Line 4 of Algorithm 10. Iterating the argument with $\boldsymbol{x} - \alpha_j \boldsymbol{b}_j$, and $\boldsymbol{v} - \alpha_j \boldsymbol{b}_j$ in place of $\boldsymbol{x}$ and $\boldsymbol{v}$ shows that Algorithm 10 outputs $\boldsymbol{v}$ exactly. □

When the basis is given as a lower triangular matrix, B, the addition in Line 4 of Algorithm 10 affects only the first i coefficients of $\boldsymbol{w}$. Therefore, treating each loop iteration as a one-dimensional CVP , Proposition 17 can be applied to the individual coefficients of $\boldsymbol{x} - \boldsymbol{v}$.

Corollary 18. *Suppose L is a rank n lattice with a lower triangular basis B with rows $\{\boldsymbol{b}_i\}$, satisfying $\|\boldsymbol{b}_i^*\| \geq 2k$. If $\boldsymbol{x} \in L \otimes \mathbb{R}$ and there exists $\boldsymbol{v} \in L$ satisfying $\|\boldsymbol{x} - \boldsymbol{v}\|_\infty \leq k$ then Algorithm 10 outputs $\boldsymbol{v}$ on input $\boldsymbol{x}$.*

Note that any basis for a lattice, $L \subseteq \mathbb{Z}^n$, can be transformed into a lower triangular basis for an isomorphic lattice, $L' \subseteq \mathbb{R}^n$, by applying an orthogonal transformation on the right. Certainly the content of the corollary is identical to that of the proposition if such a transformation is applied: a bound of k on the coefficients of a shortest vector of L tells you little about the coefficients of a shortest vector of L'. The fact that the orthogonal transformation could map a shortest vector of L to a vector with a single coefficient equal to $\lambda_1(L)$ suggests the best bound we can take is $k\sqrt{n}$. However, if we assume the triangularizing transformation is random, then we should not expect such a conspiracy.

Howgrave-Graham's analysis of the hybrid attack in [42] models the triangularizing transformation as a random orthonormal matrix. Hence a vector of length ℓ is mapped uniformly at random to a point on the $(n-1)$-sphere of radius ℓ. Under such an assumption, the individual coefficients of a shortest vector in L' can be modeled as normally distributed random variables with standard deviation $\ell^2/\operatorname{rank}(L)$. Rather than relying on such an approximation, Howgrave-Graham computed empirical estimates for coefficient distribution of the transformed vectors.

With an estimate for the coefficient distribution in hand, one can determine the strength of the lattice reduction that must be applied to L_2 in order for the attack to succeed. This is explained in Section 5.5.3.

5.5.2 Matrix theoretic description for NTRU lattices†

One first chooses $1 \leq r_1 < r_2 < 2N$ and extracts a block, H_2', of $(r_2 - r_1) \times (r_2 - r_1)$ coefficients from the center of a $\mathbb{Z}$-basis[1] for the NTRU lattice in question:

$$H = \left(\begin{array}{c|c} qI_N & 0 \\ \hline H & I_N \end{array}\right) = \left(\begin{array}{c|c|c} qI_{r_1} & 0 & 0 \\ \hline * & H_2' & 0 \\ \hline * & * & I_{2N-r_2} \end{array}\right). \tag{5.8}$$

A lattice reduction algorithm is applied to find a unimodular transformation, U', such that $U'H_2'$ is reduced, and an orthogonal transformation, Y', is computed such that $T' = U'H_2'Y'$ is in lower triangular form. These transformations are applied to the original basis to produce a basis for an isomorphic lattice:

$$T = UHY \tag{5.9}$$

$$= \left(\begin{array}{c|c|c} I_{r_1} & 0 & 0 \\ \hline 0 & U' & 0 \\ \hline 0 & 0 & I_{2N-r_2} \end{array}\right) \left(\begin{array}{c|c|c} qI_{r_1} & 0 & 0 \\ \hline * & H_2' & 0 \\ \hline * & * & I_{2N-r_2} \end{array}\right) \left(\begin{array}{c|c|c} I_{r_1} & 0 & 0 \\ \hline 0 & Y' & 0 \\ \hline 0 & 0 & I_{2N-r_2} \end{array}\right) \tag{5.10}$$

$$= \left(\begin{array}{c|c|c} qI_{r_1} & 0 & 0 \\ \hline * & T' & 0 \\ \hline * & * & I_{2N-r_2} \end{array}\right). \tag{5.11}$$

Notice that the lattice corresponding to T is isomorphic to that corresponding to H and $(g, f)Y$ is a short vector in the former.

5.5.3 Choosing r_1 and r_2 for NTRU lattices

It is not necessary for the extracted block to be taken from the center of H, and it is sometimes useful to consider blocks shifted s indices to the top left along the main diagonal. The entries on the diagonal of T will have values $\{q^{\alpha_1}, q^{\alpha_2}, \ldots, q^{\alpha_{2N}}\}$, where $\alpha_1 + \cdots + \alpha_{2N} = N$, and the α_i, for i in the range $[r_1, r_2]$, will come very close to decreasing linearly. That is to say, the reduced basis for L_2 will roughly obey the geometric series assumption (GSA). The rate at which the α_i decrease can be predicted very well based on the root Hermite factor achieved by the lattice reduction algorithm used. Since L_2 lies in a hyperplane

[1] Note that we have permuted the blocks here to draw out a correspondence between the diagonal entries of the preprocessed basis and the Gram-Schmidt vector lengths, however the lattices generated by the rows of Eq. 5.8 and Eq. 3.2 are easily seen to be isomorphic.

orthogonal to both L_1 and L_3, the reduction applied to L_2 does not affect the lengths of the Gram-Schmidt vectors for $i \notin [r_1, r_2]$. Hence $\log_q(\|\boldsymbol{b}_i^*\|) = 1$ for $i < r_1$ and $\log_q(\|\boldsymbol{b}_i^*\|) = 0$ for $i > r_2$, i.e. the profile of the basis will look like one of the examples in Figure 5.1.

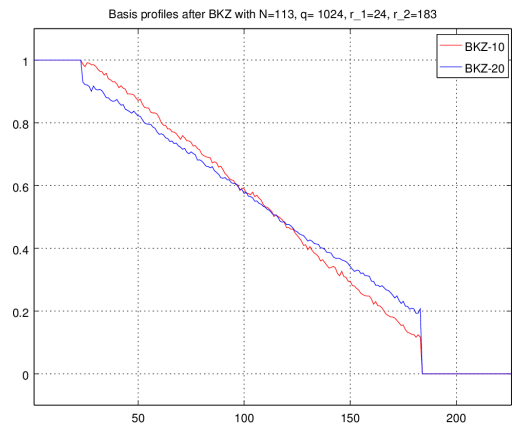

Figure 5.1: Log length of ith Gram-Schmidt vector, $\log_q(\|\boldsymbol{b}_i^*\|)$.

Let $\alpha_i = \log_q(\|\boldsymbol{b}_i^*\|)$. The "height of the cliff," the parameter that will determine whether an adversary can expect to lift short vectors in L_3 to L by Corollary 18, is α_{r_2}. Heuristically, we expect the attack to succeed with non-negligible probability provided that $\alpha_{r_2} > 2\|\boldsymbol{g}\|_\infty$.

In order to meaningfully apply the Geometric Series Assumption we will also need the expected value of α_{r_1} to satisfy $\alpha_{r_1} \leq 1$. Otherwise the Gram-Schmidt vectors near the beginning of the reduced block will be q-vectors, and the profile will be flat in a region where we have assumed it to be decreasing linearly.

Since the volume of L_2 is preserved throughout reduction we have

$$\log_q(\mathrm{vol}(L_2)) = \sum_{i=r_1}^{r_2} \alpha_i = \sum_{i=r_1}^{r_2} \left(\alpha_{r_1} - (i - r_1)\log_q \eta\right) = m\alpha_{r_1} - \frac{m(m+1)}{2}\log_q \eta,$$

which implies that

$$\alpha_{r_1} = \frac{\mathrm{vol}(L_2)}{m} + \frac{m+1}{2}\log_q \eta.$$

Applying the GSA yields

$$\alpha_{r_2} = \frac{\mathrm{vol}(L_2)}{m} - \frac{m-1}{2}\log_q \eta.$$

Substituting $\eta = \delta^{2m/(m-1)}$ (Eq. 5.6), this gives us

$$\alpha_{r_1} = \frac{1}{m}\log_q(\mathrm{vol}(L_2)) + m\frac{m+1}{m-1}\log_q(\delta) \approx \log_q(\mathrm{vol}(L_2)^{1/m} \cdot \delta^m) \tag{5.12}$$

$$\alpha_{r_2} = \frac{1}{m}\log_q(\mathrm{vol}(L_2)) - m\log_q(\delta) = \log_q(\mathrm{vol}(L_2)^{1/m}/\delta^m). \tag{5.13}$$

The cliff must have height at least $\log_q(2)$ for typical NTRU instantiations, hence the preprocessing for the hybrid attack requires lattice reduction that achieves a root Hermite factor, δ, satisfying:

$$\mathrm{vol}(L_2)^{1/m} \cdot \delta^m \leq q \tag{5.14}$$

$$\mathrm{vol}(L_2)^{1/m} \cdot \delta^{-m} \geq 2. \tag{5.15}$$

The optimal choice of r_1 and r_2 is determined by the balancing the cost of combinatorial search on $K = 2N - r_2$ coordinates against the cost of the minimal lattice reduction that satisfies these requirements. The blocksize and number of rounds of BKZ reduction required to reach root Hermite factor δ in dimension m can be estimated using the simulation method of Section 5.3.1. Finally the number of rounds can be multiplied by the number of calls to the enumeration subroutine per round, and the estimated cost per call, to obtain a rough estimate for the difficulty of performing the preprocessing step of the hybrid attack.

When using the classical meet-in-the-middle collision search rather than a quantum search the optimal choice of r_1 will also depend on the probability that near-collisions can be detected on L_1. For a quantum search the choice of r_1 and r_2 should be optimized over all pairs satisfying Equations 5.14 and 5.15.

5.5.4 Comparison with Lindner-Peikert Nearest Plane

If we perform only the preprocessing of the hybrid attack, to create a basis with a profile of the form shown in Figure 5.1, and perform an exhaustive search on L_3 instead of a meet-in-the-middle search, then the enumeration routine used in the hybrid attack can be seen to be identical to that presented by Lindner and Peikert in [53] (Algorithm 12). In particular the vector $\boldsymbol{d}$ defining the search space is given by $d_i = 0$ for $i \in [0, N-K]$, $d_i = 2$ for $i \in [N-K, N]$. Note of course that this is a slight simplification, an attacker would typically not exhaustively search all 3^K candidate vectors, as the distribution on candidates, conditional on knowledge of the key generation process, is non-uniform.

Algorithm 12 Lindner-Peikert Nearest Plane [53]

Input: $(B, B^*, \boldsymbol{d}, \boldsymbol{t})$ where
B is a basis of a lattice L of rank n.
B^* is the Gram-Schmidt orthogonalization of B.
$\boldsymbol{d} \in \mathbb{Z}^n$ is a vector of small positive integers.
$\boldsymbol{t} \in L \otimes \mathbb{R}$ is the target point.
Output: A vector in L close to $\boldsymbol{t}$.

```
S = ∅
for i = n to 1 do
    T = ∅
    for w ∈ S do
        Let c_1, ... c_{d_i} be the d_i integers nearest to ⟨t − w, b*_i⟩/⟨b*_i, b*_i⟩
        T = T ∪ {w + c_i b_i : 1 ≤ i ≤ d_i}
    end for
    S = T
end for
return S
```

Nguyen and Liu noted in [54] that the Lindner-Peikert Nearest Plane variant also subsumes the algorithm used by Schnorr's Random Sampling reduction, wherein $\boldsymbol{d}$ is given by $(1, 1, \ldots, 1, 2, \ldots 2)$ and the index of the transition from $1s$ to $2s$ is a parameter of the algorithm.

Nguyen and Liu go on to contextualize algorithms like Lindner-Peikert's Nearest Plane variant as *secondary prunings* of an enumeration tree

Propose the following variant:

Algorithm 13 Nguyen-Liu Nearest Plane [54]

Input: $(B, B^*, \boldsymbol{d}, \boldsymbol{t}, \ell)$ where
$(B, B^*, \boldsymbol{d}, \boldsymbol{t})$ are as in Algorithm 12
ℓ is a number of iterations

```
S = ∅
for i = 1 to ℓ do
    Randomize and reduce B to obtain a new reduced basis B′
    Call Algorithm 12 on (B′, (B′)*, d, t) and join the result to S
end for
return S
```

In the context of the hybrid attack, the Nguyen-Liu variant of Nearest Plane suggests producing a fixed number of random reduced bases for L_2 during preprocessing. Assuming the same quality of reduction is used each time, this gives an explicit means by which to randomize the triangularizing transformation that makes Corollary 18 useful. The assumption that the triangularizing transformation is a random orthogonal transformation is one of the more difficult to justify parts of the hybrid attack, but it may be possible to provide a rigorous justification if Algorithm 13 is used in place of Algorithm 12. I hope this will be explored in future work.

By sampling *exactly* from the distribution on L_3 induced by knowledge of the key-space, the hybrid attack can be seen to be the optimal secondary pruning of a GNR enumeration tree.

5.6 Bounded Distance Decoding in an isomorphic module

One may wonder whether the different key structures for NTRUEncrypt change the difficulty of key recovery. Perhaps the use of $\boldsymbol{f} = 1 + p\boldsymbol{F}$ produces harder problems than simply $\boldsymbol{f} \in \mathcal{T}_N(d_f + 1, d_f)$. Here we will show this is not the case.

Suppose $\boldsymbol{f} = 1 + 3\boldsymbol{F}$, then the point

$$3F \cdot (1, h) = (1 + 3F) \cdot (1, g/(1 + 3F)) - (1, h) = (3F, -h + g) \bmod q$$

is in the lattice. Since both F and g are small, this point is a good approximation to $(0, -h)$ (which is not in the lattice). So, naïvely, key recovery is no harder than bounded distance decoding with radius $\|(3F, g)\| \approx \sqrt{20N/3}/\lambda_1(L)$, but we can show that, in fact, key recovery is no harder than BDD with radius $\sqrt{4N/3}/\lambda_1(L)$ by considering the same problem in an isomorphic module.

This technique was used by Ducas and Nguyen to solve the NTRU Challenges in R_N for $N = 131, 139, 149, 163$, and 173 as follows.

Let u be a unit in $R_{N,q}$. The map $\tau_u : R_{N,q}^2 \to R_{N,q}^2$ defined by

$$(a, b) \mapsto (u^{-1}a, b)$$

is a bijection, and hence if M is an R_N-module then $\tau_u(M)$ is isomorphic, as an $R_{N,q}$-module, to M. Of course, for most u, τ_u does not preserve the distance function on R^2, i.e.

$\|(a,b)\| \neq \|\tau_u(a,b)\|$, so $\tau_u(M)$ and M are not isomorphic as lattices. Despite this, there may be short vectors of M whose images are also short in $\tau_u(M)$. We can use this property to construct instances of a bounded distance decoding problem with an expected distance from the closest vector that is smaller than what we could have constructed in M.

As an example,

$$\tau_3(3F, 3Fh) = \tau_3(3F, -h+g) = (F, -h+g) = (F, g) + (0, -h).$$

Hence the publicly constructable point $(0, -h)$ is valid input to the bounded distance decoding problem in $\tau_3(M)$ with promise $\approx \sqrt{4N/3}$, whereas it is only a valid instance in M if the promise is $\approx \sqrt{20N/3}$. While small, this difference in the BDD radii may have a noticeable impact on the practical performance of BDD solvers.

In their solutions to the NTRU challenges thus far, Ducas and Nguyen have applied lattice reduction to a $\mathbb{Z}$-basis for $\tau_3(M)$, e.g. the circulant basis corresponding to $\{(1, 3h), (0, q)\}$, and then applied the BDD algorithm of [54]. Their success up to $N = 173$ indicates that this is an extremely effective strategy.

Can one do better than τ_3? Clearly τ_{3F} would be a very nice, though improbable, choice. In general one might try to guess a factor F' of F such that F/F' is sparser than F. An obvious choice is $(x-1)$, since each one of F_1, F_2, and F_3 is divisible by $(x-1)$. This choice does not fit into the above description since $x-1$ is not a unit in R, however one can still consider the $R_{N,q}$-module generated by $(1, 3(x-1)h)$ in which $(F/(x-1), g-h)$ will be close to $(0, -h)$. For most F this will be a harder instance of BDD than that which would be obtained by considering the module generated by $(1, 3h)$ – heuristically, $F/(x-1)$ will only be sparser than F if F has a large number of adjacent coefficients with opposite signs.

5.7 Quantum Attacks

Quantum cryptanalysis is still a very new field – we know a few algorithms, Shor, Grover PIP, and their asymptotic run times. A few more we know to be polynomial time, but we do not have rigorous analyses of their exact runtimes. For a few more problems there are no proven improvements, but we might expect quantum heuristics to outperform classical techniques asymptotically. We are beginning to see what the first quantum computers capable of solving interesting problems will look like.

The field of quantum cryptanalysis is still in its infancy, and we are only able to give crude estimates for the costs of quantum computation. However it seems that many cryptanalysts are unaware even of those estimates that can be made today. This is evident in the

oft repeated claim that Grover search requires one to double the length of ones symmetric keys.

Certainly, if one is given a function $f : \{0,1\}^\ell \to \{0,1\}$ that is promised to satisfy $f(x) = 1$ iff $x = s$ for some unknown bitstring s, it is the case that one will have a high probability of determining s after $\Theta(2^{\ell/2})$ Grover iterations, whereas it would take $\Theta(2^\ell)$ calls to f classically. However to cost a Grover iteration and a call to f identically is to ignore the enormous overhead involved in performing quantum operations. Until this overhead can be reliably estimated we should keep in mind that while the call to double keysizes is sound advice for the conservative cryptographer, it is overly optimistic about the abilities of the cryptanalyst.

5.7.1 Quantum hybrid attack

The hybrid attack gives the lowest estimate for the time complexity of NTRU key recovery amongst all known classical attacks. Likewise for message recovery. Perhaps this is because it is overly optimisitic, in particular by ignoring the probability of failure in approximate collision search. In a quantum setting, there is an obvious algorithm with the same temporal complexity and fewer assumptions.

Briefly: replace the meet-in-the-middle search with Grover search. Let $\mathcal{M}$ be the set of projections of possible shortest vectors into L_3. An optimal parameterization of the classical hybrid attack would chose $\mathcal{S}$ of size $\Theta(\sqrt{|\mathcal{M}|})$ such that $\mathcal{M} \subseteq \mathcal{S} \oplus \mathcal{S}$, and would have to be tuned to ensure that approximate collisions can be identified. The quantum variant attempts to find a preimage (of norm $\lambda_1(L)$) of any element of $\mathcal{M}$ under π_{B,r_2}. The basis B is preprocessed in exactly the same way as in the classical attack. Preimages are identified by simply lifting an element of $\mathcal{M}$ from L_3 to the corresponding Babai point in L. Since $\lambda_1(L) = \|(\boldsymbol{f}, \boldsymbol{g})\|$ is known exactly, and there are likely only N lattice points with this norm, each with distinct projections, this requires $\Theta(\sqrt{\mathcal{M}/N})$ Grover iterations. Each Grover iteration has a cost $O(\mathsf{S} + \mathsf{C})$ where S is the cost to sample $\mathcal{M}$ and C is the cost of the nearest plane algorithm. These are negligible compared with the size of $\mathcal{M}$, but larger than N, hence the entire algorithm runs in time $|\mathcal{M}|^{1/2+o(1)}$. The only assumption remaining is that the preprocessing is sufficient for Corollary 18 to hold.

Can we do better? Would it be better to replace the meet-in-the-middle stage with a quantum collision search? Is the choice of r_1 and r_2 used for the classical attack also optimal for the quantum variant?

A number of papers have attempted to address the complexity of quantum variants of the hybrid attack [24, 76, 77, 79], however all of these assume an unrealistic model of

quantum computation in which the oracle used by the Grover iteration can be replaced with an exponentially large, say $|\mathcal{M}|^{1/3+o(1)}$ bit, table of classical data in which membership of an element can be queried, in superposition, in $O(1)$ time. There are no proposals for quantumly accessible classical memory for which this is realistic.

Time/memory trade-offs for collision search on quantum computers was studied in general in [15] and an algorithm with claimed $|\mathcal{M}|^{1/3}$ complexity was presented. However it was later argued that, when instantiated in a realistic model of quantum computation, this algorithm is no better than classical parallel collision search [10]. It is currently unknown how to achieve a quantum speedup for collision search in a data locality-sensitive model.

5.7.2 Attacking NTRUEncrypt keys

We will assume that we are given an explicit map $\{0,1\}^* \to \mathcal{T}_N(d_g+1, d_g)$ used for generating the $\boldsymbol{g}$ component of the private key. Any concrete instantiation of NTRUEncrypt in software must use some function of this sort. Let G denote the restriction of this function to inputs of length $\lceil \log_2(3)K \rceil$ bits – here we are assuming that the actual instantiation uses at least $\lceil \log(3)K \rceil$ bits to seed its random number generator, otherwise we can choose a more restricted input set.

Check$(\mathcal{I}, B, r_2, b)$

1. Use a IGF-2 with seed b to sample $\boldsymbol{g} \in \mathcal{T}_N(d_g+1, d_g)$.
2. Let $\boldsymbol{v} = \pi_{B,r_2}((0, \boldsymbol{g}))$
3. Lift $\boldsymbol{v}$ from L_3 to L using the nearest plane algorithm.
4. If the norm of the result is equal to $\lambda_1(L)$ output 1.
5. Otherwise output 0.

QHybridNTRU$(\mathcal{I}, H)$

1. Determine optimal hybrid attack parameters r_1, r_2.
2. Preprocess H to obtain a reduced basis B as in Section 5.5.2.
3. Let $t = \lceil \log_2 3 \cdot (N - r_2) \rceil$.
4. Let $H^{\otimes t}$ denote the Hadamard transform on t qubits.

5. Let Q_1 denote a quantum circuit that performs

$$|b\rangle \mapsto \begin{cases} -|b\rangle & \text{if } \mathsf{Check}(\mathcal{I}, B, r_2, b) \\ |b\rangle & \text{otherwise} \end{cases}$$

6. Let Q_2 denote a quantum circuit that performs

$$|b\rangle \mapsto \begin{cases} -|b\rangle & \text{if } b = 0 \\ |b\rangle & \text{otherwise} \end{cases}$$

7. Let $|\psi\rangle = H^{\otimes t}|0\rangle$
8. Apply the Grover iteration $(-H^{\otimes t} Q_2 H^{\otimes t} Q_1)$ a number of $2^{t/2}$ times.
9. Measure the result.

5.8 Approximate SVP in $\Lambda_q(\boldsymbol{f}, \boldsymbol{g})$ †

Key recovery in both NTRUEncrypt and NTRUMLS reduces to an approximate shortest vector problem in $\Lambda_q(\boldsymbol{f}, \boldsymbol{g})$, and we can therefore give the two schemes a unified treatment.

We will assume that the quantum hybrid attack is the best mechanism for exposing a short vector of the lattice, and we will make the extremely conservative choice of costing a Grover iteration as 1 operation. Thus we need only determine the quality of the lattice reduction needed in preprocessing to balance the cost of Grover search for the last K coefficients.

Our analysis will also assume that $\boldsymbol{g}$ is easier to search for than $\boldsymbol{f}$. For the key spaces recommended in this document, this assumption is valid.

Let $\Pi : \mathbb{Z}^N \to \mathbb{Z}^K$ be a projection[2] onto K coordinates of $\mathbb{Z}^N$.

For large K it is unlikely that the distribution on $\mathbb{Z}^K$ induced by sampling a key uniformly from the keyspace and projecting through Π will be close to uniform on $\mathcal{T}_K$, so we must consider an adversary that chooses to target a small set of high probability sequences. Consequently we must estimate the size of the set of elements that are typical under the projection.

[2]We will abuse notation slightly and allow Π to act on elements of R_N by acting on their coefficient vectors lifted to $\mathbb{Z}^N$.

Fix N, K, Π, $d = d_g + 1$, and $e = d_g$. Let $\mathcal{S} = \mathcal{T}_N(d, e)$. Let $p : \mathcal{T}_K \to \mathbb{R}$ be the probability mass function on $\mathcal{T}_K$ induced by sampling an element uniformly at random from $\mathcal{S}$ and projecting its coefficient vector onto $\mathbb{Z}^K$ through Π. We will estimate the size of the search space in the hybrid attack as, roughly, $2^{H(p)}$, where $H(p)$ is the Shannon entropy of p.

Let $\mathcal{S}_\Pi(a, b)$ be the subset of $\mathcal{S}$ consisting of vectors, v, such that $v\Pi$ has exactly a coefficients equal to $+1$ and b coefficients equal to -1. By the symmetry of $\mathcal{S}$ under coordinate permutations we have that $p(v\Pi) = p(v'\Pi)$ for all pairs $v, v' \in \mathcal{S}_\Pi(a, b)$. We choose a fixed representative of each type: $v_{a,b} = v\Pi$ for some $v \in \mathcal{S}_\Pi(a, b)$, and write

$$p(v_{a,b}) = \frac{1}{\binom{K}{a}\binom{K-a}{b}} \frac{|\mathcal{S}_\Pi(a, b)|}{|\mathcal{S}|} = \frac{\binom{N-K}{d-a}\binom{N-K-d+a}{d-b}}{\binom{N}{d}\binom{N-d}{d}}. \tag{5.16}$$

As there are exactly $\binom{K}{a}\binom{K-a}{b}$ distinct choices for $v_{a,b}$ this gives us:

$$H(p) = -\sum_{v \in \mathcal{T}_K} p(v) \log_2 p(v) = -\sum_{0 \le a,b \le d} \binom{K}{a}\binom{K-a}{b} p(v_{a,b}) \log_2 p(v_{a,b}). \tag{5.17}$$

The size of the search space is further decreased by a factor of N since $x^i * g$ is likely to be a distinct target for each $i \in [0, N-1]$.

A reasonable way to sample from the typical distribution on projected keys is to simply generate a random seed of length $H(p)$ and run the key generation algorithm with that seed. By rotational symmetry we expect that we can reduce the length of the seed to $H(p) - \log_2(N)$ and still have projections of short vectors in our set.

The cost of the Grover iteration, in $\log_2$ "operations," or bits, will then be

$$\frac{1}{2}(H(p) - \log_2(N)). \tag{5.18}$$

The only variable not fixed by the parameter set itself is K. In order to fix K we must consider the cost of lattice reduction.

The root Hermite factor required is determined by maximizing δ (large δ are easier to achieve than small δ) over the choice of r_1 and r_2 in Equations 5.14 and 5.15. Then the block size β, and the number of rounds required to reach δ can be obtained through simulation. Finally the number of nodes visited per call to the enumeration subroutine can be extrapolated from the estimates of [16]. Fitting curves to their table we have

$$\mathrm{LogNodes}(\beta) = 0.000784314\beta^2 + 0.366078\beta - 6.125$$

and following [16] in assuming a 2^7 cost per node, the total cost can be estimated as

$$\mathrm{Cost}(\beta, dim, \#rounds) = \mathrm{LogNodes}(\beta) + \log_2(dimension \cdot \#rounds) + 7. \tag{5.19}$$

One must also check that a naïve meet-in-the-middle search on the keyspace will not succeed with a lesser cost.

5.9 Other considerations for NTRUEncrypt

Message Recovery Attacks

Message recovery in NTRUEncrypt depends on the difficulty of Bounded Distance Decoding with a promise distance that is related to the minimum length of a message vector. The d_m constraint sets this length as $\sqrt{2d_m}$. The complexity is derived in [37], however in practice $d_m \approx N/3$ and message recovery has essentially the same cost as private key recovery using the analysis of the previous section.

5.10 Other considerations for NTRUMLS

Signature Forgery Attacks

To forge a signature on a document D one must find a point $(\boldsymbol{s}, \boldsymbol{t}) \in L$ of bounded norm such that

$$(\boldsymbol{s}, \boldsymbol{t}) \equiv \mathsf{Hash}(D) \pmod{p}.$$

Since p and q are coprime, and the lattice is q-ary, one can easily find elements of the lattice satisfying the congruence. The difficulty lies in finding a point that lies in the requisite hypercube.

A point in the correct equivalence class modulo p can be obtained by translating an arbitrary lattice point by a suitable multiple of q.[3] A natural strategy for constructing a forgery from such a point is to solve an approximate closest vector problem in $L \cap p\mathbb{Z}^{2N}$.

[3] Let $\boldsymbol{v} \in L$ and r be such that $rq \equiv 1 \pmod{p}$. Then $\boldsymbol{v} + r \cdot q \cdot ((\boldsymbol{s}_p, \boldsymbol{t}_p) - \boldsymbol{v}))$ is in L and in the correct equivalence class modulo p. One can reduce $(\boldsymbol{s}_p, \boldsymbol{t}_p) - \boldsymbol{v}$ modulo p in the above expression to obtain a shorter point.

The following lemma allows us to describe $L \cap p\mathbb{Z}^{2N}$ when L is the lattice associated to an NTRUMLS key.

Lemma 19. *Let q and p be coprime prime-powers in $\mathbb{Z}$. Let $\boldsymbol{h}$ be an arbitrary element of R_N. Then,*

$$\Lambda_q\left((1, \boldsymbol{h})\right) \cap p\mathbb{Z}^{2N} = \Lambda_{pq}\left((c, c\boldsymbol{h})\right).$$

where c is a constant that depends on p and q.

Proof. By the definition of $\Lambda_q\left((1, \boldsymbol{h})\right)$ we have that $\vec{\boldsymbol{y}} \in \Lambda_q\left((1, \boldsymbol{h})\right) \cap p\mathbb{Z}^{2N}$ iff there exists $\boldsymbol{s} \in R_N$ such that

$$\begin{aligned} \boldsymbol{s} \cdot (1, \boldsymbol{h}) &\equiv \vec{\boldsymbol{y}} \pmod{q}, \text{ and} \\ 0 &\equiv \vec{\boldsymbol{y}} \pmod{p}. \end{aligned}$$

Let c_p be such that $c_p p \equiv 1 \pmod{q}$. Suppose $\vec{\boldsymbol{y}} \in \Lambda_q\left((1, \boldsymbol{h})\right) \cap p\mathbb{Z}^{2N}$ and let $\boldsymbol{s}$ be such that $\boldsymbol{s} \cdot (1, \boldsymbol{h}) \equiv \vec{\boldsymbol{y}} \pmod{q}$. By the Chinese remainder theorem we have

$$\vec{\boldsymbol{y}} \equiv c_p \cdot p \cdot \boldsymbol{s} \cdot (1, \boldsymbol{h}) \equiv \boldsymbol{s} \cdot (c_p p,\ c_p p\boldsymbol{h}) \pmod{pq}.$$

By definition, such a $\vec{\boldsymbol{y}}$ is in $\Lambda_{pq}\left((c_p p,\ c_p p\boldsymbol{h})\right)$. The reverse direction follows by noting that $(c_p p,\ c_p p\boldsymbol{h}) \equiv (1, \boldsymbol{h}) \pmod{q}$. The claim follows with $c = c_p p$.

Note that if $c_p \not\equiv 0 \pmod{p}$ then we can take $c = p$. This is because the choice of generator is free up to a unit in $R_{N,pq}$. □

We can now attempt to analyze the difficulty of approximating CVP in $\Lambda_q\left((1, \boldsymbol{h})\right) \cap p\mathbb{Z}^{2N}$ to within a factor that is sufficient to forge NTRUMLS signatures. Of course, as we are sketching a particular cryptanalytic algorithm, we will only obtain an upper bound on the cost of forgery.

Let H be the Hermite normal form basis for $\Lambda_{pq}\left((c, c\boldsymbol{h})\right)$. The adversary selects r_1 and r_2 (in a manner that is to be described momentarily) and performs the hybrid preprocessing to obtain a reduced basis B. Let $\{\alpha_i = \log_{pq}(\|\boldsymbol{b}_i^*\|)\}_{1 \le i \le 2N}$ be the profile of the reduced basis. The quality of the reduction should be such that $\alpha_{r_1+1} \le \log_{pq}(q - 2B_s)$. If such reduction has been applied, then one can expect (making the same assumptions as for Corollary 18) that the final $2N - r_1$ coefficients of an approximate CVP solution found via Babai's Nearest Plane algorithm will have coefficients in $[-\frac{q}{2} + B_s, \frac{q}{2} - B_s]$. Each of the first r_1 coefficients will be correct with probability $1/p$. Hence all of the first r_1 coefficients will be correct with probability p^{-r_1}.

The optimal strategy for the adversary is to choose r_1 and r_2 such that the cost of lattice reduction to obtain the requisite profile is $\Theta(p^{r_1})$.

For instance, with $N = 443$ if an attacker selects $r_1 = 64$ and $r_2 = 2N$ they must reach a root Hermite factor of approximately 1.006672. Using the simulation method of Chen and Nguyen [16] this costs of approximately 2^{108} operations. Following preprocessing, each attempt at forgery has a probability of success of roughly $3^{-r_1} \approx 2^{-101}$. The two costs could be tuned slightly better by a more careful analysis, however the cost of forgery by the above method can be expected to lie near 2^{108} operations. Since key recovery against the $N = 443$ parameter set only costs 2^{88} operations, the attacker's best strategy is to simply attack the private key directly.

This trend continues for the other parameter sets in Table 4.3; key recovery is consistently easier than forgery.

Appendix A

APPENDICES

A.1 Software

Concurrent with the analyses presented above we have developed reference implementations of NTRUEncrypt and NTRUMLS, as well as automated parameter generation tools for both. These are available at https://github.com/NTRUOpenSourceProject.

Implementations instrumented for benchmarking may be found at the same URL. Benchmarking results may be found at http://bench.cr.yp.to/.

A.2 NTRUEncrypt Challenges

In early 2015 a set of 27 challenges were posted to the Security Innovation website with cash prizes ranging from $1000 - 5000$ USD. Each provides a basic set of parameters $\{N, q, d_1, d_2, d_3, d_g\}$ and a single NTRUEncrypt public key. The challenge is to find an element in the corresponding module that is shorter than the Gaussian heuristic length $\sqrt{N \cdot q/(\pi \cdot \epsilon)}$.

Each of the parameter sets was generated according to the method of Section 3.5. As of writing, the first 7 challenges, in dimensions ranging from $N = 107$ to $N = 173$ have been solved.

A.3 N suitable for use when q is a power of two and $p = 3$

101,	107,	131,	139,	149,	163,	173,	179,	181,	197,
211,	227,	269,	293,	317,	347,	349,	373,	379,	389,
419,	421,	443,	461,	467,	491,	509,	523,	541,	547,
557,	563,	587,	613,	619,	653,	659,	661,	677,	701,
709,	757,	773,	787,	797,	821,	827,	829,	853,	859,
877,	883,	907,	941,	947,	1019,	1061,	1091,	1109,	1117,
1123,	1171,	1187,	1213,	1229,	1237,	1259,	1277,	1283,	1291,
1301,	1307,	1373,	1381,	1427,	1451,	1453,	1483,	1493,	1499,
1523,	1531,	1549,	1571,	1619,	1621,	1637,	1667,	1669,	1693,
1733,	1741,	1747,	1787,	1861,	1867,	1877,	1901,	1907,	1931.

Table A.1: First 100 primes > 100 for which $\mathrm{ord}_{(\mathbb{Z}/N\mathbb{Z})^*}(2) = (N-1)$, i.e. (2) is inert

[h!]	103,	137,	167,	191,	193,	199,	239,	263,	271,	311,
	313,	359,	367,	383,	401,	409,	449,	463,	479,	487,
	503,	521,	569,	599,	607,	647,	719,	743,	751,	761,
	769,	809,	823,	839,	857,	863,	887,	929,	967,	977,
	983,	991,	1009,	1031,	1039,	1063,	1087,	1129,	1151,	1223,
	1231,	1279,	1297,	1303,	1319,	1361,	1367,	1409,	1439,	1447,
	1487,	1489,	1511,	1543,	1559,	1567,	1583,	1607,	1663,	1697,
	1759,	1783,	1823,	1847,	1871,	1873,	1879,	1951,	1993,	2039.

Table A.2: First 80 primes > 100 for which $\mathrm{ord}_{(\mathbb{Z}/N\mathbb{Z})^*}(2) = (N-1)/2$, i.e. (2) is a product of two prime ideals

References

[1] NSA Suite B Cryptography - NSA/CSS. https://www.nsa.gov/ia/programs/suiteb_cryptography/.

[2] Implementation aspects of NTRUEncrypt and NTRUSign v. 2.0. *Efficient Embedded Security Standards (EESS) #1*, June 2003.

[3] IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices. *IEEE Std 1363.1-2008*, pages C1–69, March 2009.

[4] Leonard M. Adleman. On Breaking Generalized Knapsack Public Key Cryptosystems. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, pages 402–412, New York, NY, USA, 1983. ACM.

[5] 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, pages 99–108, New York, NY, USA, 1996. ACM.

[6] Miklós Ajtai and Cynthia Dwork. A Public-key Cryptosystem with Worst-case/Average-case Equivalence. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 284–293, New York, NY, USA, 1997. ACM.

[7] Daniel Augot, Lejla Batina, Daniel J. Bernstein, Joppe Bos, Johannes Buchmann, Wouter Castryck, Orr Dunkelman, Tim Güneysu, Shay Gueron, Andreas Hülsing, Tanja Lange, Mohamed Saied Emam Mohamed, Christian Rechberger, Peter Schwabe, Nicolas Sendrier, Frederik Vercauteren, and Bo-Yin Yang. Initial recommendations of long-term secure post-quantum systems. http://pqcrypto.eu.org/docs/initial-recommendations.pdf, September 2015. Revision 1.

[8] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, March 1986.

[9] William D. Banks and Igor E. Shparlinski. A Variant of NTRU with Non-invertible Polynomials. In Alfred Menezes and Palash Sarkar, editors, *Progress in Cryptology – INDOCRYPT 2002*, number 2551 in Lecture Notes in Computer Science, pages 62–70. Springer Berlin Heidelberg, December 2002. DOI: 10.1007/3-540-36231-2_6.

[10] Daniel J. Bernstein. Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete? In *Workshop Record of SHARCS09: Special-purpose Hardware for Attacking Cryptographic Systems*. 2009.

[11] Daniel J. Bernstein. A subfield-logarithm attack against ideal lattices. http://blog.cr.yp.to/20140213-ideal.html, February 2014.

[12] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors. *Post-Quantum Cryptography*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[13] Joppe W Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. Technical report, IACR Cryptology ePrint Archive, 2014. http://eprint. iacr. org/2014/599. 3, 16, 2014.

[14] Vincent Boyle, Lily Chen, and Adrian Stanger. NIST and NSA Future Plans for Quantum Resistant Cryptography, October 2015.

[15] Gilles Brassard, Peter Hoyer, and Alain Tapp. Quantum Algorithm for the Collision Problem. *arXiv:quant-ph/9705002*, May 1997. arXiv: quant-ph/9705002.

[16] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better Lattice Security Estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, number 7073 in Lecture Notes in Computer Science, pages 1–20. Springer Berlin Heidelberg, December 2011. DOI: 10.1007/978-3-642-25385-0_1.

[17] Henri Cohen. *A Course in Computational Algebraic Number Theory*. Number 138 in Graduate Texts in Mathematics. Springer Berlin Heidelberg, 1993. DOI: 10.1007/978-3-662-02945-9_2.

[18] J. H. Conway and N. J. A. Sloane. A lattice without a basis of minimal vectors. *Mathematika*, 42(01):175–177, June 1995.

[19] Don Coppersmith. Finding a Small Root of a Univariate Modular Equation. In Ueli Maurer, editor, *Advances in Cryptology – EUROCRYPT 1996*, number 1070 in Lecture Notes in Computer Science, pages 155–165. Springer Berlin Heidelberg, May 1996. DOI: 10.1007/3-540-68339-9_14.

[20] Don Coppersmith and Adi Shamir. Lattice Attacks on NTRU. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT 1997*, number 1233 in Lecture Notes in Computer Science, pages 52–61. Springer Berlin Heidelberg, May 1997. DOI: 10.1007/3-540-69053-0_5.

[21] Léo Ducas and Phong Q. Nguyen. Learning a Zonotope and More: Cryptanalysis of NTRUSign Countermeasures. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, number 7658 in Lecture Notes in Computer Science, pages 433–450. Springer Berlin Heidelberg, January 2012.

[22] Paul Dupuis, Jim (Xiao) Zhang, and Philip Whiting. Refined Large Deviation Asymptotics for the Classical Occupancy Problem. *Methodology and Computing in Applied Probability*, 8(4):467–496, December 2006.

[23] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44(170):463–471, 1985.

[24] Scott Fluhrer. Quantum Cryptanalysis of NTRU. Cryptology ePrint Archive, Report 2015/676, 2015. http://eprint.iacr.org/2015/676.

[25] M. Furst and R. Kannan. Succinct Certificates for Almost All Subset Sum Problems. *SIAM Journal on Computing*, 18(3):550–558, June 1989.

[26] Nicolas Gama and Phong Q. Nguyen. Predicting Lattice Reduction. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, number 4965 in Lecture Notes in Computer Science, pages 31–51. Springer Berlin Heidelberg, April 2008. DOI: 10.1007/978-3-540-78967-3_3.

[27] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In *Proceedings of the 29th Annual international conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'10, pages 257–278, Berlin, Heidelberg, 2010. Springer-Verlag.

[28] Craig Gentry. Key Recovery and Message Attacks on NTRU-Composite. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, number 2045 in Lecture Notes in Computer Science, pages 182–194. Springer Berlin Heidelberg, 2001.

[29] 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, pages 197–206, New York, NY, USA, 2008. ACM.

[30] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski Jr, editor, *Advances in Cryptology – CRYPTO 1997*, number 1294 in Lecture Notes in Computer Science, pages 112–131. Springer Berlin Heidelberg, August 1997. DOI: 10.1007/BFb0052231.

[31] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton Kaliski, editor, *Advances in Cryptology – CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer Berlin / Heidelberg, 1997.

[32] Guillaume Hanrot, Xavier Pujol, and Damien Stehl. Analyzing Blockwise Lattice Algorithms Using Dynamical Systems. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, number 6841 in Lecture Notes in Computer Science, pages 447–464. Springer Berlin Heidelberg, August 2011. DOI: 10.1007/978-3-642-22792-9_25.

[33] Philip S. Hirschhorn, Jeffrey Hoffstein, Nick Howgrave-Graham, and William Whyte. Choosing NTRUEncrypt Parameters in Light of Combined Lattice Reduction and MITM Approaches. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security*, number 5536 in Lecture Notes in Computer Science, pages 437–455. Springer Berlin Heidelberg, 2009.

[34] J Hoffstein, J Pipher, and J. H. Silverman. NTRU: A new high speed public key cryptosystem. Technical report, presented at the rump session of CRYPTO '96, Aug 1996.

[35] Jeff Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, and William Whyte. Practical Signatures from the Partial Fourier Recovery Problem. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *Applied Cryptography and Network Security*, number 8479 in Lecture Notes in Computer Science, pages 476–493. Springer International Publishing, June 2014.

[36] Jeff Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, and William Whyte. Transcript Secure Signatures Based on Modular Lattices. In Michele Mosca, editor, *Post-Quantum Cryptography*, number 8772 in Lecture Notes in Computer Science, pages 142–159. Springer International Publishing, October 2014.

[37] Jeff Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing Parameters for NTRUEncrypt. Technical Report 708, 2015.

[38] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSign: Digital Signatures Using the NTRU Lattice. In Marc Joye, editor, *Topics in Cryptology – CT-RSA 2003*, volume 2612, pages 122–140. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[39] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic Number Theory*, number 1423 in Lecture Notes in Computer Science, pages 267–288. Springer Berlin Heidelberg, June 1998. DOI: 10.1007/BFb0054868.

[40] Jeffrey Hoffstein and Joseph H Silverman. Protecting NTRU Against Chosen Ciphertext and Reaction Attacks. Technical Report #016, NTRU Cryptosystems, 2000. Version 1.

[41] Jeffrey Hoffstein and Joseph H. Silverman. Random small Hamming weight products with applications to cryptography. *Discrete Applied Mathematics*, 130(1):37–49, August 2003.

[42] Nick Howgrave-Graham. A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, number 4622 in Lecture Notes in Computer Science, pages 150–169. Springer Berlin Heidelberg, August 2007. DOI: 10.1007/978-3-540-74143-5_9.

[43] Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. The Impact of Decryption Failures on the Security of NTRU Encryption. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, number 2729 in Lecture Notes in Computer Science, pages 226–246. Springer Berlin Heidelberg, August 2003. DOI: 10.1007/978-3-540-45146-4_14.

[44] Nick Howgrave-Graham, Joseph H. Silverman, and William Whyte. Choosing Parameter Sets for NTRUEncrypt with NAEP and SVES-3. Technical Report 045, 2005.

[45] David Jao and Luca De Feo. Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, number 7071 in Lecture Notes in Computer Science, pages 19–34. Springer Berlin Heidelberg, November 2011. DOI: 10.1007/978-3-642-25405-5_2.

[46] Éliane Jaulmes and Antoine Joux. A Chosen-Ciphertext Attack against NTRU. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, number 1880 in Lecture Notes in Computer Science, pages 20–35. Springer Berlin Heidelberg, August 2000. DOI: 10.1007/3-540-44598-6_2.

[47] Ravi Kannan. Improved Algorithms for Integer Programming and Related Lattice Problems. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, pages 193–206, New York, NY, USA, 1983. ACM.

[48] Philip Klein. Finding the Closest Lattice Vector when It's Unusually Close. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, pages 937–941, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.

[49] D. H. Lehmer. A photo-electric number sieve. *The American Mathematical Monthly*, 40(7):401–406, 1933.

[50] Derrick N. Lehmer. Hunting big game in the theory of numbers. *Scripta Mathematica*, 1932.

[51] Emma Lehmer. On the magnitude of the coefficients of the cyclotomic polynomial. *Bulletin of the American Mathematical Society*, 42(6):389–392, June 1936.

[52] A. K. Lenstra, H. W. Lenstra Jr, and L. Lovsz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, December 1982.

[53] Richard Lindner and Chris Peikert. Better Key Sizes (and Attacks) for LWE-Based Encryption. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, number 6558 in Lecture Notes in Computer Science, pages 319–339. Springer Berlin Heidelberg, February 2011. DOI: 10.1007/978-3-642-19074-2_21.

[54] Mingjie Liu and Phong Q. Nguyen. Solving BDD by Enumeration: An Update. In Ed Dawson, editor, *Topics in Cryptology – CT-RSA 2013*, number 7779 in Lecture Notes in Computer Science, pages 293–309. Springer Berlin Heidelberg, February 2013.

[55] Christoph Ludwig. A Faster Lattice Reduction Method Using Quantum Search. In Toshihide Ibaraki, Naoki Katoh, and Hirotaka Ono, editors, *Algorithms and Computation*, number 2906 in Lecture Notes in Computer Science, pages 199–208. Springer Berlin Heidelberg, December 2003.

[56] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *Advances in Cryptology – ASIACRYPT 2009*, pages 598–616. Springer, 2009.

[57] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, number

7237 in Lecture Notes in Computer Science, pages 738–755. Springer Berlin Heidelberg, January 2012.

[58] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A Toolkit for Ring-LWE Cryptography. Technical Report 293, 2013.

[59] Daniele Micciancio and Michael Walter. Practical, predictable lattice basis reduction. Cryptology ePrint Archive, Report 2015/1123, 2015. http://eprint.iacr.org/.

[60] Michele Mosca. Setting the scene for the etsi quantum-safe cryptography workshop. In *e-proceedings of the 1st Quantum-Safe-Crypto Workshop*. ETSI, September 2013.

[61] Phong Nguyen and Oded Regev. Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 271–288. Springer Berlin / Heidelberg, 2006.

[62] Century of Progress International Exposition. *Official Guide Book of the Fair: 1933*.

[63] John Proos. Imperfect Decryption and an Attack on the NTRU Encryption Scheme. Cryptology ePrint Archive, Report 2003/002, 2003. http://eprint.iacr.org/.

[64] John Schanck, William Whyte, and Zhenfei Zhang. A quantum-safe circuit-extension handshake for tor. Cryptology ePrint Archive, Report 2015/287, 2015. http://eprint.iacr.org/.

[65] C. P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(23):201–224, 1987.

[66] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66(1-3):181–199, August 1994.

[67] C. P. Schnorr and H. H. Hörner. Attacking the Chor-Rivest Cryptosystem by Improved Lattice Reduction. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology – EUROCRYPT 1995*, number 921 in Lecture Notes in Computer Science, pages 1–12. Springer Berlin Heidelberg, May 1995. DOI: 10.1007/3-540-49264-X_1.

[68] Claus Peter Schnorr. Lattice Reduction by Random Sampling and Birthday Methods. In Helmut Alt and Michel Habib, editors, *STACS 2003*, number 2607 in Lecture Notes

in Computer Science, pages 145–156. Springer Berlin Heidelberg, February 2003. DOI: 10.1007/3-540-36494-3_14.

[69] Ernst S. Selmer. On the irreducibility of certain trinomials. *Mathematica Scandinavica*, 4(0):287–302, December 1956.

[70] Jean Pierre Serre. *Topics in Galois theory*. A K PETERS Limited (MA), 2008.

[71] Joseph H Silverman. A Meet-In-The-Middle Attack on an NTRU Private Key. Technical Report #004, NTRU Cryptosystems, 1997. Version 1.

[72] Joseph H Silverman. Wraps, Gaps, and Lattice Constants. Technical Report #011, NTRU Cryptosystems, 2001. Version 1.

[73] Joseph H. Silverman and William Whyte. Timing Attacks on NTRUEncrypt Via Variation in the Number of Hash Calls. In Masayuki Abe, editor, *Topics in Cryptology – CT-RSA 2007*, number 4377 in Lecture Notes in Computer Science, pages 208–224. Springer Berlin Heidelberg, February 2007.

[74] Ari Singer. NTRU Cipher Suites for TLS. Internet Draft draft-ietf-tls-ntru-00, IETF Secretariat, 2001.

[75] Damien Stehl and Ron Steinfeld. Making NTRU as Secure as Worst-Case Problems over Ideal Lattices. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, number 6632 in Lecture Notes in Computer Science, pages 27–47. Springer Berlin Heidelberg, May 2011. DOI: 10.1007/978-3-642-20465-4_4.

[76] Hong Wang, Zhi Ma, and ChuanGui Ma. An efficient quantum meet-in-the-middle attack against NTRU-2005. *Chinese Science Bulletin*, 58(28-29):3514–3518, October 2013.

[77] Xiang Wang, WanSu Bao, and XiangQun Fu. A quantum algorithm for searching a target solution of fixed weight. *Chinese Science Bulletin*, 56(6):484–489, February 2011.

[78] Lawrence C. Washington. *Introduction to Cyclotomic Fields*, volume 83 of *Graduate Texts in Mathematics*. Springer New York, New York, NY, 1997.

[79] Zhijian Xiong, Jinshuang Wang, Yanbo Wang, Tao Zhang, and Liang Chen. An Improved MITM Attack Against NTRU. *International Journal of Security and Its Applications*, 6(2):269–274.