Sealing the Leak on Classical NTRU signatures

Conference Paper · October 2014

DOI: 10.1007/978-3-319-11659-4_1

CITATIONS
18

READS
114

4 authors, including:

Jean-Christophe Deneuville
Ecole Nationale de l'Aviation Civile
10 PUBLICATIONS **81** CITATIONS

SEE PROFILE

Philippe Gaborit
University of Limoges
147 PUBLICATIONS **2,930** CITATIONS

SEE PROFILE

Sealing the Leak on classical NTRU signatures

C. Aguilar Melchor[1], X. Boyen[2], J.C. Deneuville[1], P. Gaborit[1]

[1] XLIM-DMI, Université de Limoges, France
[2] QUT, Brisbane, Autralia

Abstract. Initial attempts to obtain lattice based signatures were closely related to reducing a vector modulo the fundamental parallelepiped of a secret basis (like GGH [9], or `NTRUSign` [12]). This approach leaked some information on the secret, namely the shape of the parallelepiped, which has been exploited on practical attacks [24]. `NTRUSign` was an extremely efficient scheme, and thus there has been a noticeable interest on developing countermeasures to the attacks, but with little success [6].
In [8] Gentry, Peikert and Vaikuntanathan proposed a randomized version of Babai's nearest plane algorithm such that the distribution of a reduced vector modulo a secret parallelepiped only depended on the size of the base used. Using this algorithm and generating large, close to uniform, public keys they managed to get provably secure GGH-like lattice-based signatures. Recently, Stehlé and Steinfeld obtained a provably secure scheme very close to `NTRUSign` [26] (from a theoretical point of view).
In this paper we present an alternative approach to seal the leak of `NTRUSign`. Instead of modifying the lattices and algorithms used, we do a classic leaky `NTRUSign` signature and hide it with gaussian noise using techniques present in Lyubashevky's signatures. Our main contributions are thus a set of strong `NTRUSign` parameters, obtained by taking into account latest known attacks against the scheme, a statistical way to hide the leaky `NTRU` signature so that this particular instantiation of CVP-based signature scheme becomes zero-knowledge and secure against forgeries, based on the worst-case hardness of the $\tilde{\mathcal{O}}(N^{1.5})$-Shortest Independent Vector Problem over `NTRU` lattices. Finally, we give a set of concrete parameters to gauge the efficiency of the obtained signature scheme.

Keywords: Lattice-based Cryptography, Digital Signatures, NTRUSign, Provable Security, SIS

1 Introduction

Lattice based cryptography has met growing interest since the seminal work of Ajtai [1] which introduced the so called worst-case to average-case reductions. Based upon this work, a long list of cryptographic primitives such as One Way Functions, Collision-Resistant Hash Functions, Digital Signatures, or Identification schemes have been revisited to provide more confidence about security. The most efficient known digital signature scheme provably secure is BLISS [5] [1] which leads to signatures of about 5kb [2] for a security level of 128 bits.

Digital signatures have shown great promise since 1997, when was introduced GGH [9]. The most famous particular instantiation of GGH is `NTRUSign`, which uses convolution modular lattices. The particularity of those schemes is their lack of strong worst-case to average-case security reductions, but they offer amazing performances regarding classical schemes based on number theory or discrete logarithm. For instance, for a 128 bit security level, a `NTRUSign` signature would be only 1784 bits long (see [11]).

`NTRUSign`, first known as `NSS` [13], was first introduced at EuroCrypt'01 by Hoffstein, Pipher and Silverman. It was amazingly fast and benefited from small keys due to the cyclic structure of the underlying convolution modular lattices that were used. The authors were aware that their scheme was vulnerable to transcript attacks *i.e.* wasn't zero-knowledge, but unfortunately they overestimated its length, and Nguyen and Regev succeeded in breaking the scheme in 2006 [24] by a nice gradient descent over the even moment polynomials. Initially, their attack required about 100.000 `NTRU` signatures to recover the hidden parallelepiped that reveals the secret basis, but still due to the cyclic structure of convolution modular lattices, they were able to shrink this threshold to about only 400 signatures for a claimed security level of 80 bits. In order to tackle this issue, several heuristical countermeasures were proposed such as the use of perturbations [12] and the deformation of the fundamental parallelepiped [15], but none of them were capable to resist to the improved attack by Ducas and Nguyen [6].

[1] which improves [19] with a better rejection sampling
[2] for the space-optimized version, see Table 3 of [5] for more details

1.1 Our contribution

We revisit NTRUSign in order to provide it with a zero-knowledge proof. Our technique is inspired from Lyubashevsky's scheme [19], where the secret key $\mathbf{S}$ consists of a matrix in $\{-d, \dots, d\}^{m \times k}$, the message is hashed to a vector $\mathbf{c} \leftarrow \{-1, 0, 1\}^k$ such that $\|\mathbf{c}\|_1 \leq \kappa$, and the signature consists of $\mathbf{Sc}$ shifted by a mask $\mathbf{y} \xleftarrow{\$} D_\sigma^m$ where D_σ^m represents the discrete gaussian distribution in dimension m with standard deviation σ.

Instead of hiding $\mathbf{Sc}$, we get a leaky signature from NTRUSign, and then use this signature as the secret and hide it with a well chosen $\mathbf{y}$. The critical technicality resides in the choice of the dimension N and the standard deviation σ : if it was chosen too small, the secret isn't properly hidden, and our modification doesn't seal any leak, if σ is too big, so will be our signatures and our scheme loses in efficiency and practicality.

We note that unlike other provably secure signature schemes such as GPV [8] or [25], we do not modify the initial NTRU signature scheme, except by chosing public parameters more conservatively, and thus keep its inherent size and computational efficiency. Of course masking the signature in a second step comes at a price, but we manage to get signature of size $\approx 10kb$ together with public and secret keys respectively around 7000 and 1500 kb.

We choose to hide NTRU signatures with a noise based on the assumption that the leak in the signatures is exploitable but that there are no structural attacks against the public NTRU key (and thus we suppose that sealing the leak is enough to secure the scheme). This is based on the observation that the research community has published no noticeable structural attacks on NTRU lattices in the last decade and that problems such as SIS do not seem to be easier than in random lattices (if we take into account the gap induced by the small secret key).

1.2 Organization of the paper

In section 2, we present the basic elements and notations used in NTRUSign and Lyubashevsky's signature scheme, then describe these schemes respectively in sections 3 and 4. Finally, we present the scheme we propose in section 5 along with its security proofs and sets of parameters.

2 Background and Problems

In this section, we introduce basics of lattice-based cryptography. Nevertheless, due to space restriction, some of them will be omitted and we refer the reader to [23] for further details and proofs.

2.1 Notation

Sets. Throughout this paper, $\mathbb{Z}$ will denote the set of integer numbers, and for $q \in \mathbb{Z}$, $\mathbb{Z}_q$ will denote the set of integers taken modulo q, in the set $\left[-\frac{q}{2}; \frac{q}{2}\right)$. We will make heavy use of the notation $\mathcal{R}_q$ to represent $\mathbb{Z}_q[X]/(X^N - 1)$, the ring of polynomials of degree less than N, modulo q and $X^N - 1$. Vectors and/or polynomials will be represented with bold-face letters, for any $\mathbf{x} \in \mathcal{R}_q$, we will use either its polynomial notation $\mathbf{x} = \sum_{i=0}^{N-1} x_i \cdot X^i$ or its vector representation $\mathbf{x} = (x_0, x_1, \dots, x_{N-1})^t$. Matrices such as the public key will be represented with bold-face capital letters $\mathbf{A} \in \mathbb{Z}_q^{N \times 2N}$.

In section 3, the NTRUSign secret polynomials $\mathbf{f}, \mathbf{g}$ will be sampled from a particular subset $\mathcal{T}(d)$ of $\mathcal{R}_q$, which consists of polynomials $\mathbf{f}$ of degree strictly less that N, with exactly $d+1$ coefficients equal to 1, d equal to -1 and the $N - 2d - 1$ others equal to 0. All logarithms will be based 2 unless explicitly mentioned.

Norms. For any $\mathbf{s}, \mathbf{t} \in \mathcal{R}_q$, we will make use of several norms :

- The centered norm : $\|\mathbf{s}\|_c^2 = \sum_{i=0}^{N-1} s_i^2 - \frac{1}{N}\left(\sum_{i=0}^{N-1} s_i\right)^2 = N \cdot \text{Variance}(\mathbf{s})$
- The balanced norm : $\|(\mathbf{s}, \mathbf{t})\|_\nu^2 = \|\mathbf{s}\|_c^2 + \nu^2 \cdot \|\mathbf{t}\|_c^2$

– The euclidian norm : $\|\mathbf{s}\|_2^2 = \sum_{i=0}^{N-1} s_i^2$, or just $\|\mathbf{s}\|^2$ for simplicity as this norm is the most standard in lattice-based cryptography

The first and second norms are somehow typical to `NTRU`, and don't give many intuition of the actual length of a vector, in the common (euclidian) sense. Therefore, we describe a method to experimentally translate a balanced norm into a euclidian one with arbitrary desired probability. As $\|\mathbf{s}\|_c = \sigma_\mathbf{s}\sqrt{N}$ where $\sigma_\mathbf{s}$ is the standard deviation of the s_is, so that each s_i is approximately $\sigma_\mathbf{s}$, and by lemma 2.3.2 we have

$$\|\mathbf{s}\|_2 \leq \alpha \cdot \sigma_\mathbf{s}\sqrt{N} \text{ with probability } 1 - 2^{-k}$$

where k is the security parameter and the corresponding α can be read in table 1. Even if the s_is are not necessarily distributed according to a gaussian, it is possible to lower $\|\mathbf{s}\|_2$ with the same probability using tighter rejection sampling. We use this fact for size-optimizations in parameters given in table 3.

2.2 Digital Signatures

For completeness, we recall the definition of a Digital Signature scheme.

Definition 2.2.1 (Signature Scheme) *A signature scheme is composed of* 3 *polynomial-time algorithms* (K, S, V) *:*

1. `KeyGen` K *: which given a security parameter* k *as input returns a couple of keys* (pk, sk)
2. `Sign` S *: which given the secret key* sk *and a message* μ *returns a signature* s *of this message*
3. `Verify` V *: which given the public key* pk*, the signature* s *and the message* μ*, ensures that this signature was indeed generated using* sk

such that for any $(pk, sk) \leftarrow K(k)$*,* $Pr[V(pk, \mu, S(sk, \mu)) = 1] = 1$*.*

There are two ways to attack a signature scheme, either try to create a signature from other couples (μ, s) and the public key pk, or recover the secret key sk directly from pk and eventually some signed messages. The former idea leads to the following definition :

Definition 2.2.2 (Forgery) *A signature scheme* (K, S, V) *is said to be secure against forgeries, if for any polynomial-time adversary* $\mathcal{A}$ *who has access to* pk *and couples* $\{(\mu_1, s_1), \ldots, (\mu_n, s_n)\}$*,* $\mathcal{A}$ *only has a negligible probability (depending on the security parameter* k*) to create a couple* $(\mu \neq \mu_i, s')$ *such that* $V(pk, \mu, s') = 1$*, that is to say a valid signature.*

2.3 Discrete Normal Distribution

In this section, we define the Discrete Normal Distribution and describe some of its desirable properties, that fit particularly well with lattices.

Definition 2.3.1 (Continuous Normal Distribution) *The Continuous Normal Distribution over* $\mathbb{R}^{2N}$ *centered at* $\mathbf{v}$ *with standard deviation* σ *is defined by* $\rho_{\mathbf{v},\sigma}^{2N}(\mathbf{x}) = (\frac{1}{\sigma\sqrt{2\pi}})^{2N} \cdot \exp(-\frac{\|\mathbf{x}-\mathbf{v}\|_2^2}{2\sigma^2})$*.*

In order to make the Normal Distribution fitting with lattices and obtain a probability function, we need to scale this distribution by the lattice quantity $\rho_{\mathbf{0},\sigma}^{2N}(\mathbb{Z}^{2N}) = \sum_{\mathbf{x}\in\mathbb{Z}^{2N}} \rho_{\mathbf{0},\sigma}^{2N}(\mathbf{x})$.

Definition 2.3.2 (Discrete Normal Distribution) *The Discrete Normal Distribution over* $\mathbb{Z}^{2N}$ *centered at* $\mathbf{v}$ *with standard deviation* σ *is defined by* $D_{\mathbf{v},\sigma}^{2N}(\mathbf{x}) = \rho_{\mathbf{v},\sigma}^{2N}(\mathbf{x})/\rho_{\mathbf{0},\sigma}^{2N}(\mathbb{Z}^{2N})$*.*

The next lemma gives us an idea of how big the standard deviation must be to ensure that the inner product of two vectors doesn't overflow a certain amount. This lemma is crucial to determine our signature size in table 3 with overwhelming probability.

Lemma 2.3.1 ([19]) $\forall \mathbf{v} \in \mathbb{R}^{2N}, \forall \sigma, r > 0$*, we have* $Pr\left[|\langle \mathbf{x}, \mathbf{v}\rangle| > r; \mathbf{x} \xleftarrow{\$} D_\sigma^{2N}\right] \leq 2e^{-\frac{r^2}{2\|\mathbf{v}\|^2\sigma^2}}$*.*

Optimally, we will set $r = \alpha \cdot \|\mathbf{v}\|\sigma$. Table 1 shows how big α should be to ensure k bits of security. We also need a few more material to prove that our NTRU signature will be correctly hidden by our mask. This material is given by the following lemma.

Lemma 2.3.2 ([19])

1. $\forall \alpha > 0, Pr\left[|x| > \alpha\sigma; x \xleftarrow{\$} D_\sigma^1\right] \leq 2e^{-\frac{\alpha^2}{2}}$
2. $\forall \eta \geq 1, Pr\left[\|\mathbf{x}\| > \eta\sigma\sqrt{2N}; \mathbf{x} \xleftarrow{\$} D_\sigma^{2N}\right] < \eta^{2N} e^{N(1-\eta^2))}$
3. $\forall \mathbf{x} \in \mathbb{Z}^{2N}$ *and* $\sigma \geq \frac{3}{\sqrt{2\pi}}$, *we have* $D_\sigma^{2N}(\mathbf{x}) \leq 2^{-2N}$

Security parameter k	80	100	112	128	160
Gap factor α	11	12	13	14	15

Table 1: $\alpha = \lceil \sqrt{2(k+1)\ln(2)} \rceil$ as a function of the security level k

For our purposes, as the mask $\mathbf{y}$ is sampled from a Discrete Normal Distribution, we might have to re-sample several times before obtaining a valid signature, but still, we want our signature procedure to terminate, in a reasonable (polynomial) time. This is ensured by the next lemma, whose proof is essentially detailed in [19] for a security level of $k = 100$ bits. We extend the proof of this lemma (in appendix A.1) to make it fitting better with different security levels.

Lemma 2.3.3 ([19] extended) *For any* $\mathbf{v} \in \mathbb{Z}^{2N}$ *and* $\sigma = \omega(\|\mathbf{v}\|_2\sqrt{\log_2(2N)})$, *we have :*

$$Pr\left[D_\sigma^{2N}(\mathbf{x})/D_{\mathbf{v},\sigma}^{2N}(\mathbf{x}) = \mathcal{O}(1); \mathbf{x} \xleftarrow{\$} D_\sigma^{2N}\right] = 1 - 2^{-\omega(\log_2(2N))}$$

and more concretely, $\forall \mathbf{v} \in \mathbb{Z}^{2N}$, *if* $\sigma = \alpha\|\mathbf{v}\|$ *for some positive* α, *then*

$$Pr\left[D_\sigma^{2N}(\mathbf{x})/D_{\mathbf{v},\sigma}^{2N}(\mathbf{x}) < e^{1+1/(2\alpha^2)}; \mathbf{x} \xleftarrow{\$} D_\sigma^{2N}\right] > 1 - 2^{-k}$$

This lemma ensures us that Lyubashevsky's layer of the signing procedure will be called at most $M = e^{1+1/(2\alpha^2)}$ times with probability at least $1 - 2^{-k}$. Keeping this repetition rate down is of major importance especially as this layer involves a NTRUSign procedure which is itself also a loop. In table 3, we provide two versions of parameters for each security level. In the first one, the NTRUSign part is generated in only one round with overwhelming probability, before applying the rejection step with $M \approx 2.8$, leading to a speed-optimized version. In the second one, we allow the generation of the NTRU signature to take at most 5 rounds whilst reducing its norm. This implies more rejection steps ($M \approx 7.5$) but allows us to shrink the signature sizes by approximately 15%.

To prove the security of our scheme, we also need the following rejection sampling lemma, which will be used in the proof of theorem 2.3.5 that will help us getting our security reduction to SIS.

Lemma 2.3.4 ([19]) *For any set* V, *and probability distributions* $h : V \to \mathbb{R}$ *and* $f : \mathbb{Z}^{2N} \to \mathbb{R}$, *if* $g_{\mathbf{v}} : \mathbb{Z}^{2N} \to \mathbb{R}$ *is a family of probability distributions indexed by* $v \in V$ *such that* $\exists M \in \mathbb{R}$ / $\forall v \in V, Pr[Mg_v(\mathbf{z}) \geq f(\mathbf{z}); \mathbf{z} \xleftarrow{\$} f] \geq 1 - \epsilon$ *then the outputs of algorithms* $\mathcal{A}$ *and* $\mathcal{F}$

Algorithm $\mathcal{A}$
1: $v \xleftarrow{\$} h$
2: $\mathbf{z} \xleftarrow{\$} g_v$
3: *output* $(\mathbf{z}, v)$ *with probability* $\min\left(\frac{f(\mathbf{z})}{Mg_v(\mathbf{z})}, 1\right)$

Algorithm $\mathcal{F}$
1: $v \xleftarrow{\$} h$
2: $\mathbf{z} \xleftarrow{\$} f$
3: *output* $(\mathbf{z}, v)$ *with probability* $1/M$

are within statistical distance ϵ/M.

The next theorem is a direct consequence of lemmas 2.3.3 and 2.3.4 by replacing V by the subset of $\mathbb{Z}^{2N}$ of vector $\mathbf{v}$ of length at most T, f by D_{σ}^{2N} and g_v by $D_{\mathbf{v},\sigma}^{2N}$.

Theorem 2.3.5 ([19]) *Let $V = \{\mathbf{v} \in \mathbb{Z}^{2N}; \|\mathbf{v}\| \leq T\}$, $\sigma = \omega(T\sqrt{\log 2N}) \in \mathbb{R}$ and $h : V \to \mathbb{R}$ a probability distribution. Then $\exists M = \mathcal{O}(1)$ such that distributions of algorithms $\mathcal{A}$ and $\mathcal{F}$ below are within statistical distance $\frac{2^{-\omega(\log 2N)}}{M}$. Moreover, $\mathcal{A}$ outputs something with probability at least $\frac{1-2^{-\omega(\log 2N)}}{M}$*

Algorithm $\mathcal{A}$	*Algorithm $\mathcal{F}$*
1: $\mathbf{v} \xleftarrow{\$} h$	*1:* $\mathbf{v} \xleftarrow{\$} h$
2: $\mathbf{z} \xleftarrow{\$} g_{\mathbf{v}}$	*2:* $\mathbf{z} \xleftarrow{\$} f$
3: output $(\mathbf{z}, \mathbf{v})$ *with probability* $\min\left(\frac{f(\mathbf{z})}{Mg_{\mathbf{v}(\mathbf{z})}}, 1\right)$	*3: output* $(\mathbf{z}, \mathbf{v})$ *with probability* $1/M$

2.4 Average-Case SIS Problems

Problems. The last part of this background section describes the main average-case lattice problem we will base our signature scheme upon, namely the Short Integer Solution (SIS) Problem, which is a least as hard as the worst-case of Shortest Independent Vector Problem (SIVP) [1] up to a polynomial approximation factor.

Definition 2.4.1 (ℓ_2-SIS$_{q,N,2N,\beta}$ problem) *For any $\mathbf{A} \in \mathbb{Z}_q^{N\times 2N}$, the ℓ_2-SIS$_{q,N,2N,\beta}$ problem consists in finding a vector $\mathbf{v} \in \mathbb{Z}_q^{2N} \setminus \{0\}$ such that $\mathbf{Av} = \mathbf{0}$ and $\|\mathbf{v}\|_2 \leq \beta$.*

Relations between parameters q, N and β will be discussed later in this section as they condition the length of the shortest expected vector, but we can already mention that for a ℓ_2-SIS$_{q,N,2N,\beta}$ solution to exist, we need to set $\beta \geq \sqrt{2Nq}$.

Definition 2.4.2 (SIS$_{q,N,2N,d}$ distribution) *Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{N\times 2N}$, and a random $\mathbf{v} \in \mathbb{Z}_q^{2N}$, output $(\mathbf{A}, \mathbf{Av} \mod q)$.*

Definition 2.4.3 (Search SIS$_{q,N,2N,d}$) *Given $(\mathbf{A}, \mathbf{t}) \in \mathbb{Z}_q^{N\times 2N} \times \mathbb{Z}_q^N$, find $\mathbf{v} \in \{-d, \ldots, 0, \ldots, d\}^{2N}$ such that $\mathbf{Av} = \mathbf{t}$.*

Definition 2.4.4 (Decisional SIS$_{q,N,2N,d}$) *Given $(\mathbf{A}, \mathbf{t}) \in \mathbb{Z}_q^{N\times 2N} \times \mathbb{Z}_q^N$, decide whether it comes from the $SIS_{q,N,2N,d}$ distribution or the uniform distribution over $\mathbb{Z}_q^{N\times 2N} \times \mathbb{Z}_q^N$ with non-negligible advantage.*

Relations between these problems. We now recall existing relations betweens the problems described above, together with relationships between their parameters which somehow strengthen or weaken these problems. Fisrt, it is rather intuitive that the smaller d, the harder the problem, but this remark doesn't take the modulus q into account. We can see the matrix multiplication by $\mathbf{A} \in \mathbb{Z}_q^{N\times 2N}$ as a linear map whose domain is $\mathbb{Z}_q^{2N}$ (of size q^{2N}) and range is $\mathbb{Z}_q^N$ (of size q^N). So by constraining the domain to $\mathbb{Z}_d^{2N}$, we need d to be of order $\sqrt{q}$ for domain and range to be in one-to-one correspondence (even if it is not a sufficient condition). As a consequence, when $d \ll \sqrt{q}$ there will be only one $\mathbf{v} \in \{-d, \ldots, 0, \ldots, d\}^{2N}$ satisfying $\mathbf{Av} = \mathbf{t}$ with high probability, which makes it easier to distinguish between the SIS$_{q,N,2N,d}$ distribution and the uniform one. On the other hand, increasing d far beyond $\sqrt{q}$ leaves room for multiple solutions to the Search SIS$_{q,N,2N,d}$ Problem with high probability. Therefore, we can reasonably expect the hardest SIS$_{q,N,2N,d}$ instances to rely where $d \approx \sqrt{q}$.

Besides relationships between those parameters, there are reductions from some of these problems to others. For instance, as it is often the case between search and decisional problems, one can build a distinguisher from an oracle solving Decisional SIS$_{q,N,2N,d}$ to solve the search version, and that is what the following theorem states :

Theorem 2.4.6 ([16, 21]) *For any $d \in \mathcal{O}(N)$, there is a polynomial-time reduction from solving Search $SIS_{q,N,2N,d}$ to solving Decisional $SIS_{q,N,2N,d}$.*

Actually, the best (known) way to solve the search version of SIS appears to be solving the decisional version. However, the next lemma gives us confidence about the hardness of the decisional SIS problem when the solution is allowed to be larger and larger, which translates the fact that the SIS distribution comes closer and closer to the uniform distribution.

Lemma 2.4.7 ([19]) *For any $\alpha \geq 0$ such that $\gcd(2\alpha+1, q) = 1$, there is a polynomial-time reduction from solving Decisional $SIS_{q,N,2N,d}$ to solving Decisional $SIS_{q,N,2N,(2\alpha+1)d+\alpha}$.*

As mentioned in [23], when the dimension equals twice the rank ($m = 2N$), and above all if β is small enough, the actual best known way to solve Decisional $\text{SIS}_{q,N,2N,d}$ is to solve the ℓ_2-$\text{SIS}_{q,N,2N,\beta}$ problem.

Lemma 2.4.8 ([23]) *If $4d\beta \leq q$, there is a polynomial-time reduction from solving Decisional $SIS_{q,N,2N,d}$ to solving ℓ_2-$SIS_{q,N,2N,\beta}$.*

As a consequence, it has been shown in [22, 19] that for ℓ_2-$\text{SIS}_{q,N,2N,\beta}$ to be hard, one has to ensure that the following inequality is satisfied for any desired security level k :

$$2\beta\sqrt{N \cdot \frac{d(d+1)}{3}} > \frac{q}{\pi}\sqrt{k \cdot \ln(2)} \tag{1}$$

This lemma already gives us a first restriction for setting the parameters. Indeed, by rewriting the above inequality, we have $4 \cdot \left(\frac{d\beta}{q}\right)^2 \cdot \frac{N(d+1)\pi^2}{3\ln(2)} > k$, and as $\frac{4\pi^2}{3\ln(2)} \approx 4^2$ and $\frac{4d\beta}{q} \leq 1$, this condition means that $N \cdot (d+1)$ is greater than k by some multiplicative gap. We now discuss about another kind of restriction due to the expected length of "the" shortest vector in a given convolution modular lattice (*i.e* Gaussian Heuristic) relatively to lattice basis reduction techniques.

Since its introduction in 1982 by Lenstra, Lenstra and Lovász [17] with the LLL algorithm, lattice reduction has known great applications and generalizations. Among all those techniques lives a perpetual trade-off between the running time of the reduction algorithm and the quality of the (eventual) output, which is gauged by the Hermit Factor δ. This factor plays a crucial role in the hardness of the ℓ_2-$\text{SIS}_{q,N,2N,\beta}$ problem in the sense that lattice reduction algorithms can find vectors $\mathbf{v} \in \mathbb{Z}_q^{2N}$ such that $\mathbf{Av} = \mathbf{0}$ and $\|\mathbf{v}\|_2 \leq \delta^{2N}\sqrt{q}$ [7]. Even if $\delta \approx 1.007$ seems to be a lower bound for reasonable future [4], deepest explorations on this factor have been made in [18], and more precise approximations have been extrapolated for different security levels. Parameter δ in table 4 has been set sticking to these extrapolations.

Further analysis led Micciancio and Regev [23] to the conclusion that the SIS problem *does not* become that harder by increasing the number of columns. Actually, they show that one can find a lattice vector $\mathbf{v}$ such that

$$\|\mathbf{v}\| \approx \min\left(q, 2^{2\sqrt{N \log q \log \delta}}\right) \tag{2}$$

and $\mathbf{Av} = \mathbf{0}$ using only $\sqrt{N \log q / \log \delta}$ of the $2N$ columns of the matrix $\mathbf{A}$. This bound will gives us another restriction when setting our parameters in section 5.

3 General overview of NTRUSign

In this section, we briefly describe the NTRUSign scheme. For a complete description of the scheme, we refer the reader to [10, 11]. The basic set for NTRUSign is $\mathcal{R}_q = \mathbb{Z}_q/(X^N - 1)$ with addition and polynomial multiplication modulo $X^N - 1$, also known as convolution product and denoted by $*$:

$$(\mathbf{f} * \mathbf{g})(X) = \sum_{k=0}^{N-1} \left(\sum_{i+j \equiv k \mod N} f_i g_i \right) X^k \tag{3}$$

The public and private keys will be matrices $\mathbf{P}$, and $\mathbf{S}$ defined by :

$$\mathbf{P} = \left(\begin{array}{cccc|ccc} 1 & 0 & \dots & 0 & h_0 & h_{N-1} & \dots & h_1 \\ 0 & 1 & \dots & 0 & h_1 & h_0 & \dots & h_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & h_{N-1} & h_{N-2} & \dots & h_0 \\ \hline 0 & 0 & 0 & 0 & q & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{array}\right) \quad \mathbf{S} = \left(\begin{array}{ccc|ccc} f_0 & f_{N-1} & \dots f_1 & F_0 & F_{N-1} & \dots F_1 \\ f_1 & f_0 & \dots f_2 & F_0 & F_{N-1} & \dots F_1 \\ \vdots & \vdots & \ddots \vdots & \vdots & \vdots & \ddots \vdots \\ f_{N-1} & f_{N-2} & \dots f_0 & F_{N-1} & F_{N-2} & \dots F_0 \\ \hline g_0 & g_{N-1} & \dots g_1 & G_0 & G_{N-1} & \dots G_1 \\ g_1 & g_0 & \dots g_2 & G_0 & G_{N-1} & \dots G_1 \\ \vdots & \vdots & \ddots \vdots & \vdots & \vdots & \ddots \vdots \\ g_{N-1} & g_{N-2} & \dots g_0 & G_{N-1} & G_{N-2} & \dots G_0 \end{array}\right) \tag{4}$$

where $\mathbf{h} = \mathbf{f}^{-1} * \mathbf{g} \mod q$ for $\mathbf{f}, \mathbf{g} \in \mathcal{R}_q$, $\mathbf{F}, \mathbf{G} \in \mathcal{R}_q$ are given by the *keyGen* algorithm 1, and verify $\mathbf{f} * \mathbf{G} - \mathbf{g} * \mathbf{F} = q$. As operations are performed modulo q and as $(\mathbf{F}, \mathbf{G})$ can be obtained efficiently from $(\mathbf{f}, \mathbf{g})$ using [12], we will denote $\mathbf{P} = (\mathbf{1}\ \mathbf{h})$ and $\mathbf{S} = (\mathbf{f}\ \mathbf{g})$ for short. The NTRU lattice is :

$$\Lambda_q(\mathbf{A}) = \{(\mathbf{y_1}, \mathbf{y_2}) / \mathbf{y_2} = \mathbf{y_1} * \mathbf{h} \mod q\} \tag{5}$$

Algorithms. We now recall the algorithms used in NTRUSign. More sophisticated versions of this signature scheme have been elaborated, such as the one with perturbations [12] or the deformation of the fundamental parallelepiped [15] to counter the attack of [24], but all these upgrades have been broken with the later improved attack of [6]. Therefore, we will further use the basic instantiation of NTRUSign for our scheme, which offers greater performances together with smaller keys. We will discuss the security of our scheme in section 5.

Key generation and signing procedures are described respectively in algorithms 1 and 2. The NTRU signature $\mathbf{s} \in \mathcal{R}_q$ is a simple Babaï's round-off [2] of the target $(\mathbf{0}, \mathbf{m})$ using the secret key sk, where $\mathbf{m} = H(\mu)$ is the hash of the message μ to be signed by $H : \{0,1\}^* \to \mathcal{R}_q$. In order to process this round-off, for any $x \in \mathbb{R}$ we will denote by $\lfloor x \rceil$ the nearest integer to x so that $\{x\} = x - \lfloor x \rceil \in (-\frac{1}{2}, \frac{1}{2}]$. By extension, for any $\mathbf{x} \in \mathcal{R}_q$, $\{\mathbf{x}\}$ will denote the previous operation applied to every x_i. Due to the particular structure of the NTRU lattice and to the fact that the NTRU signature is a lattice vector, giving $\mathbf{s}$ as the signature suffices to reconstruct the right part using the public key. This trick permits to save half of the space needed to represent the NTRU signature.

Polynomials $\mathbf{F}$ and $\mathbf{G}$ in algorithm 1 can be obtained efficiently using the technique described in [12], but as we are using the transpose NTRU lattice, those polynomials are not even used for signing nor verifying the signature. So in the case of a constrained environment, one can just skip this computation. Nevertheless, $\mathbf{F}$ and $\mathbf{G}$ play a role in the size of the error when rounding-off the target. The technique of [12] permits to find those polynomials in such a way that $\|\mathbf{F}\| \approx \|\mathbf{G}\| \approx \sqrt{\frac{N}{12}}\|\mathbf{f}\|$ so that the error when signing using sk is of size approximately $(\sqrt{\frac{N}{6}} + \nu\frac{N}{6\sqrt{2}})\|\mathbf{f}\|$. As a comparison, a invalid signer trying to sign using pk instead of sk would generate an error of magnitude $\nu\sqrt{\frac{N}{12}}q$.

k	N	d	q	ν	$\mathcal{N}$	ω_{cmb}	c	ω_{lk}	ω_{frg}	γ	ω_{lf}	R	L
100	431	34	2^{10}	0.16686	141	167	3.714	187	172	0.0516	415	131	180
112	479	42	2^{10}	0.15828	165	200	4.232	209	137	0.0558	470	157	200
128	541	61	2^{11}	0.14894	211	269	3.711	239	329	0.0460	541	207	226
160	617	57	2^{11}	0.13946	217	269	3.709	272	360	0.0431	627	210	258

Table 2: New NTRUSign parameters, $\rho = 1$, no perturbation

Parameters. Setting concrete NTRUSign parameters for a given security level seems to be an unclear task to perform. Nevertheless, the authors of [10] provide a generic algorithm to generate such parameters,

Algorithm 1: KeyGen(N, q, d, $\mathcal{N}$, ν)

Input: N, q, d, $\mathcal{N}$, and ν
Output: $pk = \mathbf{h} = \mathbf{f}^{-1} * \mathbf{g} \mod q$ and $sk = \mathbf{f}, \mathbf{g}$
begin
  repeat
    $\mathbf{f} \xleftarrow{\$} \mathcal{T}(d)$, $\mathbf{g} \xleftarrow{\$} \mathcal{T}(d)$;
  until $\mathbf{f}$ *is invertible in* $\mathcal{R}_q$;
  $\mathbf{h} = \mathbf{g} * \mathbf{f}^{-1}$;
  return $pk = \begin{pmatrix} \mathbf{1} & \mathbf{h} \\ \mathbf{0} & \mathbf{q} \end{pmatrix}$, $sk = \begin{pmatrix} \mathbf{f} & \mathbf{F} \\ \mathbf{g} & \mathbf{G} \end{pmatrix}$;

Algorithm 2: NTRUSign(pk, sk, μ)

Input: Public and private keys, and $\mu \in \{0,1\}^*$ the message to sign
Output: $\mathbf{s}$ the NTRU signature
begin
  $cpt \leftarrow 0$;
  repeat
    $cpt \leftarrow cpt + 1$;
    $\mathbf{m} \leftarrow H(\mu, cpt) \in \mathcal{R}_q$;
    $(\mathbf{x}, \mathbf{y}) = (\mathbf{0}, \mathbf{m}) \cdot \begin{pmatrix} \mathbf{G} & -\mathbf{F} \\ -\mathbf{g} & \mathbf{f} \end{pmatrix} / q$;
    $\mathbf{s} = -\{\mathbf{x}\} * \mathbf{f} - \{\mathbf{y}\} * \mathbf{g}$;
  until $\|(\mathbf{s}, \mathbf{s} * \mathbf{h} - \mathbf{m})\|_\nu \leq \mathcal{N}$;
  return $(\mathbf{s}, cpt)$;

Algorithm 3: Verify($pk = \mathbf{h}, \mathbf{s}, cpt, \mu$)

Input: Public key pk, the signature $\mathbf{s}$, and the message $\mu \in \{0,1\}^*$
Output: ***true*** if and only if $\mathbf{s}$ is a valid signature of μ
begin
  $\mathbf{m} = H(\mu, cpt)$;
  if $\|(\mathbf{s}, \mathbf{s} * \mathbf{h} - \mathbf{m})\|_\nu \leq \mathcal{N}$ **then**
    return ***true***;
  else
    return ***false***;

given the security parameter k, the signing tolerance ρ [3], and an upper bound N_{max} on the degree of the polynomials $\mathbf{f}$ and $\mathbf{g}$. Even if this algorithm doesn't take into account best known attacks, it can provide one with a hint of how the parameters should look like, relatively to one another. Therefore we will use it to get N, q, d, $\mathcal{N}$, and ν, and then check that best known attacks are out of range. We will not care about the transcript length as the fix we propose hides the leaky part of the signature, and an adversary would not learn anything more from issued signatures.

4 General overview of Lyubashevsky's scheme

In this section, we recall briefly the signature scheme presented by Lyubashevsky at EuroCrypt'12, and refer the reader to the original paper [19] for more details. The most efficient instantiations of this scheme rely on the average-case hardness of two problems : the $\text{SIS}_{q,n,m,d}$ decisional problem and the ℓ_2-$\text{SIS}_{q,n,m,\beta}$ problem, which are at least as hard as the worst-case of the $\mathcal{O}(n^{1.5})$-SIVP [1].

As mentioned by the author, key sizes can be shrunk by a factor k using more structured matrices and relying on the ring version of the SIS problem, but we will skip this detail in this section for simplicity. Public and private keys are respectively uniformly random matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{S} \in \{-d, \ldots, 0, \ldots, d\}^{m \times k}$ and the signature process invokes a random oracle $H : \{0,1\}^* \to \left\{\mathbf{v} : \mathbf{v} \in \{0,1\}^k, \|\mathbf{v}\|_1 \leq \kappa\right\}$. A signature $(\mathbf{z}, \mathbf{c})$ of a message μ corresponds to a combination of the secret key and the hash of this message, shifted by a commitment value also used in the random oracle.

5 Description of our scheme

5.1 Putting the pieces together

Before exposing our scheme, we want to recall an important property over the NTRU lattice that we will make use of. We denote :

$$\Lambda_q^\perp(\mathbf{P}) = \left\{(\mathbf{y_1}, \mathbf{y_2}) / (\mathbf{1} \quad \mathbf{h}) \cdot (\mathbf{y_1} \quad \mathbf{y_2})^t = \mathbf{0} \mod q\right\} = \left\{(-\mathbf{h} * \mathbf{x} \mod q, \mathbf{x}), \mathbf{x} \in \mathbb{Z}_q^N\right\} \tag{6}$$

[3] if $\mathcal{E}$ is the expected size of a signature, the verifying process should fail for every signature whose size is greater than $\rho\mathcal{E}$. Notice that the author also use one in [19], namely η. We will be tempted to set $\rho = \eta$ in next sections.

Algorithm 4: KeyGen(n, m, k, q)

Input: n, m, k, q
Output: $pk = (\mathbf{A}, \mathbf{T}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times k}$ and $sk = \mathbf{S} \in \mathbb{Z}_q^{m\times k}$
begin
  $\mathbf{S} \xleftarrow{\$} \{-d, \dots, 0, \dots, d\}^{m\times k}$;
  $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$;
  $\mathbf{T} \leftarrow \mathbf{AS}$;
  return $pk = (\mathbf{A}, \mathbf{T}), sk = \mathbf{S}$;

Algorithm 5: Sign(pk, sk, μ)

Input: Public and private keys, and $\mu \in \{0,1\}^*$ the message to sign
Output: $(\mathbf{z}, \mathbf{c})$ the signature
begin
  $\mathbf{y} \xleftarrow{\$} D_\sigma^m$;
  $\mathbf{c} \leftarrow H(\mathbf{Ay}, \mu)$;
  $\mathbf{z} \leftarrow \mathbf{Sc} + \mathbf{y}$;
  return $(\mathbf{z}, \mathbf{c})$ *with probability* $\min\left(\frac{D_\sigma^m(\mathbf{z})}{M \cdot D^m_{\mathbf{Sc},\sigma}(\mathbf{z})}, 1\right)$;

Algorithm 6: Verify$(pk, (\mathbf{z}, \mathbf{c}), \mu)$

Input: Public key, message μ, and the signature $(\mathbf{z}, \mathbf{c})$ to check
Output: ***true*** if and only if $(\mathbf{z}, \mathbf{c})$ is a valid signature of μ
begin
  if $H(\mathbf{Az} - \mathbf{Tc}, \mu) = \mathbf{c}$ ***and*** $\|\mathbf{z}\| \leq \eta\sigma\sqrt{m}$ **then**
    return ***true***;
  else
    return ***false***;

Then one can see $\Lambda_q^\perp(\mathbf{P}) = q \cdot \Lambda_q(\mathbf{P})^*$, and $\Lambda_q(\mathbf{P}) = q \cdot \Lambda_q^\perp(\mathbf{P})^*$. If we borrow the notation from code-based cryptography, if $\Lambda_q(\mathbf{P})$ is generate by $\mathbf{P} = (\mathbf{1}, \mathbf{h})$ then $\Lambda_q^\perp(\mathbf{P})$ is generated by $(\mathbf{h}, -\mathbf{1})$.

As the key generation part of our scheme is exclusively constituted by the `NTRUSign` key generation process, we use the algorithm described in [10] to get N, q, d, $\mathcal{N}$, and ν, then invoke algorithm 1 for our keyGen procedure, to get the public and private matrices $\mathbf{P}$ and $\mathbf{S}$ as depicted in algorithm 7.

To sign a message $\mu \in \{0,1\}^*$, we will need a regular random oracle $H : \{0,1\}^* \to \mathcal{R}_q$. To add some randomness to our signature, the oracle's input will be an element of $\mathcal{R}_q$ represented under a bit string concatenated with our message μ. We then `NTRUSign` the oracle's output to get our leaky sample, which we shift by a mask $(\mathbf{y_1}, \mathbf{y_2})$ large enough to statistically hide this leak. Finally, we apply a rejection sampling step to ensure that the overall signature follows the expected distribution.

Algorithm 7: KeyGen$(N,\ q,\ d,\ \mathcal{N},$ and $\nu)$

Input: N, q, d, $\mathcal{N}$, and ν
Output: $pk = \mathbf{h} = \mathbf{g} * \mathbf{f}^{-1} \mod q$ and $sk = \mathbf{f}, \mathbf{g}$
begin
  repeat
    $\mathbf{f} \xleftarrow{\$} \mathcal{T}(d)$, $\mathbf{g} \xleftarrow{\$} \mathcal{T}(d)$;
  until $\mathbf{f}$ *is invertible in* $\mathcal{R}_q$;
  $\mathbf{h} = \mathbf{g} * \mathbf{f}^{-1}$;
  return $\mathbf{P} = (-\mathbf{h}, \mathbf{1})$, $\mathbf{S} = (\mathbf{f}, \mathbf{g})$;

Algorithm 8: Sign$(\mathbf{P}, \mathbf{S}, \mu)$

Input: Public and private keys, and $\mu \in \{0,1\}^*$ the message to sign
Output: $(\mathbf{x_1}, \mathbf{x_2}), \mathbf{e}$ the signature
begin
  $\mathbf{y_1} \xleftarrow{\$} D_\sigma^N$, $\mathbf{y_2} \xleftarrow{\$} D_\sigma^N$;
  $\mathbf{e} = H(\mathbf{P}(\mathbf{y_1}, \mathbf{y_2}), \mu) = H(\mathbf{y_2} - \mathbf{h} * \mathbf{y_1}, \mu)$;
  $(\mathbf{s}, \mathbf{t}) = \mathtt{NTRUSign}_{\mathbf{S}}(\mathbf{0}, \mathbf{e})$;
  $(\mathbf{x_1}, \mathbf{x_2}) = (\mathbf{0}, \mathbf{e}) - (\mathbf{s}, \mathbf{t}) + (\mathbf{y_1}, \mathbf{y_2})$;
return $(\mathbf{x_1}, \mathbf{x_2}), \mathbf{e}$ *with probability* $\min\left(\frac{D_\sigma^{2N}(\mathbf{x})}{M \cdot D^{2N}_{(-\mathbf{s},\mathbf{e}-\mathbf{t}),\sigma}(\mathbf{x})}, 1\right)$;

We insist on the fact that in the original scheme *with perturbations*, the aim was to sign the message μ enough times so that the transcript an adversary could collect with couples of messages and signatures is short enough to make all secret key recovery techniques fail. The main difference in our scheme consists in hiding the leaky part with something larger so that it becomes indistinguishable, whether than sign it again and again. In other words, the leaky part of `NTRUSign` plays the role of the secret key in [19].

Algorithm 9: Verify$(\mathbf{P}, (\mathbf{x_1}, \mathbf{x_2}), \mathbf{e}, \mu)$

Input: Public key $\mathbf{P}$, a signature $(\mathbf{x_1}, \mathbf{x_2}), \mathbf{e}$, and a message μ
Output: ***true*** if and only if $(\mathbf{x_1}, \mathbf{x_2}), \mathbf{e}$ is a valid signature of μ
begin
  if $\|(\mathbf{x_1}, \mathbf{x_2})\|_2 \leq \eta\sigma\sqrt{2N}$ ***and*** $H(\mathbf{P} \cdot (\mathbf{x_1}, \mathbf{x_2}) - \mathbf{e}, \mu) = \mathbf{e}$ **then**
    return ***true***;
  else
    return ***false;***

5.2 Sets of parameters

Hereafter is a set of parameters for our signature scheme, given different security levels. One interesting aspect of those sets is that we had to raise q and N for our `NTRUSign` part to be secure, but due to the small norm of our `NTRU` signature, this q is not raised as much as in [19]. This results in the nice property of lowering key and signature sizes.

Security parameter (bits) k	100	100	112	112	128	128	160	160
Optimized for	Size	Speed	Size	Speed	Size	Speed	Size	Speed
N	431	431	479	479	541	541	617	617
d	34	34	42	42	61	61	57	57
$\log_2(q)$	16	16	16	16	16	16	16	16
η (lemma 2.3.2)	1.296	1.296	1.297	1.297	1.299	1.299	1.314	1.314
ν	0.16686	0.16686	0.15828	0.15828	0.14894	0.14894	0.13946	0.13946
$\mathcal{N}$	109	139	128	165	160	213	165	218
α (lemma 2.3.1)	6	12	6.5	13	7	14	7.5	15
$\sigma = \eta\alpha\mathcal{N}$	848	2162	1080	2783	1455	3874	1627	4297
$M = e^{1+1/(2\alpha^2)}$ (lemma 2.3.3)	7.492	2.728	7.477	2.726	7.465	2.725	7.455	2.724
approximate signature size (bits) $\approx 2N\log_2(\alpha\sigma)$	10700	12700	12300	14600	14500	17100	16800	19800
approximate pk size (bits) $\approx N\log_2 q$	$2^{12.8}$	$2^{12.8}$	$2^{12.9}$	$2^{12.9}$	$2^{13.1}$	$2^{13.1}$	$2^{13.3}$	$2^{13.3}$
approximate sk size (bits) $\approx 2N\log_2(3)$	$2^{10.4}$	$2^{10.4}$	$2^{10.6}$	$2^{10.6}$	$2^{10.7}$	$2^{10.7}$	$2^{10.9}$	$2^{10.9}$

Table 3: Parameters, signature and key sizes for our scheme, given the security level k

5.3 Security of our scheme

In this section, k will represent the security parameter, typically $k = 80$ for a "toy" security, $k = 100$ or 112 for a current security, and $k = 128$ to 160 for a "strong" security. Due to space restrictions, we will only mention the different known kinds of attack the reader can find in the literature. For further details, we refer to [11, 20, 6, 14] for the `NTRUSign` part, and to [19, 18, 23] for Lyubashevsky's scheme. Due to the hybridness of our scheme, potential attacks could be of three types, that we exposed in what follows, before tackling them.

The first one consists in attacking the `NTRU` lattice by trying to find back the private key $(\mathbf{f}, \mathbf{g})$ only from the public key $\mathbf{h} = \mathbf{g} * \mathbf{f}^{-1}$ (and eventually some signatures after Lyubashevsky's layer). Even if there is no theoretical proof on the intractability of this attack, there hasn't been (to the best of our knowledge) any efficient way to do so neither. Parameters given in table 3 have been chosen so that someone succeeding in doing so would achieve a lattice reduction with better Hermit factors than those described in 4 respectively to the security parameter k. Such a good algorithm could obviously be used to solve worst-case of lattice problems on general convolution modular lattices. A second way to break our signature scheme, still by finding out the secret key, could be trying to isolate the `NTRU` signature inside our signature to find enough leaky parts to then proceed to a [6]-like attack. This issue is addressed by

Theorem 2.3.5. Finally, we show that if an adversary succeed in creating a forgery in polynomial-time, then we can use this forgery to solve the SIS problem, which is the main theorem (5.3.1) of this section.

Regarding attacks against the NTRUSign, all parameters have been heighten so they ensure way more than k bits of security. We are aware that some attacks might lower the security level [20, 7, 14, 6], but also due to our lack of knowledge on how to benefit from the singular structure of NTRU lattices, we take a conservative gap between claimed and effective security. Nevertheless, all parameters given in table 2 were set in such a way that lattice reduction techniques are meant to fail, either by finding a short vector too long, either by a computational complexity blow up. Also due to recent attacks such as [14, 7, 6], the NTRUSign parameters presented in [11] don't reach the claimed security. Therefore, we ran the Baseline Parameter Generation Algorithm of [10], and integrated the most recent known attacks. As one can notice, we intentionally took a "huge" degree N, and a big q for two reasons. It first gives more confidence about the security of the underlying NTRU lattice, and it was also necessary for proofs to work after applying Lyubashevsky's layer to our scheme.

As far as we know, lattice-reduction over $\Lambda_q^{\perp}(\mathbf{A})$ is the most efficient technique to solve random instances of knapsack problems. Experiments in [7] led to the ability of finding a vector $\mathbf{v} \in \Lambda_q^{\perp}(\mathbf{A})$ whose norm is at most $\|\mathbf{v}\|_2 \leq \delta^{2N} \cdot \sqrt{q}$, for δ depending on the lattice-reduction algorithm which is used (see below). Experiments from Micciancio and Regev [23] conducted to a minimum of $\delta^m \cdot q^{n/m} \approx \min(q, 2^{2\sqrt{N \log_2(q) \log_2(\delta)}})$ for $m \approx \sqrt{N \log_2(q)/\log_2(\delta)}$.

In 2011, Lindner and Peikert [18] achieved to give an approximation of the best δ reachable for a given security level k, using a conservative approximation on BKZ's running time :

$$t_{BKZ}(\delta) = \log_2(T_{BKZ}(\delta)) = 1.8/\log_2(\delta) - 110 \tag{7}$$

where $T_{BKZ}(\delta)$ is the running time of BKZ in second, on their machine. So assuming one can achieve 2^{30} operations per second on a "standard" computer, to determine δ given the security parameter k, we have :

$$\log_2(\delta) := \frac{1.8}{\log_2(\frac{T_{BKZ}(\delta)}{2^{30}}) + 110} = \frac{1.8}{k - 30 + 110} = \frac{1.8}{k + 80} \tag{8}$$

This equation gives us a way to get δ as a function of the security parameter k, see table 4. Similarly to [19], in order to hide properly our leaky part $(\mathbf{0}, \mathbf{e}) - (\mathbf{s}, \mathbf{t})$, we will use Lemmas 2.3.1 and 2.3.2 to get a proper α.

k	100	112	128	160
δ	1.00696	1.00652	1.00602	1.00521
α	12	13	14	15

Table 4: δ and α as a function of the security level k

Against forgeries. In this section, we give a short overview of the material that will be needed to base our signature scheme upon the SIS problem over random NTRU lattices. This leads to a signature scheme based on the worst-case hardness of the $\tilde{\mathcal{O}}(N^{1.5})$-SIVP problem over general convolutional modular lattices.

We now expose the core of the reduction, which allows us to base the security of our signature scheme upon the ℓ_2-$\mathrm{SIS}_{q,N,2N,\beta}$ Problem of general NTRU lattices. Our main theorem will be proved by two lemmas, mostly proved in [19], but revisited in appendix A in some of the details in order to fit best with our sets of parameters.

Theorem 5.3.1 ([19] revisited) *Assume there is polynomial-time forger $\mathcal{F}$, which makes at most s (resp. h) queries to the signing (resp. random) oracle, who breaks our signature scheme (with parameters such those in Table 3), then there is a polynomial-time algorithm to solve the ℓ_2-$SIS_{q,N,2N,\beta}$ Problem for $\beta = 2\eta\sigma\sqrt{2N}$ with probability $\approx \frac{\delta^2}{h+s}$. Moreover, the signing algorithm 8 produces a signature with*

Hybrid 1

Sign$(\mathbf{P}, \mathbf{S}, \mu)$

1. $\mathbf{y_1} \xleftarrow{\$} D_\sigma^N$, $\mathbf{y_2} \xleftarrow{\$} D_\sigma^N$
2. $\mathbf{e} \xleftarrow{\$} \mathcal{R}_q$
3. $(\mathbf{s}, \mathbf{t}) = \texttt{NTRUSign}_{\mathbf{S}}(\mathbf{0}, \mathbf{e})$
4. $(\mathbf{x_1}, \mathbf{x_2}) = (\mathbf{0}, \mathbf{e}) - (\mathbf{s}, \mathbf{t}) + (\mathbf{y_1}, \mathbf{y_2})$
5. with probability $\min(\frac{D_\sigma^{2N}(\mathbf{x})}{M \cdot D^{2N}_{(-\mathbf{s},\mathbf{e}-\mathbf{t}),\sigma}(\mathbf{x})}, 1)$:
   - Output $(\mathbf{x_1}, \mathbf{x_2}), \mathbf{e}$
   - Program $H(\mathbf{P} \cdot (\mathbf{x_1}, \mathbf{x_2}) - \mathbf{e}, \mu) = \mathbf{e}$

Hybrid 2

Sign$(\mathbf{P}, \mathbf{S}, \mu)$

1. $\mathbf{e} \xleftarrow{\$} \mathcal{R}_q$
2. $(\mathbf{x_1}, \mathbf{x_2}) \xleftarrow{\$} D_\sigma^{2N}$
3. with probability $1/M$:
   - Output $(\mathbf{x_1}, \mathbf{x_2}), \mathbf{e}$
   - Program $H(\mathbf{P} \cdot (\mathbf{x_1}, \mathbf{x_2}) - \mathbf{e}, \mu) = \mathbf{e}$

Fig. 1: Signing Hybrids

probability $\approx \frac{1}{M}$ *and the verifying algorithm 9 accepts the signature produced by an honest signer with probability at least* $1 - 2^{-2N}$.

Proof. We begin the proof by showing that our signature algorithm 8 is statistically close (within distance $\epsilon = s(h+s) \cdot 2^{-N+1} + s \cdot \frac{2^{-\omega(\log_2 2N)}}{M}$ by Lemma 5.3.2) to the one in Hybrid 2 in Figure 1. Given that Hybrid 2 outputs something with probability $1/M$, our signing algorithm will output something too with probability $(1-\epsilon)/M$. Then by Lemma 5.3.3, we show that if a forger $\mathcal{F}$ succeeds in forging with probability δ when the signing algorithm is replaced by the one in Hybrid 2, then we can use $\mathcal{F}$ to come up with a non-zero lattice vector $\mathbf{v}$ such that $\|\mathbf{v}\| \leq 2\eta\sigma\sqrt{2N}$ and $\mathbf{Pv} = 0$ with probability at least $\left(\delta - 2^{-k}\right)\left(\frac{\delta - 2^{-k}}{h+s} - 2^{-k}\right)$. □

Lemma 5.3.2 ([19] revisited) *Let $\mathcal{D}$ be a distinguisher who can query the random oracle H and either the actual signing algorithm 8 or Hybrid 2 in Figure 1. If he makes h queries to H and s queries to the signing algorithm that he has access to, then for all but a $e^{-\Omega(N)}$ fraction of all possible matrices $\mathbf{P}$, his advantage of distinguishing the actual signing algorithm from the one in Hybrid 2 is at most $s(h+s) \cdot 2^{-N+1} + s \cdot \frac{2^{-\omega(\log_2 2N)}}{M}$.*

Lemma 5.3.3 ([19] revisited) *Suppose there exists a polynomial-time forger $\mathcal{F}$ who makes at most h queries to the signer in Hybrid 2, s queries to the random oracle H, and succeeds in forging with probability δ. Then there exists an algorithm of the same time-complexity as $\mathcal{F}$ that for a given $\mathbf{P} \xleftarrow{\$} \mathbb{Z}_q^{N \times 2N}$ finds a non-zero $\mathbf{v}$ such that $\|\mathbf{v}\|_2 \leq 2\eta\sigma\sqrt{2N}$ and $\mathbf{Pv} = \mathbf{0}$ with probability at least*

$$\left(\delta - 2^{-k}\right)\left(\frac{\delta - 2^{-k}}{h+s} - 2^{-k}\right).$$

Conclusion

In this work, we described a method for sealing `NTRUSign` signatures' leak, based on the worst-case hardness of standard problems over ideal lattices. This method differs from existing heuristic countermeasures such the use of perturbations [12] or the deformation of the parallelepiped [15] - both broken [6] - but also from provably secure modifications of `NTRUSign` like [26] which uses gaussian sampling techniques in order to not disclose the secret basis [8]. Moreover, this technique seems to be sufficently generic to be applied on GGH signatures. Details on this will be provided in a longer version of this paper.

We show that it is actually possible to use the rejection sampling technique from [19] instead of gaussian sampling to achieve zero-knnowledgeness, while keeping most of `NTRUSign`'s efficiency. Moreover, parameter refinements allowed us to lower the rejection rate, leading to performance improvements regarding [19], together with smaller signature and secret key sizes.

It might be possible to imrove the rejection sampling procedure even more using techniques such those in [5], but it seems necessary to break the public key's particular shape to do so. Therefore, it is still an open question whether the resulting benefit in the signature size would worth the key sizes growth.

References

[1] Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: In Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, ACM (1996) 99–108

[2] Babai, L.: On lovász' lattice reduction and the nearest lattice point problem (shortened version). In: Proceedings of the 2nd Symposium of Theoretical Aspects of Computer Science. STACS '85, London, UK, UK, Springer-Verlag (1985) 13–20

[3] Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. CCS '06, New York, NY, USA, ACM (2006) 390–399

[4] Chen, Y., Nguyen, P.Q.: Bkz 2.0: Better lattice security estimates. In: ASIACRYPT. Volume 7073 of Lecture Notes in Computer Science., Springer (2011) 1–20

[5] Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. In: CRYPTO (1). (2013) 40–56

[6] Ducas, L., Nguyen, P.Q.: Learning a zonotope and more: cryptanalysis of ntrusign countermeasures. In: Proceedings of the 18th international conference on The Theory and Application of Cryptology and Information Security. ASIACRYPT'12, Berlin, Heidelberg, Springer-Verlag (2012) 433–450

[7] Gama, N., Nguyen, P.Q.: Predicting Lattice Reduction. In: Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology. EUROCRYPT'08, Berlin, Heidelberg, Springer-Verlag (2008) 31–51

[8] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC. (2008) 197–206

[9] Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology. CRYPTO '97, London, UK, UK, Springer-Verlag (1997) 112–131

[10] Hoffstein, J., Howgrave-graham, N., Pipher, J., Silverman, J.H., Whyte, W.: Performance Improvements and a Baseline Parameter Generation Algorithm for NTRUSign. In: In Proc. of Workshop on Mathematical Problems and Techniques in Cryptology. (2005) 99–126

[11] Hoffstein, J., Howgrave-Graham, N., Pipher, J., Whyte, W.: Practical lattice-based cryptography: `NTRUEncrypt` and `NTRUSign`. Nguyen, Phong Q. (ed.) et al., The LLL algorithm. Survey and applications. Dordrecht: Springer. Information Security and Cryptography, 349-390 (2010). (2010)

[12] Hoffstein, J., Howgrave-graham, N., Pipher, J., Silverman, J.H., Whyte, W.: NTRUSign: Digital Signatures Using the NTRU Lattice. In: City University of Hong Kong, Springer-Verlag (2002)

[13] Hoffstein, J., Pipher, J., Silverman, J.H.: Nss: An ntru lattice-based signature scheme. In: EUROCRYPT. (2001) 211–228

[14] Howgrave-Graham, N.: A hybrid lattice-reduction and meet-in-the-middle attack against ntru. In: CRYPTO. (2007) 150–169

[15] Hu, Y., Wang, B., He, W.: Ntrusign with a new perturbation. IEEE Trans. Inf. Theor. **54**(7) (July 2008) 3216–3221

[16] Impagliazzo, R., Naor, M.: Efficient cryptographic schemes provably as secure as subset sum. Journal of Cryptology **9** (1996) 236–241

[17] Lenstra, H.j., Lenstra, A., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261** (1982) 515–534

[18] Lindner, R., Peikert, C.: Better Key Sizes (and Attacks) for LWE-based Encryption. In: Proceedings of the 11th international conference on Topics in cryptology: CT-RSA 2011. CT-RSA'11, Berlin, Heidelberg, Springer-Verlag (2011) 319–339

[19] Lyubashevsky, V.: Lattice Signatures Without Trapdoors. In: Proceedings of the 31st Annual international conference on Theory and Applications of Cryptographic Techniques. EUROCRYPT'12, Berlin, Heidelberg, Springer-Verlag (2012) 738–755

[20] May, A., Silverman, J.H.: Dimension reduction methods for convolution modular lattices. In Silverman, J.H., ed.: Cryptography and Lattices, International Conference, CaLC 2001, Providence, RI, USA, March 29-30, 2001, Revised Papers. Volume 2146 of Lecture Notes in Computer Science., Springer (2001) 110–125

[21] Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of lwe search-to-decision reductions. In: Proceedings of the 31st annual conference on Advances in cryptology. CRYPTO'11, Berlin, Heidelberg, Springer-Verlag (2011) 465–484
[22] Micciancio, D., Regev, O.: Worst-case to Average-case reductions based on Gaussian measure. SIAM Journal on Computing **37**(1) (2007) 267–302 Preliminary version in FOCS 2004.
[23] Micciancio, D., Regev, O.: Lattice-based Cryptography. In Bernstein, D., Buchmann, J., Dahmen, E., eds.: Post-Quantum Cryptography. Springer Berlin Heidelberg (2009) 147–191
[24] Nguyen, P.Q., Regev, O.: Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures. In: Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques. Volume 4004 of Lecture Notes in Computer Science., Springer (2006) 271–288
[25] Stehlé, D., Steinfeld, R.: Making ntru as secure as worst-case problems over ideal lattices. In: Proceedings of the 30th Annual international conference on Theory and applications of cryptographic techniques: advances in cryptology. EUROCRYPT'11, Berlin, Heidelberg, Springer-Verlag (2011) 27–47
[26] Stehlé, D., Steinfeld, R.: Making ntruencrypt and ntrusign as secure as standard worst-case problems over ideal lattices. Cryptology ePrint Archive, Report 2013/004 (2013) http://eprint.iacr.org/.

A Proofs

A.1 Section 2

Most of the lemmas of Section 2 are proved in [19]. We therefore refer the reader to the original paper for these proofs. Nevertheless, we adapted lemma 2.3.3 to make bounds tigher with respect to different security levels. We prove the corectness of our modification :

Proof.

$$D_{\sigma}^{2N}(\mathbf{x})/D_{\mathbf{v},\sigma}^{2N}(\mathbf{x}) = \rho_{\sigma}^{2N}(\mathbf{x})/\rho_{\mathbf{v},\sigma}^{2N}(\mathbf{x}) = \exp\left(\frac{\|\mathbf{x}-\mathbf{v}\|^2 - \|\mathbf{x}\|^2}{2\sigma^2}\right) = \exp\left(\frac{\|\mathbf{v}\|^2 - 2\langle \mathbf{x}, \mathbf{v}\rangle}{2\sigma^2}\right)$$

By lemma 2.3.1 and using the fact that $\sigma = \omega(\|\mathbf{v}\|\sqrt{\log(2N)})$, with probability $1 - 2^{-\omega(\log(2N))}$ we have

$$\exp\left(\frac{\|\mathbf{v}\|^2 - 2\langle \mathbf{x}, \mathbf{v}\rangle}{2\sigma^2}\right) < \exp\left(\frac{\|\mathbf{v}\|^2 + \omega(\sigma\|\mathbf{v}\|\sqrt{\log(2N)})}{2\sigma^2}\right) = \mathcal{O}(1).$$

And more precisely, by setting $r = \alpha\|\mathbf{v}\|\sigma$ in lemma 2.3.1 with α determined by the security parameter k in table 1, we obtain with probability $1 - 2^{-k}$ that

$$\exp\left(\frac{\|\mathbf{v}\|^2 - 2\langle \mathbf{x}, \mathbf{v}\rangle}{2\sigma^2}\right) < \exp\left(\frac{\|\mathbf{v}\|^2 + 2\alpha\|\mathbf{v}\|\sigma}{2\sigma^2}\right) = \exp\left(\frac{\|\mathbf{v}\|^2}{2\sigma^2} + \frac{\alpha\|\mathbf{v}\|}{\sigma}\right) \overset{\sigma=\alpha\|\mathbf{v}\|}{=} e^{1+1/(2\alpha^2)}.$$

□

A.2 Proofs of Section 5

We begin with the proof of lemma 5.3.2, which states that our actual signing algorithm 8 is indistinguishable from Hybrid 2 depicted in Figure 1, using Hybrid 1 as an intermediate step.

Proof. First, let us prove that $\mathcal{D}$ has an advantage of at most $s(h+s)\cdot 2^{-N+1}$ of distinguishing between the actual signature scheme 8 and Hybrid 1. The only difference between those algorithms is the output of the random oracle H. It is chosen uniformly at random from $\mathcal{R}_q$ in Hybrid 1, rather than according to $H(\mathbf{Py}, \mu)$ for $y \overset{\$}{\leftarrow} D_q^{2N}$ in the real signing algorithm. Random oracle in Hybrid 1 is then programmed to answer $H(\mathbf{Px} - \mathbf{e}, \mu) = H(\mathbf{Py}, \mu)$ without checking whether $(\mathbf{Py}, \mu)$ was already queried or not. Since $\mathcal{D}$

calls H (resp. algorithm 8) h (resp s) times, at most $s+h$ values of $(\mathbf{Py}, \mu)$ will be set. We now bound the probability of generating such an already set value. Using lemma 2.3.2, we can see that for any $\mathbf{t} \in \mathbb{Z}_q^N$,

$$Pr[\mathbf{Py} = \mathbf{t}; \mathbf{y} \xleftarrow{\$} D_q^{2N}] = Pr[\mathbf{y_1} = (\mathbf{t} - \mathbf{h} * \mathbf{y_0}); \mathbf{y} \xleftarrow{\$} D_q^{2N}] \leq \max_{\mathbf{t}' \in \mathbb{Z}_q^N} Pr[\mathbf{y_1} = \mathbf{t}'; \mathbf{y_1} \xleftarrow{\$} D_q^N] \leq 2^{-N}.$$

Therefore, if Hybrid 1 is called s times with the probability of getting a collision begging less than $(s+h) \cdot 2^{-N+1}$ for each call, then the probability of coming up with a collision after s calls is at most $s(s+h) \cdot 2^{-N+1}$.

We pursue by showing that the outputs of Hybrids 1 and 2 are statistically within distance $\frac{2^{-\omega(\log_2 2N)}}{M}$. As noticed in [19], this is an almost straightforward consequence of theorem 2.3.5 : assuming both Hybrids output $(\mathbf{x}, (-\mathbf{s}, \mathbf{e} - \mathbf{t}))$ with respective probabilities $\min\left(\frac{D_\sigma^{2N}(\mathbf{x})}{M \cdot D^{2N}_{(-\mathbf{s},\mathbf{e}-\mathbf{t}),\sigma}(\mathbf{x})}, 1\right)$ for Hybrid 1 and $1/M$ for Hybrid 2, they respectively play the role of $\mathcal{A}$ and $\mathcal{F}$ (with $T = \eta\alpha\mathcal{N}$). Even if both Hybrids only output $\mathbf{e}$, this does not increase the statistical distance because given $\mathbf{e}$, one can generate $(-\mathbf{s}, \mathbf{e} - \mathbf{t})$ such that $\mathbf{P}(-\mathbf{s}, \mathbf{e} - \mathbf{t}) = \mathbf{e}$ simply by `NTRUSigning` $(\mathbf{0}, \mathbf{e})$, and this will have the exact same distribution as the $\mathbf{e}$ in both Hybrids. Finally, as the signing oracle is called s times, the statistical distance between the two Hybrids is at most $s \cdot \frac{2^{-\omega(\log_2 2N)}}{M}$, or more concretely $s \cdot \frac{2^{-k}}{M}$. The claim in the lemma is obtained by summing both distances. □

We now prove lemma 5.3.3, which provides us with a ℓ_2-SIS$_{q,N,2N,\beta}$ solver using a polynomial-time successful forger.

Proof. Let $t = h + s$ be the number of calls to the random oracle H during $\mathcal{F}$'s attack. H can be either queried by the forger or programmed by the signing algorithm when $\mathcal{F}$ asks for some message to be signed. We pick random coins ϕ (resp. ψ) for the forger (resp. the signer), along with $\mathbf{r}_1, \ldots, \mathbf{r}_t \leftarrow \mathcal{R}_q$, which will correspond to the H's responses. We now consider a subroutine $\mathcal{A}$, which on input $(\mathbf{P}, \phi, \psi, \mathbf{r}_1, \ldots, \mathbf{r}_t)$ initializes $\mathcal{F}$ by giving it $\mathbf{P}$ and ϕ and run it. Each time $\mathcal{F}$ asks for a signature, $\mathcal{A}$ runs Hybrid 2 using the signer's coins ψ to get a signature, and H is programmed to answer with the first unused $\mathbf{r}_i \in (\mathbf{r}_1, \ldots, \mathbf{r}_t)$. $\mathcal{A}$ keeps track of the answered $\mathbf{r}_i$ in case $\mathcal{F}$ queries the same message to be signed again. Similarly, if $\mathcal{F}$ queries directly the random oracle, H will answer with the first unused $\mathbf{r}_i \in (\mathbf{r}_1, \ldots, \mathbf{r}_t)$, unless the query was already made. When $\mathcal{F}$ ends and eventually come up with an output (with probability δ), $\mathcal{A}$ simply forwards $\mathcal{F}$'s output.

With probability δ, $\mathcal{F}$ succeeds in forging, coming up with $(\mathbf{x}, \mathbf{e})$ satisfying $\|\mathbf{x}\| \leq \eta\sigma\sqrt{2N}$ and $H(\mathbf{Px} - \mathbf{e}, \mu) = \mathbf{e}$ for some message μ. If H was not queried nor programmed on some input $\mathbf{w} = \mathbf{Px} - \mathbf{e}$, then $\mathcal{F}$ has only a $1/|\mathcal{R}_q| = q^{-N}$ (*i.e. negligible*) chance of generating a $\mathbf{e}$ such that $\mathbf{e} = H(\mathbf{w}, \mu)$. Therefore, $\mathcal{F}$ has at least a $\delta - q^{-N}$ chance of succeeding in a forgery with $\mathbf{e}$ being one of the $\mathbf{r}_i$'s. Assume $\mathbf{e} = \mathbf{r}_j$, we are left with two cases : $\mathbf{r}_j$ is a response to a random oracle query made by $\mathcal{F}$, or it was program during the signing procedure invoked by $\mathcal{A}$.

Let first assume that the random oracle was programmed to answer $H(\mathbf{Px}' - \mathbf{e}, \mu') = \mathbf{e}$ on input μ'. If $\mathcal{F}$ succeeds in forging $(\mathbf{x}, \mathbf{e})$ for some (possibly different) message μ, then $H(\mathbf{Px}' - \mathbf{e}, \mu') = H(\mathbf{Px} - \mathbf{e}, \mu)$. If $\mu \neq \mu'$ or $\mathbf{Px}' - \mathbf{e} \neq \mathbf{Px} - \mathbf{e}$, then $\mathcal{F}$ found a pre-image of $\mathbf{r}_j$. Therefore, $\mu = \mu'$ and $\mathbf{Px}' - \mathbf{e} = \mathbf{Px} - \mathbf{e}$, so that $\mathbf{P}(\mathbf{x} - \mathbf{x}') = \mathbf{0}$. We know that $\mathbf{x} - \mathbf{x}' \neq \mathbf{0}$ (because otherwise $(\mathbf{x}, \mathbf{e})$ and $(\mathbf{x}', \mathbf{e})$ sign the same message μ), and since $\|\mathbf{x}\|_2, \|\mathbf{x}'\|_2 \leq \eta\sigma\sqrt{2N}$, we have that $\|\mathbf{x} - \mathbf{x}'\| \leq 2\eta\sigma\sqrt{2N}$.

Let now assume that $\mathbf{r}_j$ was a response of the random oracle invoked by $\mathcal{F}$. We start by recording $\mathcal{F}$'s output $(\mathbf{x}, \mathbf{r}_j)$ for the message μ, then generate fresh random elements $\mathbf{r}'_j, \ldots, \mathbf{r}'_t \leftarrow \mathcal{R}_q$. We then run $\mathcal{A}$ again with input $(\mathbf{P}, \phi, \psi, \mathbf{r}_1, \ldots, \mathbf{r}_{j-1}, \mathbf{r}'_j, \ldots, \mathbf{r}'_t)$, and by the General Forking Lemma [3], we obtain that the probability that $\mathbf{r}'_j \neq \mathbf{r}_j$ and the forger uses the random oracle response $\mathbf{r}'_j$ (and the query associated to it) in its forgery is at least

$$\left(\delta - \frac{1}{|\mathcal{R}_q|}\right)\left(\frac{\delta - 1/|\mathcal{R}_q|}{t} - \frac{1}{|\mathcal{R}_q|}\right),$$

and thus with the above probability, $\mathcal{F}$ outputs a signature $(\mathbf{x}', \mathbf{r}'_j)$ of the message μ and $\mathbf{Px}-\mathbf{e} = \mathbf{Px}'-\mathbf{e}'$ where we let $\mathbf{e} = \mathbf{r}_j$ and $\mathbf{e}' = \mathbf{r}'_j$. By rearranging terms in the above equality we obtain

$$\begin{aligned} \mathbf{P}(\mathbf{x}-\mathbf{x}') - \overbrace{(\mathbf{e}-\mathbf{e}')}^{\mathbf{P}((\mathbf{0},\mathbf{e})-(\mathbf{0},\mathbf{e}')-((\mathbf{s},\mathbf{t})-(\mathbf{s}',\mathbf{t}')))} &= \mathbf{0} \\ \mathbf{P}\left(\mathbf{y}-\mathbf{y}'\right) &= \mathbf{0} \end{aligned} \tag{9}$$

But since $H(\mathbf{Py}, \mu) = \mathbf{e} = \mathbf{r}_j \neq \mathbf{r}'_j = \mathbf{e}' = H(\mathbf{Py}', \mu)$, necessarly $\mathbf{y} \neq \mathbf{y}'$, and as $\|\mathbf{y}\|_2, \|\mathbf{y}'\|_2 \leq \eta\sigma\sqrt{2N}$, we finally have that $\|\mathbf{y}-\mathbf{y}'\|_2 \leq 2\eta\sigma\sqrt{2N}$ with probability

$$\left(\delta - 2^{-k}\right)\left(\frac{\delta - 2^{-k}}{h+s} - 2^{-k}\right).$$

□