# BAT: Small and Fast KEM over NTRU Lattices

Pierre-Alain Fouque[1], Paul Kirchner[1], Thomas Pornin[2] and Yang Yu[3]

[1] Rennes Univ, IRISA, Rennes, France pierre-alain.fouque@irisa.fr, paul.kirchner@irisa.fr
[2] NCC Group, Quebec, Canada thomas.pornin@nccgroup.com
[3] BNRist, Tsinghua University, Beijing, China yang.yu0986@gmail.com

**Abstract.** We present BAT – an IND-CCA secure key encapsulation mechanism (KEM) that is based on NTRU but follows an encryption/decryption paradigm distinct from classical NTRU KEMs. It demonstrates a new approach of decrypting NTRU ciphertext since its introduction 25 years ago. Instead of introducing an artificial masking parameter $p$ to decrypt the ciphertext, we use 2 linear equations in 2 unknowns to recover the message and the error. The encryption process is therefore close to the GGH scheme. However, since the secret key is now a short basis (not a vector), we need to modify the decryption algorithm and we present a new NTRU decoder. Thanks to the improved decoder, our scheme works with a smaller modulus and yields shorter ciphertexts, smaller than RSA-4096 for 128-bit security with comparable public-key size and much faster than RSA or even ECC. Meanwhile, the encryption and decryption are still simple and fast in spite of the complicated key generation. Overall, our KEM has more compact parameters than all current lattice-based schemes and a practical efficiency. Moreover, due to the similar key pair structure, BAT can be of special interest in some applications using Falcon signature that is also the most compact signature in the round 3 of the NIST post-quantum cryptography standardization. But different from Falcon, our KEM does not rely on floating-point arithmetic and can be fully implemented over the integers.

**Keywords:** Lattice-based cryptography · NTRU · KEM · Falcon

## 1 Introduction

Lattice-based schemes, especially when they have a polynomial structure, are a very strong contender for post-quantum cryptography. They can be faster than widely deployed cryptosystems based on RSA and ECDH. However, the sizes of public keys, signatures and ciphertexts are significantly larger than in RSA and even larger by an order of magnitude compared with ECDH cryptosystems. Such a large size is a major drawback of lattice schemes, and can be a crucial obstacle in the following situations:

- Real-world protocols may have a maximum length designed for classical cryptography. For a standard Ethernet connection, the maximum transmission unit (MTU) is 1500 bytes[1], and forward secrecy requires several objects.
- Large communication sizes increase the risk of lost packets and delays. Recent experiments on post-quantum TLS [51] show that communication sizes come to govern the performance when the packet loss rate is higher than 3%. Moreover, [57] examines how the initial TCP window size affects post-quantum TLS and SSH performance, and show that even a small size increase can reduce the observed post-quantum slowdown by 50%. In addition, transmission energy can also be a

[1] https://en.wikipedia.org/wiki/Maximum_transmission_unit and Distributions on IP packets.

significant part of the energy consumption on cryptography [55]. The size of signature schemes in TLS handshakes is also important as analyzed in [58].

– In some lightweight applications, e.g. internet of things (IoT), encryption and verification are done by some constrained devices. These devices only have small on-board storage and modest processors so that they may not be compatible with large public keys, signatures and ciphertexts.

With post-quantum cryptography standardization and deployment underway, it is important to explore new lattice-based cryptosystems with smaller parameters.

In this light, a natural choice is NTRU [39], as its structure reduces the data to one ring element. There have been many high-performance NTRU-based schemes ranging from Falcon [30], BLISS [26] for signature to NTRU-HRSS [41], NTTRU [47], NTRUEncrypt [20], NTRU Prime [9] for encryption and KEM. In particular, Falcon is the most compact signature in the round 3 of the NIST post-quantum cryptography standardization [50].

NTRU-based schemes are defined over some polynomial ring $\mathcal{R}$ that is $\mathcal{R} = \mathbb{Z}[x]/(x^n+1)$ with $n$ a power-of-2 in this work. The secret key of an NTRU cryptosystem is essentially a pair of short polynomials $(f, g) \in \mathcal{R}^2$ while the public key is $h = f^{-1}g \bmod q$. All NTRU encryption schemes, ranging from the earliest proposal [39] to the round 3 NIST submission [14], have followed essentially the same design rationale for more than 20 years. Concretely, the ciphertext is $c = phr + m \bmod q$ where $p$ is the masking modulus, $r$ is the randomness, $m$ is the message. A correct decryption is built upon that $c' = pgr + fm$ is short so that $c' = (fc \mod q)$. The masking modulus $p$ is also necessary to decrypt: one needs to first clean out $pgr$ via reduction modulo $p$ and then to recover $m$ by multiplying the inverse of $f$ modulo $p$. For typical NTRU KEMs, $(f, g)$ is sparse and of length about $C\sqrt{n}$ for small constant $C$.

By contrast, the design rationale for NTRU-based signatures went through some significant changes. The first NTRU-based signature is NTRUSign [38] that is a hash-and-sign scheme. However, its signature transcripts leak some secret key information so that NTRUSign and some variants were broken by statistical attacks [49, 28]. Later, Ducas et al. made use of the GPV hash-and-sign framework [33] and proposed a provably secure NTRU-based signature [27] that further developed into Falcon [30]. The public key of Falcon is still $h = f^{-1}g \bmod q$ for some short $(f, g)$, while the actual secret key is a trapdoor basis $\mathbf{B}_{f,g} = \begin{pmatrix} g & G \\ f & F \end{pmatrix} \in \mathcal{R}^{2\times 2}$ such that $gF - fG = q$. The signing of Falcon is essentially Gaussian sampling with a trapdoor [33, 29]. Consequently, the signature size depends on the maximal Gram-Schmidt norm of $\mathbf{B}_{f,g}$. As analyzed in [27], Falcon chooses $(f, g)$ of $\|(f, g)\| \approx 1.17\sqrt{q}$ for optimal parameters. Therefore, $\|(f, g)\|$ in Falcon is independent of $n$, which is different from the case of NTRU KEMs.

Some other NTRU-based signatures [26, 24], do just use one vector $(f, g)$ as the secret key as the case of NTRU KEMs. However, to the best of our knowledge, there is no practical NTRU-based KEM using a trapdoor basis as the secret key. Intuitively we expect $(F, G)$ to yield one more equation in decryption so that one can recover both the message and encryption randomness via two equations. This in effect gets rid of the masking modulus $p$ in classical NTRU KEMs and thus hopefully allows smaller parameters. Moreover, when we unify the trapdoor function for both signature and KEM part of the code can be shared, and we may also reduce some storage and communication. Therefore, it would be interesting to investigate the practicality of a trapdoor basis for KEM.

Indeed, the earliest lattice-based cryptosystem GGH (Goldreich-Goldwasser-Halevi [34]) uses a trapdoor basis as the secret key and implements both encryption and signature based on that trapdoor function. Later Micciancio improved the GGH trapdoor function by using the Hermite normal form [48]. While GGH encryption has a long history as NTRU, its practicality is far from well-studied and there is no GGH-like encryption/KEM with concrete parameter and security analysis so far.

**Our contributions.** We present a new KEM based on NTRU, called BAT[2]. Similar to Falcon signature, BAT uses $h = f^{-1}g \bmod q$ as the public key and its secret key is a trapdoor basis $\mathbf{B}_{f,g}$ with an additional ring element (for faster decapsulation). In addition, BAT shares the same leading design principle with Falcon, i.e. minimizing the communication size.

Our main improvement in BAT-KEM is a better decapsulation algorithm, which represents a major modification to the NTRU encryption scheme since its introduction 25 years ago. Instead of following the original NTRU, we modify it according to the GGH-Micciancio blueprint [35, 48]. The message $m$ is now encapsulated as $c = hm + e \bmod q$ where $h$ is the NTRU public key and $e$ is a small error. The decapsulation corresponds to applying Babai's nearest plane algorithm with the secret basis to decode the closest lattice point, and therefore recovering the message. Compared with other NTRU KEMs, we do not need the masking modulus $p$ to extract the message. Instead of multiplying only by $f$, we multiply by $F$ so that we get 2 linear equations in the 2 unknowns $e, m$.

However, Babai's nearest plane algorithm heavily relies on floating-point arithmetic, although most expensive calculation can be done in a pre-computation phase. To avoid floating-point arithmetic in the decapsulation, we replace the high-precision Gram-Schmidt vectors with integral approximations. Additionally, notice that $m$ and $e$ do not necessarily follow the identical distribution, hence we take into account their different sizes to optimize the decoding. We also use the Learning With Rounding (LWR) assumption [7, 15] in order to further reduce the size of the ciphertext. Our improved decoding algorithm can be used with a smaller modulus and dropping more bits, and thereby increases the security of the scheme *and* decreases the communication.

Overall, our KEM achieves very impressive performance. First of all, for the same NIST security level, BAT achieves the smallest communication size, namely "public key size + ciphertext size", among all current lattice cryposystems and even RSA cryptosystems. Secondly, the complexity of the code as well as its running time is asymmetric: while the key generation is complicated, the frequent key usage is quite efficient. Specifically, the encryption is very simple — essentially a ring multiplication —, and the decryption also boils down to a few ring additions and multiplications. Cheap daily operations make BAT particularly compatible with small devices. Thirdly, we can implement the whole scheme fully over integers, which is different from the case of Falcon. Our implementation is constant-time and uses some AVX2 optimizations. We can notice that BAT has performance comparable to Kyber while being more compact. Furthermore, it is comparable to SIKE p434 in size while being much more efficient. We gave the timing with x86 assembly optimization, while with the same level of optimization we did, the SIKE performance timing would be higher by one order of magnitude. We summarize the detailed comparisons with some well-known schemes in Table 2.

Finally, we explain in a simplistic way why the new decryption algorithm leads to smaller parameters. To correctly decrypt, our KEM needs $(fc \bmod q) = gm + fe$, i.e. $\|gm + fe\|_\infty < \frac{q}{2}$, while previous NTRU KEMs need $\|pgr + fm\|_\infty < \frac{q}{2}$. We also compare with ring-LWE-based KEMs. For a typical ring-LWE-based KEM, its secret key is $(f, g) \in \mathcal{R}^2$, public key is $(a, b = af + g) \in (\mathcal{R}/q\mathcal{R})^2$ and ciphertext is $(c_1 = ae_0 + e_1, c_2 = be_0 + e_2 + \lfloor \frac{q}{2} \rceil m) \in (\mathcal{R}/q\mathcal{R})^2$. The requirement for correct decryption is $\|e_0 g - e_1 f + e_2\|_\infty < \frac{q}{4}$. Suppose that $m, e, r, e_i$ are drawn from a distribution of standard deviation $\sigma_e$ and $f, g$ from a distribution of standard deviation $\sigma_f$. The coefficients of $gm + fe, pgr + fm, e_0 g - e_1 f + e_2$ are modeled as Gaussian. The comparison on parameter restrictions are summarized in Table 1. It can be seen that given $(n, \tau, \sigma_e, \sigma_f)$, BAT allows a smaller modulus $q$. Note that for fixed $(n, \sigma_e, \sigma_f)$, a smaller $q$ implies higher security.

[2] BAT stands for "Basis with Attractive Trapdoor".

**Table 1:** The parameter restrictions for correct decryption. The parameter $\tau$ is the tail-bound parameter determining the decryption failure rate.

| | Requirement for correct decryption |
|---|---|
| NTRU | $\tau\sigma_e\sigma_f\sqrt{(p^2+1)n} < \frac{q}{2}$ |
| Ring-LWE | $\tau\sigma_e\sigma_f\sqrt{2n} < \frac{q}{4}$ |
| BAT | $\tau\sigma_e\sigma_f\sqrt{2n} < \frac{q}{2}$ |

**Table 2:** Comparisons with other KEMs including NTRU-HRSS [41], NTTRU [47], Kyber [4], Saber [8], LAC [45], Round5 [5], ECC, RSA and SIKE [42]. Timings do not include random generation (from the operating system) or key derivation costs. Sizes for BAT and LW-BAT include an optional one-byte identifying header. The implementation of LW-BAT was not fully optimized with AVX2 opcodes.

| | Security | Ciphertext (bytes) | PK (bytes) | Keygen (kcycles) | Encaps (kcycles) | Decaps (kcycles) |
|---|---|---|---|---|---|---|
| **BAT** | 128 bits | 473 | 521 | $30.6 \times 10^3$ | 8.4 | 54.3 |
| **BAT** | 256 bits | 1006 | 1230 | $185.7 \times 10^3$ | 18.5 | 118.6 |
| **LW-BAT** | 80 bits | 203 | 225 | $23.6 \times 10^3$ | 55.7 | 248.0 |
| **NTRU-HRSS** | 128 bits | 1140 | 1140 | 220.3 | 34.6 | 65.0 |
| **NTTRU** | 128 bits | 1248 | 1248 | 6.4 | 6.1 | 7.8 |
| **Kyber** | 128 bits | 768 | 800 | 33.9 | 45.2 | 34.6 |
| **Kyber** | 256 bits | 1568 | 1568 | 73.5 | 97.3 | 79.1 |
| **Saber** | 128 bits | 736 | 672 | 45.2 | 62.2 | 62.6 |
| **Saber** | 256 bits | 1472 | 1312 | 126.2 | 153.8 | 155.7 |
| **LAC** | 128 bits | 712 | 544 | 59.6 | 89.1 | 140.2 |
| **LAC** | 256 bits | 1424 | 1056 | 135.8 | 208.0 | 359.2 |
| **Round5** | 128 bits | 620 | 461 | 46 | 68 | 95 |
| **Round5** | 256 bits | 1285 | 978 | 105 | 166 | 247 |
| **Round5-iot**[a] | 96 bits | 394 | 342 | 41 | 52 | 28 |
| **RSA 4096** | 128 bits | 512 | 512 | $2.19 \times 10^6$ | 212.1 | 13690 |
| **ECC** | 128 bits | 32 | 32 | 46 | 176 | 130 |
| **SIKE p434**[b] | 128 bits | 346 | 330 | $5.9 \times 10^3$ | $9.7 \times 10^3$ | $10.3 \times 10^3$ |
| **Compressed SIKE p434**[b] | 128 bits | 236 | 197 | $10.2 \times 10^3$ | $15.1 \times 10^3$ | $11.1 \times 10^3$ |

[a] Here Round5-iot is only IND-CPA secure rather than IND-CCA secure.
[b] SIKE and compressed-SIKE use x64 assembly optimizations.

**Comparison with Falcon.** BAT is similar in spirit to Falcon signature: they both achieve good compactness by using some nice NTRU trapdoor basis as the secret key. Nevertheless, some crucial distinctions exist between BAT and Falcon.

– At a high level, BAT and Falcon exploit their trapdoor to solve CVP (closest vector problem), but the used CVP algorithms are very different. Specifically, Falcon makes use of the KGPV Gaussian sampler [33] that is a *randomized* Babai's nearest plane algorithm. In contrast, BAT decrypts with a *deterministic* NTRU decoder that can be viewed as a hybrid of Babai's round-off and nearest plane algorithms.

– The algorithms of BAT are simpler than those of Falcon. On the one hand, the signing of Falcon relies on *high-precision Gaussian sampling*, but the encryption and

decryption of BAT only need *basic integer operations*. On the other, Falcon includes *many* high-precision intermediate values along with the trapdoor for faster signing, but BAT just adds *one* integral polynomial for faster decryption.

– The NTRU trapdoors of BAT and Falcon are generated in different ways. In fact, Falcon chooses its trapdoor for *smaller signatures*, which is equivalent to minimizing the maximal Gram-Schmidt norm of the trapdoor basis. As for BAT, the trapdoor is generated to *minimize the decryption failure*, and according to our new decoder, the distributions of the message and error will also affect the trapdoor generation (see Section 3 for more details).

**Related work.** Very recently, Chuengsatiansup et al. [18] propose some extensions of Falcon signature and NTRU encryption over Module-NTRU lattices. This allows more flexible parameters for NTRU-based cryptosystems. Our techniques are likely to apply to the Module-NTRU-based schemes as well.

In order to improve parameters, some schemes [17, 61] are built upon a variant of LWE in which the secret and error follow different distributions. Our work makes use of a similar idea. Yet the main difference is that our KEM follows a novel pattern which is essential to minimize the parameters.

**Roadmap.** We start in Section 2 with some preliminary materials. In Section 3, we introduce a new decoding algorithm that is the building block of our NTRU-based KEM. Section 4 presents our KEM in details. We give security proofs and estimates in Section 5 and implementation details in Section 6.

# 2 Preliminaries

## 2.1 Notations

We follow the setting $\mathbb{Z}_q = \{-(\lceil q/2 \rceil - 1), -(\lceil q/2 \rceil - 2), \cdots, q - \lceil q/2 \rceil\}$ and $(a \bmod q) \in \mathbb{Z}_q$ for any $a \in \mathbb{Z}$. Let ln (resp. log) denote the logarithm with base $e$ (resp. 2). For an integer $q > 0$, let $\lfloor a \rceil_q = \lfloor aq \rceil / q \in (1/q) \cdot \mathbb{Z}$ for $a \in \mathbb{R}$. For a real-valued function $f$ and a countable set $S$, we write $f(S) = \sum_{x \in S} f(x)$ assuming that this sum is absolutely convergent.

## 2.2 Linear algebra

Let $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}) \in \mathbb{Q}^{n \times n}$ of rank $n$. The Gram-Schmidt orthogonalization of $\mathbf{B}$ is $\mathbf{B} = \mathbf{B}^* \mathbf{U}$, where $\mathbf{U} \in \mathbb{Q}^{n \times n}$ is upper-triangular with 1 on its diagonal and $\mathbf{B}^* = (\mathbf{b}_0^*, \ldots, \mathbf{b}_{n-1}^*)$ is a matrix with pairwise orthogonal columns.

Let $\mathcal{R}_n = \mathbb{Z}[x]/(x^n + 1)$ with $n$ a power-of-2 and $\mathbb{K}_n = \mathbb{Q}[x]/(x^n + 1)$. We denote by $(\mathcal{R}_n \bmod q)$ the ring $\mathcal{R}_n / q\mathcal{R}_n$. When the context is clear, we may write $\mathcal{R}_n$ (resp. $\mathbb{K}_n$) as $\mathcal{R}$ (resp. $\mathbb{K}$). We identify $f = \sum_{i=0}^{n-1} f_i x^i \in \mathbb{K}_n$ with its coefficient vector $\mathbf{coef}(f) = (f_0, \cdots, f_{n-1})$. Let $\|f\| = \|\mathbf{coef}(f)\|$ and $\|f\|_\infty = \|\mathbf{coef}(f)\|_\infty$. We denote by $\overline{f}$ the conjugate of $f$, i.e. $f(x^{-1})$. Let $\mathbf{1} = \sum_{i=0}^{n-1} x^i \in \mathcal{R}$. The symbol $\lfloor \cdot \rceil_q$ is naturally generalized to $\mathbb{K}_n$ by applying it coefficient-wise.

## 2.3 Probability and statistics

Given a distribution $\chi$, we write $z \leftarrow \chi$ when the random variable $z$ is drawn from $\chi$. For $z \leftarrow \chi$, let $\mu[z]$ (resp. $\sigma[z]$) denote the expectation (resp. standard deviation) of $z$, and $\mu[\chi] := \mu[z]$ (resp. $\sigma[\chi] := \sigma[z]$). If $\mu[\chi] = 0$, then $\chi$ is called *centered*. For a random $a \in \mathbb{K}$, if all its coefficients independently follow a distribution $\chi$, then we call $a$ *iid-random*

*over* $\chi$. If $a \in \mathbb{K}$ is iid-random over $\chi$, we write $\sigma_a = \sigma[\chi]$ and $\mu_a = \mu[\chi]$. We call it *centered* when $\mu_a = 0$.

For a distribution $\chi$, we denote by $\mathsf{Sample}(\chi)$ the procedure of generating a random sample of $\chi$ and by $\mathsf{Sample}(\chi; \mathsf{seed})$ the sampling procedure with seed $\mathsf{seed}$. For a finite set $S$, let $U(S)$ be the uniform distribution over $S$. In particular, for a positive integer $k$, $\sigma[U(\mathbb{Z}_k)] = \sqrt{\frac{k^2-1}{12}}$ and $\mu[U(\mathbb{Z}_k)] = 0$ if $k$ is odd; otherwise $\mu[U(\mathbb{Z}_k)] = \frac{1}{2}$.

For $c \in \mathbb{R}$ and $\sigma > 0$, let $\rho_{\sigma,c}(x) = \exp\left(-\frac{(x-c)^2}{2\sigma^2}\right)$ be the one-dimensional Gaussian function with center $c$ and standard deviation $\sigma$. When $c = 0$, we just write $\rho_\sigma(x)$. The discrete Gaussian over integers with center $c$ and standard deviation $\sigma$ is defined by the probability function

$$D_{\mathbb{Z},\sigma,c}(x) = \frac{\rho_{\sigma,c}(x)}{\rho_{\sigma,c}(\mathbb{Z})}, \forall x \in \mathbb{Z}.$$

Let $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\mathrm{d}t$ be the error function. For a random variable $X$ following a normal distribution with mean 0 and variance $1/2$, $\mathrm{erf}(x)$ is the probability of $X$ in the range $[-x, x]$.

### 2.4 NTRU

Given $f, g \in \mathcal{R}$ such that $f$ is invertible modulo some $q \in \mathbb{Z}$, let $h = f^{-1}g \bmod q$. The NTRU lattice defined by $h$ is denoted by $\mathcal{L}_{h,q} = \{(u, v)^t \mid u = hv \bmod q\}$. Given $(f, g)$, one can compute $\begin{pmatrix} g & G \\ f & F \end{pmatrix} \in \mathcal{R}^{2\times 2}$ a basis of $\mathcal{L}_{h,q}$ by solving the NTRU equation $gF - fG = q$ [38, 53]. Fixing $(f, g)$, there are infinitely many such bases, whereas these bases have the same Gram-Schmidt norms. Hence, we simply write $\mathbf{B}_{f,g} = \begin{pmatrix} g & G \\ f & F \end{pmatrix} \in \mathcal{R}^{2\times 2}$.

While the public key of an NTRU-based scheme is $h$ itself, the secret key can have different forms. For most NTRU encryption schemes, the secret key is $(g, f)$ itself, i.e. one short vector of $\mathcal{L}_{h,q}$. But for some other applications, e.g. signature and IBE, the secret key is $\mathbf{B}_{f,g}$ called an NTRU *trapdoor basis*. Falcon [30] is a representative example. Falcon is an NTRU-based signature following the GPV hash-and-sign framework [33]. To sign a message $m$, the signer computes a pair of short polynomials $(s_1, s_2)$ such that $s_1 + hs_2 = \mathsf{Hash}(m)$. This procedure is accomplished by lattice Gaussian sampling with $\mathbf{B}_{f,g}$ and the length of the signature $(s_1, s_2)$ depends on the sampled Gaussian width. The Gaussian sampler of Falcon is a fast Fourier variant [29] of the KGPV sampler [33], hence the signature size is proportional to the maximal Gram-Schmidt norm of $\mathbf{B}_{f,g}$. For optimal parameters, Falcon generates $(f, g)$ such that $\|(f, g)\| \approx 1.17\sqrt{q}$ as per [27].

## 3 A New NTRU Decoder

In this section we present a new NTRU decoding algorithm that is the key component of our KEM. In the context of NTRU, the code words are $hs + e \bmod q$ where $h$ is the public key and $s, e$ are small polynomials. The decoding process recovers $(s, e)$ with an NTRU trapdoor. An ideal decoder is supposed to satisfy:

1. All operations are simple and efficient; no high-precision arithmetic is needed.
2. The decoding distance is large, i.e. being able to recover large errors $(s, e)$. Note that for our KEM, larger errors correspond to higher security level.

There have been two famed decoding algorithms due to Babai [6]: Babai's round-off algorithm ($\mathsf{RO}$ for short) and Babai's nearest plane algorithm ($\mathsf{NP}$ for short). They have respective pros and cons. The $\mathsf{RO}$ algorithm outperforms $\mathsf{NP}$ in efficiency and simplicity.

In addition, RO is particularly compatible with $q$-ary lattices: all operations are over $\mathbb{Z}_q$. By contrast, NP is capable of decoding larger errors in both the worst and average cases [54]. Yet the principal drawback of NP is its reliance on high-precision arithmetic.

Our decoder improves on both RO and NP. First, it is able to tackle a larger decoding distance than RO. Second, while complicated computations are still required, all involved algorithms can be implemented using fixed-point arithmetic in practice, which outperforms NP. Meanwhile these expensive computations can be done in the pre-computation and therefore do not affect the decoding efficiency. With an auxiliary integer vector, our algorithm achieves the same efficiency as RO and totally performs over integers. To optimize the decoding, our algorithm also takes into account the distributions of $s$ and $e$.

## 3.1 Babai's algorithms for NTRU

For better contrast, we first recall RO and NP briefly in the NTRU setting. Let $h \in (\mathcal{R} \bmod q)$ be the public key and $\mathbf{B}_{f,g} = \begin{pmatrix} g & G \\ f & F \end{pmatrix} \in \mathcal{R}^{2\times 2}$ be the trapdoor basis. In later discussion, we shall treat $\mathcal{L}_{h,q}$ as a $\mathcal{R}$-*module* of rank 2 rather than a $\mathbb{Z}$-*module* of rank $2n$.

The application of RO related to NTRU dates back to NTRUSign [38]. Given $c = hs + e \in (\mathcal{R} \bmod q)$, we have $\begin{pmatrix} c \\ 0 \end{pmatrix} \in \mathcal{L}_{h,q} + \begin{pmatrix} e \\ -s \end{pmatrix}$. The RO algorithm computes $\begin{pmatrix} e' \\ -s' \end{pmatrix} = \begin{pmatrix} c \\ 0 \end{pmatrix} - \mathbf{B}_{f,g} \left\lfloor \mathbf{B}_{f,g}^{-1} \begin{pmatrix} c \\ 0 \end{pmatrix} \right\rceil$ as $\begin{pmatrix} e \\ -s \end{pmatrix}$. As $\mathbf{B}_{f,g}^{-1} = \frac{1}{q} \begin{pmatrix} F & -G \\ -f & g \end{pmatrix}$, a correct decoding such that $(e', s') = (e, s)$ follows if

$$\begin{pmatrix} F & -G \\ -f & g \end{pmatrix} \begin{pmatrix} e \\ -s \end{pmatrix} = \begin{pmatrix} Fc \bmod q \\ -fc \bmod q \end{pmatrix}.$$

This is equivalent to

$$\max\{\|fe + gs\|_\infty, \|Fe + Gs\|_\infty\} \leq \frac{q}{2}.$$

The NP algorithm is more complicated. It involves the Gram-Schmidt orthogonal basis $\mathbf{B}_{f,g}^* = \begin{pmatrix} g & G^* = G - vg \\ f & F^* = F - vf \end{pmatrix}$ where $v = \frac{F\overline{f} + G\overline{g}}{f\overline{f} + g\overline{g}}$. A correct decoding follows if $\max\{\|fe + gs\|_\infty, \|F^*e + G^*s\|_\infty\} \leq \frac{q}{2}$.

In some applications [27, 30], the trapdoor basis is optimal with respect to Gram-Schmidt norms: $\|(g, f)\| \approx \|(G^*, F^*)\|$. However, $\|(g, f)\|$ and $\|(G, F)\|$ are not so close: $\|(G, F)\| \approx \sqrt{\frac{n}{12}} \cdot \|(g, f)\|$ [38]. As a consequence, $\|(e, s)\|$ is dominated by the large $\|(G, F)\|$ in the RO algorithm but by the small $\|(g, f)\|$ in the NP algorithm, which leads to a gap of $O(\sqrt{n})$.

## 3.2 Our decoding algorithm for NTRU

As shown in Section 3.1, RO boils down to solving two linear equations over $\mathcal{R}$ (without modular reduction). To enlarge its decoding range, we hope to replace the large $(G, F)$ with some small vector $(G', F')$ of size $\approx \|(g, f)\|$. A natural candidate is $(G^*, F^*) = (G - vg, F - vf)$ with $v = \frac{F\overline{f} + G\overline{g}}{f\overline{f} + g\overline{g}}$ as in NP. However, if we want to work with $(G^*, F^*)$ directly, we have to resort to high-precision arithmetic. To overcome the precision issue, we choose $(G', F') = (G - g\lfloor v \rceil_{q'}, F - f\lfloor v \rceil_{q'}) \in (1/q')\mathcal{R}^2$. When $q'$ is sufficiently large, $(G', F')$ converges to $(G^*, F^*)$ whose norm is about $\|(g, f)\|$. In practice, a moderate $q'$ suffices to significantly improve the decoding.

We further refine our decoder as per the distributions of $s$ and $e$. We focus on the common case where both $s$ and $e$ are iid-random over some publicly known distributions $\chi_s$ and $\chi_e$. In practice, $\chi_s$ and $\chi_e$ are not necessarily same even close, which may cause a gap

between the sizes of $s$ and $e$. For example, when $\|s\| \ll \|e\|$, we expect a better decoding by using a basis $\begin{pmatrix} g' & G' \\ f' & F' \end{pmatrix}$ with $\|g'\| > \|g\| \approx \|f\| > \|f'\|$ and $\|G'\| > \|G\| \approx \|F\| > \|F'\|$. To this end, we introduce a parameter $\gamma$, by default $\gamma = \sigma_e/\sigma_s$, to compute the optimal decoding basis. Moreover, $\chi_s$ and $\chi_e$ do not have to be centered neither, e.g. $\chi_s = U(\mathbb{Z}_2)$. For given $(f, g)$, non-centered $s$ and $e$ lead to a non-zero average of $fe + gs$. Therefore, we also consider the impact of $\mu_s$ and $\mu_e$ during decoding. Here we assume $\mu_s, \mu_e \in \frac{1}{Q} \cdot \mathbb{Z}$ for some $Q \in \mathbb{N}$, which is indeed the case of our later schemes.

The decoding algorithm consists in two steps: (1) computing the auxiliary polynomial $w$ and (2) recovering $(s, e)$. They are illustrated in Algorithms 3.1 and 3.2 respectively. Notably, both algorithms can be fully implemented over the integers. In Algorithm 3.1, the computation of $v$ consists in one polynomial division, but the final output is actually an integral approximation of $q'v$, which can be computed with fixed-point values. More details are presented in Section 6.1.

**Algorithm 3.1** ComputeVec

**Input:** a trapdoor basis $\mathbf{B}_{f,g} = \begin{pmatrix} g & G \\ f & F \end{pmatrix} \in \mathcal{R}^{2\times 2}$, $q' \in \mathbb{N}$ and $\gamma > 0$;

**Output:** $w \in \mathcal{R}$

1: $v \leftarrow \frac{\gamma^2 F\overline{f} + G\overline{g}}{\gamma^2 f\overline{f} + g\overline{g}}$
2: return $w = q'\lfloor v \rceil_{q'}$

**Algorithm 3.2** Decode

**Input:** a trapdoor basis $\mathbf{B}_{f,g} = \begin{pmatrix} g & G \\ f & F \end{pmatrix} \in \mathcal{R}^{2\times 2}$, $w \in \mathcal{R}$, $q, q', Q \in \mathbb{N}$,
$c = (hs + e \bmod q) \in \mathcal{R}$ with small $(e, s)$ and $\mu_e, \mu_s \in \frac{1}{Q} \cdot \mathbb{Z}$

**Output:** $(e, s) \in \mathcal{R}^2$

1: $(G_d, F_d) \leftarrow (q'G - gw, q'F - fw)$
2: $c' \leftarrow (Qfc - f(Q\mu_e\mathbf{1}) - g(Q\mu_s\mathbf{1}) \bmod qQ)$
3: $c'' \leftarrow (q'QFc - F(q'Q\mu_e\mathbf{1}) - G(q'Q\mu_s\mathbf{1}) - c'w \bmod qq'Q)$
4: solving $\begin{pmatrix} e' \\ s' \end{pmatrix}$ from $\begin{pmatrix} f & g \\ F_d & G_d \end{pmatrix} \begin{pmatrix} e' \\ s' \end{pmatrix} = \frac{1}{Q} \cdot \begin{pmatrix} c' \\ c'' \end{pmatrix}$
5: return $(e = e' + \mu_e\mathbf{1}, s = s' + \mu_s\mathbf{1})$

Theorem 1 gives the probability of correct decoding of Algorithm 3.2 and its proof is in Appendix A.

**Theorem 1.** *Let $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ with $n$ a power-of-2 and $q \in \mathbb{N}$. Let $f, g \in \mathcal{R}$ be iid-random over $D_{\mathbb{Z},\sigma_f}$ and $h = f^{-1}g \bmod q$. Let $\mathbf{B}_{f,g} = \begin{pmatrix} g & G \\ f & F \end{pmatrix} \in \mathcal{R}^{2\times 2}$ be an NTRU basis. Let $s, e \in \mathcal{R}$ be iid-random over $\chi_s$ and $\chi_e$ respectively, and $\mu_e, \mu_s \in \frac{1}{Q} \cdot \mathbb{Z}$ for some $Q \in \mathbb{N}$. Let $\gamma = \sigma_e/\sigma_s$, $q' \in \mathbb{N}$ and $w = \mathsf{ComputeVec}(\mathbf{B}_{f,g}, q', \gamma)$. Let $I(\gamma) = \frac{2\ln(\gamma)}{\gamma^2 - 1}$ for $\gamma \neq 1$ and $I(1) = 1$. Let $\sigma_1 = \sqrt{n}\sigma_f\sigma_s\sqrt{\gamma^2 + 1}$, $\sigma_2 = \sigma_s\left(\frac{q\gamma}{\sqrt{n}\cdot\sigma_f}\sqrt{I(\gamma)} + \frac{n^{1.5}\sqrt{\gamma^2+1}}{2q'}\sigma_f\right)$. Let $\tau = \min\left\{\frac{q}{2\sqrt{2}\sigma_1}, \frac{q}{2\sqrt{2}\sigma_2}\right\}$. Then the probability of $\mathsf{Decode}(\mathbf{B}_{f,g}, w, q, q', Q, (hs+e \bmod q)) = (e, s)$ is heuristically estimated at least $1 - 2n \cdot (1 - \mathrm{erf}(\tau))$ over the randomness of $s$ and $e$.*

### 3.3 Decoding failure rate

Theorem 1 gives a heuristic estimate for $\mu[P(\mathbf{B}_{f,g}, w, \chi_s, \chi_e)]$ (over the randomness of $(f, g)$) where

$$P(\mathbf{B}_{f,g}, w, \chi_s, \chi_e) = \Pr[\mathsf{Decode}(\mathbf{B}_{f,g}, w, q, q', Q, (hs + e \bmod q)) \neq (e, s) \mid e \leftarrow \chi_e, s \leftarrow \chi_s]$$

is the decoding failure rate for given $(\mathbf{B}_{f,g}, w)$. In fact, it is hard to numerically compute $\mu[P(\mathbf{B}_{f,g}, w, \chi_s, \chi_e)]$, since the distribution of $(G', F')$ (defined in Lemma 1) is complicated. But it is easy to numerically compute $P(\mathbf{B}_{f,g}, w, \chi_s, \chi_e)$ given $(\mathbf{B}_{f,g}, w)$.

We experimentally calculate $P(\mathbf{B}_{f,g}, w, \chi_s, \chi_e)$ for some $(\mathbf{B}_{f,g}, w)$ generated with the suggested parameters (see Tables 3 and 4). The failure probabilities are smaller than for a naive Gaussian model, and significantly so for a ring dimension of 256.

## 4 BAT KEM

In this section, we present a KEM scheme, called BAT, constructed following the aforementioned encode/decode paradigm. Its secret key is an NTRU trapdoor basis as in the Falcon signature. As a consequence, some codes for Falcon implementation can be reused.

BAT permits very compact parameters. Specifically, the modulus $q$ is greatly reduced in contrast to Falcon. More remarkably, the ciphertext is well compressed: each coefficient requires only less than one byte of storage.

### 4.1 Algorithm description

Prior to the description of our KEM, we first present the underlying public key encryption. It is specified by the following parameters:

- $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ with $n = 2^l$.
- $q = bk + 1$ with $b, k \in \mathbb{N}$. Note that $b$ determines the size of each ciphertext coefficient and $k$ determines the decoding distance.
- $q' \in \mathbb{N}$ is used to control the decryption failure rate.

At a high level, our idea is to build an encryption scheme upon a one-way trapdoor function. Indeed, for a pseudorandom public key $h$, the function $F(s, e) = hs + e \bmod q$ is one-way under the Ring-LWE assumption, but one can invert it with the trapdoor $\mathbf{B}_{f,g}$ as shown in Section 3. In our scheme, the encryption consists in computing $c = F(s, e)$ and the decryption in recovering $s$ by inverting $F(s, e)$.

The key generation is shown in Algorithm 4.1. The first step is to generate an NTRU trapdoor basis $\mathbf{B}_{f,g}$ along with the public key $h$. This is similar to the Falcon key generation, but the size of the secret key is changed. The second step pre-computes an auxiliary vector $w$. As explained in Section 3, $w$ is used for decoding a larger error while avoiding floating-point arithmetic in the decapsulation. We include it as a part of the secret key. Note that Falcon key generation also pre-computes the Falcon tree for signing, but that computation is useless in our scheme.

The encryption algorithm is described in Algorithm 4.2. The message space is $\mathcal{M} = \{0, 1\}^\lambda$ where $\lambda$ denotes the claimed security level. To achieve the IND-CPA security, we compute the trapdoor function on an ephemeral $s$ and then mask the message $m$ with hashed $s$. For better compactness, we replace $F(s, e)$ with $F(s) = \left\lfloor \frac{(hs \bmod q)}{k} \right\rceil$ and thus use Ring-LWR as the hardness assumption. It is easy to see that the storage of a ciphertext is $n \log b + \lambda$ bits. The decryption stems from the decoding algorithm in Section 3. The formal description is provided in Algorithm 4.3.

**Algorithm 4.1** $\mathsf{KeyGen}_{\mathsf{Enc}}$

**Input:** the ring $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$, integers $q, k, q' \in \mathbb{N}$

**Output:** public key $h \in \mathcal{R}$, secret key $(\mathbf{B}_{f,g}, w) \in \mathcal{R}^{2\times 2} \times \mathcal{R}$ where $\mathbf{B}_{f,g} = \begin{pmatrix} g & G \\ f & F \end{pmatrix}$

$\gamma \leftarrow \sqrt{\frac{k^2-1}{3}}$
$\sigma_f \leftarrow \sqrt{\frac{q\gamma}{n}\sqrt{\frac{I(\gamma)}{\gamma^2+1}}}$ where $I(\gamma)$ is defined as in Theorem 1
$f, g \leftarrow D_{\mathcal{R},\sigma_f}$
**if** $f$ or $g$ is not invertible in $(\mathcal{R} \bmod q)$ **then**
    restart
**end if**
**if** $\|(g, \gamma f)\| > \sqrt{n}\sigma_f\sqrt{\gamma^2+1}$ **then**
    restart
**end if**
$h \leftarrow f^{-1}g \bmod q$
compute $(F, G) \in \mathcal{R}^2$ such that $gF - Gf = q$
$\mathbf{B}_{f,g} \leftarrow \begin{pmatrix} g & G \\ f & F \end{pmatrix} \in \mathcal{R}^{2\times 2}$
$w \leftarrow \mathsf{ComputeVec}(\mathbf{B}_{f,g}, q', \gamma)$
$(G_d, F_d) \leftarrow (q'G - gw, q'F - fw)$
**if** $\|(G_d, \gamma F_d)\| > q'\left(\frac{q\gamma}{\sqrt{n}\cdot\sigma_f}\sqrt{I(\gamma)} + \frac{n^{1.5}\sqrt{\gamma^2+1}}{2q'}\sigma_f\right)$ **then**
    restart
**end if**
return $(h, (\mathbf{B}_{f,g}, w))$

**Algorithm 4.2** $\mathsf{Encrypt}$

**Input:** public key $h \in \mathcal{R}$, integers $q, k \in \mathbb{N}$, message $m \in \mathcal{M}$, seed $\mathsf{seed}$

**Output:** ciphertext $(c_1, c_2)$

$s \leftarrow \mathsf{Sample}(U(\mathcal{R} \bmod 2); \mathsf{seed})$
$c_1 \leftarrow \left\lfloor \frac{(hs \bmod q)}{k} \right\rceil$
$e \leftarrow (hs \bmod q) - kc_1$, $\gamma \leftarrow \sqrt{\frac{k^2-1}{3}}$
**if** $\|(\gamma s, e)\| > 1.08\sqrt{\frac{n(k^2-1)}{6}}$ **then**
    return $\perp$
**end if**
$c_2 \leftarrow \mathsf{Hash}_m(s) \bigoplus m$ where $\mathsf{Hash}_m$ is some hash with range $\mathcal{M}$
return $(c_1, c_2)$

**Algorithm 4.3** Decrypt

**Input:** ciphertext $(c_1, c_2)$, public key $h \in \mathcal{R}$, secret key $(\mathbf{B}_{f,g}, w)$, integers $q, k, q' \in \mathbb{N}$
**Output:** message $m$

$c' \leftarrow c_1 k$
$(e, s) \leftarrow \mathsf{Decode}\,(\mathbf{B}_{f,g}, w, q, q', 2, c')$
$\gamma \leftarrow \sqrt{\frac{k^2-1}{3}}$
**if** $\|(\gamma s, e)\| > 1.08\sqrt{\frac{n(k^2-1)}{6}}$ **then**
    return $\perp$
**end if**
**if** $c_1 = \left\lfloor \frac{(hs \bmod q)}{k} \right\rceil$ **then**
    return $\mathsf{Hash}_m(s) \bigoplus c_2$
**else**
    return $\perp$
**end if**

By some standard techniques [31, 25], an IND-CCA secure KEM immediately follows from our IND-CPA encryption. Algorithms 4.4, 4.5 and 4.6 describe the key generation, encapsulation and decapsulation algorithms respectively, in which $\mathcal{S}$ (resp. $\mathcal{K}$) is the set of seed (resp. shared key). The detailed security arguments are given in Section 5.2.

Our KEM tolerates a small decryption failure rate for better performance, as many current lattice-based KEMs. Yet some works [32, 22, 37, 23] also show the impact of decryption failures on security. We bound the size of the error used to 1.08 its average to limit the impact of precomputed messages on the decryption failure rates: the exponent may be reduced by 20 % in the worst case.

**Algorithm 4.4** $\mathsf{KeyGen}_{\mathsf{Encap}}$

**Input:** the ring $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$, integers $q, k, q' \in \mathbb{N}$
**Output:** public key $h \in \mathcal{R}$, secret key $(\mathbf{B}_{f,g}, w, r) \in \mathcal{R}^{2\times 2} \times \mathcal{R} \times \mathcal{R}$

$r \leftarrow \mathsf{Sample}(U(\mathcal{M}))$
$(h, (\mathbf{B}_{f,g}, w)) \leftarrow \mathsf{KeyGen}_{\mathsf{Enc}}(\mathcal{R}, q, k, q')$
return $(h, (\mathbf{B}_{f,g}, w, r))$

**Algorithm 4.5** Encapsulate

**Input:** public key $h \in \mathcal{R}$, integers $q, k \in \mathbb{N}$
**Output:** ciphertext $c$, key $K$

$m \leftarrow \mathsf{Sample}(U(\mathcal{M}))$
$(c_1, c_2) \leftarrow \mathsf{Encrypt}(h, q, k, m, \mathsf{Hash}_s(m))$ where $\mathsf{Hash}_s$ is some hash with range $\mathcal{S}$
**if** $(c_1, c_2) = \perp$ **then**
    restart
**end if**
$K \leftarrow \mathsf{Hash}_k(m, c_1, c_2)$ where $\mathsf{Hash}_k$ is some hash with range $\mathcal{K}$
return $((c_1, c_2), K)$

## 4.2 Parameter selection

We keep the same ring $\mathcal{R}$ as Falcon but choose a much smaller modulus $q$. Indeed, a smaller modulus forbids the applications of a complete NTT, nevertheless similar techniques [46, 47] still allow a very fast polynomial multiplication. For security, a smaller $q$ implies a smaller

**Algorithm 4.6** Decapsulate

**Input:** ciphertext $(c_1, c_2)$, public key $h \in \mathcal{R}$, secret key $(\mathbf{B}_{f,g}, w, r)$, integers $q, k, q' \in \mathbb{N}$
**Output:** key $K$

$m' \leftarrow \mathsf{Decrypt}((c_1, c_2), h, (\mathbf{B}_{f,g}, w), q, k, q')$
**if** $m' \neq \perp$ and $(c_1, c_2) = \mathsf{Encrypt}(h, q, k, m', \mathsf{Hash}_s(m'))$ **then**
  return $K \leftarrow \mathsf{Hash}_k(m', c_1, c_2)$
**else**
  return $K \leftarrow \mathsf{F}(r, c_1, c_2)$ where $\mathsf{F}$ is some PRF with range $\mathcal{K}$
**end if**
return $K$

standard deviation of the secret key distribution and then less entropy of the secret key. But such loss does not reduce much the concrete security level.

A notable modification exists in key generation. In Falcon, $(f, g)$ is sampled to make $\|(g, f)\| \approx \|(G^*, F^*)\|$ where $(G^*, F^*)$ is the Gram-Schmidt orthogonalization of $(G, F)$ in $\mathbf{B}_{f,g}$. However, in BAT, we choose $\sigma_f$ satisfying $\sqrt{n}\sigma_f\sqrt{\gamma^2+1} = \frac{q\gamma}{\sqrt{n}\cdot\sigma_f}\sqrt{I(\gamma)}$. This gives rise to a nearly optimal decryption failure rate according to Theorem 1. In particular, when $\gamma = 1$, the $\sigma_f$ we use also makes $\|(g, f)\| \approx \|(G^*, F^*)\|$ as the case of Falcon. But for this case, $\sigma$ used by BAT is different from that by Falcon, which is explained in Remark 4.

Let us recall that

$$c' = ck = hs - k\left(\frac{hs}{k} - \left\lfloor\frac{(hs \bmod q)}{k}\right\rceil\right) := hs + e \bmod q.$$

We model $e$ drawn from $U(\mathbb{Z}_k^n)$ and thus[3] $\sigma_e = \sqrt{\frac{k^2-1}{12}}$. As $\sigma_s = \frac{1}{2}$, it follows that $\gamma = \sigma_e/\sigma_s = \sqrt{\frac{k^2-1}{3}}$. The decryption failure rate is equal to the probability of incorrect decoding. According to Theorem 1, the decryption failure rate is heuristically bounded by $2n \cdot (1 - \mathrm{erf}(\tau))$ for some $\tau$. We also exactly computed the decryption failure rate for 100 keys: the standard deviation of the logarithm of the rate over the secret key distribution is around 8, and the exact values are even smaller than their heuristic estimates. Therefore we present the *exact* values for the decryption failure rate and the tail-bound parameter $\tau$ for the *heuristic* estimates.

Table 3 shows the suggested parameters.

**Table 3:** Suggested parameters for BAT.

| Security | $n$ | $(b, k, q)$ | $\sigma_f$ | $q'$ | $\tau$ | Decryption Failure |
|---|---|---|---|---|---|---|
| 128 bits | 512 | $(128, 2, 257)$ | 0.596 | 64513 | 9.46 | $2^{-146.7}$ |
| 256 bits | 1024 | $(192, 4, 769)$ | 0.659 | 64513 | 10.42 | $2^{-166.7}$ |

**The parameter set for lightweight BAT.** We further suggest one more parameter set particularly aiming at a lower security level, say 80 bits of security, which may be of interest for some lightweight use-cases. We call this lightweight variant LW-BAT. In LW-BAT, the degree $n$ is only 256; for better compactness the modulus $q$ does not support NTT anymore. We choose a relatively high decryption failure rate $2^{-71.9}$, but it should be sufficient for lightweight applications, e.g. IoT: for the Round5 IoT parameters [5], the decryption failure rate is $2^{-41}$ even larger than ours. Table 4 summarizes the concrete parameter set.

[3]The actual distribution is within statistical distance $1/q$ of $U(\mathbb{Z}_k^n)$.

**Table 4:** Suggested parameters for LW-BAT.

| Security | $n$ | $(b, k, q)$ | $\sigma_f$ | $q'$ | $\tau$ | Decryption Failure |
|---|---|---|---|---|---|---|
| 80 bits | 256 | $(64, 2, 128)$ | 0.595 | 64513 | 6.71 | $2^{-71.9}$ |

# 5 Security

We now report on the security of BAT. First, we demonstrate the IND-CCA security of our KEM under some hardness assumptions. Then we estimate the concrete security according to the best known attacks.

## 5.1 Assumptions

**The (decision) NTRU assumption.** Let $\mathcal{R}_q^\times$ be the set of invertible elements in $\mathcal{R}/q\mathcal{R}$. Let $\chi$ be some distribution over $\mathcal{R}_q^\times$. The advantage of adversary $\mathsf{A}$ in solving the decision NTRU problem $\mathrm{NTRU}_{\mathcal{R},q,\chi}$ is

$$\mathsf{Adv}^{\mathrm{NTRU}}_{\mathcal{R},q,\chi}(\mathsf{A}) = \left|\Pr[b = 1 \mid f, g \hookleftarrow \chi; b \leftarrow \mathsf{A}(f^{-1}g \bmod q)] - \Pr[b = 1 \mid u \hookleftarrow U(\mathcal{R}_q^\times); b \leftarrow \mathsf{A}(u)]\right|.$$

In our case, $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ and $\chi$ is the distribution of the secret key $f$ and $g$.

*Remark* 1. There are some researches on the hardness of the decision NTRU assumption over $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$. Notably, as shown in [59], when $\chi$ is a discrete Gaussian of standard deviation $\sigma = \widetilde{\omega}(n\sqrt{q})$, the ratio of $f$ and $g$ is *statistically* indistinguishable from uniform, which gives a firm theoretical grounding. The decision NTRU assumption with a narrow distribution $\chi$ is also closely related to Falcon [30] and sometimes referred to as the Decisional Small Polynomial Ratio (DSPR) assumption [44, 12].

**The (search) Ring-LWR assumption.** Let $\chi$ be some distribution over $\mathcal{R}$. The advantage of adversary $\mathsf{A}$ in solving the search Ring-LWR problem $\mathrm{RLWR}_{\mathcal{R},q,k,\chi}$ is

$$\mathsf{Adv}^{\mathrm{RLWR}}_{\mathcal{R},q,k,\chi}(\mathsf{A}) = \Pr_{a \hookleftarrow U(\mathcal{R}_q^\times), s \hookleftarrow \chi}\left[\mathsf{A}\left(a, \left\lfloor \frac{(as \bmod q)}{k} \right\rceil\right) = s\right].$$

In our case, $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ and $\chi = U(\mathcal{R} \bmod 2)$.

*Remark* 2. The theoretical foundation of the search Ring-LWR assumption is developed in [7, 11, 15]. There are also some practical schemes, e.g. Lizard [17] and Saber [8], using the Ring-LWR over $\mathbb{Z}[x]/(x^n + 1)$ or its module variant as their hardness assumption. Indeed, the provable hardness of Ring-LWR with a binary secret $s$ remains open. Yet this would not weaken the concrete security especially when $q$ is relatively small (as in our case).

## 5.2 KEM security

The security notion we prove for BAT is IND-CCA security (indistinguishability against chosen-ciphertext attacks). To this end, we first note that the underlying encryption (Algorithms 4.2 and 4.3) is IND-CPA secure (indistinguishability against chosen-plaintext attacks) under the assumptions in Section 5.1.

**Theorem 2.** *Let $\Pi$ be the public key encryption scheme defined by Algorithms 4.1, 4.2 and 4.3. Let $\mathsf{Hash}_m$ be modeled as a random function. For any adversary $\mathsf{A}$, there exist adversaries $\mathsf{A}_1$ and $\mathsf{A}_2$ of roughly the same running time as that of $\mathsf{A}$ such that*

$$\mathit{Adv}^{\mathit{IND\text{-}CPA}}_{\Pi}(\mathsf{A}) \leq \mathit{Adv}^{\mathit{NTRU}}_{\mathcal{R},q,\chi_1}(\mathsf{A}_1) + \mathit{Adv}^{\mathit{RLWR}}_{\mathcal{R},q,k,\chi_2}(\mathsf{A}_2)$$

*where $\chi_1$ is the distribution of the secret key $(f, g)$ and $\chi_2 = U(\mathcal{R} \bmod 2)$.*

*Proof.* We prove via a sequence of games. Let $\Pr[W_i]$ denote the probability of the adversary winning Game $i$.

Game 0 is the classical bit-guessing version of the IND-CPA security game, thus $\Pr[W_0] = \frac{1}{2} + \mathsf{Adv}^{\text{IND-CPA}}_{\Pi}(\mathsf{A})$. The challenger in Game 0 proceeds as follows:

| | | |
|---|---|---|
| *Initialization*: | (1) | $(pk = h, sk) \leftarrow \mathsf{KeyGen}_{\mathsf{Enc}}()$ |
| | (2) | $s \leftarrow U(\mathcal{R} \bmod 2); c_1 \leftarrow \left\lfloor \frac{(hs \bmod q)}{k} \right\rceil$ |
| | | initialize an empty associative array $Map : \mathcal{R} \bmod 2 \to \mathcal{M}$ |
| | (3) | $v \leftarrow U(\mathcal{M})$; $b \leftarrow U(\{0,1\})$ |
| | (4) | $Map[s] \leftarrow v$ |
| | | send $pk$ to $\mathsf{A}$ |
| *Encryption query on* $(m_0, m_1) \in \mathcal{M}^2$*:* | (5) | $c_2 \leftarrow v \bigoplus m_b$; send $(c_1, c_2)$ to $\mathsf{A}$ |
| *Random oracle query on* $s' \in \mathcal{R} \bmod 2$: | (6) | if $s' \notin Domain(Map)$, then $Map[s'] \leftarrow U(\mathcal{M})$ |
| | | send $Map[s']$ to $\mathsf{A}$ |

Game 1 and Game 0 only differ in the generation of $pk$: line (1) in Game 0 is replaced by "$pk = h \leftarrow U(\mathcal{R}_q^{\times})$". We define $\mathsf{A}_1$ for the decision NTRU problem. Upon input $h \in \mathcal{R}_q^{\times}$, $\mathsf{A}_1$ plays the challenger in Game 0 except for setting $pk = h$ in line (1). Then it is easy to see that

$$\mathsf{Adv}^{\text{NTRU}}_{\mathcal{R},q,\chi_1}(\mathsf{A}_1) = |\Pr[W_0] - \Pr[W_1]|.$$

Game 2 is precisely the same as Game 1, except that we delete line (4). Let $E$ be the event that the adversary queries the random oracle at the point $s$ in Game 2. Clearly, Game 1 and Game 2 proceed identically unless $E$ occurs. Therefore,

$$|\Pr[W_1] - \Pr[W_2]| \leq \Pr[E].$$

If $E$ happens, then one of the random oracle queries made by the adversary is $s$ such that $c_1 = \left\lfloor \frac{(hs \bmod q)}{k} \right\rceil$, which gives a solution to the search Ring-LWR problem with input $(h, c_1)$. Precisely, we define $\mathsf{A}_2$ for the search Ring-LWR problem. Upon input $(h, c)$, $\mathsf{A}_2$ plays the challenger in Game 2 except for setting $pk = h$ and $c_1 = c$. Additionally, when $\mathsf{A}_2$ terminates, if there is some $s' \in Domain(Map)$ such that $c = \left\lfloor \frac{(hs' \bmod q)}{k} \right\rceil$, then $\mathsf{A}_2$ outputs $s'$ otherwise outputs "failure". By definition, $E$ happens if and only if $s \in Domain(Map)$, then we have

$$\Pr[E] = \mathsf{Adv}^{\text{RLWR}}_{\mathcal{R},q,k,\chi_2}(\mathsf{A}_2).$$

In Game 2, $v$ is uniformly random drawn from $\mathcal{M}$ and independent of other intermediate variables. Therefore, $(m_0, m_1)$ is completely indistinguishable, i.e. $\Pr[W_2] = \frac{1}{2}$. It immediately follows that

$$\begin{aligned}
\mathsf{Adv}^{\text{IND-CPA}}_{\Pi}(\mathsf{A}) = \Pr[W_0] - \frac{1}{2} &\leq \mathsf{Adv}^{\text{NTRU}}_{\mathcal{R},q,\chi_1}(\mathsf{A}_1) + \Pr[W_1] - \frac{1}{2} \\
&\leq \mathsf{Adv}^{\text{NTRU}}_{\mathcal{R},q,\chi_1}(\mathsf{A}_1) + \Pr[W_2] - \frac{1}{2} + \Pr[E] \\
&\leq \mathsf{Adv}^{\text{NTRU}}_{\mathcal{R},q,\chi_1}(\mathsf{A}_1) + \mathsf{Adv}^{\text{RLWR}}_{\mathcal{R},q,k,\chi_2}(\mathsf{A}_2).
\end{aligned}$$

□

The BAT KEM is obtained via applying a tweaked Fujisaki-Okamoto transform [10, 43] to the IND-CPA secure encryption. Theorem 3 gives the concrete security statement of the IND-CCA security of BAT when $\mathsf{Hash}_m, \mathsf{Hash}_s$ and $\mathsf{Hash}_k$ are modeled as quantum random oracles.

**Theorem 3.** *[[43], Corollary 4.7] Let $\Pi$ be the public key encryption scheme defined by Algorithms 4.1, 4.2 and 4.3. Let $\mathcal{M}$, $\mathcal{C}$, $\mathcal{K}$ and $\mathcal{S}$ be the message, ciphertext, key and seed spaces of $\Pi$. Let $\Pi_{CCA}$ be the IND-CCA secure KEM defined by Algorithms 4.4, 4.5 and 4.6. Let $\mathsf{Hash}_m : \mathcal{R} \to \mathcal{M}$, $\mathsf{Hash}_s : \mathcal{M} \to \mathcal{S}$ and $\mathsf{Hash}_k : \mathcal{M} \times \mathcal{C} \to \mathcal{K}$ be quantum-accessible random oracles. Let $\mathsf{F} : \mathcal{M} \times \mathcal{C} \to \mathcal{K}$ be a PRF. Let $\delta$ be the decryption error rate of $\Pi$ and $\eta$ be the injectiveness parameter of the derandomized $\Pi$ by $\mathsf{Hash}_s$.[4] For an IND-CCA adversary $\mathsf{A}$ against $\Pi_{CCA}$ issuing at most $q_H$ (resp. $q_S$) quantum queries to $\mathsf{Hash}_k$ (resp. $\mathsf{Hash}_s$) with query depth at most $d_H$ (resp. $d_S$) and at most $q_{dec}$ classical decapsulation queries, we can construct two adversaries of running time roughly bounded by $3 \cdot Time_{\mathsf{A}}$. These adversaries are:*

- *an IND-CPA adversary $\mathsf{A}'$ against $\Pi$;*
- *a PRF adversary $\mathsf{A}''$ against $F$ issuing at most $q_{dec}$ queries.*

*These adversaries satisfy:*

$$\begin{aligned}\mathsf{Adv}^{IND\text{-}CCA}_{\Pi_{CCA}}(\mathsf{A}) \leq\ & 8 \cdot d_H \cdot (d_S + 1) \cdot \left( \mathsf{Adv}^{IND\text{-}CPA}_{\Pi}(\mathsf{A}') + \frac{8 \cdot (3q_S + 1)}{|\mathcal{M}|} \right) \\ & + 6 \cdot (3q_S + q_{dec}) \cdot \left( (8d_S + 1)\delta + \sqrt{3\eta} \right) \\ & + (4d_H + 12) \cdot \eta + 2\mathsf{Adv}^{PRF}_{\mathsf{F}}(\mathsf{A}'').\end{aligned}$$

## 5.3 Concrete security

We estimate the concrete security based on the primal attack and the hybrid attack that are two best known attacks in lattice-based cryptography. Additionally, an attacker can break a KEM through either key recovery or message recovery. For BAT, the cost of key recovery relies on the hardness of the NTRU assumption, while the cost of message recovery relies on the hardness of the Ring-LWR assumption. Thus, we respectively analyze the costs of key recovery and message recovery based on the primal and hybrid attack.

### 5.3.1 Cost of lattice reduction

We begin with a brief introduction to lattice reduction that is heavily used by both primal and hybrid attacks. Currently, the most practical lattice reduction algorithms are BKZ [56] and BKZ 2.0 [16]. Let BKZ-$\beta$ denote the BKZ/BKZ 2.0 with blocksize $\beta$. For a $d$-dimensional lattice $\mathcal{L}$, BKZ-$\beta$ would generally find some $\mathbf{v} \in \mathcal{L}$ with $\|\mathbf{v}\| \leq \delta_\beta^d \,\mathrm{vol}(\mathcal{L})^{1/d}$ and

$$\delta_\beta \approx \left( \frac{(\pi\beta)^{\frac{1}{\beta}}\beta}{2\pi e} \right)^{\frac{1}{2(\beta-1)}}$$

when $d > \beta > 50$.

The cost of running BKZ-$\beta$ on a $d$-dimensional lattice is estimated by

$$C_{\text{BKZ-}\beta} = t \cdot d \cdot C_{\text{SVP-}\beta}$$

where $t$ is the tour number BKZ-$\beta$ takes and $C_{\text{SVP-}\beta}$ is the cost of solving SVP on a $\beta$-dimensional lattice. We follow a typical setting taking quantum speedups into account:

$$t = 1, \quad C_{\text{SVP-}\beta} = 2^{0.265\beta + 16.4}, \quad C_{\text{BKZ-}\beta} = d \cdot 2^{0.265\beta + 16.4}.$$

*Remark* 3. The BKZ cost model we use is not extremely conservative: some lattice-based schemes use the Core-SVP model in which $C_{\text{BKZ-}\beta} = 2^{0.265\beta}$ (resp. $2^{0.292\beta}$) for quantum (resp. classical) setting. For a fair comparison, we shall also show the required blocksize $\beta$ along with the estimated cost.

[4] The specific definitions of $\delta$ and $\eta$ are given in [10].

### 5.3.2 Primal attack

The primal attack consists of constructing a uSVP (unique-SVP) instance and solving it by lattice reduction. We refer to [3, 2, 1] for details.

For key recovery, the uSVP instance is $\begin{pmatrix} q\mathbf{I}_n & M_n(h) \\ & \mathbf{I}_n \end{pmatrix} \in \mathbb{Z}^{2n\times 2n}$ where $M_n(h)$ is the matrix form of the public key $h$. The secret key pair $\begin{pmatrix} g \\ f \end{pmatrix}$ is a short vector of the uSVP instance. To optimize the primal attack, one can reduce the instance dimension by "forgetting" some equations and take homogeneousness into account.

For message recovery, it suffices to recover $s$ from $c_1 = \left\lfloor \frac{(hs \bmod q)}{k} \right\rceil$. Let $c = k \cdot c_1$, then $c = hs - k\left(\frac{hs}{k} - \left\lfloor \frac{(hs \bmod q)}{k} \right\rceil\right) := hs + e \bmod q$. We construct the uSVP instance as $\begin{pmatrix} q\mathbf{I}_n & -M_n(h) & \mathbf{coef}(c) \\ & \mathbf{I}_n & \\ & & 1 \end{pmatrix} \in \mathbb{Z}^{(2n+1)\times(2n+1)}$ that contains $\begin{pmatrix} e \\ s \\ 1 \end{pmatrix}$ as a short vector. Unlike the case of key recovery, the unknowns $s$ and $e$ have different distributions. Therefore, the primal attack can be improved by re-scaling technique. Also, the strategy of "forgetting" some equations still works here.

The primal attack with all above optimizations is systematically discussed in [21]. We estimate the cost of primal attack with the open-source script[5] of [21]. Numbers are shown in Table 5.

### 5.3.3 Hybrid attack

The hybrid attack was first proposed to cryptanalyze NTRU [40]. Later, the hybrid attack is adapted to solving uSVP instances and then analyzed taking quantum speedups into account [13, 36, 60].

To estimate the hybrid attack, we first construct the uSVP instance as Section 5.3.2 as per key recovery and message recovery. The hybrid attack solves a uSVP instance $\begin{pmatrix} \mathbf{B} & \mathbf{C} \\ & \mathbf{I}_r \end{pmatrix} \in \mathbb{Z}^{d\times d}$ as follows. Let $\begin{pmatrix} v_l \\ v_g \end{pmatrix}$ be the short vector of the uSVP instance. First, it guesses the last $r$ coefficients $v_g$. Next, it solves a BDD instance of $(\mathbf{B}, \mathbf{C}v_g)$ by lattice reduction and obtains the first $(d-r)$ coefficients $v_l$. For each $(v_g, v_l)$, it labels $v_g$ with some addressed determined by $v_l$. After collecting many such $v_g$'s, it finds a collision, that is $(v'_g, v''_g)$ at one same address, which implies a uSVP solution with some probability.

We estimate the cost of the hybrid attack with the open-source script[6] by Thomas Wunderer. Our estimate is based on the hybrid attack with quantum speedups. Numbers are shown in Table 5.

**Table 5:** Concrete security estimate for BAT. The item "A/B" denotes the attack cost A and the required BKZ blocksize B. The item "A" denotes the attack cost A.

| Security | Key Recovery | | Message Recovery | |
|---|---|---|---|---|
| | primal | hybrid | primal | hybrid |
| 80 bits | 87.8 / 236 | 90.3 | 83.3 / 219 | 79.5 |
| 128 bits | 152.1 / 475 | 164.1 | 144.6 / 447 | 140.1 |
| 256 bits | 274.4 / 933 | 314.4 | 278.6 / 949 | 275.7 |

[5] https://github.com/lducas/leaky-LWE-Estimator
[6] https://github.com/lducas/LatRedHybrid

# 6 Implementation Details

We implemented BAT with integer-only computations. We provide here some details on the used implementation techniques.

## 6.1 Key pair generation

Key pair generation starts with producing the short polynomials $f$ and $g$, then solving the NTRU equation to obtain $F$ and $G$. This specific step uses the algorithm described in [53]. Compared with the reference implementation of Falcon, the following differences are noteworthy:

- BAT polynomials have a lower norm than their Falcon equivalent. The polynomial resultants obtained at the deepest level of the recursive algorithm are then shorter, which improves performance.
- All uses of floating-point operations (for Babai's nearest-plane algorithm) have been replaced with fixed-point values (over 64 bits, with 32 fractional bits), which removes all dependencies on the floating-point unit. Since fixed-point values have a limited range, this implies that the reduction may fail, leading to a key pair generation restart. Failed cases can be efficiently filtered out early in the process by checking the current partial solution to the NTRU equation modulo a small prime integer; hence, the overhead implied by these restarts is low. At degree $n = 512$, about 30% of candidate $(f, g)$ pairs lead to a restart.
- Some memory reorganization allowed for additional RAM savings, down to 12288 and 24576 bytes for $n = 512$ and 1024, respectively (compared to 14336 and 28672 bytes for Falcon).

Once the complete NTRU basis $(f, g, F, G)$ has been obtained, ComputeVec is used to obtain $w$. The polynomials $\gamma^2 F\overline{f} + G\overline{g}$ and $\gamma^2 f\overline{f} + g\overline{g}$ are first computed modulo a small prime where NTT can be applied for efficient computations (a 31-bit prime is used; since the basis coefficients are all small, it is easily seen that coefficients do not exceed $2^{19}$ in absolute value). The $v$ polynomial is then obtained by performing the division in the FFT domain, using the same fixed-point code as the one used for solving the NTRU equation. The division itself is performed with a constant-time bit-by-bit routine.

Since fixed-point values are approximations of the real coefficients of $v$, the rounding step may occasionally be wrong by 1. Extensive tests show that it is a relatively uncommon occurrence (it happens in about 0.5% of keys at $n = 512$) and always when $v$ is close to $z + 1/2$ for some integer $z$; over 30000 random key pairs, the largest observed deviation of $v - 1/2$ from the closest integer, for coefficients where our implementation rounds to $w$ incorrectly, is lower than $2 \times 10^{-4}$. This means that in all observed cases, $|w_i - v_i| < 1/2 + 2 \times 10^{-4}$. Since the decoding process works as long as $|w_i - v_i| < 1$, the impact on the decryption failure rate is negligible.

## 6.2 Field operations

Efficient and secure (constant-time) operations in the small base fields (modulo $q$ and $q'$) are implemented with Montgomery multiplication. Namely, a value $x$ modulo $q$ is represented by an integer $y$ in the 1 to $q$ range (inclusive), such that $y = 2^{32}x \bmod q$. Montgomery reduction can be implemented in two 16-bit multiplications, two shifts and one addition; they can moreover be mutualized because analysis shows that reduction works properly for values up to close to $2^{32}$. For details on this technique, see [52].

On recent x86 platforms, SIMD opcodes can be used to further optimize operations. AVX2 registers can store 16 values modulo $q$ (or $q'$) and perform 16 Montgomery reductions

in parallel. The `_mm256_mullo_epi16()` and `_mm256_mulhi_epu16()` intrinsics compute, respectively, the low and high halves of a 16-bit product, with a very low reciprocal throughput (0.5 cycles). Computing 16 modular multiplications in parallel requires in total only 6 invocations of such intrinsics.

## 6.3 NTT multiplication

Since $q-1$ and $q'-1$ are multiples of 256 for BAT, the NTT can be applied to speed up computations over polynomials modulo $X^n+1$, when working with integers modulo either $q$ or $q'$. For $n$ a power of two up to $2^7 = 128$, the NTT representation of a polynomial $f$ is the set of $f(\zeta^{2i+1})$ for $0 \leq i \leq n-1$, where $\zeta$ is a primitive $2n$-th root of 1 modulo $q$ (or $q'$). In NTT representation, addition and multiplication of polynomials can be done coefficient-wise, hence with cost $O(n)$ operations modulo $q$. Moreover, conversion to and from NTT representation can be done in $O(n \log n)$ steps.

For larger degrees, we cannot use full NTT representation, but we can still optimize operations by splitting polynomials as follows. Consider $n = 512$; the polynomial $f$ modulo $X^{512}+1$ can be split into four sub-polynomials as follows:

$$f = f_0(X^4) + Xf_1(X^4) + X^2 f_2(X^4) + X^3 f_3(X^4)$$

the polynomials $f_i$ being of degree up to 127, and operating modulo $X^{128}+1$. Then, operations on such polynomials can be expressed as a relatively small number of operations on the sub-polynomials, which themselves can be implemented in the NTT domain, since the sub-polynomials are of degree less than 128.

In our implementation, the NTT representations of the sub-polynomials are interleaved, so as to maximize parallelization efficiency.

## 6.4 Polynomial splitting and Karatsuba multiplication

For LW-BAT, we use $q = 128$, which prevents us from using the NTT straightforwardly[7]. Instead, for polynomial multiplications, we use Karatsuba with an even/odd split, by writing a polynomial $f$ as:

$$f = f_0(X^2) + Xf_1(X^2)$$

with $f_0$ and $f_1$ being half-size polynomials (they operate modulo $X^{n/2}+1$). We can then express the product of $f$ and $g$ as:

$$fg = (f_0g_0)(X^2) + X^2(f_1g_1)(X^2) + X((f_0+f_1)(g_0+g_1) - f_0g_0 - f_1g_1)(X^2)$$

i.e. we turn the multiplication of two polynomials modulo $X^n+1$ into three multiplications of polynomials modulo $X^{n/2}+1$. We use this reduction recursively, until polynomials have degree less than 4.

The same split is used to compute polynomial divisions modulo $X^n+1$: this is used to compute the public key $h = g/f \bmod q$, and also to rebuild $G$ from $f$, $g$ and $F$ when the short format for private key storaged was used. The even-odd split allows us to write:

$$\begin{aligned}\frac{1}{f} &= \frac{f_0(X^2) - Xf_1(X^2)}{(f_0(X^2) + Xf_1(X^2))(f_0(X^2) - Xf_1(X^2))} \\ &= \frac{f_0(X^2) - Xf_1(X^2)}{(f_0^2 + Xf_1^2)(X^2)}\end{aligned}$$

which reduces inversion modulo $X^n+1$ to a multiplication (modulo $X^n+1$) and an inversion modulo $X^{n/2}+1$. Applied recursively, this method leads us to the simple problem of inverting an integer modulo $q = 128$, which can be done in a few inexpensive multiplications.

[7] A very recent work [19] shows that by introducing an extra prime $p$ such that $p > nq^2/2$ and $p = 1 \bmod 2n$, one can implement the multiplication over $\mathcal{R}_q$ with NTT over $\mathcal{R}_p$.

## 6.5 Decoding

Decoding involves computing polynomials with integer coefficients modulo $q$, $q'$ and $Q$. The final step requires solving for $e'$ and $s'$; we only need $s'$ in practice, since we can use encapsulation to verify the result. Moreover, there are only two possible values for each coefficient of $s'$ (for $1/2$ and $-1/2$) and we merely need to disambiguate between these two values. To keep to integer values, we do not recover $s'$ but $qq'Qs'$; moreover, we perform computations modulo an additional small prime distinct from $q$ and $q'$. In practice, when $q = 257$, we perform the last step by working modulo 769; when $q = 128$ or 769, we use computations modulo 257.

## 6.6 Encoding and storage

We defined compact encoding formats for public keys, private keys, and ciphertexts. Each format starts with a single header byte which identifies the object type and parameter set.

**Public keys** are polynomials with coefficients modulo $q$. When $q = 257$, we encode coefficients by groups of eight, each group using 65 bits: each coefficient is split into a low half (4 bits) and a high half (value 0 to 16, inclusive); eight "high halves" are encoded over 33 bits in base 17. For $q = 769$, a similar mechanism is used, with 5 coefficients being encoded over 48 bits. All encoding and decoding operations can be implemented with only simple 32-bit multiplications, and can be done efficiently in a constant-time manner (this last property does not nominally matter for public keys, which are public).

**Ciphertexts** are mainly polynomials with small, signed integer coefficients. When $q = 257$, coefficients of $c_1$ are in the $-64$ to $+64$ range; eight coefficients are encoded over 57 bits, in a way similar to public key encoding. For $q = 769$, coefficients of $c$ are in the $-96$ to $+96$ range, and five coefficients are encoded over 38 bits in base 193. The value $c_2$, which is a fixed-size binary value, is simply appended to the encoding of $c_1$.

**Private keys** have a "short" and a "long" formats. The long format includes the 32-byte seed that was used to generate $f$, $g$, and the 32-byte value $r$ (which is used when decapsulation fails). This seed is followed by a copy of $r$, then the polynomials $f$, $g$, $F$, $G$ and $w$ themselves, and the public key $h$. Coefficients of $f$ and $g$ are encoded over 4 bits each, in two's complement notation; for $F$ and $G$, 6 bits are used per coefficient, and 17 bits for $w$. The public key $h$ uses the same encoding as in the public key. The short format only stores the 32-byte seed, and the polynomial $F$: the value $r$ and the polynomials $f$ and $g$ are regenerated with the same pseudorandom (deterministic) process that was used during key pair generation; $G$ is recomputed using the NTRU equation (modulo $q$); and $w$ and $h$ are recomputed. While the short format is substantially shorter, decoding a private key stored in the short format has a nonnegligible overhead, but is still much cheaper than key pair generation, since the most expensive part (solving the NTRU equation from $f$ and $g$ alone) is avoided.

The numbers for required storages are listed in Table 6.

**Table 6:** The required storage of BAT (full format, including the header byte).

| Security | Public Key (bytes) | Ciphertext (with FO, bytes) | Private key (short, bytes) | Secret Key (long, bytes) |
|---|---|---|---|---|
| 80 bits | 225 | 203 | 225 | 1473 |
| 128 bits | 521 | 473 | 417 | 2953 |
| 256 bits | 1230 | 1006 | 801 | 6030 |

## 6.7 Speed benchmarks

We provide two implementations — (1) plain portable C version and (2) AVX2 version. We measured the speed on an Intel i5-8259U CPU clocked at 2.3 GHz; TurboBoost is disabled. Compiler is Clang-10.0, with optimization flags "`-O3`". The AVX2 implementation uses intrinsic functions, and an additional optimization flag "`-march=native`". For key pair generation, reported value is an average over several hundreds of key pairs (by nature, that process takes a varying time, since each candidate $(f, g)$ may or may not lead to a successful key pair generation; it is still "constant-time" in that timing variations are independent of the value of the private key which is ultimately generated). For encapsulation and decapsulation, reported value is the core process, without random generation (from the operating system), hashing to derive the shared secret, and encoding/decoding costs; on the test system, these extra operations add up to about 20k extra cycles.

The timing data for two implementations are illustrated in Tables 7 and 8 respectively.

**Table 7:** The performance of the plain C implementation of BAT.

| Security | Key Generation (cycles) | Encapsulation (cycles) | Decapsulation (cycles) |
|---|---|---|---|
| 80 bits | $\approx 23.8 \times 10^6$ | 82131 | 392036 |
| 128 bits | $\approx 37.2 \times 10^6$ | 35785 | 279260 |
| 256 bits | $\approx 264.7 \times 10^6$ | 71007 | 537580 |

**Table 8:** The performance of the AVX2 implementation of BAT.

| Security | Key Generation (cycles) | Encapsulation (cycles) | Decapsulation (cycles) |
|---|---|---|---|
| 80 bits | $\approx 23.6 \times 10^6$ | 55736 | 248038 |
| 128 bits | $\approx 30.6 \times 10^6$ | 8355 | 54264 |
| 256 bits | $\approx 185.7 \times 10^6$ | 18479 | 118592 |

# Acknowledgements

This work was partly supported by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701). Paul Kirchner is partly funded by the Direction Générale de l'Armement (Pôle de Recherche CYBER). This work is also partly supported by the National Natural Science Foundation of China (No. 62102216), the National Key Research and Development Program of China (Grant No. 2018YFA0704701), the Major Program of Guangdong Basic and Applied Research (Grant No. 2019B030302008) and Major Scientific and Technological Innovation Project of Shandong Province, China (Grant No. 2019JZZY010133).

# References

[1] Albrecht, M.R., Curtis, B.R., Deo, A., Davidson, A., Player, R., Postlethwaite, E.W., Virdia, F., Wunderer, T.: Estimate all the {LWE, NTRU} schemes! In: SCN 2018. pp. 351–367 (2018)

[2] Albrecht, M.R., Göpfert, F., Virdia, F., Wunderer, T.: Revisiting the Expected Cost of Solving uSVP and Applications to LWE. In: ASIACRYPT 2017. pp. 297–322 (2017)

[3] Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum Key Exchange—A New Hope. In: USENIX Security 16. pp. 327–343 (2016)

[4] Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Kyber (2020), https://pq-crystals.org/kyber/data/kyber-specification-round3.pdf

[5] Baan, H., Bhattacharya, S., Cheon, J.H., Fluhrer, S., Garcia-Morchon, O., Laarhoven, T., Player, R., Rietman, R., Saarinen, M.J.O., Son, Y., Tolhuizen, L., Arce, J.L.T., Zhang, Z.: Round5: KEM and PKE based on (Ring) Learning with Rounding (2020), https://round5.org/doc/Round5_Submission042020.pdf

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

[7] Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: EUROCRYPT 2012. pp. 719–737 (2012)

[8] Basso, A., Mera, J.M.B., D'Anvers, J.P., Karmakar, A., Roy, S.S., Beirendonck, M.V., Vercauteren, F.: SABER: Mod-LWR based KEM (2020), https://www.esat.kuleuven.be/cosic/pqcrypto/saber/files/saberspecround3.pdf

[9] Bernstein, D.J., Chuengsatiansup, C., Lange, T., van Vredendaal, C.: NTRU prime: Reducing attack surface at low cost. In: SAC 2017. pp. 235–260 (2017)

[10] Bindel, N., Hamburg, M., Hövelmanns, K., Hülsing, A., Persichetti, E.: Tighter proofs of cca security in the quantum random oracle model. In: TCC 2019. pp. 61–90 (2019)

[11] Bogdanov, A., Guo, S., Masny, D., Richelson, S., Rosen, A.: On the hardness of learning with rounding over small modulus. In: TCC 2016. pp. 209–224 (2016)

[12] Brakerski, Z., Döttling, N.: Lossiness and Entropic Hardness for Ring-LWE. In: TCC 2020 (2020)

[13] Buchmann, J., Göpfert, F., Player, R., Wunderer, T.: On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack. In: AFRICACRYPT 2016. pp. 24–43 (2016)

[14] Chen, C., Danba, O., Hoffstein, J., Hülsing, A., Rijneveld, J., Schanck, J.M., Schwabe, P., Whyte, W., Zhang, Z.: NTRU: A submission to the NIST post-quantum standardization effort (2020), https://ntru.org/

[15] Chen, L., Zhang, Z., Zhang, Z.: On the hardness of the computational Ring-LWR problem and its applications. In: ASIACRYPT 2018. pp. 435–464 (2018)

[16] Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better Lattice Security Estimates. In: ASIACRYPT 2011. pp. 1–20 (2011)

[17] Cheon, J.H., Kim, D., Lee, J., Song, Y.: Lizard: Cut off the tail! A practical post-quantum public-key encryption from LWE and LWR. In: International Conference on Security and Cryptography for Networks. pp. 160–177 (2018)

[18] Chuengsatiansup, C., Prest, T., Stehlé, D., Wallet, A., Xagawa, K.: ModFalcon: compact signatures based on module NTRU lattices. In: ASIACCS 2020 (2020)

[19] Chung, C.M.M., Hwang, V., Kannwischer, M.J., Seiler, G., Shih, C.J., Yang, B.Y.: NTT Multiplication for NTT-unfriendly Rings. IACR Transactions on Cryptographic Hardware and Embedded Systems p. to appear (2021)

[20] Cong Chen, Jeffrey Hoffstein, W.W., Zhang, Z.: NIST PQ Submission: NTRUEncrypt A lattice based encryption algorithm (2017), https://ntru.org/resources.shtml

[21] Dachman-Soled, D., Ducas, L., Gong, H., Rossi, M.: Lwe with side information: Attacks and concrete security estimation. In: Crypto 2020 (2020)

[22] D'Anvers, J.P., Guo, Q., Johansson, T., Nilsson, A., Vercauteren, F., Verbauwhede, I.: Decryption failure attacks on IND-CCA secure lattice-based schemes. In: PKC 2019. pp. 565–598 (2019)

[23] D'Anvers, J.P., Rossi, M., Virdia, F.: (One) Failure Is Not an Option: Bootstrapping the Search for Failures in Lattice-Based Encryption Schemes. In: EUROCRYPT 2020. pp. 3–33 (2020)

[24] Das, D., Hoffstein, J., Pipher, J., Whyte, W., Zhang, Z.: Modular lattice signatures, revisited. Designs, Codes and Cryptography 88(3), 505–532 (2020)

[25] Dent, A.W.: A designer's guide to KEMs. In: IMA International Conference on Cryptography and Coding. pp. 133–151. Springer (2003)

[26] Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice Signatures and Bimodal Gaussians. In: CRYPTO 2013. pp. 40–56 (2013)

[27] Ducas, L., Lyubashevsky, V., Prest, T.: Efficient Identity-Based Encryption over NTRU Lattices. In: ASIACRYPT 2014. pp. 22–41 (2014)

[28] Ducas, L., Nguyen, P.Q.: Learning a Zonotope and More: Cryptanalysis of NTRUSign Countermeasures. In: ASIACRYPT 2012. pp. 433–450 (2012)

[29] Ducas, L., Prest, T.: Fast Fourier Orthogonalization. In: ISSAC 2016. pp. 191–198 (2016)

[30] Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU (2020), https://falcon-sign.info/

[31] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Crypto'99. pp. 537–554 (1999)

[32] Gama, N., Nguyen, P.Q.: New chosen-ciphertext attacks on NTRU. In: PKC 2007. pp. 89–106 (2007)

[33] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for Hard Lattices and New Cryptographic Constructions. In: STOC 2008. pp. 197–206 (2008)

[34] Goldreich, O., Goldwasser, S., Halevi, S.: Public-Key Cryptosystems from Lattice Reduction Problems. In: CRYPTO '97. pp. 112–131 (1997)

[35] Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Crypto'97. pp. 112–131 (1997)

[36] Göpfert, F., van Vredendaal, C., Wunderer, T.: A hybrid lattice basis reduction and quantum search attack on LWE. In: PQCrypto 2017. pp. 184–202 (2017)

[37] Guo, Q., Johansson, T., Yang, J.: A Novel CCA Attack Using Decryption Errors Against LAC. In: ASIACRYPT 2019. pp. 82–111 (2019)

[38] Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J.H., Whyte, W.: NTRUSIGN: Digital Signatures Using the NTRU Lattice. In: CT-RSA 2003. pp. 122–140 (2003)

[39] Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: ANTS 1998. pp. 267–288 (1998)

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

[41] Hülsing, A., Rijneveld, J., Schanck, J., Schwabe, P.: High-Speed Key Encapsulation from NTRU. In: CHES 2017. pp. 232–252 (2017)

[42] Jao, D., Azarderakhsh, R., Campagna, M., Costello, C., Feo, L.D., Hess, B., Jalali, A., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Renes, J., Soukharev, V., Urbanik, D., Pereira, G.: Supersingular Isogeny Key Encapsulation (2020), https://sike.org/files/SIDH-spec.pdf

[43] Kuchta, V., Sakzad, A., Stehlé, D., Steinfeld, R., Sun, S.F.: Measure-rewind-measure: Tighter quantum random oracle model proofs for one-way to hiding and cca security. In: EUROCRYPT 2020. pp. 703–728 (2020)

[44] López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: STOC 2012. pp. 1219–1234 (2012)

[45] Lu, X., Liu, Y., Jia, D., Xue, H., He, J., Zhang, Z., Liu, Z., Yang, H., Li, B., Wang, K.: LAC: Lattice-based Cryptosystems (2019), https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions

[46] Lyubashevsky, V., Seiler, G.: Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In: EUROCRYPT 2018. pp. 204–224 (2018)

[47] Lyubashevsky, V., Seiler, G.: NTTRU: Truly Fast NTRU Using NTT. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 180–201 (2019)

[48] Micciancio, D.: Improving lattice based cryptosystems using the Hermite normal form. In: International Cryptography and Lattices Conference. pp. 126–145 (2001)

[49] Nguyen, P.Q., Regev, O.: Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures. In: EUROCRYPT 2006. pp. 271–288 (2006)

[50] NIST: Round 3 candidates of the NIST Post-Quantum Cryptography Standardization (2020), https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions

[51] Paquin, C., Stebila, D., Tamvada, G.: Benchmarking post-quantum cryptography in TLS. In: PQCrypto 2020. pp. 72–91 (2020)

[52] Pornin, T.: Efficient Elliptic Curve Operations On Microcontrollers With Finite Field Extensions. Cryptology ePrint Archive, Report 2020/009 (2020), https://eprint.iacr.org/2020/009

[53] Pornin, T., Prest, T.: More efficient algorithms for the NTRU key generation using the field norm. In: PKC 2019. pp. 504–533 (2019)

[54] Prest, T.: Gaussian Sampling in Lattice-Based Cryptography. Ph.D. thesis, École Normale Supérieure (2015)

[55] Saarinen, M.J.O.: On PQC message lengths and some energy consumption myths (2019), https://groups.google.com/a/list.nist.gov/forum/#!forum/pqc-forum

[56] Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical programming 66(1-3), 181–199 (1994)

[57] Sikeridis, D., Kampanakis, P., Devetsikiotis, M.: Assessing the overhead of post-quantum cryptography in TLS 1.3 and SSH. In: Han, D., Feldmann, A. (eds.) CoNEXT '20: The 16th International Conference on emerging Networking EXperiments and Technologies, Barcelona, Spain, December, 2020. pp. 149–156. ACM (2020)

[58] Sikeridis, D., Kampanakis, P., Devetsikiotis, M.: Post-quantum authentication in TLS 1.3: A performance study. In: 27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020. The Internet Society (2020)

[59] Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: EUROCRYPT 2011. pp. 27–47 (2011)

[60] Wunderer, T.: On the Security of Lattice-Based Cryptography Against Lattice Reduction and Hybrid Attacks. Ph.D. thesis, Darmstadt University of Technology, Germany (2018), http://tuprints.ulb.tu-darmstadt.de/8082/

[61] Zhang, J., Yu, Y., Fan, S., Zhang, Z., Yang, K.: Tweaking the Asymmetry of Asymmetric-Key Cryptography on Lattices: KEMs and Signatures of Smaller Sizes. In: PKC 2020 (2020)

# A Proofs of Theorem 1

To prove Theorem 1, we need Lemma 1 to estimate the norm of the involved vectors.

**Lemma 1.** *Let $\mathcal{R} = \mathbb{Z}[x]/(x^n+1)$ with $n$ a power-of-2. Let $f, g \in \mathcal{R}$ be iid-random over $D_{\mathbb{Z},\sigma_f}$, $\mathbf{B}_{f,g} = \begin{pmatrix} g & G \\ f & F \end{pmatrix} \in \mathcal{R}^{2\times 2}$ be an NTRU basis. Let $v = \frac{\gamma^2 F\overline{f}+G\overline{g}}{\gamma^2 f\overline{f}+g\overline{g}}$ with $\gamma > 0$. Let $(G^\perp, F^\perp) = (G - gv, \gamma F - \gamma f v)$ and $(G', F') = (G - g\lfloor v \rceil_{q'}, F - f\lfloor v \rceil_{q'})$. Let $I(\gamma) = \frac{2\ln(\gamma)}{\gamma^2-1}$ for $\gamma \neq 1$ and $I(1) = 1$. Then*

- $\|(G^\perp, F^\perp)\| \approx \frac{q\gamma}{\sqrt{n}\cdot\sigma_f}\sqrt{I(\gamma)}$;
- $\|(G', \gamma F')\| \leq \frac{q\gamma}{\sqrt{n}\cdot\sigma_f}\sqrt{I(\gamma)} + \frac{n^{1.5}\sqrt{\gamma^2+1}}{2q'}\sigma_f$.

*Proof.* First, it can be verified that $(G^\perp, F^\perp) = q\gamma\left(\frac{-\gamma\overline{f}}{\gamma^2 f\overline{f}+g\overline{g}}, \frac{\overline{g}}{\gamma^2 f\overline{f}+g\overline{g}}\right)$. Let $\xi_n$ be a $2n$-th primitive root of 1, then $\|f\|^2 = \frac{1}{n}\sum_{i=1}^{n} f(\xi_n^{2i-1})\overline{f}(\xi_n^{2i-1})$ for any $f \in \mathbb{K}_n$. Some routine computation yields that

$$\|(G^\perp, F^\perp)\|^2 = \frac{q^2\gamma^2}{n}\sum_{i=1}^{n}\frac{1}{\gamma^2 f(\xi_n^{2i-1})\overline{f}(\xi_n^{2i-1}) + g(\xi_n^{2i-1})\overline{g}(\xi_n^{2i-1})}.$$

We heuristically model all these $f(\xi_n^{2i-1})\overline{f}(\xi_n^{2i-1})$ and $g(\xi_n^{2i-1})\overline{g}(\xi_n^{2i-1})$ as independent random variables drawn from the chi-square distribution of parameter $k = 2$ scaled by $\frac{n}{2}\sigma_f^2$. The average of $\frac{\frac{n}{2}\sigma_f^2}{q^2\gamma^2} \cdot \|(G^\perp, F^\perp)\|^2$ is then estimated as

$$\begin{aligned}\int_0^\infty \int_0^\infty \frac{1}{\gamma^2 x + y} \frac{e^{-\frac{x+y}{2}}}{4} dxdy &= \int_0^\infty \int_0^z \frac{1}{(\gamma^2-1)x + z} \frac{e^{-\frac{z}{2}}}{4} dxdz \\ &= \begin{cases} \frac{\ln(\gamma^2)}{4(\gamma^2-1)} \int_0^\infty e^{-\frac{z}{2}} dz, & \gamma \neq 1; \\ \frac{1}{4}\int_0^\infty e^{-\frac{z}{2}} dz, & \gamma = 1 \end{cases} \\ &= \frac{I(\gamma)}{2}.\end{aligned}$$

Further, we get the following approximation

$$\|(G^\perp, F^\perp)\| \approx \frac{q\gamma}{\sqrt{n} \cdot \sigma_f} \sqrt{I(\gamma)}.$$

Let $\Delta v = v - \lfloor v \rceil_{q'}$, then $\|\Delta v\|_\infty \leq \frac{1}{2q'}$. It is known that $\|f\Delta v\| \leq \frac{n}{2q'}\|f\|$ for any $f \in \mathbb{K}_n$. We estimate $\|f\|$ and $\|g\|$ as $\sqrt{n}\sigma_f$. Then it follows that

$$\begin{aligned}\|(G', \gamma F')\| \leq& \|(G^\perp, F^\perp)\| + \|(g\Delta v, \gamma f \Delta v)\| \\ \leq& \|(G^\perp, F^\perp)\| + \frac{n}{2q'}\|(g, \gamma f)\| \\ \approx& \frac{q\gamma}{\sqrt{n} \cdot \sigma_f}\sqrt{I(\gamma)} + \frac{n^{1.5}\sqrt{\gamma^2+1}}{2q'}\sigma_f.\end{aligned}$$

This completes the proof. □

*Remark* 4. When $\gamma = 1$, our estimation of $\|(G^\perp, F^\perp)\|$ is $\frac{q}{\sqrt{n}\cdot\sigma_f}$. This implies that for optimal trapdoor, $\sigma_f$ should be $\sqrt[4]{2}\sqrt{\frac{q}{2n}} \approx 1.19\sqrt{\frac{q}{2n}}$ slightly larger than the previous heuristic bound $1.17\sqrt{\frac{q}{2n}}$ suggested in [27]. The explanation of the factor 1.17 is given in [54]: the argument makes use of the projection heuristic on both $(g, f)$ and $(G, F)$ two parts. In fact, this explanation seems problematic: as $\mathbf{B}_{f,g}$ is symplectic, once the Gram-Schmidt norms of the $(f, g)$ part are fixed, those of the $(G, F)$ part are determined as well. Thus, the optimal bound for the Gram-Schmidt norm of $\mathbf{B}_{f,g}$ is entirely determined by $\frac{\|\mathbf{b}_0^*\|^2}{\|\mathbf{b}_{n-1}^*\|^2} = \mathrm{Tr}(f\overline{f} + g\overline{g}) \,\mathrm{Tr}\left(\frac{1}{f\overline{f}+g\overline{g}}\right)$ where Tr denotes the trace over $\mathbb{K}_n$.

*Proof of Theorem 1.* Let $s' = s - \mu_s\mathbf{1}$ and $e' = e - \mu_e\mathbf{1}$. It can be verified that

$$c' = (Q(gs' + fe') \bmod qQ)$$

and

$$c'' = (q'Q(Fe' + Gs') - c'w \bmod qq'Q).$$

If $\|fe' + gs'\|_\infty \leq \frac{q}{2}$, then $c' = Q(gs' + fe')$ and $c'' = (Q(F_de' + G_ds') \bmod qq'Q)$. Further, if

$$\max\left\{\|fe' + gs'\|_\infty, \frac{\|F_de' + G_ds'\|_\infty}{q'}\right\} \leq \frac{q}{2},$$

it follows that $c' = Q(gs' + fe')$ and $c'' = Q(F_de' + G_ds')$, which ensures a correct decoding similar to the RO algorithm.

Let $T_1 = fe' + gs'$ and $T_2 = \frac{F_d}{q'}e' + \frac{G_d}{q'}s'$. We approximate $T_1, T_2$ by two random Gaussian vectors of standard deviations $\sigma_{T_1} = \sqrt{n}\sigma_f\sigma_s\sqrt{\gamma^2+1} = \sigma_1$ and $\sigma_{T_2} = \frac{\sigma_s}{q'}\|(G_d, \gamma F_d)\|$. By Lemma 1, we have $\sigma_{T_2} \leq \sigma_2$ and then $\tau \leq \min\left\{\frac{q}{2\sqrt{2}\sigma_{T_1}}, \frac{q}{2\sqrt{2}\sigma_{T_2}}\right\}$. Therefore, the probability of correct decoding is at least $1 - 2n \cdot (1 - \mathrm{erf}(\tau))$. □

# B Changes with respect to Last Submission

In the following we list the changes we made as per the reviews on our last submission "TCHES 2021, issue2 Paper #22".

**Editorial quality.** First we added more connections between sections and explained why some presented results are relevant for this paper. To improve the paper's organization, security arguments and implementation details have been split into two sections. Secondly, some missing references and backgrounds on Falcon have been added. Thirdly, we included "Comparison with Falcon" and an intuitive explanation of why BAT achieves smaller parameters in Section 1, which may help readers to better understand our work. Finally, we proofread the whole paper and clarify the variables and terms before they are used.

**Changed algorithms and parameters.** The encryption (Algorithm 4.2) was modified: the new encryption is IND-CPA secure rather than OW-CPA as the old one. This modification leads to a simpler security proof while increasing a little the cost on the ciphertext size. Also we refined the key generation as per Section 3: the trapdoor of the updated BAT does not always satisfy $\|(f, g)\| \approx \|(F^*, G^*)\|$, which is different from Falcon. This improves the decryption failure rate. All parameters were changed accordingly.

**Decryption failure.** To reduce the decryption failure rate, we increased $q'$ to 64513 and modified the key generation (explained above). In the previous submission, our estimate of decryption failure was heuristic (Theorem 1). In the updated version, we also presented exact values computed for actual secret keys: exact values are even smaller than heuristic ones. Detailed numbers and explanations were shown in Section 4.2.

**Security arguments.** All security arguments including the assumptions, the security proof and the concrete security estimates are now in Section 5. Some references were added after each used assumption to show why it is assumed to hold. We gave a formal security proof of CPA security and followed the updated results of FO transformation. The modifications on concrete security estimates are significant including 1. more details about considered attacks and their relevance to the used assumptions, 2. costing hybrid attacks, and 3. re-estimates made by open-source scripts of the state-of-the-art works.

**Implementation and benchmarking.** We explained more about the fixed-point approximation in Section 6.1. We also compared with SIKE and compressed SIKE in Table 2 as suggested by Review A. Yet we did not report the full timings including random generation, hashing and encoding/decoding, as we explained in the rebuttal. Again, the costs of random generation, hashing and encoding/decoding are not included in the timings of other algorithms in Table 2, even though these operations are more costly than the core process that we measured. Moreover, we cannot provide them for the other schemes.