On adapting NTRU for Post-Quantum Public-Key Encryption

Dutto Simone, Morgari Guglielmo, Signorini Edoardo

Politecnico di Torino & Telsy S.p.A.

De Cifris AugustæTaurinorum - September 30th, 2020

Introduction

Post-Quantum Cryptography (PQC) aims to find non-quantum asymmetric cryptographic schemes able to resist attacks by quantum computers [1].

The main developments in this field were obtained after the realization of the first functioning quantum computers in the 2000s.

Of all the ongoing research projects, one of the most followed by the international community is the NIST PQC Standardization Process, which began in 2016 and is now in its final stages [2].

This public, competition-like process focuses on selecting post-quantum Key-Encapsulation Mechanisms (KEMs) and Digital Signature schemes.

Public-Key Encryption (PKE) schemes won't be standardized since, in general, the submitted KEMs are obtained from PKE schemes and the inverse processes are simple. However, there are cases for which re-obtaining the PKE scheme from the KEM is not straightforward, like the NTRU submission [3].

Our work focused on solving this problem by introducing a PKE scheme obtained from the KEM proposed in the NTRU submission, while maintaining its security.

NTRU
The original PKE scheme

The NTRU submission is inspired by a PKE scheme introduced by Hoffstein, Pipher, Silverman in 1998 [4].

In this original work, polynomial algebra and modular arithmetic are exploited to obtain a cryptosystem whose security relies on the fact that it is difficult to find extremely short vectors in the so-called NTRU lattice.

This problem is directly related to the more general Shortest Vector Problem (SVP) on lattices, which is considered to be resistant to quantum attacks.

The main parameters are:

- n, p prime numbers;
- q integer such that $\mathrm{GCD}(n,q) = \mathrm{GCD}(p,q) = 1$.

Polynomials are represented as arrays, i.e.,
$\mathbf{f}(x) = f_0 + f_1 \cdot x + \ldots + f_{n-1} \cdot x^{n-1} \sim \mathbf{f} = (f_0, f_1 \ldots, f_{n-1})$.

The required polynomial spaces are:

- $R_p = \mathbb{Z}_p[x]/(x^n - 1)$, where $\mathbb{Z}_p =]-\frac{p}{2}, \frac{p}{2}[$;
- $R_q = \mathbb{Z}_q[x]/(x^n - 1)$, where $\mathbb{Z}_q = [-\frac{q}{2}, \frac{q}{2}[$;
- $T_n = \mathbb{Z}_3[x]/(x^n - 1)$, where $\mathbb{Z}_3 = \{-1, 0, 1\}$;
- $T_n(d_1, d_{-1}) \subset T_n$ with d_1 coefficients equal to 1 and d_{-1} coefficients equal to -1 (the others are 0).

Key generation outputs public and secret key:

1. sample two ternary polynomials $\mathbf{f}, \mathbf{g} \in T_n$;
2. try $\mathbf{f}_p = \mathbf{f}^{-1} \in R_p$, $\mathbf{f}_q = \mathbf{f}^{-1} \in R_q$, if fails change $\mathbf{f}$;
3. obtain $\mathbf{h} = \mathbf{g} \cdot \mathbf{f}_q \in R_q$.

Encrypt a message $msg \in \mathbb{Z}_{2^8}^L$ using $\mathbf{h}$:

1. encode msg as a polynomial $\mathbf{m} \in R_p$;
2. sample a random ternary polynomial $\mathbf{r} \in T_n$;
3. obtain $\mathbf{c} = p\,\mathbf{r} \cdot \mathbf{h} + \mathbf{m} \in R_q$.

Decrypt the ciphertext $\mathbf{c}$ using $\mathbf{f}$ and $\mathbf{f}_p$:

1. obtain $\mathbf{a} = \mathbf{c} \cdot \mathbf{f} \in R_q$;
2. obtain $\mathbf{m} = \mathbf{a} \cdot \mathbf{f}_p \in R_p$.

The correctness of the scheme depends on:

$$\begin{aligned}\mathbf{m} &= \mathbf{a}\cdot\mathbf{f}_p \mod p\\ &= (\mathbf{c}\cdot\mathbf{f} \mod q)\cdot\mathbf{f}_p \mod p\\ &= ((p\,\mathbf{r}\cdot\mathbf{h}+\mathbf{m})\cdot\mathbf{f} \mod q)\cdot\mathbf{f}_p \mod p\\ &= (p\,\mathbf{r}\cdot(\mathbf{g}\cdot\mathbf{f}_q)\cdot\mathbf{f}+\mathbf{m}\cdot\mathbf{f} \mod q)\cdot\mathbf{f}_p \mod p\\ &\overset{?}{=} (p\,\mathbf{r}\cdot\mathbf{g}+\mathbf{m}\cdot\mathbf{f})\cdot\mathbf{f}_p \mod p\\ &= p\,\mathbf{r}\cdot\mathbf{g}\cdot\mathbf{f}_p+\mathbf{m}\cdot\mathbf{f}\cdot\mathbf{f}_p \mod p \equiv \mathbf{m}\,.\end{aligned}$$

Decryption failures occur when the reduction mod q in ? is not $p\,\mathbf{r}\cdot\mathbf{g}+\mathbf{m}\cdot\mathbf{f}$ (coefficients not in $[-\frac{q}{2},\frac{q}{2}[$), so they introduced constraints depending on the security level:

$$\mathbf{f}\in T_n(d_f+1,d_f)\,,\ \mathbf{g}\in T_n(d_g,d_g)\,,\ \mathbf{r}\in T_n(d_r,d_r)\,.$$

The best known attacks on NTRU exploit the reduction to a lattice (additive and discrete subgroup of $\mathbb{R}^N$).

The NTRU lattice is generated by the columns of

$$\mathbf{B} = \left(\begin{array}{cccc|cccc} \alpha & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{n-1} \\ 0 & \alpha & \dots & 0 & h_{n-1} & h_0 & \dots & h_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \alpha & h_1 & h_2 & \dots & h_0 \\ \hline 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{array}\right) \in \mathbb{R}^{2n\times 2n}.$$

Since $\mathbf{f}\cdot\mathbf{h} = \mathbf{f}\cdot(\mathbf{g}\cdot\mathbf{f}_q) = \mathbf{g} + q\,\mathbf{u}$, the lattice contains $(\alpha\,\mathbf{f}, \mathbf{g}) = (\mathbf{f}, -\mathbf{u})\cdot\mathbf{B}$ and it is one of its shortest vectors, so that solving the SVP consists in a key-recovery attack.

The NTRU submission

Among all the developed versions of the original NTRU scheme, we considered the finalist submission of the NIST PQC Standardization Process called NTRU [3].

The NTRU submission consists in a KEM with two parameter sets: NTRU-HPS and NTRU-HRSS-KEM. Because of its greater simplicity and its larger range of security levels addressed, we focused on NTRU-HPS.

The submitted KEM achieves IND-CCA2 security and exploits an OW-CPA PKE scheme very similar to the original one, except for the polynomial spaces considered.

The OW-CPA PKE scheme

The parameter set NTRU-HPS consists in:

- $(n, q) \in \{(509, 2048), (677, 2048), (821, 4096)\}$, which define three levels with increasing security;
- $p = 3$, $d = \frac{q}{8} - 2$.

Considering $\phi_n(x) = x^{n-1} + x^{n-2} + \ldots + x + 1$, the required polynomial spaces are:

- $R_q = \mathbb{Z}_q[x]/(x^n - 1)$, where $\mathbb{Z}_q = [-\frac{q}{2}, \frac{q}{2}[$;
- $S_q = \mathbb{Z}_q[x]/(\phi_n)$, where $\mathbb{Z}_q = [-\frac{q}{2}, \frac{q}{2}[$;
- $T = \mathbb{Z}_3[x]/(\phi_n)$, where $\mathbb{Z}_3 = \{-1, 0, 1\}$;
- $T(d) \subset T$ with $d/2$ coefficients equal to 1 and $d/2$ coefficients equal to -1 (the others are 0).

Key generation outputs public and secret key:

1. sample $\mathbf{f} \in T, \mathbf{g} \in T(d)$;
2. obtain $\mathbf{f}_q = \mathbf{f}^{-1} \in R_q$ and $\mathbf{f}_3 = \mathbf{f}^{-1} \in T$;
3. obtain $\mathbf{h} = 3\,\mathbf{g} \cdot \mathbf{f}_q \in R_q$ and $\mathbf{h}_q = \mathbf{h}^{-1} \in S_q$.

Encrypt a pair $(\mathbf{r}, \mathbf{m}) \in T \times T(d)$ using $\mathbf{h}$

1. obtain $\mathbf{c} = \mathbf{r} \cdot \mathbf{h} + \mathbf{m} \in R_q$.

Decrypt the ciphertext $\mathbf{c}$ using $\mathbf{f}$, $\mathbf{f}_3$ and $\mathbf{h}_q$:

1. obtain $\mathbf{a} = \mathbf{c} \cdot \mathbf{f} \in R_q$;
2. obtain $\mathbf{m} = \mathbf{a} \cdot \mathbf{f}_3 \in T$;
3. obtain $\mathbf{r} = (\mathbf{c} - \mathbf{m}) \cdot \mathbf{h}_q \in S_q$;
4. check if $(\mathbf{r}, \mathbf{m}) \in T \times T(d)$.

As for the original PKE scheme, correctness depends on

$$3\,\mathbf{r}\cdot\mathbf{g}+\mathbf{m}\cdot\mathbf{f} \mod q \overset{?}{=} 3\,\mathbf{r}\cdot\mathbf{g}+\mathbf{m}\cdot\mathbf{f}\,.$$

This equality holds because coefficients are in $[-\frac{q}{2}, \frac{q}{2}[$:

- $\mathbf{r} \in T$ and $\mathbf{g} \in T(d)$ so the absolute value of any coefficient of $3\,\mathbf{r}\cdot\mathbf{g}$ is at most $3\,d$;
- $\mathbf{m} \in T(d)$ and $\mathbf{f} \in T$ so the absolute value of any coefficient of $\mathbf{m}\cdot\mathbf{f}$ is at most d.

Thus, in the worst case, $4\,d < \frac{q}{2}$ must hold, and the chosen $d = \frac{q}{8} - 2$ is the maximum possible (even) value.

The advantages of adopting this PKE scheme instead of the original one are:

- obtain always invertible $\mathbf{f}$;
- reduce the quantity of polynomial operations;
- avoid re-encryption to confirm the decryption;
- achieve perfect correctness;
- obtain higher security levels.

The main drawback is that, because of the introduced constraint on $\mathbf{m}$ (i.e., $\mathbf{m} \in T(d)$ instead of T), the scheme is not directly suitable for PKE.

NTRUEncrypt

In order to obtain a PKE scheme using the OW-CPA PKE scheme from the NTRU submission, another submission is considered: NTRUEncrypt [5].

This was submitted for the first round but, because of general similarities, was admitted to the second round only in a merge with NTRU-HRSS-KEM, resulting in the NTRU submission mentioned above [3].

However, the PKE scheme was not considered and only some features from the NTRUEncrypt submission have been incorporated in the proposed KEMs.

Key generation outputs public and secret key:

1. sample $\mathbf{f}, \mathbf{g} \in T_n(d+1, d)$;
2. if $\mathbf{f}$ is not invertible in R_q, sample $\mathbf{f}$ again;
3. obtain $\mathbf{h} = \mathbf{g}/(p\,\mathbf{f}+1) \in R_q$.

Encrypt a message $msg \in \mathbb{Z}_{2^8}^L$ using $\mathbf{h}$:

1. encode (*padding*) msg as $\mathbf{m} \in T_n$ using a *seed*;
2. sample $\mathbf{r} \in T_n$ using $rseed = \mathrm{Hash}(\mathbf{m}|\mathbf{h})$;
3. obtain $\mathbf{t} = \mathbf{r} \cdot \mathbf{h}$ and $mseed = \mathrm{Hash}(\mathbf{t})$;
4. sample $\mathbf{m}_{mask} \in T_n$ using $mseed$;
5. obtain $\mathbf{m}' = \mathbf{m} - \mathbf{m}_{mask} \in R_p$;
6. obtain $\mathbf{c} = \mathbf{t} + \mathbf{m}'$.

Decrypt the ciphertext $\mathbf{c}$ using $\mathbf{f}$:

1. obtain $\mathbf{m}' = \mathbf{f} \cdot \mathbf{c} \in R_p$
2. obtain $\mathbf{t} = \mathbf{c} - \mathbf{m}'$ and $mseed = \text{Hash}(\mathbf{t})$;
3. sample $\mathbf{m}_{mask} \in T_n$ using $mseed$;
4. obtain $\mathbf{m} = \mathbf{m}' + \mathbf{m}_{mask} \in R_p$;
5. sample $\mathbf{r} \in T_n$ using $rseed = \text{Hash}(\mathbf{m}|\mathbf{h})$;
6. if $p\,\mathbf{r} \cdot \mathbf{h} = \mathbf{t}$, decode msg from $\mathbf{m}$.

The scheme is the original one with the introduction of a *padding* function to encode the message in a polynomial with at least 256 bits of entropy and a message *masking* to achieve the wanted security.

This PKE scheme was not part of the second round submission NTRU because, when compared to the adopted OW-CPA PKE scheme described before:

- the sample space for $\mathbf{f}$ does not assure perfect correctness, unless increasing the communication cost at the expense of the security;
- the adopted sampling function can fail to find an invertible polynomial $\mathbf{f}$;
- $\mathbf{m}$ must be masked in order to assure IND-CPA security and to avoid the recovery of short messages through lattice reduction.

However, the structure can be considered as an inspiration for obtaining our IND-CCA2 PKE scheme.

A new PKE scheme

Outline

The new PKE scheme is obtained by:

- adopting from the NTRU submission the OW-CPA PKE scheme with parameter set NTRU-HPS;
- obtaining the IND-CCA2 PKE scheme from the adopted OW-CPA PKE scheme and a padding function, as in NTRUEncrypt.

Because NTRU-HPS and NTRUEncrypt exploit different polynomial spaces, a new padding function is required.

Message padding function

The padding function must be an invertible map

$$\mathrm{Pad} : \mathbb{Z}_{2^8}^L \times \{0,1\}^* \to T(d)\,,\ (msg, seed) \mapsto \mathbf{m}\,,$$

where the *seed* allows to add bits of entropy.

The required padding differs from that of NTRUEncrypt since, instead of $\mathbf{m} \in T_n$, the parameter set NTRU-HPS has $\mathbf{m} \in T(d)$, i.e., there must be $d/2$ coefficients equal to 1 and $d/2$ coefficients equal to -1.

Thus, we apply an encoding function to obtain a ternary polynomial and then exploit the *seed* to both add bits of entropy and achieve the constraint.

The encoding function exploits a map $\zeta : \mathbb{Z}_2^5 \to \mathbb{Z}_3^4$. Those are the smallest sizes that allow to encode at least 32 bytes and add at least 256 bits of entropy.

In order to respect the constraint, the outputs of ζ are taken among the permutations of the ternary arrays $(0, 0, 1, -1), (0, 1, 1, -1)$ and $(0, 1, -1, -1)$.

With this choice there are 36 possible outputs.

Since $\#\mathbb{Z}_2^5 = 32$ four permutations must be excluded. These should be four of those of $(0, 0, 1, -1)$, because excluding some of the others would change the ratio of 1's and -1's in the outputs of the encoding function.

The encoding function is defined as

$$\begin{aligned} \underline{\zeta} : \mathbb{Z}_{2^8}^{L} &\longrightarrow \mathbb{Z}_3^{32\,L/5} \\ (m_1, \ldots, m_L) &\mapsto \zeta\big(m_1[1:5]\big) \| \ldots \| \zeta\big(m_L[4:8]\big)\,, \end{aligned}$$

where $32\,L/5$ is the *bit* length of *msg* multiplied by the ratio between output and input length for ζ (i.e., $4/5$).

As a result, the quantity of 1's in the output is at least $8\,L/5$ and at most $16\,L/5$ (when ζ outputs always two 1's and one -1) and analogously for -1's.

Thus, the maximum length of *msg* is $L \in \mathbb{N}$ such that

$$5 | L \text{ and } 16\,L/5 \leq d/2 \;\Rightarrow\; L = 5\lfloor d/32 \rfloor\,.$$

By construction, $\underline{\zeta}(msg)$ is a ternary array of length $32\,L/5$, while the required length of $\mathbf{m}$ is $n-1$.

Thus, the *seed* is used to generate the remaining $r = n - 1 - 32\,L/5$ coefficients, while reaching the constraint and adding at least 256 bits of entropy.

In the worst cases, the missing 1's and -1's are $a = d/2 - 16\,L/5$ and $b = d/2 - 8\,L/5$ (or viceversa). Thus, the number of possible completions through the *seed* is $\binom{r}{a}\binom{r-a}{b}$, and the minimum entropy is

$$H_{\min} = \log_2\left(\binom{r}{a}\binom{r-a}{b}\right).$$

The table shows the results for each security level.

n	q	d	L	r	a	b	H_{min}
509	2048	254	35	284	15	71	301
677	2048	254	35	452	15	71	367
821	4096	510	75	340	15	135	399

Our padding function allows to:

- add always more than 256 bits of entropy;
- achieve a message length greater than 32 bytes for low security levels and 64 bytes for high security levels, even when the first byte is used to include the message length.

The inverse takes the first $32\,L/5$ entries and applies the inverse of $\underline{\zeta}$, its output is always a byte array of length L.

Message Masking

As in NTRUEncrypt, the IND-CPA security is achieved by masking the message.

However, the only way to modify the polynomial $\mathbf{m} \in T(d)$ obtained through the padding function while maintaining the constraint on the entries is to apply a permutation, which is not secure enough.

Thus, we choose to mask the message before the application of the padding function, exploiting the digest of the required random polynomial:

1. sample $\mathbf{r} \in T$;
2. obtain $\mathbf{m} = \mathrm{Pad}(msg \oplus \mathrm{Hash}(\mathbf{r}), seed) \in T(d)$.

The scheme in detail

Key generation outputs public and secret key:

1. obtain $(\mathbf{h}, \mathbf{f}, \mathbf{f}_3, \mathbf{h}_q) = \text{OW-CPA.keygen}$.

Encrypt a message $msg \in \mathbb{Z}_{2^8}^L$ using $\mathbf{h}$

1. sample $\mathbf{r} \in T$;
2. obtain $\mathbf{m} = \text{Pad}(msg \oplus \text{Hash}(\mathbf{r}), seed) \in T(d)$;
3. obtain $\mathbf{c} = \text{OW-CPA.encrypt}(\mathbf{h}, \mathbf{r}, \mathbf{m})$.

Decrypt the ciphertext $\mathbf{c}$ using $\mathbf{f}$, $\mathbf{f}_3$ and $\mathbf{h}_q$:

1. $(\mathbf{r}, \mathbf{m}, fail) = \text{OW-CPA.decrypt}(\mathbf{f}, \mathbf{f}_3, \mathbf{h}_q, \mathbf{c})$;
2. if $fail = \text{false}$, $msg = \text{Pad}^{-1}(\mathbf{m}) \oplus \text{Hash}(\mathbf{r})$.

Security Assessment

Our transformation is analogous to the NAEP transformation [6] used in NTRUEncrypt.

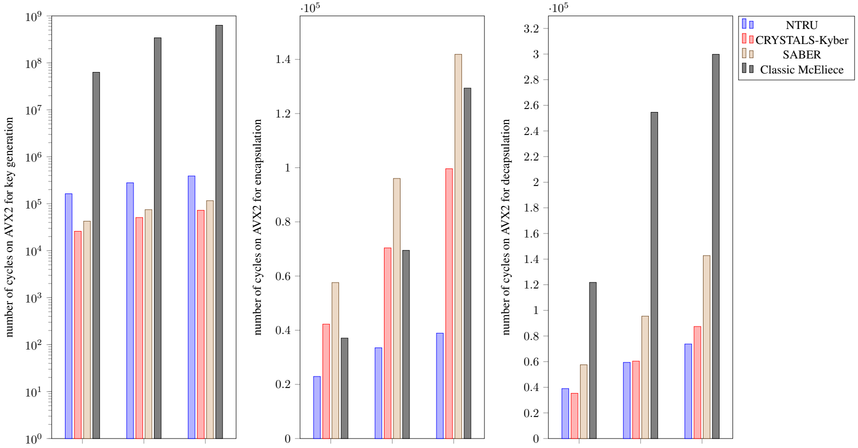
Thus, there is a proved reduction from the IND-CCA2 security of our PKE scheme to the OW-CPA security of the underlying PKE scheme from the NTRU submission.

In addition, our PKE scheme inherit the security levels from the NTRU submission.

n	q	Minimum hardness
509	2048	AES128 with exhaustive key search
677	2048	AES192 with exhaustive key search
821	4096	AES256 with exhaustive key search

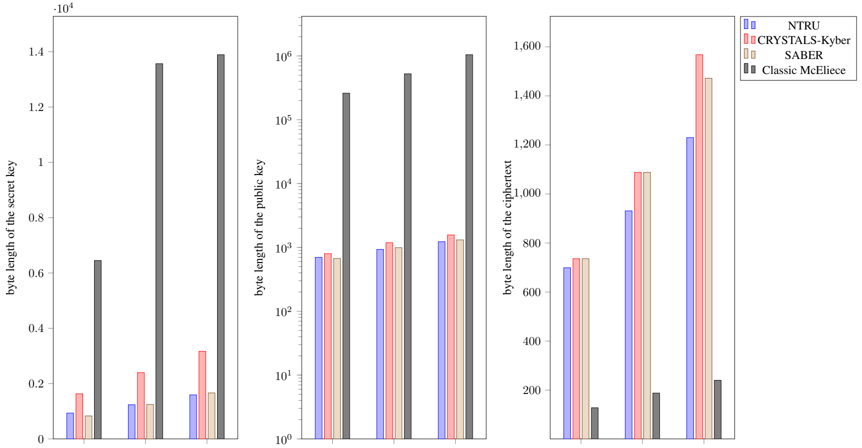
Performance Assessment

Our PKE scheme performs similarly to the KEM in the NTRU submission, here compared with the others [7].



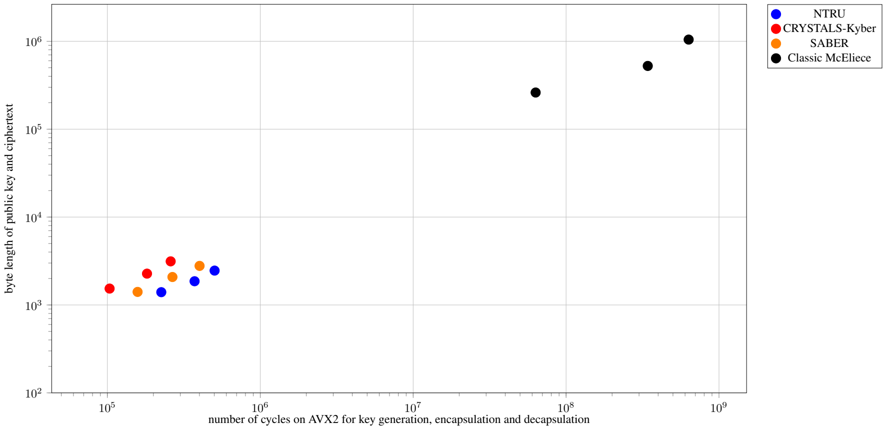
Data-size Assessment

Data in our PKE scheme have size equal to those of the KEM in the NTRU submission, here is a comparison.



Performance-Size Overview

Finally, performance and data-size of the finalists can be compared together.



Bibliography

[1] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and S.-T. Daniel, "NISTIR 8150. Report on Post-Quantum Cryptography," tech. rep., National Institute of Standards & Technology, 2016.

[2] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kesley, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. Smith-Tone, "NISTIR 8309. Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process," tech. rep., National Institute of Standards & Technology, 2020.

[3] C. Chen, O. Danba, J. Hoffstein, A. Hülsing, J. Rijneveld, J. M. Schanck, P. Schwabe, W. Whyte, and Z. Zhenfei, "NTRU, Algorithm Specifications And Supporting Documentation," tech. rep., NIST PQC Standardization, Round 2, 2019.

[4] J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in *Algorithmic Number Theory*, pp. 267–288, Springer Berlin Heidelberg, 1998.

[5] C. Chen, J. Hoffstein, W. Whyte, and Z. Zhenfei, "NIST PQ Submission: NTRUEncrypt, a lattice based encryption algorithm," tech. rep., NIST PQC Standardization, Round 1, 2017.

[6] N. Howgrave-Graham, J. H. Silverman, A. Singer, and W. Whyte, "Naep: Provable security in the presence of decryption failures." IACR Eprint archive, 2003.

[7] VAMPIRE: Virtual Applications and Implementations Research Lab, "eBACS: ECRYPT Benchmarking of Cryptographic Systems - Measurements of key-encapsulation mechanisms, indexed by machine." `https://bench.cr.yp.to/results-kem.html`.