NTRU-LPR IND-CPA: A New Ideal Lattices-based Scheme

Soda Diop[1,3], Bernard Ousmane Sané[1,3], Nafissatou Diarra[1,2], and Michel Seck[1,2]

[1] Cheikh Anta Diop University of Dakar, Senegal
[2] {nafissatou.diarra, michel.seck}@ucad.edu.sn
[3] {sodettes, ousmanendiour2}@gmail.com

Abstract. In this paper, we propose NTRU-LPR IND-CPA, a new secure scheme based on the decisional variant of Bounded Distance Decoding problem over rings (DR-BDD). This scheme is IND-CPA secure and has two KEM variants IND-CCA2 secure in the random oracle model. NTRU-LPR IND-CPA is similar to NTRU LPRime and LPR Cryptosystem. NTRU-LPR IND-CPA doesn't have a problem of decryption failures. Our polynomial ring can be any ring of the form $\mathbb{Z}[x]/(q, f(x))$, where f is a polynomial of degree n and q is an integer. Relatively to the DR-BDD problem, we propose to use square-free polynomials and such polynomials include $f(x) = x^n - x - 1$ (as in NTRU LPRime) and $f(x) = x^n - 1$ (as in NTRU). To avoid some weaknesses in Ring-LWE or NTRU-like schemes (Meet-in-the-middle attack, Hybrid attack, Weak keys, etc.), we do not use sparse polynomials or inversion of polynomials. Furthermore, to avoid backdoors, all polynomials in our scheme can be generated by hash functions. We also give a short comparative analysis between our new scheme and some proposals of the NIST Post-Quantum call (November 2017).
Keywords: *Lattices-based Post-quantum Cryptography, NTRUEncrypt, NTRU-Prime, NTRU-LPRime, NTRU IND-CPA, KEM, Ring-LWE, Titanium, Kyber, NewHope, FrodoKEM, NTRU-HRSS-KEM, Security proof.*

Introduction

Ring-LWE and NTRU-like schemes in Post-quantum cryptography.

On lattices, many problems (CVP, SVP, BDD, SIS,...[50, 29, 44, 46]) are believed to be hard even against quantum computers [5–7], in contrast to factorization and discrete logarithm problems which can be solved easily with quantum computers (Shor's algorithm[52]).

Recently, the NIST proposed the transition into quantum-resistant cryptography, and several proposals were done.

NTRUEncrypt as a candidate for the NIST Post-Quantum call (November 2017) [38] is a public key encryption system designed in 1998 by Hoffstein *et al.* [39]. NTRUEncrypt is designed over the ring $\mathbb{Z}[X]/(q, x^n - 1)$, with $\gcd(n, q) = 1$. The public key is $H = g'/f'$ where g', f' are small and sparse polynomials, and the cipertext is $c = prH + m \mod q$ where r, m are small and sparse polynomials, $\gcd(p, q) = 1$ (r is a secret random, m is the message and p is much more smaller than q). NTRUEncrypt has a problem of decryption failures which decreases its security. It does not have a security proof and the public key of NTRUEncrypt is not proven to be uniformly distributed (except the version of Banks and Sparlinski [8] and those of Stehlé and Steinfeld namely NTRU-IND-CPA [53, 55]). NTRUEncrypt has a KEM variant that is IND-CCA secure in the random oracle model.

A Toolkit for Ring-LWE Cryptography was proposed by Lyubashevsky, Peikert and Regev [32][33]. Some of the NIST Post-Quantum proposals are based on this toolkit. The following scheme is considered as the LPR cryptosystem. It is designed over the ring $\mathbb{Z}[x]/(q, x^n + 1)$, where n is a power of 2 and $2n$ divides $q - 1$. The public key is $G = aH + b$ where a, b are small polynomials, and the cipertext is $c_1 = rH + e_1 \mod q, c_2 = rG + e_2 + (q/2)m \mod q$ where e_1, e_2, r are small polynomials, m is a binary polynomial (r is a secret random, m is the message and e_1, e_2 are the noises). LPR cryptosystem is IND-CPA and is related to RLWE.

NTRU-IND-CPA, as a noisy variant of NTRU, was introduced by Damien Stehlé and Ron Steinfeld [53] in 2011. Stehlé and Steinfeld proved that their NTRU-like scheme is IND-CPA secure in the standard model by using Gaussian distributions. The security of their scheme follows from the already proven hardness of R-LWE problem [32, 43].

NTRU Prime and NTRU LPRime are candidates for the NIST Post-Quantum call [38] proposed by D. J. Bernstein, C. Chuengsatiansup, T. Lange, and C. van V.[11]. These schemes are designed over the field $\mathbb{Z}[X]/(q, x^n - x - 1)$, where n, q are primes and are similar to NTRU and LPR cryptosystem respectively. Recently, Bernstein and other authors have pointed out some vulnerabilities of rings of cyclotomic number fields used in NTRU and NTRU IND-CPA. Their analysis was confirmed later by Albrecht *et al.* in [2] (subfield attacks), Cramer *et al.* in [14] (short generators), etc. To avoid these weaknesses, Bernstein *et al.*[11] propose to use the field $\mathbb{Z}[X]/(q, x^n - x - 1)$ instead of cyclotomic rings. NTRU Prime and NTRU LPRime, as NTRU, do not have a security proof in the standard model. But, there is no problem of decryption failures in NTRU-Prime and NTRU LPRime. NTRU LPRime has a KEM variant, based on Dent [16] transformation that is IND-CCA secure in the random oracle model.

NEWHOPE-CPA-PKE is a candidate for the NIST Post-Quantum call [38] proposed by E. Alkim, R. Avanzi, J. Bos, L. Ducas, A. d. l. Piedra, T. Pöppelmann, P. Schwabe and D. Stebila. It is a variant of the NewHope-Simple scheme [1]. For the distribution of the secret and the error related to RLWE, the authors used the centered binomial distribution. NEWHOPE-CPA-PKE has a problem of decryption failures. NTRU HRSS has a KEM variant (based on a variant of FO transformation) that is IND-CCA secure in the random oracle model.

CRYSTALS-Kyber is a candidate for the NIST Post-Quantum call [38] proposed by P. Schwabe, R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, G. Seiler and D. Stehlé . The authors applied a modification to the LPR encryption scheme(introduced by Lyubashevsky, Peikert, and Regev for Ring-LWE at Eurocrypt 2010 [32]) by using Module-LWE instead of Ring-LWE. In the design of CRYSTALS-Kyber, the authors used a centered binomial distribution (like in NewHope) which relies on the hardness of the LWE instead of LWR(Learning With-Rounding) as the underlying problem. Kyber has a problem of decapsulation failures. Kyber has a KEM variant that is IND-CCA secure in the random oracle model.

Titanium-CPA is a candidate for the NIST Post-Quantum call [38] proposed by R. Steinfeld, A. Sakzad and R. K. Zha [56]. It is a public-key encryption scheme based on the MP-LWE problem(Middle-Product Learning With Errors) [47]. The scheme is an adaptation of Regev's cryptosystem [44]. Titanium-CPA uses a binomial difference distribution (like in New Hope), and has a problem of decryption failures. Titanium has a KEM variant that is IND-CCA secure in the random oracle model.

FrodoKEM is a candidate for the NIST Post-Quantum call [38] proposed by M. Naehrig, E. Alkim, J. W. Bos, L. Ducas, K. Easterbrook, B. LaMacchia, P. Longa, I. Mironov, V. Nikolaenko, C. Peikert, A. Raghunathan and D. Stebila[36]. It is an IND-CPA secure scheme relatively to the hardness of a corresponding LWE problem. The FrodoKEM scheme is a modification of the Lindner–Peikert scheme[28]. The authors used an alternative distribution that is very close to a Gaussian distribution. FrodoPKE has a problem of decryption failures.Frodo has a KEM variant that is IND-CCA secure in the random oracle model.

NTRU-HRSS is a candidate for the NIST Post-Quantum call [38] proposed by A. Hülsing, J. Rijneveld, J. M. Schanck and P. Schwabe. It is a One-Way-CPA secure scheme obtained by a parametrization of NTRUEncrypt but it does not have a security proof in the standard model. NTRU-HRSS eliminates decryption failures by using a large modulus q. NTRU HRSS has a KEM variant that is IND-CCA secure in the random oracle model.

Our proposal.
We remark that all the previous schemes based on Ring-LWE (or Module-LWE, MP-LWE) (over the ring $\mathbb{Z}[x]/(q, x^n + 1)$) are IND-CPA. These schemes use Gaussian or binomial-like distributions for the secret and the noise. Such schemes have a problem of decryption failures which makes difficult in general to design a clear security proof with a tight security reduction.

The others basic variants of NTRUEncrypt and NTRU-HRSS over the ring $\mathbb{Z}[x]/(q, x^n - 1)$, and NTRU-Prime/NTRU-LPRime over the ring $\mathbb{Z}[x]/(q, x^n - x - 1)$) are not IND-CPA but just one-way (and each of these schemes has a KEM variant that is IND-CCA in the random oracle model).

From these observations, our goal in this paper is to design a new scheme:
- similar to NTRU-LPRime and LPR cryptosystem;
- over the ring $\mathbb{Z}[x]/(q, f(x))$, where f is a polynomial of degree n and q is an integer;
- which is IND-CPA and based on the decisional variant of the BDD problem;
- with uniform distribution for the secret and the noise;

- without decryption failures
- and which has a KEM variant that is IND-CCA2 in the random oracle model.

We designed a noisy scheme (called NTRU-LPR IND-CPA) with a security proof , assuming the hardness of the Decisional Ring Bounded Distance Decoding Problem (denoted DR-BDD, the decisional variant of BDD). The encryption and the key generation algorithms are both based on the DR-BDD problem.

We can remark that if the decisional variant of BDD problem is easy then breaking NTRUEncrypt, NTRU-HRSS, NTRU Prime and NTRU LPRime, is also easy by distinguishing their encryption ($c = prH + m \mod q$ or $c_1 = aH + b \mod q$) from random, therefore choosing DR-BDD as our hard problem for NTRU-LPR IND-CPA makes sense.

From our scheme, one can obtain a KEM (following the generic construction of Dent[16] or the transformation of Fujisaki-Okamoto[17]) with an IND-CCA2 level of security in the random oracle model, while maintaining its IND-CPA level of security in the standard model.

Since we have multiple choices for the polynomial ring, one can use the same field than those of NTRU-Prime in order to avoid recent attacks on rings of cyclotomic number fields [2, 14].

In our scheme, it is easier to avoid meet-in-the-middle-attack [24] on the public key and the ciphertext because we do not use sparse "small" polynomials, or inversion of "small" polynomials.

To prevent attacks based on backdoors, all polynomials in our scheme can be generated by hash functions.

This paper is organized as follows.

- In **Section 1:** We give a description of our new scheme, followed by a discussion on the choice of our ring and how we can avoid decryption failures.
- In **Section 2:** We give a security analysis of our new scheme against principal known attacks, and we also describe how to avoid weak keys. The section ends by the security proof.
- In **Section 3:** We describe two KEMs derived from our scheme, which are both IND-CPA-secure and IND-CCA2-secure in the random oracle model.
- In **Section 4:** We discuss about the choice of the parameters of our scheme relatively to some security level. We finish by a comparative analysis between our scheme and some of the NIST Post-Quantum candidates (namely the lattice-based ones).

1 A new Noisy Encryption scheme

As NTRU-LPRime, the scheme that we propose here is similar to LPR cryptosystem.

1.1 Description of the scheme

We consider the rings $\mathcal{R}_s = \mathbb{Z}[x]/(s, f)$ where $s = p, q$ and $\gcd(p, q) = 1$ such that p is much smaller than q (in order to avoid decryption with failures in the following scheme) and f is a polynomial of degree n.

Key generation To generate a pair (Private key, Public key), Alice should do the following:

1. Choose uniformly at random a polynomial H in $\mathcal{R}_q^*$.
2. Choose uniformly at random two (secret) polynomials $a, b \in \mathcal{R}_p$.
3. Compute $U = aH + b \mod q \in \mathcal{R}_q$.
4. Keep a as the private key (and destroy b), and output the public key (H, U).

Encryption
To encrypt a message m with Alice's public key, Bob should do the following:

1. Represent m as an element in $\mathcal{R}_p$.
2. Choose uniformly at random (3 secret small nonzero polynomials) $z, d, \alpha \in \mathcal{R}_p$.
3. Compute $V = -zH + d \mod q$ and $W = p(zU + \alpha) + m \mod q$.
4. Output the ciphertext $c = (V, W) \in \mathcal{R}_q \times \mathcal{R}_q$.

Decryption
To recover the message m from c, Alice should do the following:

1. Obtain the private key a and the ciphertext $c = (V, W)$,
2. Compute $C = apV + W \mod q = ap(-zH + d) + p(zU + \alpha) + m \mod q = pda + pbz + p\alpha + m \mod q = p(zb + da + \alpha) + m \mod q$,
3. Compute $(C \mod q) \mod p = m$ (note by theorem 1 below that $m + p[\alpha + ad + bz] \mod q = m + p[\alpha + ad + bz]$),
4. Output m.

1.2 Choice of the polynomial ring

Much of NTRU-like and RLWE -like cryptoystems [53, 55, 23, 32, 33] are based on rings of cyclotomic number fields and recently many attacks exploiting weaknesses of such rings were proposed [2, 14].
In our scheme, there is no need to invert polynomials. So in theory we can use any polynomial ring of the form $\mathcal{R}_s = \mathbb{Z}[x]/(s, f)$, where $s = p, q$ with $\gcd(p, q) = 1$, f is a square-free polynomial of degree n. It is necessary to choose a specific polynomial f in order to :

- avoid decryption failures;
- obtain a ring compatible with the underlying hard problem (DR-BDD);
- make the polynomial multiplications more efficient;
- avoid the known attacks.

In the following, we propose to use $f(x) = x^n - x - 1 \mod q$ (where n and q are prime, as in NTRU LPRime) or $f(x) = x^n - 1 \mod q$ (where n is prime, q is a power of 2 as in the original NTRU).

1.3 Avoiding Decryption Failures

As previously mentioned, we must choose f in order to avoid decryption failures. The following theorem (similar to those of NTRU Prime[11]) works for an arbitrary prime p; but for reasons of efficiency, p should be restricted to 2 or 3.

Theorem 1. *Fix an integer $n \geq 2$. Let $a, b, z, d, \alpha, m \in \mathcal{R}_p$ be small polynomials and f a polynomial. The polynomial $(p[zb + da + \alpha] + m) \mod f$ has each coefficient:*

1. *when $f(x) = x^n - x - 1$:*
   (a) in the interval $[0, 12n + 3]$, for $p = 2$;
   (b) in the interval $[-18n - 4, 18n + 4]$ for $p = 3$.
2. *when $f(x) = x^n - 1$:*
   (a) in the interval $[0, 8n + 3]$, for $p = 2$;
   (b) in the interval $[-12n - 4, 12n + 4]$, for $p = 3$.

2 Security analysis of the scheme

2.1 Classical attacks

Algebraic computation Let A, T be two elements selected uniformly at random in the field $\mathcal{R}_q$ and consider the equation $T = xA + y \mod q$ $(*)$. Then any solution of (*) is of the form $(x = x_0 + \gamma f \mod q, y = y_0 - \gamma g \mod q)$, where (x_0, y_0) is a solution of (*), (f, g) verifies $fA = g \mod q$ (similar to DSPR of NTRU) and $\gamma \in \mathcal{R}_q$.

Lattice attacks and BDD problem The public key $U = aH + b \mod q$ and the ciphertext $V = -zH + d \mod q$, $W = p(zU + \alpha) + m \mod q$ are all of the form $T = Au + v \mod q$ where u, v are small "random" polynomials in $\mathcal{R}_q$ and A is generated randomly in $\mathcal{R}_q$; thus there exists w such that $T = Au + v + qw$ in $\mathbb{Z}^n$ with identification of polynomials of degree less than $n - 1$ in $\mathbb{Z}[x]$ and vectors of length n (with coefficients $\mathbb{Z}$). Using matrix, we have
$\begin{bmatrix} 1 & 0 \\ A & q \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix} + \begin{bmatrix} -u \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ T \end{bmatrix}$, hence we get an instance of the Bounded Distance Decoding Problem (BDD).

In the context of linear codes, the hardness of BDD was studied by Vardy [57], and later in the context of lattices by Liu et al. [30]. In the case of uSVP(Unique SVP) and BDD, the connection established by [9, 15, 29, 34] is very tight. Therefore, we have an equivalence (within a small constant approximation factor) between the two most central problems used in lattice based public key cryptography and coding theory[9, 15, 29, 34].

It is easy to verify that the lattice of our scheme is the same than those of NTRU ciphertext $c = prH + m \mod (q, f(x))$ (where $f(x) = x^n - 1$, n is prime and $\gcd(n, q) = 1$). It is also the same lattice than some other candidates for the NIST Post-Quantum call [38] such as:

- NTRU Prime, NTRU-HRSS for the ciphertext;

– NTRU LPRime and most of the schemes based on RLWE (such as LPR cryptosystem) for the key generation and the ciphertext.

Peikert [41] says that this lattice (similar to those of RLWE) is as hard as the lattice of NTRU public key. In fact, in a NTRU lattice for public key L_h (where the public key $h = g'/f'$ is given as a ratio of two sparse polynomials f' and g'), we are sure of the presence of an unusual short vector (namely (f', g')). But in our proposal (like in Ring-LWE lattice), there is no unusually short vectors because the polynomials are chosen uniformly at random in $\mathcal{R}_q^*$ and $\mathcal{R}_p$. This analysis of Peikert is true if one consider only the lattice of the public key or the lattice of the ciphertext. But as remarked by Bernstein *et al.* in their NIST proposal [38], if the security analysis is extended on the whole scheme, we can remark that the reuse of the secret r in the ciphertext in NTRU LPRime or LPR cryptosystem is a weakness which does not appear in the previous analysis. Therefore the possibility of the reuse of the secret must be included in the underlying hard problem. That is why, in the decisional variant of BDD problem in our scheme in subsection 2.4, the reuse of the secret is included. The decisional variant of BDD problem that we use is similar to RLWE where all secrets and errors are generated uniformly at random in $\mathcal{R}_p$.

Meet-in-the-middle attack It is known that Odlyzko's meet-in-the-middle attack [24] works over $T = Au + v \mod q$ whenever u, v are small and sparse polynomials in $\mathcal{R}_q$. Here we assume that our polynomials are selected uniformly at random in $\mathcal{R}_p$. Also note that in our proposal, we do not use neither sparse polynomials, nor inversion of polynomials.

For "meet-in-the-middle attack", splits $u = u_1 || u_2$ and test whether $T - u_1.A + u_2.A$ is small. Let $|u_i|$ be the size of u_i then the number of possible pairs (u_1, u_2) is $p^{|u_1|} \times p^{|u_2|}$ and the number of loops can be estimated as $\left(p^{|u_1|} \times p^{|u_2|}\right)^{1/2} = p^{(|u_1|+|u_2|)/2}$. If the polynomials are selected uniformly at random in $\mathcal{R}_p$ then $|u_1| + |u_2| \sim n \log p$, therefore the number of expected steps of this attack is $p^{n/2}$ for polynomials that are small and selected uniformly at random in $\mathcal{R}_p$. Therefore this attack cannot be better than exhaustive search which have a success probability greater than $1/2$.

Hybrid attack The most powerful attack against most of the NTRU-like cryptosystems(for certain parameters sets) is the combination of lattice-basis reduction and meet-in-the-middle attack [24]. For some NTRU variants where the secrets are not sparse polynomials (this is the case for our proposal and for NTRU IND-CPA also), the hybrid attack still work but might be inefficient.

2.2 How to avoid backdoors in the public key

It is important to protect the public key against trapdoors introduced by a dishonest authority (see NewHope [38, 1]).

The public key in our scheme is $U = aH + b \mod q \in \mathcal{R}_q$, where H and (a, b) are randomly selected in $\mathcal{R}_q$ and $\mathcal{R}_p \times \mathcal{R}_p$ respectively. Assume that the Certificate Authority (CA) selects small random polynomials (f, g) with f invertible $\mod q$ and computes $H = f^{-1}.g \mod q \Leftrightarrow f.H = g \mod q$ (as in classical NTRU). Since H looks random, then it can be difficult for Alice to remark this trapdoor. Similar problems can happen with the polynomials a and b by choosing them very sparse. To compute H, a, b securely, Alice can do the following:

1. Choose n to avoid the best known ideal-lattices attacks over $\mathcal{R}_q$.
2. Consider 3 identification numbers: Id_A for Alice, Id_C for the CA and id_P for the current (valid) system parameters, and $\mathrm{ID} = id_A || id_C || id_P$ the identity of Alice encryption scheme.
3. Select a hash function $\mathcal{H}_0$ on $\mathcal{R}_q$.
4. Select a random parameter r of size $|r|$ with $256 \leq |r| \leq 512$.
5. Compute $H = \mathcal{H}_0(\mathrm{ID}, r, 00) \in \mathcal{R}_q$.
6. Select randomly $a, b \in \mathcal{R}_p$ (a, b can be generated via hash functions).
7. Compute $U = aH + b \mod q$ and destroy (b, r).
8. The public key is then (H, U).

NB: To reduce the size of the public key, one can send (r, U) and destroys H; in this case, the computation of H must be included in the encryption algorithm.

2.3 On the Decisional variant of BDD problem

We recall here a decisional variant of BDD (called Decisional Ring Bounded Distance Decoding Problem (DR-BDD)) over $\mathcal{R}_q = \mathbb{Z}[x]/(q, f(x))$ where f is a polynomial of degree n.

- Setup: $\mathcal{R}_q$, p, g, g' three integers with $\gcd(p, q) = 1$.
- Distribution DR-BDD: $\mathbf{Dist}^0_{g,\mathcal{R}_p}$
  - For $1 \leq i \leq g$, $1 \leq j \leq g'$, sample $A_j \overset{\$}{\leftarrow} \mathcal{U}(\mathcal{R}_q^*)$ (public elements generated uniformly at random), and $(v_{ij}, u_i) \overset{\$}{\leftarrow} \mathcal{U}(\mathcal{R}_p \times (\mathcal{R}_p \setminus \{0\}))$ (small secret elements generated uniformly at random)
  - Return $(A_j, T_{ij} = A_j u_i + v_{ij} \mod q)_{1 \leq i \leq g,\ 1 \leq j \leq g'}$.
- Uniform distribution : $\mathbf{Dist}^1_{g,\mathcal{R}_p}$:
  - For $1 \leq i \leq g$, $1 \leq j \leq g'$, sample $(A_j, T_{ij}) \overset{\$}{\leftarrow} \mathcal{U}(\mathcal{R}_q^* \times \mathcal{R}_q)$.
  - Return $(A_j, T_{ij})_{1 \leq i \leq g,\ 1 \leq j \leq g'}$.
- DR-BDD Problem
  Given $(f, q\mathcal{R}_p)$ distinguish with a non negligible probability $\mathbf{Dist}^1_{g,\mathcal{R}_p}$ and $\mathbf{Dist}^0_{g,\mathcal{R}_p}$.

For the choice of our rings adapted to DR-BDD, we can make the following remarks.

1. Let n and q be two prime integers and $f(x) = x^n - x - 1$ an irreducible polynomial over the field $\mathbb{Z}/q\mathbb{Z}$, then the ring $\mathcal{R}_q = \mathbb{Z}[x]/(q, x^n - x - 1)$ is a field (the same as in NTRU-Prime and NTRU-LPRime [11, 38]). Now,

select uniformly at random A in R_q^* and $u \in \mathcal{R}_p, u \neq 0$. Since u is invertible as an element in $\mathcal{R}_q$ then $Au \mod q$ is indistinguishable from random. Therefore v and T are uncorrelated whenever $T = Au + v \mod q$. If u and v are statistically independent, we can assume that $T = Au + v \mod q$ is indistinguishable from a uniform random even if v is not a uniform random in $\mathcal{R}_q$ but only in $\mathcal{R}_p$.

2. The previous result of uniform distribution of $Au \mod q$ and its consequence for non correlation between v and $T = Au + v \mod q$ are proven by Banks and Shparlinski [8] over the polynomial ring $\mathbb{Z}[x]/(q, f(x))$, where f is square-free, even if u is not invertible in $\mathbb{Z}[x]/(q, f)$. Therefore we can use the ring of NTRUEncrypt with $f(x) = x^n - 1$ and $\gcd(n, q) = 1$ (see [8, 39]).

2.4 The IND-CPA security proof

A proof of security of an encryption scheme generally proceeds by demonstrating that if a polynomial-time adversary $\mathcal{A}$ is able to break a security notion (IND-CPA, IND-CCA1 or IND-CCA2) in the encryption scheme, it can be used by a reduction algorithm $\mathcal{B}$ to solve in polynomial time some hard problem related to the encryption scheme.

Given an attacker $\mathcal{A}$ which is able to break a security notion in the encryption scheme in time τ_A with success probability at least ε_A, for the reduction proof, $\mathcal{B}$ must simulate the environment of $\mathcal{A}$ and solves the hard problem with time $\tau_B \geq \tau_A$ and success probability $\varepsilon_B \leq \varepsilon_A$.

For tightness of the reduction it is required to have $\varepsilon_B = \varepsilon_A + \text{negli}(k)$ and $\tau_B = \tau_A + \text{polynom}(k)$ where k is a security parameter, $\text{negl}(k)$ is a negligible function in k and $\text{polynom}(k)$ is a polynomial in k).

Theorem 2. *If the Decisional Ring Bounded Distance Decoding (DR-BDD) problem is hard, then our scheme achieves IND-CPA security in the standard model. More precisely,* $\text{Adv}^{\text{IND-CPA}}(\mathcal{A}) \leq 3\text{Adv}^{\text{DR-BDD}}(\mathcal{B})$.

Proof

In the real scheme, there are 3 pairs: (H, U) (with secret (a, b)); (H, V) (with secret (z, d)) and (U, W') (with secret (z, α)) where $W = pW' + m \mod q$, this leads to the following games: G_0, G_1, G_2. Let (H_2, U_2), (H_2, V_2) and (U_2, W_2') be an instance of DR-BDD generated at random. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an attacker against IND-CPA in time τ_A

G_0 It is the real scheme. Let k be a security parameter. The simulator $\mathcal{B}$ takes k as input and generates a public key $(H, U = Ha + b \mod q)$ where $H \in \mathcal{R}_q^*$ and $a, b \in \mathcal{R}_p$ are selected uniformly at random. $\mathcal{A}_1$ takes (H, U) as input and generates two valid messages of same length (m_0, m_1). $\mathcal{B}$ takes (m_0, m_1) as input and generates a random bit b and encrypt m_b : $V_b = -Hz + d \mod q, W_b = p(Uz + \alpha) + m_b \mod q$ where $z, d, \alpha \in \mathcal{R}_p$. $\mathcal{A}$ takes the ciphertext (V_b, W_b) as input and generates a random bit b^* as its evaluation of b. We denote by Γ_0, this event and we denote by $\Pr(\Gamma_0)$ the probability of

Γ_0. Then $\mathrm{Adv}^{\mathrm{IND-CPA}}(\mathcal{A}) = 2\Pr(\Gamma_0) - 1$. If we denote $\mathrm{Adv}^{\mathrm{IND-CPA}}(\mathcal{A}) = \varepsilon$, then $\Pr(\Gamma_0) = \dfrac{1+\varepsilon}{2}$.

G_1 In G_0, we make just the following change: $(H, U) \longleftarrow (H_2, U_2)$. We denote by $\Pr(\Gamma_1)$ the probability of Game G_1.
Reduction algorithm between Game G_0 and Game G_1: $\mathcal{B}$, define a reduction algorithm $\mathcal{B}_1$ that takes as input (H, U) and is distributed as
- Game G_0 if (H, U) is computed as in the real scheme;
- Game G_1 if (H, U) is selected at random.
Thus, if $\mathcal{A}$ can distinguish Game G_0 from Game G_1, then $\mathcal{B}_1$ can distinguish a distribution of DR-BDD from random. Therefore
$|\Pr(\Gamma_0) - \Pr(\Gamma_1)| \leq \mathrm{Adv}^{\mathrm{DR-BDD}}(\mathcal{A} \circ \mathcal{B}_1)$.

G_2 In G_1, we make just the following change: $(H_2, V_b) \longleftarrow (H_2, V_2)$ and $(U_2, W'_b) \longleftarrow (U_2, W'_2)$. We denote by $\Pr(\Gamma_2)$ the probability of G_2.
Reduction algorithm between Game G_1 and Game G_2: $\mathcal{B}$ define a reduction algorithm $\mathcal{B}_2$ takes as input $(H, V)$and (U, W') and is distributed as:
- Game G_1 if $(H, V)$and (U, W') are computed as in the real scheme;
- Game G_2 if (H, V) and (U, W') are selected at random.
Thus, if $\mathcal{A}$ can distinguish Game G_1 from Game G_2, then $\mathcal{B}_2$ can distinguish one of the two distributions of DR-BDD from random. Therefore
$|\Pr(\Gamma_1) - \Pr(\Gamma_2)| \leq 2\mathrm{Adv}^{\mathrm{DR-BDD}}(\mathcal{A} \circ \mathcal{B}_2)$.
<u>Analysis of Game G_2</u>. The adversary is asked to guess b^* and thereby distinguish between m_0 and m_1. Since $W_b = pW'_2 + m_b$ where W'_2 is selected informally at random and p is invertible then W_b and m_b are uncorrelated thus W_b is independent from b. Therefore, the adversary has no information about b, thus $P(\Gamma_2) = 1/2$.

In summary, we have: $\mathrm{Adv}^{\mathrm{IND-CPA}}(\mathcal{A}) = |\Pr(\Gamma_0) - 1/2| = |\Pr(\Gamma_0) - \Pr(\Gamma_2)|$ $\leq |\Pr(\Gamma_0) - \Pr(\Gamma_1)| + |\Pr(\Gamma_1) - \Pr(\Gamma_2)|$. Therefore we have $\mathrm{Adv}^{\mathrm{IND-CPA}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{DR-BDD}}(\mathcal{A} \circ \mathcal{B}_1) + 2\mathrm{Adv}^{\mathrm{DR-BDD}}(\mathcal{A} \circ \mathcal{B}_2) \leq 3\mathrm{Adv}^{\mathrm{DR-BDD}}(\mathcal{B})$. □

3 KEM from our NTRU-LPR IND-CPA

In this section, we design two variants of KEM derived from the above scheme, and we show that they are both IND-CPA-secure in the standard model and IND-CCA2-secure in the random oracle model.
<u>**Description of the first KEM**</u>: It is similar to those of NTRU LPRime.
Encapsulation
For the encapsulation mechanism, Bob should do the following:

1. Choose uniformly at random $d, z \in \mathcal{R}_p$ and compute $V = -zH + d \mod q$.
2. Choose uniformly at random $\alpha \in \mathcal{R}_p$ and compute $W' = zU + p^{-1}\alpha \mod q$.
3. Round each coefficient of W', viewed as an integer between $-(q-1/2)$ and $(q-1/2)$, to the nearest multiple of p, producing $W = W' + m \mod q = zU + p^{-1}\alpha + m$.
4. Compute and split $\mathcal{H}_1(\alpha \mod 2, \mathrm{ID}, 00) = \mathcal{C}||\mathcal{K}$, where $\mathrm{ID} = id_A||id_C||id_P$ is the identity of Alice and $\mathcal{H}_1$ is a hash function.

5. Output $(V, W, \mathcal{C})$; the session key $\mathcal{K}$ and the key confirmation $\mathcal{C}$.

Decapsulation

For the decapsulation mechanism, Alice should do the following:

1. Alice picks the private key a and the ciphertext $(V, W, \mathcal{C})$
2. Alice computes $C = p(aV+W) \mod q = pad-pazH+pzb+pazH+\alpha+pm$.
3. By the above theorem we know that $\alpha+p[m+ad+bz] \mod q = \alpha+p[m+ad+bz]$. Alice computes $\alpha = (C \mod q) \mod p$.
4. Alice computes and splits $\mathcal{H}_1(\alpha, \mathrm{ID}, 00) = \mathcal{C}'||\mathcal{K}'$,
5. If $\mathcal{C}' = \mathcal{C}$, then she outputs the session key $\mathcal{K}'$; otherwise, she outputs false.

Security proof

1. In the standard model, the IND-CPA security follows from those of the previous variant, since the only change is in $W = zU+p^{-1}\alpha+m$ where $p^{-1}\alpha \mod q$ has the same distribution than α (because p is invertible) where the hard problem is the DR-BDD Problem.
2. In the random oracle model, the IND-CCA2 security follows from those of NTRU-Prime [11] and [16] where the hard problem is the inversion of the underlying encryption function in the One way-CPA model.

We conclude that this KEM variant of our Noisy NTRU scheme, is IND-CPA in the standard model and IND-CCA2 in the random oracle model.

Description of the second KEM

The design of KEM by A. Dent in [16] (table 3 section 6) can directly be applied in our Noisy NTRU scheme as follows.

Encapsulation

For the encapsulation mechanism, Bob should do the following:

1. Generate a suitably bit-string $Y \in \{0;1\}^n$.
2. Compute and split $\mathcal{H}'_1(Y, \mathrm{ID}, 00) = \mathcal{C}"||\mathcal{K}" \in \{0,1\}^{n+k}$, where $|\mathcal{C}"| = n,\ |\mathcal{K}"| = k,\ \mathrm{ID} = id_A||id_C||id_P$ is the identity of Alice encryption scheme and $\mathcal{H}'_1$ is a hash function.
3. Transform $\mathcal{C}"$ as an element $M = \phi(\mathcal{C}")$ of $\mathcal{R}_p$ (an efficient reversible injective encoding ϕ: this encoding can be done by using the canonical embedding since $\mathcal{C}"$ is a binary string with $p \geq 2$)
4. Choose uniformly at random (3 secret small polynomials) $d, z, \alpha \in \mathcal{R}_p$, and compute $V = -zH+d \mod q$ and $W = p(zU+\alpha)+m \mod q$.
5. $D = \mathcal{C}" \oplus Y$ (onetime pad).
6. Output: the ciphertext is $c = (V, W, D)$ and the session key $\mathcal{K}"$ (the key confirmation is $\mathcal{C}"$).

Decapsulation

For the decapsulation mechanism, Alice should do the following:

1. Alice picks the private key a and the ciphertext $C = (V, W)$.
2. Alice computes $C = p(aV+W) \mod q$, $M' = (C \mod q) \mod p$, $D' = \phi^{-1}(M')$ and $Y' = D \oplus D'$.
3. Alice computes and split $\mathcal{H}_1(Y', \mathrm{ID}, 00) = \mathcal{C}"||\mathcal{K}"$,
4. If $\mathcal{C}" = D' \Leftrightarrow \phi(\mathcal{C}") = M'$, output the session key $\mathcal{K}"$ otherwise output false.

4 Comparative analysis and Choice of parameters

4.1 Choice of the parameters

Recently many improvements (BKZ2.0, Sieving algorithms, Quantum search...) with pre-quantum and post-quantum methods, were proposed to decrease the complexity of finding a shortest vector in any lattice [13, 26, 27, 35, 40, 48–50, 59, 58].

Becker, Ducas, Gama and Laarhoven propose in [10] an efficient algorithm that breaks dimension-n SVP in time $2^{(c+o(1))n}$ as $n \longrightarrow +\infty$ with $c \equiv 0.292$; therefore increasing the dimension of the lattice can decrease the security.

BKZ algorithm [13, 4, 22, 51] reduces a lattice basis by using an SVP oracle in smaller dimension b.

The hardness of Ring-BDD is evaluated as an SVP problem, because as far as we know, the best known attacks do not make use of the ring structure. The most efficient attacks are Primal and Dual. The primal attack consists of constructing a unique-SVP instance from the LWE problem and solving it using BKZ. The dual attack consists of finding a short vector in the dual lattice with BKZ.

There are two approaches for BKZ: enumeration (super-exponential running time) and sieving (exponential in time and in memory). For sieving approach, by neglecting the $o(b)$ term, the best known classical and quantum algorithms have time costs of $CBKZ = 2^{0.292b}$ and $QBKZ = 2^{0.265b}$, where b is block size for BKZ 2.0. One must also take in account required size ($SBKZ = 2^{0.2075b}$) for lists of vectors.

1. For $p = 2$ and $f = x^n - x - 1$ (as in NTRU-LPPrime), we need to choose the following parameters: n a prime, q a prime such that $q > 12n + 4$ in order to avoid decryption failures), $x^n - x - 1$ is irreducible in $\mathbb{Z}_q[x]$ and $\mathcal{R}_q$ has a large Galois group, namely the symmetric group S_n (we have $\#S_n = n!$).
2. For $p = 3$ and $f = x^n - 1$ (as in NTRUEncrypt), we need to choose the following parameters: n a prime, q a power of 2 such that $q > 12n + 4$ in order to avoid decryption failures.
3. For $p = 2$ and $f = x^n - 1$ (as in NTRUEncrypt), we need to choose the following parameters: n a prime, $q = 2^t - 1$ such that $q > 12n + 4$ in order to avoid decryption failures.

For example we propose the following table.

f	n	b	p	q	CBKZ	QBKZ	SBKZ	Space Requirement
$x^n - x - 1$	739	607	2	9829	177	160	155	$> 2^{155}$
$x^n - 1$	743	603	3	2^{14}	176	159	155	$> 2^{155}$
$x^n - 1$	743	603	2	$2^{14} - 1$	176	159	155	$> 2^{155}$

Fig. 1. Classical and Quantum security with sieving algorithms

4.2 Comparison with NTRU-like and RLWE-like schemes

Comparison with NTRU-IND-CPA

Stehlé *et al.*[54] proposed a modified version of classical NTRU, for which they showed that it is IND-CPA in the standard model. The public key is uniform but it is generated by a Gaussian distribution with a large standard deviation. This modified version of NTRU is not compatible to the fact of avoiding decryption failures, but in our scheme, we take care of decryption failures.

Comparison with NIST Post-Quantum Proposals

This scheme vs NTRU-like schemes: All the NTRU-like schemes in the NIST Post-Quantum call use rings of the form $x^n - 1$ (NTRUEncrypt, NTRU-HRSS) or $x^n - x - 1$ (NTRU-Prime, NTRU-LPRime) and are more subject to hybrid attacks by using sparse polynomials. In our scheme, we do not restrict ourselves to one of these rings and we do not use sparse polynomials or inversion of polynomials. Our scheme is IND-CPA and is equivalent to the Decisional Ring Bounded Distance Decoding Problem (DR-BDD), which is not the case of the others NTRU-like schemes: if DR-BDD is easy, then NTRU NTRU Prime and NTRU HRSS can be broken.

This scheme vs Ring-LWE (or Module-LWE, MP-LWE) schemes: Most practical Ring-LWE and LWE-like schemes (Kyber, Frodo, Titanium, LPR, NewHope, NTRU-IND-CPA etc.) have a problem of decryption failures because they use Gaussian or binomial distribution in the generation of the secrets and the errors. This weakness makes more difficult to design a clear security proof with a very tight security reduction. We can remark that if DR-BDD problem is easy in the underlying ring, then it is also easy to break all theses schemes. Furthermore, the Ring-LWE schemes are based on a cyclotomic ring $\mathbb{Z}[X]/(q, x^n + 1)$, where n is a power of 2 and $2n$ divides $q - 1$ but the security of most of these schemes does not work over other rings such as $\mathbb{Z}[X]/(q, x^n - 1)$ and $\mathbb{Z}[X]/(q, x^n - x - 1)$. In our scheme all distributions are uniform and there are no decryption failures.

Conclusion

We have proposed a new Lattice-based encryption scheme which is proved to be IND-CPA in the standard model, assuming the Decisional Ring Bounded Distance Decoding Problem (DR-BDD) is hard. We have showed how to turn our scheme into a KEM with IND-CPA level in the standard model and IND-CCA2 level in the random oracle model. We also have compared our work to some Lattice-based candidates of the NIST-Post Quantum call. An interesting work now would be to design a IND-CCA2 secure variant in the standard model.

A Appendix: Implementation in SAGE and Challenge

Implementation

```
import itertools
def concat(lists): return list(itertools.chain.from_iterable(lists))
def bits2hexa(bits):
```

```
    return hex(sum([bits[i]*2**(3-i) for i in range(4)]))[-1]
def hexa2bits(hexa):
    b = int(hexa, 16)
    return [b//8, (b//4)%2, (b//2)%2, b%2]
def encodeZx(m):
    M = [m[i] for i in range(n)]+[0]*(-n % 4)
    return ''.join([bits2hexa(M[i:i+4]) for i in range(0,n,4)])
def decodeZx(mstr):
    return Zx(concat(map(hexa2bits, list(mstr))))
def int2hexaRq(integer):
    strs = hex(integer)[2:]
    return "0"*(4-len(strs))+strs
def hexa2intRq(hexas):
    return int(hexas,16)
def encodeRq(h):
    H = [int(h[i]) for i in range(n)]
    H = ''.join([int2hexaRq(H[i]) for i in range(n)])
    return H
def decodeRq(hstr):
    h = [hexa2intRq(hstr[i:i+4]) for i in range(0,len(hstr),4)]
    if max(h) >= q: raise Exception("pk out of range")
    return Rq(h)
###############################################################################
n = 739; q = 9829; p = 2
Zx.<x> = ZZ[]; R.<xn> = Zx.quotient(x^n-x-1)
Fq = GF(q); Fqx.<xq> = Fq[]; Rq.<xqn> = Fqx.quotient(x^n-x-1)
#Key generation algorithm
def randomS():
    L = Rq.random_element()
    S = Zx([int(L[i])%2 for i in range(n)])
    return S
def keygeneration():
    H = Rq.random_element()
    a, b = randomS(), randomS()
    U = Rq(a)*H+Rq(b)
    return (encodeZx(a),(encodeRq(H),encodeRq(U)))
##############################################################################
# Encryption and decryption algorithm
def encrypt(pk,m):
  H, U = decodeRq(pk[0]), decodeRq(pk[1])
  d, z, alpha = randomS(), randomS(), randomS()
  V = Rq(-z)*H+Rq(d)
  W = p*(Rq(z)*U+Rq(alpha))+Rq(m)
  return encodeRq(V), encodeRq(W)
def decrypt(sk,(V,W)):
    C = Rq(decodeZx(sk))*p*decodeRq(V)+decodeRq(W)
    m = Zx([int(C[i])%2 for i in range(n)])
    return encodeZx(m)
###########################################################################
#Verification
(sk,pk) = keygeneration()
#The symmetric key to cipher
K =  "c6896f6d1cf25aeb86b07795e4f1f0e1af8833f818493c9db0d52b2dff9113a27f066802"\
       +"22e146775074bf8f3da07c83d8d1566ced96f57d28fdb72387742a9a15a85861cab51391"\
       +"8358c59e55912ca0df0a62061685aad66253d8d00"
V, W = encrypt(pk,K)
C =  decrypt(sk,(V,W))
if C == K:
    print "The decryption is well done"
else:
    print "The decryption was failure"
#####################################################################
K0 = 0
encoding of K0:
K0 = "00000000000000000000000000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000"
K1 = 1
encoding of K1:
K1 = "80000000000000000000000000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000"
sk = "???"
Pk=(C1,C2)
C1 = "0b341eaf0de61c67017f239c1b421a23258f00700374219c091606e00cfb0001046e1833230a1b8b0bc10fc4168d260c18ee0021221f130a181925490af207b4080510df1262055\
713ca243824ff05421dca1f2703f0237c19290ff121b5027f036f15d20aa318c224d2086216450fe0213801be17270a9907db1ee003611f4e14a81dba08df026e07871b5806e625e\
6172f1fbe13231187110c05440fc5263d0a4b126501f122a40c4f25b712c91a3113ac1b05253b20631bb318740ef7184b1f4600bd007212302547043b1738182e1e7c1609171d02f\
a0bb90d690f3817740b5709081b730dd105df00e11f4c2574115c17a61f0909a31bd30ff1213a1c9d25f51708142809ea167d12541b8b01c21e371c2f064d24f50b7407cf2173176\
d10df19490a2402d713a218810a89115e24d9061d0fc004ce1066151302f11b990b060706261f1d3b2241083b007b157e0b631a1c023525bd0caa202a203600d314b5228817fd11b\
919b722cb1da9107512ef0026083700500a95090a068219a107e20df9066a1fb00b8823561ffc0d81121f251611290050231d141715620dcd0ecb1b7405a61738049721300a55073\
2244e19a312bc2462251a08be059c141e1c731dcb0a0f216310290b3d07df1d2e1d43180d02c301be197e128b17790cbe0cac1fba0e0e03a11b4b148416c926460bd522e3157811e\
314db1d8313902107018d145d256b13501bf219d01e0e0b5820e0218a0e3704a5162a23fa1f530e9d0655137a1af90edc1308018b232e134c060c2605264105b9206d062718dc0af\
603130b31006f0f8f069b02a211c90a9d217b1c1a1c1422d708bd21580c3f227804ba0b81015e122f063412cc0e4f0dfc08a02662121c1b4b0d8021c50751022f257202841afe25b\
f0b821cb7155f0d6606da14ec1323242609610cf81f0c0f3d022d208e04a7133e1dcc15b0221311aa016613f424d8136d1f201d310d4f18940698141f25941dd103a904071daf1a2\
320460de626440457179d2599077b0e761f6e2271056213260f481c5b01ab01901b5905f411bd15051e48103323a414040ea7115c140e05e110fc196e02790be7206612972079156\
f033b06f51ea315240ea5089e1ea3199f124a059a1a0105f0223c126a255112e52643105d1d9e141d06e825dc259f1a8213960ffb0f5401dc086600d3092d18d91e3e204a194a0fa\
9185c0e7018b716e001fd17ac033c0fe818ff24d6201d080c053c1d3808200f8e1377156100b3253a13641729114801ef0a49264c211501991d9e088f1d6c236a212b0bed00ab0e5\
b1a8914fd16c413d609bd226e1f2c2434238d07171273229d15281cb41038089811a004b21f48147c15a410180af1053c10fa1d4c1d76260310e5115a13871fff0f3d25f104e10a3\
f157f206a0f8703d21436262f0f4112101b6b23621c9816ee0a4f1a42195c170503f60dcd00591e9800b913d3066e062920d4262b0a780d52256a14a81d080c9300f107380d2b023\
d19c315d9126c1e222213010518f90f15203614ec0ec3191f18db1d811bf21f240c4900f51ef91a521a7d03c612ac0d2714d31d2c219d0c3606dd04b103491eed146b20231936171\
```

```
30ecb130e08c901b9029d064c020b08c3186b084a15dc02d71ec918a00983085a23b81eb805961a14059c19e21dfa15d9004407d11892029d0b260d911fe724ad07211fad2638035\
3129015d2125b240225d806010c410db30a470ddd1bb51ce8028c07d3077a195326012146228103 1a0640007c04962610002b12a51e500845012c1ed01a24142a02a912d20e7308d\
2194f20440c6e0a9d1d0414e0049b040102f813e020c90e5a263f22ac0ff222c62355004a03060ae41aa71004113b0d660a780c1520841a0f0b0415fd074e264611f9094314a90ce\
718f925d423e90e8f1c010c07069f234c140b088a077f0b2e019105dd0e0c06a520730c0212e206a600d8093b058f08a416f5245b1d0009ca06d9067b1c3a0a7423f6106717f718e\
d26270ff225a615e910080653129723fb143600a30a660aa80d39001a19832211070025b71645"

C2 = "1bef12811511194907c111b413711ada05c0061e12d01576023e0add11cb074d0d250c0b085b1fcf20550a550ee51c4f21be046220090b56115b23ef12520f920d500239034110b\
616bb092419dd058e195c064f1d8f1a100a5418a211cc0e450566166222 6721b513f71c601b9103ae0d2b24db1525159a16d111450f68204a09921ee108e51fb20d1a177319f5105\
f1eef110d241e132115ad14a81f820a7f152e072b13a20cd71aed21190dab212e026114e81897082b07f2205f0d8c00a0120615ff0de1050c03730ac520ec0c1d01b11c5417240f4\
e1c650727166a0d51098c24610bee00320eec22ea0ca30cb7069e142804f80529026011df115302101b9b0ea21e6816bc24960e1d1c7e178e00451e0d223c157b162b0d0f0b0f04a\
d01bb0dfa131c0a8c0591042f0ac00794170f22bc17ff1ff003330d801f72141113a31bfc2141222f11500158247415ac057822692555 12e6120b112307fe23eb044c081012dc13f\
e03971ebe2444033725a90073156d20ea1c5f139301991ffb0ca61af201b71dbf17030f2713f406b508de0f1b0803076c2327065f03430d0202cf13af206b149a04651d9402441b5\
c0919161a0ccb1352255d16571e34075408c90f0a016e1ecd05e0149606c71e40141d09e02171162c17dd197014180e9e2043024d250d0cd80c0a130e1fb708a1163d00b7124908b\
e117b1ef70f07264721881aef22e70dea11411b111a8915d81f8509a609f40dd706e6242204f808a213b6089e0faf138d0212034303be1d860dbc1db121db20931f5d213402b20e8\
b1ba214eb1d5002e50bbc14ab222c13eb236f181c137207ff0b7618540ea1019023ed17951ebe1d1d204a2518182d1441188a12bc0a0803c51d2620c003510fd022dd063d257007d\
606d00a720217263017b90fe0083b020b18da0680155105ae1a9e21691fc01f4601820fcb08fd00d31ffd06ee0f9c250400d30eaa00470c8f187e08d721981c8e24f121980a721b9\
7197306c402ab061a249f05c1086c103605cb16ea0a8f254f02bf16e316cb1e0710a517f41c521bae0058095106270d471dc101b211131a3f0d520d011d510e8c16060a4c075d1cb\
b19f711c71179024d0d5521dc0f8e1b3b1cee018525950d2e23140c350ea1153a0f04088110ac02db1f3615f6003718d418c40ba102b4021f02d9082a0d8a067512d922010a850ac\
a096d0a7b04cc02bf182124ac1f45168717b103e615ab259000682624172d0e141d9622030b6f1471209e03df1255246607951730158d203d25b0227a1e51130c09931a381136066\
91ffb149305f2002916a80e441aaf02cc236f1f0c21fa0c741cf3119f140e1309030f0568050819292264212410d606b703ad1959016b1c7309ec17e005c9230e25e009a708ea0d7\
a0c8925941ad308fd0614247b027b206a0a4f0e19011a098c07c51d9b110b20bf0946045c24cc05c7125b0f1625a316e916a413de05ca019e139314f814ea258e14760b390a95091\
c245d123218e71a1a217609fe0fc10a510efb127a145907c51eec23d50b0609e51d311b8b04b800d3181d145e03b21c6705821e9c2457215a0f5f02ae0e511e131c612148133c189\
6235d215c138c09f11aa80ed4236c193c03c520ce08992192016219cd12ea06ab16d506a01f6007d6079c0c6d234a1cf703fd09b80ef10c7f039f244b04a024ce0fb21fd1239a161\
71ed5129310de1a2f115b17050b1d245c1e7c0aa10f7a11fc17b104bc154d1fab09e406ed0f350ab90c0500b41ed50e9021da12f5131811370db90a111ea4161d20d2022e2545089\
d200c14820a31250219b110fe24d8101a0f210bc61302233917c422840c892389250624b41750216602d625bc0c401f60244523d41f8d1556018711ce040024da18231cd206d1232\
113a011ac016a07d90c0311570d6506a310ee12c2142d092325f70d61257209481e9909df0f3d198602151d7a04851ab01a7c12681b7f035a10dd02c1171b144c142210ca1a2304d\
40b490624037314aa159d018c0d2120c001cc1b621127045d0244023e15b7205d238c26112263"

#The 1st ciphertext (V,W)
V = "24a41f79113c0bf5147f1d8f07871ace255109f9175b0685079c04fd06691dff026a0ecc00b407f2002006db0be11dea0d4b1d9809741d3213a9031b1862163c0b7e034f1d02250\
1201c09e0095b1c9a0ccb11b70ca2106009210d4c02411a481251031d255725220ac7055207f61ab8265621b80aae061723fa030b218710a7017518d308ac1939142a0dd8162b164\
31a861ab50daf2141215c121218c60c7c2172049b208a086c05481f0b060810d11b5e20da25911c3a05381f921b6700eb0b4e22bd17280b5921e209051f7125200128096907ae10c\
f086c1ae10cc50ff60120183900151428152 11e231c2810d216811d3d01a71419079425751 26210f40587132d247209e211931b131a581dd70c0b16c31fed136a0fe108fa1d35118\
e103f1284174f048f047d05ab0184039322741dd20b561a4807da0e67141d25b615f90a731b1a233202ea221a148d0a42136107e81bae1b0b0b780f740e6b241201772158089512a\
c1fbf0936020e0f450004171f14460c2e07e720a81b5b082615ae146c02f70ae3208519e60fd60a8d196918f603061cab04f3062103d119d51b3008a2011f210121f3041909670d2\
f128e265100ad23fb09481cd60cb3233c1b45020511e20fc003c813a5120a103a0ab013af1d1003611f312484225c002f0c5c1992047d19ea24aa15b31ced22e0088d10e10d1b0d9\
b16b8259e00b60aae0f8b0ccd0f6a0625113921e6213523a918532395137f159305d903390ff916f907c41e750fb5034517780f60029a0c6306571719237a1e5400fc188913321b4\
32610029500c3082a0e18179324311a5f17d60ac50097068d226c0f26050f021817ac0fa6072c226614b52127029f1c95033509c808e0064625af0f0f13f01c1e1bc804450d9b1c7\
407a216fe09a51ac6190b01dd133413c322330c9820ca1e95006002641b0d0d081ab20be61c760405115405d4050c191a027410a81be01f750097262b1ec91afb1d18043b07d1237\
006f51350047e257c1b630ef01cd8116d0edb10f004a3001116341156037704d0119a085c22232382265e14eb04ad092708a103e7092a1a501573255e08f71ff923c205f40ab7232\
208710a910aab20b0139f030a10380e3c262a11661a2c2554201123cb174a1038079f00351f561eb91538154a1c8d23200cac26490c4e070e142d2193066a1b25178116860afc0cd\
f18ec1712030d04b4017a237b1bad110518d80086127a234f0b63128318720c2225b015711674199a0d89243d10e71fce01d41d02100e0de10d4d0c7b1b5e24f1205011421c91091\
7238123f21c600325046e0d381b8d1bc10b3a00ec15e900471b3f0089053d14580f70112e23f0249c09521eb160d13181486106321190a7b1aea032e153818bd23ff16211888041\
c0c5403f40e23122a1e410e9700a113841b5913fd0e6415591dce078e05570b8720ae12a31c85230123ce1ae41c5c1d55132b125606561c8e02601eb618610bec0b5d0d8a0750254\
9181c000f25680c9e0e9a10f1138113e322721d6f0a580db20fdf1e360b321d52058e01af22ca1ecd140119710d9a1b8717621bd51dd223811508144d0a221387211d0e7816a3159\
60ec006fd24ae15d21687164205b412c618bb03980d18050d1511229b0dce08e41ccc19b021fa25d5189404101d7a236b1c2322860619107b02f60cf6064a253b0c32186903281a7\
0088319431c4516ba23a11826078820a7135a25ed1f0b1ca524ac001d212a179604050bf508ac05ee126316ca19810dc908e52463231121c30f0a098b133e193318160f5b1693052\
a181b159f178d21ef1d290751090b052819db19e012640b870956094 1121f02a520cb020d04991061178405b723930d380717016d0b9901271d2e22e91e372446180a24710513203\
606e81bb005f323be007c14a116080623063e1e9a02fb04d508c50cf91da10df10231068122ee215e082d064c21bd073c00cd238100f301511d201a2107ff10a8217b16b40160185\
b07700ee001181f2010830eb4056303ff2307114c17031cbc0a0a00ee0a6e155a038e02c51fd3"

W = "18ba1a1318dd127012ee184a178d10ce01e018ea21961b96185424090f3d2290235308b30f3316de22b0003a0f1d0ad4215925d5186e0a47076c19580ab42644176b087524041e2\
70d4c17b010da0e47160e1dad12af1dbb03d30c6d19cc243a042e12891e2b0560066f20190315205319671c9b1d9a2060035b1821051a12c1024b0caf1a951faf04be074b25f3241\
b25690f6c095a082f1c2c051e184000a708e72017130b0d8825aa14b7250522230daf0df125f706b1010604a903e520d006d912da03ac0053222d1d50224900d11f3b2431150904c\
6155a1311250e15c511b1062c13ba171a1fec213403881c471cc6147509f7039105fb0f1e0cd21b1e0b6f0de70dfa1a320035082f1fdb1db20c910ded1286076925c3014e0bc817c\
219b405da1bb625ee1ead12d411a60d0e1cf006a40cd7196520041a3101eb1923220a17a016d800d11195079f19ce1147040c19e2171f0a760e5713651d5e00a020750e421a32086\
516bc24eb0ce015bf0f2022fd1107175816e222ca02e4120a1a10161220a61c191fc003800c250aa31c2208df22b223be19ba0abb244325ec1d910cd11ec6258e1eb4227f078908d\
d247c0a8f0a111a54156b18521c9802a6053b0ba003a41bab1fb109dc1ff41ecb142d167e1eed23b31a000fc308f41bea197308d603420ed51fc009e5216409cc20dd05650e73108\
41d631afe0e59085e1cfe178125e8085c137c0e81241b0fe70886016302be0e0c0b3e2552060f107b24460be819dd23cd2177245d0d2425ee10861b511e46082a11500b362661109\
8235a21ff097f1e570a7720de1ceb1fa51f73091c0bb31e6e1642155a219f1bc80ca800fb0af302f90f3911f9072908c025bb0c60010a193d20e71c7921ba218605ec03ee244e1e7\
219a21c0e02f508960ee119a41403078520b708c305f508a11dd80c5d1a000cc8119a1b880a7c2250076614d216991dcb212506cd1396181410200ff92483049610432 40923f8081\
b1802004b17140579101e11f40e0818b410b9023f04d925652446234a22b025631cd704cc20c51f6a221e09020c6901282259243b04811b9c104305410ce21abb031e26211c27234\
b16880dc206080fab22e517fb0a9b224c1e3711cf237221c31ac51f7013190af5133d18d2188723890f0a1638230b07be1e18162d259312bb0102204f22800cff19c60fbf0c6202b\
507b1033221070c3f2459180306a118601ecf134c245707141ef219a90d120c52001c16430fbd00b202e00cd3075c246523111dcb0aeb25750bc41f371f1b0677163c141d06f11ee\
0209018921cd00a89170d140c138b071e0d9e231e1e8a00370f6e12c41a7a01401b7124511b56129a01a5038c09f612c50544114706a70e5b00401a56137705830dcf0d3420e220a\
9110e1e61126a015700b00f200ca302d420081d6214c6121b08c315b0141610da1e24032611a2044b1afb119d12ef20180fb40214026b01cd083d13b7218b20801c520c9122e611c\
7103201db0e7c0b77125f082523d02057097004fc084b04071a9f16f70f7d04501335079207a411840090122b22c71622086b0d381d4b05b61e4014be1d340df40d1821dc0b020ec\
90cbd18f71ff001a20450107d1b9a0dc11d7a165a0f3f0cf50a45045f0af025401d8e1e4c181f25da0e4b25c525711575250a16f0214d07a71bcd14721c2c1a43097c180f077d0bb\
30a1b050320ad077616341fd5142101f5218c06fd1f6b08b3238510b22323086a118f1ee6080016b21a08112e13b617f509051f9f1ca40e0013de09b124e51c8e1f0a252b14621b7\
a14ca042d08e71c8e15b10b18229e127b04b607cd191724ac0e7f1e211fd20b34121e0e070cc213750e9f0f0f17a5139b01f90d5e157d1e52122916f2012c11870d6200e10b1106b\
816c31fd60bf0219a1875089c23bf0687148f1a9322731761236014 1d0ae0064e21ee08490904074d165208d313bf0f070d22164d1e9e11af1d760c14126011060c300c750ab61ed\
905b00ed71afd127708f502bb23840efb0014226f0e430e330f151c1114d921c908a6028919ec"
##########################################################################
# The 2nd ciphertext (X,Y)
X = "1c15150f0cbd227f2555064f0681216e1d1d0af11d73184d1d800e7c0e1e13a11645224d055b218201cc129c0bd81e6e127c19631eb11d5e192818f51c9a0be31c2c248a06171d6\
705f615e006d80e2e17281477074c260912ce01b324370bed07ef22a11dc51be40a5a16ab1d0a254e1ff71d7618fc1baf14be066323b20c1f179b0dd20f20045f1ea711d51590249\
c103318fb135504c41647071f0be308ff0e0223b01c48064513ad19401d4424b209ba0c941f981f1911610f0a0f612634195c1eff0faf101d0c831d00140f06b0239a08a306af0d8\
723a21409035c18b9196e12b918b51ec613c91806142a05f6074112cd24c409c80e320e0312d3240317b01dc917810c051e69181c0aec230c04081aa10d7801ad1521080a00cc0e3\
b04001bcf0872007213e708e81e17201304370317156a039f0ca11c7f10c911811616035c1df422101e4d21b01fcf1f17229f2070161f0aa61d581fce265e0150017b0ae6099621f\
409131aa317a811a412b211e22099083325ed13d80bda11f11c5402821a551a1c178e05be0c6c0eaa157a0f5108940b970de5096b1c92037515ff215e24c41b120f60171301e1004\
60040224a21c51bee121a17e2195016f90cd52658188202561227 10df12661e4303070ce9147d20ea229309441f8c1a56067c080f082f1197124b17780c1720d111f406cc23ee196\
f18b521211efb0a3d0496124a1c66040914d11d45095d010c019a241721d51467150d0209254d01d225e618f60b3113ab09910ce904b707d40d4d18a81a851e920a940e53073911f\
502c324dd147e23e401ac0b0b25d6179311e61a2c015720e91c56239215fe0967183500481c201bf1168b0de0099f1cd70b1f0ebb25001a68254303f519a91ec900d521a70ce10f2\
70d04146724ff1d050237161c14d5083d0de206890fc5180011e101cb196f0d9a238b022e2619244921c30c6c13a306020d340da4193e0352068f1c2e25bc1ad3160a08250cb6025\
706311ac10c170ce30a03013910df238307581216098301512114235215680616 00c417df24aa21800c5d202c0d361fea024a06361af103ca1eca22812181181a239712181e25147\
918741a4a24a10e2823431334258c1f49179317c7169c14a213f3258705290d9520c200f4136e00e124ea13f312690e3013dc0e531491007a1500239e048e1d1f17c7142301b4001\
211e8199018470bc305c91eac1ca31fcf181d041a00e91beb221001281a45044921212277085501361af00d851de405b6225f045920421b5a009b196823e5185c18d300cd12a6080\
301861f2010441c5f148c17f2189915230cfc09511e200c8a04da209a12e90c0619e119c119be06de230e0fe11f8117fa22130ff00be6137d1a2c1c891aee00fc0df80a0911c21f9\
61dc91d02242219b7060d2012197f138a0c7523d40d9e17e8058017011b980af713b006a604af05c91e190b28127c1e4d01e1176a26212214042715bc07d00cd506331d8b15b2136\
709de0e3d051116941ce006b514f60ea3198503371ed914021d231f6a086c21b7216217e211bc0a3116a01c5816ca0e291a2224da06b720270edc1f7c108d14ef1a5d1ba30bb522b\
```

```
a06ba1ac6144c006a0cfd22ae234103a0003e14a00bc418d921ff05ab1bd521e104b40ac0000323ff009920a5038e02de0b3f2607004a13c706200ebe1f561e231b1822520f1f006\
b1129156c140e0e35100806a8088307160319 1c021f170177172805cb0e65168f0d5e1cfc151e22e2132103d11eb2234b20bb1c1b0c91078d0b56001f1417157f144a19e1061c074\
41f5e020a01b205780ab024f2065a16c514830bed1bd0179b0b830339091a1d45091615e003911a12046108892037 0fc31c9312a1111615b0193713c80d9415d200241b771c3f173\
51f3d08740d4c03ce17fa189c23cd140c11220249059a13f11d86178c0539168424f722691c8f253711ad1b0b12b20971110f210c14b6096525bf02f5198f06281b2b102c1a5b11f\
10ed1235c14e10d6a092e1b4e12c411a517a404501f701c931128100212b4067e2115263a155e"

Y = "105e21f514801a2b20b413b521320ef018c218aa06681d0f1b87091b114b0fc315871d290ae417a602aa18fc0b7d0a6e1f850dcd02f7059a0ab125600d3e15e91ee115ca0dac1a6\
2127e04c316541bed040a1000115e00ec1d3117b614a912c124981bf4177423c70f160a7d028f159f09e4002b1d6425000c002091197418480cf30e4716101b35095713ca0fdf0a2\
5027a009b1e9920c915f61ea1198524b21af41b8b1bbb1534064417051b40204e1f4604e8225e127202db1a4405351cc0142012321f9707010e3b1067152a24b701fc1ed8160b1e1\
e13fe1fba257908a1113218960c11232f12621f9e157a0b081f9c00501abb1251252707b51de6129f06080f8f03ba20a806a50d65162523cb1c771e90022218d62391135b2429174\
804ef16c7247c05c715540d6f0008211001c720f222ca10e61ce902b2129a0b10252e163d1a4423640fda1ca219840b011c621ba51e7517a90e9f09e61c43202e10af113906d70a8\
0040d04ed04ce201014f616d114b51c9710d504482080240f208f0edf018407aa16d2060212aa25491f7821390f9c1cb106d11eac152009400fcd22c4072e098616cf1ef60b6d171\
c23cf012b1af622b00a1f0dd716c608a01ccd01e3258720ff1e3021330d1902a2047f253915670574166a19900740176d00e4187f03d90d491e1f0cf51f120ace1ec50ed1237e090\
21be118d91a3f0fba05150aae171d20fc0af51adf09e021841ae41a9d013a20fe231604f409f40b66202d16a2182112e21e8702d1011e1348115411ba242d0e6d0464260a078b122\
214380a5e07410b6910851375239903fa1bbf032b127b14c4096e0d3e1f11149f0ce41ffe067d04df0e1020d90f471ab91e3e168c057905be1f661dd314e01ec6195001d51a3d171\
2118c14431cbd19cc254c21f00b352018127110642239093924831 2a60f420266239416651efb0940025021281ef1104213651d42002a1e3411d7175715951d720d971990121e0f6\
f1ea6074319d60c091390188a06c100950736058f01bc13be20b1012e248e06391f29061d10282216251422ea20080a9013fb1dde0d6b19e904e00ae61c7d175e20200499194513a\
51bba2067217409610b6b1f201f8d1a9807cf0231141a056e038704df2483214201ad0cab05c213b61ed40627187f0f2322430d910080249f1eb4160c0c2c1e5e158f0cd8242c11d\
60086009521dd127821dd095a253609d318611d781a6d02f80db10421134f14a30d99265022eb128b0307045c148d1351058e01a42202228d0af90b6c0e22130508d822ba17050b5\
a229908fe144a08c20cfe23361b2207b00b8c020c23db019123f704311ebc07070f23256b19e908ac18811d2f02481ca71ec1226c261c2286000106eb0a721706209712b605d8236\
30ef01b6b03f01c0500d8193a19840037170025380e8a1ea2141f118812b41ca0205206ea2128005908a811d9010d1bc4019401b91c9e179100c024c50d0125fa0d501605208f0ac\
c0cca1889089d1e2d00340a8500ed17f90c8911f9089305c00f161f72072f027e03f30f5103ac03cb170f00c60bcb22d412d3073105ef1c1e1fe6032e116d216410e203d306390c1\
51b5013a6229a1396087409091c31135f17a31bea24a71c7600ce0ee808d51c5e1be60d5723a813f70f840527114c0a36050624e222a00cd21e5b17bf05f8063c1e60063e06e412f\
8251f12ca1e30162622fa1d590db516b41e9801631eb71b5c148a2432230e0b2f169f20e41de715ae1bb72095237713a3102e22210b6714ea23bf19cb18210fef153f114c06bb1c8\
20bf41ef017bb22b206b920cd20b31cd11ec40fe41730114b03551f4e06dc132910d6019f1d571129143a11c809b61eee1a0011571fea163d0382186501690a0b0e070d1e25c512e\
d0db405511a0305a619d025a404cf1293215c16ae184101471eee0de1004f22d614a424db1ca304b119591ad81d1320e8131b1619108009c812560e8e020811c415ab0abb020f262\
61aa01ab10597053b068c1fa2105e2145157601dc21cc0d1f00da12c51ab519401f651af424ef"
```


Challenge: With probability greater than 1/2, find which of the ciphertexts (V, W) and (X, Y) is the encryption of $K0$.

References

1. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. *Post-quantum key exchange - A new hope.* In T. Holz and S. Savage, editors, Proceedings of the 25th USENIX Security Symposium, pages 327-343. USENIX Association, 2016. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim.
2. Albrecht M., Bai S., Ducas L. *A Subfield Lattice Attack on Overstretched NTRU Assumptions.* In: Robshaw M., Katz J. (eds) Advances in Cryptology - CRYPTO 2016. Lecture Notes in Computer Science, vol 9814. Springer, Berlin, Heidelberg, pp 153-178.
3. M. R. Albrecht, R. Player, and S. Scott. *On the concrete hardness of learning with errors.* J. Mathematical Cryptology, 9(3):169-203, 2015. URL: http://www.degruyter.com/view/j/jmc.2015. 9.issue-3/jmc-2015-0016/jmc-2015-0016.xml.
4. Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. *Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator.* In Marc Fischlin and Jean-Sébastien Coron, editors, Advances in Cryptology – EUROCRYPT 2016, volume 9665 of LNCS, pages 789-819. Springer, 2016. https://eprint.iacr.org/2016/146. 20
5. Aharonov, D., Regev, O.: *A lattice problem in quantum NP.* In: FOCS, pp. 210–219 (2003).
6. Ajtai, M.: *The shortest vector problem in L_2 is NP-hard for randomized reductions.* In: STOC, pp. 10-19 (1998).
7. Ambainis, A.: *Quantum walk algorithm for element distinctness.* In: FOCS, pp. 22-31 (2003).
8. W. D. Banks, I. E. Shparlinski, *A Variant of NTRU with Non-invertible Polynomials*, INDOCRYPT 2002, Hyderabad, India, LNCS vol. 2551, pp. 62-70, Springer, 2002.

9. Shi Bai, Damien Stehlé and Weiqiang Wen. *Improved Reduction from the Bounded Distance Decoding Problem to the Unique Shortest Vector Problem in Lattices.* In Springer Proc. of ICALP'2016, pp. 76:1-76:12.
10. A. Becker, L. Ducas, N. Gama, and T. Laarhoven. *New directions in nearest neighbor searching with applications to lattice sieving.* Robert Krauthgamer, editor. Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016. SIAM, 2016, pages 10-24. https://eprint.iacr.org/2015/1128.
11. Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. *NTRU Prime.* In Jan Camenisch and Carlisle Adams, editors, Selected Areas in Cryptography - SAC 2017, LNCS, to appear. Springer, 2017. http://ntruprime.cr.yp.to/papers.html
12. Hao Chen, Kristin Lauter, and Katherine E. Stange. *Vulnerable Galois RLWE families and improved attacks.* IACR Cryptology ePrint Archive, 2016. *https://eprint.iacr.org/2016/193.*
13. Yuanmi Chen and Phong Q. Nguyen. *BKZ 2.0: Better lattice security estimates.* In Dong Hoon Lee and Xiaoyun Wang, editors, Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings, volume 7073 of LNCS, pp. 1-20. Springer.

    T 97, volume 1233 of Lecture Notes in Comput. Sci., pp 52-61. Springer, Berlin, 1997.
14. Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. *Recovering Short Generators of Principal Ideals in Cyclotomic Rings* Marc Fischlin and Jean-Sébastien Coron (Eds.). In Advances in Cryptology Eurocrypt May 2016, Lecture Notes in Computer Science, Springer-Verlag,Proceedings, Part II, pp. pp 559-585.
15. D. Dadush, O. Regev, and N. Stephens-Davidowitz. *On the closest vector problem with a distance guarantee.* In Proc. of CCC, pages 98-109. IEEE Computer Society Press, 2014.
16. Alexander W. Dent. *A designer's guide to KEMs.* In Kenneth G. Paterson, editor, Cryptography and Coding, 9th IMA International Conference, Cirencester, UK, December 16-18, 2003, Proceedings, vol. 2898 of Lecture Notes in Computer Science, pp. 133-151. Springer.
17. E. Fujisaki and T. Okamoto. *Secure integration of asymmetric and symmetric encryption schemes.* In Advances in Cryptology - CRYPTO '99, pages 537-554, 1999. Available at https://link.springer.com/chapter/10.1007/3-540-48405-1 34.
18. Nicolas Gama, Malika Izabachène, Phong Q. Nguyen, and Xiang Xie *Structural Lattice Reduction: Generalized Worst-Case to Average-Case Reductions and Homomorphic Cryptosystems.* Marc Fischlin and Jean-Sébastien Coron (Eds.), In Advances in cryptology Eurocrypt May 2016, Lecture Notes in Computer Science, Springer-Verlag Proceedings, Part II, pp. 528-558.
19. Grover, L. K.: *A fast quantum mechanical algorithm for database search.* In: STOC, pp. 212–219 (1996)39.
20. Grover, L. K., Rudolph, T.: *How significant are the known collision and element distinctness quantum algorithms?* Quantum Info. Comput.4 (3), pp. 201-206 (2004).
21. Jung Hee Cheon, Jinhyuck Jeong, Changmin Lee *An Algorithm for NTRU Problems and Cryptanalysis of the GGH Multilinear Map without a Low Level Encoding of Zero.* IACR Cryptology ePrint Archive, *https://eprint.iacr.org/2016/139.pdf.*

22. Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. *Terminating BKZ.* IACR Cryptology ePrint Archive report 2011/198, 2011. https://eprint.iacr.org/2011/198.
23. J. Hoffstein, J. Pipher, and J. H. Silverman. *NTRU: A Ring Based Public Key Cryptosystem in Algorithmic Number Theory,* Lecture Notes in Computer Science 1423, Springer-Verlag, pp. 267-288, 1998.
24. Nick Howgrave-Graham. *A hybrid lattice-reduction and meet-in-the-middle attack against NTRU.* In Alfred Menezes, editor, Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings, volume 4622 of Lecture Notes in Computer Science, pp. 150-169. Springer.
25. Nick Howgrave-Graham, Joseph H. Silverman, and William Whyte. *A meet-in-the middle attack on an NTRU private key.* Technical report, Technical report, NTRU Cryptosystems, June 2003. Report, 2003. *https://www.securityinnovation.com/uploads/Crypto/NTRUTech004v2.pdf.*
26. Thijs Laarhoven. *Sieving for shortest vectors in lattices using angular locality-sensitive hashing.* In Rosario Gennaro and Matthew Robshaw, editors, Advances in Cryptology CRYPTO 2015 -35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I, volume 9215 of Lecture Notes in Computer Science, pages 3-22. Springer, 2015. https://eprint.iacr.org/2014/744.pdf.
27. Thijs Laarhoven, Michele Mosca, and Joop van de Pol.*Finding shortest lattice vectors faster using quantum search.* Des. Codes Cryptography, 77(2-3):375-400, 2015.
28. R. Lindner and C. Peikert. *Better key sizes (and attacks) for LWE-based encryption.* In A. Kiayias, editor, Topics in Cryptology - CT-RSA 2011, vol. 6558 of LNCS, pp. 319-339. Springer, Heidelberg, Feb. 2011.
29. M. Liu, X. Wang, G. Xu, and X. Zheng. *A note on BDD problems with $\lambda 2$-gap.* Inf. Process. Lett., 114(1-2):9-12, January 2014.
30. Y. K. Liu, V. Lyubashevsky, and D. Micciancio. *On bounded distance decoding for general lattices.* In Proc. of RANDOM, volume 4110 of LNCS, pages 450-461. Springer, 2006.
31. Adriana López-Alt, Eran Tromer and Vinod Vaikuntanathan *On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption.* STOC 2012 Proceedings of the forty-fourth annual ACM symposium on Theory of computing. pp. 1219-1234.
32. V. Lyubashevsky, C. Peikert, and O. Regev. *On ideal lattices and learning with errors over rings.* In EUROCRYPT 2010, pages 1-23. Springer, 2010.
33. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. *A toolkit for ring-LWE cryptography.* In EUROCRYPT 2013, pp. 35-54.
34. V. Lyubashevsky and D. Micciancio. *On bounded distance decoding, unique shortest vectors, and the minimum distance problem.* In Proc. of CRYPTO 2009, pp. 577-594.
35. Micciancio, D., Voulgaris, P.: *Faster exponential time algorithms for the shortest vector problem.* In SODA(2010), pp. 1468-1480.
36. M. Naehrig, E. Alkim, J. W. Bos, L. Ducas, K. Easterbrook, B. LaMacchia, P. Longa, I. Mironov, V. Nikolaenko, C. Peikert, A. Raghunathan and D. Stebila. *FrodoKEM Practical quantum-secure key encapsulation from generic lattices.* Available at https://frodokem.org/. November 2017.

37. M. Naor and M. Yung. Public Key Cryptosystems *Provably Secure against Chosen Ciphertext Attacks*. In Proc. of the 22nd ACM STOC, pages 427-437. ACM Press, New York, 1990.
38. NIST Post-Quantum Cryptography- Call for Proposals. Available at https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization/Call-for-Proposals. List of First Round candidates available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions
39. J. Hoffstein, J. Pipher, and J. H. Silverman. *NTRU : A Ring Based Public Key Cryptosystem in Algorithmic Number Theory*. Lecture Notes in Computer Science 1423, Springer-Verlag, pages 267-288, 1998.
40. Xavier Pujol and Damien Stehlé. *Solving the shortest lattice vector problem in time* $2^{2,465.n}$. IACR Cryptology ePrint Archive, 2009. *https://eprint.iacr.org/2009/605*.
41. C. Peikert. *A useful fact about Ring-LWE that should be known better: it is *at least as hard* to break as NTRU, and likely strictly harder*. Available at http://archive.is/B9KEW.
42. C. Peikert. *Public-key cryptosystems from the worst-case shortest vector problem*. In STOC 2009, pp. 333-342. ACM.
43. Regev, O. *On lattices, learning with errors, random linear codes, and cryptography*. In: STOC, pp. 84–93 (2005).
44. O. Regev. *On lattices, learning with errors, random linear codes, and cryptography*. J. ACM, 56(6), 2009.
45. Regev, O.:*Lattices in computer science*. Lecture notes for a course at the Tel Aviv University (2004)78.
46. Regev, O.:*Quantum computation and lattice problems*. SIAM J. Comput. 33 (3), pp. 738-760 (2004).
47. Miruna Roşca, A. Sakzad, D. Stehlé and R. Steinfeld. *Middle-Product Learning With Errors*. Cryptology ePrint archive. Available at https://eprint.iacr.org/2017/628.pdf. 2017.
48. Santha, M.: *Quantum walk based search algorithms*. In: TAMC (2008), pp. 31–46 .
49. Schneider, M.: *Analysis of Gauss-Sieve for solving the shortest vector problem in lattices*. In: WALCOM (2011), pp. 89-97.
50. Schneider, M.: *Sieving for short vectors in ideal lattices*. In: AFRICACRYPT (2013), pp. 375-391.
51. C. P. Schnorr and M. Euchner. *Lattice basis reduction: Improved practical algorithms and solving subset sum problems*. Mathematical Programming, 66(1):181-199, 1994
52. Shor, P.W.:*Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*. SIAM J. Comput. 26 (5), pp. 1484-1509 (1997).
53. D. Stehlé and R. Steinfeld. *Making NTRU as secure as worst-case problems over ideal lattices*. Draft of full extended version of Eurocrypt 2011 paper, ver. 10, Oct. 2011. Available at *http://web.science.mq.edu.au*.
54. D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. *Efficient public key encryption based on ideal lattices*. In ASIACRYPT 2009, pp. 617-635. Springer.
55. D. Stehlé and R. Steinfeld. *Making NTRU as secure as worst-case problems over ideal lattices*. In EUROCRYPT 2011, pp. 27-47. Springer.
56. R. Steinfeld, A. Sakzad and R. K. Zhao. *Titanium: Post-Quantum Public-key Encryption and KEM Algorithms*. Available at http://users.monash.edu.au/ rste/Titanium.html. November 2017.
57. A. Vardy. *Algorithmic complexity in coding theory and the minimum distance problem*. In Proc. of STOC, pp. 92-109. ACM, 1997.

58. Wang, X., Liu, M., Tian, C., Bi, J.: *Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem.* In: ASIACCS (2011), pp. 1-9.
59. Zhang, F., Pan, Y., Hu, G.: *A three-level sieve algorithm for the shortest vector problem.* In: SAC (2013), pp. 29-47.